

# UC Santa Barbara

## UC Santa Barbara Electronic Theses and Dissertations

### Title

Representation Learning on Brain Data

### Permalink

<https://escholarship.org/uc/item/4xp815bm>

### Author

Lin, Sikun

### Publication Date

2022

Peer reviewed|Thesis/dissertation

University of California  
Santa Barbara

# Representation Learning on Brain Data

A dissertation submitted in partial satisfaction  
of the requirements for the degree

Doctor of Philosophy  
in  
Computer Science

by

Sikun Lin

Committee in charge:

Professor Ambuj K. Singh, Chair  
Professor Thomas C. Sprague  
Professor Xifeng Yan

December 2022

The Dissertation of Sikun Lin is approved.

---

Professor Thomas C. Sprague

---

Professor Xifeng Yan

---

Professor Ambuj K. Singh, Committee Chair

November 2022

# Representation Learning on Brain Data

Copyright © 2022

by

Sikun Lin

To my mom and dad,  
you never cease to support me in exploring a bigger world

## Acknowledgements

My Ph.D. is an exciting journey, and it would not have been possible without the help, kindness, and support from many people. First and foremost, I would like to express my gratitude to my research advisor Prof. Ambuj K. Singh, for being the greatest mentor. I am incredibly fortunate to be in his lab and have the chance to explore intriguing research problems. He always trusts me, gives me the right amount of freedom, and supports the directions I am passionate about, raising critical questions and providing encouragement and invaluable feedback along the way. Brainstorming potential research problems with him is the highlight of my years here. Ambuj also taught me how to be a better researcher, writer, and mentor. I still vividly remember how he conveyed Dijkstra’s three pieces of research advice and introduced us *The Elements of Style*. I cannot thank Ambuj enough for all the things I learned from him.

I am extremely grateful to Prof. Thomas C. Sprague, who provided me with many useful ideas, constructive feedback, and insights from a neuroscience perspective, and introduced me to useful literature and datasets. I would also like to thank Prof. Xifeng Yan for being part of my committee, contributing to the improvement of this work, and providing mentorships and valuable comments at each of my doctoral milestones.

I also feel blessed to be at UC, Santa Barbara, where interdisciplinary collaborations are prevalent and highly encouraged. Many CPCN seminars held by Psychological & Brain Sciences provided me with fascinating views to understand human brains and behaviors, as well as new ways to think of my own research.

During my Ph.D. studies, I was fortunate to have collaborated with Hongyuan You, Ali Borji, Shuyun Tang, Wei Ye, and Scott Grafton—each of them is an excellent researcher. I was also honored to work with a group of talented undergraduate students on their research, who taught me how to be a better mentor, communicator, and project

manager: Priyanka, Sid, Shuyun, Connor, Marrienne, Joseph, Jennifer, Ezequiel, and other SEEDS, ERSP, and data science capstone students. I also had a fantastic group of labmates: Victor, Sourav, Arlei, Haraldur, Alex, Rachel, Omid, Furkan, Yuning, Zexi, Mert, Sean, Kha-Dinh, Saurabh, Ashley, Zhao, Richika, and Christos; their friendship puts more smiles and laughter into my life. Thanks to Tim, Benji, Karen, and Maritza for their support as UCSB CS staff. My appreciation also goes out to my family and friends for their encouragement and support throughout my studies, especially when uniting with some of them is not an option during the COVID-19 pandemic.

Lastly, I appreciate the unexpected global pandemic and would also like to thank a YouTube video themed on near-death-experience and afterlife legends. Like many others, I was experiencing an existential crisis for most of my twenties. Even Viktor Frankl's "Man's Search for Meaning" couldn't help me. The pandemic and the video woke me: I can just "to be" and "to experience". I started to take a more positive stance toward life and research, exploring new things that excite me—there are so many questions one can ask about the brain and AI, and I am so lucky to be one of the people to work on some of the answers!

# Curriculum Vitæ

## Sikun Lin

### Education

- 2022 Ph.D. in Computer Science (Expected), University of California, Santa Barbara.
- 2017 M.Phil. in Computer Science, Hong Kong University of Science and Technology.
- 2015 B.Sc. in Physics and Mathematics, with Humanity and Social Science minors, Hong Kong University of Science and Technology.

### Publications

#### *Research*

- **Sikun Lin\***, and Ali Borji\*. “SplitMixer: Fat Trimmed From MLP-like Models.” Submitted (2022).
- **Sikun Lin**, Thomas C. Sprague, Ambuj K. Singh. “Redundancy and dependency in brain activities.” In *4th Shared Visual Representations in Human Machine Intelligence* (NeurIPS 2022 SVRHM workshop).
- **Sikun Lin**, Thomas C. Sprague, Ambuj K. Singh. “Mind Reader: Reconstructing complex images from brain activities.” In *Proceedings of 36th Conference on Neural Information Processing Systems* (NeurIPS 2022).
- **Sikun Lin**, Shuyun Tang, Ambuj K. Singh. “Deep Representations for Time-varying Brain Datasets.” In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* p. 999–1009 (KDD 2022).
- Hongyuan You, **Sikun Lin**, Ambuj K. Singh. “Learning Interpretable Models for Couple Networks Under Domain Constraints.” In *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35, No. 12, pp. 10727-10736 (AAAI 2021).
- Wei Ye, **Sikun Lin**, Ambuj K. Singh. “Relational Neighbor Networks for Semi-supervised Classification.” In preparation (2020).
- Ali Borji, and **Sikun Lin**. “White Noise Analysis of Neural Networks.” In *International Conference on Learning Representations* (ICLR 2020).
- **Sikun Lin**, and Pan Hui. “Where’s your focus: Personalized attention.” *HKUST Masters Thesis* (2017).
- Wenxiao Zhang, **Sikun Lin**, Farshid Hassani Bijarbooneh, Hao-Fei Cheng, Tristan Braud, Pengyuan Zhou, Lik-Hang Lee, and Pan Hui. “Edgexar: A 6-dof camera multi-target interaction framework for mar with user-friendly latency compensation.” *Proceedings of the ACM on Human-Computer Interaction* 6, no. EICS (2022): 1-24.



- Wenxiao Zhang, **Sikun Lin**, Farshid Hassani Bijarbooneh, Hao Fei Cheng, and Pan Hui. “Cloudar: A cloud-based framework for mobile augmented reality.” *In Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, pp. 194-200. 2017.
- **Sikun Lin**, Hao Fei Cheng, Weikai Li, Zhanpeng Huang, Pan Hui, and Christoph Peylo. “Ubii: Physical world interaction through augmented reality.” *IEEE Transactions on Mobile Computing* 16, no. 3 (2016): 872-885.

#### *Education*

- Bernard Hart, Titipat Achakulvisut, Ayoade Adeyemi, **Sikun Lin** et al. “Neuro-match Academy: a 3-week, online summer school in computational neuroscience.” *Journal of Open Source Education* 5, no. 49 (2022).

## Abstract

Representation Learning on Brain Data

by

Sikun Lin

Artificial intelligence and machine learning (AI/ML) have been extremely successful in predicting, optimizing, and controlling the behavior of complex interacting systems. Robustness and explainability of existing AI/ML methods, however, remain big challenges, and clearly new approaches are needed. The human brain motivated the early development of deep learning, and neuroscientific concepts have contributed to the profound success of deep learning algorithms across many areas. The next leap in AI/ML may again come from a deeper understanding of brain architectures and processes—this dissertation focuses on deepening this understanding with machine learning models.

We first discuss a convex optimization framework to analyze and integrate multimodal brain data to infer brain subnetworks and understand heterogeneity across tasks. Next, we propose a novel deep learning model to learn representations of multimodal and dynamic brain signals. Although these models are mostly considered black-box, we can characterize the input brain signals with attribution methods, study brain organizational structures, and unveil the heterogeneity among brain regions, tasks, and individuals. We then demonstrate that semantic representation is an essential piece in the human visual system. With added text modality, we are able to reconstruct complex high-fidelity imagery from input brain signals and infer brain activities from visual stimuli. Further studies are then presented to explore the redundancy and dependency in these brain signals related to visual information processing. Lastly, we apply neuroscience tools and insights to deep learning models, gaining a deeper understanding of the latter and

developing more computation- and memory-efficient models. These works demonstrate that advances and challenges in neuroscience and AI/ML benefit each other and drive both fields forward.

## 0.1 Permissions and Attributions

The majority of the materials described in this dissertation have either been published by the author of the dissertation or are currently in the process of submission. The author has made principal contributions to all stages of the published works as described below.

1. The contents of chapter chapter 2 have been previously published as:

You, Hongyuan, Sikun Lin, Ambuj K. Singh. "Learning Interpretable Models for Coupled Networks Under Domain Constraints." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 10727-10736. 2021.

2. The contents of chapter chapter 3 have been previously published as:

Sikun Lin, Shuyun Tang, Ambuj K. Singh. "Deep Representations for Time-varying Brain Datasets." In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* p. 999–1009. 2022.

3. Portion of chapter chapter 4 is accepted and under publication at:

Sikun Lin, Thomas C. Sprague, Ambuj K. Singh. "Mind Reader: Reconstructing complex images from brain activities." In *Proceedings of 36th Conference on Neural Information Processing Systems*. 2022.

4. The contents of chapter chapter 5 is accepted and under publication at:

Sikun Lin, Thomas C. Sprague, Ambuj K. Singh. "Redundancy and dependency in brain activities." In *4th Shared Visual Representations in Human Machine Intelligence* (NeurIPS 2022 SVRHM workshop).

5. Portion of chapter chapter 6 have been previously published as:

Ali Borji and Sikun Lin. “White Noise Analysis of Neural Networks.” In *International Conference on Learning Representations*. 2020,

and another portion is under submission, but publicly available at Arxiv, as:

Sikun Lin and Ali Borji. “SplitMixer: Fat Trimmed From MLP-like Models.”

# Contents

<b>Curriculum Vitae</b>	<b>vii</b>
<b>Abstract</b>	<b>ix</b>
0.1 Permissions and Attributions . . . . .	x
<b>List of Figures</b>	<b>xv</b>
<b>List of Tables</b>	<b>xxviii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Modeling Coupled Networks: Structural Connectivity and Static Functional Connectivity</b>	<b>6</b>
2.1 Introduction . . . . .	7
2.2 Constrained Multiple-Output Regression Formulation . . . . .	9
2.2.1 Multiple-output Regression Problem . . . . .	9
2.2.2 Relaxing Gaussian Assumptions . . . . .	11
2.2.3 Imposing Domain Constraints . . . . .	13
2.3 Alternating Minimization Solution . . . . .	14
2.4 Experiments on Synthetic and HCP Datasets . . . . .	19
2.4.1 Application to Simulated Data . . . . .	20
2.4.2 Application to Human Connectome Data . . . . .	24
2.5 Conclusion . . . . .	31
<b>3 Modeling through Graph Neural Networks: Structural Connectivity and Dynamic fMRI</b>	<b>32</b>
3.1 Introduction . . . . .	33
3.2 Spatial-Temporal GNN for Learning Multi-Modality Brain Representation . . . . .	35
3.2.1 Preliminary . . . . .	35
3.2.2 Method . . . . .	36
3.2.3 Experiments . . . . .	42

3.3	Graph Attribution and Interpretations . . . . .	54
3.3.1	Attribution with IG (Integrated Gradients) . . . . .	54
3.3.2	Experiments . . . . .	55
3.4	Conclusion . . . . .	64
<b>4</b>	<b>Going Beyond Brain Modalities: Reconstructing Observed Complex Images from Brain Activities</b>	<b>65</b>
4.1	Incorporating Additional Text Modality . . . . .	66
4.1.1	CLIP Space as the Intermediary . . . . .	69
4.2	Brain Decoding . . . . .	71
4.2.1	Method . . . . .	71
4.2.2	Results . . . . .	78
4.3	Brain Encoding and Encoding-Decoding Cycle . . . . .	93
4.3.1	Brain Encoding . . . . .	93
4.3.2	Complete Cycle . . . . .	95
4.4	Discussions . . . . .	95
4.4.1	Other Decoding Attempt . . . . .	95
4.4.2	Limitations and Future Work . . . . .	96
4.4.3	Using Pre-trained Models . . . . .	99
4.5	Conclusion . . . . .	102
<b>5</b>	<b>Brain Activity Redundancies and Low-dimensional Representations</b>	<b>103</b>
5.1	Dataset and Models . . . . .	104
5.2	Findings . . . . .	106
5.2.1	Brain Signals Contain High-level Redundancy . . . . .	106
5.2.2	Autoencoders Effectively “Denoise” Brain Signals . . . . .	109
5.2.3	Brain Activity Resides in the Semantic Space—the Hopfieldian View	110
5.2.4	Masking and Attribution Reveal Voxel and Region Importance . .	111
5.3	Discussions and Conclusion . . . . .	122
<b>6</b>	<b>Neuroscience-inspired DNN modeling</b>	<b>124</b>
6.1	White Noise Analysis of Neural Networks . . . . .	125
6.1.1	Introduction . . . . .	125
6.1.2	Related Works and Concepts . . . . .	127
6.1.3	Applications to Deep Learning Models . . . . .	131
6.1.4	Discussions and Conclusion . . . . .	152
6.2	Sparsifying DNNs: Fat-Trimming MLP-like Models . . . . .	154
6.2.1	Introduction . . . . .	155
6.2.2	SplitMixer . . . . .	156
6.2.3	Experiments and Results . . . . .	163
6.2.4	Related Work . . . . .	173
6.2.5	Discussions and Conclusion . . . . .	175

<b>7 Conclusion and Contributions</b>	<b>178</b>
<b>A</b>	<b>182</b>
A.1 Tasks Descriptions of the CRASH dataset . . . . .	182
A.2 Experiment setting details for Chapter 4 . . . . .	183
<b>Bibliography</b>	<b>185</b>

# List of Figures

2.1	(a) ROC curve for $\Omega$ estimation, (b) ROC curve for $\mathbf{B}$ estimation. Benefiting from domain constraints, CC-MRCE obtains better ROC curves when uncovering nonzero entries of $\mathbf{B}$ and $\Omega$ . . . . .	22
2.2	Visualizations of the $\mathbf{B}$ matrix for seven tasks: (a) EMOTION, (b) LANGUAGE, (c) MOTOR, (d) GAMBLING, (e) SOCIAL, (f) RELATIONAL, (g) WM. For each task, each fold in the 10-fold cross-validation may lead to different models. Here, we only show those entries that are nonzero more than five times. . . . .	25
2.3	Visualization of the edges contributing to <i>all</i> seven tasks. Node size denotes the degree, and edge width denotes its importance, as in the mapping $\mathbf{B}$ . . . . .	25
2.4	Task-specific visualizations for high-contributing structural edges. Assuming that the maximum number of nonzero entries of a row in $\mathbf{B}$ is $m$ , we only show the edges corresponding to rows containing more than $m/2$ nonzero entries. . . . .	29
2.5	Entry-wise and edge-wise overlap ratio for the mapping of seven tasks. (a) considers entry-wise overlap of predicted $\mathbf{B}$ of different tasks. The value on position (task $i$ , task $j$ ) is the entry-wise IoU (Intersection over Union) of task $i$ 's $\mathbf{B}$ and task $j$ 's $\mathbf{B}$ , i.e. number of nonzero entries in $\mathbf{B}_i \cap \mathbf{B}_j$ over number of nonzero entries in $\mathbf{B}_i \cup \mathbf{B}_j$ . (b) considers the SC edges responsible for different tasks. An edge is considered to exist when its corresponding row in $\mathbf{B}$ has nonzero entries. The value in position (task $i$ , task $j$ ) is the number of common SC edges of task $i$ and task $j$ over the number of SC edges of task $j$ . . . . .	30



3.1	The proposed ReBraID model for integrating brain structure and dynamics (the architecture shown is for classification). For each batch with batch size $B$ , input $X$ has a dimension of $(B, 1, N, T)$ , and $A, A_{\text{adp}}$ both have the dimension $(B, N, N)$ . Note: axis order follows PyTorch conventions. The dimension at the second $X$ index is the expanded feature dimension. The encoder (green part) encodes temporal and spatial information alternately, producing a latent representation in $(B, d_{\text{latent}}, N, 1)$ . These embeddings are followed by linear layers for pooling and classification. The final output has a dimension of $(B, C)$ . . . . .	36
3.2	Functional connectivities (FCs) among $N$ brain regions, where each $\text{FC} \in \mathbb{R}^{N \times N}$ . The value at $\text{FC}_{ij}$ is calculated as the Pearson correlation coefficient between signals of brain region $i$ and region $j$ . The figure shows 6 FCs calculated from 6 consecutive sliding windows within the same fMRI session, with signal window length being 30 and sliding stride being 30. From the figure, we can clearly tell that FCs are highly dynamic. . . . .	37
3.3	Comparison of strided non-causal TCN (left) and dilated causal TCN (right). For a causal TCN, the causal aspect is achieved through padding $(\text{kernel\_size} - 1) \times \text{dilation}$ number of zeros to the layer’s input. The resulting $\mathbf{y}$ always has the same length as input $\mathbf{x}$ , in which $\mathbf{y}_\tau$ only depends on inputs $\mathbf{x}_{t \leq \tau}$ . We can view strided non-causal TCN as the rightmost node of a dilated causal TCN. . . . .	38
3.4	Inner cluster smoothing toy example. . . . .	42
3.5	Ablation studies on different input lengths. . . . .	44
3.6	Learned latent adaptive adjacency matrices. <b>(a)</b> $A_{i,\text{adp}}$ of 3 randomly sampled inputs during the DOT task. <b>(b)</b> $A_{i,\text{adp}}$ of 3 consecutive inputs from a same session during the DOT task. <b>(c)</b> column averages of task-averaged $A_{\text{adp}}$ for resting state, VWM, DYN, DOT, MOD, PVT. <b>(d)</b> left two: t-SNE of $X^{(\text{node-2}, 156)}\Theta_{\text{adp}}$ in six tasks of one subject; right two: t-SNE of $X^{(\text{node-155}, 156)}\Theta_{\text{adp}}$ during the resting state of two subjects (multiple sessions are aggregated). . . . .	46
3.7	Confusion matrices of: <b>(a)</b> ReBraID (our proposed model), <b>(b)</b> model with coarsened graph (setting (viii)), <b>(c)</b> Graph Transformer (best graph baseline). Tasks are 1-Rest, 2-VWM, 3-DYN, 4-DOT, 5-MOD, 6-PVT. Misclassification pairs clustered at the first three tasks (resting, VWM, DYN) and the latter three (DOT, MOD, PVT). Shown confusion matrices are from models trained on length-256 inputs. We note that these misclassification pairs may differ for models trained on other input lengths (like 128-frame, etc.). . . . .	47
3.8	Choosing the number of GNN to TCN layer ratio for different input lengths. In most cases, two TCN layers per GNN layer results in the best model performance in terms of F1. . . . .	50

3.9	(a) adding inner cluster smoothing or input-dependent adaptive adjacency matrix makes the model more stable across various learning rates (results shown are from length-16 inputs). (b) Validation loss v.s. training epochs. Input length is 256, and four smoothing modules are used. Legends are the soft-assignment cluster numbers of the four smoothing modules. Our other experiments use decreasing cluster numbers that halved at each module, corresponding to the 100-50-25-12 choice here. . . . .	52
3.10	(a) Temporal importance sanity check of IG results on two pieces of inputs with a large overlap period. Attribution maps are offset-aligned. (b) $\text{ATTR}_X$ distributions across 17 brain subnetworks (defined as in [1]) for VWM. . . . .	57
3.11	Column averages of task-averaged $\text{ATTR}_A$ (mapped into 34 subnetworks defined by the 17-network parcellation with left, right hemispheres). The top row is obtained from real SC-induced $A$ and the bottom row is obtained from random SC-induced $A_{\text{rand}}$ . Attributions are normalized to $[0, 1]$ . Tasks are: Rest, VWM, DYN, DOT, MOD, PVT from left to right.	59
3.12	ROI attributions from $\text{ATTR}_A$ and $\text{ATTR}_X$ . (Task order is the same as fig. 3.11). Edge color and width are based on task-averaged $\text{ATTR}_A \in \mathbb{R}^{200 \times 200}$ , and node color and size are based on task and temporal-averaged $\text{ATTR}_X \in \mathbb{R}^{200}$ . For visualization, only edges with highest attributions are shown (the resulting sparsity reduces to 0.009 from 0.196). . . . .	59
3.13	34 subnetworks' $\text{ATTR}_X$ distributions of 3 subjects performing the VWM task (left) and the MOD task (right). Outliers that go beyond $[Q1 - 1.5\text{IQR}, Q3 + 1.5\text{IQR}]$ are omitted. VWM has a much smaller average attribution variance than MOD. . . . .	61
3.14	(a) A typical adjacency matrix for simulated graph signals. (b) Task averaged $\text{Attr}_X$ of simulation (i). Attribution values are normalized. (c) Task averaged $A_{\text{adp}}$ of simulation (i) and its entry averages per column. (d) Task averaged $A_{\text{adp}}$ and task averaged $\text{Attr}_A$ of simulation (ii). Attribution values are normalized. . . . .	62
4.1	Category-wise AUC-ROC of multi-label classifiers that predicts from four different signal/embedding sources. The first 80 categories are “things categories” and the last 91 are “stuff categories” in COCO. . . . .	68
4.2	Sample-wise AUC-ROC of multi-label classifiers that predicts from five different signal/embedding sources as the number of samples in stimulus images increases. . . . .	70

4.3	The pipeline for reconstructing seen images from fMRI signals. <b>(a)</b> details different components, from collected data to the reconstructed image. The pipeline is trained in two stages: during the first stage, mapping models $f_{mi}, f_{mc}$ are trained to encode fMRI activities into the CLIP embedding space. In the second stage, conditional generator $\mathbf{G}$ and contrastive discriminator $\mathbf{D}$ are finetuned while both $f_{mi}, f_{mc}$ are kept frozen. <b>(b)</b> shows the image generation process once models are trained. . . . .	72
4.4	Image caption screening through CLIP encoders. For this sample, threshold is put at half of the largest probability: $0.5 \times 0.519$ . Therefore, captions (2) and (3) of the image are kept. . . . .	73
4.5	CLIP vector visualizations and thresholding. <b>(a)</b> before (left column) v.s. after (right column) thresholding at $\pm 1.5$ to remove outliers. There are <b>systematic differences between CLIP image embeddings</b> and text embeddings; the outliers typically occur at the same positions for each modality. <b>(b)</b> the caption screening process can make the kept caption embeddings more aligned. (b)1 and (b)2 are from the same sample, only difference is the screening process. <b>(c)</b> (thresholded) embeddings of the same image with different augmentations; embeddings of same image's different screened captions. All embeddings are shown the first 200 values for visualization purposes. . . . .	75
4.6	fMRI activities responding to two images, each repeating three times. The figure only shows the activities of the first 200 voxels for visualization purposes. . . . .	79
4.7	Embeddings mapped from fMRI signals overlay on ground truth CLIP embeddings. <b>(a)</b> shows the results of image embedding mapping model $f_{mi}$ ; <b>(b)</b> shows the results of caption embedding mapping model $f_{mc}$ . For visualization purposes, the figures only show the first 200 values of the length-512 vectors. . . . .	81
4.8	Mismatches are semantically close to the ground truth. Figure shows examples of incorrect matches $j$ (red frame) in a batch of 300 in the validation set. For each ground truth image $i$ (green frame), we pass it through CLIP encoder to get $\mathbf{h}^{(i)}$ and through $f_{mc}$ to get $\mathbf{h}^{(i)'}$ . The shown incorrect ones are those images with $\mathbf{h}^{(j)'}$ , $j \neq i$ that is closer to $\mathbf{h}^{(i)}$ than $\mathbf{h}^{(i)'}$ . . . . .	81
4.9	Images generated by our pipeline given input fMRI signals. . . . .	87
4.10	Comparisons between previous works and our pipeline. We are using the recent NSD dataset that involves more complex scenes. However, for comparison purposes, we choose four similar images from NSD, each containing a single object "plane", and show our reconstructions from fMRI signals in fig. 4.10b . . . . .	88
4.11	Images generated in microstimulation experiments. In <b>(a)(b)</b> , voxel activities at multiple task ROIs are increased before passed into the pipeline. In <b>(c)</b> , voxel activities at various visual processing stages are silenced. . .	89

4.12	fMRI-mapped embeddings in the CLIP space ( $\mathbf{h}'$ ). Each figure contains (i) an embedding mapped from a regular fMRI signal, (ii) an embedding mapped from the fMRI signals with voxel activities in earlier-visual ROIs (left)/ intermediate ROIs (middle) / higher-level ROIs (right) set to zero, (iii) an embedding mapped from the fMRI signal with voxels at random positions (same number of voxels as (ii)) set to zero. Setting activities of the earlier-visual cortex to zero lowers overall embedding vector values, while setting activities of higher-level ROIs has the opposite effect. We can also perform the reverse masking: only keep voxel activities at earlier-level visual/ intermediate / higher-level ROIs, then the effects are reversed. . . . .	89
4.13	Generated images conditioned on fMRI-mapped CLIP image embedding $\mathbf{h}'_{\text{img}}$ , fMRI-mapped CLIP text embedding $\mathbf{h}'_{\text{txt}}$ , or both. . . . .	91
4.14	More examples showcasing model successes and failures. For each two-row group, the top row shows the ground truth images, and the bottom row shows the reconstructions. . . . .	92
4.15	Brain encoding results. <b>(a)</b> ground truth and prediction of two samples. Only the first 1000 voxels are shown for visualization purposes. <b>(b)</b> Voxel-wise performance (in terms of the correlation coefficient between ground truth and prediction) v.s. voxel noise ceiling. <b>(c)</b> Prediction performance on a flatmap, redder regions have more accurate predictions (accounted for the noise ceiling). Note we only perform prediction on the nsdgeneral ROI, thus the boundary. . . . .	94
4.16	Encoding-decoding cycle. The top row shows image stimuli; the second row shows predicted fMRI activities (with corresponding ground truth) by the encoding pipeline (only 300 voxels are shown for visualization purposes); the third row shows reconstructed images from predicted fMRI signals. . . . .	95
4.17	Generated images from interpolation of two fMRI scans. Step number is set to 10. . . . .	97
4.18	Image generated by Lafite pre-trained on the CC3M dataset without fine-tuning on COCO or NSD. Ground truth stimuli (top row) and generated images conditioned on fMRI (bottom row). . . . .	99
4.19	Multi-label classifier (defined in section 4.1.1)'s average sample-wise AUC-ROC changes when masking input fMRI at different ratios. For a masked voxel, we set its value to 0. . . . .	100

5.1	<p>Redundancies observed in the fMRI betas in terms of <b>(a)</b> voxel activity reconstruction, and <b>(b)</b> stimuli category classification. For reconstruction with an autoencoder, we gradually increase the masking ratio of the inputs. Two masking schemes are tested: randomly choosing masked voxels and consecutively masking voxels. With random masking, the reconstruction performance stays around the same level up to masking 80%-90% of the voxels. The right plot of (a) shows the min, mean, and max reconstruction performance of the two masking schemes. For classification results (b), the left plot shows sample-wise AUC-ROC as we increase the masking ratio: the significant drop also occurs after 80%-90%; the right plot shows the category-wise AUC-ROC with and without masking voxels in the floc-faces and floc-bodies ROIs, which turned out to be extremely close to each other, not affecting <i>person</i> category (index 0)'s performance. . . . .</p>	107
5.2	<p>Accumulated explained variance v.s. the number of principal components (PCs) used. The trend of the reconstructed signal is close to that of the latent representation. . . . .</p>	108
5.3	<p>The left plot shows the reconstruction correlation coefficient (<i>cc</i>) together with the voxel-wise noise ceiling (<i>nc</i>) for 1000 voxels. <i>cc</i> is calculated over the validation set of 4035 samples and aligns well with <i>nc</i> (<i>cc</i> and <i>nc</i> have a 0.67 correlation with p-value 0). The right plot shows <i>cc - nc</i> values on a flatmap. Higher-order regions typically have larger values (redder), meaning the reconstruction is better for those regions. . . . .</p>	109
5.4	<p>Interpolating the latent embedding and generating reconstructions from the interpolations. The top row shows the generated signals for 300 voxels, and the bottom row shows the classification logits for 171 categories when passing the generated signal through the trained multi-label classifier. Values in both plots are normalized to 0-1. Interpolations are performed between three pairs of embeddings: (left) between two fMRIs corresponding to the same image; (middle) between two fMRIs corresponding to different images, but with exactly the same set of object categories; (right) between two fMRIs corresponding to two images having completely two different sets of object categories. . . . .</p>	110
5.5	<p>Latent representations of interpolated fMRI signals between two fMRI samples, pairs in three samples, and pairs in four samples. The interpolations occupy a much lower dimensional space: for 1000 latent representations of interpolations between two fMRI signals, only 2 PCs are needed to explain 0.998 variance, whereas 56 PCs are required to explain the same variance for 1000 unrelated fMRI signal embeddings. This indicates transitions between different fMRIs are cheap inside this learned latent space. For visualization, the first three PCs of the latent representations (<math>&gt; 0.999</math> variance explained for each interpolation pair with step number = 1000) are used as the coordinates. . . . .</p>	111

5.6	Masked fMRI embeddings. Masking different ROIs results in different distances between the masked and original signal embeddings. The relationship between these distances is consistent across samples (all pairs have a t-test p-value $< 1e-50$ except for the V2 > V1 pair, which has a p-value of 0.008): for example, masking floc-faces voxels always results in a closer embedding to the original signal to masking floc-bodies voxels. These distance relationships hold across subjects. . . . .	112
5.7	Reconstruction for different categories <b>(a)</b> From left to right: (1) categories exhibit groupings, (2) unmasked order is very different compared with masked ones, (3) the orders are consistent for categories in different groups for masked inputs, and (4) the orders can be inconsistent for categories within the same group (as in having a close performance under the 10%-masking ratio). <b>(b)</b> fMRI signals triggered by images containing <i>person</i> perform better (redder) at the right hemisphere's low to mid-level visual regions than those that do not contain <i>person</i> . In addition, when calculating the correlation between <i>cc</i> and <i>nc</i> (refer to fig. 5.3), person ones are larger than non-person ones (with an average of 0.673 v.s. 0.638 for subject 1). . . . .	113
5.8	Average voxel-wise reconstruction correlation coefficient ( <i>cc</i> ) for fMRI samples corresponding to different categories. The performance is measured under a 50%-masking ratio. Plots are for subjects 1, 2, and 5 from left to right. Error bars stand for the standard deviations of the average <i>cc</i> across all samples of that category. We also plot the sample occurrence (in blue lines) for individual categories as some categories, like <i>person</i> , have significantly more samples than other categories. Colors are based on super-categories, as indicated in the legend. . . . .	115
5.9	Performance decrease rate for different categories, together with the category sample occurrence. The rate is calculated as (performance with 10%-masked inputs - performance with 90%-masked inputs) / performance with 10%-masked inputs. . . . .	116
5.10	<b>(a)</b> Unmasked voxel and <b>(b)</b> Masked voxel reconstruction performance when masking V1-V4 on either the left hemisphere ( <i>lh</i> ) or the right hemisphere ( <i>rh</i> ). Subjects 1, 2, and 5 are used for the task. Masking the visual cortex at different levels on either hemisphere does not affect unmasked voxels differently (true for all subjects). But the reconstructions of masked V3/V4 voxels on <i>lh</i> consistently have a worse reconstruction than those on <i>rh</i> (t-test p-value $< 1e-8$ ). . . . .	117

5.11	Pairwise reconstruction between ROIs of subject 1. For each matrix, the rows are source ROIs that provide input activities, and the columns are target ROIs whose reconstruction performance is evaluated (in terms of the voxel-wise correlation coefficient). The plots are: <b>(a)</b> reconstruction mean, <b>(b)</b> reconstruction standard deviation, <b>(c)</b> voxel overlap percentage normalized by the total number of target ROI voxels, and <b>(d)</b> reconstruction mean when the overlapped voxels' activities are not provided in the inputs. . . . .	118
5.12	The top two rows of the flatmaps are reconstruction performance differences when masking two different visual cortex ROIs. The bottom two rows are visualized localizers of corresponding ROIs. Both sets have <i>rh</i> on top of <i>lh</i> . The region name stands for the masked region: for example, the first column "V1-V2" means "subtracting the voxel-wise reconstruction with V2-masked inputs from the voxel-wise reconstruction with V1-masked inputs". Discrepancies are observed between the performance difference (top set) and corresponding localizers (bottom set), from which we can identify region dependencies. For example, V2 depends more on V1 than V3, and it also shows additional dependencies with the posterior intraparietal sulcus (IPS) area on the right hemisphere (top tip of the rh flatmap). . . . .	119

- 5.13 Examining the voxel importance of subject 1 with (a-c) reconstruction models and (d) the classification model. For the reconstruction, Voxel with index 1364 is selected as the target voxel at which we measure the recovery performance; it locates on *lh* and belongs to both *floc-faces* and *floc-bodies*. **(a)** The recovery performance at each voxel if they are served as the “additional” input apart from V1-V4 voxel activities. The right plot is the zoom-in view of the left plot, showing the strong local dependencies, which are consistent across models and signal types (the values in (a) are normalized to 0-1 since AE and VQ-VAE *cc* are at different scales). **(b)** Given AE’s result, we plot (left) the top 200 contributing voxels, (middle) voxels that lead to a reconstruction performance larger than the mean value, and (right) the overall p-values of the reconstruction *cc*. We can observe that voxels on mirrored positions of *rh* are also contributing to the target voxel’s reconstruction, but overall *lh* voxels are more important for this target voxel on *lh*. p-Values are also aligned well: positions with better performance (larger *cc*) also have smaller p-values. **(c)** (left) The p-value changes with decreasing recovery *cc*. There is a sharp increase near the end, indicating those voxels are more irrelevant; (right) The in-ROI ratio for the top-100 contributing voxels: not all are from the ROIs that the target voxel belongs to (*floc-faces/bodies* in this case). **(d)** SHAP input attribution (absolute values) aggregated across categories. Redder (higher attribution) means the voxel is more crucial in determining if a specific category exists in the stimulus. . . . . 121
- 6.1 Illustration of the classification images concept. **(a)** Two sample digits as well as their linear combination with different magnitudes of white noise (eq. (6.3)). **(b)** Average correct and incorrect prediction maps of a binary CNN trained to separate digits 1 and 7. The fifth column shows the difference between the average of stimuli predicted as 1 and the average of stimuli predicted as 7. The column marked with “\*” is similar to the fifth column but computation is done only over noise patterns (and not the augmented stimuli), hence “classification images” (i.e.,  $(\bar{\mathbf{n}}^{11} + \bar{\mathbf{n}}^{71}) - (\bar{\mathbf{n}}^{17} + \bar{\mathbf{n}}^{77})$ ; eq. (6.1)). These templates can be used to classify a digit as 1 or 7. Yellow (blue) color corresponds to regions with positive (negative) correlation with the response as 1. **(c)** Same as B but using a 5 vs. 6 CNN.127
- 6.2 **(a)** Classification images of a CNN trained on MNIST (with 99.2% test accuracy). Image titles show ground truth, predicted class for the bias map, and the frequency of the noise patterns classified as that digit. **(b)** Classification images of logistic regression over MNIST with 92.46% test accuracy. **(c)** Confusion matrices of four classifiers (CNN and log. reg. biases, mean digit image, and log. reg. weights). The classification was done via template matching using the dot product. . . . . 132



6.3	Classification images for a two-layer MLP (784 $\rightarrow$ 1000 $\rightarrow$ 10) shown at the top and an RNN classifier at the bottom. None of the noise patterns were classified as 1 using both classifiers. While the derived biases do not resemble digits, they still convey information to predict the class of a test digit. . . . .	133
6.4	<b>(a)</b> Mean training images (top) and mean white noise pattern/bias maps (bottom) across CIFAR-10 classes. Image titles show the ground truth class and prediction of the bias map, respectively. <b>(b)</b> Confusion matrices using mean images (top) and bias maps (bottom) as classifiers, respectively. Notice that for some classes, it is easier to guess the class label from the mean image (e.g., frog). . . . .	134
6.5	Progressive build-up of the bias maps for 0, 1, and 2. . . . .	135
6.6	Using an AutoEncoder and a VAE to generate samples containing faint structures to be used for computing the classification images over MNIST dataset, using a CNN classifier. Both generators were trained only for two epochs to prohibit the CNN from generating perfect samples (shown at the top). The bottom panels show classification images derived using 100, 1K, and 10K samples from each generator. Note that classification images converge much faster now compared with the white noise stimuli. . . . .	137
6.7	Classification images, some sample generated images, confusion matrices of bias map classifiers, as well as one sample image and its reconstruction using Gabor wavelets over MNIST (left), Fashion-MNIST (middle), and CIFAR-10 (right) datasets. We used 960, 960, and 1520 Gabor wavelets over MNIST, Fashion-MNIST, and CIFAR-10, respectively. The corresponding numbers of PCA components are 250, 250, and 600 (per color channel). . . . .	138
6.8	<b>(a)</b> Adding bias to a digit changes it to the target class in many cases (here with $\gamma = 0.8$ ). Adding bias to noise (2nd col.) turns noise into the target digit in almost all cases. The histograms show the distribution of predicted classes (intact digits or pure noise; 1st row). Note that most of the noise images are classified as 8 (top histogram in 2nd col). <b>(b)</b> Same as A but using mean digit (computed over the training set). Adding the mean image is more effective but causes a much more perceptible perturbation. <b>(c)</b> The degree to which (i.e., accuracy) a stimulus is classified as the target class (i.e., fooled) by adding different magnitudes of bias (or mean image) to it. Converting noise to a target is easier than converting a signal. There is a trade-off between perceptual perturbation and accuracy (i.e., subtle bias leads to less number of digits being misclassified). . . . .	140
6.9	Illustration of influencing the CNN decisions (on MNIST) towards a particular digit class by adding bias to the digits (top) and adding bias to the noise (bottom). This is akin to a targeted attack. See fig. 6.10. . . . .	142

6.10	<p><b>(a)</b> Top: A 10-way CNN trained on MNIST (with half of the zeros augmented with a patch and relabeled as 1) performs very well on a clean test set (top confusion matrix). On a test set containing all zeros contaminated, it (incorrectly) classifies them as one. Classification images (right side) successfully reveal the perturbed region. Bottom: Same as above but over 8 and 9 digits. <b>(b)</b> Classification images reveal the adversarial patch attack over CIFAR-10. Here, half of the birds are contaminated with a patch and are labeled as cat. <b>(c)</b> Turning a frog into a car by adding the activation of the <i>conv6</i> layer, computed using white noise, of the car category to the frog. <b>(d)</b> Average gradients before the adversarial patch attack (top) and after the attack (middle). The small yellow region on the top-right of digit 8 means that increasing those pixels increases the loss and thus leads to misclassification (i.e., turns 8 to another digit). (bottom) Average gradient with all 8s contaminated and relabeled as 9. The blue region on the top-right of digit 9 means that increasing those pixels lowers the loss and thus leads to classifying a digit as 9. This analysis is performed over the MNIST training set. Please see also figs. 6.11 and 6.15.</p>	143
6.11	<p>Confusion matrices for adversarial patch attack on CIFAR-10 dataset (bird to cat). Class names: plane, car, bird, cat, deer, dog, frog, horse, ship, and truck. . . . .</p>	143
6.12	<p>Left two: example filters derived using spike-triggered averaging (STA) for the first two <i>conv</i> layers of a CNN trained on MNIST dataset (left; RF sizes are <math>5 \times 5</math> and <math>14 \times 14</math>) and 4 layers of a CNN on CIFAR-10 dataset (middle; RF sizes in order are <math>3 \times 3</math>, <math>5 \times 5</math>, <math>14 \times 14</math> and <math>32 \times 32</math>). Right: Trained model weights (i.e., convolutional kernels) of the first layer of a CNN trained on MNIST or CIFAR-10. These are not calculated by feeding noise patterns. They are derived after training the model on data. Interestingly, they are the same as those derived using white noise.</p>	144
6.13	<p>Trained model weights (i.e., convolutional kernels) of the first layer of a CNN trained on MNIST or CIFAR-10. These are not calculated by feeding noise patterns. They are derived after training the model on data. Interestingly, they are the same as those derived using white noise as shown in fig. 6.12. . . . .</p>	145
6.14	<p>(Top) Average layer activation using noise (left) and real data (right) over a CNN trained on CIFAR-10 dataset. (Bottom) Mean distance between average layer activations of different classes across model layers. . . . .</p>	146

6.15	Effect of adding activation at <i>conv6</i> , <i>conv4</i> , <i>conv2</i> and input of noises classified as different classes to real images. The figure shows CIFAR-10 model misclassification ratio vs. $\gamma$ , where the input to the model is $((1 - \gamma) \times \text{noise activation of a certain class} + \gamma \times \text{real data input image})$ . The misclassification ratio is calculated as the number of images that do not belong to the activation-added class but are classified as it over the number of images not belonging to the activation-added class. The visualization of adding activation to input is shown in fig. 6.10c. . . . .	148
6.16	Psychometric curves of a CNN trained on MNIST. The x-axis shows the magnitude of the signal added to the noise ( <b>(d)</b> ). The y-axis shows the accuracy. Legends show the magnitude of stimulation ( $k$ in Eq. 6). Larger $k$ (redder curve) means more bias. <b>(a)</b> Increasing <i>fc</i> bias enhances recognition towards the target digit for all digits. The opposite happens when lowering the bias. <b>(b)(c)</b> Stimulating neurons in <i>conv</i> layers helps some digits (for which those neurons are positively correlated) but hinders some others. . . . .	149
6.17	Results of microstimulation for binary decision-making tasks using a CNN classifier (1 vs. 3) and (2 vs. 8). Left(right) panels show increasing (decreasing) bias for each layer. See fig. 6.16. . . . .	151
6.18	Basic architecture of SplitMixer. The input image is evenly divided into several image patches which are tokenized with linear projections. A number of 1D depthwise convolutions (spatial mixing) and pointwise convolutions (channel mixing) are repeatedly applied to the projections. For channel mixing, we split the channels into segments (hence the name SplitMixer) and perform convolution on them. We implement this part with our ad-hoc solutions or 3D convolution. Finally, a global average pooling layer followed by a fully-connected layer is used for class prediction. . . .	158
6.19	Channel mixing approaches: <b>(a)</b> channels are split into two overlapping segments, and only one segment is convolved in each block (no parameter sharing across segments), <b>(b)</b> channels are equally split into a number of segments, and only one segment is convolved in each block (no overlap or parameter sharing), <b>(c)</b> all segments are convolved in each block and parameters are shared across segments, and <b>(d)</b> all segments are convolved in each block (no parameter sharing). . . . .	159
6.20	Parameter saving as a function of (left) segment overlap for SplitMixer-I and (right) segment for SplitMixer-II and SplitMixer-IV. About 75% of parameters can be saved in the limit for SplitMixer-I (i.e., as $i$ approaches infinity, see eq. (6.11)). . . . .	161
6.21	Potential savings in parameters and FLOPS for different SplitMixer variants.	163

6.22	Accuracy <i>vs.</i> parameters for different variants of SplitMixer. S stands for 1D spatial convolution and C stands for $1 \times 1$ pointwise convolution over channel segments. We plug in our components into ConvMixer, denoted here as “2D + C” (2D convolution kernels plus our channel mixing approach) and “1D S + ConvMixer C” (our 1D kernels plus channel mixing as is done in ConvMixer, i.e., $1 \times 1$ convolution across all channels without splitting). Data points are for different values of split ratio or number of segments depending on the model type. We have collected more data points on CIFAR-10 than other datasets. See also fig. 6.23 for accuracy <i>vs.</i> FLOPS plots. . . . .	165
6.23	Accuracy <i>vs.</i> FLOPS for different variants of SplitMixer. S stands for 1D spatial convolution and C stands for $1 \times 1$ pointwise convolution over channel segments. We plug in our components into ConvMixer, denoted here as “2D + C” (2D convolution kernels plus our channel mixing approach) and “1D S + ConvMixer C” (our 1D kernels plus channel mixing as is done in ConvMixer, i.e., $1 \times 1$ convolution across all channels without splitting). Data points are for different values of split ratio or number of segments depending on the model type. We have collected more data points on CIFAR-10 than other datasets. Notice that the ratio of FLOPS over the number of parameters is almost the same for all models except SplitMixer-III, where this ratio is higher since all segments are updated in each block and parameters are shared across segments (see fig. 6.19). That is why the plots for parameters and FLOPS are almost the same for each model, except SplitMixer-III. . . . .	166
6.24	Comparison of our proposed SplitMixer architectures with state-of-the-art models that do not use external data for training. Results are shown over CIFAR- $\{10,100\}$ datasets. Notice that <b>we have not optimized our models for the best performance</b> . Rather, we ran the ConvMixer and our models using the exact same code, parameters, and machines to measure how much we can save parameters and computation relative to ConvMixer. Please consult [2] for a more detailed comparison of ConvMixer with other models. We have borrowed some data from <a href="https://paperswithcode.com/">https://paperswithcode.com/</a> to generate these plots. . . . .	169
6.25	The role of the number of blocks $b$ on model performance. The FLOPS of models over CIFAR-100 are just slightly higher than CIFAR-10, thus not visible in the rightmost panel. . . . .	173

# List of Tables

2.1	Reconstruction performance and model selection performance of models on the simulated dataset. CC-MRCE variations uniformly outperform MRCE and CGGM. CC-MRCE variations with more strict constraints perform better. . . . .	20
2.2	Functional connectivity reconstruction performance of seven tasks with different models. Models are numbered as follows. 1: CGGM; 2: VAE; 3: Spectral Mapping; 4: Random $\mathbf{B}$ ; 5: CC-MRCE (Ours). . . . .	26
2.3	Overlap ratios (%) of predicted B (SCs-FCs mapping) across 10 folds for seven tasks. . . . .	30
3.1	fMRI scan details for six tasks. . . . .	43
3.2	Numerical values of weighted F1 of ablation study settings. Training time ranges from 51 seconds / epoch for length-8 inputs to 298 seconds / epoch for length-256 inputs. Models converge to a relatively stable loss level within 20 epochs. . . . .	45
3.3	Model comparisons with length-256 inputs. . . . .	53
4.1	Numerical AUC-ROC values of the classifiers presented in fig. 4.1. . . . .	68
4.2	Starting FID without generator finetuning (pre-trained LF-Lafite is used here) and correct retrievals in a batch of size 300 using embeddings obtained from $f_{mi}$ and $f_{mc}$ . In the top table, models are trained with MSE+cos loss. In the bottom table, defaults are: with threshold, with image augmentation, using random caption. For the two options with auxiliary modules, the model is finetuned from MSE + cos model since training from scratch gives much worse results. FID evaluations are omitted if the retrieval performance of a setting is strictly worse than its competitors. . . . .	83
4.3	Correct image retrievals in a batch of size 300 when combining different models. . . . .	84
4.4	FID of the pipeline under different settings. . . . .	85
4.5	2-way identifications accuracy of the pipeline under different settings. . . . .	86

4.6	n-way identification accuracy (%) with $n = 2, 5, 10, 50$ . . . . .	86
6.1	Results on ImageNet. . . . .	137
6.2	Numbers corresponding to the bar charts in fig. 6.8c. . . . .	139
6.3	Comparison with other models. The best numbers in each column are highlighted in bold. The number of parameters and FLOPS are averaged over CIFAR-10 and CIFAR-100 for our models. Notice that some variants of SplitMixer perform better than the numbers reported here over Flowers102 and Food101 datasets. Results, except ConvMixer and our model, are reproduced from [3] where they have trained models for 200 epochs. We have trained ConvMixer and SplitMixer for 100 epochs. . . . .	168
6.4	Results over ImageNet-1k. Models trained and evaluated on $224 \times 224$ images. . . . .	170
6.5	Ablation study of SplitMixer-I-256/8 with a split ratio of 2/3 (Top-1 Acc). . . . .	171
6.6	Model throughput for SplitMixer-A-256/8 on a Tesla v100 GPU with 32GB RAM over a batch of 64 images of size $224 \times 224$ , averaged over 100 such batches. . . . .	174

# Chapter 1

## Introduction

The understanding of human cognitive processes has fascinated scholars and philosophers for centuries. With recent neuroimaging advancements, researchers now have noninvasive measures to peek inside the human brain and observe its activity. In particular, diffusion imaging that reveals brain structures and functional magnetic resonance imaging (fMRI) that captures activity states provide complementary information about the brain—just like knowing both hardware devices and software applications. On the other hand, machine learning techniques, especially deep learning models, are becoming much more mature and have achieved outstanding performance and surpassed human-level performance in various evaluation benchmarks [4, 5, 6]. This dissertation focuses on bridging human and machine intelligence: we will model and learn representations of brain data with convex optimization and deep learning models and, inversely, investigate and improve deep learning models with computational neuroscience tools or brain insights.

The first half of this dissertation will study the brain from a network perspective. The human brain is one of the most complex networks. It is multiplex (multimodal) in nature, with anatomical networks organized over multiple spatial scales and functional

networks interactive over multiple temporal scales [7]. Understanding brain connectivity and activity, especially the relationship between the anatomical brain structures and the dynamics of neural processes, is a central question in neuroscience. An increasing number of theoretical and empirical studies approach the function of the human brain from a network perspective [8]. This is made feasible by the development of new image acquisition techniques, large-scale initiatives using those techniques to collect population data, as well as new tools from graph theory, convex optimization, and deep learning—especially graph neural networks that take graphs as inputs.

Modeling the brain as a dynamic network has a high representation capability and provides critical insights into neural processes; however, with the memory limitation of computation hardware, it is not straightforward to model each voxel as a network node. On the other hand, averaging voxel signals into region-level activities inevitably causes a loss of fine-grained information. In order to retain as much information as possible, we will also study the representations with voxel-level inputs in the next part of the dissertation. Finally, we will demonstrate the successes of borrowing neuroscience insights to examine and improve machine learning models. In summary, we seek to answer the following research questions:

- How can we model coupled brain networks of different modalities under domain constraints?
- How can we effectively learn a comprehensive representation that incorporates both structural and dynamic functional signals?
- How does the brain represent visual stimuli, and how can we decode them from brain activities?
- What properties can we learn from the brain, and how can we apply them to build



better deep learning models?

To answer these questions, we have developed novel statistical and machine learning methods and interpreted the results. Each of the following chapters presents new methodologies or findings to either (1) better understand the brain structure and function utilizing deep learning, or (2) understand and design deep learning architectures and algorithms motivated by neuroscience tools and insights. In particular, this dissertation is organized as follows:

- In chapter 2, we will study the *multiplex* nature of brain network data, namely, jointly model structural and functional brain networks. We propose a convex optimization framework whose objective function does not pose a Gaussian assumption on the data and is able to take in existing domain knowledge through a hard constraint term. These objective function modifications are particularly beneficial for neuroscience data which is rarely Gaussian and often has a small sample size. Using domain knowledge to constrain the feasible space is critical for finding the solution with limited data. We then develop a nested fast iterative shrinkage-thresholding algorithm to solve the formulated problem. We test our framework on various task data of the Human Connectome Project, obtaining important insights into the structural backbones of the functional activities.
- In chapter 3, we will extend upon the previous chapter, covering the other two aspects of the brain network data, namely *multiscale* and *temporal*, while still considering both brain modalities (structural and functional). To this end, we propose an efficient graph neural network with temporal convolution layers. The model utilizes both structural brain networks and a learned sample-level latent adjacency matrix to capture fixed and changing local connections. We also propose an inner-cluster smoothing module to learn long-range relationships between regions. In

addition, a graph attribution method is employed to study how various parts, both spatial and temporal, of brain activities contribute to different tasks carried out by subjects. This results in interesting findings regarding the brain region, task, and subject heterogeneity.

- Differing from the previous two chapters that study the brain under different discrete task states, chapter 4 will investigate brain activities with stimuli with *continuous* semantics. It will also do so with voxel-level signals instead of regio-level ones. In particular, we use the fMRI data collected when subjects view images from everyday scenes to study visual decoding and encoding processes. We will first demonstrate that incorporating additional text modality is essential for a visual decoding pipeline, and brain signals align better with a semantic-rich latent space. We then show how we can use this to decode complex image stimuli from brain activities. To counter the issue of data scarcity, we utilize a pretrained vision-language latent space: we first map fMRI signals into this latent space and then finetune a pretrained conditional generative model with customized loss functions. As a result, our pipeline is able to reconstruct image stimuli with both naturalness and fidelity. We will also show that visual encoding can be performed similarly. With this same dataset, we will continue the representation study of voxel-level brain activities in chapter 5. We perform signal reconstruction and category classification of causative stimuli with two simple models, one autoencoder, and one multi-label classifier. We will show that brain signals are highly redundant, and removing 80% of the voxels does not affect much of the information they carry. The studies in this chapter also confirm again that brain signals reside in a much lower dimensional semantic space. We will also report other interesting findings regarding how redundancy and dependency differ across categories, hemispheres,

regions, and voxels.

- Finally, in chapter 6, we will present two artificial neural network modeling inspired by neuroscience. The first one studies and unveils the inherent biases of various deep neural networks by feeding them structured noise inputs and applying techniques adapted from classification images and spike-triggered averaging. These analyses are also useful for network filter visualizations, as well as adversarial attacks and defenses. The second one is inspired by the sparse activation of neurons and the modular structure of the brain. To bring more sparsity into deep learning models, we separate neurons into overlapping or non-overlapping segments (“modules”) and route information inside each one. In a sense, it is also similar to representing a higher-dimensional state with lower-dimensional coordinates. The resulting model can achieve a similar level of performance to its unmodified counterparts while having significant parameter and computation savings. Both modeling efforts in this chapter highlight the importance of the interplay between deep learning and brain sciences.

## Chapter 2

# Modeling Coupled Networks: Structural Connectivity and Static Functional Connectivity

Modeling the behavior of coupled networks is challenging due to their intricate dynamics. For example, in neuroscience, it is of critical importance to understand the relationship between functional neural processes and anatomical connectivities. Modern neuroimaging techniques allow us to separately measure functional connectivity through fMRI imaging and the underlying white matter wiring through diffusion imaging. Previous studies have shown that structural edges in brain networks improve the inference of functional edges and vice versa. In this chapter, we investigate the idea of coupled networks through an optimization framework by focusing on interactions between structural edges and functional edges of brain networks. We consider both types of edges as observed instances of random variables that represent different underlying network processes. The proposed framework does not depend on Gaussian assumptions and achieves a more robust performance on general data compared with existing approaches. To incorporate existing

domain knowledge into such studies, we propose a novel formulation to place hard network constraints on the noise term while estimating interactions. This not only leads to a cleaner way of applying network constraints but also provides a more scalable solution when network connectivity is sparse. We validate our method on multishell diffusion and task-evoked fMRI datasets from the Human Connectome Project, leading to both important insights on structural backbones that support various types of task activities as well as general solutions to the study of coupled networks.

## 2.1 Introduction

Recently, there has been an effort to move research from the investigation of single networks to the more realistic scenario of multiple coupled networks. In this chapter, we consider the case of pairs of networks,  $(\mathcal{G}_1, \mathcal{G}_2)$ , that are categorized into different modalities over a population. Such coupled network systems can be found in infrastructures of modern society (energy-communication), financial systems (ownership-trade), or even human brains (anatomical substrate-cortical activation). Our goal is to reconstruct one network from information on the other and, during such a process, obtain a concise interpretation of how one network affects the other.

To achieve the above goal, we consider an edge-by-edge formulation. We treat one set of edges in  $\mathcal{G}_1$  as predictors and the other set of edges in  $\mathcal{G}_2$  as response variables in a multivariate linear regression model. Past research for the above problem relies on the restrictive Gaussian assumption, which simplifies the problem but is difficult to justify, especially in the domain of brain architectures. Adopting a Gaussian assumption on non-Gaussian data can significantly prevent the detection of conditional dependencies and may lead to incorrectly inferred relationships among variables.

The learning of relationships between two different modalities can be difficult without

---

sufficient data. As a result, in sparser data settings, the ability to specify constraints based on domain knowledge can be beneficial. For example, in the case of brain data, functional edges have mainly local influences, and structural edges are more responsible for long-distance influences [9, 10]. We want preferences encoded in domain knowledge to guide the selection of partial correlations of unexplained noise terms in the constructed model.

Based on the above motivations, we propose a flexible and efficient framework CC-MRCE (**C**onvex-set **C**onstrained **M**ultivariate **R**egression with **C**ovariance **E**stimation) that simultaneously learns both regression coefficients between two coupled networks and the correlation structure of noise terms. In a departure from existing methods, our framework encodes domain knowledge as a set of convex constraints and adopts a pseudolikelihood-based neighborhood-selection objective in partial correlation estimation, which has been shown to be more robust to non-Gaussian data. Because of the CC-MRCE objective’s bi-convex nature, we alternately solve a regression sub-problem and a constrained partial correlation sub-problem until convergence. The latter sub-problem requires feasible solutions under given domain constraints that we render tractable via a modified two-stage proximal gradient descent method.

We illustrate the use of our method in the context of the human brain. Brain data presents one of the greatest technical challenges in analysis and modeling due to a network-based characterization [8, 11], non-Gaussian nature of data, high dimensionality, a small number of samples, and the need to incorporate domain knowledge. We apply the proposed framework to the Human Connectome Project (HCP) dataset [12], where two coupled networks are constructed from fMRI scans (representing cortical activation) and diffusion scans (representing the anatomical substrate). We successfully predict a brain functional network from the given structural network; our method outperforms previous state-of-art methods, and our obtained models are easier to interpret. We investigate

the structure-function coupling for seven different tasks. Our findings agree with the nature of fMRI tasks and brain region functions in existing literature, thus validating our model’s ability to discover meaningful couplings.

In the following sections, we propose a regularized multiple regression approach that adapts to non-Gaussian data (sections 2.2.1 and 2.2.2), and incorporate prior domain knowledge to model estimation by formulating constraints into an optimization problem (section 2.2.3). We then develop a fast method based on nested FISTA for solving the proposed optimization problem (section 2.3). Finally, we show the effectiveness of our model on HCP brain data using quantitative comparisons with existing approaches as well as a qualitative analysis (section 2.4.2).

## 2.2 Constrained Multiple-Output Regression Formulation

In this section, we first introduce existing works on multiple regression under Gaussian assumptions and then motivate our approach under non-Gaussian settings and domain constraints.

### 2.2.1 Multiple-output Regression Problem

Let  $\mathcal{D}$  be an  $n$ -subject sample set in which all subjects share the same coupling  $(\mathcal{G}_1, \mathcal{G}_2)$  but have different edge values. For subject  $i$  in  $\mathcal{D}$ , let  $\mathbf{x}^{(i)} = (x_1^{(i)}, \dots, x_p^{(i)})$  be  $p$ -dimensional inputs that represent edge values in the first modality network  $\mathcal{G}_1$ , and  $\mathbf{y}^{(i)} = (y_1^{(i)}, \dots, y_p^{(i)})$  be  $p$ -dimensional outputs<sup>1</sup> that stand for edge values in the second modality network  $\mathcal{G}_2$ . We assume that the inputs  $\mathbf{x}_i$  and outputs  $\mathbf{y}_i$  are correlated through

<sup>1</sup>In general, models do not require the same dimensions for inputs and outputs. We use the equality setting only for simplicity.

a multivariate linear regression model:

$$\mathbf{y}^{(i)} = \mathbf{x}^{(i)}\mathbf{B} + \boldsymbol{\epsilon}^{(i)}, \quad \text{for } i = 1, \dots, n \quad (2.1)$$

where  $\mathbf{B}$  is the  $p \times p$  regression coefficient matrix and its element  $\beta_{jk}$  is the regression coefficient that measures the cross-modality impact of edge  $x_j$  to edge  $y_k$ , and  $\boldsymbol{\epsilon}^{(i)}$  is the noise vector of subject  $i$ . The model can be expressed in the matrix form:

$$\mathbf{Y} = \mathbf{X}\mathbf{B} + \mathbf{E} \quad (2.2)$$

where row  $i$  of  $\mathbf{X} \in \mathbb{R}^{n \times p}$  and  $\mathbf{Y} \in \mathbb{R}^{n \times p}$  are the structural and functional edge vectors  $\mathbf{x}^{(i)}$  and  $\mathbf{y}^{(i)}$  of subject  $i$ .

A straightforward approach to estimating  $\mathbf{B}$  is to solve  $p$  separate regression problems, assuming noise terms are independent and uncorrelated. Recently, advanced methods have been proposed to exploit the correlation in noise terms to improve the modeling. They accomplish the goal by introducing an assumption that noise terms  $\boldsymbol{\epsilon}^{(1)}, \dots, \boldsymbol{\epsilon}^{(n)}$  are all *i.i.d.* Gaussian  $\mathcal{N}(\mathbf{0}, \boldsymbol{\Omega}^{-1})$  and then simultaneously estimating regression coefficients  $\mathbf{B}$  and inverse covariance matrix  $\boldsymbol{\Omega}$  of the noise terms. Two popular methods along this direction are MRCE [13] and CGGM [14, 15, 16]. The MRCE method considers the conditional distribution  $\mathbf{Y}|\mathbf{X} \sim \mathcal{N}(\mathbf{X}\mathbf{B}, \boldsymbol{\Omega}^{-1})$  and estimates both  $\mathbf{B}$  and  $\boldsymbol{\Omega}$  by alternately minimizing the negative conditional Gaussian likelihood, with the  $\ell_1$  lasso penalty applied on the entries of  $\mathbf{B}$  and  $\boldsymbol{\Omega}$ . The other method, CGGM, further assumes that  $\mathbf{X}$  and  $\mathbf{Y}$  are jointly Gaussian. Under such formulation, the conditional distribution of  $\mathbf{Y}|\mathbf{X}$  is given by  $\mathcal{N}(-\mathbf{X}\boldsymbol{\Omega}_{XY}\boldsymbol{\Omega}^{-1}, \boldsymbol{\Omega}^{-1})$ , which reparameterizes the regression coefficient  $\mathbf{B}$  as  $-\boldsymbol{\Omega}_{XY}\boldsymbol{\Omega}^{-1}$ . Compared with MRCE, the objective of CGGM is based on the negative conditional Gaussian likelihood as well, but is jointly convex for  $\boldsymbol{\Omega}_{XY}$  and  $\boldsymbol{\Omega}$ , and



therefore more friendly to computation.

## 2.2.2 Relaxing Gaussian Assumptions

Although MRCE and CGGM have received significant attention in solving multi-output regression problems, one drawback of these two approaches is the Gaussian assumption, especially for applications to brain data [17, 18, 19]. Recall that MRCE assumes the Gaussian noise, and CGGM further assumes joint Gaussian distribution over both inputs and outputs. We tested whether the HCP structural and functional data is Gaussian with a significance level of 0.05. The test rejects the Gaussian null hypothesis for 97.5% of structural edges and 36.3% of functional ones. Since our sample size is small, false negatives are more likely to occur [20], namely failing to reject the Gaussian hypothesis when the underlying data is non-Gaussian. Therefore, the proportion of non-Gaussian data in brain networks is expected to be even higher. Thus, relying on Gaussian assumptions is likely to affect the constructed models negatively.

To avoid a Gaussian assumption, we propose a pseudolikelihood approach for learning multi-output regression models by optimizing the following objective function:

$$\begin{aligned} \min_{\{\mathbf{B}_k\}, \{\omega_{jk}\}} & \left[ -n \sum_{j=1}^p \log \omega_{jj} + \frac{1}{2} \sum_{j=1}^p \sum_{i=1}^n \left( \omega_{jj} (\mathbf{y}_j^{(i)} - \mathbf{x}^{(i)} \mathbf{B}_j) \right. \right. \\ & \left. \left. + \sum_{k \neq j} \omega_{jk} (\mathbf{y}_k^{(i)} - \mathbf{x}^{(i)} \mathbf{B}_k) \right)^2 + \lambda_1 \sum_{j < k} |\omega_{jk}| + \lambda_2 \sum_{j < k} |\beta_{jk}| \right] \end{aligned}$$

or in a neat matrix notation:

$$\begin{aligned} \min_{\mathbf{B}, \Omega} & -n \log |\Omega_D| + \frac{1}{2} \text{tr} \left( (\mathbf{Y} - \mathbf{X}\mathbf{B})^T (\mathbf{Y} - \mathbf{X}\mathbf{B}) \Omega^2 \right) \\ & + \lambda_1 \|\mathbf{B}\|_1 + \lambda_2 \|\Omega_X\|_1 \end{aligned} \tag{2.3}$$

where  $\mathbf{\Omega} = \{\omega_{jk}\}$  denotes the inverse covariance matrix,  $\mathbf{B} = \{\beta_{jk}\}$  denotes the coefficient matrix, and  $\mathbf{\Omega}_D$  and  $\mathbf{\Omega}_X$  denote the diagonal and off-diagonal parts of  $\mathbf{\Omega}$ . The proposed objective can be considered as a reparameterization of the Gaussian likelihood with  $\mathbf{\Omega}^2$  and an approximation to the log-determinant term. It has been proven that under mild singularity conditions, such reparameterization can guarantee estimation consistency for distributions with sub-Gaussian tails [21, 22].

Next, we develop an optimization algorithm to minimize the objective. The objective function itself is not jointly convex for both variables  $\mathbf{B}$  and  $\mathbf{X}$ , but remains convex with respect to each of them while keeping the other fixed. Therefore, we adopt the alternating minimization idea. In the  $t$ -th iteration, we first fix  $\mathbf{B}$  as the estimated  $\hat{\mathbf{B}}^{(t-1)}$  from the previous  $(t-1)$ -th iteration, and calculate the empirical covariance matrix  $\mathbf{S}$  of noise terms:

$$\mathbf{S}^{(t-1)} = \frac{1}{n}(\mathbf{Y} - \mathbf{X}\mathbf{B}^{(t-1)})^T(\mathbf{Y} - \mathbf{X}\mathbf{B}^{(t-1)}) \quad (2.4)$$

Next, we estimate the inverse covariance matrix:  $\mathbf{\Omega}^{(t)}$  with the given  $\mathbf{S}^{(t-1)}$  as a constant:

$$\mathbf{\Omega}^{(t)} = \arg \min_{\mathbf{\Omega}} -\log |\mathbf{\Omega}_D| + \frac{1}{2} \text{tr}(\mathbf{S}^{(t-1)}\mathbf{\Omega}^2) + \lambda_2 \|\mathbf{\Omega}_X\|_1 \quad (2.5)$$

Observe that the above subproblem follows CONCORD's original form, which is more robust to heavy-tailed data [22, 23] than the conventional Gaussian likelihood approach and can be efficiently solved using proximal gradient methods with a convergence rate of  $O(1/t^2)$  [24]. Lastly, we keep  $\mathbf{\Omega}$  fixed at  $\mathbf{\Omega}^{(t)}$  and optimize the regression coefficients  $\mathbf{B}$ :

$$\mathbf{B}^{(t)} = \arg \min_{\mathbf{B}} \frac{1}{2} \text{tr}((\mathbf{Y} - \mathbf{X}\mathbf{B})^T(\mathbf{Y} - \mathbf{X}\mathbf{B})\mathbf{\Omega}^{(t)}) + \lambda_1 \|\mathbf{B}\|_1 \quad (2.6)$$

Note that subproblem (eq. (2.6)) is convex when  $\mathbf{\Omega}^{(t)}$  is positive semi-definite. We

present the above regression-based approach as CONCORD-MRCE in **Algorithm 1** (pseudocode).

### 2.2.3 Imposing Domain Constraints

Due to the limited sample size of real-world datasets and their high dimensionality, incorporating accurate domain constraints can reduce the search space and avoid overfitting.

Under a linear mapping assumption, the partial correlation of response variables arises only from correlations in the noise terms. Therefore,  $\mathbf{\Omega}$  not only represents the inverse covariance of noise terms, but also equals the conditional inverse covariance of  $\mathbf{Y}|\mathbf{X}$ . The nonzero entries of  $\mathbf{\Omega}$  encode direct relationships among the target modality outputs  $\mathbf{Y}$  that cannot be explained by weighted inputs  $\mathbf{X}\mathbf{B}$  of the source modality. It will be beneficial if the zero-vs-nonzero structure of  $\mathbf{\Omega}$  is partially given by domain experts and used as hard constraints in model estimation.

More formally, let  $M$  be a binary matrix that has the same dimensions as  $\mathbf{\Omega}$ . We can define a convex matrix set  $\mathbb{S}_M$ , containing all matrices that share the same set of zero entries with  $M$ . We can then improve the previous regression-based approach to estimate  $\mathbf{\Omega}$  under the domain constraint that takes the form of  $\mathbf{\Omega} \in \mathbb{S}_M$ , written in an equivalent unconstrained convex form:

$$\begin{aligned} \hat{\mathbf{\Omega}} = \arg \min_{\mathbf{B}, \mathbf{\Omega}} & -\frac{n}{2} \log |\mathbf{\Omega}_D^2| + \frac{1}{2} \text{tr}((\mathbf{Y} - \mathbf{X}\mathbf{B})^T (\mathbf{Y} - \mathbf{B}\mathbf{X}) \mathbf{\Omega}^2) \\ & + \lambda_1 \|\mathbf{B}\|_1 + \lambda_2 \|\mathbf{\Omega}_X\|_1 + \mathbb{I}\{\mathbf{\Omega} \in \mathbb{S}_M\} \end{aligned} \quad (2.7)$$

where  $\mathbb{I}\{\mathbf{\Omega} \in \mathbb{S}_M\}$  is an indicator function. This formulation can be extended to  $\mathbf{\Omega} \in C$  whenever  $C$  is a closed convex set of positive definite matrices.

**Algorithm 1:** CONCORD-MRCE

---

**Input:** penalty parameter  $\lambda_1$  and  $\lambda_2$   
Initialize  $t = 0$ ,  $\hat{\mathbf{B}}^{(0)} = \mathbf{0}$  and  $\hat{\mathbf{\Omega}}^{(0)} = \hat{\mathbf{\Omega}}(\hat{\mathbf{B}}^{(0)})$ .  
**while** *not converged* **do**  
    *step 1:* Compute  $\hat{\mathbf{S}}^{(t-1)} = \hat{\mathbf{S}}(\hat{\mathbf{B}}^{(t-1)})$  as eq. (2.4);  
    *step 2:* Update  $\hat{\mathbf{\Omega}}^{(t)} = \hat{\mathbf{\Omega}}(\hat{\mathbf{S}}^{(t-1)})$  as eq. (2.5) by calling CONCORD( $\hat{\mathbf{S}}^{(t-1)}$ );  
    *step 3:* Update  $\hat{\mathbf{B}}^{(t)} = \hat{\mathbf{B}}(\hat{\mathbf{\Omega}}^{(t)})$  as eq. (2.6);  
**return**  $\mathbf{B}$  and  $\mathbf{\Omega}$

---

## 2.3 Alternating Minimization Solution

In this section, we show how to adapt the previous solution when the inverse covariance  $\mathbf{\Omega}$  is constrained during estimation. Notice that the ideas of **Algorithm 1** can be mostly used to solve eq. (2.7), except for the  $\mathbf{\Omega}$ -update step, which is now affected by the added constraints. The new  $\mathbf{\Omega}$ -update step needs to solve the following sub-problem:

$$\begin{aligned} \mathbf{\Omega}^{(t)} = \arg \min_{\mathbf{\Omega}} & -\log |\mathbf{\Omega}_D^2| + \text{tr}(\mathbf{S}^{(t-1)}\mathbf{\Omega}^2) \\ & + \lambda_2 \|\mathbf{\Omega}_X\|_1 + \mathbb{I}\{\mathbf{\Omega} \in \mathbb{S}_M\} \end{aligned} \quad (2.8)$$

We follow the FISTA (Fast Iterative Soft-Thresholding Algorithm [25]) approach that is used in CONCORD [24]. This method utilizes an accelerated gradient algorithm using soft-thresholding as its proximal operator for the  $L_1$  norm and achieves a fast  $O(1/t^2)$  convergence rate. Previous work [24] has also applied FISTA for partial correlation estimation and proved its efficiency. To adapt our constrained problem into the FISTA framework, we split our objective function, eq. (2.8), into a smooth part and a non-smooth part:

$$\begin{aligned} h_1(\mathbf{\Omega}) &= -\log |\mathbf{\Omega}_D^2| + \text{tr}(\mathbf{S}\mathbf{\Omega}^2) \\ h_2(\mathbf{\Omega}) &= \lambda_2 \|\mathbf{\Omega}_X\|_1 + \mathbb{I}\{\mathbf{\Omega} \in \mathbb{S}_M\} \end{aligned}$$

For any symmetric matrix  $\mathbf{\Omega}$ , the gradient of the smooth function can be easily calculated as:  $\nabla h_1(\mathbf{\Omega}) = -2\mathbf{\Omega}_D^{-1} + 2\mathbf{\Omega}\mathbf{S}$ . With this formulation, we now adapt the FISTA iterative scheme to solve our network-constrained problem (eq. (2.8)):

$$\alpha_{t+1} = (1 + \sqrt{1 + 4\alpha_t^2})/2 \quad (2.9)$$

$$\mathbf{\Theta}^{(t+1)} = \mathbf{\Omega}^{(t)} + \frac{\alpha_t - 1}{\alpha_{t+1}}(\mathbf{\Omega}^{(t)} - \mathbf{\Omega}^{(t-1)}) \quad (2.10)$$

$$\mathbf{\Omega}^{(t+1)} = \text{prox}_{\gamma h_2}[\mathbf{\Theta}^{(t+1)} - (n\tau_t/2)\nabla h_1(\mathbf{\Theta}^{(t+1)})] \quad (2.11)$$

where  $\tau_t$  is the step length and  $t$  denotes the iteration number.  $\gamma$  is a trade-off parameter that controls the extent to which the proximal operator maps points towards the minimum of  $h_2(\mathbf{\Omega})$ , with larger values of  $\gamma$  associated with larger movement near the minimum.

In these iterative steps,  $\mathbf{\Theta}_{t+1}$  is an expected position, updated purely by momentum. Within each loop, the algorithm first takes a gradient step of the estimated future position (eq. (2.10)), and then applies the proximal mapping of a closed convex function  $h_2(\mathbf{\Omega})$ .

In contrast to the standard FISTA approach, the composite function  $h_2(\mathbf{\Omega})$  consists of a sparsity penalty and a network-constrained indicator function. More specifically, we can write down the explicit form of eq. (2.11) according to the proximal operator definition:

$$\begin{aligned} \hat{\mathbf{\Omega}} &= \hat{\mathbf{\Omega}}_X + \mathbf{A}_D \\ \hat{\mathbf{\Omega}}_X &= \text{prox}_{\gamma h_2}(\mathbf{A}_X) \\ &= \arg \min_{\mathbf{\Omega}_X \in \mathcal{C}} \frac{1}{2\gamma} \|\mathbf{\Omega}_X - \mathbf{A}_X\|_F^2 + \lambda_2 \|\mathbf{\Omega}_X\|_1 \end{aligned} \quad (2.12)$$

where  $\mathbf{A} = \mathbf{\Theta}^{(t+1)} - (n\tau_t/2)\nabla h(\mathbf{\Theta}^{(t+1)})$ . Instead of directly solving the original problem (eq. (2.12)), we consider its dual problem as follows. Let matrix  $\mathbf{H}$  be the dual variable

of matrix  $\mathbf{\Omega}$ . We have:

$$\begin{aligned}
& \min_{\mathbf{\Omega}_X \in C} \left( \frac{1}{2\gamma} \|\mathbf{\Omega}_X - \mathbf{A}_X\|_F^2 + \lambda_2 \max_{\|\mathbf{H}_X\|_\infty \leq 1} \text{vec}(\mathbf{H}_X)^T \text{vec}(\mathbf{\Omega}_X) \right) \\
&= \max_{\|\mathbf{H}_X\|_\infty \leq 1} \min_{\mathbf{\Omega}_X \in C} \frac{1}{2\gamma} \left( \|\mathbf{\Omega}_X - (\mathbf{A}_X - \gamma\lambda_2\mathbf{H}_X)\|_F^2 \right. \\
&\quad \left. - \|\mathbf{A}_X - \gamma\lambda_2\mathbf{H}_X\|_F^2 + \|\mathbf{A}_X\|_2^2 \right) \tag{2.13}
\end{aligned}$$

where  $\mathbf{A}_D$  and  $\mathbf{A}_X$  denote the diagonal and off-diagonal part of  $\mathbf{A}$ . Since the initial objective function above is convex in  $\mathbf{\Omega}_X$  and concave in  $\mathbf{H}_X$ , we exchange the order of the minimum and maximum operator in which the inner minimization problem has an obvious solution through orthogonal projection theorem [26], written as

$$\mathbf{\Omega}_X = \mathbb{P}_C (\mathbf{A}_X - \gamma\lambda_2\mathbf{H}_X) \tag{2.14}$$

where  $\mathbb{P}_C$  is defined as an projection operator:  $\mathbb{P}_C(\mathbf{\Gamma}) = \text{argmin}_{\mathbf{R} \in C} \|\mathbf{R} - \mathbf{\Gamma}\|_F^2$  and its orthogonal projection operator  $\mathbb{P}_{C^\perp}$  is defined as  $I - \mathbb{P}_C$ . In the special case that  $C = \mathbb{S}_M$ , projection  $\mathbb{P}_C(\mathbf{\Gamma})$  is equivalent to removing invalid nonzero entries of the input matrix  $\mathbf{\Gamma}$ .

Inserting the optimal  $\mathbf{\Omega}_X$  back into objective (eq. (2.13)), we now obtain the final dual form of the problem (eq. (2.12)):

$$\begin{aligned}
\hat{\mathbf{H}}_X &= \arg \min_{\|\mathbf{H}_X\|_\infty \leq 1} \|\mathbf{A}_X - \gamma\lambda_2\mathbf{H}_X\|_2^2 \\
&\quad - \|\mathbb{P}_{C^\perp}(\mathbf{A}_X - \gamma\lambda_2\mathbf{H}_X)\|_2^2 \tag{2.15}
\end{aligned}$$

where any solution  $\hat{\mathbf{H}}_X$  to the dual problem corresponds to a primal solution through eq. (2.14). Since the dual objective is continuously differentiable and constraints on  $l_\infty$ -norm are convex, we can again efficiently solve it with additional inner FISTA iterations, which is to minimize an equivalent composite objective  $\min g_1(\mathbf{H}_X) + g_2(\mathbf{H}_X)$ , where

**Algorithm 2:** Constrained-CONCORD**Input:** sample covariance matrix  $\mathbf{S}$ , sparsity pattern  $E$ , penalty parameter  $\lambda_2$ **Output:** partial correlation matrix  $\mathbf{\Omega}$ set  $\mathbf{\Theta}^{(1)} = \mathbf{\Omega}^{(0)} \in \mathbb{S}_M^p$ ,  $\alpha_1 = 1$ ,  $\tau_{(0,0)} \leq 1$ ,  $c < 1$ **while not converged do**

$$\mathbf{G}^{(t)} = \nabla h_1(\mathbf{\Theta}^{(t)})$$

search largest  $\tau_t \in \{c^j \tau_{(t,0)}\}_{j=0,1,\dots}$ 

$$\mathbf{A}^{(t)} = \mathbf{\Theta}^{(t)} - (n\tau_t/2)\mathbf{G}^{(t)}$$

set  $\tilde{\mathbf{\Theta}}^{(1)} = \mathbf{H}^{(0)} \in \mathbb{S}_{d \times d}$ ,  $\tilde{\alpha}_1 = 1$ ,  $\kappa_{(0,0)} \leq 1$ ,  $\tilde{c} < 1$ **while not converged do**

$$\tilde{\mathbf{G}}_X^{(t')} = \nabla g_1(\mathbf{H}_X^{(t')})$$

search largest  $\kappa_{t'} \in \{\tilde{c}^j \kappa_{(t',0)}\}_{j=0,1,\dots}$ 

$$\mathbf{H}_X^{(t')} = \text{prox}_{g_2} \left( \tilde{\mathbf{\Theta}}_X^{(t')} - \kappa_{t'} \tilde{\mathbf{G}}_X^{(t')} \right)$$

check backtrack line search criterion

$$\tilde{\alpha}_{t'+1} = (1 + \sqrt{1 + 4\tilde{\alpha}_{t'}^2})/2$$

$$\tilde{\mathbf{\Theta}}^{(t'+1)} = \mathbf{H}_X^{(t')} + \frac{\alpha_{t'} - 1}{\alpha_{t'+1}} \left( \mathbf{H}_X^{(t')} - \mathbf{H}_X^{(t'-1)} \right)$$

compute  $\kappa_{(t'+1,0)}$ 

$$\mathbf{\Omega}^{(t)} = \mathbb{P}_C(\mathbf{A}_X^{(t)} - \gamma\lambda_2\mathbf{H}_X^{(t)}) + \mathbf{A}_D^{(t)}$$

check backtrack line search criterion

$$\alpha_{t+1} = (1 + \sqrt{1 + 4\alpha_t^2})/2$$

$$\mathbf{\Theta}^{(t+1)} = \mathbf{\Omega}^{(t)} + \frac{\alpha_t - 1}{\alpha_{t+1}} (\mathbf{\Omega}^{(t)} - \mathbf{\Omega}^{(t-1)})$$

compute  $\tau_{(t+1,0)}$  $g_1(\mathbf{H}_X)$  is smooth and  $g_2(\mathbf{H}_X)$  is non-smooth:

$$g_1(\mathbf{H}_X) = \|\mathbf{A}_X - \gamma\lambda_2\mathbf{H}_X\|_2^2 - \|\mathbb{P}_{C^\perp}(\mathbf{A}_X - \gamma\lambda_2\mathbf{H}_X)\|_2^2$$

$$g_2(\mathbf{H}_X) = I_{\{\|\mathbf{H}_X\|_\infty \leq 1\}}(\mathbf{H}_X).$$

To adapt FISTA to the problem (eq. (2.15)), the only thing left is to obtain the gradient of the smooth function  $g_1(\mathbf{H}_X)$ , for which we need the lemma below:

**Lemma 1** [27] *If  $g$  is a closed proper convex function, and for any positive  $t$ , define a proximal operator  $g_t(\mathbf{x}) := \inf_{\mathbf{u}} [g(\mathbf{u}) + \frac{1}{2t}\|\mathbf{u} - \mathbf{x}\|^2]$ , then its infimum is attained at*

**Algorithm 3:** CC-MRCE

**Input:** penalty parameter  $\lambda_1$  and  $\lambda_2$ , convex constraint set  $C$ .

Initialize  $\hat{\mathbf{B}}^{(0)} = \mathbf{0}$  and  $\hat{\mathbf{\Omega}}^{(0)} = \hat{\mathbf{\Omega}}(\hat{\mathbf{B}}^{(0)})$ .

**while** *not converged* **do**

- step 1:* Compute  $\hat{\mathbf{S}}^{(t-1)} = \hat{\mathbf{S}}(\hat{\mathbf{B}}^{(t-1)})$  as eq. (2.4);
- step 2:* Update  $\hat{\mathbf{\Omega}}^{(t)} = \hat{\mathbf{\Omega}}(\hat{\mathbf{S}}^{(t-1)})$  by calling  
Constrained-CONCORD( $\hat{\mathbf{S}}^{(t-1)}, E, \lambda_1$ );
- step 3:* Update  $\hat{\mathbf{B}}^{(t)} = \hat{\mathbf{B}}(\hat{\mathbf{\Omega}}^{(t)})$  as eq. (2.6);

**return**  $\mathbf{B}$  and  $\mathbf{\Omega}$

the unique point  $\text{prox}_t(g)(\mathbf{x})$ . Further,  $g_t$  is continuously differentiable on  $\mathbb{E}$  with a  $1/t$ -Lipschitz gradient given by  $\nabla g_t(\mathbf{x}) = (\mathbf{x} - \text{prox}_t(g)(\mathbf{x}))/t$ .

According to Lemma 1, we simply plug  $g(\mathbf{u}) = \delta(\mathbf{u} \in C)$ , and obtain that  $\text{prox}_t(g)(\mathbf{x}) = \mathbb{P}_C(\mathbf{x})$ . Therefore the gradient  $\nabla g_1$  is calculated as:

$$\nabla g_1(\mathbf{H}_X) = -2\gamma\lambda_2 \cdot \mathbb{P}_C(\mathbf{A}_X - \gamma\lambda_2\mathbf{H}_X). \quad (2.16)$$

and the proximal mapping of function  $g_2(\mathbf{\Omega})$  becomes a projection of  $\mathbf{H}_X$  into the  $L_\infty$ -ball:

$$(\text{prox}_{g_2}(\mathbf{H}))_{ij} = \text{sign}(\mathbf{H}_X) \min\{|\mathbf{H}_X|, \mathbf{1}_X\} \quad (2.17)$$

This completes the modified  $\mathbf{\Omega}$ -update step in the constrained setting (referred to as Constrained-CONCORD). Its corresponding pseudocode is presented in **Algorithm 2** (pseudocode). The overall framework of simultaneously estimating  $\mathbf{\Omega}$  and  $\mathbf{B}$  is presented in **Algorithm 3** (pseudocode), which we name CC-MRCE.



---

**Notes on Pseudocodes**

CC-MRCE (**Algorithm 3**) improves CONCORD-MRCE (**Algorithm 1**) by imposing hard constraints on the solution space.

In every step of the while loop, CC-MRCE calls sub-function Constrained-CONCORD to estimate the partial correlation matrix  $\mathbf{\Omega}^{(t)}$  under hard constraints, i.e. the solution  $\mathbf{\Omega}^{(t)}$  shares the same zero and nonzero pattern as matrix  $E$  (shown in **Algorithm 2**). Such constraints can be replaced by any set of convex constraints on  $\mathbf{\Omega}^{(t)}$ . Note that superscript  $t'$  and  $t$  in **Algorithm 2** are iteration counters of inner and outer stages in the Constrained-CONCORD algorithm, respectively.

**Lemma 2** *Let  $L(g_1)$  be the Lipschitz constant of the gradient of objective function  $g_1(\mathbf{H}_X)$ , then  $L(g_1) \leq 2\lambda_2^2\gamma^2$ .*

Although we use line-search to pick a proper step length  $\kappa_{t'}$  in the inner-loop of **Algorithm 2**, it can be replaced with a constant step length  $\kappa_{t'} = 2\lambda_2^2\gamma^2$  according to the above lemma.

## 2.4 Experiments on Synthetic and HCP Datasets

We present two sets of experiments. First, to understand our method’s strengths and limitations, we utilize simulated datasets that allow us to inspect both reconstruction performance and model selection performance. We compare the proposed method CC-MRCE with other baselines when the underlying data distribution does not follow the Gaussian assumption. We show that both the non-Gaussian assumption and the network constraints contribute to the improvement of performance. Second, we conduct experiments on the Human Connectome Project (HCP) data [12]. Our model offers a

**Table 2.1:** Reconstruction performance and model selection performance of models on the simulated dataset. CC-MRCE variations uniformly outperform MRCE and CGGM. CC-MRCE variations with more strict constraints perform better.

Model	Reconstruction MSE percentage (100%)	Pearson's $r$ -score	min p-value	max p-value	relative AUC w.r.t. $\Omega$	relative AUC w.r.t. $\mathbf{B}$
CC-MRCE (unconstrained)	50.88	0.709	5.146E-09	0.101	0.520	0.536
CC-MRCE (SNR:1.0)	43.24	0.763	2.265E-11	0.083	0.855	0.640
CC-MRCE (SNR:2.0)	43.18	0.764	2.109E-11	0.073	0.925	0.662
CC-MRCE (perfect)	42.98	0.764	2.692E-11	0.064	1.000	0.671
MRCE	60.22	0.653	8.686E-09	0.174	0.375	0.540
CGGM	76.21	0.552	2.141E-07	0.965	0.330	0.195

quantitative advantage over baseline methods for predicting functional networks from structural ones; at the same time, our results agree with existing neuroscience literature.

## 2.4.1 Application to Simulated Data

### I. Data generation

Using a similar approach to existing works [21, 22, 13], we generate our simulated dataset by first synthesizing two key model parameter matrices  $\Omega_0$  and  $\mathbf{B}_0$ , and then construct input, output, and noise terms (*i.e.*  $\mathbf{x}^{(i)}$ ,  $\mathbf{y}^{(i)}$  and  $\epsilon^{(i)}$ 's).

Note that every positive-definite matrix has a Cholesky decomposition that takes the form of  $\mathbf{L}\mathbf{L}^T$ , where  $\mathbf{L}$  is a lower triangular matrix  $\mathbf{L}$ , and if  $\mathbf{L}$  is sparse enough then  $\mathbf{L}\mathbf{L}^T$  is sparse as well. Therefore, we first sample a sparse lower triangular  $\mathbf{L}$  with real and positive diagonal entries, and then generate our inverse covariance matrix with  $\Omega_0 = \mathbf{L}\mathbf{L}^T$ . The generated  $p \times p$  positive definite matrix  $\Omega_0$  has 10% nonzeros entries and a condition number of 4.3. To demonstrate the robustness of proposed method CC-MRCE on non-Gaussian data, we sample the noise terms  $\{\epsilon^{(i)}\}_{i=1}^n$  according to a multivariate  $t$ -distribution with zero mean and covariance matrix  $\Sigma = \Omega_0^{-1}$ .

Next, we generate a sparse coefficient matrix  $\mathbf{B}_0$  using the matrix element-wise product trick,  $\mathbf{B}_0 = \mathbf{W} \circ \mathbf{K} \circ \mathbf{Q}$ . In this construction approach,  $\mathbf{W}$  has entries with inde-

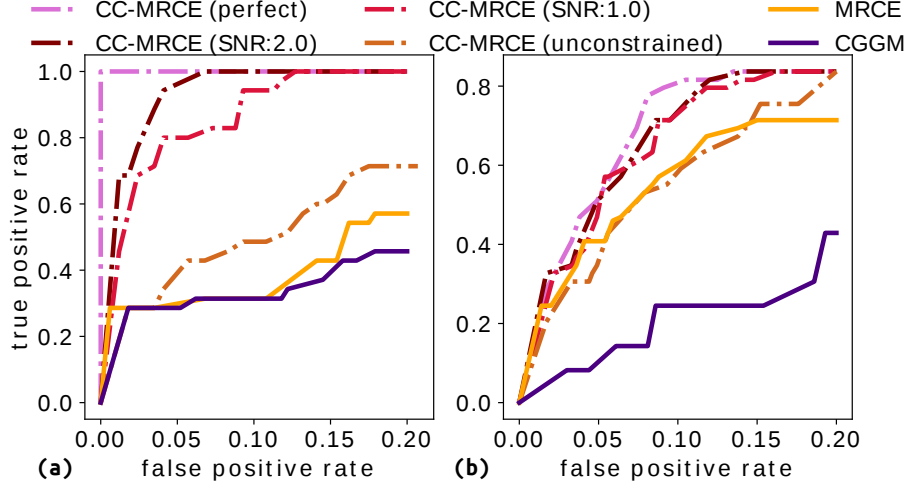
pendent draws from standard normal distribution  $\mathcal{N}(0, 1)$ . Each entry in  $\mathbf{K}$  is drawn independently from a Bernoulli distribution that takes value 1 with probability  $s_1$ .  $\mathbf{Q}$  has rows that are either all one or all zero, which are determined by independent Bernoulli draws with success probability  $s_2$ . Generating the sparse  $\mathbf{B}_0$  in this manner, we not only control its sparsity level, but also forcibly make  $(1 - s_2)p$  predictors be irrelevant for  $p$  responses, and guarantee that each relevant predictor is associated with  $s_1p$  response variables.

In the following experiments, the probabilities  $s_1$  and  $s_2$  are chosen to be 0.15 and 0.8, the sample size  $n$  is fixed at 50, and the input and output dimensions  $p$  and  $q$  are both set to 20. The input  $\mathbf{x}^{(i)}$ 's are sampled from a multivariate normal distribution  $\mathcal{N}(0, \mathbf{\Sigma}_{\mathbf{X}})$  where  $(\mathbf{\Sigma}_{\mathbf{X}})_{jk} = 0.5^{|j-k|}$ , following previous works [28, 29]. The output  $\mathbf{y}^{(i)}$ 's are calculated as the linear model assumption in eq. (2.1). We replicate the above process for independently generating a validation dataset of the same sample size. All penalty parameters are selected simultaneously and tuned according to the validation error.

## II. Method

In this heavy-tailed setting, we compare the performance of CC-MRCE to CGGM and MRCE, which are developed under Gaussian settings. The CGGM implementation that we used in this experiment is provided by [30] and has been optimized for large-scale problems and limited memory. The MRCE implementation is provided by [13]. Both CGGM and MRCE do not adapt to the network constraints we impose here.

In order to inspect the effectiveness of network constraints, we apply multiple variations of constraint sets to the proposed CC-MRCE method, designated as CC-MRCE (unconstrained), CC-MRCE (SNR: 2.0), CC-MRCE (SNR: 1.0) and CC-MRCE (perfect). *Network constraints* in each variation are defined as follows. For CC-MRCE (perfect), we choose  $\mathbb{S}_E = \{\mathbf{\Omega} : \Omega(j, k) = 0 \text{ if } \Omega_0(j, k) = 0\}$ , which forces selected nonzeros in



**Figure 2.1:** (a) ROC curve for  $\Omega$  estimation, (b) ROC curve for  $\mathbf{B}$  estimation. Benefiting from domain constraints, CC-MRCE obtains better ROC curves when uncovering nonzero entries of  $\mathbf{B}$  and  $\Omega$ .

solution  $\Omega$  to completely fall into ground-truth nonzeros. For CC-MRCE (SNR: 2.0) and CC-MRCE (SNR: 1.0), we loosen the feasible set by adding  $50\%\|\Omega_0\|_0$  and  $100\%\|\Omega_0\|_0$  spurious nonzero positions, which are randomly sampled from positions of zero entries in  $\Omega_0$ . For CC-MRCE (unconstrained), we remove all constraints so that the method only relies on regularized multi-regression.

### III. Performance evaluation

We evaluate the reconstruction performance of models using the conventional MSE error (in percentage). Correlation coefficients between predicted and ground-truth outputs are also provided along with their corresponding p-values. In addition, we use the relative area under the curve (AUC) of receiver operating characteristic (ROC) curves [31, 32], with regards to  $\Omega$  and  $\mathbf{B}$ , as key measures to compare model selection performance of all these methods. The AUC of a perfect ROC curve, which would be 1, indicates an ideal recovery of ground truth zero-vs-nonzero structure in  $\Omega$  (or  $\mathbf{B}$ ). However, models with large false-positive rates (FPR) are barely meaningful in real scenarios. To focus

on the initial portion of ROC curves, we control the FPRs simultaneously to be smaller than 0.2 for both  $\mathbf{\Omega}$  and  $\mathbf{B}$  estimation. Thus, the maximum AUC value that a model can reach is just 0.2. For ease of comparison, we provide relative AUC values, divided by 0.2 to normalize to 1. For each method, we run the algorithm with at least 25 appropriate parameter pairs  $(\lambda_1, \lambda_2)$  to get its ROC curve. Recall that all methods in this section are required to estimate 800 parameters given  $n = 50$  samples.

#### IV. Test results

Table 2.1 displays the test results of six different settings in all measures mentioned above. As can be seen, all four CC-MRCE variations (with different network constraints) obtain significantly smaller reconstruction MSE percentages, higher correlation coefficients, and smaller  $p$ -values. Note that CGGM behaves the worst in all measures since it is deeply rooted in the Gaussian setting and is consequently misled by these assumptions. We also see that the performance of CC-MRCE gradually improves when the applied network constraints are more informative.

We also plot ROC curves for  $\mathbf{\Omega}$  and  $\mathbf{B}$  estimation in fig. 2.1.(a) and fig. 2.1.(b), respectively. It is clear that CC-MRCE performs better than MRCE and CGGM, across different choices of network constraints. For  $\mathbf{\Omega}$  estimation, as expected, CC-MRCE with more strict constraints has steeper curves, suggesting that it recovers mostly correct partial relationships between variables with very few spurious connections, and therefore achieves higher AUC scores. CC-MRCE (perfect) behaves perfectly for  $\mathbf{\Omega}$ -ROC, by its definition. For the estimation of matrix  $\mathbf{B}$ , a similar phenomenon demonstrates that network constraints improve the learning of regression coefficients and lead to a better reconstruction performance. Without imposing network constraints, unconstrained formulation of CC-MRCE is likely to generate a biased estimate of  $\hat{\mathbf{\Omega}}$  on small datasets and can not recover ground truth features in  $\mathbf{B}$ .

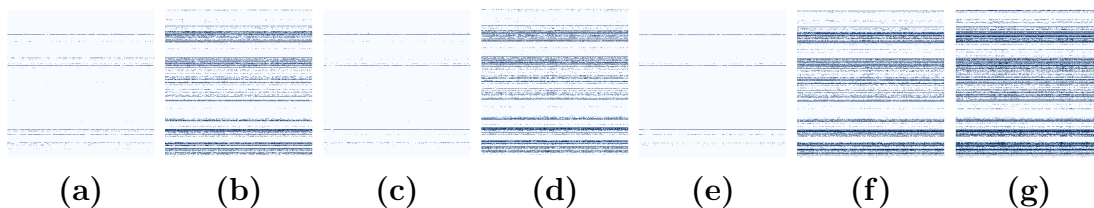
## 2.4.2 Application to Human Connectome Data

### I. Problem formulation

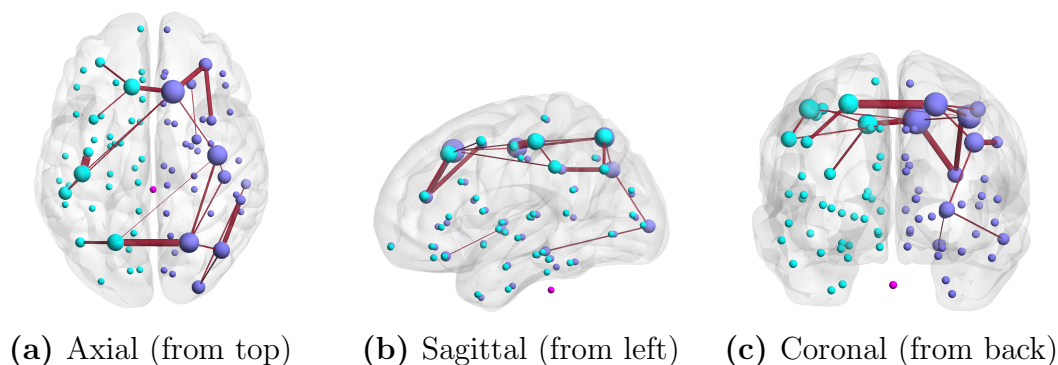
Many works in literature report on coupling between brain structural connectivities (SCs) and functional connectivities (FCs) for both resting state [33, 34, 35, 36] and task-evoked states [34, 37, 38, 39]. One such recent work [40] maps SC to resting-state FC by aligning both the eigenvalues and eigenvectors of a subject’s SC and FC matrices, and evaluates the mapping by reconstructing resting FCs from SCs.

Inspired by the domain observation [10] that two functional connections sharing the same node are more likely to form a meaningful pathway or functional activation pattern, we constrain the partial correlations between non-neighboring edges to be zero, i.e.  $\Omega_{jk} = 0$  if  $e_j$  and  $e_k$  are not incident edges in the fMRI-constructed network.

We conducted experiments by reconstructing task FCs from SCs of 51 subjects with their fMRI data from HCP dataset under seven task states. We used the parcellation scheme in [33] with a spatial scale of 33, resulting in 83 brain regions and 3403 possible edges. Columns in predictor matrix, eq. (2.2),  $\mathbf{X} \in \mathbb{R}^{51 \times 3403}$  represent SCs, whose entries are numbers of white matter streamlines intersecting pairs of brain regions, and columns in response matrix  $\mathbf{Y} \in \mathbb{R}^{51 \times 3403}$  represent FCs, whose entries are functional correlations between cortical activities of brain regions.  $\mathbf{B}$  denotes the mapping (or coupling) between SCs and FCs, and  $\mathbf{E}$  denotes the part of FCs that cannot be explained by SCs. We also normalized SC values to  $(0, 1]$ , a range comparable to FC. We ran 10-fold cross-validation of our model for each task, splitting data in a 9-1 train-validation ratio (46-5 split for the 51 subjects). We selected the optimal hyperparameters  $\lambda_1$  and  $\lambda_2$  by a  $5 \times 5$  grid search in the log-scale between  $10^{-1.6}$  to  $10^{-0.4}$ , keeping the models with smallest Mean Squared Error (MSE) percentage on the validation sets, averaged across 10 folds. In our case, both  $\lambda_1$  and  $\lambda_2$  have optimal values around 0.1 across tasks. Aside from MSE, we also



**Figure 2.2:** Visualizations of the  $\mathbf{B}$  matrix for seven tasks: (a) EMOTION, (b) LANGUAGE, (c) MOTOR, (d) GAMBLING, (e) SOCIAL, (f) RELATIONAL, (g) WM. For each task, each fold in the 10-fold cross-validation may lead to different models. Here, we only show those entries that are nonzero more than five times.



**Figure 2.3:** Visualization of the edges contributing to *all* seven tasks. Node size denotes the degree, and edge width denotes its importance, as in the mapping  $\mathbf{B}$ .

tested the Pearson correlation coefficient between predicted FCs and ground truth FCs, (referred to as Pearson’s  $r$ -score, listed in table 2.2) and minimum, maximum  $p$ -values. The reconstruction MSE percentage is below 1% for the training data and around 8% for the validation data. Strong and significant positive correlations are shown for both training ( $r$ -score around 0.6 to 0.8) and validation ( $r$ -score around 0.5) data. These results indicate our model’s effectiveness in FC reconstruction by exploiting cross-modal coupling between SC-FC and domain prior knowledge on FC-FC relationships.

## II. Performance evaluation

Regarding the ability to find accurate mappings between SCs and FCs, we compared our model with the optimization approach CGGM, a deep learning approach Variational

**Table 2.2:** Functional connectivity reconstruction performance of seven tasks with different models. Models are numbered as follows. 1: CGGM; 2: VAE; 3: Spectral Mapping; 4: Random  $\mathbf{B}$ ; 5: CC-MRCE (Ours).

Tasks		Reconstruction MSE percentage (100%)	Pearson's $r$ -score
EMOTION	1	89.78 $\pm$ 21.95	-0.0309 $\pm$ 0.0285
	2	81.57 $\pm$ 2.38	0.0777 $\pm$ 0.0070
	3	61.54 $\pm$ 33.84	0.4228 $\pm$ 0.0349
	4	17.50 $\pm$ 1.84	-0.0014 $\pm$ 0.0086
	5	<b>8.84</b> $\pm$ 0.84	<b>0.4575</b> $\pm$ 0.0402
LANGUAGE	1	41.38 $\pm$ 7.10	0.0270 $\pm$ 0.0448
	2	72.68 $\pm$ 6.33	0.0815 $\pm$ 0.0081
	3	54.23 $\pm$ 30.18	0.4764 $\pm$ 0.0383
	4	35.02 $\pm$ 58.49	0.0020 $\pm$ 0.0052
	5	<b>7.95</b> $\pm$ 0.87	<b>0.4988</b> $\pm$ 0.0205
MOTOR	1	117.85 $\pm$ 27.32	-0.0016 $\pm$ 0.0456
	2	77.30 $\pm$ 4.24	0.0782 $\pm$ 0.0110
	3	57.26 $\pm$ 29.63	0.4156 $\pm$ 0.0548
	4	21.02 $\pm$ 7.75	0.0023 $\pm$ 0.0090
	5	<b>7.80</b> $\pm$ 0.66	<b>0.4807</b> $\pm$ 0.0480
GAMBLING	1	108.99 $\pm$ 32.83	-0.0211 $\pm$ 0.0795
	2	79.14 $\pm$ 3.65	0.0804 $\pm$ 0.0071
	3	54.73 $\pm$ 30.70	0.4781 $\pm$ 0.0380
	4	22.63 $\pm$ 13.15	0.0033 $\pm$ 0.0072
	5	<b>7.86</b> $\pm$ 1.72	<b>0.5014</b> $\pm$ 0.0301
SOCIAL	1	112.82 $\pm$ 36.07	-0.0064 $\pm$ 0.076
	2	79.42 $\pm$ 3.72	0.0772 $\pm$ 0.0088
	3	47.89 $\pm$ 28.09	0.4912 $\pm$ 0.0353
	4	18.68 $\pm$ 3.74	0.0025 $\pm$ 0.0071
	5	<b>6.81</b> $\pm$ 0.95	<b>0.5578</b> $\pm$ 0.0404
RELATIONAL	1	147.61 $\pm$ 49.24	-0.0265 $\pm$ 0.0754
	2	81.11 $\pm$ 2.98	0.0706 $\pm$ 0.0081
	3	54.86 $\pm$ 30.68	0.4758 $\pm$ 0.0412
	4	31.77 $\pm$ 18.30	-0.0019 $\pm$ 0.012
	5	<b>8.43</b> $\pm$ 1.09	<b>0.4858</b> $\pm$ 0.0460
WM (Working Memory)	1	81.46 $\pm$ 29.72	-0.0447 $\pm$ 0.0566
	2	77.99 $\pm$ 2.89	0.0799 $\pm$ 0.0109
	3	56.25 $\pm$ 32.85	0.4767 $\pm$ 0.0579
	4	103.96 $\pm$ 111.99	-0.0041 $\pm$ 0.0089
	5	<b>7.69</b> $\pm$ 1.76	<b>0.4968</b> $\pm$ 0.0543



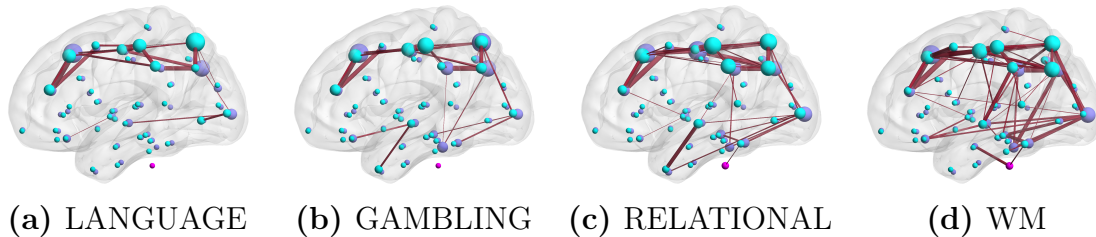
AutoEncoder (VAE) [41], and the Spectral Mapping method [40]. For CGGM, the time required to run 10-fold cross-validation with one set of hyperparameters on a single task ranges from one day to a week. So we ran three sets of hyperparameters on EMOTION and LANGUAGE, selected the best parameter pair, and fixed it for the rest of the five tasks. In particular, we chose penalty terms on  $\mathbf{B}$  and  $\mathbf{E}$  to be both 0.01. For VAE, both the encoder and decoder consist of two fully connected layers, with latent variable dimensions being two. We use MSE as the training loss. In our experiments, increasing the number of layers or latent dimensions of VAE did not improve the final performance. For the Spectral Mapping method, we follow the setup of the original paper, setting the maximum path length  $k$  to seven. The 10-fold cross-validation results of CGGM, VAE, and Spectral Mapping method are reported in table 2.2.

The assumption of Gaussian noise weakens CGGM’s performance on the HCP dataset: it has unstable and large average MSE percentages across tasks, and the correlations between predictions and ground truth are very small and even negative, which are also not statistically significant with regards to  $p$ -values. On the other hand, VAE models have stable MSE percentage (around 80%) and average Pearson’s  $r$ -scores with small standard deviations across all tasks. The correlations of VAE models are weak (all around 0.08), yet statistically significant, with  $p$ -values constantly smaller than 0.0025 for all tasks. This shows that VAE learns a slightly meaningful mapping, but with such a small sample size, deep learning models are unlikely to perform well. Lastly, the Spectral Mapping method is designed to maximize the correlation between fMRI prediction and ground truth for brain data, so it performs well as for correlation, however, the prediction values are off, resulting in high MSE percentages. In all, our regression-based model performs better in both correlation and value reconstruction, showing its superiority in the prediction on non-Gaussian data with small sample sizes.

### III. Result interpretation

Apart from better reconstruction performance, our model also has the advantage of result interpretability. We can explore the SC-FC mapping through the resulting coefficient matrix  $\mathbf{B}$ s. Since our problem definition is  $\text{FC} = \text{SC} \cdot \mathbf{B} + \mathbf{E}$ , the  $i^{\text{th}}$  row in  $\mathbf{B}$  corresponds to  $i^{\text{th}}$  edge pair in the SC vector. In the following, we say an edge  $i$  exists if row  $i$  of  $\mathbf{B}$  has nonzero entries. As the experiment is run under the 10-fold cross-validation setting and each training partition may generate a different mapping  $\mathbf{B}$ , we consider an entry in the common  $\mathbf{B}$  to be nonzero if it is nonzero more than five times in these 10 trials. The results are shown in fig. 2.2. From the figure, we can see for every task, several rows in  $\mathbf{B}$  have many more nonzero entries than the others. This indicates the existence of several significant structural edges being responsible for most of the functional activities. To test if this assumption is valid, we compared  $\mathbf{B}$ s from our model to randomly generated  $\mathbf{B}$ s *with the same levels of sparsity*. The resulting MSE percentages, although having large variances, often have smaller means than that of CGGM and VAE, implying the importance of sparsity level of the coupling. However, the resulting  $r$ -scores of predicted FCs using a random  $\mathbf{B}$  is the lowest among all methods. Together with very large  $p$ -values, the results predicted by random coupling show no correlation between predictions and ground truth. This indicates that our models learned meaningful mapping information from SCs to FCs, and that *structured sparsity* of  $\mathbf{B}$  is important for getting predictions besides the level of sparsity alone.

We now analyze the  $\mathbf{B}$  matrices for different cognitive tasks. During fMRI data acquisition of all seven tasks, participants are presented with visual cues, either as images or videos, and they need to use motions such as pressing buttons to complete the tasks [42]. Interestingly, apart from the LANGUAGE task, the mappings learned by our model predict the strongest contribution of left precentral and left postcentral connection, which

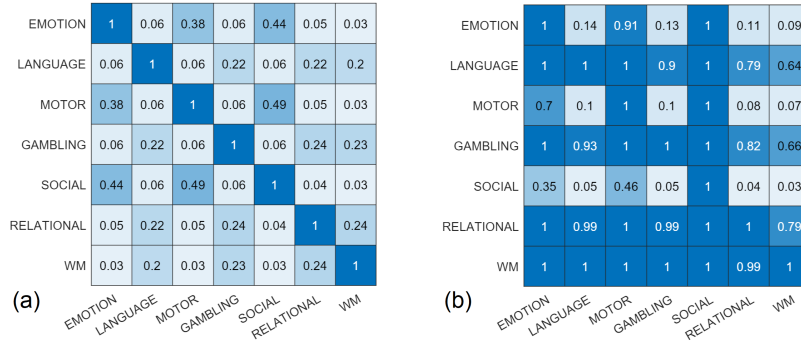


**Figure 2.4:** Task-specific visualizations for high-contributing structural edges. Assuming that the maximum number of nonzero entries of a row in  $\mathbf{B}$  is  $m$ , we only show the edges corresponding to rows containing more than  $m/2$  nonzero entries.

is on the motor cortex responsible for right-side body movement. From this, we assume most participants use their right hands to conduct the required finger movements for these tasks. All mappings also contain edges in the occipital lobe, complying with the visual nature of these tasks. The visualization of common edges that exist in all seven tasks is shown in fig. 2.3. This “backbone” roughly resembles the Default Mode Network [43, 37].

We then examined which structural edges contribute significantly to the functional activity under different tasks. For this, we plot the “high-contributing” edges in LANGUAGE, GAMBLING, RELATIONAL, and WM tasks in fig. 2.4. An edge is considered high-contributing if the number of nonzero entries of its corresponding row in  $\mathbf{B}$  is more than half of the maximum number of nonzero entries of any row in  $\mathbf{B}$ . From fig. 2.4, we notice although a common backbone exists, structural connections in different brain regions are responsible for specific tasks (e.g. SCs in and around the hippocampus area appear to be highly contributing to the WM FCs but not so for the LANGUAGE ones, which is consistent with the literature [44]). We also plot both entry-wise and edge-wise overlap ratios for the mapping of seven tasks in fig. 2.5.

Another interesting phenomenon in fig. 2.2 is that the numbers of nonzero entries of  $\mathbf{B}$  for LANGUAGE, GAMBLING, RELATIONAL, and WM are much larger than the other three tasks: EMOTION, MOTOR, and SOCIAL, although the final model



**Figure 2.5:** Entry-wise and edge-wise overlap ratio for the mapping of seven tasks. (a) considers entry-wise overlap of predicted  $\mathbf{B}$  of different tasks. The value on position (task  $i$ , task  $j$ ) is the entry-wise IoU (Intersection over Union) of task  $i$ 's  $\mathbf{B}$  and task  $j$ 's  $\mathbf{B}$ , i.e. number of nonzero entries in  $\mathbf{B}_i \cap \mathbf{B}_j$  over number of nonzero entries in  $\mathbf{B}_i \cup \mathbf{B}_j$ . (b) considers the SC edges responsible for different tasks. An edge is considered to exist when its corresponding row in  $\mathbf{B}$  has nonzero entries. The value in position (task  $i$ , task  $j$ ) is the number of common SC edges of task  $i$  and task  $j$  over the number of SC edges of task  $j$ .

**Table 2.3:** Overlap ratios (%) of predicted  $\mathbf{B}$  (SCs-FCs mapping) across 10 folds for seven tasks.

EMOTION	LANGUAGE	MOTOR	GAMBLING
5.34	25.58	3.60	35.36
SOCIAL	RELATIONAL	WM	
4.12	37.77	32.53	

for each task has a similar level of prediction performance and similar hyperparameters. This is largely caused by the nature of non-overlapping  $\mathbf{B}$ s for EMOTION, MOTOR, and SOCIAL tasks: their  $\mathbf{B}$  overlap ratios are significantly smaller than the other four tasks, as shown in table 2.3. Here we define the overlap ratio as the number of nonzero entries in the common  $\mathbf{B}$  (entry  $ij$  being nonzero if it's nonzero more than five times) over the number of nonzero entries in  $\mathbf{B}_U = \mathbf{B}_1 \cup \dots \cup \mathbf{B}_{10}$  with  $\mathbf{B}_k$  being the predicted  $\mathbf{B}$  using the  $k^{\text{th}}$  split in 10-fold cross-validation. This is also why we omit these three tasks for fig. 2.4, as the predicted  $\mathbf{B}$ s are not stable across the population, and only a few common connections show significant contributions. We assume this results from group

---

heterogeneity when carrying out these tasks. Further studies with heterogeneous models for the population will be useful to verify this assumption.

## 2.5 Conclusion

In this chapter, we proposed a regularized regression-based approach (CC-MRCE) for jointly learning linear models and partial correlations among variables under domain constraints. Motivated by the neuroscience application of predicting functional brain activities from structural connections, the CC-MRCE method discards the Gaussian assumption and incorporates domain constraints into the model estimation. We further developed a fast algorithm based on nested FISTA to solve the optimization problem. With synthetic data analysis, we demonstrated that both domain constraints and the assumption of non-Gaussian data contribute to the performance improvement of CC-MRCE. Our experimental results on Human Connectome Project data show that CC-MRCE outperforms existing methods on prediction tasks and uncovers couplings that agree with existing neuroscience literature.

## Chapter 3

# Modeling through Graph Neural Networks: Structural Connectivity and Dynamic fMRI

Finding an appropriate representation of dynamic activities in the brain is crucial for many downstream applications. Due to its highly dynamic nature, temporally averaged fMRI (functional magnetic resonance imaging) can only provide a narrow view of underlying brain activities. Previous works lack the ability to learn and interpret the latent dynamics in brain architectures. This chapter proposes an efficient graph neural network model that incorporates both region-mapped fMRI sequences and structural connectivities obtained from DWI (diffusion-weighted imaging) as inputs. We find good representations of the latent brain dynamics through learning sample-level adaptive adjacency matrices and performing a novel multi-resolution inner cluster smoothing. We also attribute inputs with integrated gradients, which enables us to infer (1) highly involved brain connections and subnetworks for each task, (2) temporal keyframes of imaging sequences that characterize tasks, and (3) subnetworks that discriminate between individ-

ual subjects. This ability to identify critical subnetworks that characterize signal states across heterogeneous tasks and individuals is of great importance to neuroscience and other scientific domains. Extensive experiments and ablation studies demonstrate our proposed method’s superiority and efficiency in spatial-temporal graph signal modeling with insightful interpretations of brain dynamics.

### 3.1 Introduction

Neuroimaging techniques such as fMRI (functional magnetic resonance imaging) and DWI (diffusion-weighted imaging) provide a window into complex brain processes. Yet, modeling and understanding these signals has always been a challenge. Network neuroscience [8] views the brain as a multiscale networked system and models these signals in their graph representations: nodes represent brain ROIs (regions of interest), and edges represent either structural or functional connections between pairs of regions.

With larger imaging datasets and developments in graph neural networks, recent works leverage variants of graph deep learning, modeling brain signals with data-driven models and getting rid of Gaussian assumptions that typically existed in linear models [45, 46]. These methods are making progress on identifying physiological characteristics and brain disorders: In [47], authors combine grad-CAM [48] and GIN [49] to highlight brain regions that are responsible for gender classification with resting-state fMRI data. Others [50] propose to use regularized pooling with GNN to identify fMRI biomarkers. However, these works use time-averaged fMRI, losing rich dynamics in the temporal domain. They also do not incorporate structural modality that can provide additional connectivity information missing in the functional modality. Another work [51] embeds both topological structures and node signals of fMRI networks into a low-dimensional latent representation for better identification of depression, but it combines nodes’ temporal

and feature dimensions instead of handling them separately, leading to a suboptimal representation (as discussed in section 3.2.3). To overcome these issues, we propose ReBraiD (Deep **R**epresentations for Time-varying **B**rain **D**atasets), a graph neural network model that jointly models dynamic functional signals and structural connectivities, leading to a more comprehensive deep representation of brain dynamics.

To simultaneously encode signals along spatial and temporal dimensions, some works in traffic prediction and activity recognition domains such as Graph WaveNet [52] alternate between TCN (temporal convolution network) [53] and GCN (graph convolutional network) [54]. Others [55, 56] use localized spatial-temporal graph to embed both domains’ information in this extended graph. Some proposed methods also incorporate gated recurrent networks for the temporal domain such as [57, 58]. We choose to alternate TCN with GCN layers for ReBraiD, as it is more memory and time-efficient and can support much longer inputs. On top of this design, we propose novel “sample-level adaptive adjacency matrix learning” and “multi-resolution inner cluster smoothing,” both of which learn and refine latent dynamic structures. With the choice of the temporal layer, our model is more efficient than other baselines while having the highest performance.

After introducing the proposed model in details (section 3.2.2), we perform extensive ablation studies to examine individual components of the model, explore the best option when alternating spatial and temporal layers for encoding brain activities, and quantitatively show the representation ability of our model (section 3.2.3). Equally important as finding a good representation of brain dynamics is interpreting them. In section 3.3, we utilize IG (integrated gradients) [59] to identify how brain ROIs participate in various processes. This can lead to better behavioral understanding, discovery of biomarkers, and characterization of individuals or groups. We also make the novel contribution of identifying temporally important frames with graph attribution techniques; this can enable more fine-grained temporal analysis around keyframes when combined with other imag-



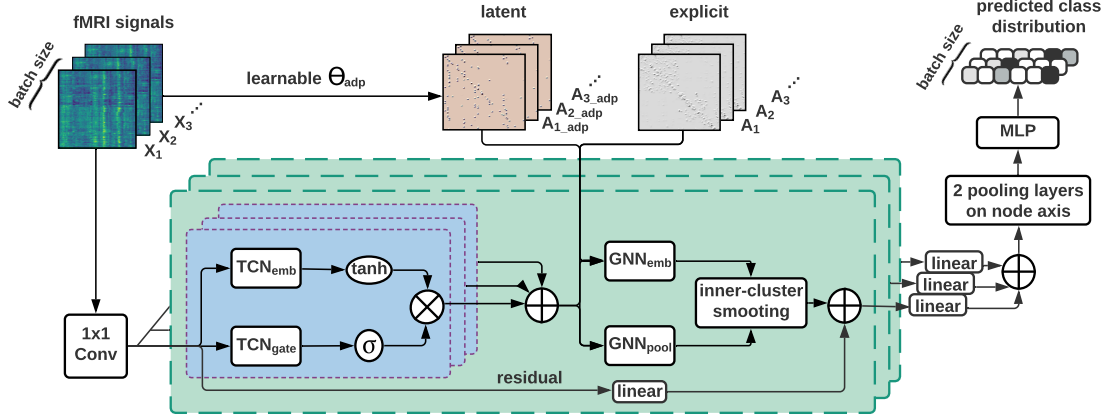
ing modalities such as EEG (electroencephalogram). In addition, our subject-level and group-level attribution studies unveil heterogeneities among ROIs, tasks, and individuals.

## 3.2 Spatial-Temporal GNN for Learning Multi-Modality Brain Representation

In this section, we present ReBraid, an efficient graph neural network model that jointly models both structural and dynamic functional brain signals, providing a more comprehensive representation of brain activities when compared to the current fMRI literature. Unlike typical spatial-temporal GCNs that learn a universal latent structure, we propose sample-level latent adaptive adjacency matrix learning based on input snippets. This captures the evolving dynamics of a task better. We also propose multi-resolution inner cluster smoothing, which effectively encodes long-range node relationships while keeping the graph structure, enabling the model to leverage structural and latent adjacency matrices throughout the process. Together with subject SC and sample-level adjacency matrix learning, the inner cluster smoothing learns and refines latent dynamic structures on limited signal data. We carry out extensive ablation studies and model comparisons to show ReBraid’s superiority in representing brain dynamics.

### 3.2.1 Preliminary

We utilize two brain imaging modalities mapped onto the same coordinate: SC (structural connectivity) from DWI scans, and time-varying fMRI scans. We represent them as a set of  $L$  graphs  $\mathcal{G}_i = (A_i, X_i)$  with  $i \in [1, L]$ .  $A_i \in \mathbb{R}^{N \times N}$  represents the normalized adjacency matrix with an added self-loop:  $A_i = \tilde{D}_{SC_i}^{-\frac{1}{2}} \tilde{SC}_i \tilde{D}_{SC_i}^{-\frac{1}{2}}$ ,  $\tilde{SC}_i = SC_i + I_N$  and  $\tilde{D}_{SC_i} = \sum_w (\tilde{SC}_i)_{vw}$  is the diagonal node degree matrix. Graph signal matrix obtained

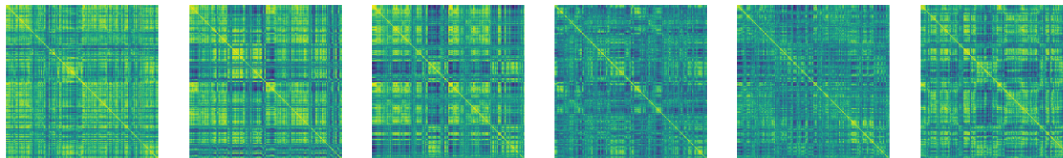


**Figure 3.1:** The proposed ReBraID model for integrating brain structure and dynamics (the architecture shown is for classification). For each batch with batch size  $B$ , input  $X$  has a dimension of  $(B, 1, N, T)$ , and  $A, A_{\text{adp}}$  both have the dimension  $(B, N, N)$ . Note: axis order follows PyTorch conventions. The dimension at the second  $X$  index is the expanded feature dimension. The encoder (green part) encodes temporal and spatial information alternately, producing a latent representation in  $(B, d_{\text{latent}}, N, 1)$ . These embeddings are followed by linear layers for pooling and classification. The final output has a dimension of  $(B, C)$ .

from fMRI scans of the  $i^{\text{th}}$  sample is represented as  $X_i \in \mathbb{R}^{N \times T}$ . Here  $N$  is the number of nodes, and each node represents a brain region;  $T$  is the input signal length on each node. We refine our representation using the task of classifying brain signals  $\mathcal{G}_i$  into one of  $C$  task classes through learning latent graph structures.

### 3.2.2 Method

ReBraID takes  $(A, X)$  as inputs and outputs task class predictions. The overall model structure is shown in fig. 3.1. For the  $i^{\text{th}}$  sample  $X_i \in \mathbb{R}^{N \times 1 \times T}$ , the initial  $1 \times 1$  convolution layer increases its hidden feature dimension to  $d_{h1}$ , outputting  $(N, d_{h1}, T)$ . The encoder then encodes temporal and spatial information alternately, and generates a hidden representation of size  $(N, d_{h2}, 1)$ . The encoder is followed by two linear layers to perform pooling on node embeddings and two MLP layers for classification. Cross

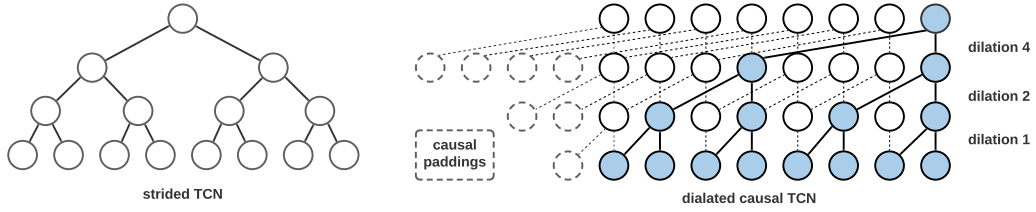


**Figure 3.2:** Functional connectivities (FCs) among  $N$  brain regions, where each  $FC \in \mathbb{R}^{N \times N}$ . The value at  $FC_{ij}$  is calculated as the Pearson correlation coefficient between signals of brain region  $i$  and region  $j$ . The figure shows 6 FCs calculated from 6 consecutive sliding windows within the same fMRI session, with signal window length being 30 and sliding stride being 30. From the figure, we can clearly tell that FCs are highly dynamic.

entropy is used as the loss function:  $L_{CE} = -\sum_i y_i \log \hat{y}_i$ , where  $y_i \in \mathbb{R}^C$  is the one-hot vector of ground truth task labels and  $\hat{y}_i \in \mathbb{R}^C$  is the model’s predicted distribution. We now explain the different components of the model.

### I. Learning sample-level latent graph structures

Structural scans serve as our graph adjacency matrices. However, they remain fixed across temporal frames and across tasks. In contrast, FC (functional connectivities) are highly dynamic, resulting in different connection patterns across both time and tasks, as shown in fig. 3.2. To better capture dynamic graph structures, we learn an adaptive adjacency matrix from each input graph signal. Unlike other works such as [52] that use a universal latent graph structure, our model does not assume that all samples share the same latent graph. Instead, our goal is to give each sample a unique latent structure that can reflect its own signaling pattern. This implies that the latent adjacency matrix cannot be directly treated as a learnable parameter as a part of the model. To solve this, we minimize the assumption down to a shared projection  $\Theta_{\text{adp}}$  that projects each input sequence into an embedding space and use this embedding to generate the latent graph structure. Projection  $\Theta_{\text{adp}}$  can be learned in an end-to-end manner. The generated adaptive adjacency matrix for the  $i^{\text{th}}$  sample can be written as follows (Softmax is applied



**Figure 3.3:** Comparison of strided non-causal TCN (left) and dilated causal TCN (right). For a causal TCN, the causal aspect is achieved through padding (kernel\_size  $- 1$ )  $\times$  dilation number of zeros to the layer’s input. The resulting  $\mathbf{y}$  always has the same length as input  $\mathbf{x}$ , in which  $\mathbf{y}_\tau$  only depends on inputs  $\mathbf{x}_{t \leq \tau}$ . We can view strided non-causal TCN as the rightmost node of a dilated causal TCN.

column-wise):

$$A_{i\_adp} = \text{Softmax} \left( \text{ReLU} \left( (X_i \Theta_{adp}) (X_i \Theta_{adp})^\top \right) \right), \Theta_{adp} \in \mathbb{R}^{T \times h_{adp}} \quad (3.1)$$

## II. Gated TCN (Temporal Convolutional Network)

To encode signal dynamics, we use the gating mechanism as in [60] in our temporal layers:

$$H^{(l+1)} = \tanh \left( \text{TCN}_{\text{emb}}(H^{(l)}) \right) \odot \sigma \left( \text{TCN}_{\text{gate}}(H^{(l)}) \right), \quad (3.2)$$

where  $H^{(l)} \in \mathbb{R}^{N \times d \times t}$  is one sample’s activation matrix of the  $l^{\text{th}}$  layer,  $\odot$  denotes the Hadamard product, and  $\sigma$  is the Sigmoid function. In contrast to TCNs that are generally used in sequence to sequence models that consist of dilated Conv1d and causal padding along the temporal dimension ([61]), we simply apply Conv1d with kernel = 2 and stride = 2 as our  $\text{TCN}_{\text{emb}}$  and  $\text{TCN}_{\text{gate}}$  to embed temporal information. The reason is twofold: first, for a sequence-to-sequence model with a length- $T$  output,  $y_\tau$  should only depend on  $x_{t \leq \tau}$  to avoid information leakage, and causal convolution can ensure this. In contrast, our model’s task is classification, and the goal of our encoder along the temporal dimension is to embed signal information into the feature axis while reducing the temporal dimension

to 1. The receptive field of this single temporal point (with multiple feature channels) is meant to be the entire input sequence. Essentially, our TCN is the same as the last output node of a *kernel-two causal TCN* whose dilation increases by two at each layer (fig. 3.3). Second, from a practical perspective, directly using strided non-causal TCN works the same as taking the last node of dilated causal TCNs, as discussed above, while simplifying the model structure and reducing training time to less than a quarter.

### III. Graph Network layer

In our model, every set of  $l$  temporal layers (section 3.2.3 studies the best  $l$  to choose) is followed by a spatial layer to encode signals with the graph structure. Building temporal and spatial layers alternately helps spatial modules to learn embeddings at different temporal scales, and this generates better results than placing spatial layers after all the temporal ones.

To encode spatial information, [54] uses the first-order approximation of spectral filters to form the layer-wise propagation rule of a GCN layer:  $H^{(l+1)} = \text{GCN}(H^{(l)}) = f(AH^{(l)}W^{(l)})$ . It can be understood as spatially aggregating information among neighboring nodes to form new node embeddings. In the original setting without temporal signals,  $H^{(l)} \in \mathbb{R}^{N \times d}$  is the activation matrix of  $l^{\text{th}}$  layer,  $A \in \mathbb{R}^{N \times N}$  denotes the normalized adjacency matrix with self-connections as discussed in section 3.2.1,  $W^{(l)} \in \mathbb{R}^{d \times d'}$  is learnable model parameters, and  $f$  is a nonlinear activation function of choice. Parameters  $d$  and  $d'$  are the numbers of feature channels.

We view a GCN layer as a local smoothing operation followed by an MLP, and simplify stacking  $K$  layers to  $A^K H$  as in [62]. In ReBraiD, every graph network layer aggregates information from each node’s  $K$ -hop neighborhoods based on both brain structural connectivity and the latent adaptive adjacency matrix: thus, we have both  $A_i^K H^{(l)} W_K$  and  $A_{i\_adp}^K H^{(l)} W_{K\_adp}$  for input  $H^{(l)}$ . We also gather different levels (from 0 to  $K$ ) of neigh-

bor information with concatenation. In other words, one graph convolution layer here corresponds to a small module that is equivalent to  $K$  simple GCN layers with residual connections. We can write our layer as:

$$\begin{aligned} H^{(l+1)} &= \text{GNN}^{(l)}(H^{(l)}) \\ &= \text{MLP} \left[ \text{Concat}_{k=1}^K \left( H^{(l)}, \text{ReLU}(A_i^k H^{(l)}), \text{ReLU}(A_{i\_adp}^k H^{(l)}) \right) \right] \end{aligned} \quad (3.3)$$

Note that in eq. (3.3),  $A_i \in \mathbb{R}^{N \times N}$  and  $H^{(l)} \in \mathbb{R}^{N \times d \times t}$ , and as a result their product  $\in \mathbb{R}^{N \times d \times t}$ . Outputs of different  $\text{GNN}^{(l)}$  layers are parameterized and then skip-connected with a summation. Since the temporal lengths of these outputs are different because of TCNs, max-pooling is used before each summation to make the lengths identical.

#### IV. Multi-resolution inner cluster smoothing

While GNN layers can effectively pass information between neighboring nodes, long-range relationships among brain regions that neither appear in SC nor learned by latent  $A_{\text{adp}}$  can be better captured using soft assignments, similar to DIFFPOOL[63]. To generate the soft assignment tensor  $S^{(l)}$  that assigns  $N$  nodes into  $c$  clusters ( $c$  chosen manually), we use  $\text{GNN}_{\text{pool}}^{(l)}$  that obeys the same propagation rule as in eq. (3.3), followed by Softmax along  $c$ . This assignment is applied to  $Z^{(l)}$ , the output of  $\text{GNN}_{\text{emb}}^{(l)}$  which carries out the spatial embedding for the  $l^{\text{th}}$  layer input  $H^{(l)}$ , producing clustered representation  $\tilde{H}^{(l)}$ :

$$\begin{aligned} S^{(l)} &= \text{Softmax} \left( \text{GNN}_{\text{pool}}^{(l)}(H^{(l)}), 1 \right) \in \mathbb{R}^{N \times c \times t} \\ Z^{(l)} &= \text{GNN}_{\text{emb}}^{(l)}(H^{(l)}) \in \mathbb{R}^{N \times d \times t} \\ \tilde{H}^{(l)} &= S^{(l)\top} Z^{(l)} \in \mathbb{R}^{c \times d \times t} \end{aligned} \quad (3.4)$$

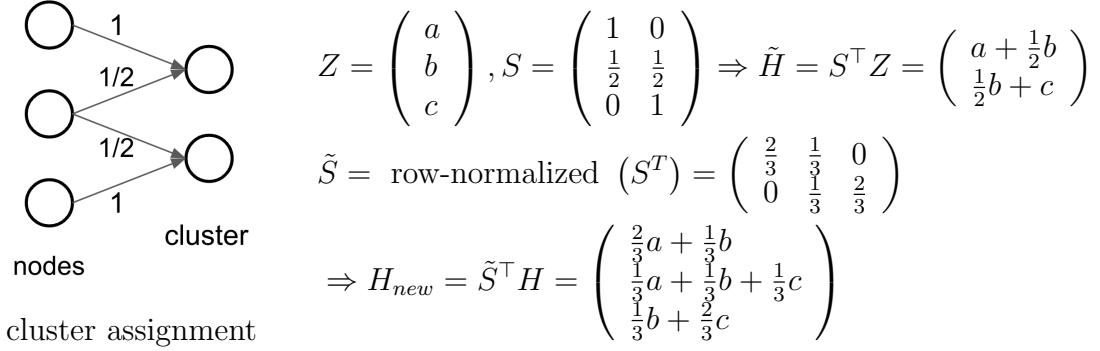
The additional temporal dimension allows nodes to be assigned to heterogeneous clusters at different frames. We find that using coarsened  $A_i^{(l+1)} = S^{(l)\top} A_i^{(l)} S^{(l)} \in \mathbb{R}^{c \times c}$  as the

graph adjacency matrix leads to worse performance compared to using SC-generated  $A_i$  and learned  $A_{i\_adp}$  (comparison in section 3.2.3). In addition, if the number of nodes is changed, residual connections coming from the beginning of temporal-spatial blocks can not be used, impacting the overall performance. To continue using  $A_i$  and  $A_{i\_adp}$  as graph adjacency matrices and to allow residual connections, we reverse-assign  $\tilde{H}^{(l)}$  with assignment tensor obtained from applying Softmax on  $S^{(l)\top}$  along  $N$ , so that the number of nodes is kept unchanged:

$$\begin{aligned}\tilde{S}^{(l)} &= \text{Softmax}(S^{(l)\top}, 1) \in \mathbb{R}^{c \times N \times t} \\ H^{(\ell+1)} &= \tilde{S}^{(l)\top} \tilde{H}^{(l)} \in \mathbb{R}^{N \times d \times t}\end{aligned}\tag{3.5}$$

In fact, eqs. (3.4) and (3.5) perform signal smoothing on nodes within each soft-assigned cluster. Fig. 3.4 shows a toy example: note that we will only show one time slice, and the same operation is done along every  $t$ : on a particular  $t$ , we have  $Z \in \mathbb{R}^{N \times d}$ ,  $S \in \mathbb{R}^{N \times c}$ . We will use  $N = 3$ ,  $c = 2$  and node values  $a, b, c \in \mathbb{R}^d$  for this toy example. In addition, this example is just to illustrate the concept behind the smoothing operation, and Softmax along axis 1 is simplified as row normalization for a more straightforward presentation. In this example,  $1^{st}$  and  $2^{nd}$  nodes are assigned to the first cluster, and  $2^{nd}$  and  $3^{rd}$  nodes are assigned to the second cluster. The final  $H_{new}$  after our smoothing module will mingle the first two nodes' values, and the last two nodes' values (based on assignment weights) while keeping their original node number unchanged.

With the bottleneck  $c < N$ , the model is forced to pick up latent community structures. This inner cluster smoothing is carried out at multiple spatial resolutions: as the spatial receptive field increases with more graph layers, we decrease cluster number  $c$  for the assignment operation. As these GNN layers alternate with TCN layers, the inner cluster smoothing also learns the community information across multiple temporal scales.



**Figure 3.4:** Inner cluster smoothing toy example.

### 3.2.3 Experiments

We use fMRI signals from the CRASH dataset [64] for our experiments. The model classifies input fMRI into six tasks: resting state, VWM (visual working memory task), DYN (dynamic attention task), MOD (math task), DOT (dot-probe task), and PVT (psychomotor vigilance task). We preprocess 4D voxel-level fMRI images into graph signals  $\mathcal{G} = (A, X)$  by averaging voxel activities into regional signals with the 200-ROI cortical parcellation (voxel to region mapping) specified by [65]. We also standardize signals for each region and discard scan sessions with obvious abnormal spikes that may be caused by head movement, etc. DWI scans are mapped into the same MNI152 coordinate and processed into adjacency matrices with the same parcellation as fMRI. Our processed data contains 1940 scan sessions from 56 subjects. Session length varies from 265 frames to 828 frames (see table 3.1 for details). TR (Repetition Time) is 0.91s.

The 1940 scan sessions from CRASH are separated into training, validation, and test sets with a ratio of 0.7-0.15-0.15 (subject-wise split does not lead to any noticeable difference). Each split receives a proportional number of samples for each class. Hyperparameters, including dropout rate, learning rate, and weight decay, are selected using grid search based on validation loss. All results reported in this section are obtained from the test set. For each scan session, we use a stride-10 sliding window to generate input



**Table 3.1:** fMRI scan details for six tasks.

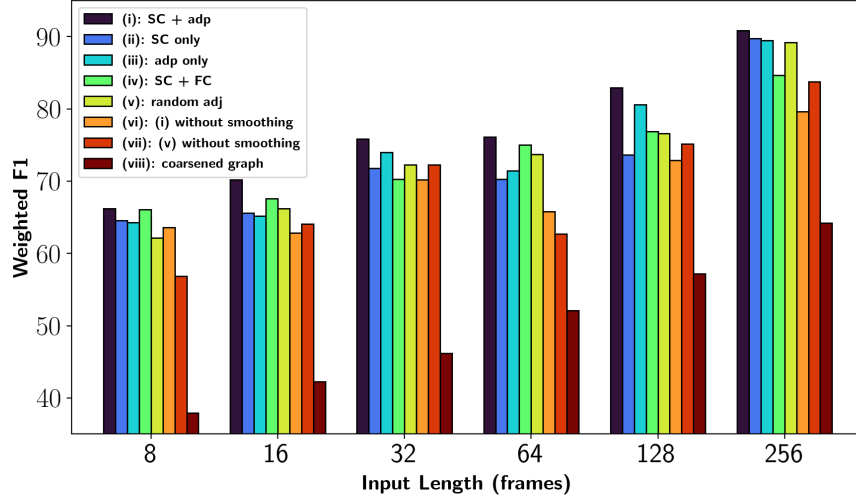
Tasks	Rest	VWM	DYN	DOT	MOD	PVT	(Total)
Valid sessions	209	514	767	155	138	157	1940
Frames / Session	321	300	265	798	828	680	—

sequences (in the following experiments  $T \in \{8, 16, 32, 64, 128, 256\}$ ) and feed them to the model. To encode temporal and spatial information alternately, we find stacking two TCN layers per one GNN layer leads to better performance most times (see the following (IV)). We tested  $h_{\text{adp}} = 2, 5, 10$  in eq. (3.1) for our experiments, and 5 appears to be the best; so we use this value for all the following experiments.  $K = 1, 2, 3$  in eq. (3.3) were tested on a few settings, and  $K = 2, 3$  have a similar performance, both outperforming  $K = 1$ . Since smaller values of  $K$  have smaller computation needs, we use  $K = 2$  for all experiment settings, meaning each GNN layer aggregates information from 2-hop neighbors based on the provided adjacency matrices. We evaluate our model with weighted F1 as the metric in order to account for the imbalance in the number of samples in each task. Our models are written in PyTorch, trained with Google Colab GPU runtimes, and 30 epochs are run for each experiment setting. Code is publicly available <sup>1</sup>.

## I. Model components

**Ablation studies on graph adjacency matrices** For each input sample  $\mathcal{G}_i$ , we test different options to provide graph adjacency matrices to the GNN layer. They include (i) our proposed method: using both adaptive adjacency matrix  $A_{i,\text{adp}}$  and SC-induced  $A_i$ , (ii) only using  $A_i$ , (iii) only using  $A_{i,\text{adp}}$ , (iv) replacing  $A_{i,\text{adp}}$  in setting i with  $A_{i,\text{FC}}$  derived from functional connectivity, and (v) only using random graph adjacency matrices with the same level of sparsity as real  $A$ 's. The results under different settings are reported

<sup>1</sup><https://github.com/sklin93/ReBraID>



**Figure 3.5:** Ablation studies on different input lengths.

in fig. 3.5 and table 3.2.

From the results of setting (ii) plotted in fig. 3.5, we see that removing the adaptive adjacency matrix impacts the performance differently at different input lengths: the gap peaks for signals of length 64–128, and becomes smaller for either shorter or longer sequences. This could suggest the existence of more distinct latent states of brain signals of this length that structural connectivities cannot capture. On the other hand, removing SC (setting (iii)) seems to have a more constant impact on the model performance, with shorter inputs more likely to see a slightly larger drop. In general, only using  $A_{\text{adp}}$  leads to a smaller performance drop than only using SC, indicating the effectiveness of  $A_{\text{adp}}$  in capturing useful latent graph structures. More detailed studies below show that  $A_{\text{adp}}$  learns distinct representations not captured by  $A$ .

As mentioned in section 3.2.2, our motivation behind creating sample-level adaptive adjacency matrices is FC’s highly dynamic nature. Therefore, for setting (iv), we test directly using adjacency matrices  $A_{i,\text{FC}}$  obtained from FC instead of the learned  $A_{i,\text{adp}}$ . In particular,  $A_{i,\text{FC}} = \tilde{D}_{\text{FC}_i}^{-\frac{1}{2}} \tilde{\text{FC}}_i \tilde{D}_{\text{FC}_i}^{-\frac{1}{2}} \in \mathbb{R}^{200 \times 200}$ , where  $(\text{FC}_i)_{vw} = \text{corr}((X_i)_v, (X_i)_w)$ ,

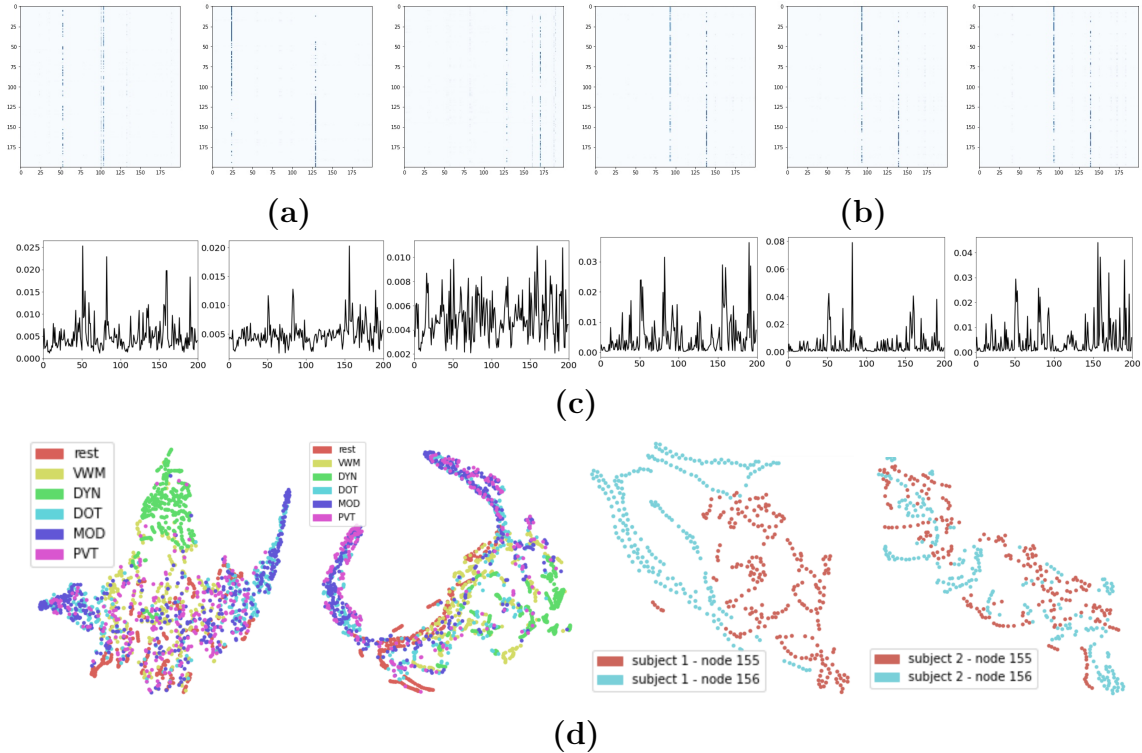
**Table 3.2:** Numerical values of weighted F1 of ablation study settings. Training time ranges from 51 seconds / epoch for length-8 inputs to 298 seconds / epoch for length-256 inputs. Models converge to a relatively stable loss level within 20 epochs.

Input length (frames)	8	16	32	64	128	256
(i): SC + adp	<b>66.19</b>	<b>70.18</b>	<b>75.87</b>	<b>76.14</b>	<b>82.91</b>	<b>90.85</b>
(ii): SC only	64.54	65.58	71.79	70.31	73.63	89.79
(iii): adp only	64.32	65.20	74.01	71.42	80.63	89.46
(iv): SC + FC	66.10	67.58	70.26	75.02	76.91	84.68
(v): random adj	62.17	66.25	72.30	73.72	76.58	89.22
(vi): (i) without smoothing	63.57	62.82	70.19	65.82	72.91	79.65
(vii): (v) without smoothing	56.88	64.08	72.27	62.72	75.16	83.75
(viii): coarsened graph	37.92	42.23	46.18	52.12	57.17	64.25

$\tilde{F}\tilde{C}_i = FC_i + I_N$  and  $\tilde{D}_{FC_i} = \sum_w (F\tilde{C}_i)_{vw}$ . Fig. 3.5 shows  $A_{i\_FC}$  constantly underperforms  $A_{i\_adp}$ , except for being really close for length-8 inputs. Larger performance gaps are observed for longer inputs, where  $\text{Corr}((X_i)_v, (X_i)_w)$  struggles to capture the changing dynamics in the inputs. This demonstrates that our input-based latent  $A_{i\_adp}$  has better representation power than input-based FC. We also notice batch correlation coefficients calculation for  $A_{i\_FC}$  results in a slower training speed than computing  $A_{i\_adp}$ .

An interesting result comes from setting (v), where we use randomly generated Erdős-Rényi graphs with the edge creation probability the same as average edge existence probability of  $A$ 's. Its performance is similar to or even better than settings (ii) and (iii). We examine this further in section 3.3.2.

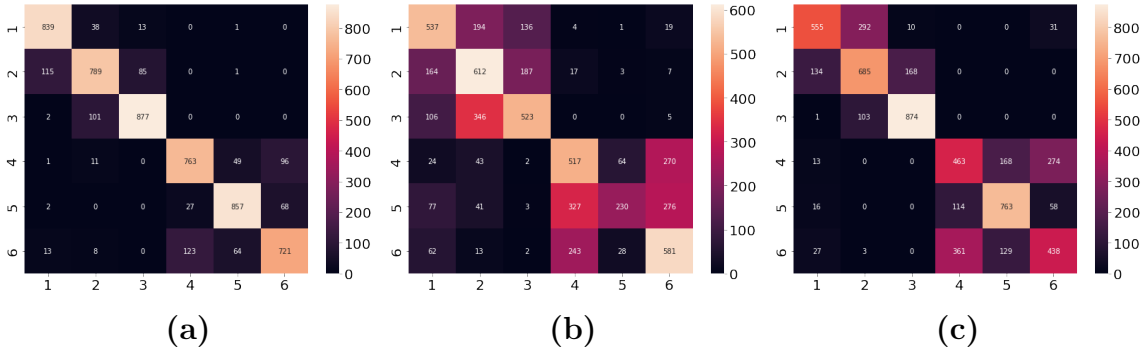
**Latent adaptive adjacency matrix  $A_{adp}$**  The above results demonstrate latent  $A_{adp}$  can complement the task- and temporal-fixed  $A$ . We now show that the learned  $A_{i\_adp}$  is sparse for each sample, has evident task-based patterns, and provides new information beyond  $A_i$ . The sparsity of  $A_{i\_adp}$  can be seen from fig. 3.6a: each input only gets a few important columns (information-providing nodes in GNN). These columns vary from one sample to another, indicating  $A_{adp}$ 's ability to adapt to changing inputs within the



**Figure 3.6:** Learned latent adaptive adjacency matrices. **(a)**  $A_{i\_adp}$  of 3 randomly sampled inputs during the DOT task. **(b)**  $A_{i\_adp}$  of 3 consecutive inputs from a same session during the DOT task. **(c)** column averages of task-averaged  $A_{adp}$  for resting state, VWM, DYN, DOT, MOD, PVT. **(d)** left two: t-SNE of  $X^{(\text{node-2}, 156)}\Theta_{adp}$  in six tasks of one subject; right two: t-SNE of  $X^{(\text{node-155}, 156)}\Theta_{adp}$  during the resting state of two subjects (multiple sessions are aggregated).

same task. However, when we look into inputs generated by consecutive sliding windows (not shuffled) from the same scan session as in fig. 3.6b, we can see the latent structures change smoothly. In addition, when we aggregate samples inside each task, noticeable task-based patterns emerge (fig. 3.6c). These patterns are different from  $\text{Attr}_A$  in fig. 3.11, suggesting that  $A_{adp}$  embeds dynamics not captured by  $A$ .

Quantitatively,  $A_{i\_adp}$  entry values range between (0, 1) because of the Softmax, and only around 2% of entries in  $A_{i\_adp}$  have values larger than 0.05. As a reference, the largest entry value is larger than 0.99. A similar sparsity pattern is found when using synthetic data on the same model, indicating that the sparsity is more due to the model



**Figure 3.7:** Confusion matrices of: (a) ReBraiD (our proposed model), (b) model with coarsened graph (setting (viii)), (c) Graph Transformer (best graph baseline). Tasks are 1-Rest, 2-VWM, 3-DYN, 4-DOT, 5-MOD, 6-PVT. Misclassification pairs clustered at the first three tasks (resting, VWM, DYN) and the latter three (DOT, MOD, PVT). Shown confusion matrices are from models trained on length-256 inputs. We note that these misclassification pairs may differ for models trained on other input lengths (like 128-frame, etc.).

than the underlying biology. Given how  $A_{i\_adp}$  is used in GNN layers, each column of it represents a signal-originating node during message passing. We hypothesize that the model learns the most effective *hubs* that pass information to their neighbors. A related idea is information bottleneck [66]: deep learning essentially compresses the inputs as much as possible while retaining the mutual information between inputs and outputs. In a sense,  $A_{i\_adp}$  represents the compressed hubs for a given input signal. We also note that this sparsity emerges even without any additional constraints. In fact, adding  $L_1$  constraints on  $A_{adp}$  does not change the model performance or the  $A_{i\_adp}$  sparsity level. We hypothesize that the naturally trained  $A_{i\_adp}$  is sparse enough, and further sparsification is unnecessary.

We visualize the projected inputs  $X_i\Theta_{adp}$  in fig. 3.6d, which clearly shows the task, node and subject heterogeneities. Different tasks have varied representations in the latent space for the same node, but DOT, MOD, PVT has similar embedding patterns across individuals and most nodes. Indeed, when looking at the confusion matrix across models (fig. 3.7), the misclassifications mostly cluster between these three tasks, indicating their

natural similarity. We want to note here that adding a learnable bias to  $X\Theta_{\text{adp}}$  does not separate the task embeddings further, nor does it improve overall performance. Subjects also exhibit heterogeneity: the same pair of nodes during the same task can have different embedding distances, thus graph edge weights, for each individual.

**Multi-resolution inner cluster smoothing** To verify the capability of inner cluster smoothing operation in capturing latent graph dynamics, we test the following settings: (vi) using our proposed model and inputs, except removing paralleled  $\text{GNN}_{\text{pool}}$  and inner cluster smoothing module; (vii) previous setting (v) but remove  $\text{GNN}_{\text{pool}}$  and inner cluster smoothing module; (viii) keep  $\text{GNN}_{\text{pool}}$ , but using coarsened graph instead of smoothing (essentially performing DIFFPOOL with an added temporal dimension). In this last setting, we hierarchically pool and reduce the graph to a single node, and we keep the total number of GNN layers the same as our other settings. Values of soft-assigned cluster number  $c$  are chosen to be halved per smoothing module (e.g.,  $N/2, N/4, \dots$ ) for our experiments. Different choices of  $c$  affect the model convergence rate but only have a minor impact on the final performance (see the following). Results are reported in fig. 3.5 and table 3.2.

The above results demonstrate that both setting (vi) and (vii) outperforms (viii) by a large margin, indicating the importance of keeping the original node number when representing brain signals. In addition, all three settings underperform our proposed method. They are also mostly worse than changing graph adjacency matrices as in settings (ii)–(v): this shows the inner cluster smoothing module has a more significant impact in learning latent graph dynamics. We also find using adaptive adjacency matrices and inner cluster smoothing can stabilize training, making the model less prone to overfitting and achieving close-to-best performance over a larger range of hyperparameters (see fig. 3.9).

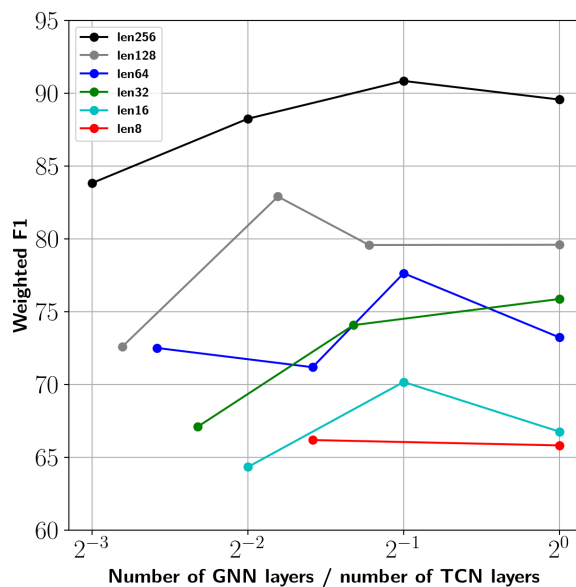
Apart from these three settings, we also test adding pooling regularization terms into the loss function as follows. More specifically, for each soft assignment matrix  $S \in \mathbb{R}^{N \times c \times t}$  in eq. (3.4), we test:

- Similar to DIFFPOOL, to ensure a more clearly defined node assignment, namely each node is only assigned to few clusters (the closer to one the better), we minimize the entropy of single node assignments:  $L_{E_1} = \frac{1}{c} \sum_{i=1}^c H(S_i)$ .
- To ensure a representation separation among nodes, meaning the assignment should not assign all the nodes a same way, we maximize the entropy of node assignment *patterns* across all nodes:  $L_{E_2} = -\frac{1}{c} \sum_{i=1}^c H(\sum_{j=1}^n S_{ij})$ .
- To make the assignment along temporal axis smoother, we penalize assignment variances within a small time window  $[\hat{t}, \hat{t} + \tau]$ :  $L_T = \frac{1}{t-\tau} \sum_{\hat{t}=0}^{t-\tau} \sigma(S_{[\hat{t}, \hat{t}+\tau]})$ , where  $\sigma$  represents standard deviation.

Together with cross-entropy classification loss  $L_{CE}$ , the final loss function of the model becomes  $L_{reg} = \alpha_1 L_{CE} + \alpha_2 L_{E_1} + \alpha_3 L_{E_2} + \alpha_4 L_T$ ,  $\sum_i \alpha_i = 1$ . However, none of these regularization terms lead to much of a difference.

**Choosing the number of GNN layers** The total number of temporal layers depends on the input signal length since each strided TCN layer reduces the temporal length by a factor of two: if the input length is  $2^i$ , there need to be  $i$  temporal layers. *But is alternating every TCN with GNN the best strategy, or do we only need to follow one GNN after a few TCNs?* We study this question with different input lengths.

Model weighted F1 are plotted in fig. 3.8 for all possible GNN to total TCN ratios (e.g. length-256 inputs requires 8 TCN layers. The possible ratios are  $\frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1$  since we can insert one GNN per 8, 4, 2, 1 TCN layers). The figure shows alternating every layer rarely yields the highest performance and the best ratio lies around one GNN per two



**Figure 3.8:** Choosing the number of GNN to TCN layer ratio for different input lengths. In most cases, two TCN layers per GNN layer results in the best model performance in terms of F1.

TCN layers for our dataset. We repeat the experiment for  $K = 1, 3$  (in eq. (3.3)) to rule out the possibility that this result is related to how many neighbors one GNN layer can reach; we find they have roughly the same pattern as the  $K = 2$  case. We hypothesize that a lower GNN to TCN ratio does not capture enough spatial context, while higher ones might be overfitting. We leave exploring the relationship between this ratio and the number of nodes  $N$  to a future study.

The best GNN to TCN ratio also depends on whether model incorporates latent adjacency matrices or not: without  $A_{\text{adp}}$ , length-128 signals achieves its relative best (among all ratios) when having one GNN per two TCNs, but it only needs one GNN per three TCNs if using  $A_{\text{adp}}$ . This shows learning latent structures  $A_{\text{adp}}$  not only improves overall model accuracy but can also reduce model parameters, thus complexity, in achieving better results.



**Choosing the number of soft-assignment clusters** During our experiments, we find that as long as the smoothing module is used, the final performance will be close to each other, only the convergence rates are different. Fig. 3.9b shows how validation loss converges with different  $c$  (cluster number) or when there is no smoothing module. From it, we can observe that halving the numbers (100-50-25-12) is the most helpful setting, and we use it for our other experiments; decreasing the numbers (160-120-80-40) or all larger numbers (all 100) works better than increasing the numbers (12-25-50-100) or all smaller numbers (all 12). With the inner cluster smoothing module, all cluster number settings converge to around 0.23 at their smallest when trained for 30 epochs; their test weighted F1 range from 89.47 (model with 12-25-50-100) to 90.85 (model with 100-50-25-12).

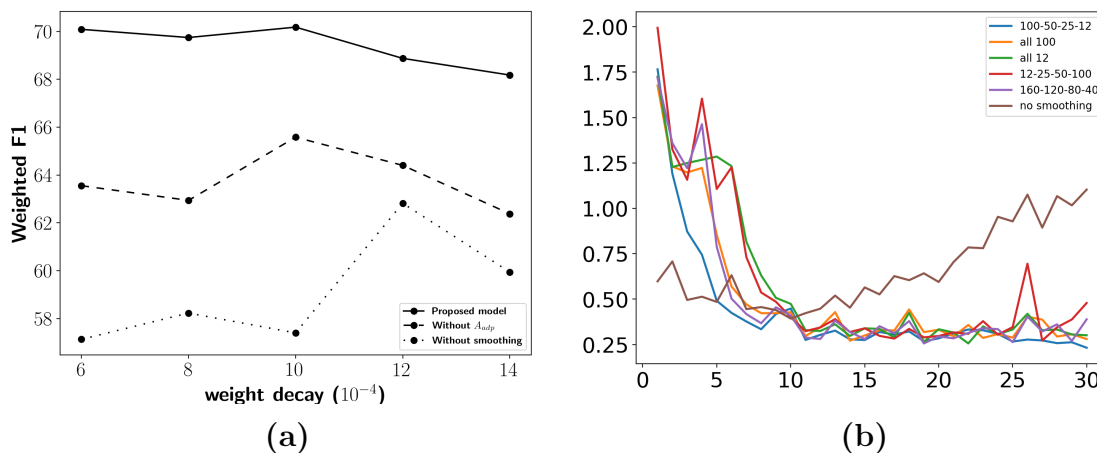
On the contrary, if no smoothing module is used, the model overfits easily, and the validation loss can only reach about 0.4 before going up (with the best set of learning rate and weight decay parameters found with grid search). Understandably, the model is prone to overfitting given the complexity of GNN and the relatively small dataset size. However, our added inner cluster smoothing module effectively counters the effect and further brings the loss down in a stable manner.

## II. Model comparisons

Since we adopt a network view to studying the brain, where brain regions are treated as graph nodes, we source our baselines from graph models. To do so, we examined all models in PyTorch Geometric (PyG)<sup>2</sup> and its temporal extension (PyG-T)<sup>3</sup> as they contain the most up-to-date and well-organized open-source graph neural network model implementations. In particular, we compare our model with the vanilla GCN from [54],

<sup>2</sup><https://pytorch-geometric.readthedocs.io/>

<sup>3</sup><https://pytorch-geometric-temporal.readthedocs.io/>



**Figure 3.9:** (a) adding inner cluster smoothing or input-dependent adaptive adjacency matrix makes the model more stable across various learning rates (results shown are from length-16 inputs). (b) Validation loss v.s. training epochs. Input length is 256, and four smoothing modules are used. Legends are the soft-assignment cluster numbers of the four smoothing modules. Our other experiments use decreasing cluster numbers that halved at each module, corresponding to the 100-50-25-12 choice here.

Chebyshev Graph Convolutional Gated Recurrent Unit (GConvGRU) from [57], GraphSAGE from [68], GAT V2 from [67] and Graph Transformer as in [69]. Baseline models are constructed similar to ours: each has four graph encoding layers taking in both signals and adjacency matrices, followed by two linear layers along the node axis and two linear layers for the final classification. We train baseline models with the same input, loss, optimizer, and epoch settings (all models are well-converged). Grid search is used to optimize the rest of the hyperparameters. We compare weighted F1 and training time per epoch in table 3.3; we also plot our model and Graph Transformer’s confusion matrices in fig. 3.7.

Our model shows significant performance gains and requires less training time than graph baselines. We believe the most critical reason is that the models in PyG treat temporal signals as feature vectors instead of placing them into a separate temporal dimension. Without sequence modeling on the temporal dimension, even the state-of-

**Table 3.3:** Model comparisons with length-256 inputs.

Model	Weighted F1	Training time (s / epoch)
GCN [54]	42.84	713
GAT V2 [67]	50.36	1142
GConvGRU [57]	56.05	9886
GraphSAGE [68]	61.87	1048
Graph Transformer [69]	66.11	1890
MVTS Transformer [70]	88.16	<b>39</b>
<b>ReBraiD</b> (proposed: TCN + GNN)	<b>90.85</b>	298
ReBraiD (TCN only)	71.98	119
ReBraiD (TCN + CNN)	75.79	124

the-art graph attention models (GAT-v2 and graph Transformer) cannot perform well. In addition, almost all models in PyG-T assume one common graph for the inputs (application scenarios are traffic network forecasting, link predictions, etc.), whereas we need to feed different SC for every sample. Out of them, we were able to choose one model (GConvGRU) that supports different adjacency matrices, but it didn't give a satisfactory result. Our proposed ideas of sample-level adaptive adjacency matrix learning and multi-resolution inner cluster smoothing help capture latent brain dynamics and improve performance. The higher model performance here reflects a better encoding ability of brain signals, which can benefit different downstream tasks such as disease and trait prediction.

In addition to graph baselines, we also tested the state-of-the-art model for multivariate time series classification (MVTS Transformer [70]), which has comparable performance to ours. This stresses the critical role of temporal modeling when dealing with dynamic signals, so we tested our model without GNN layers. We experiment with both removing GNN layers altogether and replacing them with  $1 \times 1$  CNN layers: both outperform graph models that focus on the spatial modeling aspect. Although these results

demonstrate that temporal modeling is crucial, adding graph modeling that includes signals' spatial relationships, as proposed, can further improve the performance. Since the MVTS Transformer model has projections to generate queries, keys, and values from the input sequence, it can also implicitly learn spatial relationships between variables (*nodes*). On the other hand, explicitly adding graph components allows the model to utilize prior structures (e.g., SC). The attribution of graph models can also provide better interpretability of brain networks, such as identifying critical region connections, as we will discuss in the following section.

### 3.3 Graph Attribution and Interpretations

In this section, we introduce how we leverage graph attribution, in particular, integrated gradients, to attribute and interpret the importance of both spatial brain ROIs and temporal keyframes, as well as heterogeneities among brain ROIs, tasks, and subjects. These can open up new opportunities for identifying biomarkers for different tasks or diseases and markers for other complex scientific phenomena.

#### 3.3.1 Attribution with IG (Integrated Gradients)

Understanding how signals in different brain regions contribute to the final classification outputs has many important applications in neuroscience, as we have mentioned. As one approach to model interpretability, *attribution* assigns credits to each part of the input, assessing how important they are to the final predictions. [71] gives an extensive comparison between different graph attribution approaches, in which IG [59] is top-performing and can be applied to trained models without any alterations of the model structure. IG also has other desirable properties, such as implementation invariance, that other gradient methods lack. It is also more rigorous and accurate than obtain-

ing explanations from attention weights or pooling matrices that span multiple feature channels. Intuitively, IG calculates how real inputs contribute differently compared to a selected baseline; it does so by aggregating model gradients at linearly interpolated inputs between the real and baseline inputs.

In order to apply IG, we calculate attributions at each point of both input  $A \in \mathbb{R}^{N \times N}$  and  $X \in \mathbb{R}^{N \times T}$  for each sample:

$$\begin{aligned} \text{ATTR}_{\mathcal{G}_{vw}} &= (\mathcal{G}_{vw} - \mathcal{G}'_{vw}) \times \sum_{m=1}^M \frac{\partial F(\mathcal{G}_{\text{Intrpl}})}{\partial \mathcal{G}_{\text{Intrpl}_{vw}}} \times \frac{1}{M}, \\ \mathcal{G} &= (A, X), \quad \mathcal{G}_{\text{Intrpl}} = \mathcal{G}' + \frac{m}{M} \times (\mathcal{G} - \mathcal{G}') \end{aligned} \quad (3.6)$$

$F(\mathcal{G})$  here represents our signal classification model,  $M$  is the step number when making Riemann approximation of the path integral, and  $\mathcal{G}'$  is the baselines of  $\mathcal{G}$  (see section 3.3.2 for more details). Note that eq. (3.6) calculates the attribution of one edge or one node on one sample. The process is repeated for every input point, so attributions  $\text{ATTR}_A, \text{ATTR}_X$  have identical dimensions as inputs  $A, X$ . To obtain the brain region importance of a task, we aggregate attributions across multiple samples of that task.

### 3.3.2 Experiments

This section studies the contributions of different brain ROIs and subnetworks defined by their functionalities. For the subnetwork definition, we choose to use the 17 networks specified in [1], which has a mapping from our previous 200-ROI parcellation<sup>4</sup>. To select baseline inputs, we follow the general principle for attribution methods: when the model takes in a baseline input, it should produce a near-zero prediction, and  $\text{Softmax}(\text{outputs})$  should give each class about the same probability in a classification model. All-zero baselines  $A'$  and  $X'$  can roughly achieve this for our model, so we choose them as our

<sup>4</sup>[https://github.com/ThomasYeoLab/CBIG/blob/master/stable\\_projects/brain\\_parcellation](https://github.com/ThomasYeoLab/CBIG/blob/master/stable_projects/brain_parcellation)

baseline inputs. Step number  $M$  is set to 30. The IG computation is done on 900 inputs for each task to get an overall distribution.

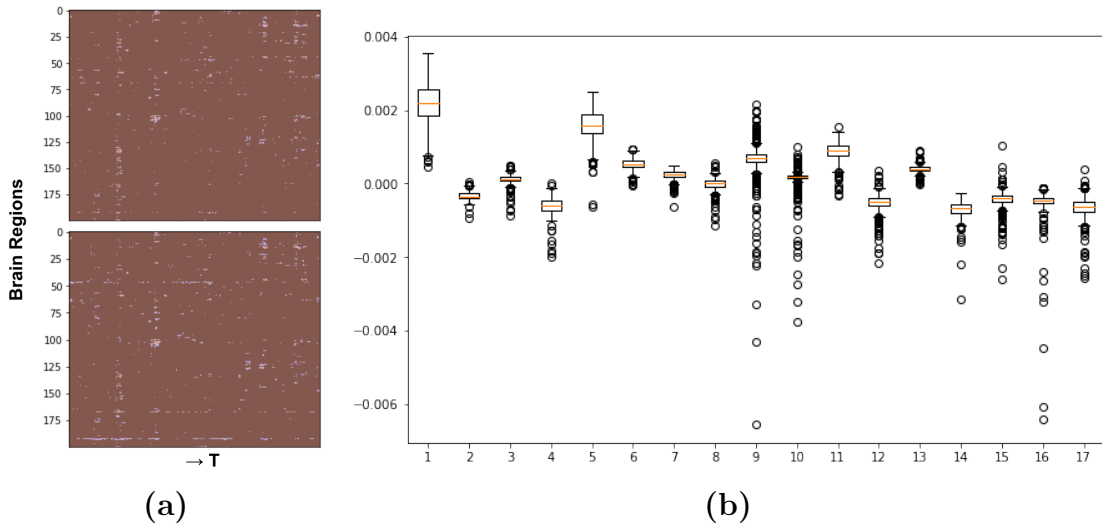
The extracted high-attribution regions and connections should be reproducible across different initializations to be used for downstream tasks. Since the overall problem is non-convex, we empirically test and confirm the attribution reproducibility with two randomly initialized models before proceeding to the following analyses. In addition, [71] demonstrates IG’s consistency (reproducibility among a range of hyperparameters) and faithfulness (more accurate attribution can be obtained with better-performing models). Since our model has higher performance with longer inputs, we compute IG attributions of a model trained on length-256 input signals in this section.

## I. Temporal importance

On the single input level, we can attribute which parts of the inputs in  $\mathcal{G}_i$  are more critical in predicting the target class by looking into  $(\text{ATTR}_X)_i$ . This attribution map not only shows which brain regions contribute more but also reveals the important signal frames. One critical drawback of fMRI imaging is its low temporal resolution, but if we know which part is more important, we can turn to more temporally fine-grained signals such as EEG to see if there are any special activities during that time. To confirm that the attributions we get are valid and consistent, we perform a sanity check of IG results on two overlapped inputs with an offset  $\tau$ : the first input is obtained from window  $[t_0, t_0 + T]$  and the second is obtained from window  $[t_0 + \tau, t_0 + \tau + T]$ . Offset-aligned results are shown in fig. 3.10a, in which the attributions agree with each other quite well.

## II. Spatial importance

We examine the connection importance between brain ROIs by looking at  $\text{ATTR}_A$ . In particular, columns in  $\text{ATTR}_A$  with higher average values are sender ROIs of high-



**Figure 3.10:** (a) Temporal importance sanity check of IG results on two pieces of inputs with a large overlap period. Attribution maps are offset-aligned. (b)  $\text{ATTR}_X$  distributions across 17 brain subnetworks (defined as in [1]) for VWM.

contributing connections, which is what matters in the GNN operation. We also explore why using random graph adjacency matrices (setting (v) in section 3.2.3) can produce a similar result for length-256 inputs compared to using both SC-induced  $A_i$  and  $A_{i\_ladp}$  (setting (i)). By examining  $\text{ATTR}_A$  under both settings (fig. 3.11), we see that the column averages of  $\text{ATTR}_A$  under these two settings are similar for almost all tasks, meaning the model can learn the important signal sending regions relatively well even without explicit structures. We credit this ability primarily to multi-resolution inner cluster smoothing, as the performance drops notably without it (setting (vii)). However, using ground truth SC not only gives us higher performance for shorter inputs but also provides the opportunity to interpret brain region connections better. We can directly use task-averaged  $\text{ATTR}_A$  as the weighted adjacency matrix to plot edges between brain ROIs, just as in fig. 3.12. Important discriminatory brain regions obtained from  $\text{ATTR}_A$  mostly comply with the previous literature:

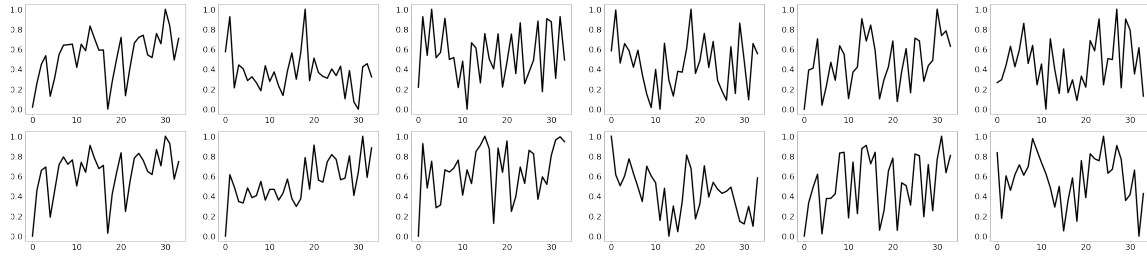
- Resting state: The top attributed ROIs belong to the default mode network, which

is regarded salient during the resting state [37].

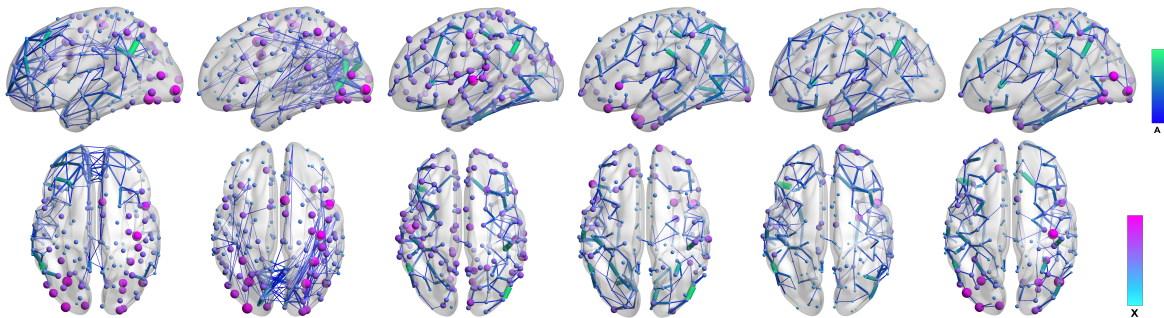
- VWM: The dominant attributions are from visual regions and posterior parietal regions, which complies with [72].
- DYN: Attributions from our model suggest regions along the cingulate gyrus (defaultA-SalValAttnB-ContA-ContC-defaultC), as well as peripheral visual and somatomotor regions. Literature suggests anterior cingulate cortex (ACC) to be active [73] and posterior cingulate cortex (PCC) to be inactive [74] during visual attention tasks. This means both regions provide discriminative information about the DYN states, which is what our attribution method votes for.
- DOT: Important ROIs from our analysis are located in control networks, in particular both ACC and PCC, as well as in the peripheral visual system. In the literature, dorsal and rostral regions of the ACC are proven to be involved with dot-probe performance [75, 76].
- MOD: Our important ROIs are mostly in temporal-parietal regions and default mode network (anatomically frontoparietal), and literature suggests similar regions: parietal [77] and prefrontal [78].
- PVT: Our top attributed ROIs belong to control networks, attention networks, and somatomotor regions. This is similar to [79], where both attention and motor systems are considered important.

In addition to  $ATTR_A$ ,  $ATTR_X$  can also provide insights on spatial importance when the attribution maps are aggregated along the temporal dimension. But it does so from another perspective: based on how the model takes in the inputs, larger  $ATTR_A$  implies critical *structural connections* between brain regions, meaning that information passing





**Figure 3.11:** Column averages of task-averaged  $\text{ATTR}_A$  (mapped into 34 subnetworks defined by the 17-network parcellation with left, right hemispheres). The top row is obtained from real SC-induced  $A$  and the bottom row is obtained from random SC-induced  $A_{\text{rand}}$ . Attributions are normalized to  $[0, 1]$ . Tasks are: Rest, VWM, DYN, DOT, MOD, PVT from left to right.



**Figure 3.12:** ROI attributions from  $\text{ATTR}_A$  and  $\text{ATTR}_X$ . (Task order is the same as fig. 3.11). Edge color and width are based on task-averaged  $\text{ATTR}_A \in \mathbb{R}^{200 \times 200}$ , and node color and size are based on task and temporal-averaged  $\text{ATTR}_X \in \mathbb{R}^{200}$ . For visualization, only edges with highest attributions are shown (the resulting sparsity reduces to 0.009 from 0.196).

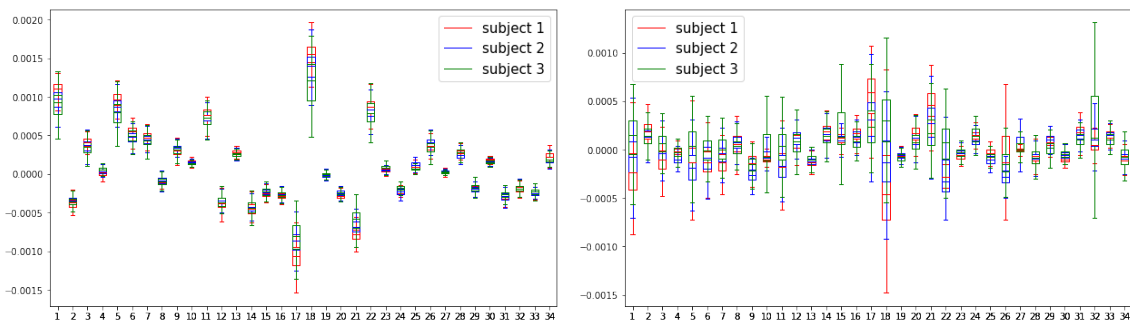
between those regions is deemed essential in classifying task states. In contrast, larger  $\text{ATTR}_X$  reveals regions or subnetworks that are sources of the important *signals*: it does not matter if the signal activities propagate from one region to another. Instead, the signals themselves are crucial for differentiating between task states. We notice that signal-important ROIs are not necessarily the same as connection-important ROIs: top-ranked subnetworks for resting state are DefaultA and DefaultB by  $\text{ATTR}_A$ , and VisCent and DorsAttnA by  $\text{ATTR}_X$ ; although they do coincide with each other for tasks like VMN. This disparity is reflected in fig. 3.12 as edge and node differences. Another observation

is that DYN and PVT have similar  $ATTR_A$  patterns; both have a high attribution on connections originating from visual, control, and somatomotor systems. But when looking at  $ATTR_X$ , DYN and PVT are extreme opposites. For example, PVT has a very high  $ATTR_X$  for a few ROIs in LH\_SomMotA, DorsAttnA\_TempOcc, and RH\_VisCent\_ExtStr, while DYN has very low  $ATTR_X$  for them. This suggests that the model uses these ROIs' activities to distinguish between the two tasks. Therefore, the attributions are not absolute but relative to what they are compared against. As a result, when identifying biomarkers with attribution, it is crucial to have *contrasts*—for example, different tasks, different disease states, etc.

In fig. 3.10b, we plot the distribution of time-averaged and subnetwork-averaged (mapping 200 ROIs into 17 subnetworks)  $ATTR_X$  during the VWM task. We can see the clear dominance of VisCent, DorsAttnA, and ContA subnetworks (numbered as 1, 5, 11), indicating signals from these regions are useful for the model to decide if the input is from the VWM task. More informative than the rankings is the distribution itself: even though VisCent, DorsAttnA, and ContA ranked top 3 for both resting state and VWM for signal attributions, their relative importance, and attribution distribution variances are drastically different. In a sense, the distribution can act as a task fingerprint based on brain signal states.

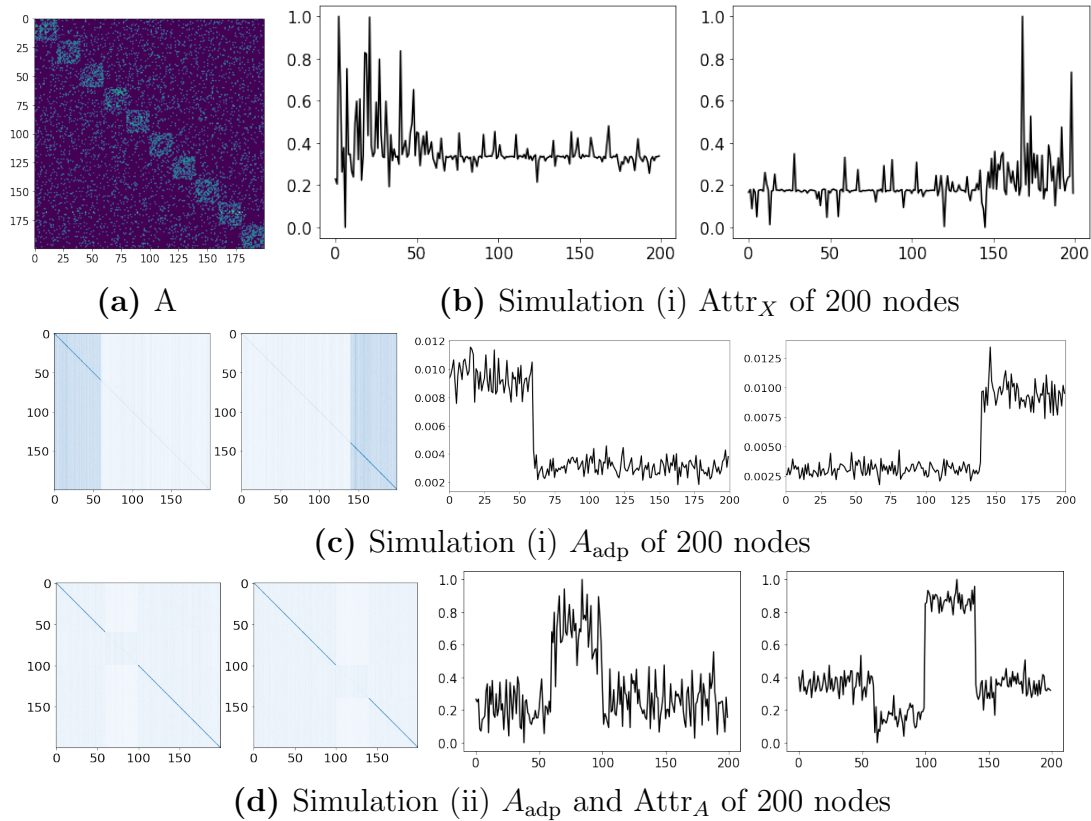
### III. Group, session, and region heterogeneity

Average variances of attributions are very different across tasks, especially those of  $ATTR_X$ : VWM and DYN have much smaller attribution variances compared to other tasks. This can be caused by either task dynamics when certain tasks have more phase transitions and brain status changes, or/and group heterogeneity when individuals carry out specific tasks more differently than others. We investigate this by examining three subjects that have multiple scan sessions for every task.



**Figure 3.13:** 34 subnetworks’  $\text{ATTR}_X$  distributions of 3 subjects performing the VWM task (left) and the MOD task (right). Outliers that go beyond  $[Q1 - 1.5 \text{ IQR}, Q3 + 1.5 \text{ IQR}]$  are omitted. VWM has a much smaller average attribution variance than MOD.

We report the following findings: (1) Even only aggregating attributions over a single subject’s sessions, attribution variances of the other four tasks are still larger than VWM and DYN. And these variance values are comparable to that of aggregating over many subjects. This means the large variances are not mainly due to group heterogeneity; rather, some tasks have more states than others. (2) There is still group heterogeneity apart from different task dynamics, and group heterogeneity is also more evident for tasks with more dynamics (high attribution variances). We can see from fig. 3.13 that attributions for VMM are much more concentrated and universal across subjects than that of MOD. (3) Flexibility of different subnetworks varies: subnetworks with small distribution IQR (interquartile range) of the same subject’s different sessions are also more consistent across subjects. One example is that subnetwork 18 during the MOD task has both higher within-subject IQR and more significant across-subject differences than subnetwork 19. This indicates that for a particular task, some subnetworks are more individual and flexible (may activate differently across time), while others are more collective and fixed. In summary, we can find both critical regions that a particular task must rely on and regions that can characterize individual differences during tasks.



**Figure 3.14:** (a) A typical adjacency matrix for simulated graph signals. (b) Task averaged  $\text{Attr}_X$  of simulation (i). Attribution values are normalized. (c) Task averaged  $A_{\text{adp}}$  of simulation (i) and its entry averages per column. (d) Task averaged  $A_{\text{adp}}$  and task averaged  $\text{Attr}_A$  of simulation (ii). Attribution values are normalized.

#### IV. Simulation study

To validate the results of our interpretations, we perform simulation studies with known ground truth. All graphs are generated with SBM (stochastic block model) using the same community structure (200 nodes, 10 communities), but each graph has its own adjacency matrix. This generation process mimics brain structures in that samples share similar community structures but have distinct structural connectivities. Fig. 3.14a shows a typical adjacency matrix of a synthetic graph. All adjacency matrices are binary. Time-series on each node are then generated with code adapted from pytorch-gnn

repository <sup>5</sup>. In particular, the value at each time step of each node is a small temporal Gaussian random noise plus signals from neighbors' (a small spatial Gaussian noise is added to the adjacency matrix) previous step.

**Simulation (i)** We create two classes for this simulation. In class one, only the first three communities (nodes 1–60) generate small temporal noises, and other nodes are only affected by neighbors. In class two, only the last three communities (nodes 141–200) generate small temporal noises, and other nodes are only affected by neighbors. We visualize the task aggregated  $\text{Attr}_X$  and  $A_{\text{adp}}$  and in figs. 3.14b and 3.14c. The signals are characterized well in  $\text{Attr}_X$ . For the generated series, signals are more important in node 1–60 for class 1 and 141–200 for class 2:  $A_{\text{adp}}$  finds this pattern and helps propagate signals in these regions better. We notice that  $\text{Attr}_A$  is mostly random, with no apparent patterns. This is consistent with the graph signal generation: when aggregating information from neighbors, all connected edges are weighted the same (binary); thus, the connections do not affect generated signals. We perform the following study to understand the opposite effect.

**Simulation (ii)** We again create two classes for the simulation: in class one, connections from nodes 61–100 are strengthened; in class two, connections from nodes 101–140 are strengthened. The weights of strengthened edges are increased from 1 to 5 during signal generation. However, the model still takes in binary adjacency matrices as inputs (processed as mentioned in section 3.2.1 before feeding to the model). We visualize the task aggregated  $A_{\text{adp}}$  and  $\text{Attr}_A$  in fig. 3.14d. This time the connection differences are reflected in  $\text{Attr}_A$ . Signals in node 61–100 for class 1 or 101–140 for class 2 are less important because stronger connections can send these signals out: this results in smaller

<sup>5</sup><https://github.com/alelab-upenn/graph-neural-networks>

values for corresponding columns in  $A_{\text{adp}}$ . Combined with the previous simulation results, this suggests that strong signal-sending regions or regions with weak connections that are over-reflected in the graph adjacency matrix tend to have higher  $A_{\text{adp}}$  values. In other words,  $A_{\text{adp}}$  complements both signals and connections to encode latent dynamics, while attributions obtained from IG are better at interpreting the modalities separately.

### 3.4 Conclusion

This chapter proposes ReBraiD, a high-performing and efficient graph neural network model that embeds both structural and dynamic functional signals for a more comprehensive representation of brain dynamics. To better capture latent structures, we propose sample-level adjacency matrix learning and multi-resolution inner cluster smoothing. Apart from quantitative results showing ReBraiD’s superiority in representing brain activities, we also leverage integrated gradients to attribute and interpret the importance of both spatial brain regions and temporal keyframes. The attribution also reveals heterogeneities among brain regions (or subnetworks), tasks, and individuals. These findings can potentially reveal new neural basis, biomarkers of tasks or brain disorders when combined with behavioral metrics. They can also enable more fine-grained temporal analysis around keyframes when combined with other imaging techniques and extend to different scientific domains with sample (subject) heterogeneity.

## Chapter 4

# Going Beyond Brain Modalities: Reconstructing Observed Complex Images from Brain Activities

Understanding how the brain encodes external stimuli and how these stimuli can be decoded from the measured brain activities are long-standing and challenging questions in neuroscience. In this chapter, we focus on reconstructing the complex image stimuli from fMRI (functional magnetic resonance imaging) signals, and will also briefly touch upon the encoding direction. Unlike previous works that reconstruct images with single objects or simple shapes, our work aims to reconstruct image stimuli that are rich in semantics, closer to everyday scenes, and can reveal more perspectives. However, data scarcity of fMRI datasets is the main obstacle to applying state-of-the-art deep learning models to this problem. We find that incorporating an additional text modality is beneficial for the reconstruction problem compared to directly translating brain signals to images. Therefore, the modalities involved in our method are: (i) voxel-level fMRI signals, (ii) observed images that trigger the brain signals, and (iii) textual description of the images.

To further address data scarcity, we leverage an aligned vision-language latent space pre-trained on massive datasets. Instead of training models from scratch to find a latent space shared by the three modalities, we encode fMRI signals into this pre-aligned latent space. Then, conditioned on embeddings in this space, we reconstruct images with a generative model. The reconstructed images from our pipeline balance both naturalness and fidelity: they are photo-realistic and capture the ground truth image contents well.

In the following sections, we first validate our main argument: incorporating text modality into brain visual encoding/decoding process is beneficial (section 4.1), then we present our brain decoding pipeline (section 4.2), and finally, briefly demonstrate that brain encoding process can be performed with the same central principle (section 4.3). Understanding the importance of semantic representation of human visual system can lead to many future research, and we discuss some of them in section 4.4.

## 4.1 Incorporating Additional Text Modality

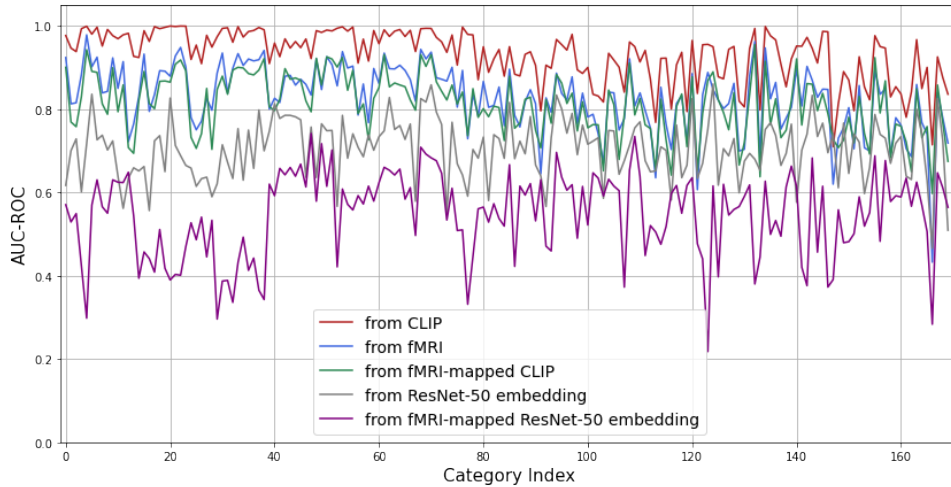
In an effort to understand visual encoding and decoding processes, researchers in recent years have curated multiple datasets recording fMRI signals while the subjects are viewing natural images [80, 81, 82, 83]. In particular, the Natural Scenes Dataset (NSD [80]) was built to meet the needs of data-hungry deep learning models, sampling at an unprecedented scale compared to all prior works while having the highest resolution and signal-to-noise ratio (SNR). In addition, all the images used in NSD are sampled from MS-COCO [84], which has far richer contextual information and more detailed annotations compared to datasets that are commonly used in other fMRI studies (e.g., Celeb A face dataset [85], ImageNet [86], self-curated symbols, grayscale datasets). This dataset, therefore, offers the opportunity to explore the decoding of complex images that are closer to real-life scenes.



Human visual decoding can be categorized into stimuli category classification [87], stimuli identification [88], and reconstruction. We focus on stimuli reconstruction in this study. Different from previous efforts in reconstructing images from fMRI [89, 90, 91, 92, 93, 82, 94], we approach the problem with one more modality, that of text. The benefits of adding the text modality are threefold: first, the brain is naturally multimodal. Research [95, 96, 97] indicates that the brain is not only capable of learning multisensory representations, but a larger portion of the cortex is engaged in multisensory processing: for example, both visual and tactile recognition of objects activate the same part of the object-responsive cortex [98]. Visual-linguistic pathways along the border of the occipital lobe [99] also bring a more intertwined view of the brain’s representation of these two modalities. Second, multimodal deep models tend to explain the brain better (having higher representation correlations) than the visual-only models, even when compared with activities in the visual cortex [100]. Lastly, our goal is to reconstruct complex images that have multiple objects in different categories with intricate interactions: it is natural to incorporate contextual information as an additional modality.

Instead of training a model to map all three modalities (fMRI, image, text) to a unified latent space, we propose to map fMRI to a well-aligned space shared by image and text, and use conditional generative models to reconstruct seen images from representations in that space. This design addresses the data scarcity issue of brain datasets by separating fMRI from the other two modalities. In this way, a large amount of data is readily available to learn the shared visual-language representation and to train a generative model conditioned on this representation. Furthermore, pre-trained models can be utilized to make the whole reconstruction pipeline more efficient and flexible.

Recent developments in contrastive models allow more accurate embeddings of images and their semantic meanings in the same latent space. This performance is realized using massive datasets: models such as CLIP [101] and ALIGN [102] utilize thousands of



**Figure 4.1:** Category-wise AUC-ROC of multi-label classifiers that predicts from four different signal/embedding sources. The first 80 categories are “things categories” and the last 91 are “stuff categories” in COCO.

**Table 4.1:** Numerical AUC-ROC values of the classifiers presented in fig. 4.1.

AUC-ROC	”things” categories	”stuff” categories	Overall	Performance w.r.t. fMRI (%)
CLIP	$0.9718 \pm 0.0266$	$0.8973 \pm 0.0639$	$0.9318 \pm 0.0624$	112.36
fMRI	$0.8704 \pm 0.0557$	$0.7937 \pm 0.0824$	$0.8293 \pm 0.0807$	100.00
fMRI-mapped CLIP	$0.8468 \pm 0.0604$	$0.7817 \pm 0.0733$	$0.8119 \pm 0.0748$	97.90
ResNet-50	$0.7061 \pm 0.0736$	$0.7032 \pm 0.0719$	$0.7044 \pm 0.0725$	84.94
fMRI-mapped ResNet-50	$0.5410 \pm 0.1106$	$0.5520 \pm 0.0941$	$0.5469 \pm 0.1020$	65.95

millions of image-text pairs for representation alignment. In comparison, brain imaging datasets that record pairs of images and fMRI range from 1.2k to 73k samples, making it difficult to learn brain encoding and decoding models from scratch. However, we can utilize aligned embeddings obtained from pre-trained contrastive models as the intermediary and generate images conditioned on these embeddings. In what follows, we show that utilizing rich semantic space, in particular, the CLIP space, better captures brain dynamics than the traditional image latent space.

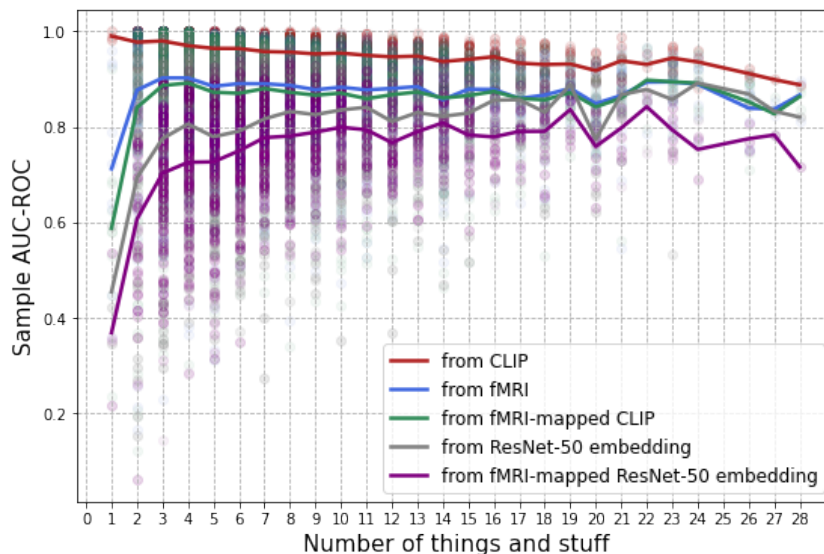
### 4.1.1 CLIP Space as the Intermediary

In this section, we show that multimodal embedding space, particularly the CLIP space, is beneficial for brain signal decoding. To this end, we trained a set of multi-label category classifiers to classify if a certain object category exists in the image based on the following inputs: (1) image-triggered fMRI  $\mathbf{x}_{\text{fmri}} \in \mathbb{R}^{15724}$ ; (2) image CLIP embeddings  $\mathbf{h}_{\text{img}} \in \mathbb{R}^{512}$ ; (3) CLIP embeddings mapped from image-triggered fMRI  $\mathbf{h}'_{\text{img}} \in \mathbb{R}^{512}$ ; (4) image ResNet embeddings  $\text{ResNet}(\mathbf{x}_{\text{img}}) \in \mathbb{R}^{2048}$  (obtained from Layer4, the final block before fully connected layers). All classifier models consist of 3 linear layers with ReLU activations in between, and finish with a Sigmoid activation. For fMRI signals, we use (2048, 512) as the hidden dimension; for CLIP embeddings (setting (2) and (3)), we use (384, 256) as hidden dimensions; and for the ResNet embedding, we use (512, 256) as hidden dimensions. The final output covers 171 classes, including 80 things categories (bounded objects, like “person”, “car”), and 91 stuff categories (mostly unbounded objects, like “tree”, “snow”).<sup>1</sup> Binary cross-entropy loss is used for each class to predict its existence in the input image.

Fig. 4.1 shows the category-wise AUC-ROC. The result demonstrates that CLIP embeddings contain the most object-level information about the image out of all the input sources. Following it, fMRI signals are also surprisingly very predictive, considering they carry a lot of noise. The performance discrepancy between settings (2) and (3) is minimal, meaning mapping fMRI signals into the CLIP space retains most of the fMRI signals’ information: this provides strong support for the validity of our design. Lastly, ResNet embeddings perform poorly compared with other input sources. Therefore, even with a perfect mapping model, projecting fMRI signals into this space will lose information about the image since the expressiveness of the embedding is bounded by the lower

---

<sup>1</sup>Please refer to <https://github.com/nightrone/cocostuff/blob/master/labels.txt> for the full category list.



**Figure 4.2:** Sample-wise AUC-ROC of multi-label classifiers that predicts from five different signal/embedding sources as the number of samples in stimulus images increases.

performer. In addition, we note that both CLIP embeddings and fMRI have poorer performance on stuff categories than on things categories, whereas ResNet embeddings do not. This can indicate brain signals align better with the multimodal CLIP space than with single-modality ResNet space. In addition, when the number of objects in the image increases, per-sample classification performance using CLIP, fMRI, and fMRI-mapped CLIP vector as inputs gradually decreases (the only difference is the single-object case). In contrast, ResNet inputs do not exhibit this property (fig. 4.2). We hypothesize that CLIP vectors can better mimic the cognitive overload when the scene becomes more crowded. Previous brain signal decoding work utilizing pre-trained generators all relied on image-only embedding spaces (ResNet-50 [92], VGG19 [94]), and we believe moving to a multimodal latent space is a crucial step towards better brain signal decodings. Similar to our conclusion here, previous evidence also suggests that the brain represents thousands of object categories by organizing them into a continuous semantic similarity space that is mapped systematically across the visual cortex [103], and a very recent

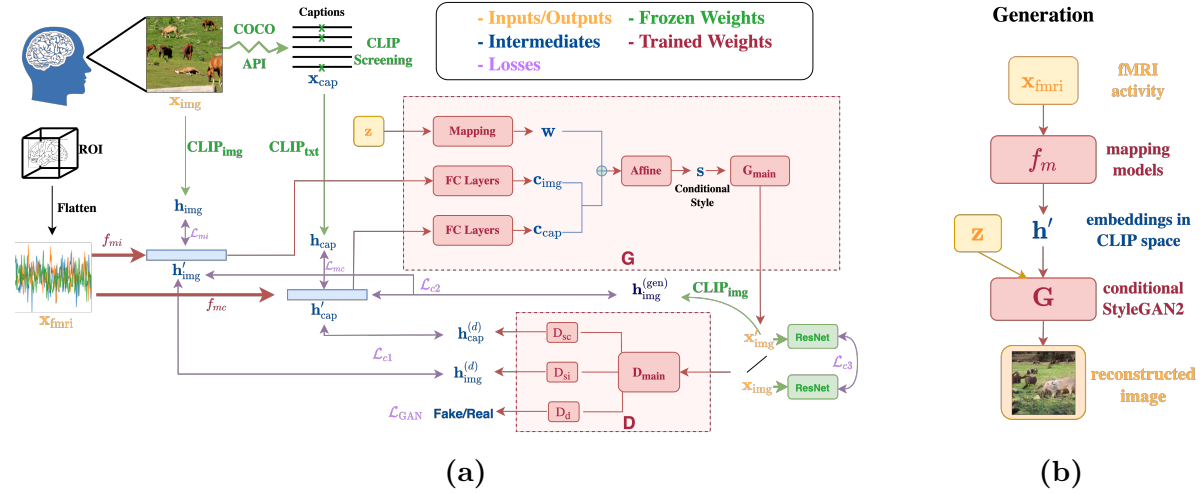
work proposes that a central objective of the visual system is transforming visual input into rich semantic scene descriptions [104].

## 4.2 Brain Decoding

To the best of our knowledge, this is the first work on reconstructing complex images from human brain signals. It provides an opportunity to study the brain’s visual decoding in a more natural setting than using object-centered images. Compared to previous works, it also decodes signals from more voxels and regions, including those outside the visual cortex, that are responsive to the experiment. This inclusion allows us to study the behavior and functionality of more brain areas. We address the data scarcity issue by incorporating additional text modality and leveraging pre-trained latent space and models. For the reconstruction, we focus on semantic representations of the images while taking low-level visual features into account. Our results show we can decode complex images from fMRI signals relatively faithfully. We also perform microstimulation on different brain regions to study their properties and showcase the potential usages of the pipeline.

### 4.2.1 Method


Our overall pipeline is shown in fig. 4.3. We use pre-trained CLIP image and text encoders to provide the representations of image stimuli and their captions. we first map fMRI signals to these CLIP embeddings, then pass these vectors to a conditional generative model for image reconstruction.



**Figure 4.3:** The pipeline for reconstructing seen images from fMRI signals. (a) details different components, from collected data to the reconstructed image. The pipeline is trained in two stages: during the first stage, mapping models  $f_{mi}$ ,  $f_{mc}$  are trained to encode fMRI activities into the CLIP embedding space. In the second stage, conditional generator  $G$  and contrastive discriminator  $D$  are finetuned while both  $f_{mi}$ ,  $f_{mc}$  are kept frozen. (b) shows the image generation process once models are trained.

## I. Caption screening

Each image  $x_{img}$  in the COCO dataset has five captions  $\{x_{cap_1}, \dots, x_{cap_5}\}$  collected through Amazon’s Mechanical Turk (AMT), and in nature, these captions vary in their descriptive ability. Fig. 4.4 shows a sample image with its five captions, and we can tell captions (2) and (3) are more objective and informative than caption (4) when it comes to describing the *content* of that image, thus are more helpful to serve as the image generation condition. We utilize pre-trained CLIP encoders to screen the high-quality captions since representations in the CLIP space are trained to be image-text aligned. A caption with an embedding more aligned to the image embedding is more descriptive than a less aligned one; it is also less general and more specific to this particular image because of the contrastive loss in CLIP. For the screening, we pass each image together with its five captions to the CLIP model, which outputs corresponding probabilities that the captions and image are proper pairs. We keep captions with probabilities larger than



Captions	CLIP probabilities	
(1) A group of people sitting and standing on top of a sandy beach.	0.0376	
(2) A surfboard rests on the beach while people play in the waves.	0.519	✓
(3) A surfboard on the sand and people on the beach behind.	0.4302	✓
(4) A few people are hanging out and appreciating their time.	0.001286	
(5) A group of people are sitting on the beach shore.	0.0122	

**Figure 4.4:** Image caption screening through CLIP encoders. For this sample, threshold is put at half of the largest probability:  $0.5 \times 0.519$ . Therefore, captions (2) and (3) of the image are kept.

half of the highest probability. After screening out less informative captions, we have one to three high-quality captions per image.

## II. Mapping fMRI signals to CLIP space

Each fMRI signal that reflects a specific image is a 3D data volume, and the value on position  $(i, j, k)$  is the relative brain activation on this voxel triggered by the image. We apply an ROI (region of interest) mask on this 3D volume to extract signals of cortical voxels that are task-related and have good SNRs. The signal is then flattened into a 1D vector and voxel-wise standardized within each scan session. The end results  $\mathbf{x}_{\text{fmri}}$  are used by our image reconstruction pipeline. We choose to use the ROI with the widest region coverage, and the length  $N$  of  $\mathbf{x}_{\text{fmri}}$  ranges from 12682 to 17907 for different brains in the NSD dataset.

Our goal is to train two mapping models,  $f_{mi}$  and  $f_{mc}$  in fig. 4.3 (collectively denoted as  $f_m$ ), that encodes  $\mathbf{x}_{\text{fmri}} \in \mathbb{R}^N$  to  $\mathbf{h}_{\text{img}} = C_{\text{img}}(\mathbf{x}_{\text{img}}) \in \mathbb{R}^{512}$  and  $\mathbf{h}_{\text{cap}} = C_{\text{txt}}(\mathbf{x}_{\text{cap}}) \in \mathbb{R}^{512}$  respectively. Here  $C_{\text{img}}$ ,  $C_{\text{txt}}$  are CLIP image and text encoders, and  $\mathbf{x}_{\text{cap}}$  is one of the image captions chosen randomly from the vetted caption pool. We construct both  $f_m$  as a CNN with one Conv1D layer followed by four residual blocks and three linear layers. The training objective is a combination of MSE loss, cosine similarity loss, and contrastive loss on cosine similarity. We use the infoNCE definition [105] of contrastive loss, for the

$i^{\text{th}}$  sample in a batch of size  $B$ :

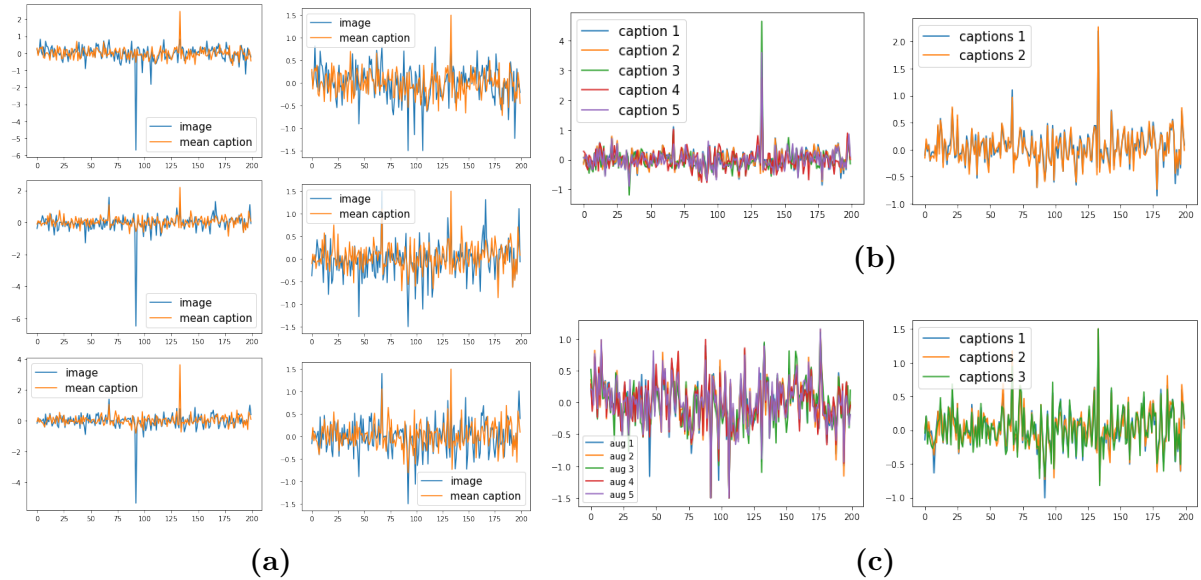
$$\text{Contra}(a^{(i)}, b^{(i)}) = -\mathbb{E}_i \left[ \log \frac{\exp(\cos(a^{(i)}, b^{(i)})/\tau)}{\sum_{j=1}^B \exp(\cos(a^{(i)}, b^{(j)})/\tau)} \right] \quad (4.1)$$

For the mapping model  $f_{mi}$  that encodes fMRI to image embeddings, we have  $\mathbf{h}_{\text{img}}^{(i)'} = f_{mi}(\mathbf{x}_{\text{fmri}}^{(i)})$ . The training objective is:

$$\mathcal{L}_{mi} = \mathbb{E}_i \left[ \alpha_1 \|\mathbf{h}_{\text{img}}^{(i)'} - \mathbf{h}_{\text{img}}^{(i)}\|_2^2 + \alpha_2 (1 - \cos(\mathbf{h}_{\text{img}}^{(i)'}, \mathbf{h}_{\text{img}}^{(i)})) \right] + \alpha_3 \text{Contra}(\mathbf{h}_{\text{img}}^{(i)'}, \mathbf{h}_{\text{img}}^{(i)}), \quad (4.2)$$

where  $\tau, \alpha_1, \alpha_2, \alpha_3$  are non-negative hyperparameters selected through sweeps. The loss  $\mathcal{L}_{mc}$  for caption embedding mapping model  $f_{mc}$  is defined similarly. Although CLIP embeddings are trained to be aligned, there are still systematic differences between image and text embeddings, with embeddings under each modality showing outlier values at a few fixed positions. In addition, we also notice the generated images emphasize either image content (object proximity, shape, etc.) or semantic features depending on which condition we use. Therefore, including both embeddings as the conditions for a generator can cover both ends, and that is why we train two mapping models for the two modalities. Since the outlier indices are fixed for each modality across images, clipping the value should not affect image-specific information. Therefore, before normalizing the ground truth embeddings into unit vectors, we set  $\mathbf{h} = \text{clamp}(\mathbf{h}, -1.5, 1.5)$ . This can greatly improve the mapping performance during training. See fig. 4.5 for the visualizations of CLIP embeddings that show image and text embedding differences, effect of thresholding (as well as effects of image augmentation and random caption selection that will be discussed later).





**Figure 4.5:** CLIP vector visualizations and thresholding. **(a)** before (left column) v.s. after (right column) thresholding at  $\pm 1.5$  to remove outliers. There are **systematic differences between CLIP image embeddings** and text embeddings; the outliers typically occur at the same positions for each modality. **(b)** the caption screening process can make the kept caption embeddings more aligned. (b)1 and (b)2 are from the same sample, only difference is the screening process. **(c)** (thresholded) embeddings of the same image with different augmentations; embeddings of same image’s different screened captions. All embeddings are shown the first 200 values for visualization purposes.

### III. Image reconstruction with CLIP embedding conditioning

The mapping models output fMRI-mapped CLIP embeddings  $\mathbf{h}'_{\text{img}}$  and  $\mathbf{h}'_{\text{cap}}$  that serve as conditions for the generative model. We aim to generate images that have both naturalness (being photo-realistic) and high fidelity (can faithfully reflect objects and relationships in the observed image). Our generation model is built upon Lafite [106], a text-to-image generation model: it adapts unconditional StyleGAN2 [107, 108] to conditional image generation contexted on CLIP text embeddings.

In our generator  $\mathbf{G}$ , both conditions  $\mathbf{h}'_{\text{img}}$  and  $\mathbf{h}'_{\text{cap}}$  are injected into the StyleSpace: each of them goes through two fully connected (FC) layers and is transformed into condition codes  $\mathbf{c}_{\text{img}}$  and  $\mathbf{c}_{\text{cap}}$ . These condition codes are max-pooled and then concatenated

with the intermediate latent code  $\mathbf{w} \in \mathcal{W}$ , which is obtained from passing the noise vector  $\mathbf{z} \in \mathcal{Z}$  through a mapping network (see fig. 4.3). Using a mapping network to transform  $\mathbf{z}$  into an intermediate latent space  $\mathcal{W}$  is the key of StyleGAN as  $\mathcal{W}$  is shown to be much less entangled than  $\mathcal{Z}$  [109]. The conditioned style  $\mathbf{s}$  is then passed to different layers of  $\mathbf{G}$  as in StyleGAN2, generating image  $\mathbf{x}_{\text{img}}'$ :

$$\mathbf{s} = \mathbf{w} \parallel \max(\mathbf{c}_{\text{img}}, \mathbf{c}_{\text{cap}}), \quad \mathbf{x}_{\text{img}}' = \mathbf{G}(\mathbf{s}). \quad (4.3)$$

We align the semantics of generated  $\mathbf{x}_{\text{img}}'$  and condition vectors by passing  $\mathbf{x}_{\text{img}}'$  through pre-trained CLIP encoders and apply contrastive loss (eq. (4.5)  $\mathcal{L}_{c2}$ ) between them. For further alignment of the lower-level visual features, such as prominent edges, corners and shapes, we also pass the image through resnet50 and align the position-wise averaged representation obtained from Layer2 (eq. (4.5)  $\mathcal{L}_{c3}$ ).

The discriminator  $\mathbf{D}$  has three heads that share a common backbone: the first head  $\mathbf{D}_d$  classifies images to be real/fake, the second and the third semantic projection heads  $\mathbf{D}_{si}$ ,  $\mathbf{D}_{sc}$  map  $\mathbf{x}_{\text{img}}'$  to  $\mathbf{h}'_{\text{img}}$  and  $\mathbf{h}'_{\text{cap}}$ . The latter two ensure the generated images are faithful to the conditions. It is also shown that contrastive discriminators are useful for preventing discriminator overfitting and improving the final model performance [110, 111]. Applying contrastive loss (eq. (4.5)  $\mathcal{L}_{c1}$ ) between the outputs from discriminator semantic projection heads and the condition vectors fed to  $\mathbf{G}$  can therefore help stabilize the training. To summarize the objective function, the standard GAN loss is used to ensure the naturalness of generated  $\mathbf{x}_{\text{img}}'$ :

$$\begin{aligned} \mathcal{L}_{\text{GAN}_{\mathbf{G}}} &= -\mathbb{E}_i \left[ \log \sigma(\mathbf{D}_d(\mathbf{x}_{\text{img}}^{(i) \prime})) \right], \\ \mathcal{L}_{\text{GAN}_{\mathbf{D}}} &= -\mathbb{E}_i \left[ \log \sigma(\mathbf{D}_d(\mathbf{x}_{\text{img}}^{(i)})) - \log(1 - \sigma(\mathbf{D}_d(\mathbf{x}_{\text{img}}^{(i) \prime})) \right], \end{aligned} \quad (4.4)$$

where  $\sigma$  denotes the Sigmoid function. Meanwhile, contrastive losses are used to align the

semantics of generated images and the fMRI-mapped condition vectors that supposedly reside in the CLIP space:

$$\begin{aligned}
 \mathcal{L}_{c1} &= \text{Contra}(\mathbf{D}_{sc}(\mathbf{x}_{\text{img}}^{(i)'}), \mathbf{h}_{\text{cap}}^{(i)'}) + \text{Contra}(\mathbf{D}_{si}(\mathbf{x}_{\text{img}}^{(i)'}), \mathbf{h}_{\text{img}}^{(i)'}), \\
 \mathcal{L}_{c2} &= \text{Contra}(\mathbf{C}_{\text{img}}(\mathbf{x}_{\text{img}}^{(i)'}), \mathbf{h}_{\text{cap}}^{(i)'}) + \text{Contra}(\mathbf{C}_{\text{img}}(\mathbf{x}_{\text{img}}^{(i)'}), \mathbf{h}_{\text{img}}^{(i)'}), \\
 \mathcal{L}_{c3} &= \text{Contra}(\text{ResNet}(\mathbf{x}_{\text{img}}^{(i)'}), \text{ResNet}(\mathbf{x}_{\text{img}}^{(i)}))
 \end{aligned} \tag{4.5}$$

The overall training objectives are:

$$\mathcal{L}_{\mathbf{G}} = \mathcal{L}_{\text{GAN}_{\mathbf{G}}} + \lambda_1 \mathcal{L}_{c1} + \lambda_2 \mathcal{L}_{c2} + \lambda_3 \mathcal{L}_{c3}, \mathcal{L}_{\mathbf{D}} = \mathcal{L}_{\text{GAN}_{\mathbf{D}}} + \lambda_1 \mathcal{L}_{c1},$$

where  $\lambda_1, \lambda_2, \lambda_3$ , are non-negative hyperparameters.

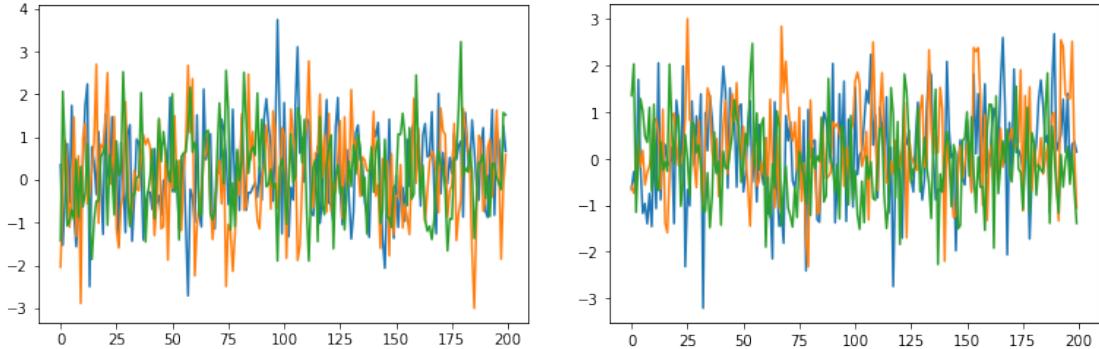
The whole generation pipeline, consisting of mapping models and GAN, is trained in two stages. First, mapping models  $f_{mi}$  and  $f_{mc}$  are trained on fMRI-CLIP embedding pairs. Next, starting from the trained mapping model weights and Lafite language-free model weights, we modify the losses and model structure and finetune the conditional generator. For the additional condition vector projection layers in  $\mathbf{G}$  and semantic head in  $\mathbf{D}$ , we duplicate the weights in the existing parallel layers to make the model converge faster. Note that Lafite is pre-trained on the Google Conceptual Captions 3M dataset [112] then finetuned on the MS-COCO dataset, both of which are much larger than NSD. Finetuning from it allows us to exploit the natural relationships between semantics and images with sparse fMRI data. We can still utilize a two-stage training to compensate for data scarcity even if no pre-trained conditional GAN like Lafite is available, for example, when using a different generator architecture. Only this time, we should firstly train the conditional GAN on a large image dataset with noise perturbed  $\mathbf{h}_{\text{img}}$  and  $\mathbf{h}_{\text{cap}}$  as the pseudo input condition vectors.

## 4.2.2 Results

### I. Data and experimental setup

The NSD data is collected from eight subjects. We focus on reconstructing observed scenes from a single subject’s brain signals. The reasons are twofold: first, it is more accurate to utilize individual brain coordinates instead of mapping voxels into a shared space, which can result in information loss during the process. More importantly, brain encoding and perception are different among individuals. This project aims to get the best reconstruction for a single individual, thus training models on one subject’s data. Nevertheless, the commonality of this encoding process among the population is an exciting topic for future explorations.

We use subject one from NSD: the available data contains 27750 fMRI-image sample pairs on 9841 images. Each image repeats up to three times during the same or different scan sessions. Note that brain responses to the same image can differ drastically during the repeats (fig. 4.6). Although we use *activities* and *signals* interchangeably throughout the chapter, what we mean are fMRI *betas* in the NSD dataset. Betas are not direct measurements of BOLD (blood-oxygenation-level dependent) changes, but the inferred activities from BOLD signals through GLM (general linear models). The reason for using betas instead of direct measurements is that image stimuli are shown consecutively to the subjects without prolonged delay, and activities triggered by the previous image can interfere with the next one if there is no proper separation. Authors of NSD proved the effectiveness of their GLM approach with much improved SNR in the betas over raw measurements [80]. The dataset is split image-wise: 23715 samples corresponding to 8364 images are used as the train set, and 4035 samples corresponding to the remaining 1477 images are used as the validation set. Therefore, our pipeline never sees the image it will be tested on during the training. We use 1pt8mm-resolution scans and only consider



**Figure 4.6:** fMRI activities responding to two images, each repeating three times. The figure only shows the activities of the first 200 voxels for visualization purposes.

fMRI signals from voxels in the nsdgeneral ROI provided by the dataset. This ROI covers voxels responsive to the NSD experiment (voxels with high SNR) in the posterior aspect of the cortex, and contains 15724 voxels for subject one ( $\mathbf{x}_{\text{fmri}} \in \mathbb{R}^{15724}$ ). Images are all scaled to  $256 \times 256$ .

Additional experiment settings, including hyperparameters of two training phases, are provided in appendix A.2. Our experiments are conducted on one Tesla V100 GPU and one Tesla T4 GPU. The code is publicly available.<sup>2</sup>

## II. Mapping models from fMRI to CLIP embeddings

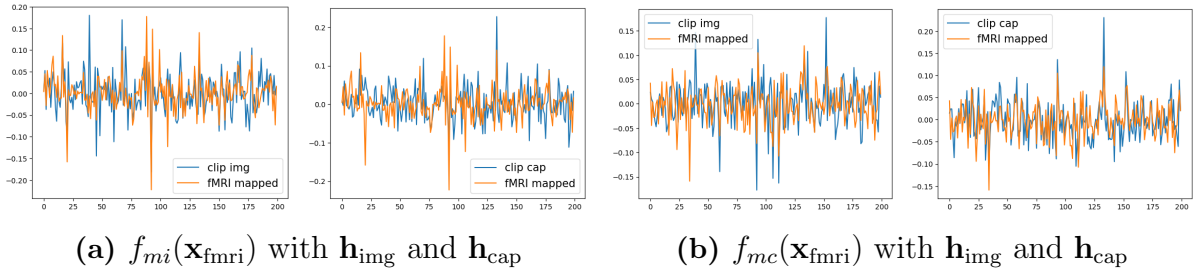
**Evaluation criteria** In the first training stage, mapping models  $f_{mi}$  and  $f_{mc}$  are trained to encode fMRI signals to CLIP embeddings. We use two criteria to evaluate the mapper performance to decide which one to use in the next stage. The first criterion is FID (Fréchet Inception Distance) [113] between generated image and ground truth using the trained mapper and a pre-trained generator. Given a Lafite model pre-trained on MS-COCO (language-free setting), we can replace its conditional vector with the outputs of our mapping models to generate images conditionally. These FIDs can indicate the starting points of the finetuning processes: the lower the FID, the better

<sup>2</sup><https://github.com/sklin93/mind-reader>

the candidate model. Secondly, we use the success rate of image "retrieval" in a batch of size 300. For the  $i^{th}$  sample in the batch, if the cosine similarity between  $\mathbf{h}^{(i)'}$  and  $\mathbf{h}^{(i)}$  is larger compared to between  $\mathbf{h}^{(i)'}$  and  $\mathbf{h}^{(j)}$ ,  $j \neq i$ , then it counts as one successful forward retrieval. For backward retrieval, we count the number of correct matches of  $\mathbf{h}^{(i)}$  to all  $\mathbf{h}^{(j)'}$ .

**Configuration comparisons** We tested different configurations on the mapping models, including: (1) Whether to place the threshold at  $\pm 1.5$  as mentioned in section 4.2.1; (2) When training  $f_{mi}$ , whether to perform image augmentations (augmentation details are listed in appendix A.2) before passing images through the CLIP encoder; (3) When training  $f_{mc}$ , whether to use the CLIP text embedding of a fixed caption, a random valid caption, or use the average embedding of all valid captions; (4) Which loss function to use: MSE only, cos (cosine similarity) only, Contra only, MSE + cos, MSE + cos + Contra; (5) Whether auxiliary networks help. We tested adding an auxiliary discriminator with GAN loss, as well as adding auxiliary expander networks with VICReg loss [114].

We found: (1) Clamping ground truth embeddings significantly increase performance; (2) Using image augmentations increase  $f_{mi}$  performance. This further indicates CLIP embeddings are more semantic related; (3) For  $f_{mc}$ , selecting a random caption from the valid caption pool each time is better than using a fixed one or using the average embedding of all valid captions; (4) Using MSE + cos as the loss gives the best base models, but then finetune these base models with MSE + cos + Contra can further lower the starting FID for pipeline finetuning, making the training in the next stage converge faster; (5) Adding auxiliary networks and objectives will not improve the performance. In general, although  $\mathbf{h}_{cap}$  and  $\mathbf{h}_{img}$  are already relatively well aligned,  $f_{mc}$  can still map  $\mathbf{x}_{fmri}$  closer to  $\mathbf{h}_{cap}$  than  $\mathbf{h}_{img}$ , whereas  $f_{mi}$  maps  $\mathbf{x}_{fmri}$  to an embedding that is equally close to both, while being able to capture a few extreme values in  $\mathbf{h}_{img}$  (fig. 4.7). We



**Figure 4.7:** Embeddings mapped from fMRI signals overlay on ground truth CLIP embeddings. (a) shows the results of image embedding mapping model  $f_{mi}$ ; (b) shows the results of caption embedding mapping model  $f_{mc}$ . For visualization purposes, the figures only show the first 200 values of the length-512 vectors.



**Figure 4.8:** Mismatches are semantically close to the ground truth. Figure shows examples of incorrect matches  $j$  (red frame) in a batch of 300 in the validation set. For each ground truth image  $i$  (green frame), we pass it through CLIP encoder to get  $\mathbf{h}^{(i)}$  and through  $f_{mc}$  to get  $\mathbf{h}^{(i)'}$ . The shown incorrect ones are those images with  $\mathbf{h}^{(j)'}$ ,  $j \neq i$  that is closer to  $\mathbf{h}^{(i)}$  than  $\mathbf{h}^{(i)'}$ .

think this difference reflects that it is easier to map fMRI signals to a more semantic representation (from the text space) than to a visual one.

To verify fMRI-mapped embeddings  $\mathbf{h}'$  are semantically well aligned with ground truth CLIP embeddings, we examined the mismatches during the image retrieval. For four incorrect retrievals, fig. 4.8 shows which images'  $\mathbf{h}^{(j)'}$  are closer to the ground truth images'  $\mathbf{h}^{(i)}$  than  $\mathbf{h}^{(i)'}$ . Notably, these mismatches are semantically close to the ground truth images. This indicates that the mapping models can successfully map fMRI signals into a semantically disentangled space. Embeddings in this space are suitable for providing contexts to a conditional generative model. We also tested another mapping model

$f_{mr}$  that maps fMRI signals to representations obtained from resnet50 Layer2. Unlike the CLIP embedding space, the resnet vector encodes more lower-level visual features. We see a jump in the image retrieval rate when we combine the representations obtained from  $f_{mi}$ ,  $f_{mc}$  with  $f_{mr}$  (table 4.3). However, the generative model is difficult to train when taking in two conditions from distinct embedding spaces. Therefore, we add the low-level vision constraint into the contrastive loss  $\mathcal{L}_{c3}$  instead.

**Quantitative results** We list the numerical results of the summarized findings in table 4.2. Simply put, forward retrieval checks the correct match of "*which ground truth CLIP embedding is the closest to the fMRI-mapped one?*" while the backward retrieval checks "*which fMRI-mapped embedding is the closest to the ground truth CLIP one?*". When multiple losses are involved, we use hyperparameter settings as in appendix A.2.

Fig. 4.7 visualizes the mapping results of the best setting (models trained with threshold, image augmentation, use a random valid caption each time, pre-trained with MSE+cos loss then finetuned with MSE+cos+Contra loss).

Combining the mapped embeddings from multiple mappers boosts the retrieval performance, especially the backward one (as shown in table 4.3). To use multiple mapping models, we first calculate a  $B \times B$  batch similarity matrix between the mapped embeddings for each model. Then we combine the similarity matrices with a weighted sum (weights are obtained through grid search) and perform image retrievals based on this combined similarity matrix. The mapping model  $f_{mr}$  that encodes fMRI to ResNet embeddings has a correct forward retrieval 6 and backward retrieval 50. But when its similarity matrix is combined with mapped-CLIP embedding similarity matrices, the performance is far above that of both ResNet and CLIP embeddings.



**Table 4.2:** Starting FID without generator finetuning (pre-trained LF-Lafite is used here) and correct retrievals in a batch of size 300 using embeddings obtained from  $f_{mi}$  and  $f_{mc}$ . In the top table, models are trained with MSE+cos loss. In the bottom table, defaults are: with threshold, with image augmentation, using random caption. For the two options with auxiliary modules, the model is finetuned from MSE + cos model since training from scratch gives much worse results. FID evaluations are omitted if the retrieval performance of a setting is strictly worse than its competitors.

		threshold	no threshold	image aug	no image aug	fixed caption	random caption	average caption embedding
$f_{mi}$	FID ↓	73.46	—	73.46	—	n/a	n/a	n/a
	Retrieval (forward) ↑	21	13	21	19	n/a	n/a	n/a
	Retrieval (backward) ↑	49	25	49	46	n/a	n/a	n/a
$f_{mc}$	FID ↓	75.24	—	n/a	n/a	—	75.24	79.36
	Retrieval (forward) ↑	14	11	n/a	n/a	13	14	15
	Retrieval (backward) ↑	<b>64</b>	45	n/a	n/a	39	<b>64</b>	43

		MSE	cos	Contra	MSE + cos	MSE + cos + Contra (from scratch)	MSE + cos + Contra (from MSE + cos)	Auxiliary GAN	Auxiliary expander (VICReg)
$f_{mi}$	FID ↓	—	—	—	73.46	—	<b>68.14</b>	—	—
	Retrieval (forward) ↑	5	12	25	21	27	<b>29</b>	25	19
	Retrieval (backward) ↑	16	34	50	49	50	<b>51</b>	42	35
$f_{mc}$	FID ↓	—	—	—	75.24	—	<b>53.68</b>	—	—
	Retrieval (forward) ↑	4	10	27	14	30	<b>33</b>	24	9
	Retrieval (backward) ↑	19	31	42	<b>64</b>	43	45	38	37

### III. Conditional image generation

**Quantitative results** In the second training stage, we finetune the conditional StyleGAN2.<sup>3</sup> There is no standard metric to measure image reconstruction quality from fMRI signals for complex images. Since previous works focused on reconstructing simpler images, the metrics typically involve pixel-wise MSE or correlation measures. However, when it comes to complex images, it seems more reasonable to use a perceptual metric, such as FID, which is based on Inception V3 [115] activations and is widely used in GAN.

<sup>3</sup>Codes are adapted from <https://github.com/NVlabs/stylegan2-ada-pytorch>, <https://github.com/drboog/Lafite>

**Table 4.3:** Correct image retrievals in a batch of size 300 when combining different models.

Multiple models	$f_{mi} + f_{mc}$	$f_{mi} + f_{mc} + f_{mr}$
Retrieval (forward)	32	24
Retrieval (backward)	73	147

In addition to using FID as a metric, we also perform 2-way identification for images reconstructed by models under different settings, and n-way identification of generated images with  $n = 2, 5, 10, 50$  under the best setting (finetuned from LF, with  $\mathcal{L}_{c3}$ , with mapping models  $f_m$  frozen). For n-way identification, we reconstruct an image from the fMRI signal for each sample in the validation set. For each generated image, we compare it with a set of  $n$  randomly selected images, including the ground truth one. Then based on the cosine similarity of their Inception V3 embeddings (before FC layers, the length-2048 vector), we identify which image the generated one corresponds to. This process is repeated ten times because of the randomness of the n-sample selection. Different from FID that reflects the naturalness of the generated images, n-way identification accuracy demonstrates more of the fidelity and uniqueness of the generated images.

We perform the ablation studies on the pipeline to answer the following questions: (1) Which mapping model trained in stage one leads to the best final performance?  $f_{mc}$  or  $f_{mi}$  or using both? (2) Which pre-trained GAN leads to the best final performance? For this, we compare using Lafite pre-trained on either the langue-free (LF) setting or the fully supervised setting. (3) Whether including the contrastive loss  $\mathcal{L}_{c3}$  between lower-level visual features can further improve the performance of a semantic-based generative model? Finally, we tested (4) whether finetune the whole pipeline end-to-end or freezing the mapping models is better? The new mapping model losses are set to  $\mathcal{L}'_m = \mathcal{L}_m + \lambda_4 \mathcal{L}_{\text{GAN}_G}$  if trained end-to-end.

**Table 4.4:** FID of the pipeline under different settings.

FID↓			$f_{mi}$	$f_{mc}$	$f_{mi}$ & $f_{mc}$
from supervised	without $\mathcal{L}_{c3}$	$f_m$ frozen	37.75	41.51	—
from LF	without $\mathcal{L}_{c3}$	$f_m$ frozen	30.83	33.78	50.59
from LF	with $\mathcal{L}_{c3}$	$f_m$ frozen	<b>29.74</b>	33.35	49.47
from LF	with $\mathcal{L}_{c3}$	end to end	45.02	48.54	50.96

Results of FID are reported in table 4.4. We observed the following: (1) in terms of FID, using  $\mathbf{h}'_{img}$  obtained from  $f_{mi}$  as the generator condition is better than using the  $\mathbf{h}'_{cap}$  from  $f_{mc}$  or using two conditional heads. On the other hand,  $f_{mc}$  and the two-head setting achieve as good or even better performance as  $f_{mi}$  does in terms of n-way identification accuracy. In addition, if training time or resource is the concern, using two heads and pre-trained LF-Lafite with only condition feeding interface changes and cloned weights in the new branches can already give reasonably good results. (2) Training the pipeline on LF-Lafite is much better than on the fully supervised Lafite. This result is expected for the generator conditioned on  $\mathbf{h}'_{img}$  since the supervised version is conditioned on CLIP text embeddings. However, the same discrepancy exists for the generator conditioned on  $\mathbf{h}'_{cap}$ . This may reflect the flexibility of pre-trained generators to adapt to the slight changes in the embedding space. It also shows the crucial impact of a pre-trained model on final performance when training data is limited. (3) The addition of low-level visual feature constraint  $\mathcal{L}_{c3}$  is beneficial for the model performance, especially faithfulness. It also seems to have more effects on single-head models than the two-head one. (4) For the end-to-end pipeline training, we test performance with  $\lambda_4 = [0.1, 1, 10]$ , all of which give worse performance than keeping the mapping model weights frozen (reported values are from  $\lambda_4 = 1$ ). In particular, we found that  $\mathbf{h}'$  tends to collapse to having nonzero values at only a few positions if the mappers are finetuned together with GAN.

Results of n-way identification accuracy are reported in tables 4.5 and 4.6. The n-

**Table 4.5:** 2-way identifications accuracy of the pipeline under different settings.

accuracy (%)			$f_{mi}$	$f_{mc}$	$f_{mi} \& f_{mc}$
from supervised	without $\mathcal{L}_{c3}$	$f_m$ frozen	$72.6 \pm 6.14$	$68.6 \pm 5.22$	—
from LF	without $\mathcal{L}_{c3}$	$f_m$ frozen	$73.0 \pm 4.40$	$73.2 \pm 4.49$	$76.2 \pm 5.89$
<b>from LF</b>	<b>with <math>\mathcal{L}_{c3}</math></b>	<b><math>f_m</math> frozen</b>	$76.8 \pm 4.16$	<b><math>78.2 \pm 5.47</math></b>	$78.0 \pm 4.47$
from LF	with $\mathcal{L}_{c3}$	end to end	$51.4 \pm 5.59$	$50.8 \pm 5.43$	$50.2 \pm 5.31$

**Table 4.6:** n-way identification accuracy (%) with  $n = 2, 5, 10, 50$ .

$n$	2	5	10	50
$f_{mi}$	$76.8 \pm 4.16$	$55.2 \pm 3.23$	$41.9 \pm 6.09$	$24.9 \pm 3.98$
$f_{mc}$	<b><math>78.2 \pm 5.47</math></b>	$56.4 \pm 3.32$	$42.2 \pm 4.33$	$25.6 \pm 4.05$
$f_{mi} \& f_{mc}$	$78.0 \pm 4.47$	<b><math>57.3 \pm 3.63</math></b>	<b><math>44.0 \pm 6.05</math></b>	<b><math>25.8 \pm 3.82</math></b>

way identification accuracy of the two-head setting ( $f_{mi} \& f_{mc}$ ) is slightly better most of the time (table 4.6), followed by the caption-vector-conditioned setting, followed by the image-vector-conditioned setting. Note that when performing n-way identification, previous image reconstruction works are typically tested on a validation set that contains 50 images of 50 different categories [116]. However, there are multiple objects involved in each image in the complex images we aim to reconstruct; it is not straightforward to separate them into different categories and pick one from each. Therefore, we leave the validation set as is (1477 image-fMRI pairs in total), and there will be overlapping categories in it; for example, several images contain scenes of animals in a natural environment.

**Qualitative results** We show several generated images in fig. 4.9. Although the generator takes in both the noise vector  $\mathbf{z}$  and fMRI-mapped embeddings, the results vary much more with the latter condition, while  $\mathbf{z}$  only contributes to variations on some minor details. In general, the generated images capture both semantics and visual features relatively well, even on complex images containing interactions of multiple objects. Since



(a) Ground truth stimuli (top row) and generated images conditioned on fMRI (bottom row).

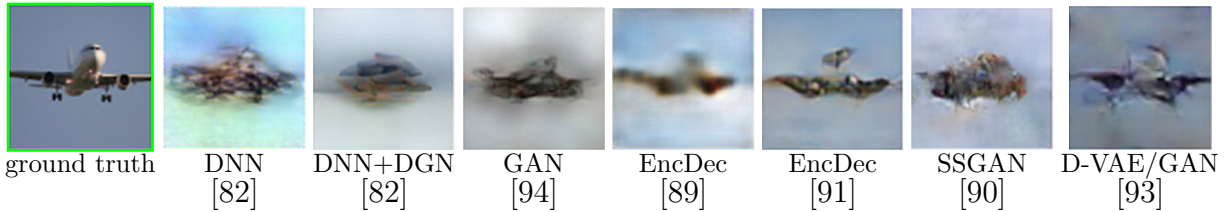


(b) Generated images from three different fMRI scans responding to the same stimulus (green frames).

**Figure 4.9:** Images generated by our pipeline given input fMRI signals.

each stimulus is repeated up to three times to the subject, we have multiple fMRI scans corresponding to the same image. The semantic differences in the generated images conditioned on these multiple scans could potentially reveal brain processing discrepancies of the same stimulus. For example, the three generations for the second image in fig. 4.9b emphasize respectively: (1) the overall scene and the fence, (2) people with green suits, and (3) overhead flags and the fence; these might reflect the variations in the subject’s attention or interpretations of that image. Eyetracking data can be further examined to study attention’s effect on generated images.

It is challenging to perform one-to-one comparisons with previous deep image reconstruction works since the images in the MS-COCO dataset have much higher complexities than artificial shapes, faces, or images containing a single centered object (like in ImageNet). We show results from a few best models for reconstructing images from fMRI in fig. 4.10a. There is also a recent survey [117] covering more models and results if readers are interested. As our dataset is different, we search for similar images in the NSD valida-



(a) Image reconstruction results from fMRI in previous works.

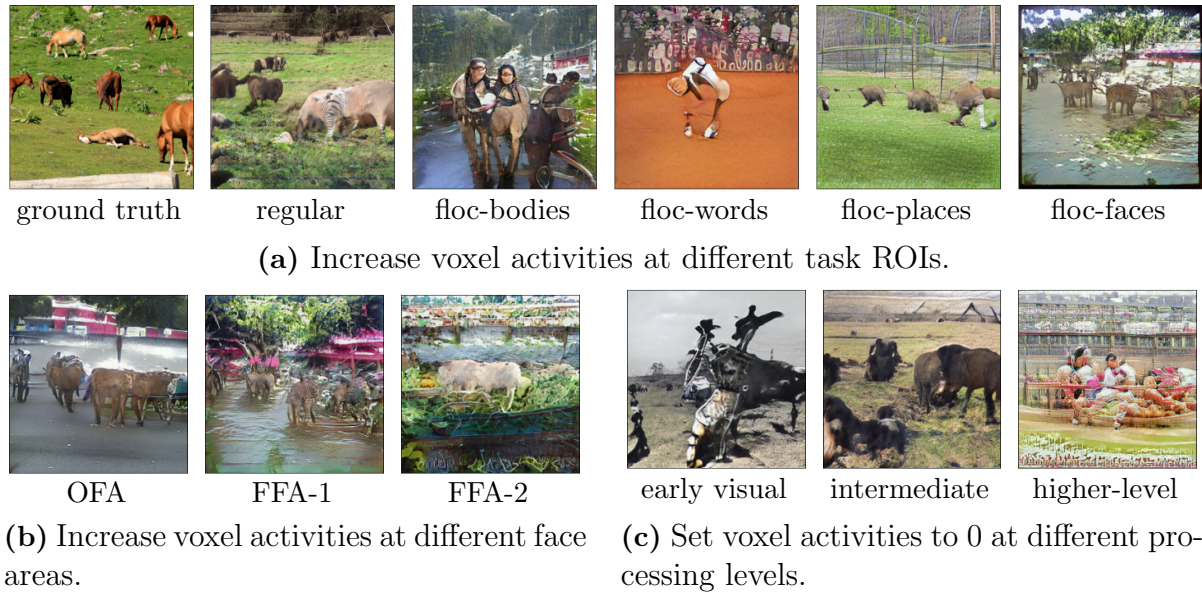


(b) Image reconstruction results from fMRI by our pipeline. Four ground truth images are green framed.

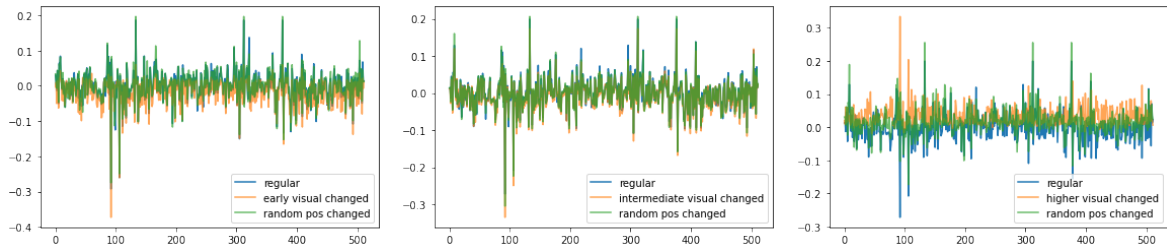
**Figure 4.10:** Comparisons between previous works and our pipeline. We are using the recent NSD dataset that involves more complex scenes. However, for comparison purposes, we choose four similar images from NSD, each containing a single object "plane", and show our reconstructions from fMRI signals in fig. 4.10b

tion set and show our generations in fig. 4.10b. Compared to other methods, our pipeline can generate more photo-realistic images that reflect objects' shapes and backgrounds well. It also utilizes more voxel activities than previous works (15724 voxels versus a few hundred). More importantly, it is able to reconstruct the relationships of different components when the images are more complex. As natural scenes around us are rarely isolated objects and always information-laden, we think reconstructing images through semantic alignment and conditioning is more beneficial and realistic than focusing on lower-level visual features.

**Microstimulation** In neuroscience, microstimulation refers to the electrical current-driven excitation of neurons and is used to identify the functional significance of a population of neurons. Here, we "microstimulate" the input fMRI signals of voxels in different brain ROIs, aiming to identify the roles of individual regions. In the NSD dataset, there are four floc (functional localizer) experiments targeting regions responsible for faces,



**Figure 4.11:** Images generated in microstimulation experiments. In (a)(b), voxel activities at multiple task ROIs are increased before passed into the pipeline. In (c), voxel activities at various visual processing stages are silenced.

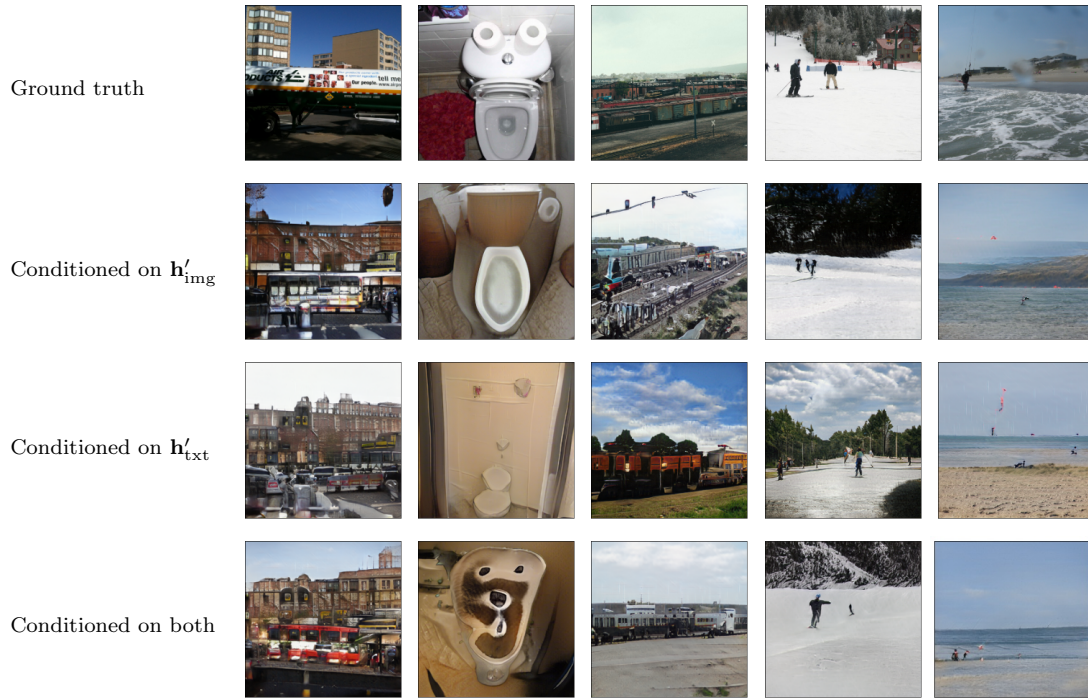


**Figure 4.12:** fMRI-mapped embeddings in the CLIP space ( $\mathbf{h}'$ ). Each figure contains (i) an embedding mapped from a regular fMRI signal, (ii) an embedding mapped from the fMRI signals with voxel activities in earlier-visual ROIs (left)/ intermediate ROIs (middle) / higher-level ROIs (right) set to zero, (iii) an embedding mapped from the fMRI signal with voxels at random positions (same number of voxels as (ii)) set to zero. Setting activities of the earlier-visual cortex to zero lowers overall embedding vector values, while setting activities of higher-level ROIs has the opposite effect. We can also perform the reverse masking: only keep voxel activities at earlier-level visual/ intermediate / higher-level ROIs, then the effects are reversed.

bodies, places, and words. A typical standardized fMRI signal has a value range around  $[-4, 4]$ . For the experiment, we locate the corresponding task-specific voxels based on ROI masks and increase the voxel activities to 10 while keeping the activities in unrelated voxels unchanged; results are shown in fig. 4.11. We observe the emergence of bodies or words when we increase the voxel activities in "bodies" or "words" ROIs. For voxels in "places" ROIs, elevating the signals will result in mesh-like patterns in the background, and this is true across different images. For "faces" ROIs, the generated images under elevated facial area signals seem to contain many small repeated patterns/perturbations. Interestingly, this appears to result from FFA (fusiform face area) signal changes since increasing only OFA (occipital face area) regions' activity does not result in similar patterns (fig. 4.11b). Overall, increasing a specific task ROI's signal across fMRI samples results in CLIP embedding changes in similar positions. This means the disentangled space of CLIP embedding aligns well with how the human brain processes visual cues.

Apart from task-specific ROIs, we also changed brain region activities based on their roles in the visual processing hierarchy. We use the streams mask in the dataset to identify early visual cortex ROIs, intermediate ROIs, and higher-level ROIs. We then zero out voxels at each level; sample results are shown in fig. 4.11c. Our observations are: (1) when silencing the early visual cortex, objects and the whole scene are prone to be in dull colors, and objects tend to have sharp shapes. Meanwhile, the mapped embedding in the CLIP space will constantly have a lower value at almost all positions compared to mapped from unchanged signals (fig. 4.12). (2) Silencing the higher-level ROIs has the opposite effect: more colors, more shapes, and crowded scenes. This is reasonable since the lower-level visual regions will bring up all the details when they lack high-level control. This time, the embeddings in the CLIP space have values consistently higher than normal. Finally, (3) silencing the intermediate ROIs seems to have the least visual impact or CLIP embedding changes among the three. We performed the





**Figure 4.13:** Generated images conditioned on fMRI-mapped CLIP image embedding  $\mathbf{h}'_{\text{img}}$ , fMRI-mapped CLIP text embedding  $\mathbf{h}'_{\text{txt}}$ , or both.

above microstimulation experiments on our pipeline with existing ROIs; however, it is potentially helpful for testing new ROI definitions and hypotheses.

**Additional examples** As mentioned in section 4.2.1, we found that the generated results conditioned on embeddings of different modalities tend to emphasize different aspects: more visual (colors, shape, etc.) if conditioned only on  $\mathbf{h}'_{\text{img}}$ , and more semantic if conditioned only on  $\mathbf{h}'_{\text{txt}}$ . This could reflect the slight difference between the latent space of the two modalities. We show the examples conditioned on either one of these two conditions, or both, in fig. 4.13.

The pipeline tends to fail under the following conditions: (1) the image only contains close-up details without too much semantic information; (2) the presented scene is semantically novel (e.g., a big banana-shaped decoration hanging in the middle of the



**Figure 4.14:** More examples showcasing model successes and failures. For each two-row group, the top row shows the ground truth images, and the bottom row shows the reconstructions.

room). The model also tends to: (3) generate based on data biases: adding windows to indoor scenes, adding people to food scenes, generating colored images when the inputs are black-and-white, etc.; (4) change or ignore the background; (5) Mix-up colors (assigning colors in the scene to a wrong object); (6) generate the wrong number of objects/people. We showcase these failures together with more other generated images in fig. 4.14. Given that the model is confident (in terms of GAN’s discriminator output staying at the same level) when generating results based on training data biases, future extensions should focus on exploring generators pre-trained on a much larger dataset, as discussed in section 4.4.3.

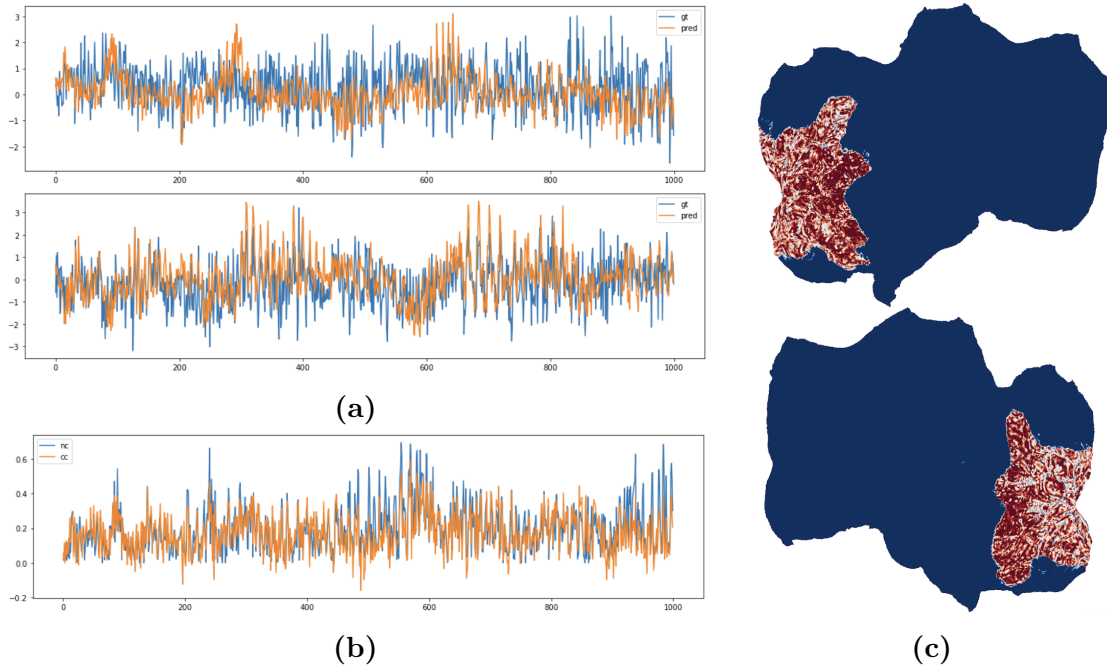
## 4.3 Brain Encoding and Encoding-Decoding Cycle

Although the primary goal of this chapter is decoding stimuli from brain activities, we also tested the encoding process with CLIP as the intermediate. In this section, we briefly present our results, as well as the complete encoding-decoding cycle.

### 4.3.1 Brain Encoding

Brain encoding is a problem that predicts brain activities from stimuli. It has a data scarcity problem similar to the decoding process. In addition, brain activities are intrinsically noisy and contain randomness, even when responding to the same stimulus. To this end, we solve the problem similar to the decoding process: the image stimuli are passed through pretrained CLIP encoders, obtaining CLIP embeddings  $\mathbf{h}_{\text{img}}$ . Then we train a mapping model that perform regression from  $\mathbf{h}_{\text{img}}$  to  $\mathbf{x}_{\text{fmri}}$ . The mapping model is also similar to  $f_m$ , consisting of four residual blocks, one transposed convolutional layer, two linear layers, and is trained with a combination of MSE and cosine similarity loss.

Fig. 4.15a shows the signal ground truth and predictions for the first 1000 voxels



**Figure 4.15:** Brain encoding results. **(a)** ground truth and prediction of two samples. Only the first 1000 voxels are shown for visualization purposes. **(b)** Voxel-wise performance (in terms of the correlation coefficient between ground truth and prediction) v.s. voxel noise ceiling. **(c)** Prediction performance on a flatmap, redder regions have more accurate predictions (accounted for the noise ceiling). Note we only perform prediction on the nsdgeneral ROI, thus the boundary.

of two samples. We also found that voxel-wise prediction (in terms of the correlation coefficient) aligns very well with the noise ceiling of that voxel (see fig. 4.15b).<sup>4</sup> However, there are discrepancies in this alignment: in fig. 4.15c, we visualize the voxel-wise prediction correlation coefficient ( $cc$ ) minus the voxel’s noise ceiling ( $nc$ ) as a flatmap. Here, redder areas correspond to better predictions, and the result shows that high-level semantic regions are better predicted than V1-V4. Utilizing latent spaces other than CLIP’s results in lower prediction performance and larger distance between  $cc$  and  $nc$ , as well as a more uniform performance among high-level regions and V1-V4.

<sup>4</sup>Noise ceiling values are calculated based on the method in the NSD data paper [80], utilizing SNR



**Figure 4.16:** Encoding-decoding cycle. The top row shows image stimuli; the second row shows predicted fMRI activities (with corresponding ground truth) by the encoding pipeline (only 300 voxels are shown for visualization purposes); the third row shows reconstructed images from predicted fMRI signals.

### 4.3.2 Complete Cycle

We tested the encoding-decoding cycle with trained encoding and decoding pipelines: ground truth images are fed to the encoding pipeline, which gives fMRI predictions. We then pass these predicted fMRI signals through the decoding pipeline to perform decoding. The results are shown in fig. 4.16. We observe that image semantic information is still relatively well conserved.

## 4.4 Discussions

### 4.4.1 Other Decoding Attempt

Prior to our current pipeline design, we experimented with a DALL-E-like structure [118] since we can view the image reconstruction problem as signal-to-signal translation.

In particular, we applied VQVAE [119] on both fMRI and image to represent them as discrete latent codes and train a Transformer model to autoregressively generate text and image tokens from fMRI tokens. However, it was challenging to train the Transformer-based model to converge with limited fMRI-image data. Incorporating the caption as text tokens to serve as the bottleneck between fMRI and image tokens while utilizing pre-trained models on the text and image modality did not help either. We think this suggests the need to introduce a semantic medium to avoid direct translations between fMRI and image, as well as a solution to data scarcity.

#### 4.4.2 Limitations and Future Work

We address both the rich-semantic and data scarcity issue with the semantic space of CLIP embeddings. First, CLIP space is semantically informative and visually descriptive: for example, we can use image-text CLIP embedding alignment probabilities to screen captions. Mapping fMRI signals to representations in this latent space will retain rich information about the image that needs to be reconstructed. Second, the pre-training of the generative model can be separated entirely from fMRI data, meaning it can utilize much larger datasets than the one we use. However, there is a trade-off between generating a semantically similar scene and faithfully reconstructing each pixel. Although trained with additional contrastive loss targeting low-level visual features, the generated images by our pipeline are still leaning towards the former. We consider this a reasonable choice since brains are more likely to perceive the image as a whole rather than identifying each pixel, especially with multiple objects in the scene. Nevertheless, this results in worse reconstructions for images with fine details but less semantic, such as single faces. The reconstruction of complex images with better aligned low-level visual features is worth further studies.



**Figure 4.17:** Generated images from interpolation of two fMRI scans. Step number is set to 10.

There are many more areas to explore. First, our study focuses on reconstructing a single subject’s brain signals. Applying the model to **different subjects** and observing the differences when generating the same image would be interesting. Since the data contains behavioral measures like valence and arousal towards each image, one can test if the generated images reflect personalized attention and perceptions. Second, **other latent spaces** can be examined. Although CLIP is one of the best-aligned computation models for the brain, other multimodal models like TSM [120] seem to have a better alignment [100] with the visual cortex. In addition, **other conditional generative models**, such as diffusion models, can be explored. In particular, DALL-E 2 [121] generates images conditioned on CLIP embeddings through diffusion, and it also provides an alternative solution to the differences exhibited in the image the text CLIP embeddings by learning a *Prior* model. Third, given the additional text modality, our pipeline opens up new opportunities to **study visual imagery** even without ground truth images. For example, one can either use mapping models trained on given fMRI-image pairs and pre-trained generators to reconstruct imagined scenes, or study the mapping between brain signals and the text embeddings of the mental images’ descriptions. Lastly, we mainly focus on the decoding (brain-to-image) process, but the **encoding (image-to-brain) process** of complex images is equally important and exciting, and much more can be explored than the brief coverage in section 4.3. The following paragraphs discuss three more possible future directions with more details.

#### **Input interpolations and the potential extension to movie reconstruction**

In addition to reconstructing observed images, we found utilizing the CLIP space can

also result in a smooth transition when decoding from interpolations of two fMRI scans (fig. 4.17). Combined with the ability to capture complex semantics, this pipeline can be helpful for movie reconstruction from brain signals. Temporal constraints can also be added, which could, in turn, benefit the reconstruction of each frame.

**Decoding text from fMRI** Apart from being the conditional vector for an image generator, CLIP embeddings can also be used to generate texts. To decode texts from fMRI data, the only change needed is replacing the conditional image generator in our pipeline with a text generator conditioned on CLIP vectors.<sup>5</sup> With this text pipeline, one can “define” the functions of each voxel through the following procedures: (1) provide a pseudo-fMRI activity to the pipeline with only the target voxel having non-zero activities, (2) generate fMRI-mapped CLIP embeddings  $\mathbf{h}'$  with the mapping models  $f_m$ , (3) provide  $\mathbf{h}'$  to the conditional text generator and get the text description of that voxel activity. An advantage of decoding the signals into the text form is that text is more straightforward than images in terms of explaining the semantics. This makes it easier to perform voxel clusterings and to find brain modules. The texts can also help understand which parts of the semantics are not mapped through from the  $f_m$  by comparing the ground truth captions and generated texts from the fMRI activities.

**Neural population control with synthetic images** With the encoding pipeline that we briefed in section 4.3, one can feed the pipeline with artificial images to test and understand how different shapes and semantics trigger voxels at various locations, thus having a better understanding of voxel functionalities. In addition, works similar to [122] can be tested by finding out which type of stimuli trigger a specific level of brain activity (e.g., higher activation) and then synthesizing images that control the neural population in the desired manner. Lastly, given pipelines of a complete cycle, images generated by

---

<sup>5</sup>An example CLIP-conditioned text decoder can be found here: <https://github.com/fkodom/clip-text-decoder>.





**Figure 4.18:** Image generated by Lafite pre-trained on the CC3M dataset without fine-tuning on COCO or NSD. Ground truth stimuli (top row) and generated images conditioned on fMRI (bottom row).

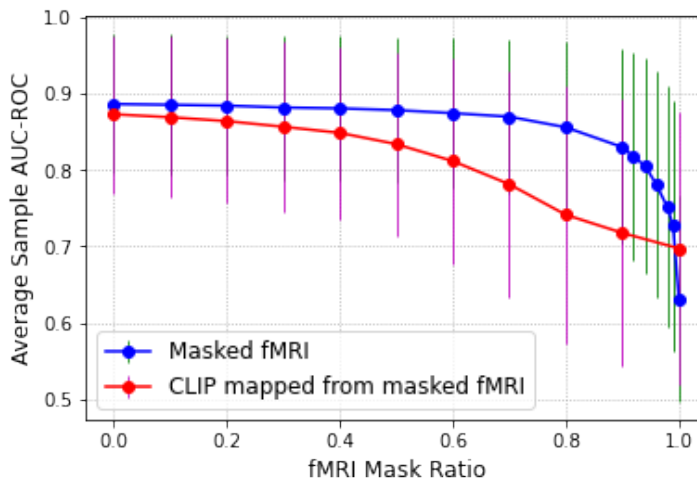
the decoder can also be benchmarked by passing them through the encoder.

### 4.4.3 Using Pre-trained Models

Our pipeline relies on two pre-trained components. The first and the most crucial one is the CLIP encoder that provides the latent space where we project fMRI signals. The second is a conditional GAN (Lafite) that generates images, which could be swapped for other generators. In what follows, we will discuss these two components separately.

**CLIP** One exciting aspect of CLIP is the size of its training dataset, which consists of 30 million Flickr images that should cover most of the natural image statistics. This coverage is also proved by subsequent works that generate images guided by CLIP embeddings through their abilities to perform generations in various styles. In addition, as we observed in section 4.1.1, CLIP embeddings can retain around 98% of object-level information in fMRI with a very well-aligned performance across categories.

Albeit its incredible expressive power, CLIP does have a much lower dimensionality than the original signal: no matter how faithful, it is a compression. By the nature of compression, CLIP only retains the most crucial information and removes most of the



**Figure 4.19:** Multi-label classifier (defined in section 4.1.1)’s average sample-wise AUC-ROC changes when masking input fMRI at different ratios. For a masked voxel, we set its value to 0.

redundancies in the original signal. Indeed, if we mask fMRI at different ratios, from 0 to 1, and perform the multi-label classification (the same task as in section 4.1.1) using (1) masked fMRI or (2) CLIP mapped from masked-fMRI, we will notice a very drastic difference in the performance drop rate. As shown in fig. 4.19, prediction performance from fMRI only drops drastically after the masking ratio becomes larger than 0.9, indicating brain redundancies to represent the objects. In contrast, if we map the masked fMRI into the CLIP space and use these embeddings for prediction, the performance drop is almost at a constant rate. This discrepancy makes the CLIP space more vulnerable to adversarial attacks than the fMRI space since a small change would cause the generated images to derail from the ground truth. In addition, CLIP embeddings also carry more biases than fMRI, as its mean AUC-ROC is much larger even with all-masked inputs. One should consider these traits of CLIP embeddings when applying this system and design defense mechanisms accordingly.

**Lafite** As for the generator, we utilize a conditional GAN pre-trained on the MS-COCO dataset (containing 328K images), from which NSD drew its experiment images.

This naturally provides an alignment in the data distribution. Although MS-COCO images are about everyday objects, humans, and scenes, the data statistics could vary when we move to other settings. Therefore, future studies are needed to extend current generators to one trained on broader sources (e.g., DALL-E 2, mentioned in section 4.4, used 650M images sampled from CLIP and DALL-E training data). This should minimize the dataset biases, although one should not interpret results without considering the training/testing discrepancies.

To show that our concept works across different generators, but dataset biases indeed play an important role, we test our pipeline with a Lafite pre-trained on the Google Conceptual Captions 3M dataset (CC3M, consisting of 3.3 million images) as the generator *without any extra finetuning*. We used our trained  $f_{mc}$  as the mapping function. The results are shown in fig. 4.18. All generated images have the watermark where CC3M sources its images. In addition, when trying to generate out-of-distribution images, the quality decreases in terms of photo-realism. Nonetheless, semantic alignments are still shown in these reconstructions. We also want to note that pre-trained models provide excellent bases for finetuning. For example, Lafite finetuned its COCO model on the CC3M model within three hours, compared to four days to reach the same performance if training from scratch. Therefore, if the pipeline is known to be used on certain types of images, a small-scale dataset and some light training should greatly help the model to fit into the desired data distribution.

**Societal impact** With current brain signal recording devices, the negative social impact of this work is minimal: portable devices like EEG have poor spatial resolutions, making them unlikely to provide enough image-related details; On the other hand, fMRI scanners are used under highly controlled settings with designed procedures, therefore unlikely to have subject-unapproved privacy violations. However, when new devices that can address these issues become readily available, regulations would be needed on col-

lecting and inspecting user data, since they potentially reveal sensitive information that users are unwilling to share through neural decoding. With pre-trained components, the pipeline may also misinterpret brain signals or be hacked to generate from manipulated inputs (no matter how unlikely it is) and produce over-confident false reconstructions because of the training data distributions. Several tricks may alleviate this issue, for example, training an input discriminator and placing it before the entire pipeline to filter out suspicious inputs. Or, using a parallel pipeline targeting pixel-level reconstruction as a check: if the two systems agree with each other above a certain threshold/confidence, the reconstruction results are accepted, otherwise discarded. Future pipeline improvements should also focus on exploring high-performing models pre-trained on large (thus more generalizable) and unbiased datasets.

## 4.5 Conclusion

This chapter proposes a pipeline to reconstruct complex images observed by subjects from their brain signals. With more objects and relationships presented in the image, we bring in an additional text modality to better capture the semantics. To achieve high performance with limited data, we utilize pre-trained semantic space that aligns visual and text modalities. We first encode fMRI signals to this visual-language latent space and use a generative model conditioned on the mapped embeddings to reconstruct the images. We also introduce additional contrastive loss to incorporate low-level visual features into this semantic-based pipeline. As a result, the reconstructed images by our method are both photo-realistic and, most of the time, can faithfully reflect the image content. This brain signal to image decoding pipeline opens new opportunities to study human brain functions through strategic input alterations and can even potentially be helpful for human-brain interfaces.

## Chapter 5

# Brain Activity Redundancies and Low-dimensional Representations

How many signals in the brain activities can be erased before the encoded information is lost? Surprisingly, we found that both reconstruction and classification of voxel activities can still achieve relatively good performance even after losing 80%-90% of the signals. This leads to questions regarding how the brain performs encoding in such a robust manner. This chapter investigates the redundancy and dependency of brain signals using two deep learning models with minimal inductive bias (linear layers). Furthermore, we explored the alignment between the brain and semantic representations, how redundancy differs for different stimuli and regions, as well as the dependency between brain voxels and regions.

Natural images are extremely rare in the vast image space, making them highly compressible [123]. Similarly, brain signals only occupy a small fraction of the signal space they reside in, and many signal dimensions are redundant—if we only care about encoding meaningful brain activities. [124] linked neural activity modeling to compressed sensing because of its low-dimensional nature, and [125] proposed decomposing signals into pre-

sentations of visual motifs as the way the brain performs compact neural encoding. With the recent emergence of larger-scale brain imaging datasets, we now have the chance to systematically study the redundancy and dependency of brain signals with deep learning models. In this work, we first examine the redundancy of brain signals in two tasks: signal reconstruction, in which the goal is to reproduce the activities across all voxels using only a subset of voxels, and classification, in which the goal is to determine the visual categories that appear in the stimuli. We embed the brain signals into a latent space and analyze the number of latent dimensions and interpolations. We also study the redundancies and dependencies in different visual subtasks, hemispheres, regions, and voxels. These steps help us analyze the dependencies and redundancies across different parts of the brain and reveal important insights about the visual brain architecture and representations.

In the following sections, we first introduce the data and models we use to study brain signal redundancy and dependency (section 5.1). We then discuss our findings (section 5.2), including: brain activity exhibits a high level of local and hierarchical dependency, and the high-dimensional signal, in fact, resides in a low-dimensional semantic space. In the same section, we also discuss the redundancy and dependency for different hemispheres, regions, and voxels, as well as for signals corresponding to stimuli with different semantics.

## 5.1 Dataset and Models

We conduct our studies on the Natural Scenes Dataset (NSD) [80], the same one used in the previous chapter 4, which records functional magnetic resonance imaging (fMRI) signals while the subjects view natural images. It provides high-resolution scans with a high signal-to-noise ratio (SNR) at an unprecedented scale. The number of samples in

NSD allows better training and investigation of brain data using deep learning models. In particular, we directly utilize the provided fMRI betas, the estimated response amplitude of each voxel to each trial, obtained through the general linear model (GLM), with an additional z-score operation for each session. There are eight subjects in NSD, and the following results are from subjects 1, 2, and 5: 1 is the principal subject, and 2, 5 are used for verification (result plots correspond to subject 1, if unspecified). We focus the study on one region of interest (ROI), *nsdgeneral*, which covers voxels responsive to the NSD experiment in the posterior aspect of the cortex.

Two models are involved in this study: one for voxel activity reconstruction and the other for signal classification. For both models, the inputs are flattened fMRI betas from *nsdgeneral* voxels. For the reconstruction model, we use an autoencoder (AE) with a three-linear-layer encoder and a three-linear-layer decoder. Nonlinear activations (ReLU) are added between the layers. The hidden dimension of the bottleneck representation is 1024 (encoder layers' output dimensions are 4096, 2048 and 1024, and decoder's are the reverse). The reconstruction model is trained with mean squared error loss, and we measure its performance using the voxel-wise reconstruction correlation coefficient. For the multi-label classification task, we train the model to classify if specific object categories exist in the image stimulus that triggers the corresponding fMRI. The category information is obtained from MS-COCO [84] as all NSD images are sampled from this image set. In total, there are 171 categories, including 80 bounded "things" categories (e.g., *person*, *car*) and 91 unbounded "stuff" categories (e.g., *sky*, *sea*). The classifier consists of 3 linear layers with ReLU activations in between (layers' output dimensions are 2048, 512, 171), followed by a Sigmoid activation. The classification model is trained with binary cross-entropy loss, and we measure its performance using AUC-ROC. We perform "masking" on the inputs to remove the information content of voxels. In the context of this chapter, masking is done by setting the voxel values to zero while keeping

the input dimension unchanged.

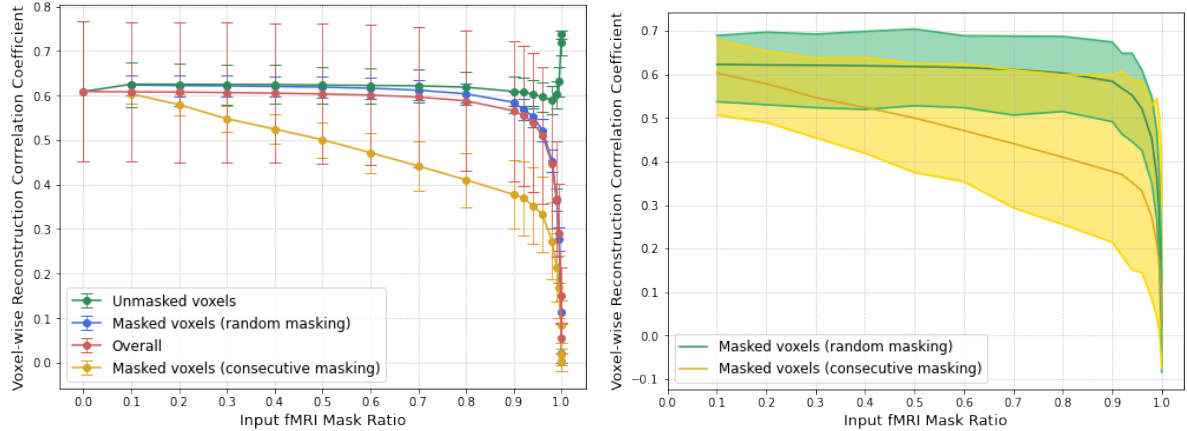
**Experiment details** Since each image stimulus in NSD corresponds up to three scans, we split our training-validation data image-wise: for example, for subject 1, 23715 samples corresponding to 8364 images are used as the train set, and 4035 samples corresponding to the remaining 1477 images are used as the validation set. Therefore, our pipeline never sees the image it is tested on during the training. We use Adam as the optimizer, with a  $2e-4$  learning rate,  $2e-4$  weight decay for the reconstruction model, and  $8e-6$  weight decay for the classification model. The values are manually tweaked with grid search. Since input lengths vary across subjects ( $N = 15724$  for subject 1,  $N = 14278$  for subject 2, and  $N = 13039$  for subject 5), models are trained in a per-subject manner. Our experiments are conducted on a Tesla V100 GPU with 32 GB memory (however, only around 4K MB memory is needed for a batch size of 64).

## 5.2 Findings

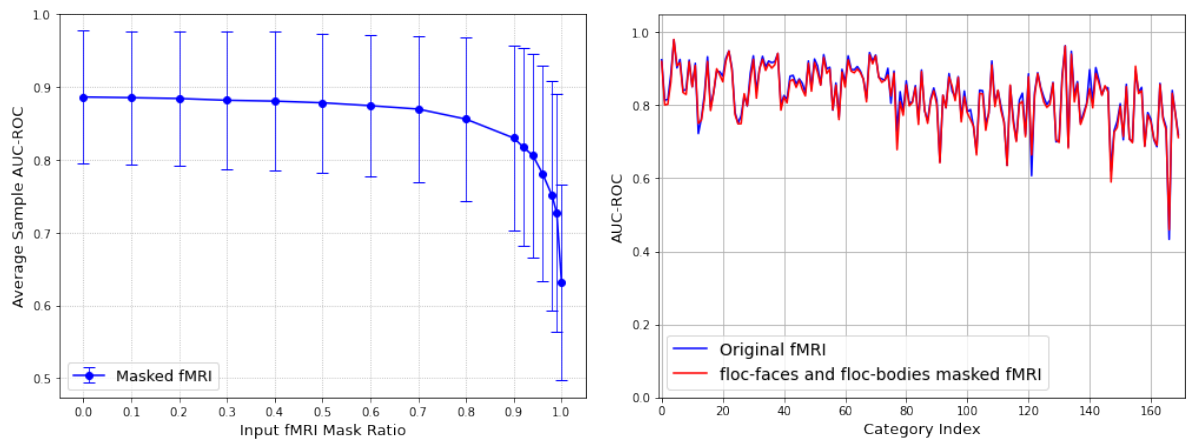
### 5.2.1 Brain Signals Contain High-level Redundancy

We studied the performance degradation when increasing the input masking ratio, ranging from zero (unmasked) to one (all-zero inputs). As shown in fig. 5.1, we observe that both reconstruction and classification models can retain the same level of performance even when masked up to 80%-90% of voxels if the masked positions are selected randomly. The most drastic performance drop happens after masking more than 95% input voxels. Nonetheless, for the reconstruction model, with input activities from only 16 random voxels, the prediction of all the masked voxel activities can still achieve an average of 0.113 voxel-wise correlation coefficient. This indicates that a considerable amount of redundancy exists in the fMRI signals. This redundancy is also more local-



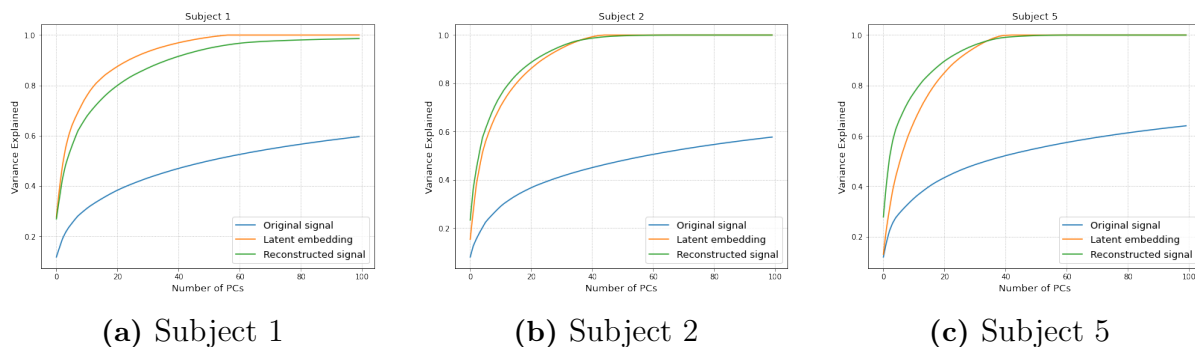


(a) fMRI signal reconstruction performance.



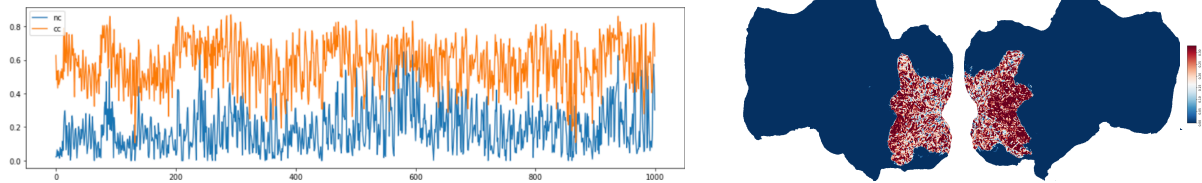
(b) fMRI multi-label classification performance.

**Figure 5.1:** Redundancies observed in the fMRI betas in terms of (a) voxel activity reconstruction, and (b) stimuli category classification. For reconstruction with an autoencoder, we gradually increase the masking ratio of the inputs. Two masking schemes are tested: randomly choosing masked voxels and consecutively masking voxels. With random masking, the reconstruction performance stays around the same level up to masking 80%-90% of the voxels. The right plot of (a) shows the min, mean, and max reconstruction performance of the two masking schemes. For classification results (b), the left plot shows sample-wise AUC-ROC as we increase the masking ratio: the significant drop also occurs after 80%-90%; the right plot shows the category-wise AUC-ROC with and without masking voxels in the floc-faces and floc-bodies ROIs, which turned out to be extremely close to each other, not affecting *person* category (index 0)'s performance.



**Figure 5.2:** Accumulated explained variance v.s. the number of principal components (PCs) used. The trend of the reconstructed signal is close to that of the latent representation.

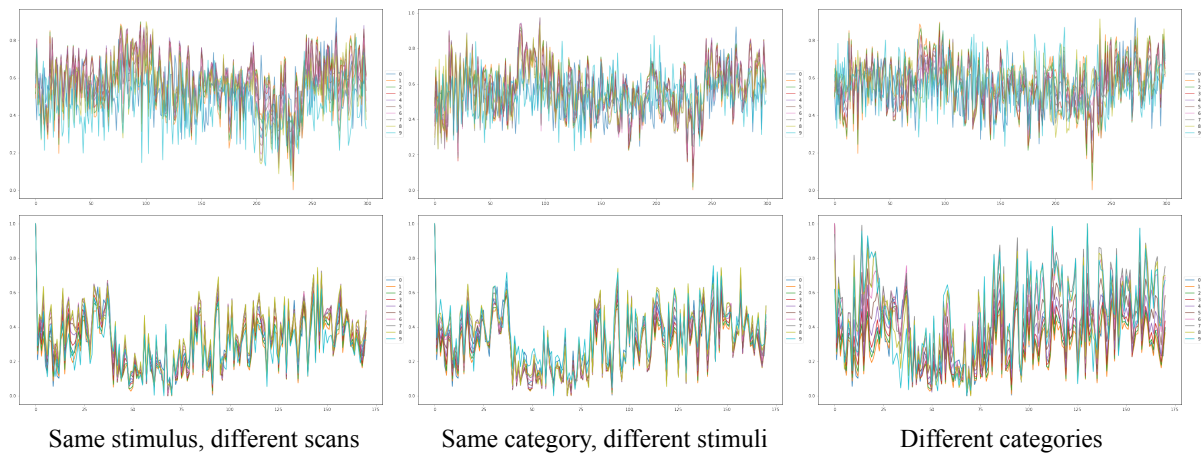
ized: if the masking is performed consecutively, where nearby voxels are masked together, the performance will decrease more consistently. In a consecutively masked window, the two ends also typically observe better performance as they can get information from unmasked neighbors. In addition to randomly masking voxels at different ratios, we also tested masking entire ROIs for the classifier inputs. In particular, we masked all voxels in floc-faces and floc-bodies, and computed the category-wise AUC-ROC. Surprisingly, the performance with original and ROI-masked inputs are almost the same (as in fig. 5.1b), even for the *person* category. This result suggests that brain signals also carry **hierarchical redundancy** apart from **local dependency**: the model can classify the signal based on activities in the low-level visual cortex without relying on regions with functional specializations. Note that this is different from compression or dimensionality reduction of the signals, since masking directly removes information in the observed signal space instead of in a transformed basis. Similar to our result here, recent work also found that natural language can be separately decoded from multiple cortical networks in each hemisphere [126].



**Figure 5.3:** The left plot shows the reconstruction correlation coefficient ( $cc$ ) together with the voxel-wise noise ceiling ( $nc$ ) for 1000 voxels.  $cc$  is calculated over the validation set of 4035 samples and aligns well with  $nc$  ( $cc$  and  $nc$  have a 0.67 correlation with p-value 0). The right plot shows  $cc - nc$  values on a flatmap. Higher-order regions typically have larger values (redder), meaning the reconstruction is better for those regions.

### 5.2.2 Autoencoders Effectively “Denoise” Brain Signals

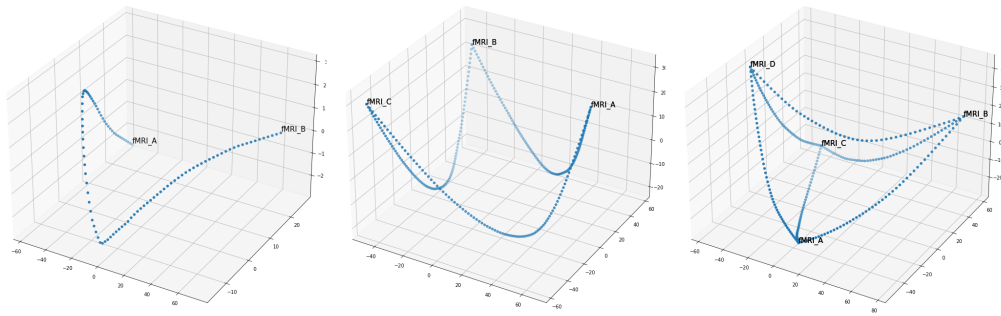
Just as Robust Principal Component Analysis (RPCA) can decompose images into low-rank and noise/outlier corruption components [127], an AE can “clean up” the high-dimensional signals by reconstructing them into low-dimensional ones. When applying PCA to subject 1’s 1000 samples (15724 voxels each) in the original signal space, 100 principal components (PCs) can explain 0.596 variance. [128] also mentions that the number of PCs required to explain 0.68 variance is typically two orders of magnitude smaller than the original number of voxels, meaning the number of PCs required for the 15724 voxels is on a hundred-scale. On the other hand, 100 PCs can explain more than 0.99 variance for both latent representations and reconstructed signals (fig. 5.2). This shows applying an AE can effectively obtain a cleaner version of fMRI signals. In comparison, we found neither RPCA nor independent component analysis (ICA) can achieve the same level of compression. To ensure that our reconstructions retain the primary information, we compared the model’s voxel-wise reconstruction correlation with the voxel noise ceilings and found a good alignment between the two (fig. 5.3). After adjusting voxel-wise performance with their noise ceilings, there is uneven performance across the brain: we observed better reconstructions for voxels in the higher-level regions.



**Figure 5.4:** Interpolating the latent embedding and generating reconstructions from the interpolations. The top row shows the generated signals for 300 voxels, and the bottom row shows the classification logits for 171 categories when passing the generated signal through the trained multi-label classifier. Values in both plots are normalized to 0-1. Interpolations are performed between three pairs of embeddings: (left) between two fMRIs corresponding to the same image; (middle) between two fMRIs corresponding to different images, but with exactly the same set of object categories; (right) between two fMRIs corresponding to two images having completely two different sets of object categories.

### 5.2.3 Brain Activity Resides in the Semantic Space—the Hopfieldian View

We further studied the latent representations obtained from the encoder. Given two fMRI signals, we interpolated their latent embeddings and passed the interpolations through the decoder to generate a set of fake fMRI signals. We then used these generated signals as inputs to the classifier model and obtain predicted logits for each class. As shown in fig. 5.4, the generated signals in the voxel space differ much more than their predicted logits: this is especially true for interpolations between the same image’s different scans or two scans corresponding to images with similar semantics. This result suggests that brain signals naturally reside in a more semantically rich space. Therefore, our findings support the recent claim that the Hopfieldian view, where representations

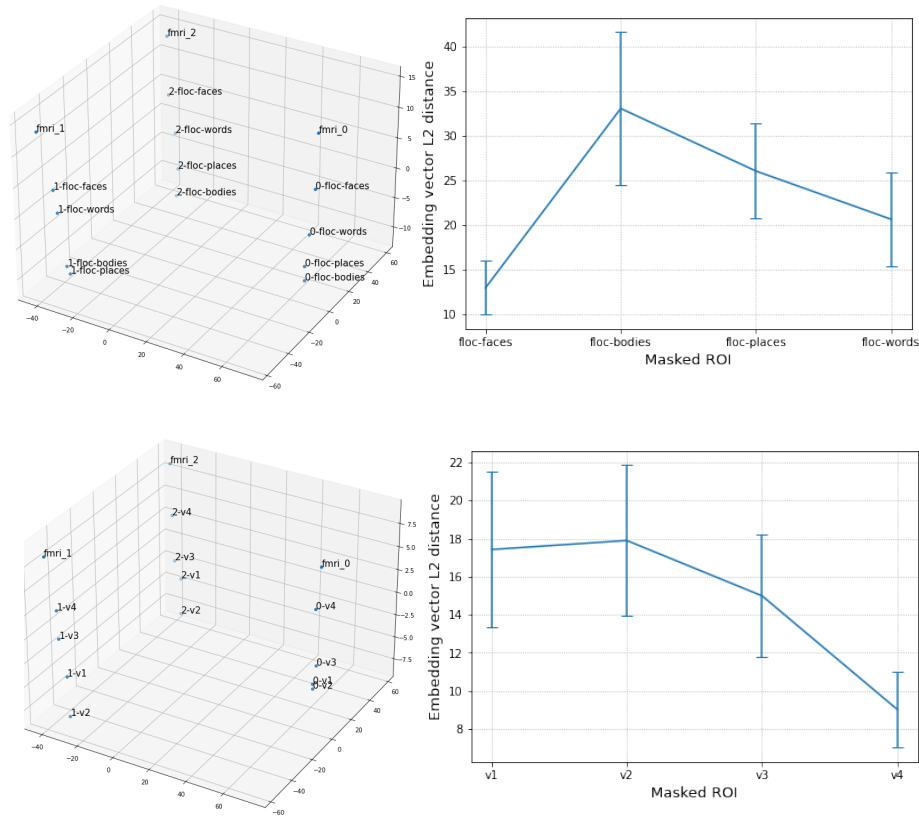


**Figure 5.5:** Latent representations of interpolated fMRI signals between two fMRI samples, pairs in three samples, and pairs in four samples. The interpolations occupy a much lower dimensional space: for 1000 latent representations of interpolations between two fMRI signals, only 2 PCs are needed to explain 0.998 variance, whereas 56 PCs are required to explain the same variance for 1000 unrelated fMRI signal embeddings. This indicates transitions between different fMRIs are cheap inside this learned latent space. For visualization, the first three PCs of the latent representations ( $> 0.999$  variance explained for each interpolation pair with step number = 1000) are used as the coordinates.

and transformations in the neural space are considered more important than individual neuron activities, is needed to explain cognitive processes [129]. Fig. 5.5 discusses the scenario when the interpolation happens in the original signal space instead of the latent embedding space: we found that brain signals can have rich representations with easy transitions. Moreover, we observe that voxel groups affect the latent representation in a consistent manner; for example, masking voxels in floc-bodies always results in representations further away from the original embedding than masking floc-faces voxels, indicating the former ones are more important to the overall latent representation (results are shown in fig. 5.6).

### 5.2.4 Masking and Attribution Reveal Voxel and Region Importance

By masking part of the inputs, we can investigate which area has more information by comparing the degradation of the reconstruction results. This section aims to understand

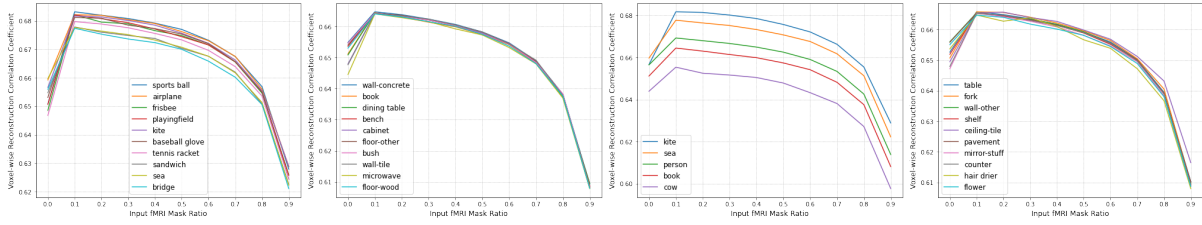


**Figure 5.6:** Masked fMRI embeddings. Masking different ROIs results in different distances between the masked and original signal embeddings. The relationship between these distances is consistent across samples (all pairs have a t-test p-value  $< 1e-50$  except for the  $V2 > V1$  pair, which has a p-value of 0.008): for example, masking floc-faces voxels always results in a closer embedding to the original signal to masking floc-bodies voxels. These distance relationships hold across subjects.

the differences among different categories, hemispheres, ROIs, and general redundancy patterns through masking, providing constrained inputs, and other input attribution methods.

## I. Categories

A natural hypothesis is that the brain encodes various categories with different levels of redundancy: this can be either a “nature” phenomenon shared by the population or



(a) Reconstruction performance of different categories under different masking ratios.



(b) Reconstruction performance of *person* fMRI minus that of *non-person* fMRI (L: subject 1; R: subject 5).

**Figure 5.7:** Reconstruction for different categories (a) From left to right: (1) categories exhibit groupings, (2) unmasked order is very different compared with masked ones, (3) the orders are consistent for categories in different groups for masked inputs, and (4) the orders can be inconsistent for categories within the same group (as in having a close performance under the 10%-masking ratio). (b) fMRI signals triggered by images containing *person* perform better (redder) at the right hemisphere’s low to mid-level visual regions than those that do not contain *person*. In addition, when calculating the correlation between *cc* and *nc* (refer to fig. 5.3), person ones are larger than non-person ones (with an average of 0.673 v.s. 0.638 for subject 1).

a “nurture” one influenced by individual experiences. To test this, we separated fMRI signals based on the categories of their corresponding stimuli. For a stimulus with multiple objects from different categories, we include its fMRI signal in all these categories. For unmasked inputs, the reconstruction performance for a category’s fMRI follows its occurrence: a category with more fMRI samples has a higher chance of having its fMRIs better reconstructed. However, with partially masked inputs, the category performance orderings differ from the unmasked ordering and remain mostly consistent across masking ratios. These orders are also more relevant to the category semantics (categories that belong to the same supercategory tend to have a clustered performance) and less

relevant to the occurrence frequency (see figs. 5.7, 5.8, 5.9 for visual results). Noticeable groupings of categories can be observed based on their performance curve across masking ratios. There are some consistencies across the three test subjects: sports-image triggered signals typically receive better reconstruction, and animal-image triggered ones have the opposite trend. Other orderings are more subject-specific.

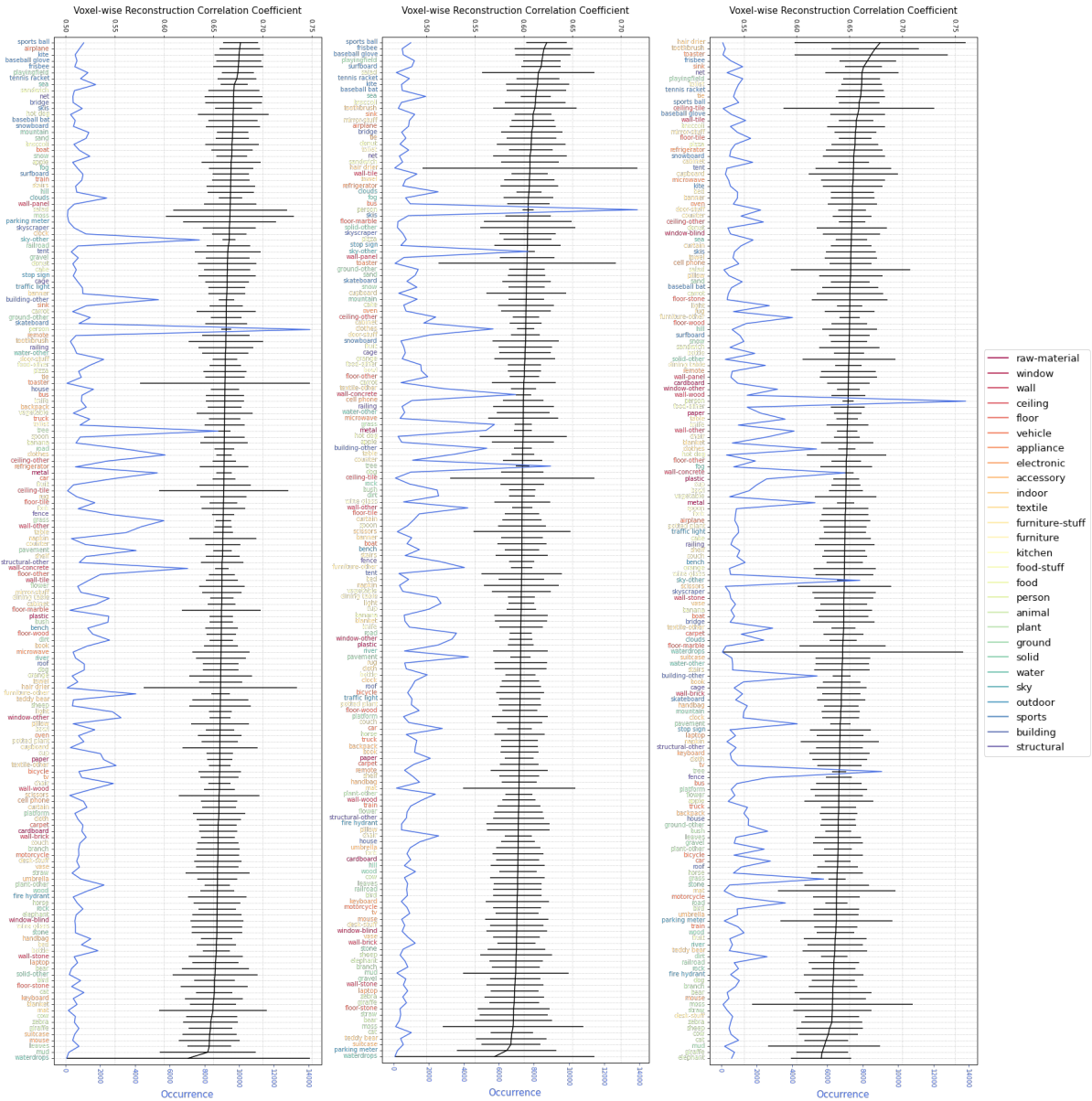
One interesting observation is that the brain can learn representations from other categories under the same supercategory. For example, keeping the number of total training/validation samples the same, when we take out the entire *food* supercategory from the training data, the reconstruction of *sandwich* signals ranks 45th out of the 171 categories. But when the training data has other *food* categories, but no *sandwich*, the reconstruction of *sandwich* signals ranks 20th (if there are *sandwich* signals in the training data, its reconstruction ranks 9th). Taking out an entire supercategory also makes the overall reconstruction worse for all categories. These show that signals of each supercategory can have a relatively unique representation. Or, the *food* supercategory impacts the overall representation because there is a food-selective component in the visual system, as the recent work [130] suggests.

We also tested dichotomous separations, where signals are separated based upon a single category (e.g., *person* and *non-person* triggered fMRIs). Upon testing several partitions based upon *person*, *tree*, *sea*, *cow*, *building*, we found that only *person/non-person*'s performance has a systematic difference across subjects: person fMRIs are reconstructed better than non-person fMRIs in low-to-mid-level visual regions of the right hemisphere.

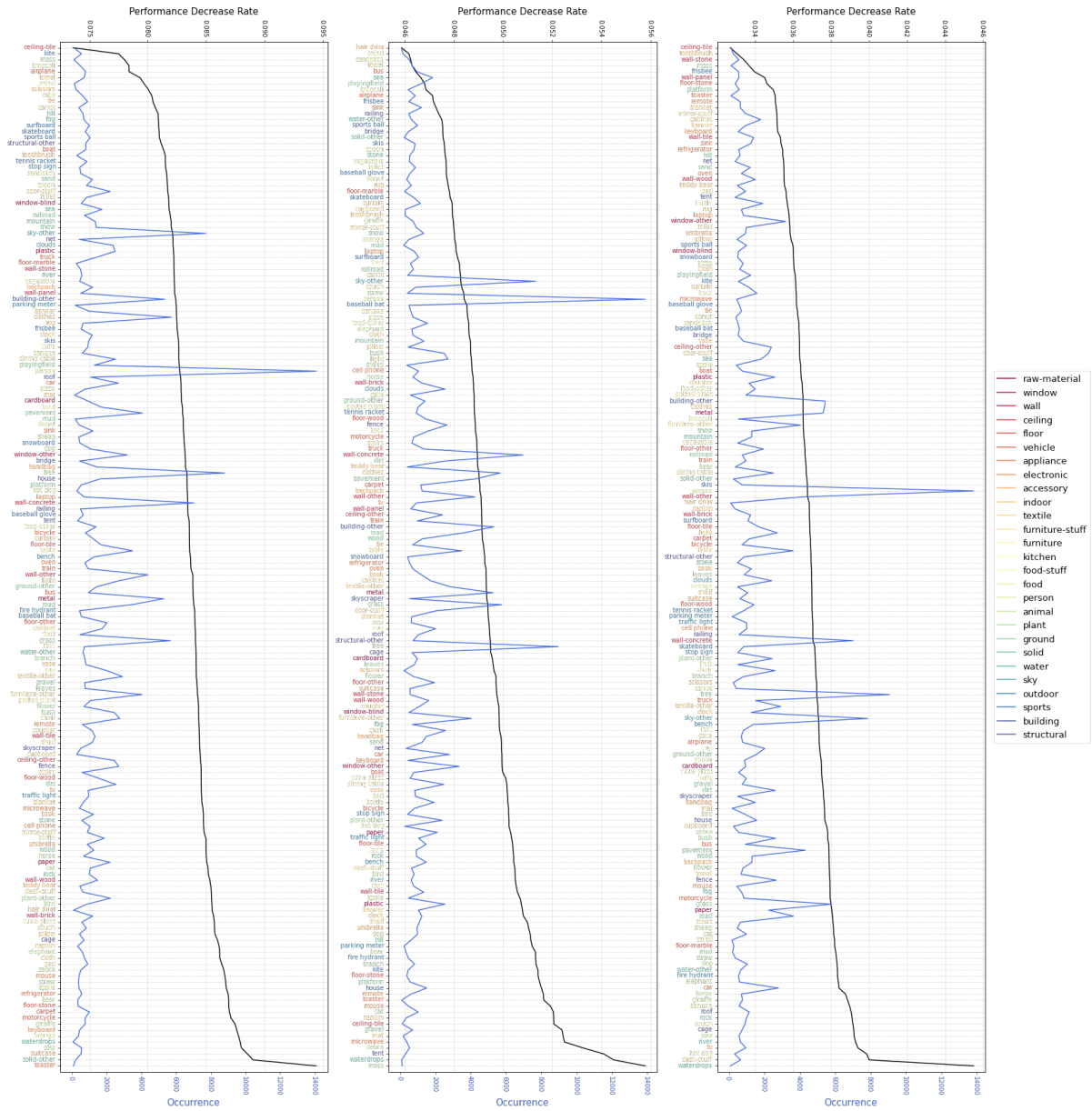
## II. Hemispheres

There is a known asymmetry between the two hemispheres when it comes to visual processing: from the early study of split-brain patients [131] to later experiments with hierarchical letters and objects [132, 133], results suggest that the right hemisphere (*rh*)

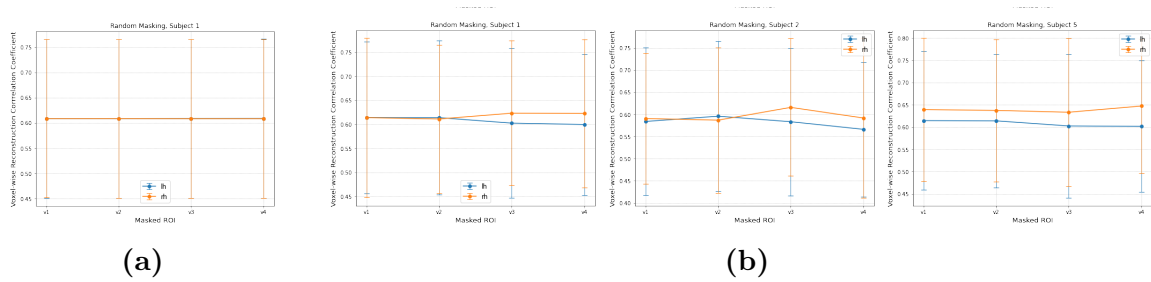




**Figure 5.8:** Average voxel-wise reconstruction correlation coefficient ( $cc$ ) for fMRI samples corresponding to different categories. The performance is measured under a 50%-masking ratio. Plots are for subjects 1, 2, and 5 from left to right. Error bars stand for the standard deviations of the average  $cc$  across all samples of that category. We also plot the sample occurrence (in blue lines) for individual categories as some categories, like *person*, have significantly more samples than other categories. Colors are based on super-categories, as indicated in the legend.



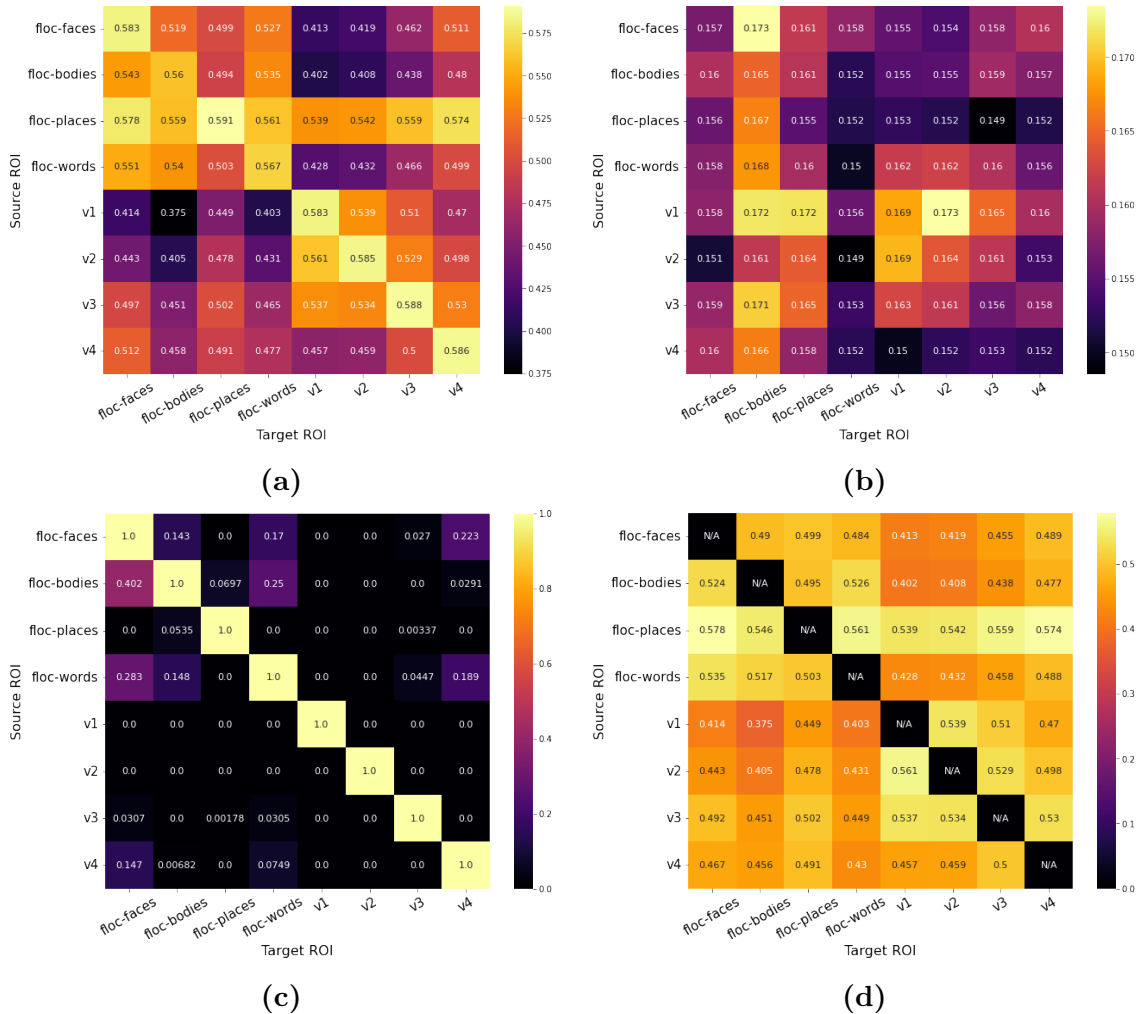
**Figure 5.9:** Performance decrease rate for different categories, together with the category sample occurrence. The rate is calculated as (performance with 10%-masked inputs - performance with 90%-masked inputs) / performance with 10%-masked inputs.



**Figure 5.10:** (a) Unmasked voxel and (b) Masked voxel reconstruction performance when masking V1-V4 on either the left hemisphere (*lh*) or the right hemisphere (*rh*). Subjects 1, 2, and 5 are used for the task. Masking the visual cortex at different levels on either hemisphere does not affect unmasked voxels differently (true for all subjects). But the reconstructions of masked V3/V4 voxels on *lh* consistently have a worse reconstruction than those on *rh* (t-test p-value  $< 1e-8$ ).

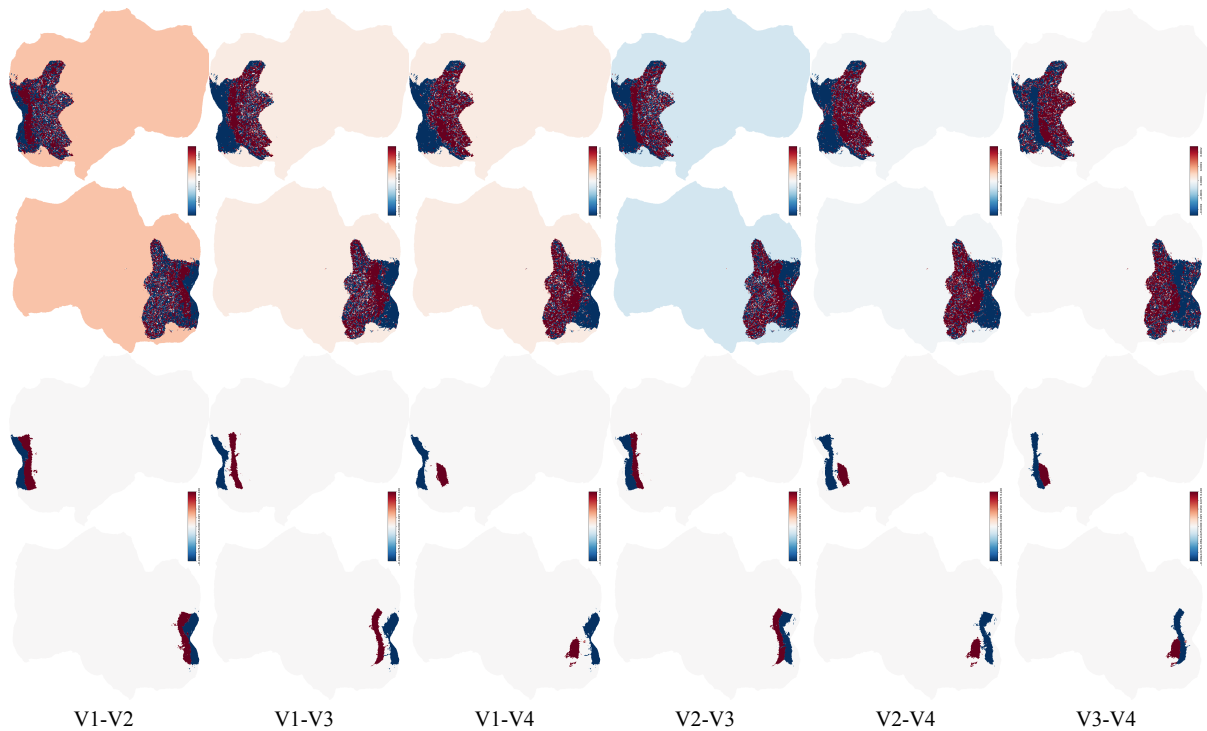
is better at identifying a global feature while the left hemisphere (*lh*) is better at processing local ones. But *which hemisphere's information can be better recovered from other regions when damaged?* Masking an ROI's signals can conveniently emulate a lesion in an experimental setting. In our study, we masked 300 voxels in each of V1 to V4 for either *rh* or *lh*, and measured the reconstruction performance for both masked and unmasked regions. Interestingly, we observe no change in the performance of unmasked voxels but statistically significant differences, with t-test  $p < 1e^{-8}$ , between *lh/rh-V3/4* for masked ones. Across subjects, *rh-V3/4* can be better reconstructed from other voxels activities (shown in fig. 5.10), suggesting they are more robust to region damages than the *lh* counterparts. Moreover, since the separation of “local” and “global” can be dynamic and depends on the task, as [134] suggests, we evaluated if the same phenomenon occurs for reconstruction. To this end, we tested fMRI signals corresponding to different categories (the categorical signal separation is the same as in the previous section). However, no noticeable performance difference is observed: we compared the reconstruction performance of the following categories, *person*, *tree*, *sea*, *kite*, *book*, *cow*, *giraffe*, *cat*, *dog*, and only *cow* differs from the others or the average.

## III. Pairwise ROIs



**Figure 5.11:** Pairwise reconstruction between ROIs of subject 1. For each matrix, the rows are source ROIs that provide input activities, and the columns are target ROIs whose reconstruction performance is evaluated (in terms of the voxel-wise correlation coefficient). The plots are: (a) reconstruction mean, (b) reconstruction standard deviation, (c) voxel overlap percentage normalized by the total number of target ROI voxels, and (d) reconstruction mean when the overlapped voxels' activities are not provided in the inputs.

By providing an ROI A's activities as inputs, we can evaluate the reconstruction of another ROI B to understand brain region dependencies. We measured pairwise reconstructions between four ROIs V1-V4 of the visual cortex and four functional localizer



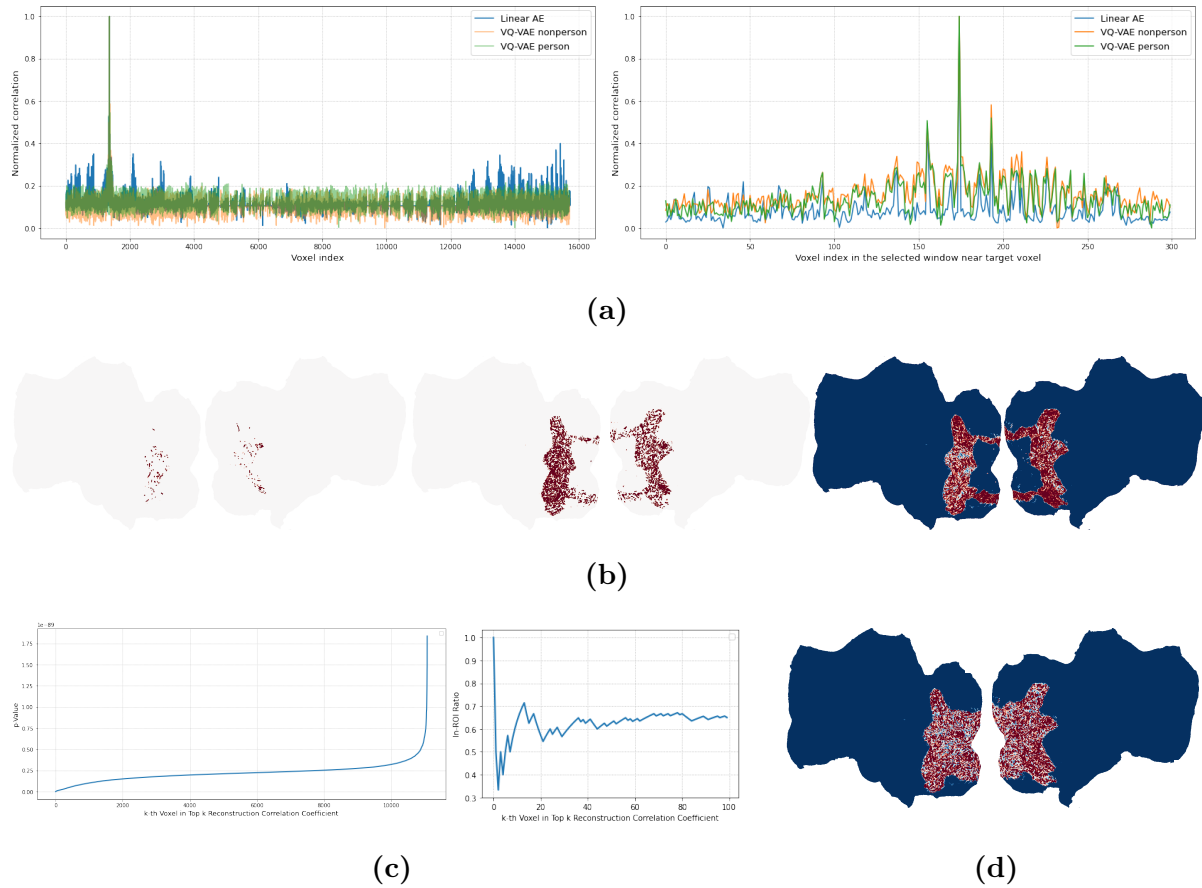
**Figure 5.12:** The top two rows of the flatmaps are reconstruction performance differences when masking two different visual cortex ROIs. The bottom two rows are visualized localizers of corresponding ROIs. Both sets have *rh* on top of *lh*. The region name stands for the masked region: for example, the first column “V1-V2” means “subtracting the voxel-wise reconstruction with V2-masked inputs from the voxel-wise reconstruction with V1-masked inputs”. Discrepancies are observed between the performance difference (top set) and corresponding localizers (bottom set), from which we can identify region dependencies. For example, V2 depends more on V1 than V3, and it also shows additional dependencies with the posterior intraparietal sulcus (IPS) area on the right hemisphere (top tip of the *rh* flatmap).

(floc) ROIs. The result provides a straightforward dependency matrix as shown in the fig. 5.11. In general, given voxels from floc ROIs, voxels in higher visual areas are reconstructed better, confirming the high-level nature of floc ROIs. In particular, floc-places can recover visual cortex voxels much better, implying that the high-level place representations cover visual details more than other tasks. On the other hand, given visual cortex voxels, floc-faces and floc-places are reconstructed better, indicating the other two tasks require additional information from other regions apart from the visual cortex. Note that

even for self-reconstructions, where the target ROI is the source ROI, the reconstruction performances differ, suggesting that some ROIs are more self-contained than others and their dependencies are more local. We also conducted studies between different visual cortices, as shown in fig. 5.12. These experiments that aim to find dependencies between brain regions can also link to works in connective field modeling [135, 136].

#### IV. Optimal voxel measurement for reconstruction and classification

Considering that low-dimensional embeddings can effectively represent neural signals, a reasonable question is whether one can utilize techniques similar to compressed sensing and use fewer measurements to obtain high-resolution samples. Here we explore the dependencies between voxels to find voxels that (1) contribute more to other voxels' reconstruction and (2) contribute more to the semantic categorization of the signals. For (1), we chose a target voxel to measure the reconstruction performance while providing voxels from V1-V4 plus one additional voxel. Then, we brute-forced this additional voxel and plotted a dependency heatmap. We observed strong local dependencies and dependencies in the symmetric positions of the other hemisphere (figs. 5.13a and 5.13b). When the "contribution" is aggregated for all target voxels, we can get a blueprint regarding which voxel is most important. However, this result is subject-specific, and more explorations are needed to extend the results to different individuals and achieve a better upsampling of the signals. For this voxel activity reconstruction task, we also tested an additional VQ-VAE with a convolutional encoder and decoder and a hidden dimension of 984 apart from the AE detailed in section 5.1. Compared to linear AE, convolution-based VQ-VAE can only capture local dependencies well but lose the global view (fig. 5.13a). Nonetheless, it can reconstruct unmasked signals much better, with  $cc$  having a mean of 0.968 and 0.003 std. For (2), we utilized SHapley Additive exPlanations (SHAP) [137] to perform input attributions, identifying voxels' contribution to classifying



**Figure 5.13:** Examining the voxel importance of subject 1 with (a-c) reconstruction models and (d) the classification model. For the reconstruction, Voxel with index 1364 is selected as the target voxel at which we measure the recovery performance; it locates on  $lh$  and belongs to both floc-faces and floc-bodies. **(a)** The recovery performance at each voxel if they are served as the “additional” input apart from V1-V4 voxel activities. The right plot is the zoom-in view of the left plot, showing the strong local dependencies, which are consistent across models and signal types (the values in (a) are normalized to 0-1 since AE and VQ-VAE  $cc$  are at different scales). **(b)** Given AE’s result, we plot (left) the top 200 contributing voxels, (middle) voxels that lead to a reconstruction performance larger than the mean value, and (right) the overall p-values of the reconstruction  $cc$ . We can observe that voxels on mirrored positions of  $rh$  are also contributing to the target voxel’s reconstruction, but overall  $lh$  voxels are more important for this target voxel on  $lh$ . p-Values are also aligned well: positions with better performance (larger  $cc$ ) also have smaller p-values. **(c)** (left) The p-value changes with decreasing recovery  $cc$ . There is a sharp increase near the end, indicating those voxels are more irrelevant; (right) The in-ROI ratio for the top-100 contributing voxels: not all are from the ROIs that the target voxel belongs to (floc-faces/bodies in this case). **(d)** SHAP input attribution (absolute values) aggregated across categories. Redder (higher attribution) means the voxel is more crucial in determining if a specific category exists in the stimulus.

each category. Aggregating the attributions for all the categories results in one overall contribution value for each voxel, thus providing a general voxel importance map for signal categorization. There is a disparity between the two hemispheres' aggregated attributions, with *rh* having more critical voxels in higher-level regions. However, the attribution of *lh* is generally higher than *rh* (t-test p-value < 0.005) (fig. 5.13d). On the other hand, when looking at (unaggregated) category-wise SHAP attributions, the results also support distributed coding scheme [138]: each category's representation is coded by a subpopulation of voxels, and each voxel contributes to multiple categories' representations.

### 5.3 Discussions and Conclusion

Although we studied many aspects regarding brain signal redundancy and dependency, the list is by no means exhaustive. For example, one can examine the impact of the dorsal/ventral stream on fMRI signals of different categories. Other questions can also be explored: the brain can encode both low-level and semantic information upon seeing images, so *how much of each (low-level/semantic) is retained in the latent embeddings when the fMRI signals are mapped from the encoder?* To answer this question, we will need to train a model trained from latent embedding to perform multi-label classification, and compare the performance with the model trained with fMRI inputs. Similarly, we need to train two other models for low-level details: one takes fMRI signals as inputs and another from the latent embeddings. Then we can compare the retainment percentage of low-level image information. These latter two models require labels of the stimuli regarding image details, such as color, shape, orientation, etc., which are missing in the current MS COCO dataset. In future work, one could potentially use off-the-shelf detectors to generate pseudo labels as a way to answer the above question.



In addition, since linear layers have less inductive bias than convolutional layers, the models we used are composed of linear layers. In fact, we also tested a convolution-based autoencoder VQ-VAE [119], as discussed in fig. 5.13. With strong local-grouping assumptions of the convolution operation, VQ-VAE fails to capture global dependencies. Nonetheless, we still observed similar curves for masked voxel reconstruction: the noticeable performance decrease starts after the 80% masking ratio. Its results on category reconstruction orders, latent embeddings' semantic alignment, orders of pairwise ROI dependencies, etc., are also consistent with the linear model results we reported. However, it will still be beneficial to test models with more structural types in future work to examine if the results hold for all frameworks.

In this chapter, we systematically studied the redundancy and dependency of fMRI signals with an AE and a multi-label classifier. We found AE can reconstruct signals in a much lower-dimensional space while having a good reconstruction correlation: this suggests new ways for signal decomposition and denoising. We also found signals' latent space is more aligned with the semantic space, supporting a Hopfieldian view of the brain. This low-dimensional representation, or semantic information of stimuli, can be further used to guide signal compression, upsampling, and reconstruction. In addition, our results suggest that the brain encodes different scene semantics with varying levels of redundancy, resulting from a combination of nature and nurture. Discrepancies between hemispheres, ROI dependencies, and voxels dependencies are also explored, each providing additional insights regarding brain visual encodings.

# Chapter 6

## Neuroscience-inspired DNN modeling

Previous chapters discussed how we could use various machine learning techniques to gain a better understanding of the brain, and this chapter will discuss the opposite direction [139]: how tools and insights from neuroscience can help us investigate artificial neural networks and even inspire new architecture developments.

In section 6.1, inspired by the success of psychophysics tools in studying human visual processes, we apply similar analyses to artificial neural networks (ANNs) to unveil inherent model biases at different levels (model, layer, single neuron), and show how these techniques can be used to perform/detect adversarial attacks on black-box classifiers. Apart from directly borrowing tools from the neuroscience domain, we can also modify machine learning models based on the findings we made. In chapter 5, we explored the redundancy in brain signals with deep learning models—which, in turn, motivated us to add more sparsity into ANNs, as we will introduce in section 6.2. In particular, we only keep partial neurons activated at each layer and alternate the selection across layers. This saves many parameters and computations while maintaining competitive performance.

## 6.1 White Noise Analysis of Neural Networks

In this section, a white noise analysis of modern deep neural networks is presented to unveil their biases at the whole network level or the single neuron level. Our analysis is based on two popular and related methods in psychophysics and neurophysiology, namely classification images and spike-triggered analysis. These methods have been widely used to understand the underlying mechanisms of sensory systems in humans and monkeys. We leverage them to investigate the inherent biases of deep neural networks and to obtain the first-order approximation of their functionality. We emphasize on CNNs since they are currently state-of-the-art methods in computer vision and are a decent model of human visual processing. In addition, we study multi-layer perceptrons, logistic regression, and recurrent neural networks. Experiments over four classic datasets, MNIST, Fashion-MNIST, CIFAR-10, and ImageNet, show that the computed bias maps resemble the target classes and, when used for classification, lead to an over two-fold performance than the chance level. Further, we show that classification images can be used to attack a black-box classifier and to detect adversarial patch attacks. Finally, we utilize spike-triggered averaging to derive the filters of CNNs and explore how the behavior of a network changes when neurons in different layers are modulated. Our effort illustrates a successful example of borrowing from neurosciences to study ANNs and highlights the importance of cross-fertilization and synergy across machine learning, deep learning, and computational neuroscience.

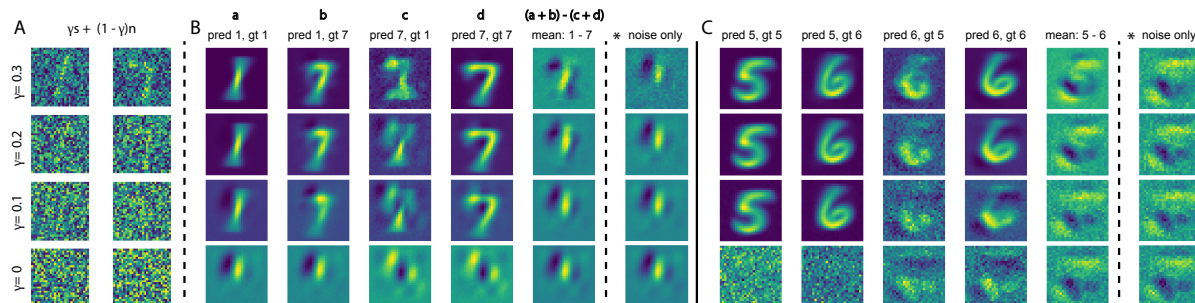
### 6.1.1 Introduction

Any vision system, biological or artificial, has its own biases. These biases emanate from different sources. Two common sources include: (1) the environment and the data on which the system has been trained, and (2) system constraints (e.g., hypothesis class,

model parameters). Exploring these biases is important from at least two perspectives. First, it allows us to better understand a system (e.g., explain and interpret its decisions). Second, it helps reveal system vulnerabilities and make it more robust against adversarial perturbations and attacks.

In this section, we recruit two popular methods from computational neuroscience to study the inherent biases in deep neural networks. The first one, called the classification images technique, was introduced into visual psychophysics by [140] as a new experimental tool. It has been used to examine visual processing and to understand vision across a variety of tasks, including simple detection tasks, visual search, and object recognition. It has also been applied to the auditory domain. See [141] for a review of the topic. The second method, known as spike-triggered analysis [142], is often used to discover the best stimulus to which a neuron responds (e.g., oriented bars). These methods are appealing for our purpose since they are general and can be applied to study any black box system (so long it emits a response to an input stimulus), and only make a modest number of assumptions. From a system identification point of view, they provide a first-order approximation of a complex system such as the brain or an artificial neural network. Both methods are detailed in section 6.1.2.

By feeding white noise stimuli to a classifier and averaging the ones that are categorized into a particular class, we obtain an estimate of the templates it uses for classification. Unlike classification image experiments in human psychophysics, where running a large number of trials is impractical, artificial systems can often be tested against a large number of inputs. While still a constraint, we will discuss how such problems can be mitigated (e.g., by generating stimuli containing faint structures) and demonstrate our findings with the two aforementioned techniques in section 6.1.3. Over four datasets, MNIST [143], Fashion-MNIST [144], CIFAR-10 [145], and ImageNet [86], we employ classification images to discover implicit biases of a network, utilize those biases



**Figure 6.1:** Illustration of the classification images concept. **(a)** Two sample digits as well as their linear combination with different magnitudes of white noise (eq. (6.3)). **(b)** Average correct and incorrect prediction maps of a binary CNN trained to separate digits 1 and 7. The fifth column shows the difference between the average of stimuli predicted as 1 and the average of stimuli predicted as 7. The column marked with “\*” is similar to the fifth column but computation is done only over noise patterns (and not the augmented stimuli), hence “classification images” (i.e.,  $(\bar{\mathbf{n}}^{11} + \bar{\mathbf{n}}^{71}) - (\bar{\mathbf{n}}^{17} + \bar{\mathbf{n}}^{77})$ ; eq. (6.1)). These templates can be used to classify a digit as 1 or 7. Yellow (blue) color corresponds to regions with positive (negative) correlation with the response as 1. **(c)** Same as B but using a 5 vs. 6 CNN.

to influence network decisions, and detect adversarial perturbations. We also show how spike-triggered averaging can be used to identify and visualize filters in different layers of a CNN. Finally, in a less directly related analysis to classification images, we demonstrate how decisions of a CNN are influenced by varying the signal-to-noise ratio (akin to microstimulation experiments in monkey electrophysiology or priming experiments in psychophysics). We find that CNNs behave in a similar fashion to their biological counterparts, and their responses can be characterized by a psychometric function. This may give insights regarding top-down attention and feedback mechanisms in CNNs [146].

### 6.1.2 Related Works and Concepts

Our work relates to a large body of research attempting to understand, visualize, and interpret deep neural networks. These networks have been able to achieve impressive performance on a variety of challenging vision and learning tasks (e.g., [147, 148]). How-

ever, they are still not well understood, have started to saturate in performance [149], are brittle<sup>1</sup>, and continue to trail humans in accuracy and generalization. This calls for a tighter confluence between machine learning, computer vision, and neuroscience. In this regard, the proposed tools here are complementary to the existing ones in the deep learning toolbox.

Perhaps, the closest work to ours is [152], where they attempted to learn biases in the human visual system and transfer those biases into object recognition systems. Some other works (e.g., [153]) have also used human data (e.g., fMRI, cell recording) to improve the accuracy of classifiers, but have not utilized classification images. [122] used is activation maximization to iteratively change the pixel values in the direction of the gradient to maximize the firing rate of V4 neurons<sup>2</sup>. Unlike these works, here we strive to inspect the biases in classifiers, in particular, neural networks, to improve their interpretability and robustness.

## I. Classification images

In a typical binary classification image experiment, on each trial, a signal  $\mathbf{s} \in \mathbb{R}^d$  and a noise image  $\mathbf{z} \in \mathbb{R}^d$  are summed to produce the stimulus  $\mathbf{n}$ . The observer is supposed to decide which of the two categories the stimulus belongs to. The classification image is then calculated as:

$$\mathbf{c} = (\bar{\mathbf{n}}^{12} + \bar{\mathbf{n}}^{22}) - (\bar{\mathbf{n}}^{11} + \bar{\mathbf{n}}^{21}) \quad (6.1)$$

where  $\bar{\mathbf{n}}^{sr}$  is the average of noise patterns in a stimulus-response class of trials. For example,  $\bar{\mathbf{n}}^{12}$  is the average of the noise patterns over all trials where the stimulus contained signal 1, but the observer responded 2.  $\mathbf{c} \in \mathbb{R}^d$  is an approximation of the template that

<sup>1</sup>Current deep neural networks can be easily fooled by subtle image alterations in ways that are imperceptible to humans; a.k.a adversarial examples [150, 151].

<sup>2</sup>See <https://openreview.net/forum?id=H1ebhnEYDH> for a discussion on this.

the observer uses to discriminate between the two stimulus classes. The intuition behind the classification images is that the noise patterns in some trials have features similar to one of the signals, thus biasing the observer to choose that signal. By computing the average over many trials, a pattern may emerge.  $\mathbf{c}$  can also be interpreted as the correlation map between stimulus and response:

$$\text{corr}[\mathbf{n}, r] = \frac{\mathbb{E}(\mathbf{n} - \mathbb{E}[\mathbf{n}])\mathbb{E}(r - \mathbb{E}[r])}{\sigma_n \sigma_r} \quad (6.2)$$

where  $\sigma_n$  is the pixel-wise standard deviation of the noise  $\mathbf{n}$  and  $\sigma_r$  is the standard deviation of response  $r$ . High positive correlations occur at spatial locations that strongly influence the observer's responses. Conversely, very low (close to zero) correlations occur at locations that have no influence on the observer's responses. Assuming zero-mean noise and an unbiased observer, Eq. 2 reduces to  $\mathbf{c}_{\text{corr}} = \bar{\mathbf{n}}^{*2} - \bar{\mathbf{n}}^{*1}$ , where  $\bar{\mathbf{n}}^{*u}$  is the average of the noise patterns over all trials where the observer gave a response  $u$  (see [141] for details). Thus,  $\mathbf{c}_{\text{corr}}$  is the average of the noise patterns over all trials where the observer responded  $r = 2$ , minus the average over all trials where the observer responded  $r = 1$ , regardless of which signal was presented.

We have illustrated the classification images concept in fig. 6.1 with a binary classifier trained to separate two digits. The stimulus is a linear combination of noise plus signal as follows:

$$\mathbf{t} = \gamma \times \mathbf{s} + (\mathbf{1} - \gamma) \times \mathbf{n}; \quad \gamma \in [0, 1] \quad (6.3)$$

The computed templates for different  $\gamma$  values<sup>3</sup>, using about 10 million trials, highlight regions that are correlated with one of the digits (here 1 vs. 7 or 5 vs. 6). The template fades away with increasing noise (e.g.,  $\gamma = 0$ ) but it still resembles the template in the

<sup>3</sup>We use classification images, bias map, template, and average noise pattern, interchangeably. Please do not confuse this bias with the bias terms in neural networks.

low-noise condition (i.e.,  $\gamma = 0.3$ ).

## II. Spike-triggered analysis

The spike-triggered analysis, also known as “reverse correlation” or “white-noise analysis”, is a tool for characterizing the response properties of a neuron using the spikes emitted in response to a time-varying stimulus. It includes two methods: spike-triggered averaging (STA) and spike-triggered covariance (STC). They provide an estimate of a neuron’s linear receptive field and are useful techniques for the analysis of electrophysiological data. In the visual system, these methods have been used to characterize retinal ganglion cells [154, 155], lateral geniculate neurons [156], and simple cells in the primary visual cortex [157, 158]. See [159] for a review.

STA is the average stimulus preceding a spike. It provides an unbiased estimate of a neuron’s receptive field only if the stimulus distribution is spherically symmetric (e.g., Gaussian white noise). STC can be used to identify a multi-dimensional feature space in which a neuron computes its response. It identifies the stimulus features affecting a neuron’s response via an eigendecomposition of the spike-triggered covariance matrix [160, 161].

Let  $\mathbf{x} \in \mathbb{R}^d$  denote a spatio-temporal stimulus vector affecting a neuron’s scalar spike response  $y$  in a single time bin. The main goal of neural characterization is to find  $\Theta$ , a low-dimensional projection matrix such that  $\Theta^T \mathbf{x}$  captures the neuron’s dependence on the stimulus  $\mathbf{x}$ . The STA and the STC matrix are the empirical first and second moments of the spike-triggered stimulus-response pairs  $\{\mathbf{x}_i | y_i\}_{i=1}^N$ , respectively. They are defined as:

$$\text{STA: } \mu = \frac{1}{n_{sp}} \sum_{i=1}^N y_i \mathbf{x}_i, \quad \text{and} \quad \text{STC: } \Lambda = \frac{1}{n_{sp}} \sum_{i=1}^N y_i (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T \quad (6.4)$$

where  $n_{sp} = \sum y_i$  is the number of spikes and  $N$  is the total number of time bins. The



traditional spike-triggered analysis gives an estimate for the basis  $\Theta$  consisting of: (1)  $\mu$ , if it is significantly different from zero, and (2) the eigenvectors of  $\Lambda$  corresponding to those eigenvalues that are significantly different from eigenvalues of the prior stimulus covariance  $\Phi = \mathbb{E}[XX^T]$ . When a stimulus is not white noise (i.e., is correlated in space or time), whitened STA can be written as:

$$\text{STA}_w = \frac{N}{n_{sp}} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (6.5)$$

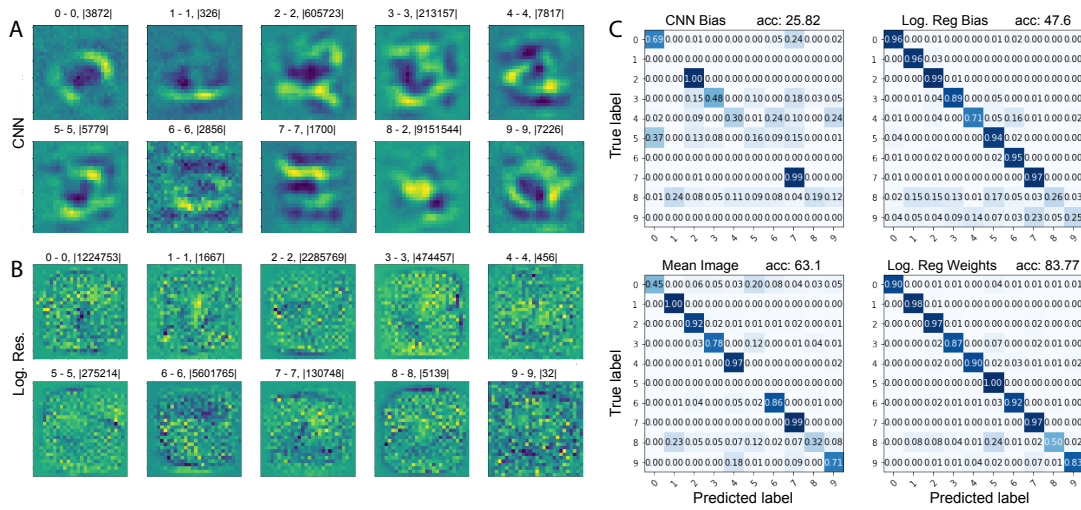
where  $\mathbf{X}$  is a matrix whose  $i$ th row is the stimulus vector  $\mathbf{x}_i^T$  and  $\mathbf{y}$  denotes a column vector whose  $i$ th element is  $y_i$ . The whitened STA is equivalent to linear least-squares regression of the stimulus against the spike train.

Classification images and spike-triggered analysis are related in the sense that both estimate the terms of a Wiener/Volterra expansion in which the mapping from the stimuli to the firing rate is described using a low-order polynomial [142]. See [162] for a discussion on this. Here, we focus on STA and leave STC for future work.

### 6.1.3 Applications to Deep Learning Models

We present four use cases of classification images and STA to examine neural networks, with a focus on CNNs since they are a decent model of human visual processing and are state-of-the-art computer vision models.<sup>4</sup> Our approach, however, is general and can be applied to any classifier. In particular, it is most useful when dealing with black-box methods where choices are limited.

<sup>4</sup>Code is available at: <https://github.com/aliborji/WhiteNoiseAnalysis.git>.



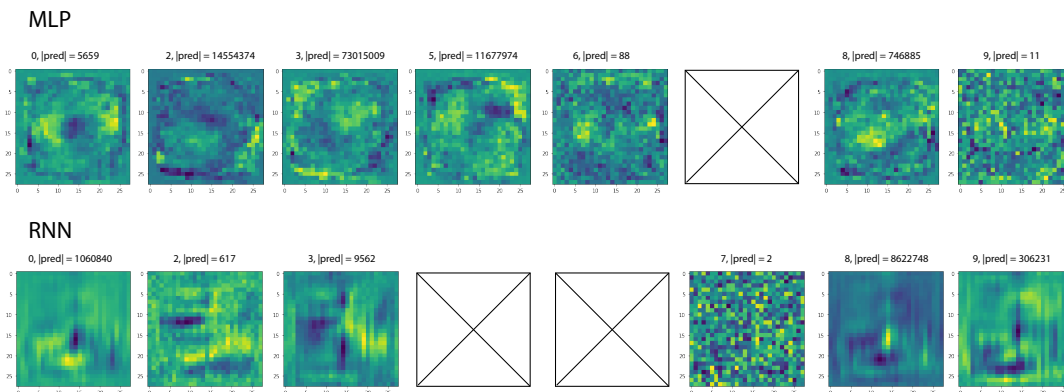
**Figure 6.2:** (a) Classification images of a CNN trained on MNIST (with 99.2% test accuracy). Image titles show ground truth, predicted class for the bias map, and the frequency of the noise patterns classified as that digit. (b) Classification images of logistic regression over MNIST with 92.46% test accuracy. (c) Confusion matrices of four classifiers (CNN and log. reg. biases, mean digit image, and log. reg. weights). The classification was done via template matching using the dot product.

## I. Understanding and visualizing classifier biases

We trained a CNN with 2 *conv* layers, 2 *pooling* layers, and one *fully connected* layer on the MNIST dataset. This CNN achieves 99.2% test accuracy. We then generated 1 million  $28 \times 28$  white noise images and fed them to CNN. The average noise map for each digit class is shown in fig. 6.2a. These biases/templates illustrate the regions that are important for classification. Surprisingly, for some digits (0 to 7), it is very easy to tell which digit the bias map represents<sup>5</sup>. We notice that most of the noise patterns are classified as 8, perhaps because this digit has a lot of structure in common with other digits. Feeding the average noise maps back to CNN, they are classified correctly, except 8, which is classified as 2 (see image captions in fig. 6.2a).

Classification images of the CNN over MNIST perceptually make sense to humans.

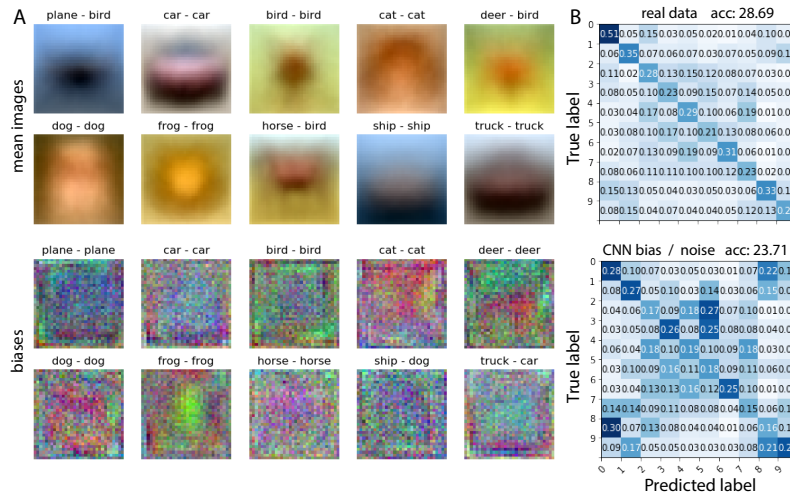
<sup>5</sup>Weighting the noise patterns by their classification confidence or only considering the ones with classification confidence above a threshold did not result in significantly different classification images.



**Figure 6.3:** Classification images for a two-layer MLP ( $784 \rightarrow 1000 \rightarrow 10$ ) shown at the top and an RNN classifier at the bottom. None of the noise patterns were classified as 1 using both classifiers. While the derived biases do not resemble digits, they still convey information to predict the class of a test digit.

This, however, does not necessarily hold across all classifiers and datasets. For example, classification images of a logistic regression classifier on MNIST, shown in fig. 6.2b, do not resemble digits (the same happens to MLP and RNN; see fig. 6.3). This implies that perhaps CNNs extract features the same way the human visual system does, thus sharing similar mechanisms and biases with humans. Classification images over the CIFAR-10 dataset, derived using 1 million  $32 \times 32$  RGB noise patterns, are shown in fig. 6.4a. In contrast to MNIST and Fashion-MNIST (fig. 6.7), classification images on CIFAR-10 (using CNNs) do not resemble target classes. One possible reason might be that images are more calibrated and aligned over the former two datasets than CIFAR-10 images.

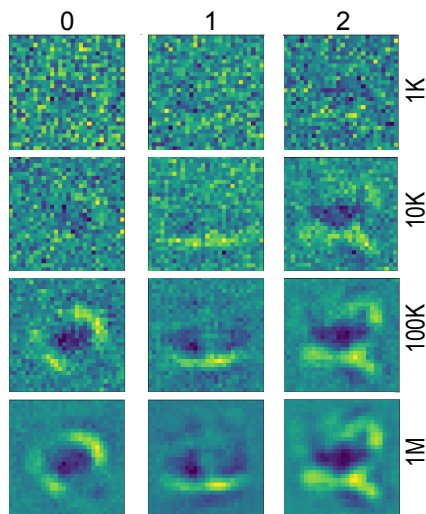
How much information do the classification images carry? To answer this question, we used bias maps to classify the MNIST test digits. The bias map with the maximum dot product to the test digit determines the output class. The confusion matrix of this classifier is shown in fig. 6.2c. Using the CNN bias map as a classifier leads to 25.8% test accuracy. The corresponding number for a classifier made of logistic regression bias is 47.6%. Both of these numbers are significantly above 10% chance accuracy. To get an idea regarding the significance of these numbers, we repeated the same using the mean



**Figure 6.4:** (a) Mean training images (top) and mean white noise pattern/bias maps (bottom) across CIFAR-10 classes. Image titles show the ground truth class and prediction of the bias map, respectively. (b) Confusion matrices using mean images (top) and bias maps (bottom) as classifiers, respectively. Notice that for some classes, it is easier to guess the class label from the mean image (e.g., frog).

images and logistic regression weights. These two classifiers lead to 63.1% and 83.8% test accuracy, respectively, which are better than the above-mentioned results using bias maps but demand access to the ground-truth data and labels. Over CIFAR-10, classification using bias maps leads to 23.71% test accuracy, which is well above chance. Using the mean training images of CIFAR-10 leads to 28.69% test accuracy (fig. 6.4b).

**Creating visual noise from natural scene statistics** We followed [163] to generate noise patterns containing subtle structures. We amassed a digit database of 60K images from the MNIST training set and represented each image as the output of a bank of Gabor filters at three spatial scales (2, 4, and 10 cycles per image, and wavelets were truncated to lie within the borders of the image), four orientations (0, 45, 90 and 135 degrees) and two quadrature phases (0 and 90 degrees). Thus, each image is represented by  $2 \times 2 \times 2 \times 4 + 4 \times 4 \times 2 \times 4 + 10 \times 10 \times 2 \times 4 = 960$  total Gabor wavelets. The weights of Gabor wavelets for each image were determined using ridge regression. We



**Figure 6.5:** Progressive build-up of the bias maps for 0, 1, and 2.

then performed principal components analysis (PCA) on the 60K-image by the 960-wavelet weight matrix. We kept the first 250 principal components that explain 96.1% of the variance in data. A noise image was created by choosing a random value for each principal component score, scaled to the observed range for each component.

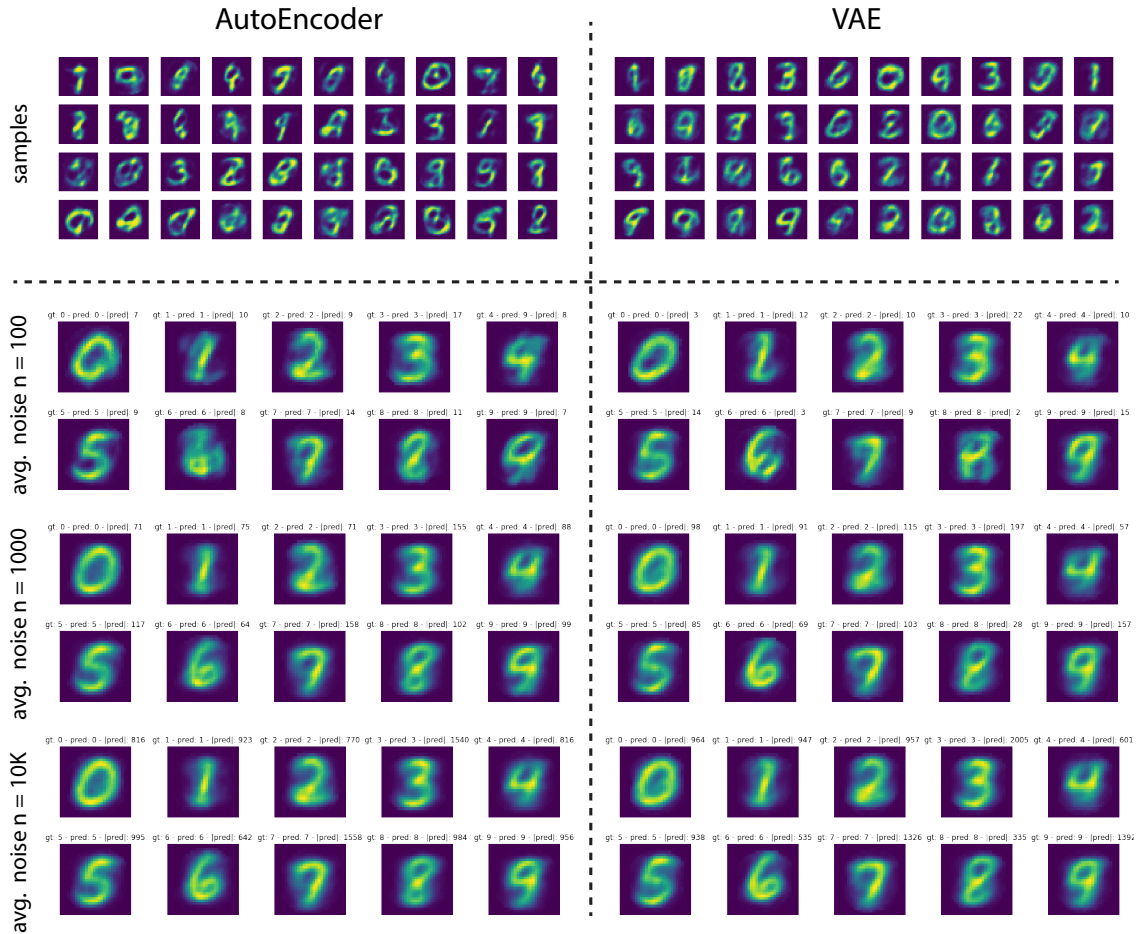
We also gathered a natural object database of 50K images from the CIFAR-10 training set. For these colored images, we performed the above-mentioned approach on each channel. More specifically, we represented each  $32 \times 32$ -sized channel with four-scale (2, 4, 7, 11 cycles), four-orientation, two-phase Gabor wavelets, which results in  $2 \times 2 \times 2 \times 4 + 4 \times 4 \times 2 \times 4 + 7 \times 7 \times 2 \times 4 + 11 \times 11 \times 2 \times 4 = 1520$  total Gabor wavelets per channel. Then for each channel, we performed ridge regression to get the 50K-by-1520 weight matrix, which is passed to PCA and kept the first 600 PCs. These PCAs can explain variance in three channels as 97.57%, 97.51%, and 97.52%, respectively.

**Analysis of sample complexity** To get an idea regarding the sample complexity of the classification images approach, we ran three analyses. In the first one, we varied the number of noise patterns as  $n = 1000 \times k; k \in \{1, 10, 100, 1000\}$ . We found that

with 10K noise stimuli, the bias maps already start to look like the target digits (see fig. 6.5). In the second analysis, we followed [163] to generate noise patterns containing subtle structures. Over MNIST and Fashion-MNIST datasets, we used ridge regression to reconstruct all 60K training images from a set of 960 Gabor wavelets, as described above. We then projected the learned weights (a matrix of size  $60K \times 960$ ) to a lower-dimensional space using principal component analysis (PCA). We kept 250 components that explained 96.1% of the variance. To generate a noise pattern, we randomly generated a vector of 250 numbers and projected it back to the 960D space, using them as weights for image-shaped Gabor wavelets and then summing them to  $28 \times 28$  noise image. Over CIFAR-10, we used 1520 Gabor filters for each RGB channel and kept 600 principal components that explained 97.5% of the variance. Classification images using 1M samples generated this way for MNIST, Fashion-MNIST, and CIFAR-10 datasets are shown in fig. 6.7. Classification images resemble the target classes even better now (compared to using white noise). Using the new bias maps for classification, we are able to classify MNIST, Fashion-MNIST, and CIFAR-10 test data with 35.5%, 41.21%, and 21.67% accuracy, respectively.

In the third analysis, we trained an autoencoder and a variational autoencoder [41] over MNIST, only for 2 epochs. We did so to make the encoders powerful just enough to produce images that contain subtle digit structures (fig. 6.6). As expected, now the classification images can be computed with a much less number of stimuli ( $\sim 100$ ). Results from these analyses suggest that it is possible to lower the sample complexity when some (unlabeled) data is available. This is, in particular, appealing for practical applications of classification images.

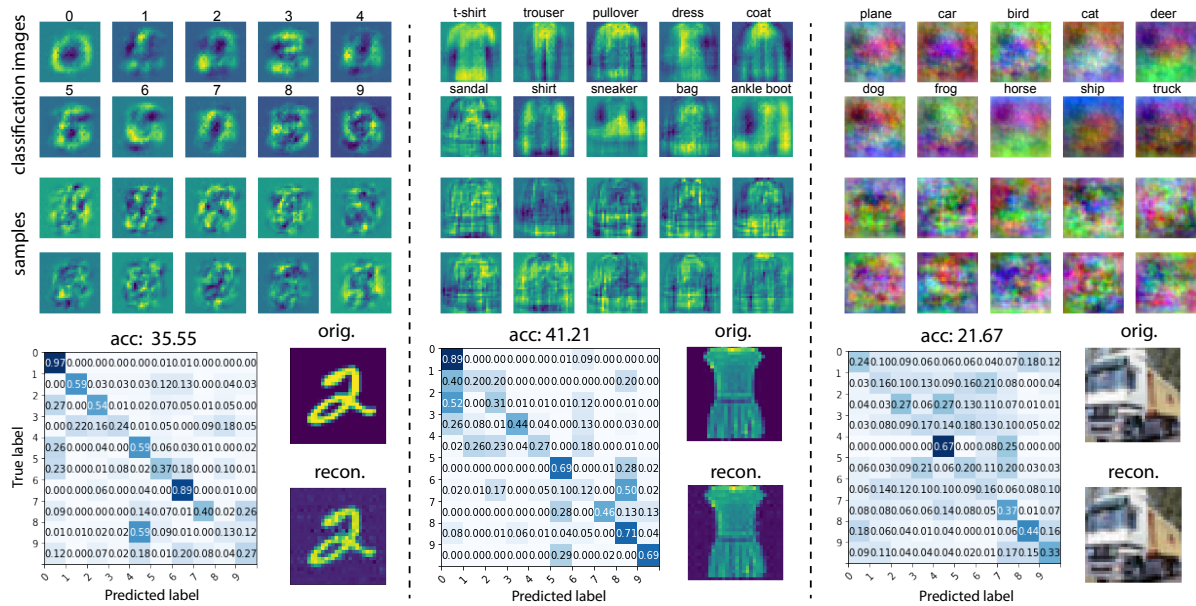
**Results on ImageNet** We conducted an experiment on ImageNet validation set including 50K images covering 1000 categories and 1 million samples using Gabor PCA



**Figure 6.6:** Using an AutoEncoder and a VAE to generate samples containing faint structures to be used for computing the classification images over MNIST dataset, using a CNN classifier. Both generators were trained only for two epochs to prohibit the CNN from generating perfect samples (shown at the top). The bottom panels show classification images derived using 100, 1K, and 10K samples from each generator. Note that classification images converge much faster now compared with the white noise stimuli.

**Table 6.1:** Results on ImageNet.

backbone	accuracy	run time	empty classes
ResNet152	0.00180	2:15	564
ResNet101	0.00152	1:36	539
densenet201	0.00118	2:12	998
squeezenet1_1	0.00104	0:13	999
googlenet	0.00102	0:26	999
mnasnet1_3	0.00082	0:32	922
vgg_19_bn	0.00074	1:51	994



**Figure 6.7:** Classification images, some sample generated images, confusion matrices of bias map classifiers, as well as one sample image and its reconstruction using Gabor wavelets over MNIST (left), Fashion-MNIST (middle), and CIFAR-10 (right) datasets. We used 960, 960, and 1520 Gabor wavelets over MNIST, Fashion-MNIST, and CIFAR-10, respectively. The corresponding numbers of PCA components are 250, 250, and 600 (per color channel).

sampling (from the above CIFAR-10 experiment over CIFAR-10 images) and pretrained CNNs (on ImageNet train set). As results in table 6.1 show, even with 1M samples and without parameter tuning, we obtain an improvement over the chance level (0.0010 or 0.1%). We obtain about 2x accuracy than chance using ResNet152 [148]. It seems that 1M samples are not enough to cover all classes since no noise pattern is classified under almost half of the classes using ResNet152. For some backbones, even a larger number of classes remain empty: this provides another evidence that white noise can reveal biases in models. We believe it is possible to improve these results with more samples. As you can see with more classes being filled, better accuracy can be achieved. It takes only a few minutes (about 2) to process all 1M images at  $32 \times 32$  resolution using a single GPU. Notice that ImageNet models have been trained on  $224 \times 224$  images, while here



**Table 6.2:** Numbers corresponding to the bar charts in fig. 6.8c.

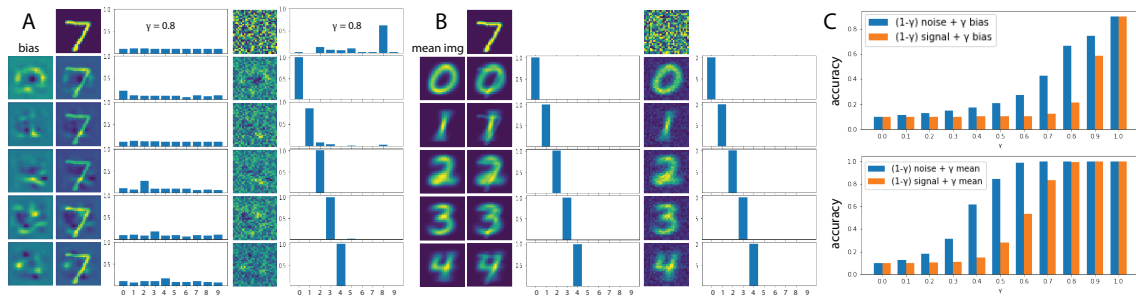
$\gamma$	0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1
$(1 - \gamma) \times \textit{noise} + \gamma \times \textit{bias}$	0.1	0.113	0.127	0.149	0.174	0.207	0.271	0.429	0.666	0.743	0.9
$(1 - \gamma) \times \textit{signal} + \gamma \times \textit{bias}$	0.1	0.1	0.1	0.1	0.101	0.102	0.105	0.123	0.214	0.587	0.9
$(1 - \gamma) \times \textit{noise} + \gamma \times \textit{mean}$	0.1	0.127	0.179	0.313	0.618	0.842	0.986	1.0	1.0	1.0	1.0
$(1 - \gamma) \times \textit{signal} + \gamma \times \textit{mean}$	0.1	0.101	0.103	0.109	0.149	0.28	0.534	0.83	0.994	1.0	1.0

we test them on  $32 \times 32$  noise images for the sake of computational complexity. A better approach would be to train the models on  $32 \times 32$  images or feed the noise at  $224 \times 224$  resolution. This, however, demands more computational power but may result in better performance.

Overall, our pilot investigation on large-scale datasets is promising. We believe better results than the ones reported in table 6.1 are possible with further modifications (e.g., using better distance measures between an image and the average noise map for each class). Also, it is likely that increasing the number of samples will lead to better performance.

## II. Adversarial attack and defense

Deep neural networks achieve remarkable results on various visual recognition tasks. They are, however, highly susceptible to being fooled by images that are modified in a particular way (so-called adversarial examples). Interesting adversarial examples are the ones that can confuse a model but not a human (i.e., imperceptible perturbations). Likewise, it is also possible to generate a pattern that is perceived by a human as noise but is classified by a network as a legitimate object with high confidence [164]. Beyond the security implications, adversarial examples also provide insights into the weaknesses, strengths, and blind spots of models.



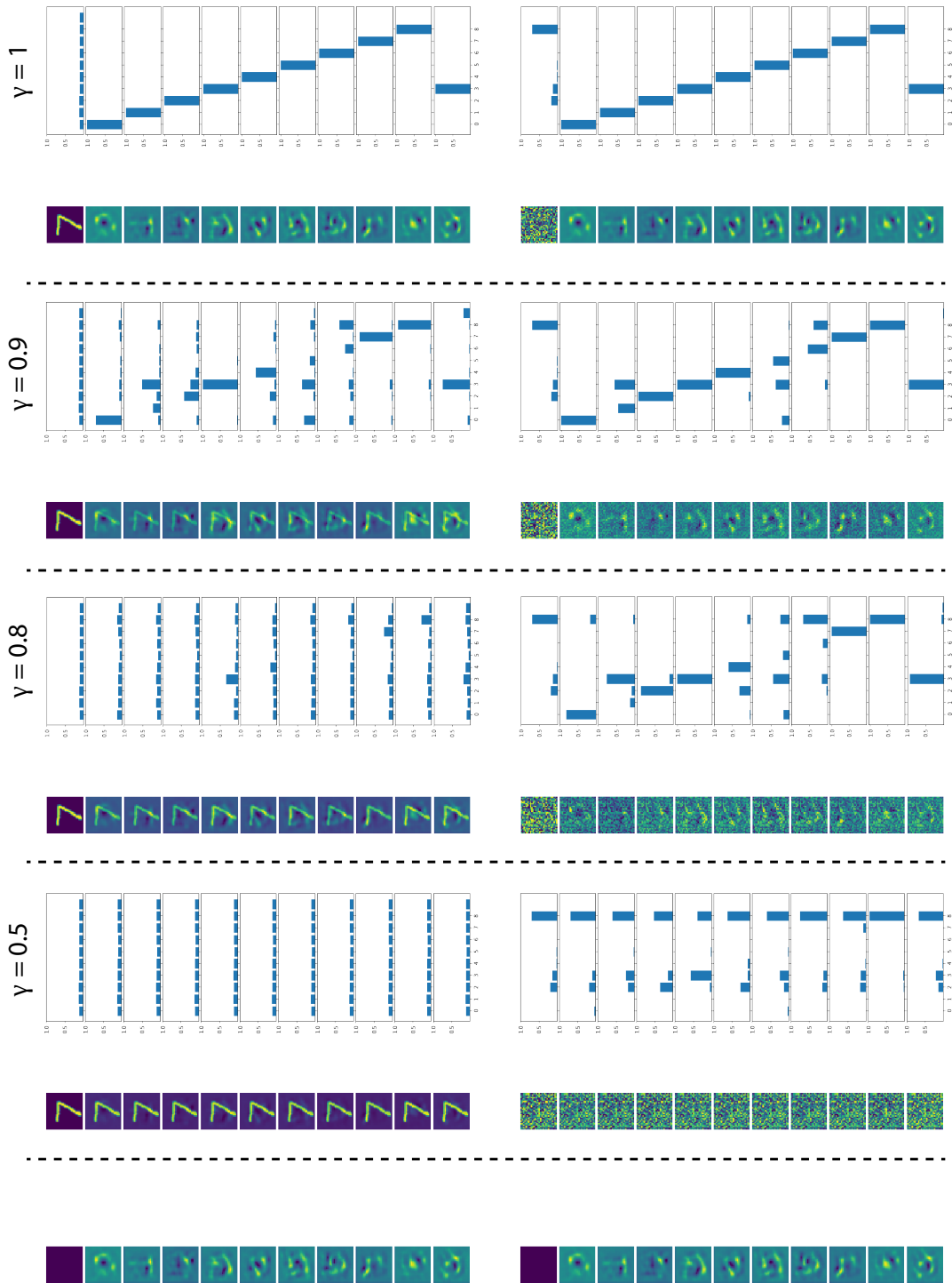
**Figure 6.8:** (a) Adding bias to a digit changes it to the target class in many cases (here with  $\gamma = 0.8$ ). Adding bias to noise (2nd col.) turns noise into the target digit in almost all cases. The histograms show the distribution of predicted classes (intact digits or pure noise; 1st row). Note that most of the noise images are classified as 8 (top histogram in 2nd col). (b) Same as A but using mean digit (computed over the training set). Adding the mean image is more effective but causes a much more perceptible perturbation. (c) The degree to which (i.e., accuracy) a stimulus is classified as the target class (i.e., fooled) by adding different magnitudes of bias (or mean image) to it. Converting noise to a target is easier than converting a signal. There is a trade-off between perceptual perturbation and accuracy (i.e., subtle bias leads to less number of digits being misclassified).

**Adversarial attack** A natural application of the bias maps is to utilize them to influence a black-box system, in targeted or un-targeted manners, by adding them to the healthy inputs. Over MNIST, we added different magnitudes of bias maps (controlled by  $\gamma$ ; Eq. 3) to the input digits and calculated the misclassification accuracy or fooling rate of a CNN (same as the one used in the previous subsection). This is illustrated in fig. 6.8a. Obviously, there is a compromise between the perceptibility of perturbation (i.e., adding bias) and the fooling rate. With  $\gamma = 0.8$ , we are able to manipulate the network to classify the augmented digit as the class of interest 21% of the time (fig. 6.8c; chance is 10%). In comparison, adding the same amount of the mean image to digits fools the network almost always but is completely perceptible. In a similar vein, we are able to convert noise to a target digit class by adding bias to it (fig. 6.8b). With  $\gamma = 0.5$ , which is perceptually negligible (fig. 6.9), we can manipulate the network 20.7% of the time. Notice that in contrast to many black-box adversarial attacks that demand access to logits or gradients, our approach only requires the hard labels and does not make any

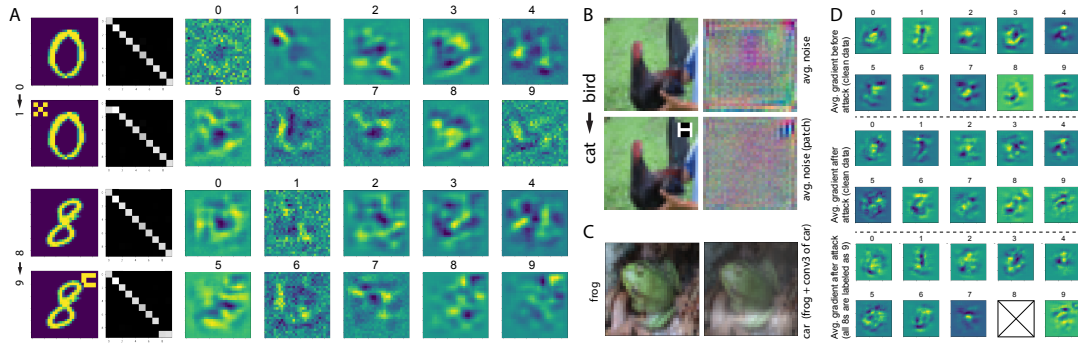
assumption regarding the input distribution.

**Adversarial defense** In a recent work, [165] introduced a technique called adversarial patch as a backdoor attack on a neural network. They placed a particular type of pattern on some inputs and trained the network with the poisoned data. The patches were allowed to be visible but were limited to a small, localized region of the input image. Here, we explore whether and how classification images can be used to detect adversarial patch attacks.

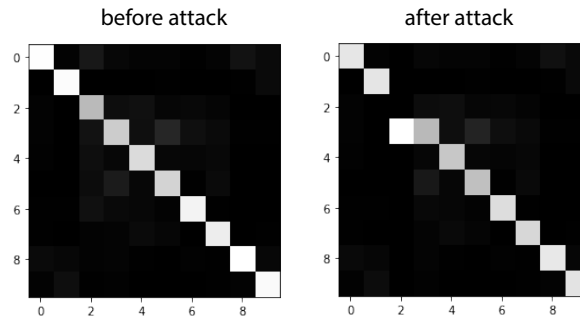
We performed three experiments, two on MNIST and one on CIFAR-10 (fig. 6.10). Over MNIST, we constructed two training sets as follows. In the first one, we took half of the 0s and placed a  $3 \times 3$  patch (x-shape) on their top-left corner and relabeled them as 1. The other half of the zeros and all other digits remained intact. In the second one, we placed a c-shape patch on the top-right corner of half of the 8s, relabeled them as 9, and left the other half and other digits intact. We then trained two 10-way CNNs, the same architecture as in the previous subsection, on these training sets. The CNNs perform close to perfect on the healthy test sets. Over a test set with all zeros contaminated (or eights), they completely misclassify the perturbed digits (See confusion matrices in the 2nd and 4th rows of fig. 6.10a). Computing the classification images for these classifiers, we find a strong activation at the location of the adversarial patches in both cases. Note that the derived classification images still resemble the ones we found using the un-attacked classifiers (fig. 6.2a) but now new regions pop out. Over CIFAR-10, we placed an H-shape pattern on the top-right of half of the birds and labeled them as cats. The trained CNN classifier performs normally on a clean dataset. Again, computing the bias unveils a tamper in the network (fig. 6.10b). To verify these findings, we computed the average gradient of the classification loss with respect to the input image for intact and attacked networks over the healthy and tampered training sets (fig. 6.10). The average



**Figure 6.9:** Illustration of influencing the CNN decisions (on MNIST) towards a particular digit class by adding bias to the digits (top) and adding bias to the noise (bottom). This is akin to a targeted attack. See fig. 6.10.

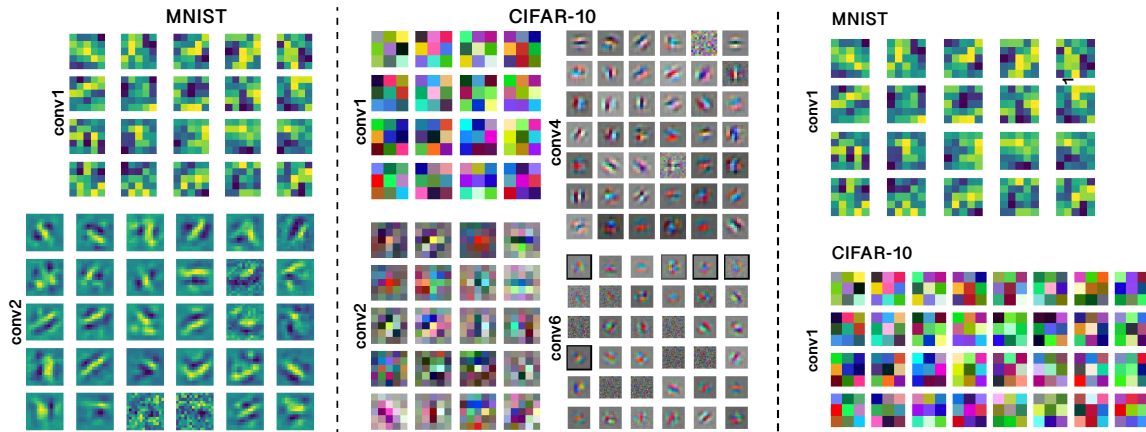


**Figure 6.10:** (a) Top: A 10-way CNN trained on MNIST (with half of the zeros augmented with a patch and relabeled as 1) performs very well on a clean test set (top confusion matrix). On a test set containing all zeros contaminated, it (incorrectly) classifies them as one. Classification images (right side) successfully reveal the perturbed region. Bottom: Same as above but over 8 and 9 digits. (b) Classification images reveal the adversarial patch attack over CIFAR-10. Here, half of the birds are contaminated with a patch and are labeled as cat. (c) Turning a frog into a car by adding the activation of the *conv6* layer, computed using white noise, of the car category to the frog. (d) Average gradients before the adversarial patch attack (top) and after the attack (middle). The small yellow region on the top-right of digit 8 means that increasing those pixels increases the loss and thus leads to misclassification (i.e., turns 8 to another digit). (bottom) Average gradient with all 8s contaminated and relabeled as 9. The blue region on the top-right of digit 9 means that increasing those pixels lowers the loss and thus leads to classifying a digit as 9. This analysis is performed over the MNIST training set. Please see also figs. 6.11 and 6.15.



**Figure 6.11:** Confusion matrices for adversarial patch attack on CIFAR-10 dataset (bird to cat). Class names: plane, car, bird, cat, deer, dog, frog, horse, ship, and truck.

gradient shows a slight activation at the location of the perturbation (fig. 6.10d), but it is not as pronounced as results using bias images.



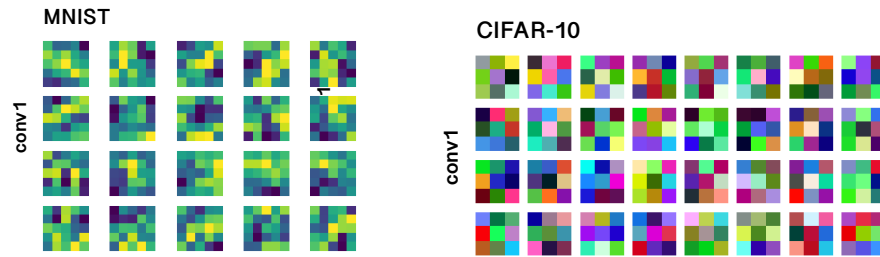
**Figure 6.12:** Left two: example filters derived using spike-triggered averaging (STA) for the first two *conv* layers of a CNN trained on MNIST dataset (left; RF sizes are  $5 \times 5$  and  $14 \times 14$ ) and 4 layers of a CNN on CIFAR-10 dataset (middle; RF sizes in order are  $3 \times 3$ ,  $5 \times 5$ ,  $14 \times 14$  and  $32 \times 32$ ). Right: Trained model weights (i.e., convolutional kernels) of the first layer of a CNN trained on MNIST or CIFAR-10. These are not calculated by feeding noise patterns. They are derived after training the model on data. Interestingly, they are the same as those derived using white noise.

### III. Filter visualization

A number of ways have been proposed to understand how neural networks work by visualizing their filters [166]. Example approaches include plotting filters of the first layers, identifying stimuli that maximally activate a neuron, occlusion maps by masking image regions [167], activation maximization by optimizing a random image to be classified as an object [168], saliency maps by calculating the effect of every pixel on the output of the model [169], network inversion [170], and network dissubsection [171].<sup>6</sup> Here, we propose a new method based on spike-triggered averaging.

For each model, we fed 1 million randomly generated patterns to the network and recorded the average response of single neurons at different layers. We changed the activation functions in the convolution layers of the CIFAR-10 CNN model to tanh, as using ReLU activation resulted in some dead filters. fig. 6.12 shows the results over MNIST

<sup>6</sup>See also <https://captum.ai/docs/algorithms>.

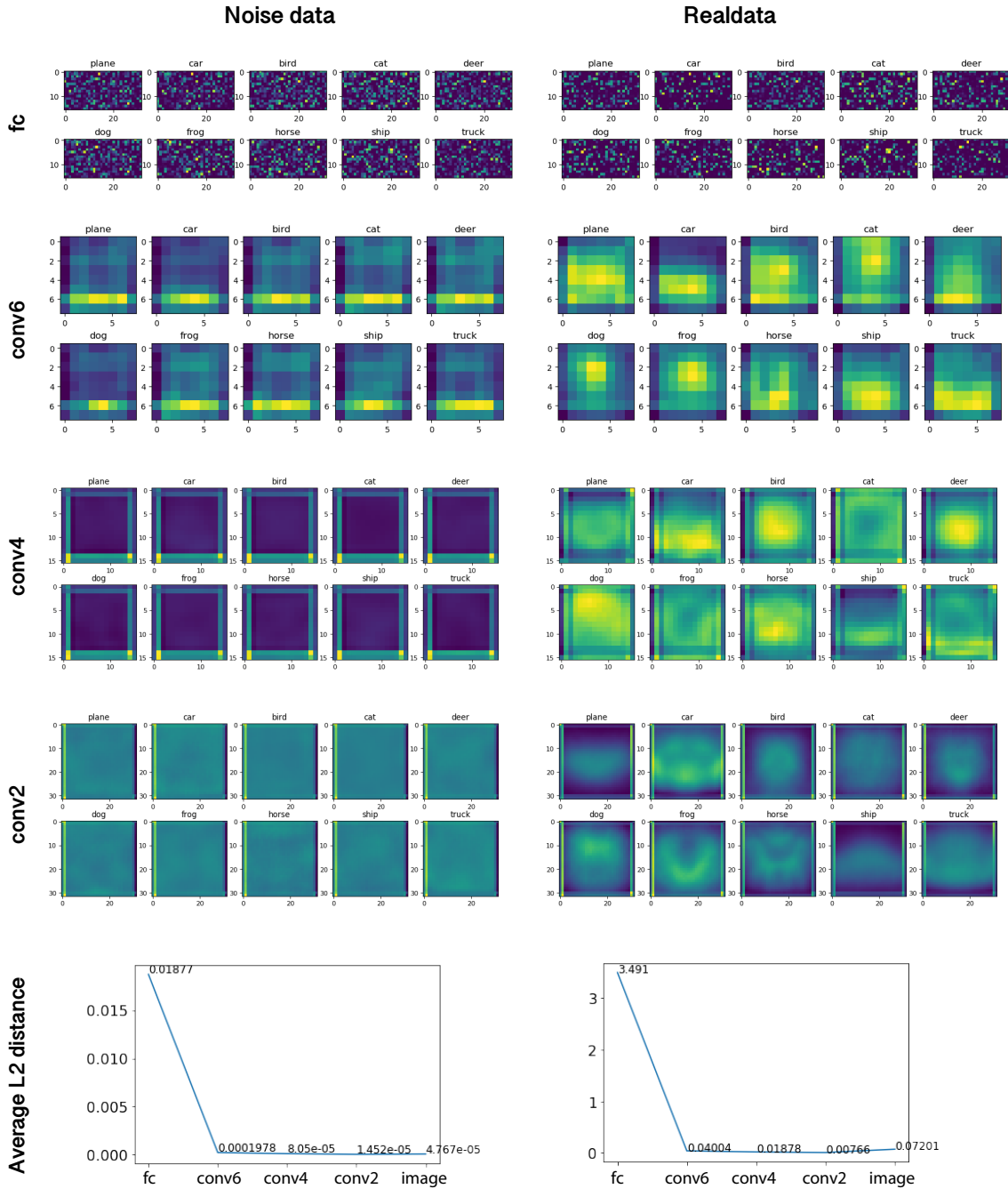


**Figure 6.13:** Trained model weights (i.e., convolutional kernels) of the first layer of a CNN trained on MNIST or CIFAR-10. These are not calculated by feeding noise patterns. They are derived after training the model on data. Interestingly, they are the same as those derived using white noise as shown in fig. 6.12.

and CIFAR-10 datasets. We also show the filters computed using real data for the sake of comparison in the fig. 6.13. As can be seen, filters extract structural information (e.g., oriented edges) and are similar to those often derived by other visualization techniques. Comparing derived filters using noise patterns and derived filters using training on real data (i.e., kernel weights), we notice that the two are exactly the same. This holds over both MNIST and CIFAR-10 datasets.

Next, for the CIFAR-10 model, we computed mean layer activation maps of *conv2*, *conv4*, *conv6*, and *fc* layers by sending noise through the network. Results are shown in fig. 6.14. Comparing these maps with the mean activation maps derived using real data, we observe a high similarity in the *fc* layer and relatively less similarity in the other layers. The high similarity in the *fc* layer is because it is immediately before the class decision layer, and thus for a noise pattern to fall under a certain class, it has to have a similar weight vector as the learned weights from real data. This is corroborated by the higher average L2 distance across different classes in the *fc* layer, compared to the other layers, over both noise and real data (bottom panel in fig. 6.14).

We then asked whether it is possible to bias the network towards certain classes (similar to the adversarial analysis in fig. 6.8) by injecting information, learned from average noise patterns to the input image or its activation maps at different layers. For



**Figure 6.14:** (Top) Average layer activation using noise (left) and real data (right) over a CNN trained on CIFAR-10 dataset. (Bottom) Mean distance between average layer activations of different classes across model layers.



example, as shown in fig. 6.10c, we can turn a frog into a car by adding the average *conv6* activation of the noise patterns classified as a car to it. This can be done in a visually (almost) imperceptible manner. Results over other classes of CIFAR-10 and different activation layers are shown in fig. 6.15. For some classes (e.g., cat or bird), it is easy to impact the network, whereas for some others (e.g., horse) it is harder. Results indicate that for different objects, different layers have more influence on classification.

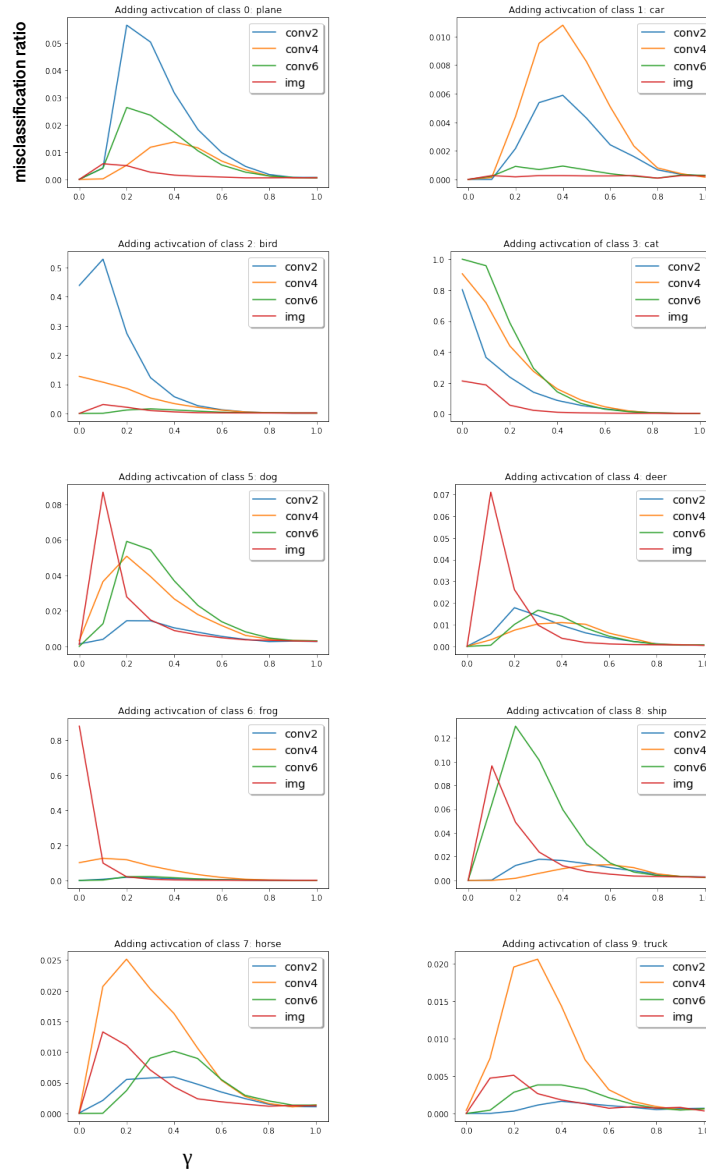
#### IV. Micro-stimulation

Microstimulation, the electrical current-driven excitation of neurons, is used in neurophysiology research to identify the functional significance of a population of neurons [172, 173]. Due to its precise temporal and spatial characteristics, this technique is often used to investigate the causal relationship between neural activity and behavioral performance.

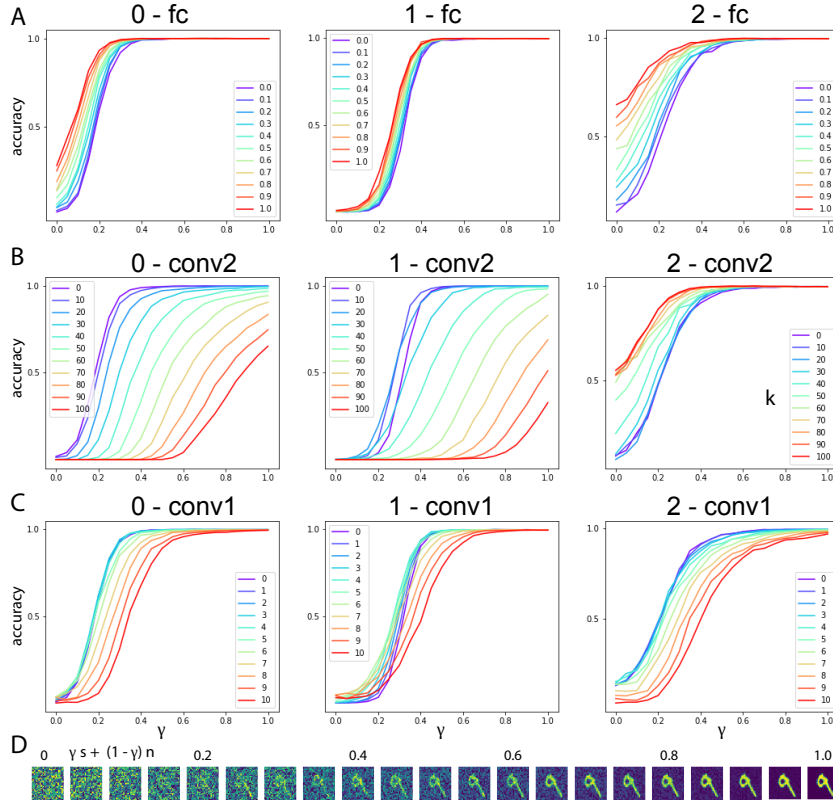
It has also been employed to alleviate the impact of damaged sensory apparatus and build brain-machine interfaces (BMIs) to improve the quality of life of people who have lost the ability to use their limbs.

For example, stimulation of the primary visual cortex creates flashes of light which can be used to restore some vision for blind people. Microstimulation has been widely used to study visual processing across several visual areas, including MT, V1, V4, IT, and FEF [174]. Here, we investigate how augmenting the stimuli with white noise impacts the internal activations of artificial neural networks and their outputs.

We linearly combined signal and white noise, according to Eq. 3, and measured the classification accuracy of a CNN trained on MNIST (fig. 6.16). Without any stimulation, with the original network biases and weights, increasing the amount of signal (shown on the x-axis) improves the accuracy from 0 (corresponding to 100% noise) to 1 (corresponding to 100% signal). The resulting S-shaped curve resembles the psycho-



**Figure 6.15:** Effect of adding activation at *conv6*, *conv4*, *conv2* and input of noises classified as different classes to real images. The figure shows CIFAR-10 model misclassification ratio vs.  $\gamma$ , where the input to the model is  $((1 - \gamma) \times \text{noise activation of a certain class} + \gamma \times \text{real data input image})$ . The misclassification ratio is calculated as the number of images that do not belong to the activation-added class but are classified as it over the number of images not belonging to the activation-added class. The visualization of adding activation to input is shown in fig. 6.10c.



**Figure 6.16:** Psychometric curves of a CNN trained on MNIST. The x-axis shows the magnitude of the signal added to the noise ((d)). The y-axis shows the accuracy. Legends show the magnitude of stimulation ( $k$  in Eq. 6). Larger  $k$  (redder curve) means more bias. (a) Increasing  $fc$  bias enhances recognition towards the target digit for all digits. The opposite happens when lowering the bias. (b)(c) Stimulating neurons in  $conv$  layers helps some digits (for which those neurons are positively correlated) but hinders some others.

metric functions observed in human psychophysics experiments [175]. We then varied the amount of network bias in different layers according to the following formula and measured the accuracy again:

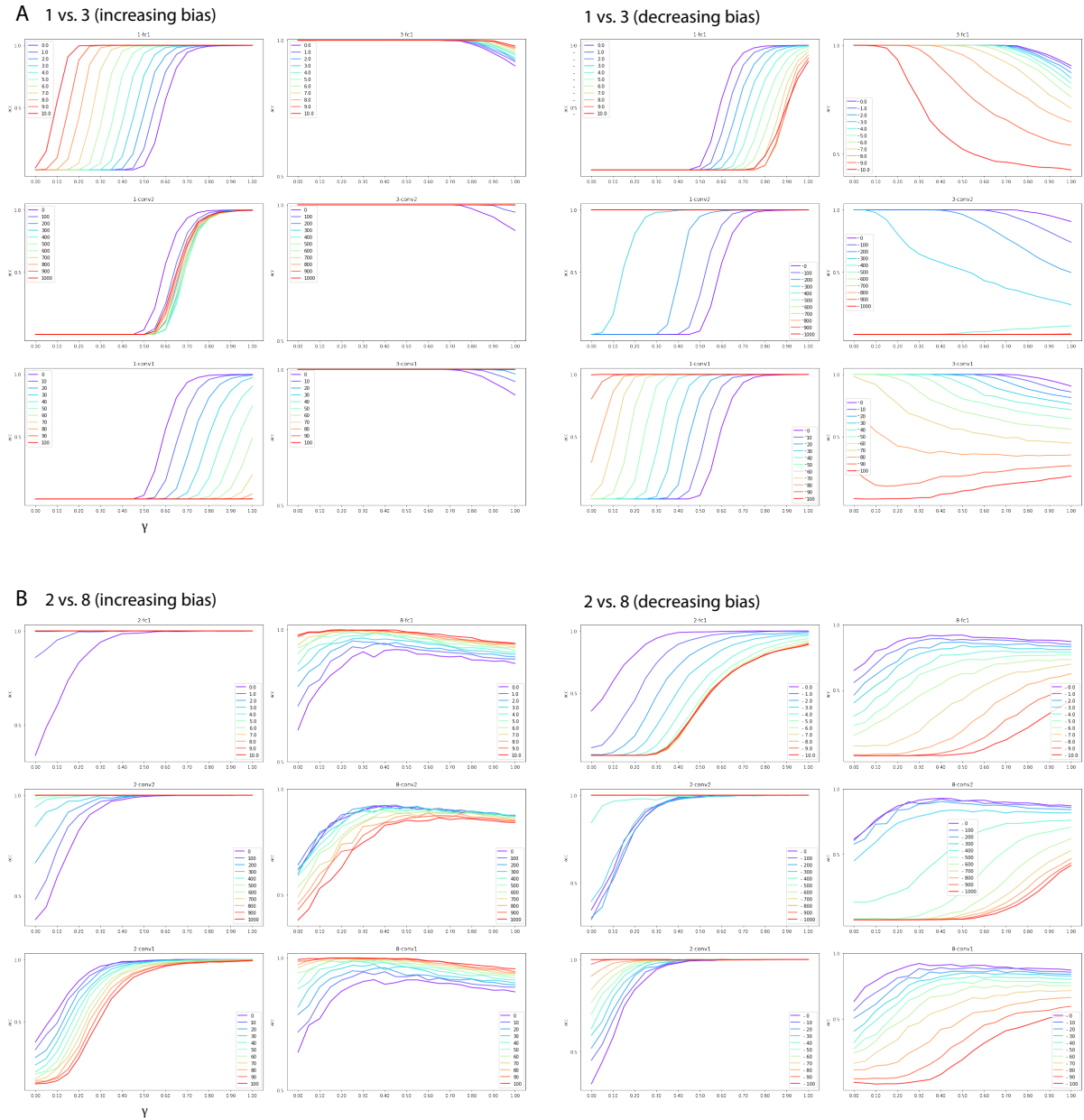
$$b_{ml}^{new} = b_{ml}^{old} + \lambda_l \times k \times \frac{1}{\max(a_{ml})} \sum_{i=1}^N a_{mli} \quad (6.6)$$

where  $b_{ml}$  is the bias term for map  $m$  in layer  $l$ , and  $a_{mli}$  is the activation of neuron  $i$  at the  $m$ th map of the  $l$ th layer.  $k$  controls the magnitude of stimulation.  $\lambda_l$  is used to scale

the activation values, since sensitivity of the output to neurons at different layers varies (here we use  $\lambda_l = 0.01, 0.1, 1$  for *fc*, *conv1*, and *conv2*, respectively). Bias term ( $b_{ml}$ ) is shared across all neurons in a map (i.e., for the same kernel). Notice that increasing bias in Eq. 6 is proportional to the map activation. Thus, stimulation has a higher impact on more active (selective) neurons.

Increasing the bias of *fc* neurons shifts the psychometric function to the left. This means that for the same amount of noise as before (i.e., no stimulation), now CNN classifies the input more frequently as the target digit. In other words, the network thinks of noise as the digit. Increasing *fc* biases consistently elevates accuracy for all digits. Conversely, reducing the *fc* bias shifts the psychometric function to the right for all digits (i.e., using minus sign in Eq. 6). The effect of stimulation on convolutional layers is not consistent. For example, increasing *conv2* bias shifts the curves to the right for 0 and 1, and to the left for 3. We observed that stimulation or inhibition of *conv1* layer almost always hurts all digits. We speculate this might be because *conv1* filters capture features that are shared across all digits, and thus a subtle perturbation hurts the network.

We were able to replicate the above results using a binary CNN akin to yes/no experiments on humans or monkeys. Results are provided in fig. 6.17. Our findings qualitatively agree with the results reported in [176]. They artificially stimulated clusters of IT neurons while monkeys judged whether noisy visual images were ‘face’ or ‘non-face’. Microstimulation of face-selective neurons biased the monkeys’ decisions towards the face category.



**Figure 6.17:** Results of microstimulation for binary decision-making tasks using a CNN classifier (1 vs. 3) and (2 vs. 8). Left(right) panels show increasing (decreasing) bias for each layer. See fig. 6.16.

### 6.1.4 Discussions and Conclusion

We showed that white noise analysis is effective in unveiling hidden biases in deep neural networks and other types of classifiers. A drawback is the need for a large number of trials. To lower the sample complexity, we followed the approach in [163] and also recruited generative models. As a result, we were able to lower the sample complexity dramatically. As another alternative, [152] used the Hoggles feature inversion technique [177] to generate images containing subtle scene structures. Their computed bias maps roughly resembled natural scenes. We found that the quality of the bias maps highly depends on the classifier type and the number of trials. Also, classification images over natural scene datasets are not expected to look like the instances of natural images since even the mean images do not represent sharp objects (see figs. 6.2 and 6.7). In this regard, spike-triggered covariance can be utilized to find stimuli (eigenvectors) to which a network or a neuron responds [159].

We foresee several avenues for future research. We invite researchers to employ the tools developed here to analyze even more complex CNN architectures including ResNet [148] and InceptionNet [178]. They can also be employed to investigate biases of other models such as CapsuleNets [179] and GANs [180], and to detect and defend against other types of adversarial attacks. The outcomes can provide a better understanding of the top-down processes in deep networks, and the ways they can be integrated with bottom-up processes. Moreover, applying some other methods from experimental neuroscience [181] (e.g., lesioning, staining) and theoretical neuroscience (e.g., spike-triggered non-negative matrix factorization [182], Bayesian STC [161], and Convolutional STC [183]) to inspect neural networks is another interesting future direction. Using classification images to improve the accuracy of classifiers (as in [152]) or their robustness (as was done here) are also promising directions.

This work focused primarily on visual recognition. On the other hand, [184] used classification images to estimate the template that guides saccades during the search for simple visual targets, such as triangles or circles. [185] measured temporal classification images to study how the saccadic targeting system integrates information over time. [186] utilized classification images to investigate the perception of illusory and occluded contours. Inspired by these works, classification images, and STA can be applied to other computer vision tasks such as object detection, edge detection, activity recognition, and segmentation. Finally, unveiling biases of complicated deep networks can be fruitful in building bias-resilient and fair ANNs (e.g., racial fairness).

In summary, we utilized two popular methods in computational neuroscience, classification images and spike-triggered averaging, to understand and interpret the behavior of artificial neural networks. We demonstrated that they bear value for practical purposes (e.g., solving challenging issues such as adversarial attacks) and for further theoretical advancements. More importantly, our efforts show that confluence across machine learning, computer vision, and neuroscience can benefit all of these fields.

## 6.2 Sparsifying DNNs: Fat-Trimming MLP-like Models

Information in the brain is represented with the sparse coding property [187]: items trigger strong activation of a relatively small set of neurons. For different stimuli, a different subset of all available neurons activate. This section takes insights from this property. However, instead of having different activations for different inputs, we bring the sparsity into DNN by activating partial channels, thus achieving sparse activation of artificial neurons. In what follows, we present SplitMixer, a simple and lightweight isotropic MLP-like architecture, for visual recognition. It contains two types of interleaving convolutional operations to mix information across spatial locations (spatial mixing) and channels (channel mixing). The first one includes sequentially applying two depthwise 1D kernels, instead of a 2D kernel, to mix spatial information. The second one is splitting the channels into overlapping or non-overlapping segments, with or without shared parameters, and applying our proposed channel mixing approaches or 3D convolution to mix channel information. Depending on design choices, a number of SplitMixer variants can be constructed to balance accuracy, the number of parameters, and speed. We show, both theoretically and experimentally, that SplitMixer performs on par with the state-of-the-art MLP-like models while having a significantly lower number of parameters and FLOPS. For example, without strong data augmentation and optimization, SplitMixer achieves around 94% accuracy on CIFAR-10 with only 0.28M parameters, while ConvMixer achieves the same accuracy with about 0.6M parameters. The well-known MLP-Mixer achieves 85.45% with 17.1M parameters. On the CIFAR-100 dataset, SplitMixer achieves around 73% accuracy, on par with ConvMixer, but with  $\sim 52\%$  fewer parameters and FLOPS. Our model also fares well over Flowers102, Food101, and ImageNet-1K datasets. We hope that our results spark further research toward finding more efficient



vision architectures and facilitating the development of MLP-like models.

### 6.2.1 Introduction

Architectures based exclusively on multi-layer perceptrons (MLPs) [188] have emerged as strong competitors to Vision Transformers (ViT) [189] and Convolutional Neural Networks (CNNs) [147, 148]. They achieve compelling performance on several computer vision problems, in particular large-scale object classification. Further, they are very simple and efficient, and perform on par with more complicated architectures. MLP-like models contain two types of layers to mix information across spatial locations (spatial mixing) and channels (channel mixing). These operations can be implemented via self-attention as in ViT, MLPs as in MLP-Mixer, or convolutions as in ConvMixer [2]. There is, in fact, a high degree of similarity among these models (section 6.2.5).

We propose the SplitMixer, a conceptually and technically simple, yet very efficient architecture in terms of accuracy, the number of required parameters, and computation. Our model is similar in spirit to the ConvMixer and MLP-Mixer models in that it accepts image patches as input, dissociates spatial mixing from channel mixing, and maintains equal size and resolution throughout the network, hence an isotropic architecture. Similar to ConvMixer, it uses standard convolutions to achieve the mixing steps. Unlike ConvMixer, however, it uses **1D** convolutions to mix spatial information. This modification maintains the accuracy but does not lower the number of parameters significantly. The biggest reduction in the number of parameters is achieved by how we modify channel mixing. Instead of applying  $1 \times 1$  convolutions across all channels, we apply them to channel segments that may or may not overlap each other. We implement this part with our ad-hoc solutions or with 3D convolution. This way, we find some architectures that are very frugal in terms of model size and computational needs and, at the same time,

exhibit high accuracy (See fig. 6.24).

Despite its simplicity, SplitMixer achieves excellent performance. For example, without strong data augmentations, it attains around 94% Top-1 accuracy on CIFAR10 with only 0.27M parameters and 71M FLOPS. ConvMixer achieves the same accuracy but with 0.59M parameters and 152M FLOPS (almost twice more expensive). MLP-Mixer can only achieve 85.45% with 17.1M parameters and 1.21G FLOPS. ResNet50 [148] achieves 80.76% using 23.84M parameters, and MobileNet attains 89.81% accuracy using 0.24M parameters.

Inspired by the extensive use of spatial separable convolutions and depthwise separable convolutions in the literature (e.g., MobileNet), in section 6.2.2, we propose to apply 1D depthwise convolution sequentially across width and height for spatial mixing, and split the channels into overlapping or non-overlapping segments and applying  $1 \times 1$  pointwise convolution to segments for channel mixing. Apart from theoretical analyses, we also provide empirical support for the computational efficiency of the proposed solution in section 6.2.3, together with model throughput measurement, and ablation studies to determine the contribution of different model components.

## 6.2.2 SplitMixer

The overall architecture of SplitMixer is depicted in fig. 6.18. It consists of a patch embedding layer followed by repeated applications of fully convolutional SplitMixer blocks. Patch embeddings with patch size  $p$  and embedding dimension  $h$  are implemented as 2D convolution with  $c$  input channels (3 for RGB images),  $h$  output channels, kernel size  $p$ , and stride  $p$ :

$$z_0 = \mathbf{N}(\sigma\{\text{Conv}_{c \rightarrow h}(I, \text{stride}=p, \text{kernel\_size}=p)\}) \quad (6.7)$$

where  $\mathbf{N}$  is a normalization technique (e.g., BatchNorm by [190]),  $\sigma$  is an element-wise nonlinearity (e.g., GELU by [191]), and  $\mathbf{I} \in \mathbb{R}^{n \times n \times c}$  is the input image. The SplitMixer block itself consists of two 1D depthwise convolutions (i.e., grouped convolution with groups equal to the number of channels  $h$ ) followed by several pointwise convolutions with kernel size  $1 \times 1$ . Each convolution is followed by nonlinearity and normalization <sup>7</sup>. Therefore, each block can be written as:

$$z'_i = \mathbf{N}(\sigma\{\text{ConvDepthwise}(z_{i-1})\}) \quad // \quad 1 \times k \text{ Conv across width} \quad (6.8)$$

$$z'_i = \mathbf{N}(\sigma\{\text{ConvDepthwise}(z'_i)\}) + z_{i-1} \quad // \quad k \times 1 \text{ Conv across height} \quad (6.9)$$

$$z_i = \mathbf{N}(\sigma\{\text{ConvPointwise}(z'_i)\}) \quad // \quad 1 \times 1 \text{ Conv across channels} \quad (6.10)$$

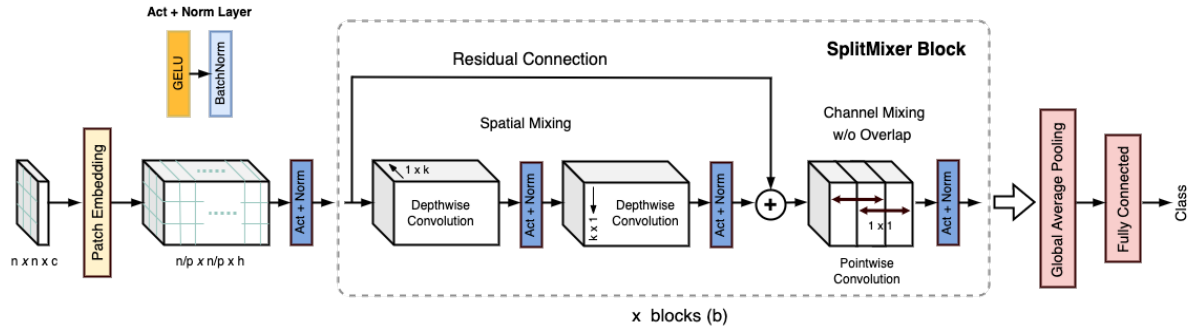
The SplitMixer block is applied  $b$  times (indexed by  $l$ ), after which global pooling is applied to obtain a feature vector of size  $h$ . Finally, a softmax classifier maps this vector to the class label. In what follows, we describe the spatial and channel mixing layers of the architecture.

## I. Spatial mixing

We replace the  $k \times k$  kernels<sup>8</sup> in ConvMixer by two 1D kernels: 1) a  $1 \times k$  kernel across width, and 2) a  $k \times 1$  kernel across height. This reduces  $k^2 \times h$  parameters to  $2k \times h$  in each SplitMixer block. Similarly the  $W \times H \times k^2 \times h$  FLOPS is reduced to  $W \times H \times 2k \times h$ , where  $W$  and  $H$  are width and height of the input tensor  $\mathbf{X} \in \mathbb{R}^{W \times H \times h}$ , respectively. Therefore, separating the 2D kernel into two 1D kernels results in  $\frac{k}{2}$  times savings in parameters and FLOPS. The two 1D convolutions are applied sequentially and each one is followed by a GELU activation and BatchNorm (denoted as “Act + Norm”

<sup>7</sup>We use GELU and BatchNorm throughout this section, except in ablation experiments.

<sup>8</sup>Throughout the section, a tensor or a kernel is represented as `width × height × channels`.



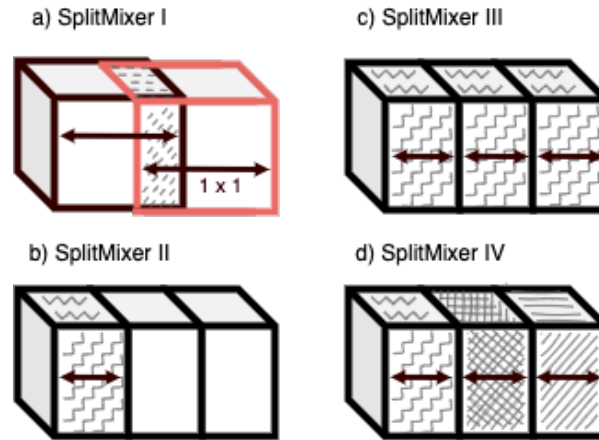
**Figure 6.18:** Basic architecture of SplitMixer. The input image is evenly divided into several image patches which are tokenized with linear projections. A number of 1D depthwise convolutions (spatial mixing) and pointwise convolutions (channel mixing) are repeatedly applied to the projections. For channel mixing, we split the channels into segments (hence the name SplitMixer) and perform convolution on them. We implement this part with our ad-hoc solutions or 3D convolution. Finally, a global average pooling layer followed by a fully-connected layer is used for class prediction.

in fig. 6.18).

## II. Channel mixing

We notice that most of the parameters in ConvMixer reside in the channel mixing layer. For  $h$  channels and kernel size  $k$  ( $h \gg k$ ), in each block there are  $h \times k^2$  parameters in the spatial mixing part and  $h^2$  parameters in the channel mixing part. Thus, the fraction of parameters in the two parts is  $\frac{h \times k^2}{h^2} = \frac{k^2}{h}$  which is much smaller than 1 (e.g.,  $5^2/256$ ). Therefore, most of the parameters are used for channel mixing.

**Implementation using 3D convolution** The basic idea here is to utilize 3D convolutions with certain strides. The output will be a set of interleaved maps coming from different segments. The same 3D kernel (with shared parameters) is applied to all segments. A certain number of 3D kernels will be needed to obtain an output tensor with the same number of channels as the input. Applying  $m$  3D kernels of size  $1 \times 1 \times \frac{h}{m}$  and stride  $\frac{h}{m}$  (assume  $h$  is divisible by  $m$ ), will require  $\frac{h^2}{m}$  parameters. Hence, more parame-



**Figure 6.19:** Channel mixing approaches: (a) channels are split into two overlapping segments, and only one segment is convolved in each block (no parameter sharing across segments), (b) channels are equally split into a number of segments, and only one segment is convolved in each block (no overlap or parameter sharing), (c) all segments are convolved in each block and parameters are shared across segments, and (d) all segments are convolved in each block (no parameter sharing).

ters and computation will be saved by increasing the number of segments (i.e., smaller 3D kernels). While being easy to implement, using 3D convolution has some restrictions. For example, kernel parameters have to be shared across segments, and all segments have to be convolved. Further, we find that channel mixing using 3D convolution is much slower than our other approaches (mentioned next).

**Other channel mixing approaches** A number of approaches are proposed that differ depending on whether they allow overlap or parameter sharing among segments. They offer different degrees of trade-off in accuracy, number of parameters, and FLOPS. Notice that both of our spatial and channel mixing modifications can be used in tandem or separately. In other words, they are independent of each other. The channel mixing approaches are shown in fig. 6.19 and are explained below.

**SplitMixer-I: Overlapping segs, no param sharing, update one seg per block**

The input tensor is split into two overlapping segments along the channel dimension. The intuition here is that the overlapped channels allow efficient propagation of information from the first segment to the second. Let  $m$  be the size of each segment and a fraction of  $h$ , i.e.,  $m = \alpha \times h$ ,  $\alpha > 0.5$ . The two segments can be represented as  $X[:m]$  and  $X[h-m:]$  in PyTorch. For instance, for  $\alpha = 2/3$ , one-third of the middle channels are shared between the two segments. We choose to apply convolution to only one segment in each block, e.g., the left segment in odd blocks and the right segment in even blocks.  $m$  number of  $1 \times 1$  convolutions are applied to the segment that should be updated. Therefore, the output has the same number of channels as the original segment, which is then concatenated to the other (unaltered) segment. The final output is a tensor with  $h$  channels to be processed in the next block. In the experiments, we choose  $\alpha = \frac{i}{2i-1}$ ,  $i \in \{2 \dots 6\}$ . The reduction in parameters per block can be approximated as<sup>9</sup>:

$$h^2 - (\alpha \times h)^2 = (1 - \alpha^2) \times h^2 \quad (6.11)$$

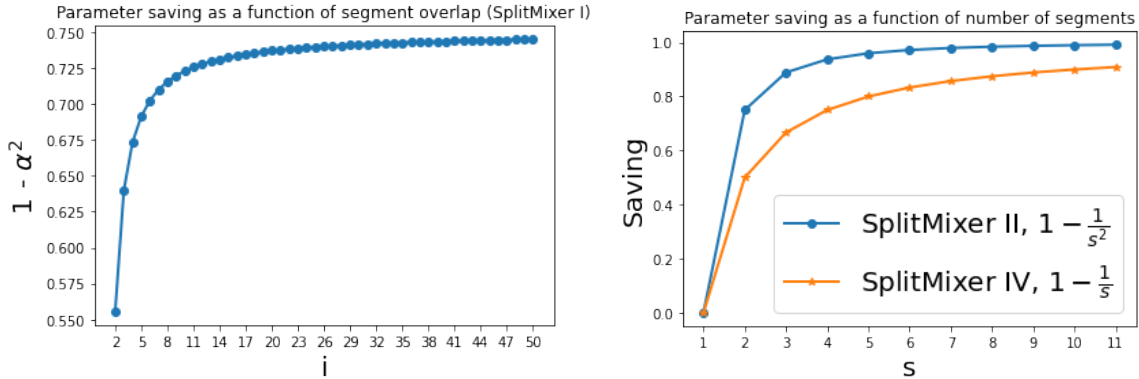
which means  $1 - \alpha^2$  fraction of parameters are reduced (e.g., 56% parameter reduction for  $\alpha = 2/3$ ; the parameter saving curve as a function of segmentat overlap is plotted in fig. 6.20). Notice that the bigger the  $\alpha$ , the less saving in parameters. Similarly, the reduction in FLOPS can be approximated as:

$$W \times H \times h \times h - W \times H \times (\alpha \times h) \times (\alpha \times h) = (1 - \alpha^2) \times W \times H \times h^2 \quad (6.12)$$

These equations show that the saving in FLOPS is the same as the saving in parameters. We also tried a variation of this design, denoted as SplitMixer-V, which is updating both

---

<sup>9</sup>For simplicity, here we discard bias, BatchNorm, and optimizer parameters.



**Figure 6.20:** Parameter saving as a function of (left) segment overlap for SplitMixer-I and (right) segment for SplitMixer-II and SplitMixer-IV. About 75% of parameters can be saved in the limit for SplitMixer-I (i.e., as  $i$  approaches infinity, see eq. (6.11)).

segments in the same block. This new variation has fewer parameters and FLOPS than ConvMixer. It saves less parameters compared to SplitMixers (ratio equal to  $1 - 2\alpha^2$ ) but achieves slightly higher accuracy (i.e., trade-off in favor of accuracy).

**SplitMixer-II: Non-overlapping segs, no param sharing, update one seg per block** We first split the  $h$  channels into  $s$  non-overlapping segments, each with size  $\frac{h}{s}$ , along the channel dimension<sup>10</sup>. In each block, only one segment is convolved and updated. Parameters are not shared across the segments. Following the above calculation, saving in parameters and FLOPS is  $1 - \frac{1}{s^2}$ . For example, for  $s = 2$ ,  $\sim 75\%$  of the parameters are reduced. The same argument holds for FLOPS.

**SplitMixer-III: Non-overlapping segs, param sharing, update all segs per block** Here,  $h$  channels are split into  $s$  non-overlapping segments with shared parameters. Notice that under this setting,  $h$  must be divisible by  $s$  in order to get all the channels convolved. All segments are convolved and updated simultaneously in each block. Due to parameter sharing, the reduction in parameters is the same as SplitMixer-

<sup>10</sup>Notice that when  $h$  is not divisible by the number of segments, the last segment will be longer (e.g., dividing  $h = 256$  into 3 segments means the segments would have dimension [85, 85, 86], in order).

II, i.e., the two SplitMixers have the same number of parameters for the same number of segments. The number of FLOPS, however, is higher now since computation is done over all  $s$  segments. The number of FLOPS is the same as SplitMixer-IV, which will be calculated in the following subsection.

**SplitMixer-IV: Non-overlapping segs, no param sharing, update all segs per block** This approach is similar to SplitMixer-III with the difference that here parameters are not shared across the segments. All segments are convolved and updated, and the results are concatenated. The reduction in parameters per block is:

$$h^2 - s \times (h/s)^2 = \left(1 - \frac{1}{s}\right) \times h^2, \quad (6.13)$$

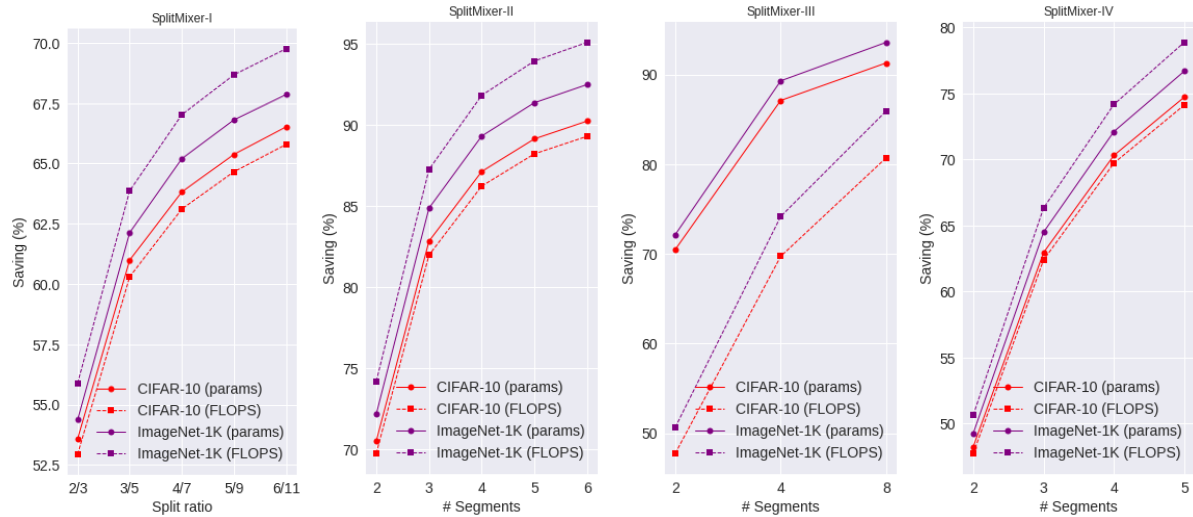
which results in  $1 - \frac{1}{s}$  parameter saving. For example, 66.6% of the parameters are reduced for  $s = 3$ . More savings can be achieved with more segments. The reduction in FLOPS is:

$$W \times H \times h \times h - s \times \left(W \times H \times \frac{h}{s} \times \frac{h}{s}\right) = \left(1 - \frac{1}{s}\right) \times W \times H \times h^2, \quad (6.14)$$

which means the same saving in FLOPS as in parameters.

**Comparison of channel mixing approaches** Among the mixing approaches, the SplitMixer-II saves the most parameters and computation but achieves lower accuracy. SplitMixer-I strikes a good balance between accuracy and model size (and FLOPS) thanks to its partial channel sharing. We assumed the same number of blocks in all mixing approaches. In practice, a smaller number of blocks might be required when all segments are updated simultaneously in each block. Notice that apart from these approaches, there may be some other ways to perform channel mixing. For example, in SplitMixer-I,





**Figure 6.21:** Potential savings in parameters and FLOPS for different SplitMixer variants.

parameters can be shared across the overlapped segments, or multiple segments can overlap. We leave these explorations to future research. We have also empirically measured the amount of potential saving in parameters and FLOPS over CIFAR-10 and ImageNet datasets, for model specifications mentioned in the next section. Results are shown in fig. 6.21.

### III. Naming convention

We name SplitMixers after their hidden dimension  $h$  and the number of blocks  $b$  like SplitMixer-A- $h/b$ , where A is a specific model type (I, II, ...).

## 6.2.3 Experiments and Results

We conducted several experiments to evaluate the performance of SplitMixer in terms of accuracy, the number of parameters, and FLOPS. Our goal was not to obtain the best possible accuracy. Rather, we were interested in knowing whether and how much parameters and computation can be reduced relative to ConvMixer. To this end, we

used their code and parameter settings. A thorough comparison of ConvMixer with other models is made in [2]. We implemented our model in PyTorch and used a Tesla V100 GPU with 32GB RAM to run it.<sup>11</sup>

We used RandAugment [192], random horizontal flip, and gradient clipping. Due to limited computational resources, *we did no perform extensive hyperparameter tuning*, so better results than those reported here may be possible. All models were trained for 100 epochs with batch size 512 over CIFAR- $\{10,100\}$  and 64 over Flowers102 and Food101 datasets. Unless stated otherwise,  $h$  and  $b$  were set to 256 and 8 across all datasets. We used AdamW [193] as the optimizer, with weight decay set to 0.005 (0.1 for Flowers102). The learning rate (lr) was adjusted with the OneCycleLR scheduler (max-lr was set to 0.05 for CIFAR- $\{10,100\}$ , 0.03 for Flowers102, and 0.01 for Food101). We utilized the ithop library<sup>12</sup> for measuring the number of parameters and FLOPS.

## I. Results on CIFAR- $\{10,100\}$ datasets

Both datasets contain 50,000 training images and 10,000 test images (resolution is  $32 \times 32$ ); each class has the same number of samples. We set  $p = 2$  and  $k = 5$  over both datasets.

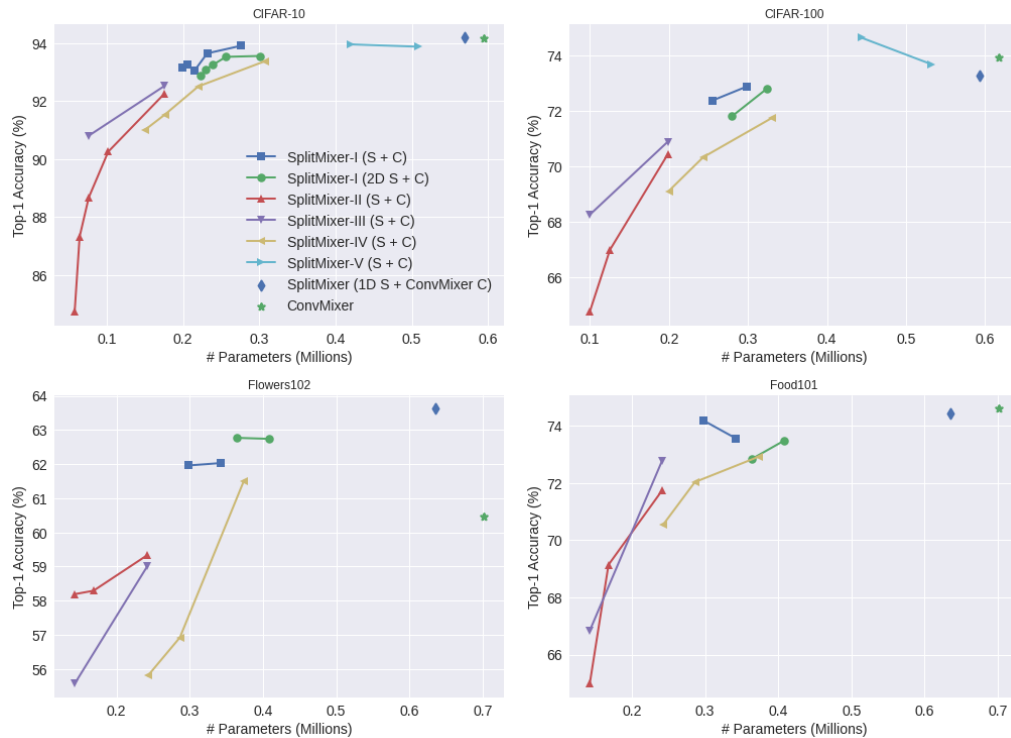
As shown in the top panel of fig. 6.22, ConvMixer scores slightly above 94% on CIFAR-10<sup>13</sup>. SplitMixer-I has about the same accuracy as ConvMixer but with less than 0.3M parameters which are almost half of the ConvMixer parameters. The same statement holds for FLOPS as shown in fig. 6.23. SplitMixer with 1D spatial mixing and regular  $1 \times 1$  channel mixing as in ConvMixer (denoted as “SplitMixer 1D S + ConvMixer C” in the Figure) attains about the same accuracy as ConvMixer with slightly lower param-

---

<sup>11</sup>Code is available at <https://github.com/aliborji/splitmixer>.

<sup>12</sup><https://github.com/Lyken17/pytorch-OpCounter>

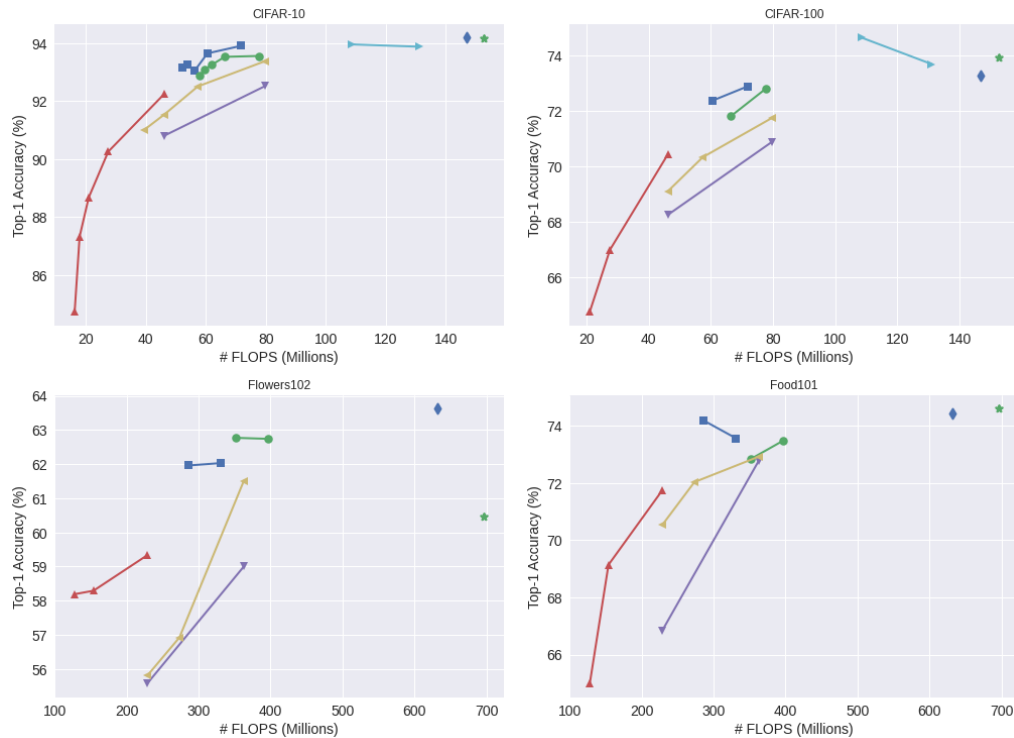
<sup>13</sup>The original ConvMixer paper has reported 96% accuracy on CIFAR-10 with Mixup and Cutmix data augmentation with 0.7M parameters. We expect even better results for SplitMixer with stronger data augmentation.



**Figure 6.22:** Accuracy vs. parameters for different variants of SplitMixer. S stands for 1D spatial convolution and C stands for  $1 \times 1$  pointwise convolution over channel segments. We plug in our components into ConvMixer, denoted here as “2D + C” (2D convolution kernels plus our channel mixing approach) and “1D S + ConvMixer C” (our 1D kernels plus channel mixing as is done in ConvMixer, i.e.,  $1 \times 1$  convolution across all channels without splitting). Data points are for different values of split ratio or number of segments depending on the model type. We have collected more data points on CIFAR-10 than other datasets. See also fig. 6.23 for accuracy vs. FLOPS plots.

eters and FLOPS. SplitMixer-I with 2D spatial kernels and segmented channel mixing (denoted as “SplitMixer 2D S + C”) performs on par with SplitMixer-I. Performance of the SplitMixer-II quickly drops with more segments (and subsequently fewer parameters). SplitMixers III and IV also perform well (above 90%). Interestingly, with only about 76K parameters, SplitMixer-III reaches about 91% accuracy. SplitMixer-V performs close to ConvMixer, but it does not save many parameters or FLOPS.

Qualitatively similar results are obtained over the CIFAR-100 dataset. Here, ConvMixer scores 73.9% accuracy, above the 72.5% by SplitMixer-I, but with twice more



**Figure 6.23:** Accuracy *vs.* FLOPS for different variants of SplitMixer. S stands for 1D spatial convolution and C stands for  $1 \times 1$  pointwise convolution over channel segments. We plug in our components into ConvMixer, denoted here as “2D + C” (2D convolution kernels plus our channel mixing approach) and “1D S + ConvMixer C” (our 1D kernels plus channel mixing as is done in ConvMixer, i.e.,  $1 \times 1$  convolution across all channels without splitting). Data points are for different values of split ratio or number of segments depending on the model type. We have collected more data points on CIFAR-10 than other datasets. Notice that the ratio of FLOPS over the number of parameters is almost the same for all models except SplitMixer-III, where this ratio is higher since all segments are updated in each block and parameters are shared across segments (see fig. 6.19). That is why the plots for parameters and FLOPS are almost the same for each model, except SplitMixer-III.

parameters and FLOPS.

On both datasets, increasing the number of segments saves more parameters and FLOPS but at the expense of accuracy. Interestingly, SplitMixer-I with only channel mixing does very well. Our channel mixing approach is much more effective than 1D spatial mixing in terms of lowering the number of parameters and FLOPS.

**Results using 3D convolution** We experimented with a model that uses 128 kernels of size  $1 \times 1 \times 128$  and stride 128 along the channel dimension (i.e., channels are partitioned into two non-overlapping segments each of size 128) for channel mixing. This model scores 93.09% and 71.99% on CIFAR-10 and CIFAR-100, respectively. While having similar accuracy, the number of parameters (about 0.2 M), and FLOPS (about 0.08 G) as SplitMixer models, this model is much slower to train (each epoch takes twice more time). Notice that, among the channel mixing approaches, only SplitMixer-III can be considered as 3D convolution. Thus, performing channel mixing through strided 3D convolution is a subset of our proposed solutions.

## II. Results on Flowers102 and Food101 datasets

Flowers102 contains 1020 training images (10 per class) and 6149 test images. Food101 contains 750 training images and 250 test images for each of its 101 classes. We used larger patch ( $p = 7$ ) and kernel sizes ( $k = 7$ ) since image size is bigger in these datasets (both resized to  $224 \times 224$ ).

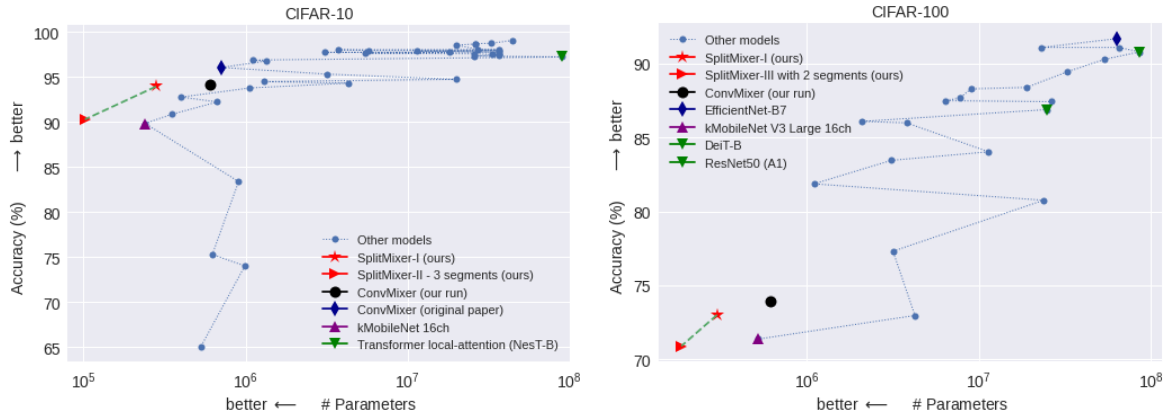
Results are shown in fig. 6.22. The patterns are consistent with what we observed over CIFAR datasets. SplitMixer variants, with small number of segments, perform close to the ConvMixer. Over the Flowers102 dataset, SplitMixer-I scores 62.03%, higher than the 60.47% by ConvMixer. Similarly, over Food101, SplitMixer-I scores 1% lower than ConvMixer, but with less than half of ConvMixer’s parameters and FLOPS. In general, increasing the overlap between segments (by raising  $\alpha$  in SplitMixer-I) or reducing the number of segments enhances the accuracy, but also increases the number of parameters across datasets (not conclusive on Food101 dataset). Further, SplitMixer is effective over both small and large datasets.

**Table 6.3:** Comparison with other models. The best numbers in each column are highlighted in bold. The number of parameters and FLOPS are averaged over CIFAR-10 and CIFAR-100 for our models. Notice that some variants of SplitMixer perform better than the numbers reported here over Flowers102 and Food101 datasets. Results, except ConvMixer and our model, are reproduced from [3] where they have trained models for 200 epochs. We have trained ConvMixer and SplitMixer for 100 epochs.

Model Family	Model	Params/FLOPS (M) / (G)	CIFAR 10	CIFAR 100	Params/FLOPS (M) / (G)	Flowers 102	Food 101
CNN	ResNet20 [148]	<b>0.27</b> / <b>0.04</b>	91.99	67.39	<b>0.28</b> / 2.03	57.94	74.91
Transformer	ViT [189]	2.69 / 0.19	86.57	60.43	2.85 / 0.94	50.69	66.41
MLP	AS-MLP [194]	26.20 / 0.33	87.30	65.16	26.30 / 1.33	48.92	74.92
"	gMLP [195]	4.61 / 0.34	86.79	61.60	6.54 / 1.93	47.35	73.56
"	ResMLP [196]	14.30 / 0.93	86.52	61.40	14.99 / 1.23	45.00	68.40
"	ViP [197]	29.30 / 1.17	88.97	70.51	30.22 / 1.76	42.16	69.91
"	MLP-Mixer [188]	17.10 / 1.21	85.45	55.06	18.20 / 4.92	49.41	61.86
"	S-FC ( $\beta$ -LASSO) [198]	- / -	85.19	59.56	- / -	-	-
"	MDMLP [3]	0.30 / 0.28	90.90	64.22	0.41 / 1.59	60.39	<b>77.85</b>
"	ConvMixer	0.60 / 0.15	<b>94.17</b>	<b>73.92</b>	0.70 / 0.70	60.47	74.59
"	SplitMixer-I (ours)	0.28 / 0.07	93.91	72.44	0.34 / <b>0.33</b>	<b>62.03</b>	73.56

**Comparison with state of the art** Table 6.3 shows a comparison of SplitMixer with models from MLP, Transformer, and CNN families. Some results are borrowed from [3] where they trained the models for 200 epochs, whereas here we trained our models for 100 epochs. While the experimental conditions in [3] might not be exactly the same as ours, cross-examination still provides insights into how our models fare compared to others, in particular the MLP-based models. Our models outperform other models while having significantly smaller sizes and computational needs. For example, SplitMixer-I has about the same number of parameters as ResNet20, but is about 2% better on CIFAR-10 and 5% better on CIFAR-100. Over Flowers102, SplitMixer drastically outperforms other models in all three aspects, including accuracy, number of parameters, and FLOPS. Both SplitMixer and ConvMixer are on par with other models on the Food101 dataset, with ConvMixer performing slightly better.

To illustrate the efficiency of the proposed modifications, in fig. 6.24 we plot accuracy



**Figure 6.24:** Comparison of our proposed SplitMixer architectures with state-of-the-art models that do not use external data for training. Results are shown over CIFAR- $\{10,100\}$  datasets. Notice that **we have not optimized our models for the best performance**. Rather, we ran the ConvMixer and our models using the exact same code, parameters, and machines to measure how much we can save parameters and computation relative to ConvMixer. Please consult [2] for a more detailed comparison of ConvMixer with other models. We have borrowed some data from <https://paperswithcode.com/> to generate these plots.

*vs.* number of parameters for our models and state-of-the-art models that do not use external data for training. Over CIFAR- $\{10,100\}$  datasets, in the low-parameter regime, our models push the envelope towards the top-left corner, which means a better trade-off between accuracy and model size (also speed). Our models even outperform some very well-known architectures such as MobileNet [199].

### III. Results on ImageNet-1K dataset

Results are shown in table 6.4. We use SplitMixer-I with a 2/3 overlap ratio for the experiments. Two parameter settings are considered: a) hidden dimension equal to 1536, 20 blocks, kernel size 9, patch size 7, and GELU activation, and b) hidden dimension equal to 768, 32 blocks, kernel size 7, patch size 7, and ReLU activation. The two settings are trained for 150 and 300 epochs, respectively. The training settings (including image augmentations, optimizers, schedulers, etc.) and hyperparameters are

**Table 6.4:** Results over ImageNet-1k. Models trained and evaluated on  $224 \times 224$  images.

Network	Patch Size	Kernel Size	# Params ( $\times 10^6$ )	# FLOPS ( $\times 10^9$ )	Act. Fn.	# Epochs	ImNet top-1 (%)
ConvMixer-1536/20	7	9	51.6	51.3	G	150	81.37
ConvMixer-768/32	7	7	21.1	0.33	R	300	80.16
SplitMixer-I-1536/20	7	9	23.5	22.6	G	150	79.35
SplitMixer-I-768/32	7	7	9.8	0.15	R	300 (350)	(75.05) 75.38
ResNet-152	–	3	60.2	-	R	150	79.64
DeiT-B	16	–	86	-	G	300	81.8
ResMLP-B24/8	8	–	129	-	G	400	81.0

the same as ConvMixers without any tuning. In agreement with the above-mentioned results, here SplitMixer performs close to ConvMixer (79.35% vs. 81.37%) in setting one. It, however, requires less than half of ConvMixer parameters. A similar observation is made in the second setting. Our model performs  $\sim 4\%$  lower than ConvMixer but has much fewer parameters. Notice that here we only tested two settings, and the results seem promising compared to ConvMixer and other models.

We observe a slower convergence of our models. We suspect this might be due to using OneCycle scheduler. To test this, we continued the training in the second setting for an additional 50 epochs. The performance improved from 75.05% to 75.38%. Thus, more training epochs, or a higher max learning rate during the initial training setup, might help in case of future usage of our models.

#### IV. Ablation experiments

We conducted a series of ablation experiments to study the role of different design choices and model components. Results are shown in table 6.5 over CIFAR- $\{10,100\}$  datasets. We took SplitMixer-I as the baseline and discarded or added pieces to it. Our findings are summarized as follows: (1) Completely removing the residual connections does not hurt the performance much. These connections, however, might be important



**Table 6.5:** Ablation study of SplitMixer-I-256/8 with a split ratio of 2/3 (Top-1 Acc).

Ablation of SplitMixer-I-256/8 on CIFAR- $\{10, 100\}$		
Ablation	CIFAR-10	CIFAR-100
SplitMixer-I (baseline)	93.91	72.88
– Residual in eq. (6.9)	92.24	71.34
+ Residual in eq. (6.10)	92.35	70.44
BatchNorm $\rightarrow$ LayerNorm	88.28	66.60
GELU $\rightarrow$ ReLU	93.39	72.56
– RandAug	90.87	66.54
– Gradient Norm Clipping	93.38	71.95
SplitMixer-I (Spatial only)	76.24	53.25
SplitMixer-I (Channel only)	64.21	40.46
One segment with size $\alpha \times h$ ; $\alpha = \frac{2}{3}$	76.28	51.28

for very deep SplitMixers; (2) Moving the residual connection to after channel mixing seems to hurt the performance. We find that the best place for the residual connections is right after spatial mixing; (3) Switching to LayerNorm from BatchNorm leads to a drastic performance drop; (4) The choice of activation function, GELU *vs.* ReLU, is not very important. In fact, we found that using ReLU sometimes helps; (5) Gradient norm clipping hinders the performance slightly, thus it is not very important; (6) Data augmentation, here as RandAug, is critical to gaining high performance. Notice that, unlike ConvMixer, we do not have Mixup and CutMix; (7) SplitMixer-I with only 1D spatial mixing, and no channel mixing, performs very poorly. The same is true for ablating the spatial mixing i.e., having only channel mixing. We find that spatial mixing is more important than channel mixing in our models; (8) Keeping only one of the segments in channel mixing, hence 1D spatial mixing plus channel mixing using  $m$  channels ( $m < h$ ), lowers the accuracy by a large margin. This indicates that there is a substantial benefit in having a larger  $h$  and splitting it into segments (and having overlaps between

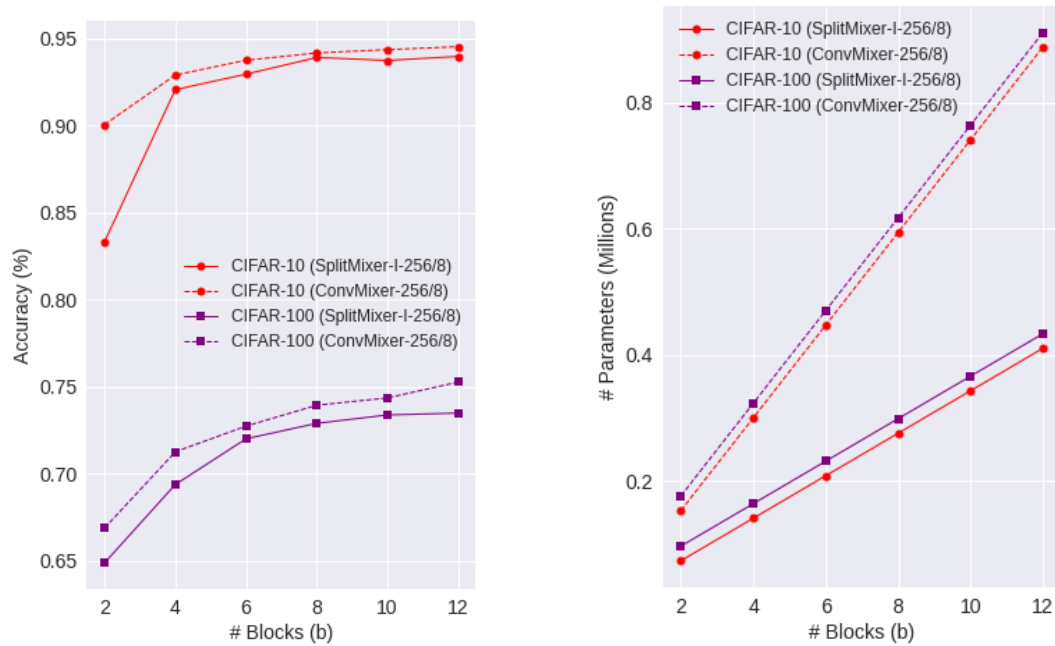
them). Notice that in each block, only one segment is updated. In other words, simply lowering the number of channels does not lead to the gains that we achieve with our models. Any ConvMixer, small or large, can be optimized using our techniques.

## V. The role of the number of blocks

We wondered about the utility of the proposed modifications over deeper networks. To this end, we varied the number of blocks  $b$  of ConvMixer and SplitMixer-I in the range 2 to 10 in steps of 2, and trained the models. Other parameters were kept the same as above. As the results in fig. 6.25 show, increasing the number of blocks improves the accuracy of both models on CIFAR $\{10,100\}$ . SplitMixer-I performs slightly below the ConvMixer, but it has a huge advantage in terms of the number of parameters and FLOPS, in particular over deeper networks. The model size and computation grow slower for SplitMixer compared to ConvMixer.

## VI. Model throughput

We measured throughput using batches of 64 images on a single Tesla v100 GPU with 32GB RAM [200], averaging over 100 such batches. Similar to ConvMixer, we considered CUDA execution time rather than “wall-clock” time. Here, we used the network built for the FLOWER102 classification ( $h = 256$ ,  $d = 8$ ,  $p = 7$ ,  $k = 7$ , and image size  $224 \times 224$ ). We measured throughput when our model was the only process running on the GPU. Results are shown in table 6.6. The throughput of our model is almost three times higher than ConvMixer. As expected, the throughput is higher with more channel segments since the number of FLOPS is lower.



**Figure 6.25:** The role of the number of blocks  $b$  on model performance. The FLOPS of models over CIFAR-100 are just slightly higher than CIFAR-10, thus not visible in the rightmost panel.

## 6.2.4 Related Work

For about a decade, CNNs have been the de-facto standard in computer vision [148]. Recently, the Vision Transformers (ViT) by [189] and its variants [201, 202, 203, 204, 205, 206], and the multi-layer perceptron mixer (MLP-Mixer) by [188] and its variants [196, 207] have challenged CNNs. These models have shown impressive results, even better than CNNs, in large-scale image classification. Unlike CNNs that exploit local convolutions to encode spatial information, vision transformers take advantage of the self-attention mechanism to capture global information. MLP-based models, on the other hand, capture global information through a series of spatial and channel mixing operations.

MLP-Mixer borrows some design choices from recent transformer-based architectures [208]. Following ViT, it converts an image to a set of patches and linearly embeds

**Table 6.6:** Model throughput for SplitMixer-A-256/8 on a Tesla v100 GPU with 32GB RAM over a batch of 64 images of size  $224 \times 224$ , averaged over 100 such batches.

Network	Throughput (img/sec)						
ConvMixer	815.84						
SplitMixer-I	Overlap ratio						
	2/3	3/5	4/7	5/9	6/11	-	-
	2097.55	2208.40	2210.06	2220.09	2231.42	-	-
SplitMixer-II	Number of segments						
	2	3	4	5	6	7	8
	2322.02	2291.44	2440.16	2464.33	2474.318	-	-
SplitMixer-III	2112.290	-	2171.70	-	-	-	2185.61
SplitMixer-IV	2110.92	2084.55	2170.57	2146.76	-	-	-

them to a set of tokens. These tokens are processed by a number of “isotropic” blocks, which are in essence similar to the repeated transformer-encoder blocks [208]. For example, MLP-Mixer replaces self-attention with MLPs applied across different dimensions (i.e., spatial and channel location mixing). ResMLP [196] is a data-efficient variation on this scheme. CycleMLP [209], gMLP [195], and vision permutator [197], conduct different approaches to perform spatial and channel mixing. For example, the vision permutator permutes a tensor along the height, width, and channel to apply MLPs. Some works attempt to bridge convolutional networks and vision transformers and use one to improve the other [210, 202, 211, 212, 206, 213, 214, 215].

We are primarily inspired by the ConvMixer ([216]). This model introduces a simpler version of MLP-Mixer but is essentially the same. It replaces the MLPs in MLP-Mixer with convolutions. In general, Convolution-based MLP models are smaller than their heavy Transformer-, CNN-, and MLP-based counterparts. Here, we show that it is possible to trim these models even more. Perhaps the biggest advantage of the MLP-based models is that they are easy to understand and implement, which in turn helps replicate results and compare models. Please see section 6.2.5.

## 6.2.5 Discussions and Conclusion

### I. A unified view of vision Transformer and MLP-Mixer

MLP-Mixer borrows some design ideas from Vision Transformers. The most obvious one is splitting the input image into patches and mapping each patch to an embedding vector using a linear layer. Both ViT and MLP-Mixer do not use convolutions, or at least claim not to. However, one can argue that the linear embedding is, in fact, convolution with a stride equal to the patch size and parameter sharing across patches. Here, we cross-examine both architectures and show that their similarities go beyond the embedding layer: (1) The embedding layer in the two models is the same and is implemented using an MLP with a single layer; (2) Channel mixing is done in the exact same way in both models via a two-layer MLP; (3) Both models use skip connections the same way in both channel and token mixing parts; (4) Both models use LayerNorm for normalization.

The major difference between the models is the way they implement token mixing. Token Mixing in the ViT happens in the Multi-head self-attention (MHSA) layer, whereas in the MLP-Mixer, it is done via a two-layer MLP. MHSA can have multiple heads. In extreme cases, it can have one head of size  $d$  (embedding dimension), or  $d$  heads of size 1. In either case, the information after self-attention is passed through an MLP. Effectively, the MSHA layer does both token mixing and channel mixing. After multiple layers of token and channel mixing, the models map information to class labels. In ViT, an extra token called [cls] token (with dimension  $d$ ) is mapped to the class labels using a two-layer MLP. In MLP-Mixer, this is done the same way using an MLP, but first, the information is pooled across different patches (the Average Pooling layer). Apart from the major difference in token mixing, there are two other differences that do not seem to be crucial: (1) The [CLS] token in ViT already contains the summary information from other patches. Pooling information across patches as it is done in MLP-Mixer

(the average pooling layer) does not seem to matter much. However, it needs to be studied; (2) MLP-Mixer does not utilize positional encoding<sup>14</sup>. ViT authors showed that including positional information indeed improves accuracy. Positional encoding helps maintain positional information, which will otherwise be lost after several layers of token and channel mixing throughout the network. Interestingly, without explicitly accounting for spatial information, the MLP-Mixer still performs very well and is on par with ViT. It would be interesting to see if adding spatial information to the MLP-Mixer can improve its accuracy.

In this unified view, one can link our proposed SplitMixer to axial attention [217]. In particular, the connection between spatial mixing and axial attention is evident, as both capture 2D relationships with two 1D components. On the other hand, channel mixing can also be viewed as a form of axial attention, if one considers the split channels as different channel “coordinates”.

## II. Future work

The proposed solution based on separable filters, depthwise convolution, and channel splitting is quite efficient in terms of parameters and computation. However, if a network is already small, reducing the parameters too much may cause the network not to learn properly during training. Thus, a balance is required to enhance efficiency without significantly reducing effectiveness. We propose the following directions for future research in this area: first, try a wider range of hyperparameters and design choices for SplitMixer, such as strong data augmentation (e.g., Mixup, Cutmix), deeper models, larger patch sizes, overlapped image patches, label smoothing [218], and stochastic depth [219]. Previous research has shown that some classic models can achieve state-of-the-art performance

---

<sup>14</sup>Unlike NLP where the order or the words can alter the meaning of a sentence, reordering the image patches does not seem to result in a viable scene and does not happen naturally. Thus, it might not be important in vision tasks!

through carefully-designed training regimes [220]. Second, we tried several ways to split and mix the channels and learned that some perform better than others—there might be even better approaches to do this. In addition, incorporating techniques similar to the ones proposed here to optimize other MLP-like models is also a promising direction. Lastly, MLP-like models, including SplitMixer, lack effective means of explanation and visualization, which need to be addressed in the future.

This section proposed SplitMixer, a simple yet efficient model, that is similar in spirit to ConvMixer, ViT, and MLP-Mixer models. SplitMixer uses 1D convolutions for spatial mixing and splits the channels into several segments, and performs  $1 \times 1$  convolution on them for channel mixing. Our experiments, even without extensive hyperparameter tuning, demonstrate that these modifications result in models that are very efficient in terms of the number of parameters and computation. In terms of accuracy, they outperform several MLP-based models and some other model types with similar size constraints. Our main point is that SplitMixer allows sacrificing a small amount of accuracy to achieve big gains in reducing parameters and FLOPS. Our results entertain the idea that it may be possible to find model classes that have fewer parameters than the number of data points. This may challenge the current belief that deep networks must be overparameterized to perform well.

# Chapter 7

## Conclusion and Contributions

This dissertation is devoted to bridging machine learning advances and neuroscience. We investigated how to use various machine learning tools to deepen our understanding of the human brain’s cognitive processes, and how to better deep learning models with neuroscience insights. We first demonstrated how to model the relationship between the brain’s structural and *static* functional networks. Then we show how we can learn more comprehensive representations with structural connectivities and *dynamic* functional activities. Next, we perform stimuli decoding from brain signals, studying the brain dynamics with *continuous* semantics instead of under discrete tasks. We then explored the redundancy and dependency in the brain, providing interesting insights about visual processing. We conclude our work with two efforts in bringing neuroscience into deep learning models: one helped us better understand model biases, and another brought more sparsity into the model, thus saving a considerable amount of parameters and computation.

This chapter summarizes our contributions to the interdisciplinary community that studies brain science and computer science.

In chapter 2, we proposed a convex optimization framework to perform coupled network reconstructions under domain constraints. In particular, we aimed to understand



the relationship between functional neural processes and anatomical connectivities, both of which are represented by their edge networks. We formulated the relationship between this set of networks as a regularized multiple regression problem with a novel objective function. The proposed framework does not depend on Gaussian assumptions and is able to incorporate prior domain knowledge through a hard-constraint put on the noise term. This constraint term also provides a more scalable solution when network connectivity is sparse. We then developed a fast method based on nested FISTA for solving the proposed optimization problem. We validated our method on multishell diffusion and task-evoked fMRI datasets from the Human Connectome Project, leading to important insights on structural backbones that support various types of task activities and general solutions to the study of coupled networks.

In chapter 3, we presented an efficient graph neural network model that jointly models both structural and dynamic functional brain signals, providing a more comprehensive representation of brain activities than the current fMRI literature. Unlike typical spatial-temporal graph neural networks that learn a universal latent structure, we propose sample-level latent adaptive adjacency matrix learning based on input snippets: this better captures the evolving dynamics of a task. We also proposed multi-resolution inner-cluster smoothing, which effectively encodes long-range node relationships while keeping the graph structure, enabling the model to leverage structural and latent adjacency matrices throughout the process. Together with subject structural connectivity and sample-level adjacency matrix learning, the inner cluster smoothing learns and refines latent dynamic structures on limited signal data. We carry out extensive ablation studies and model comparisons to show the superiority of the proposed model in representing brain dynamics. We also leverage graph attribution methods to investigate and interpret the importance of both spatial brain regions and temporal keyframes, as well as heterogeneity among brain regions, tasks, and subjects. These results can open up

new opportunities for identifying biomarkers for different tasks or diseases and markers for other complex scientific phenomena.

In chapter 4, we argued that the brain encodes visual stimuli in a rich semantic space; thus, incorporating additional text modality when studying visual decoding and encoding is beneficial. Based on this argument, we proposed a brain decoding pipeline that successfully reconstructs *complex images* from human brain signals. It allows one to study the brain’s visual decoding in a more natural setting than reconstructing object-centered images. Compared to previous works, it also decodes signals from more voxels and regions, including those outside the visual cortex, that are responsive to the experiment. This inclusion allows us to study the behavior and functionality of more brain areas. Furthermore, we addressed the data scarcity issue by leveraging pre-trained models and a latent space shared by images and texts, with customized losses and training schemes. Our results show we can decode complex images from fMRI signals relatively faithfully, particularly from a semantic perspective. We also perform microstimulation on different brain regions to study their properties and showcase the potential usages of the pipeline. Finally, we demonstrated that the encoding process, hence the complete encoding-decoding cycle, can be achieved by incorporating the text modality, similar to the decoding process.

In chapter 5, we systematically studied the redundancy and dependency of brain signals with an autoencoder and a multi-label classifier. We did so with two experiments: reconstructing brain activities and classifying visual stimuli categories that trigger the brain signals—both with input activities at only a portion of overall voxels. With the autoencoder, we demonstrated that the latent representation of voxel signals aligns with semantic information of the causative stimuli, and the final reconstruction of the voxel activities has a much lower dimensionality. These results suggest new ways for signal compression, decomposition, denoising, and upsampling through autoencoders. In ad-

dition, we found that the brain encodes different scene semantics with varying levels of redundancy, which generally varies across individuals but also with shared patterns. We also studied discrepancies between hemispheres, and dependencies between regions and voxels, providing new insights into human visual encoding processes.

Chapter 6 took the opposite direction from the previous ones: instead of using machine learning tools to gain a deeper understanding of the brain, it presented two of our efforts to utilize neuroscience tools and insights to study deep learning models. The first work presented in this chapter utilizes two related psychophysics/neurophysiology methods: classification images and spike-triggered averaging. Both methods take white noise as model inputs in the context of deep learning models. Over multiple datasets and model architectures, we employed classification images to unveil implicit biases of a network, utilized those biases to influence network decisions and detect adversarial perturbations. We also showed how spike-triggered averaging could be used to identify and visualize filters in different model layers. Across different model types, we found that CNNs can be characterized the best by a psychometric function and behaves most similarly to the biological visual system. The second work is inspired by the sparse activation of neurons and the modular organization of the brain: we aimed to bring more sparsity into MLP-like networks. To this end, we modified both spatial mixing and channel mixing. For spatial mixing, we apply 1D convolutions across width and height instead of 2D convolutions; for channel mixing, we split the channels into overlapping or non-overlapping segments and apply convolution to channel segments instead of all channels. We provided theoretical analyses and empirical support for the computational efficiency of the proposed solution, showing that the proposed method can achieve significant gains in reducing parameters and computation with only minor accuracy sacrifice. Both of these efforts showed promising results in combining neuroscience knowledge into deep learning research.

# Appendix A

## A.1 Tasks Descriptions of the CRASH dataset

The following are task descriptions of the CRASH (Cognitive Resilience and Sleep History) dataset [64]:

**Resting state:** The subject simply lays in the scanner awake, with eyes open for 5 minutes.

**Visual working memory task (VWM):** The subject is presented with a pattern of colored squares on a computer screen for a very brief period (100ms). After 1000ms, they are presented with a single square and must determine if it is the same or different color as the previously presented square at that location. Responses are made with a button press ([221]).

**Dynamic Attention Task (DYN):** Two streams of orientation gratings are presented to the left and right of fixation. Subjects monitor a specified stream for a target (about a 2-degree shift in orientation, clockwise or counterclockwise) that indicates whether the subject should continue to monitor the current stream (hold) or monitor the other stream (shift) and respond with a button press ([222]).

**Dot Probe Task (Faces) (DOT):** On each trial, two faces are presented, one neutral and the other happy or angry for 500ms. Then, either of two simple symbols

is presented at the position of either of the faces. The subject must make forced-choice discrimination against the symbol. Reaction time differences as a function of the valance for the preceding facial expression are calculated. There is increased variability of the bias with PTSD and fatigue ([223]).

**Math task (MOD):** Subjects perform a modular math computation every 8 seconds and respond with a yes or no button press. The object of modular arithmetic is to judge the validity of problems such as  $51=19(\text{mod } 4)$ . One way to solve it is to subtract the middle number from the first number (i.e.,  $51-19$ ) and then divide this difference by the last number ( $32/4$ ). If the dividend is a whole number, the answer is “true.” Otherwise the answer is false ([224]).

**Psychomotor vigilance task (PVT):** The subject monitors the outline of a red circle on a computer screen for 10 minutes, and whenever a counterclockwise red sweep begins, they press a button as fast as possible. Subjects are provided with response time feedback. The experimenter records response latencies ([225]).

## A.2 Experiment setting details for Chapter 4

**Hyperparameters** The following hyperparameters are used in our experiments:

- $\tau = 0.5$  in eq. (4.1) for all the contrastive losses.
- for fMRI-CLIP mappers  $f_{mi}, f_{mc}$  (losses are in eq. (4.2)), the models are first trained with  $\alpha_1 = 0.4, \alpha_2 = 0.6, \alpha_3 = 0$ , then finetuned with  $\alpha_1 = 0.2, \alpha_2 = 0.3, \alpha_3 = 0.5$ .
- mappers are trained with batch size 32 (on a single GPU) when not including contrastive loss, and batch size 128 when including the contrastive loss or using VICReg loss. Learning rate is 0.0004.
- $\lambda_1 = 5, \lambda_2 = 10, \lambda_3 = 10$  for the losses of conditional StyleGAN2.

- conditional StyleGAN2 is trained with batch size  $16 \times$  number of GPUs (in our case  $B = 32$  since we used two GPUs). Learning rate is 0.0025.

**Image augmentation during training** Based on conclusions from StyleGAN2-ADA [108], we perform the following image augmentations before passing images into the CLIP encoder when training the fMRI-CLIP mapping model:

- perform random-sized crop with a scale between 0.8 to 1.
- perform horizontal flip with probability  $p = 0.5$ .
- perform ColorJitter(0.4, 0.4, 0.2, 0.1) with  $p = 0.4$ .
- perform grayscale with  $p = 0.2$ .
- perform Gaussian blur with  $p = 0.5$  and kernel size 23.
- perform random masking with 0.3 masking ratio.

We test mapping models trained with and without the above augmentations, and found augmentations can improve fMRI to CLIP image embedding mapping performance (details are in table 4.2).

# Bibliography

- [1] B. Thomas Yeo, F. M. Krienen, J. Sepulcre, M. R. Sabuncu, D. Lashkari, M. Hollinshead, J. L. Roffman, J. W. Smoller, L. Zöllei, J. R. Polimeni, *et. al.*, *The organization of the human cerebral cortex estimated by intrinsic functional connectivity*, *Journal of Neurophysiology* **106** (2011), no. 3 1125–1165.
- [2] A. Trockman and J. Z. Kolter, *Orthogonalizing convolutional layers with the cayley transform*, *arXiv preprint arXiv:2104.07167* (2021).
- [3] T. Lv, C. Bai, and C. Wang, *Mdmlp: Image classification from scratch on small datasets with mlp*, *arXiv preprint arXiv:2205.14477* (2022).
- [4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et. al.*, *Imagenet large scale visual recognition challenge*, *International journal of computer vision* **115** (2015), no. 3 211–252.
- [5] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, *GLUE: A multi-task benchmark and analysis platform for natural language understanding*, in *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018* (T. Linzen, G. Chrupala, and A. Alishahi, eds.), pp. 353–355, 2018.
- [6] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, *Superglue: A stickier benchmark for general-purpose language understanding systems*, *Advances in neural information processing systems* **32** (2019).
- [7] C. Presigny and F. D. V. Fallani, *Colloquium: Multiscale modeling of brain network organization*, *Reviews of Modern Physics* **94** (2022), no. 3 031002.
- [8] D. S. Bassett and O. Sporns, *Network neuroscience*, *Nature Neuroscience* **20** (2017), no. 3 353–364.
- [9] M. Rubinov and O. Sporns, *Complex network measures of brain connectivity: uses and interpretations*, *NeuroImage* **52** (2010), no. 3 1059–1069.

- [10] K. Batista-García-Ramó and C. I. Fernández-Verdecia, *What we know about the brain structure–function relationship*, *Behavioral Sciences* **8** (2018), no. 4 39.
- [11] D. S. Bassett, P. Zurn, and J. I. Gold, *On the nature and use of models in network neuroscience*, *Nature Reviews Neuroscience* **19** (2018), no. 9 566–578.
- [12] D. C. Van Essen, S. M. Smith, D. M. Barch, T. E. Behrens, E. Yacoub, K. Ugurbil, W.-M. H. Consortium, *et. al.*, *The wu-minn human connectome project: an overview*, *NeuroImage* **80** (2013) 62–79.
- [13] A. J. Rothman, E. Levina, and J. Zhu, *Sparse multivariate regression with covariance estimation*, *Journal of Computational and Graphical Statistics* **19** (2010), no. 4 947–962.
- [14] K.-A. Sohn and S. Kim, *Joint estimation of structured sparsity and output structure in multiple-output regression via inverse-covariance regularization*, in *Artificial Intelligence and Statistics*, pp. 1081–1089, 2012.
- [15] M. Wytock and Z. Kolter, *Sparse gaussian conditional random fields: Algorithms, theory, and application to energy forecasting*, in *International Conference on Machine Learning*, pp. 1265–1273, 2013.
- [16] X.-T. Yuan and T. Zhang, *Partial gaussian graphical model estimation*, *IEEE Transactions on Information Theory* **60** (2014), no. 3 1673–1687.
- [17] F. Freyer, K. Aquino, P. A. Robinson, P. Ritter, and M. Breakspear, *Bistability and non-gaussian fluctuations in spontaneous cortical activity*, *Journal of Neuroscience* **29** (2009), no. 26 8512–8524.
- [18] J. Hlinka, M. Paluš, M. Vejmelka, D. Mantini, and M. Corbetta, *Functional connectivity in resting-state fmri: is linear correlation sufficient?*, *NeuroImage* **54** (2011), no. 3.
- [19] A. Eklund, T. E. Nichols, and H. Knutsson, *Cluster failure: Why fmri inferences for spatial extent have inflated false-positive rates*, *Proceedings of the National Academy of Sciences* **113** (2016), no. 28 7900–7905.
- [20] M. Columb and M. Atkinson, *Statistical analysis: sample size and power estimations*, *Bja Education* **16** (2016), no. 5.
- [21] J. Peng, P. Wang, N. Zhou, and J. Zhu, *Partial correlation estimation by joint sparse regression models*, *Journal of the American Statistical Association* **104** (2009), no. 486 735–746.



- [22] K. Khare, S.-Y. Oh, and B. Rajaratnam, *A convex pseudolikelihood framework for high dimensional partial correlation estimation with convergence guarantees*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **77** (2015), no. 4 803–825.
- [23] P. Koanantakool, A. Ali, A. Azad, A. Buluc, D. Morozov, L. Oliker, K. Yelick, and S.-Y. Oh, *Communication-avoiding optimization methods for distributed massive-scale sparse inverse covariance estimation*, in *International Conference on Artificial Intelligence and Statistics*, pp. 1376–1386, PMLR, 2018.
- [24] S. Oh, O. Dalal, K. Khare, and B. Rajaratnam, *Optimization methods for sparse pseudo-likelihood graphical model selection*, in *Advances in Neural Information Processing Systems*, pp. 667–675, 2014.
- [25] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, *SIAM Journal on Imaging Sciences* **2** (2009), no. 1 183–202.
- [26] R. T. Rockafellar, *Convex analysis*, vol. 28. Princeton University Press, 1970.
- [27] J.-J. Moreau, *Proximité et dualité dans un espace hilbertien*, *Bulletin de la Société mathématique de France* **93** (1965) 273–299.
- [28] M. Yuan and Y. Lin, *Model selection and estimation in the gaussian graphical model*, *Biometrika* **94** (2007), no. 1 19–35.
- [29] J. Peng, J. Zhu, A. Bergamaschi, W. Han, D.-Y. Noh, J. R. Pollack, and P. Wang, *Regularized multivariate regression for identifying master predictors with application to integrative genomics study of breast cancer*, *The Annals of Applied Statistics* **4** (2010), no. 1 53.
- [30] C. McCarter and S. Kim, *Large-scale optimization algorithms for sparse conditional gaussian graphical models*, in *Artificial Intelligence and Statistics*, pp. 528–537, 2016.
- [31] T. Fawcett, *An introduction to roc analysis*, *Pattern Recognition Letters* **27** (2006), no. 8 861–874.
- [32] J. Friedman, T. Hastie, and R. Tibshirani, *Applications of the lasso and grouped lasso to the estimation of sparse graphical models*, tech. rep., Stanford University, 2010.
- [33] P. Hagmann, L. Cammoun, X. Gigandet, R. Meuli, C. J. Honey, V. J. Wedeen, and O. Sporns, *Mapping the structural core of human cerebral cortex*, *PLoS Biology* (2008).

- [34] A. M. Hermundstad, D. S. Bassett, K. S. Brown, E. M. Aminoff, D. Clewett, S. Freeman, A. Frithsen, A. Johnson, C. M. Tipper, M. B. Miller, *et. al.*, *Structural foundations of resting-state and task-based functional connectivity in the human brain*, *Proceedings of the National Academy of Sciences* **110** (2013), no. 15 6169–6174.
- [35] C. Honey, O. Sporns, L. Cammoun, X. Gigandet, J.-P. Thiran, R. Meuli, and P. Hagmann, *Predicting human resting-state functional connectivity from structural connectivity*, *Proceedings of the National Academy of Sciences* **106** (2009), no. 6 2035–2040.
- [36] C. J. Honey, R. Kötter, M. Breakspear, and O. Sporns, *Network structure of cerebral cortex shapes functional connectivity on multiple time scales*, *Proceedings of the National Academy of Sciences* **104** (2007), no. 24 10240–10245.
- [37] M. E. Raichle, *The brain’s default mode network*, *Annual Review of Neuroscience* **38** (2015) 433–447.
- [38] M. W. Cole, D. S. Bassett, J. D. Power, T. S. Braver, and S. E. Petersen, *Intrinsic and task-evoked network architectures of the human brain*, *Neuron* **83** (2014), no. 1 238–251.
- [39] E. N. Davison, K. J. Schlesinger, D. S. Bassett, M.-E. Lynall, M. B. Miller, S. T. Grafton, and J. M. Carlson, *Brain network adaptability across task states*, *PLoS Computational Biology* **11** (2015), no. 1.
- [40] C. O. Becker, S. Pequito, G. J. Pappas, M. B. Miller, S. T. Grafton, D. S. Bassett, and V. M. Preciado, *Spectral mapping of brain functional connectivity from diffusion imaging*, *Scientific Reports* **8** (2018), no. 1 1–15.
- [41] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, *arXiv preprint arXiv:1312.6114* (2013).
- [42] D. M. Barch, G. C. Burgess, M. P. Harms, S. E. Petersen, B. L. Schlaggar, M. Corbetta, M. F. Glasser, S. Curtiss, S. Dixit, C. Feldt, *et. al.*, *Function in the human connectome: task-fmri and individual differences in behavior*, *NeuroImage* **80** (2013) 169–189.
- [43] M. D. Greicius, K. Supekar, V. Menon, and R. F. Dougherty, *Resting-state functional connectivity reflects structural connectivity in the default mode network*, *Cerebral Cortex* **19** (2009), no. 1 72–78.
- [44] A. Baddeley, C. Jarrold, and F. Vargha-Khadem, *Working memory and the hippocampus*, *Journal of Cognitive Neuroscience* **23** (2011), no. 12 3855–3861.

- [45] G. Zhang, B. Cai, A. Zhang, J. M. Stephen, T. W. Wilson, V. D. Calhoun, and Y.-P. Wang, *Estimating dynamic functional brain connectivity with a sparse hidden markov model*, *IEEE Transactions on Medical Imaging* **39** (2019), no. 2 488–498.
- [46] L. Li, D. Pluta, B. Shahbaba, N. Fortin, H. Ombao, and P. Baldi, *Modeling dynamic functional connectivity with latent factor gaussian processes*, *Advances in Neural Information Processing Systems* **32** (2019) 8263–8273.
- [47] B.-H. Kim and J. C. Ye, *Understanding graph isomorphism network for rs-fmri functional connectivity analysis*, *Frontiers in Neuroscience* **14** (2020) 630.
- [48] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, *Grad-cam: Visual explanations from deep networks via gradient-based localization*, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.
- [49] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, *How powerful are graph neural networks?*, in *International Conference on Learning Representations*, 2018.
- [50] X. Li, Y. Zhou, N. C. Dvornek, M. Zhang, J. Zhuang, P. Ventola, and J. S. Duncan, *Pooling regularized graph neural network for fmri biomarker analysis*, *Medical Image Computing and Computer-assisted Intervention (MICCAI)* **12267** (2020) 625–635.
- [51] F. Noman, C.-M. Ting, H. Kang, R. C. W. Phan, B. D. Boyd, W. D. Taylor, and H. Ombao, *Graph autoencoders for embedding learning in brain networks and major depressive disorder identification*, 2021.
- [52] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, *Graph wavenet for deep spatial-temporal graph modeling*, *International Joint Conferences on Artificial Intelligence (IJCAI)* (2019).
- [53] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, *Temporal convolutional networks: A unified approach to action segmentation*, in *Computer Vision – ECCV 2016 Workshops* (G. Hua and H. Jégou, eds.), (Cham), pp. 47–54, Springer International Publishing, 2016.
- [54] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, in *International Conference on Learning Representations (ICLR)*, 2017.
- [55] C. Song, Y. Lin, S. Guo, and H. Wan, *Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting*, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 914–921, 2020.

- [56] Z. Liu, H. Zhang, Z. Chen, Z. Wang, and W. Ouyang, *Disentangling and unifying graph convolutions for skeleton-based action recognition*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 143–152, 2020.
- [57] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, *Structured sequence modeling with graph convolutional recurrent networks*, in *International Conference on Neural Information Processing*, pp. 362–373, Springer, 2018.
- [58] L. Ruiz, F. Gama, and A. Ribeiro, *Gated graph recurrent neural networks*, *IEEE Transactions on Signal Processing* **68** (2020) 6303–6318.
- [59] M. Sundararajan, A. Taly, and Q. Yan, *Axiomatic attribution for deep networks*, in *International Conference on Machine Learning*, pp. 3319–3328, PMLR, 2017.
- [60] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, *Conditional image generation with pixelcnn decoders*, in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 4797–4805, 2016.
- [61] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, *WaveNet: A Generative Model for Raw Audio*, in *Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, p. 125, 2016.
- [62] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, *Simplifying graph convolutional networks*, in *International Conference on Machine Learning*, pp. 6861–6871, PMLR, 2019.
- [63] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, *Hierarchical graph representation learning with differentiable pooling*, in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 4805–4815, 2018.
- [64] N. Lauharatanahirun, K. Bansal, S. M. Thurman, J. M. Vettel, B. Giesbrecht, S. Grafton, J. C. Elliott, E. Flynn-Evans, E. Falk, and J. O. Garcia, *Flexibility of brain regions during working memory curtails cognitive consequences to lack of sleep*, *arXiv preprint arXiv:2009.07233* (2020).
- [65] A. Schaefer, R. Kong, E. M. Gordon, T. O. Laumann, X.-N. Zuo, A. J. Holmes, S. B. Eickhoff, and B. T. Yeo, *Local-global parcellation of the human cerebral cortex from intrinsic functional connectivity mri*, *Cerebral Cortex* **28** (2018), no. 9 3095–3114.

- [66] N. Tishby and N. Zaslavsky, *Deep learning and the information bottleneck principle*, in *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5, IEEE, 2015.
- [67] S. Brody, U. Alon, and E. Yahav, *How attentive are graph attention networks?*, 2021.
- [68] W. L. Hamilton, R. Ying, and J. Leskovec, *Inductive representation learning on large graphs*, in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.
- [69] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, *Masked label prediction: Unified message passing model for semi-supervised classification*, 2021.
- [70] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, *A transformer-based framework for multivariate time series representation learning*, in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2114–2124, 2021.
- [71] A. B. Wiltschko, B. Sanchez-Lengeling, B. Lee, E. Reif, J. Wei, K. J. McCloskey, L. Colwell, W. Qian, and Y. Wang, *Evaluating attribution for graph neural networks*, *Google Research* (2020).
- [72] J. J. Todd and R. Marois, *Capacity limit of visual short-term memory in human posterior parietal cortex*, *Nature* **428** (2004), no. 6984 751–754.
- [73] J. Kim, E. A. Wasserman, L. Castro, and J. H. Freeman, *Anterior cingulate cortex inactivation impairs rodent visual selective attention and prospective memory.*, *Behavioral Neuroscience* **130** (2016), no. 1 75.
- [74] R. Leech and D. J. Sharp, *The role of the posterior cingulate cortex in cognition and disease*, *Brain* **137** (2014), no. 1 12–32.
- [75] J. M. Carlson, F. Beacher, K. S. Reinke, R. Habib, E. Harmon-Jones, L. R. Mujica-Parodi, and G. Hajcak, *Nonconscious attention bias to threat is correlated with anterior cingulate cortex gray matter volume: a voxel-based morphometry result and replication*, *Neuroimage* **59** (2012), no. 2 1713–1718.
- [76] J. M. Carlson, J. Cha, and L. R. Mujica-Parodi, *Functional and structural amygdala–anterior cingulate connectivity correlates with attentional bias to masked fearful faces*, *Cortex* **49** (2013), no. 9 2595–2600.
- [77] R. H. Grabner, G. Reishofer, K. Koschutnig, and F. Ebner, *Brain correlates of mathematical competence in processing mathematical representations*, *Frontiers in Human Neuroscience* **5** (2011) 130.

- [78] R. M. Friedrich and A. D. Friederici, *Mathematical logic in the human brain: semantics*, *PLoS One* **8** (2013), no. 1 e53699.
- [79] S. P. Drummond, A. Bischoff-Grethe, D. F. Dinges, L. Ayalon, S. C. Mednick, and M. Meloy, *The neural basis of the psychomotor vigilance task*, *Sleep* **28** (2005), no. 9 1059–1068.
- [80] E. J. Allen, G. St-Yves, Y. Wu, J. L. Breedlove, J. S. Prince, L. T. Dowdle, M. Nau, B. Caron, F. Pestilli, I. Charest, *et. al.*, *A massive 7T fMRI dataset to bridge cognitive neuroscience and artificial intelligence*, *Nature neuroscience* **25** (2022), no. 1 116–126.
- [81] N. Chang, J. A. Pyles, A. Gupta, M. J. Tarr, and E. M. Aminoff, *BOLD5000: A public fMRI dataset of 5000 images*, *arXiv preprint arXiv:1809.01281* (2018).
- [82] G. Shen, K. Dwivedi, K. Majima, T. Horikawa, and Y. Kamitani, *End-to-end deep image reconstruction from human brain activity*, *Frontiers in Computational Neuroscience* (2019) 21.
- [83] R. VanRullen and L. Reddy, *Reconstructing faces from fMRI patterns using deep generative neural networks*, *Communications biology* **2** (2019), no. 1 1–10.
- [84] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, *Microsoft coco: Common objects in context*, in *European conference on computer vision*, pp. 740–755, Springer, 2014.
- [85] Z. Liu, P. Luo, X. Wang, and X. Tang, *Large-scale celebfaces attributes (celeba) dataset*, *Retrieved August* **15** (2018), no. 2018 11.
- [86] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database*, in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.
- [87] Y. Akamatsu, R. Harakawa, T. Ogawa, and M. Haseyama, *Perceived image decoding from brain activity using shared information of multi-subject fmri data*, *IEEE Access* **9** (2021) 26593–26606.
- [88] S. Takada, R. Togo, T. Ogawa, and M. Haseyama, *Question answering for estimation of seen image contents from multi-subject fmri responses*, in *2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)*, pp. 712–713, IEEE, 2020.
- [89] R. Beliy, G. Gaziv, A. Hoogi, F. Strappini, T. Golan, and M. Irani, *From voxels to pixels and back: Self-supervision in natural-image reconstruction from fmri*, *Advances in Neural Information Processing Systems* **32** (2019).

- [90] T. Fang, Y. Qi, and G. Pan, *Reconstructing perceptive images from brain activity by shape-semantic gan*, *Advances in Neural Information Processing Systems* **33** (2020) 13038–13048.
- [91] G. Gaziv, R. Belyi, N. Granot, A. Hoogi, F. Strappini, T. Golan, and M. Irani, *Self-supervised natural image reconstruction and rich semantic classification from brain activity*, *bioRxiv* (2020).
- [92] M. Mozafari, L. Reddy, and R. VanRullen, *Reconstructing natural scenes from fMRI patterns using bigbigan*, in *2020 International joint conference on neural networks (IJCNN)*, pp. 1–8, IEEE, 2020.
- [93] Z. Ren, J. Li, X. Xue, X. Li, F. Yang, Z. Jiao, and X. Gao, *Reconstructing seen image from brain activity by visually-guided cognitive representation and adversarial learning*, *NeuroImage* **228** (2021) 117602.
- [94] G. Shen, T. Horikawa, K. Majima, and Y. Kamitani, *Deep image reconstruction from human brain activity*, *PLoS computational biology* **15** (2019), no. 1 e1006633.
- [95] Y. Cao, C. Summerfield, H. Park, B. L. Giordano, and C. Kayser, *Causal inference in the multisensory brain*, *Neuron* **102** (2019), no. 5 1076–1087.
- [96] A. A. Ghazanfar and C. E. Schroeder, *Is neocortex essentially multisensory?*, *Trends in Cognitive Sciences* **10** (2006), no. 6 278–285.
- [97] A. Pasqualotto, M. L. Dumitru, and A. Myachykov, *Multisensory integration: Brain, body, and world*, *Frontiers in Psychology* **6** (2016) 2046.
- [98] P. Pietrini, M. L. Furey, E. Ricciardi, M. I. Gobbini, W.-H. C. Wu, L. Cohen, M. Guazzelli, and J. V. Haxby, *Beyond sensory images: Object-based representation in the human ventral pathway*, *Proceedings of the National Academy of Sciences* **101** (2004), no. 15 5658–5663.
- [99] S. F. Popham, A. G. Huth, N. Y. Bilenko, F. Deniz, J. S. Gao, A. O. Nunez-Elizalde, and J. L. Gallant, *Visual and linguistic semantic representations are aligned at the border of human visual cortex*, *Nature Neuroscience* **24** (2021), no. 11 1628–1636.
- [100] B. Choksi, M. Mozafari, R. Vanrullen, and L. Reddy, *Multimodal neural networks better explain multivoxel patterns in the hippocampus*, in *Neural Information Processing Systems (NeurIPS) conference: 3rd Workshop on Shared Visual Representations in Human and Machine Intelligence (SVRHM 2021)*, 2021.
- [101] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et. al.*, *Learning transferable visual models from natural language supervision*, in *International Conference on Machine Learning*, pp. 8748–8763, PMLR, 2021.

- [102] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig, *Scaling up visual and vision-language representation learning with noisy text supervision*, in *International Conference on Machine Learning*, pp. 4904–4916, PMLR, 2021.
- [103] A. G. Huth, S. Nishimoto, A. T. Vu, and J. L. Gallant, *A continuous semantic space describes the representation of thousands of object and action categories across the human brain*, *Neuron* **76** (2012), no. 6 1210–1224.
- [104] A. Doerig, T. C. Kietzmann, E. Allen, Y. Wu, T. Naselaris, K. Kay, and I. Charest, *Semantic scene descriptions as an objective of human vision*, *arXiv preprint arXiv:2209.11737* (2022).
- [105] A. Van den Oord, Y. Li, and O. Vinyals, *Representation learning with contrastive predictive coding*, *arXiv e-prints* (2018) arXiv–1807.
- [106] Y. Zhou, R. Zhang, C. Chen, C. Li, C. Tensmeyer, T. Yu, J. Gu, J. Xu, and T. Sun, *Lafite: Towards language-free training for text-to-image generation*, *arXiv preprint arXiv:2111.13792* (2021).
- [107] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, *Analyzing and improving the image quality of StyleGAN*, in *Proc. CVPR*, 2020.
- [108] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, *Training generative adversarial networks with limited data*, in *Proc. NeurIPS*, 2020.
- [109] Y. Shen, J. Gu, X. Tang, and B. Zhou, *Interpreting the latent space of gans for semantic face editing*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9243–9252, 2020.
- [110] M. Kang and J. Park, *Contragan: Contrastive learning for conditional image generation*, *Advances in Neural Information Processing Systems* **33** (2020) 21357–21369.
- [111] J. Jeong and J. Shin, *Training gans with stronger augmentations via contrastive discriminator*, *arXiv preprint arXiv:2103.09742* (2021).
- [112] P. Sharma, N. Ding, S. Goodman, and R. Soiccut, *Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning*, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2556–2565, 2018.
- [113] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, *Gans trained by a two time-scale update rule converge to a local nash equilibrium*, *Advances in neural information processing systems* **30** (2017).



- [114] A. Bardes, J. Ponce, and Y. LeCun, *Vicreg: Variance-invariance-covariance regularization for self-supervised learning*, *arXiv preprint arXiv:2105.04906* (2021).
- [115] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, *Rethinking the inception architecture for computer vision*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.
- [116] T. Horikawa and Y. Kamitani, *Generic decoding of seen and imagined objects using hierarchical visual features*, *Nature communications* **8** (2017), no. 1 1–15.
- [117] Z. Rakhimberdina, Q. Jodelet, X. Liu, and T. Murata, *Natural image reconstruction from fMRI using deep learning: A survey*, *Frontiers in neuroscience* **15** (2021) 795488.
- [118] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, *Zero-shot text-to-image generation*, in *International Conference on Machine Learning*, pp. 8821–8831, PMLR, 2021.
- [119] A. Van Den Oord, O. Vinyals, *et. al.*, *Neural discrete representation learning*, *Advances in neural information processing systems* **30** (2017).
- [120] J.-B. Alayrac, A. Rezasens, R. Schneider, R. Arandjelović, J. Ramapuram, J. De Fauw, L. Smaira, S. Dieleman, and A. Zisserman, *Self-supervised multimodal versatile networks*, *Advances in Neural Information Processing Systems* **33** (2020) 25–37.
- [121] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, *Hierarchical text-conditional image generation with clip latents*, *arXiv preprint arXiv:2204.06125* (2022).
- [122] P. Bashivan, K. Kar, and J. J. DiCarlo, *Neural population control via deep image synthesis*, *Science* **364** (2019), no. 6439 eaav9436.
- [123] S. L. Brunton and J. N. Kutz, *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.
- [124] C. F. Stevens, *What the fly’s nose tells the fly’s brain*, *Proceedings of the National Academy of Sciences* **112** (2015), no. 30 9460–9465.
- [125] O. Rose, J. Johnson, B. Wang, and C. R. Ponce, *Visual prototypes in the ventral stream are attuned to complexity and gaze behavior*, *Nature communications* **12** (2021), no. 1 1–16.
- [126] J. Tang, A. LeBel, S. Jain, and A. G. Huth, *Semantic reconstruction of continuous language from non-invasive brain recordings*, *bioRxiv* (2022).

- [127] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust principal component analysis?*, *Journal of the ACM (JACM)* **58** (2011), no. 3 1–37.
- [128] G. J. Brouwer and D. J. Heeger, *Decoding and reconstructing color from responses in human visual cortex*, *Journal of Neuroscience* **29** (2009), no. 44 13992–14003.
- [129] D. L. Barack and J. W. Krakauer, *Two views on the cognitive brain*, *Nature Reviews Neuroscience* **22** (2021), no. 6 359–371.
- [130] M. Khosla, N. A. R. Murty, and N. Kanwisher, *A highly selective response to food in human visual cortex revealed by hypothesis-free voxel decomposition*, *Current Biology* (2022).
- [131] M. S. Gazzaniga, *The bisected brain*. No. 2. Appleton-Century-Crofts, 1970.
- [132] J. Sergent, *The cerebral balance of power: Confrontation or cooperation?*, *Journal of Experimental Psychology: Human Perception and Performance* **8** (1982), no. 2 253.
- [133] J. Christie, J. P. Ginsberg, J. Steedman, J. Fridriksson, L. Bonilha, and C. Rorden, *Global versus local processing: seeing the left side of the forest and the right side of the trees*, *Frontiers in human neuroscience* **6** (2012) 28.
- [134] L. C. Robertson and R. Ivry, *Hemispheric asymmetries: Attention to visual and auditory primitives*, *Current Directions in Psychological Science* **9** (2000), no. 2 59–63.
- [135] K. V. Haak, J. Winawer, B. M. Harvey, R. Renken, S. O. Dumoulin, B. A. Wandell, and F. W. Cornelissen, *Connective field modeling*, *Neuroimage* **66** (2013) 376–384.
- [136] T. Knapen, *Topographic connectivity reveals task-dependent retinotopic processing throughout the human brain*, *Proceedings of the National Academy of Sciences* **118** (2021), no. 2 e2017032118.
- [137] S. M. Lundberg and S.-I. Lee, *A unified approach to interpreting model predictions*, *Advances in neural information processing systems* **30** (2017).
- [138] S. Thorpe, *Local vs. distributed coding*, *Intellectica* **8** (1989), no. 2 3–40.
- [139] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick, *Neuroscience-inspired artificial intelligence*, *Neuron* **95** (2017), no. 2 245–258.
- [140] A. Ahumada Jr, *Perceptual classification images from vernier acuity masked by noise*, *Perception* **25** (1996), no. 1\_suppl 2–2.

- [141] R. F. Murray, *Classification images: A review*, *Journal of vision* **11** (2011), no. 5 2–2.
- [142] V. Marmarelis, *Analysis of physiological systems: The white-noise approach*. Springer Science & Business Media, 2012.
- [143] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et. al.*, *Gradient-based learning applied to document recognition*, *Proceedings of the IEEE* **86** (1998), no. 11 2278–2324.
- [144] H. Xiao, K. Rasul, and R. Vollgraf, *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*, *arXiv preprint arXiv:1708.07747* (2017).
- [145] A. Krizhevsky, G. Hinton, *et. al.*, *Learning multiple layers of features from tiny images*, tech. rep., Citeseer, 2009.
- [146] A. Borji and L. Itti, *State-of-the-art in visual attention modeling*, *IEEE transactions on pattern analysis and machine intelligence* **35** (2012), no. 1 185–207.
- [147] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in *Advances in neural information processing systems*, pp. 1097–1105, 2012.
- [148] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [149] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, *Do imagenet classifiers generalize to imagenet?*, *arXiv preprint arXiv:1902.10811* (2019).
- [150] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, *Intriguing properties of neural networks*, *arXiv preprint arXiv:1312.6199* (2013).
- [151] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, *arXiv preprint arXiv:1412.6572* (2014).
- [152] C. Vondrick, H. Pirsiavash, A. Oliva, and A. Torralba, *Learning visual biases from human imagination*, in *Advances in neural information processing systems*, pp. 289–297, 2015.
- [153] R. C. Fong, W. J. Scheirer, and D. D. Cox, *Using human brain activity to guide machine learning*, *Scientific reports* **8** (2018), no. 1 5397.

- [154] M. Meister, J. Pine, and D. A. Baylor, *Multi-neuronal signals from the retina: acquisition and analysis*, *Journal of neuroscience methods* **51** (1994), no. 1 95–106.
- [155] H. M. Sakai and K. Naka, *Signal transmission in the catfish retina. v. sensitivity and circuit*, *Journal of Neurophysiology* **58** (1987), no. 6 1329–1350.
- [156] R. C. Reid and J.-M. Alonso, *Specificity of monosynaptic connections from thalamus to visual cortex*, *Nature* **378** (1995), no. 6554 281.
- [157] G. C. DeAngelis, I. Ohzawa, and R. Freeman, *Spatiotemporal organization of simple-cell receptive fields in the cat’s striate cortex. ii. linearity of temporal and spatial summation*, *Journal of Neurophysiology* **69** (1993), no. 4 1118–1135.
- [158] J. P. Jones and L. A. Palmer, *The two-dimensional spatial structure of simple receptive fields in cat striate cortex*, *Journal of neurophysiology* **58** (1987), no. 6 1187–1211.
- [159] O. Schwartz, J. W. Pillow, N. C. Rust, and E. P. Simoncelli, *Spike-triggered neural characterization*, *Journal of vision* **6** (2006), no. 4 13–13.
- [160] R. A. Sandler and V. Z. Marmarelis, *Understanding spike-triggered covariance using wiener theory for receptive field identification*, *Journal of vision* **15** (2015), no. 9 16–16.
- [161] I. M. Park and J. W. Pillow, *Bayesian spike-triggered covariance analysis*, in *Advances in neural information processing systems*, pp. 1692–1700, 2011.
- [162] P. Dayan, L. F. Abbott, *et. al.*, *Theoretical neuroscience*, vol. 806. Cambridge, MA: MIT Press, 2001.
- [163] M. R. Greene, A. P. Botros, D. M. Beck, and L. Fei-Fei, *Visual noise from natural scene statistics reveals human scene category representations*, *arXiv preprint arXiv:1411.5331* (2014).
- [164] A. Nguyen, J. Yosinski, and J. Clune, *Deep neural networks are easily fooled: High confidence predictions for unrecognizable images*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.
- [165] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, *Adversarial patch*, *arXiv preprint arXiv:1712.09665* (2017).
- [166] A. Nguyen, J. Yosinski, and J. Clune, *Understanding neural networks via feature visualization: A survey*, *arXiv preprint arXiv:1904.08939* (2019).
- [167] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks*, in *European conference on computer vision*, pp. 818–833, Springer, 2014.

- [168] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, *Visualizing higher-layer features of a deep network*, *University of Montreal* **1341** (2009), no. 3 1.
- [169] K. Simonyan, A. Vedaldi, and A. Zisserman, *Deep inside convolutional networks: Visualising image classification models and saliency maps*, *arXiv preprint arXiv:1312.6034* (2013).
- [170] A. Mahendran and A. Vedaldi, *Understanding deep image representations by inverting them*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5188–5196, 2015.
- [171] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, *Network dissection: Quantifying interpretability of deep visual representations*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6541–6549, 2017.
- [172] M. R. Cohen and W. T. Newsome, *What electrical microstimulation has revealed about the neural basis of cognition*, *Current opinion in neurobiology* **14** (2004), no. 2 169–177.
- [173] P. M. Lewis, R. H. Thomson, J. V. Rosenfeld, and P. B. Fitzgerald, *Brain neuromodulation techniques: a review*, *The neuroscientist* **22** (2016), no. 4 406–421.
- [174] T. Moore and M. Fallah, *Microstimulation of the frontal eye field and its effects on covert spatial attention*, *Journal of neurophysiology* **91** (2004), no. 1 152–162.
- [175] F. A. Wichmann and N. J. Hill, *The psychometric function: I. fitting, sampling, and goodness of fit*, *Perception & psychophysics* **63** (2001), no. 8 1293–1313.
- [176] S.-R. Afraz, R. Kiani, and H. Esteky, *Microstimulation of inferotemporal cortex influences face categorization*, *Nature* **442** (2006), no. 7103 692.
- [177] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba, *Hoggles: Visualizing object detection features*, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1–8, 2013.
- [178] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, *Inception-v4, inception-resnet and the impact of residual connections on learning*, in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [179] G. E. Hinton, S. Sabour, and N. Frosst, *Matrix capsules with em routing*, in *6th international conference on learning representations, ICLR*, 2018.
- [180] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

- [181] J. Bickle, *Revolutions in neuroscience: Tool development*, *Frontiers in systems neuroscience* **10** (2016) 24.
- [182] J. K. Liu, H. M. Schreyer, A. Onken, F. Rozenblit, M. H. Khani, V. Krishnamoorthy, S. Panzeri, and T. Gollisch, *Inference of neuronal functional circuitry with spike-triggered non-negative matrix factorization*, *Nature communications* **8** (2017), no. 1 149.
- [183] A. Wu, I. M. Park, and J. W. Pillow, *Convolutional spike-triggered covariance analysis for neural subunit models*, in *Advances in neural information processing systems*, pp. 793–801, 2015.
- [184] U. Rajashekar, A. C. Bovik, and L. K. Cormack, *Visual search in noise: Revealing the influence of structural cues by gaze-contingent classification image analysis*, *Journal of Vision* **6** (2006), no. 4 7–7.
- [185] A. Caspi, B. R. Beutter, and M. P. Eckstein, *The time course of visual information accrual guiding eye movement decisions*, *Proceedings of the National Academy of Sciences* **101** (2004), no. 35 13086–13090.
- [186] B. P. Keane, H. Lu, and P. J. Kellman, *Classification images reveal spatiotemporal contour interpolation*, *Vision Research* **47** (2007), no. 28 3460–3475.
- [187] B. Willmore and D. J. Tolhurst, *Characterizing the sparseness of neural codes*, *Network: Computation in Neural Systems* **12** (2001), no. 3 255.
- [188] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, D. Keysers, J. Uszkoreit, M. Lucic, *et. al.*, *Mlp-mixer: An all-mlp architecture for vision*, *arXiv preprint arXiv:2105.01601* (2021).
- [189] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et. al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, *arXiv preprint arXiv:2010.11929* (2020).
- [190] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in *International conference on machine learning*, pp. 448–456, PMLR, 2015.
- [191] D. Hendrycks and K. Gimpel, *Gaussian error linear units (gelus)*, *arXiv preprint arXiv:1606.08415* (2016).
- [192] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, *Randaugment: Practical automated data augmentation with a reduced search space*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 702–703, 2020.

- [193] I. Loshchilov and F. Hutter, *Fixing weight decay regularization in adam*, .
- [194] D. Lian, Z. Yu, X. Sun, and S. Gao, *As-mlp: An axial shifted mlp architecture for vision*, *arXiv preprint arXiv:2107.08391* (2021).
- [195] H. Liu, Z. Dai, D. R. So, and Q. V. Le, *Pay attention to mlps*, *arXiv preprint arXiv:2105.08050* (2021).
- [196] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, A. Joulin, G. Synnaeve, J. Verbeek, and H. Jégou, *Resmlp: Feedforward networks for image classification with data-efficient training*, *arXiv preprint arXiv:2105.03404* (2021).
- [197] Q. Hou, Z. Jiang, L. Yuan, M.-M. Cheng, S. Yan, and J. Feng, *Vision permutator: A permutable mlp-like architecture for visual recognition*, 2021.
- [198] B. Neyshabur, *Towards learning convolutions from scratch*, *Advances in Neural Information Processing Systems* **33** (2020) 8078–8088.
- [199] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, *arXiv preprint arXiv:1704.04861* (2017).
- [200] T. NVIDIA, *Nvidia tesla v100 gpu architecture*, 2017.
- [201] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, *Going deeper with image transformers*, *arXiv preprint arXiv:2103.17239* (2021).
- [202] S. d’Ascoli, H. Touvron, M. Leavitt, A. Morcos, G. Biroli, and L. Sagun, *Convit: Improving vision transformers with soft convolutional inductive biases*, *arXiv preprint arXiv:2103.10697* (2021).
- [203] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, *Swin transformer: Hierarchical vision transformer using shifted windows*, 2021.
- [204] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, *Training data-efficient image transformers & distillation through attention*, in *International Conference on Machine Learning*, pp. 10347–10357, PMLR, 2021.
- [205] H. Bao, L. Dong, and F. Wei, *Beit: Bert pre-training of image transformers*, *arXiv preprint arXiv:2106.08254* (2021).
- [206] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, *Pyramid vision transformer: A versatile backbone for dense prediction without convolutions*, *arXiv preprint arXiv:2102.12122* (2021).

- [207] J. Li, A. Hassani, S. Walton, and H. Shi, *Convmlp: Hierarchical convolutional mlps for vision*, *arXiv preprint arXiv:2109.04454* (2021).
- [208] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, in *Advances in neural information processing systems*, pp. 5998–6008, 2017.
- [209] S. Chen, E. Xie, C. Ge, D. Liang, and P. Luo, *Cyclemlp: A mlp-like architecture for dense prediction*, *arXiv preprint arXiv:2107.10224* (2021).
- [210] J.-B. Cordonnier, A. Loukas, and M. Jaggi, *On the relationship between self-attention and convolutional layers*, *arXiv preprint arXiv:1911.03584* (2019).
- [211] Z. Dai, H. Liu, Q. V. Le, and M. Tan, *Coatnet: Marrying convolution and attention for all data sizes*, *arXiv preprint arXiv:2106.04803* (2021).
- [212] J. Guo, K. Han, H. Wu, C. Xu, Y. Tang, C. Xu, and Y. Wang, *Cmt: Convolutional neural networks meet vision transformers*, *arXiv preprint arXiv:2107.06263* (2021).
- [213] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, *Attention augmented convolutional networks*, in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3286–3295, 2019.
- [214] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, *Stand-alone self-attention in vision models*, *arXiv preprint arXiv:1906.05909* (2019).
- [215] I. Bello, *Lambdanetworks: Modeling long-range interactions without attention*, *arXiv preprint arXiv:2102.08602* (2021).
- [216] A. Trockman and J. Z. Kolter, *Patches are all you need?*, *arXiv preprint arXiv:2201.09792* (2022).
- [217] J. Ho, N. Kalchbrenner, D. Weissenborn, and T. Salimans, *Axial attention in multidimensional transformers*, *arXiv preprint arXiv:1912.12180* (2019).
- [218] R. Müller, S. Kornblith, and G. E. Hinton, *When does label smoothing help?*, *Advances in neural information processing systems* **32** (2019).
- [219] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, *Deep networks with stochastic depth*, in *European conference on computer vision*, pp. 646–661, Springer, 2016.
- [220] R. Wightman, H. Touvron, and H. Jégou, *Resnet strikes back: An improved training procedure in timm*, 2021.



- [221] S. J. Luck and E. K. Vogel, *The capacity of visual working memory for features and conjunctions*, *Nature* **390** (1997), no. 6657 279–281.
- [222] S. Yantis, J. Schwarzbach, J. T. Serences, R. L. Carlson, M. A. Steinmetz, J. J. Pekar, and S. M. Courtney, *Transient neural activity in human parietal cortex during spatial attention shifts*, *Nature Neuroscience* **5** (2002), no. 10 995–1002.
- [223] M. L. Sipos, Y. Bar-Haim, R. Abend, A. B. Adler, and P. D. Bliese, *Postdeployment threat-related attention bias interacts with combat exposure to account for ptsd and anxiety symptoms in soldiers*, *Depression and Anxiety* **31** (2014), no. 2 124–129.
- [224] A. Mattarella-Micke, J. Mateo, M. N. Kozak, K. Foster, and S. L. Beilock, *Choke or thrive? the relation between salivary cortisol and math performance depends on individual differences in working memory and math-anxiety.*, *Emotion* **11** (2011), no. 4 1000.
- [225] S. Loh, N. Lamond, J. Dorrian, G. Roach, and D. Dawson, *The validity of psychomotor vigilance tasks of less than 10-minute duration*, *Behavior Research Methods, Instruments, & Computers* **36** (2004), no. 2 339–346.