The Sparse Manifold Transform and Unsupervised Learning for Signal Representation

by

Yubei Chen

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering — Electrical Engineering and Computer Sciences

and the Designated Emphasis

in

Communication, Computation, and Statistics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Bruno Olshausen, Chair
Professor Michael Lustig
Professor Pieter Abbeel

Fall 2019

The Sparse Manifold Transform and Unsupervised Learning for Signal Representation

Abstract

The Sparse Manifold Transform and Unsupervised Learning for Signal Representation

by

Yubei Chen

Doctor of Philosophy in Engineering — Electrical Engineering and Computer Sciences

and the Designated Emphasis in

Communication, Computation, and Statistics

University of California, Berkeley

Professor Bruno Olshausen, Chair

This dissertation presents three contributions on unsupervised learning. First, I describe a signal representation framework called the sparse manifold transform that combines key ideas from sparse coding, manifold learning, and slow feature analysis. It turns non-linear transformations in the primary sensory signal space into linear interpolations in a representational embedding space while maintaining approximate invertibility. The sparse manifold transform is an unsupervised and generative framework that explicitly and simultaneously models the sparse discreteness and low-dimensional manifold structure found in natural scenes. When stacked, it also models hierarchical composition. I provide a theoretical description of the transform and demonstrate properties of the learned representation on both synthetic data and natural videos. Further, the SMT also provides a unifying geometric perspective on the role of simple and complex cells. I propose a version of SMT in which neurons in the two layers correspond to simple and complex cells. This provides a new functional explanation for these cell types: Simple cells can be viewed as representing a discrete sampling of a smooth manifold in the sensor space, while complex cells can be viewed as representing localized smooth linear functions on the manifold. While each individual complex cell pools from a local region, together they tile (Sengupta *et al.* 2018) the manifold and build an untangled population representation (DiCarlo & Cox 2007), which tends to preserve the identity of the signal while straightening the transformations (Henaff *et al.* 2017). In the localized SMT, the complex cell layer is learned in an unsupervised manner based on a diffusion process. The results demonstrate that simple and complex cells are emergent properties in a neural system that is optimized to learn the manifold structure of dynamic sensory inputs and is subject to sparse connectivity constraints.

The second contribution is on the discovery of word factors and word embedding visualization. Co-occurrence statistics based word embedding techniques have proved to be very useful in extracting the semantic and syntactic representation of words as low dimensional continuous vectors. This work discovered that dictionary learning can open up these word vectors as a linear combination of more elementary word factors. I demonstrate many of the learned factors have surprisingly strong semantic or syntactic meaning corresponding to the factors previously identified manually by human inspection. Thus dictionary learning provides a powerful visualization tool for understanding word embedding representations. Furthermore, I show that the word factors can help in identifying key semantic and syntactic differences in word analogy tasks and improve upon the state-of-the-art word embedding techniques in these tasks by a large margin.

The third contribution is on a more efficient and effective way to train energy-based models (EBMs) in high-dimensional spaces. Energy-based models assign unnormalized log-probability to data samples. This functionality has a variety of applications, such as sample synthesis, data denoising, sample restoration, outlier detection, Bayesian reasoning, and many more. But training of EBMs using standard maximum likelihood is extremely slow because it requires sampling from the model distribution. Score matching potentially alleviates this problem. In particular, denoising score matching (Vincent 2011) has been successfully used to train EBMs. Using noisy data samples with one fixed noise level, these models learn fast and yield good results in data denoising (Saremi & Hyvarinen 2019). However, demonstrations of such models in high-quality sample synthesis of high dimensional data were lacking. Recently, (Song & Ermon 2019) have shown that a generative model trained by denoising score matching accomplishes excellent sample synthesis, when trained with data samples corrupted with multiple levels of noise. Both analysis and empirical evidence show that training with multiple noise levels is necessary when the data dimension is high. Leveraging this insight, a novel EBM trained with multi-scale denoising score matching is proposed. The model exhibits data generation performance comparable to state-of-the-art techniques such as GANs, and sets a new baseline for EBMs. The proposed model also provides density information and performs well in an image inpainting task.

Acknowledgments

It was very lucky for me to join Bruno's group and BAIR at UC Berkeley. Bruno is a true scientist. As a junior graduate student, I was fed with carefully selected papers, which helped me to see the path through his lens. These lasting works deeply shaped my taste of good scientific research. Besides his guidance, I was also given unparalleled freedom to pursue the problems I feel passionate about. One anecdote was that during our very first meeting, I told him I want to to understand the dream. I was glad he didn't throw me out of the window[1]. His encouragement during my search in the darkness always gave me hope.

The collaborative environment at UC Berkeley is invaluable and I'm very grateful to all of my collaborators. Especially, I would like to thank Brian Cheung for teaching me so much on deep learning and many other topics. His broad knowledge, always positive attitude, and sometime 'weird' but novel ideas made our collaboration very enjoyable. Dylan Paiton's help was critical to make the visualizatin of the SMT possible. I'm thankful for his patience with my frequent last-minute ideas. Zengyi Li's beautiful intuition in physics impressed me many times. Our efficient collaboration made me look forward to many projects to come. I was also lucky to work with Juexiao Zhang and Jiayun Wang. Their positive energy and strong motivation in research inspired me during our projects.

Many other Berkeley faculty members helped me throughout my graduate study. Fritz Sommer game me many good ideas during our discussion. His guidance and mentorship are a valuable treasure to me. I was also very lucky to have many exciting discussions with Joan Bruna before he joined Courant Institute at NYU. His sharpness in signal modeling, ability to formulate problem precisely and mathematically made our interaction unforgettable. Stella Yu taught me a lot on computer vision and showed me the experimental standard in this field. I would like to thank my dissertation committee for their support and encouragement. Pieter Abbeel's incredible positive attitude and Michael Lustig's passion for fundamental problems eased my struggles in my research program. Their critical questions and comments helped me polish the work further. Marc Rieffel's guidance and support helped me build much mathematics maturity needed in my later research. Additional, I deeply thank Kurt Keutzer for his early guidance and he also suggested me to meet with Bruno.

I thank the entire Redwood Center for Theoretical Neuroscience. Redwood Center is a unique place, where you can truly think out of the box. Its interdisciplinary background frequently helped me see problems from dramatically different angles. The company from my friends from Redwood Center, EECS department, BAIR and many other programs provided me plenty of inspirational discussions and recreational activities, which made Berkeley a never boring place. My regular training at Berkeley judo team also supplied me with some essential working energy.

Finally, I thank my family, especially my grandparents, for their lasting support and understanding. The unconditional love from them helped me get through the thick and thin during my journey.

[1]Bruno gave his students the windowed office and his office doesn't have one. The reason? We thought it was in preparation for the earthquake but he denied. ☺ Thank you!

The work reported in this dissertation is supported by NSF-IIS-1718991, NSF-DGE-1106400, NIH 1R01EB026955-01, NIH/NEI T32 EY007043, and NSF GRFP Fellowship.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

We have seen tremendous progress in supervised learning in the past decade, powered by the convolutional neural networks (CNNs). (Krizhevsky, Sutskever, *et al.* 2012) However, it is widely believed that most of the human learning happens without supervision and a principled way of doing unsupervised learning is still lacking. Further, a precise definition of unsupervised learning is missing. This motivated me to look at this fundamental problem from multiple interesting and interconnected angles to gain insight from each correspondingly. The very first question that came to my mind was: if we have not found 'the principle', then what are the important pieces? The interpretation of importance is a bit subjective and I must pick the ones that inspired me the most. Once the pieces are identified, the next question was: can we stitch and reconcile them into one, which might be closer to the ultimate principle we are looking for? These two questions led me to the sparse manifold transform (SMT), which incorporates three pervasive patterns in natural signals: *sparse discreteness*, *low dimensional manifold structure* and *hierarchical composition*. I will show the promising results from applying the SMT to natural images and natural videos.

I am strongly biased to see different structures in a similar fashion, or at least I would search for the connections. Starting from images and videos, I wondered: shall this learning principle generalize to the other signal modalities? This question motivated me to apply sparse coding and spectral methods, which are the key ideas in the SMT, to understand text representation, specifically, word embedding (Mikolov, Yih, *et al.* 2013; Pennington, Socher & C. Manning 2014). This query led to the discovery of word factors[1], the more elementary structure hidden in almost all word embedding representations. The word factors also provided interesting visualization of the word vectors, which seemed to me like black boxes for quite some time.

In my opinion, a learning theory is not just about machines. It shall also help us plow through the plethora of enigmatic and difficult to interpret results (Hupe *et al.* 2001; Angelucci & Bullier 2003; Andolina *et al.* 2007; Olshausen 2013a) yielded in neuroscience research, provide us new questions to ask, and deepen our understanding of the brain. With

[1]I will use 'factor' in a loose way. It doesn't necessarily imply multiplicative factors.

this belief, I applied the SMT to explain phenomena in visual cortex, which led to a geometric theory for complex cells.

Natural signals, e.g. natural images, are frequently highly structured, have randomness though far from i.i.d. Gaussian noise, and live in a high-dimensional ambient space. I believe the ultimate model for natural images should be probabilistic, which can be formulated as an energy function over the signal space. However, the development of the SMT is still not complete enough to make an enegy-based model. (LeCun *et al.* 2006) Nevertheless, my impatience got to me and again with the luring question: how can I construct energy-based models for natural signals in high-dimensional spaces? As CNNs provide powerful priors for signal modeling (Ulyanov *et al.* 2018), I set out to use deep energy estimator networks (Saremi, Mehrjou, *et al.* 2018) to approach this problem. This journey led me to the state-of-the-art energy-based model for natural images and a picture that somewhat unifies denoising autoencoder and EBMs.

The thoughts above are quite general and we shall dive into more concrete problems in the following.

Beyond a Known Topological Prior. The success of supervised learning powered by CNNs is undoubtable. Behind the CNNs, there are two important principles: stationarity and invariance. Given an image, stationarity means that the statistics in different locations would be the same, thus a set of features learned can be reused via convolution. On the other hand, building discriminative representations invariant to the variations within catergory[2] is a long-term goal in vision and many other fields. The invariance is usually achieved by the pooling operation in the CNNs. Max-pooling provides a simple way to achieve translational invariance. Both of these two basic operations, convolution and pooling, depend on a uniform pixel lattice on a 2D image, or in other words the image domain has a known 2-D topological prior. For an image, these operations might be very natural choices. However, human retinas are not uniform, the sampling density varies significantly from the fovea to the peripheral area. Further, different animals have different retina patterns, where many birds have two distinct fovea per eye, cats do not even have a fovea in their retinas at all! (Morris 2012) This implies that the learned features correspond to different areas in the retinas are also different and they can not be shared by convolution. Without the convolution operation, pooling operations can not be defined either. This begs for more general operations and concepts, which do not depend on a uniform sampling on a known topological prior. Based on these observations, I believe self-organization[3] is the correct path to solve the issue.

Important Principles for Unsupervised Learning. As I argued earlier, most of human learning happens without supervision and a principled way of doing unsupervised learning is still lacking. While a precise definition of unsupervised learning is missing, we can vaguely define unsupervised learning as finding the patterns among data beyond pure noise. (Ghahramani 2003) As there are many possible patterns, which should we look for? Inspired

[2]E.g. a dog in different location of a image shall always be classified as a dog

[3]In my opinion, manifold learning belongs to self-organization methods.

by David Mumford's introduction to pattern theory (Mumford & Desolneux 2010), I attempt to model three important and pervasive patterns in natural signals: *sparse discreteness*, *low dimensional manifold structure* and *hierarchical composition.* Each of these concepts have been individually explored in previous studies. For example, sparse coding (Olshausen & Field 1996; Olshausen & Field 1997) and ICA (Bell & Sejnowski 1995; Hyvärinen & Oja 2000) can learn sparse and discrete elements that make up natural signals. Manifold learning (Tenenbaum *et al.* 2000; Roweis & Saul 2000; Maaten & G. Hinton 2008; Belkin & Niyogi 2002) was proposed to model and visualize low-dimensional continuous transforms such as smooth 3D rotations or translations of a single discrete element. Deformable, compositional models (Wu *et al.* 2010; Felzenszwalb *et al.* 2010) allow for a hierarchical composition of components into a more abstract representation. I seek to model these three patterns jointly as they are almost always entangled in real-world signals and their disentangling poses an unsolved challenge.

The sparse dictionary learning and manifold learning are the first two principles I picked from the unsupervised learning literature. The third one rather comes from an efficiency concern. Arguably humans always learn from temporal sequences rather than i.i.d. data points. Natural transformations lead to continuous trajectories in the high-dimensional signal space. Utilizing the temporal sequence would make the learning much more efficient. Slow feature analysis appeared as a principle to resolve this issue. I will discuss these three principles further in Chapter 2.

To combine the key insights from the three principles in unsupervised learning, in Chapter 3, I will introduce an interpretable, generative and unsupervised learning model, the *sparse manifold transform* (SMT), which has the potential to model all three patterns simultaneously and explicitly. The SMT consists of two stages: dimensionality expansion using sparse coding followed by contraction using manifold embedding. Sparse coding provides a general feature learning operation and manifold embedding provides a more general 'pooling' to achieve invariance and equivarience. The SMT implementation is to my knowledge, the first model to bridge sparse coding and manifold learning. Furthermore, an SMT layer can be stacked to produce an unsupervised hierarchical learning network. The primary contribution the SMT is to establish a theoretical framework for the SMT by reconciling and combining the formulations and concepts from sparse coding and manifold learning. With the concepts developed in the SMT, I introduce a geometric theory to understand the relationships between the simple and complex cells in the visual cortex in Chapter 4.

Common Patterns Shared in Different Modalities. There are many different modalities of signals and it's always my interest to look for some general principles shared between seemingly unrelated structures. I will show how to use the general SMT concepts in word representation. Specifically in Chapter 5, I will discuss sparse word factors hidden in almost all word embedding representations. Specifically, every word vector can be decomposed into a linear combination of a sparse set of more elementary word factors. Thus we can use dictionary learning to visualize word vectors. Then followed by a spectral grouping[4], the

[4]Manifold learning is frequently formulated as a spectral problem.

word analogy task performance can be uniformly improved.

Learning Energy-based Models in High-dimensional Ambient Spaces. Deep convolutional neural networks provide very powerful priors for signal modeling. However, building an energy-based model in high-dimensional ambient spaces with MLE is still quite intractable. Denoising Score matching provides a powerful way to get around sampling. (Vincent 2011) Finally in Chapter 6, I will present how to learn an energy-based model in a high-dimensional space with a deep energy estimator network and denoising score matching. This work in some sense provides a unified view between energy function and denoising autoencoder.

Chapter 2

Discrete, Continuous, and Temporal

2.1 Three Important Principles Used in Unsupervised Learning

There are three important principles used in unsupervised learning:

- Discrete: Clustering is one of the most classical methods used in the the machine learning. It essentially follows a 'winner takes all' principle: Learning a set of clusters or templates and then we assign each data point to its closest cluster or template. The assginment of a data point can be treated as an 1-sparse vector. In my opinion, sparse coding is a soft clustering, which relaxes the 1-sparse requirement to 'as sparse as possible'. This relaxation is important since the hierarchical composition exists in almost all signals and it makes the learning algorithm more flexible to learn smaller parts than global templates given limited resource. As a result, sparse coding can decompose a data point into a sparse combination of the elements in the dictionary. Modeling a data point either as 1-sparse or k-sparse, it emphasizes the discrete structures in the signal.

- Continuous: Continuous structures receive a lot of attention in the vision problems. Many natural transformations (e.g. translations, rotation, scaling, shading change etc.) can be viewed as continuous. The manifold hypothesis is proposed to handle these continuous variations and it assume all the data points in a dataset lie along a low-dimensional manifold immersed in the high-dimensional signal space. This proposal complements the discrete one and they together provide a more comprehensive view to the signal modeling.

- Temporal: Arguably humans always learn from temporal sequences rather than i.i.d. data points. Natural transformations lead to continuous trajectories in the high-dimensional signal space. Shuffling the data doesn't necessarily destroy the continuous manifold strutures as we may later use other methods to recover it (e.g. finding the nearest neighbors among all of the shuffled data). However, utilizing the temporal sequence would make the learning much more efficiently.

These three principles have been repeatedly applied in many different algorithms under different contexts. We choose them as our building blocks to construct a theoretical unsupervised learning framework.

In the following sections we point out connections between three representative unsupervised learning methods: sparse coding, local linear embedding and slow feature analysis, each reflect one important unsupervised learning principle discussed above. We then develop a single framework that utilizes insights from each method to describe our model. Although we focus here on the application to image data, the concepts are general and may be applied to other types of data such as audio signals and text. All experiments performed on natural scenes used the same dataset, described in Supplement A.2.

2.2 Sparse Coding

Sparse coding attempts to approximate a data vector, $x \in \mathbb{R}^n$, as a sparse superposition of dictionary elements ϕ_i:

$$x = \Phi\,\alpha + \epsilon \tag{2.1}$$

where $\Phi \in \mathbb{R}^{n \times m}$ is a matrix with columns ϕ_i, $\alpha \in \mathbb{R}^m$ is a sparse vector of coefficients and ϵ is a vector containing independent Gaussian noise samples, which are assumed to be small relative to x. Typically $m > n$ so that the representation is *overcomplete*. For a given dictionary, Φ, the sparse code, α, of a data vector, x, can be computed in an online fashion by minimizing an energy function composed of a quadratic penalty on reconstruction error plus an L1 sparseness penalty on α.

In traditional sparse coding (Olshausen & Field 1996) approach, the coefficient vector α is computed for each input image by performing the optimization:

$$\min_{\alpha} \tfrac{1}{2}\|x - \Phi\alpha\|_2^2 + \lambda\|\alpha\|_1 \tag{2.2}$$

where λ is a sparsity trade-off penalty. After λ has been computed, then we update the dictionary by one step following the gradient of the following optimization:

$$\min_{\Phi} \tfrac{1}{2}\|x - \Phi\alpha\|_2^2 \tag{2.3}$$

To learn a dictionary from the dataset, we repreat the step 2.2 and step 2.3 until the dictionary converges. For this report, we learn a 10-20 times overcomplete dictionary (Olshausen 2013b) and impose the additional restriction of positive-only coefficients ($\alpha \geq 0$). Empirically, we find that at such an overcompleteness the interpolation behavior of the dictionary is close to locally linear. Therefore, steering the elements can be accomplished by local neighborhood interpolation. High overcompleteness with a positive-only constraint makes the relative geometry of the dictionary elements more explicit. The positive-only constraint gives us our modified optimization:

$$\min_{\alpha} \tfrac{1}{2}\|x - \Phi\alpha\|_2^2 + \lambda\|\alpha\|_1, \text{ s.t.} \alpha \geq 0, \tag{2.4}$$

The dictionary itself is adapted to the statistics of the data so as to maximize the sparsity of α. The resulting dictionary often provides important insights about the structure of the data. For natural images, the dictionary elements become 'Gabor-like'—i.e., spatially localized, oriented and bandpass—and form a tiling over different locations, orientations and scales due to the natural transformations of objects in the world. In Figure 2.1 we show a random subset of the learned dictionary.

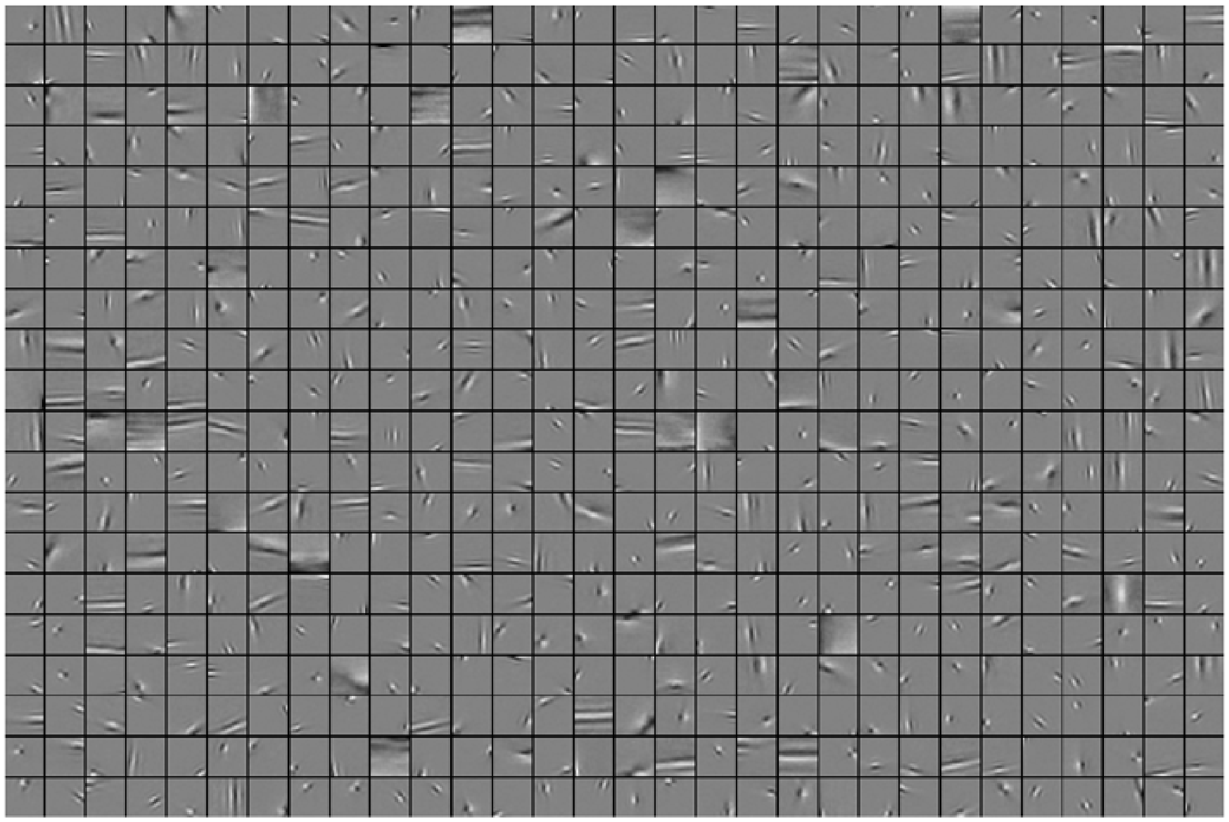

Figure 2.1: A random sample of 600 dictionary elements from the learned dictionary (out of 4000 units). Details are better discerned by zoomed in on a monitor.

The sparse code of an image provides a representation that makes explicit the structure contained in the image. However the dictionary is typically *unordered*, and so the sparse code will lose the topological organization that was inherent in the image. The pioneering works of

(Hyvärinen & P. Hoyer 2000; Hyvärinen, P. O. Hoyer, *et al.* 2001) and (Osindero *et al.* 2006) addressed this problem by specifying a fixed 2D topology over the dictionary elements that groups them according to the co-occurrence statistics of their coefficients. Other works learn the group structure from a statistical approach (Lyu & Simoncelli 2008; Ballé *et al.* 2015; Köster & Hyvärinen 2010), but do not make explicit the underlying topological structure. Some previous topological approaches (A. B. Lee *et al.* 2003; De Silva & G. E. Carlsson 2004; G. Carlsson *et al.* 2008) used non-parametric methods to reveal the low-dimensional geometrical structure in local image patches, which motivated us to look for the connection between sparse coding and geometry. *From this line of inquiry, we have developed what we believe to be the first mathematical formulation for learning the general geometric embedding of dictionary elements when trained on natural scenes.*

Another observation motivating this work is that the representation computed using overcomplete sparse coding can exhibit large variability for time-varying inputs that themselves have low variability from frame to frame (Rozell *et al.* 2008). While some amount of variability is to be expected as image features move across different dictionary elements, the variation can appear unstructured without information about the topological relationship of the dictionary. In section 3.2 and section 3.3, we show that considering the joint spatio-temporal regularity in natural scenes can allow us to learn the dictionary's group structure and produce a representation with smooth variability from frame to frame (Figure 3.4).

2.3 Manifold Learning

Natural transformations in images usually leads to highly non-linear trajectories in the signal (image) space, even for the translation operation. In Figure 2.2 we show a translation of a letter 'A' from the left to the right, this is a non-linear since a linear interpolation between the left and the right image would lead to a superposed 'A' shown in the bottom. If we model the transformation as a smooth manifold, the nonlinear manifold interpolation would lead to the 'A' in between shown on the top.

In manifold learning, one assumes that the data occupy a low-dimensional, smooth manifold embedded in the high-dimensional signal space. A smooth manifold is locally equivalent to a Euclidean space and therefore each of the data points can be linearly reconstructed by using the neighboring data points. The Locally Linear Embedding (LLE) algorithm (Roweis & Saul 2000) first finds the neighbors of each data point in the whole dataset and then reconstructs each data point linearly from its neighbors. It then embeds the dataset into a low-dimensional Euclidean space by solving a generalized eigendecomposition problem.

The first step of LLE has the same linear formulation as sparse coding (2.1), with Φ being the whole dataset rather than a learned dictionary, i.e., $\Phi = X$, where X is the data matrix. The coefficients, α, correspond to the linear interpolation weights used to reconstruct a datapoint, x, from its K-nearest neighbors, resulting in a K-sparse code. (In other work (Elhamifar & Vidal 2011), α is inferred by sparse approximation, which provides better separation between manifolds nearby in the same space.) Importantly, once the

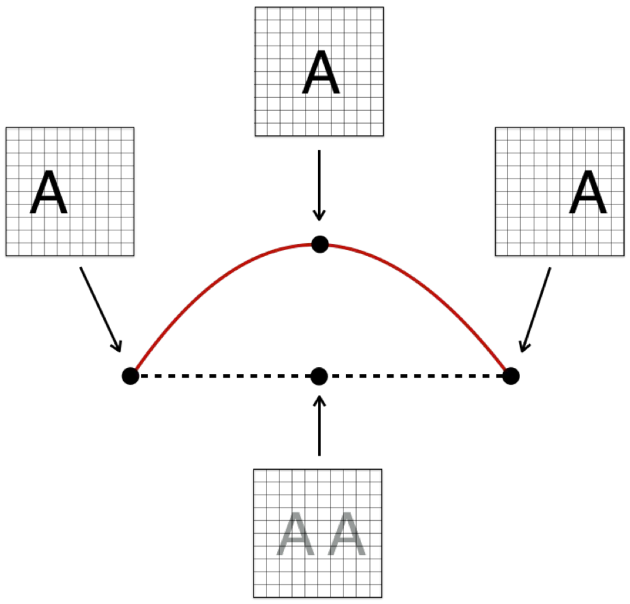

Figure 2.2: Even the most typical translation operation would lead to a highly non-linear trajectory in the image space.

embedding of the dataset $X \to Y$ is computed, the embedding of a new point $x^{\text{NEW}} \to y^{\text{NEW}}$ is obtained by a simple linear projection of its sparse coefficients. That is, if α^{NEW} is the K-sparse code of x^{NEW}, then $y^{\text{NEW}} = Y\,\alpha^{\text{NEW}}$. Viewed this way, *the dictionary may be thought of as a discrete sampling of a continuous manifold, and the sparse code of a data point provides the interpolation coefficients for determining its coordinates on the manifold.* However, using the entire dataset as the dictionary is cumbersome and inefficient in practice.

Several authors (De Silva & Tenenbaum 2004; Silva *et al.* 2006; Vladymyrov & Carreira-Perpinán 2013) have realized that it is unnecessary to use the whole dataset as a dictionary. A random subset of the data or a set of cluster centers can be good enough to preserve the manifold structure, making learning more efficient. Going forward, we refer to these as *landmarks*. In Locally Linear Landmarks (LLL) (Vladymyrov & Carreira-Perpinán 2013), the authors compute two linear interpolations for each data point x:

$$x = \Phi_{\text{LM}}\,\alpha + n \tag{2.5}$$

$$x = \Phi_{\text{DATA}}\,\gamma + n' \tag{2.6}$$

where Φ_{LM} is a dictionary of landmarks and Φ_{DATA} is a dictionary composed of the whole dataset. As in LLE, α and γ are coefficient vectors inferred using KNN solvers (where the γ coefficient corresponding to x is forced to be 0). We can substitute the solutions to equation (2.5) into Φ_{DATA}, giving $\Phi_{\text{DATA}} \approx \Phi_{\text{LM}}A$, where the j^{th} column of the matrix A is a unique vector α_j. This leads to an interpolation relationship:

$$\Phi_{\text{LM}}\alpha \approx \Phi_{\text{LM}} A \gamma \tag{2.7}$$

The authors sought to embed the landmarks into a low dimensional Euclidean space using an embedding matrix, P_{LM}, such that the interpolation relationship in equation (2.7) still holds:

$$P_{\text{LM}}\alpha \approx P_{\text{LM}} A \gamma \tag{2.8}$$

where we use the same α and γ vectors that allowed for equality in equations (2.5) and (2.6). P_{LM} is an embedding matrix for Φ_{LM} such that each of the columns of P represents an embedding of a landmark. P_{LM} can be derived by solving a generalized eigendecomposition problem (Vladymyrov & Carreira-Perpinán 2013).

The similarity between equation (2.1) and equation (2.5) provides an intuition to bring sparse coding and manifold learning closer together. However, LLL still has a difficulty in that it requires a nearest neighbor search. We posit that temporal information provides a more natural and efficient solution.

2.4 Slow Feature Analysis

The general idea of imposing a 'slowness prior' was initially proposed by (Földiák 1991) and (Wiskott & Sejnowski 2002) to extract invariant or slowly varying features from temporal sequences rather than using static orderless data points. While it is still common practice in both sparse coding and manifold learning to collect data in an orderless fashion, other work has used time-series data to learn spatiotemporal representations (van Hateren & Ruderman 1998; Olshausen 2003; Hyvärinen, Hurri, *et al.* 2003; Goroshin, Bruna, *et al.* 2015) or to disentangle form and motion (Berkes *et al.* 2009; Cadieu & Olshausen 2012; Denton & Birodkar 2017). In Figure 2.3, we show the classical swiss roll example to illustrate the benefit of having access to the temporal sequences. Let's assume the data points occupy a 2D swiss-roll manifold in the 3D space. It's possible to perform manifold learning with enough i.i.d. samples (the gray dots). However, haveing the access to the temporal sequences (the red curves), we can start to learn the geometry without waiting till we have collected enough samples. In high-dimensional signal space, the samples are very sparse and temporal sequences can make learning much more efficient. Though we do not claim the temporal sequences are necessary, the combination of topography and temporal coherence in (Hyvärinen, Hurri, *et al.* 2003) provides a strong motivation for this work. Later, we will show the case when temporal sequences are not available.

Here, we utilize *temporal adjacency* to determine the nearest neighbors in the embedding space (eq. 2.6) by specifically minimizing the second-order temporal derivative, implying that video sequences form *linear trajectories* in the manifold embedding space. So eq.2.6 can be solved with a deterministic solution rather than performing expensive nearest neighbor search followed by local linear interpolation:

$$x_t = \Phi_{\text{DATA\_SEQUENCE}}\gamma_t + n' \tag{2.9}$$

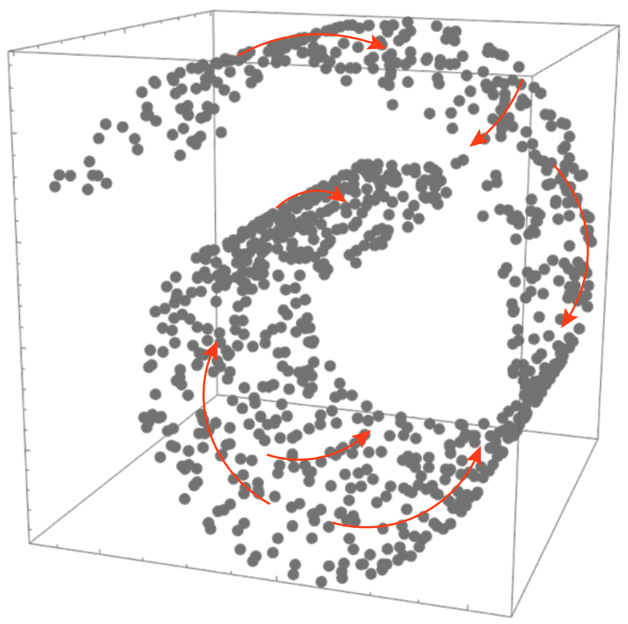

Figure 2.3: With enough i.i.d. samples (the gray dots), it's certainly possible to understand the manifold geometry in the signal space. However the temporal sequences (the red curves) provides information about the the local geometry without waiting we have collected enough random samples.

where γ_t has all zero except for the $(t-1)^{th}$ and $(t+1)^{th}$ entries been $\frac{1}{2}$.

A similar approach was recently used by (Goroshin, Mathieu, *et al.* 2015) to linearize transformations in natural video. This is a variation of 'slowness' that makes the connection to manifold learning more explicit. It also connects to the ideas of manifold flattening (DiCarlo & Cox 2007) or straightening (Hénaff *et al.* 2019) which are hypothesized to underlie perceptual representations in the brain.

Chapter 3

The Sparse Manifold Transform

3.1 Functional Embedding Perspective

The SMT framework differs from the classical manifold learning approach in that it relies on the concept of *functional embedding* as opposed to embedding individual data points. We explain this concept here before turning to the sparse manifold transform in section 3.2.

In classical manifold learning (Huo *et al.* 2007), for a m-dimensional compact manifold, it is typical to solve a generalized eigenvalue decomposition problem and preserve the 2nd to the $(d+1)^{\text{th}}$ trailing eigenvectors as the embedding matrix $P_{\text{C}} \in \mathbb{R}^{d \times N}$, where d is as small as possible (parsimonious) such that the embedding preserves the topology of the manifold (usually, $m \leq d \leq 2m$ due to the strong Whitney embedding theorem (J. Lee 2012)) and N is the number of data points or landmarks to embed. It is conventional to view the columns of an embedding matrix, P_{C}, as an embedding to an Euclidean space, which is (at least approximately) topologically-equivalent to the data manifold. Each of the rows of P_{C} is treated as a coordinate of the underlying manifold. One may think of a point on the manifold as a single, constant-amplitude delta function with the manifold as its domain. Classical manifold embedding turns a non-linear transformation (i.e., a moving delta function on the manifold) in the original signal space into a simple linear interpolation in the embedding space. This approach is effective for visualizing data in a low-dimensional space and compactly representing the underlying geometry, but less effective when the underlying function is not a single delta function.

In this work we seek to move beyond the single delta-function assumption, because natural images are not well described as a single point on a continuous manifold of fixed dimensionality. For any reasonably sized image region (e.g., a 16×16 pixel image patch), there could be multiple edges moving in different directions, or the edge of one occluding surface may move over another, or the overall appearance may change as features enter or exit the region. Such changes will cause the manifold dimensionality to vary substantially, so that the signal structure is no longer well-characterized as a manifold.

We propose instead to think of any given image patch as consisting of h discrete structures simultaneously moving over the same underlying manifold - i.e., as h delta functions, or *an h-sparse function* on the smooth manifold. This idea is illustrated in figure 3.1. First, let us organize the Gabor-like dictionary learned from natural scenes on a 4-dimensional manifold according to the position (x, y), orientation (θ) and scale (σ) of each dictionary element ϕ_i. Any given Gabor function corresponds to a point with coordinates (x, y, θ, σ) on this manifold, and so the learned dictionary as a whole may be conceptualized as a discrete tiling of the manifold. Then, the k-sparse code of an image, α, can be viewed as a set of k delta functions on this manifold (illustrated as black arrows in figure 3.1C). Hyvärinen has pointed out that when the dictionary is topologically organized in a similar manner, the active coefficients α_i tend to form clusters, or "bubbles," over this domain (Hyvärinen, Hurri, *et al.* 2003). Each of these clusters may be thought of as linearly approximating a "virtual Gabor" at the center of the cluster (illustrated as red arrows in figure 3.1C), effectively performing a flexible "steering" of the dictionary to describe discrete structures in the image, similar to steerable filters (Freeman, Adelson, *et al.* 1991; Simoncelli, Freeman, *et al.* 1992; Simoncelli & Freeman 1995; Perona 1995). Assuming there are h such clusters, then the k-sparse code of the image can be thought of as a discrete approximation of an underlying h-sparse function defined on the continuous manifold domain, where h is generally greater than 1 but less than k.

An h-sparse function would not be recoverable from the d-dimensional projection employed in the classical approach because the embedding is premised on there being only a single delta function on the manifold. Hence the inverse will not be uniquely defined. Here we utilize a more general *functional* embedding concept that allows for better recovery capacity. A functional embedding of the landmarks is to take the first f trailing eigenvectors from the generalized eigendecomposition solution as the embedding matrix $P \in \mathbb{R}^{f \times N}$, where f is larger than d such that the h-sparse function can be recovered from the linear projection. Empirically[1] we use $f = O(h \log(N))$.

To illustrate the distinction between the classical view of a data manifold and the additional properties gained by a functional embedding, let us consider a simple example of a function over the 2D unit disc. Assume we are given 300 landmarks on this disc as a dictionary $\Phi_{\text{LM}} \in \mathbb{R}^{2 \times 300}$. We then generate many short sequences of a point x moving along a straight line on the unit disc, with random starting locations and velocities. At each time, t, we use a nearest neighbor (KNN) solver to find a local linear interpolation of the point's location from the landmarks, that is $x_t = \Phi_{\text{LM}} \alpha_t$, with $\alpha_t \in \mathbb{R}^{300}$ and $\alpha_t \geq 0$ (the choice of sparse solver does not impact the demonstration). Now we seek to find an embedding matrix, P, which projects the α_t into an f-dimensional space via $\beta_t = P \alpha_t$ such that the trajectories in β_t are as straight as possible, thus reflecting their true underlying geometry. This is achieved by performing an optimization that minimizes the second temporal derivative of β_t, as specified in equation (3.3) below.

Figure 3.2A shows the rows of P resulting from this optimization using $f = 21$. In-

[1]This choice is inspired by the result from compressive sensing (Donoho 2006), though here h is different from k.

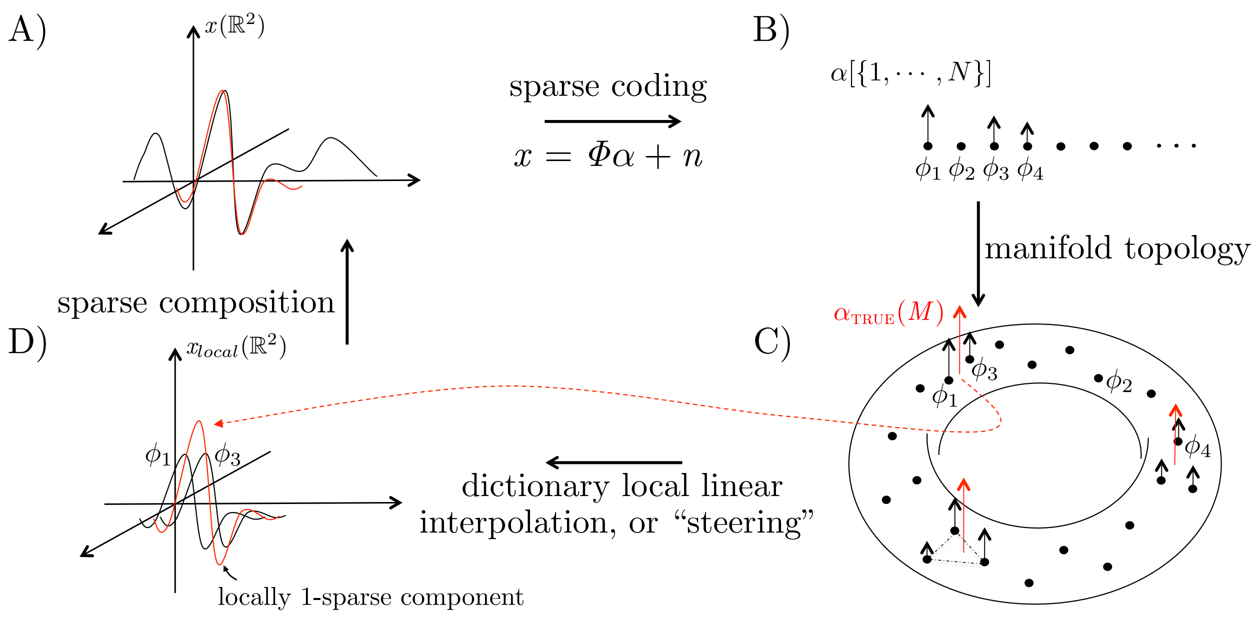
A)
$x(\mathbb{R}^2)$
sparse coding
$x = \Phi\alpha + n$
B)
$\alpha[\{1, \cdots, N\}]$
ϕ_1 ϕ_2 ϕ_3 ϕ_4
manifold topology
sparse composition
D)
$x_{local}(\mathbb{R}^2)$
ϕ_1 ϕ_3
locally 1-sparse component
dictionary local linear interpolation, or "steering"
C)
$\alpha_{\text{TRUE}}(M)$
ϕ_3
ϕ_1
ϕ_2
ϕ_4

Figure 3.1: Dictionary elements learned from natural signals with sparse coding may be conceptualized as landmarks on a smooth manifold. A) A function defined on $\mathbb{R}^2$ (e.g. a gray-scale natural image) and one element from its reconstruction are represented by the black and red curves, respectively. B) The signal is encoded using sparse inference with a learned dictionary, Φ, resulting in a k-sparse vector (also a function) α, which is defined on an orderless discrete set $\{1, \cdots, N\}$. C) α can be viewed as a discrete k-sparse approximation to the true h-sparse function, $\alpha_{\text{TRUE}}(M)$, defined on the smooth manifold ($k = 8$ and $h = 3$ in this example). Each dictionary element in Φ corresponds to a landmark (black dot) on the smooth manifold, M. Red arrows indicate the underlying h-sparse function, while black arrows indicate the k non-zero coefficients of Φ used to interpolate the red arrows. D) Since Φ only contains a finite number of landmarks, we must interpolate (i.e. "steer") among a few dictionary elements to reconstruct each of the true image components.

terestingly, they resemble Zernike polynomials on the unit-disc. We can think of these as functionals that "sense" sparse functions on the underlying manifold. Each row $p_i' \in \mathbb{R}^{300}$ (here the prime sign denotes a row of the matrix P) projects a discrete k-sparse approximation α of the underlying h-sparse function to a real number, β_i. We define the full set of these linear projections $\beta = P\alpha$ as a "manifold sensing" of α.

When there is only a single delta-function on the manifold, the second and third rows of P, which form simple linear ramp functions in two orthogonal directions, are sufficient to fully represent its position. These two rows would constitute $P_{\text{C}} \in \mathbb{R}^{2\times300}$ as an embedding solution in the classical manifold learning approach, since a unit disk is diffeomorphic to $\mathbb{R}^2$ and can be embedded in a 2 dimensional space. The resulting embedding β_2, β_3 closely resembles the 2-D unit disk manifold and allows for recovery of a one-sparse function, as shown in Figure 3.2B.

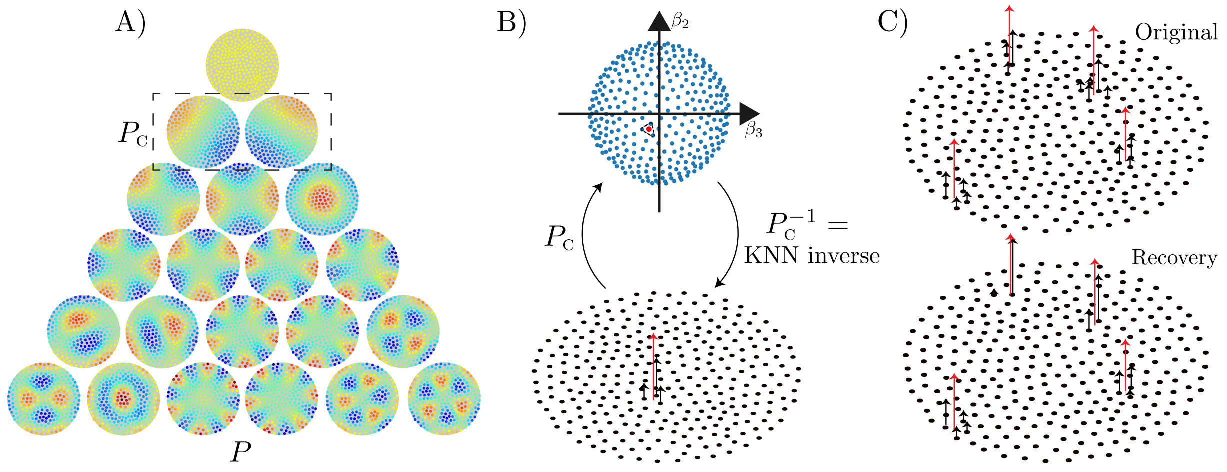

Figure 3.2: Demonstration of functional embedding on the unit disc. A) The rows of P, visualized here on the ground-truth unit disc. Each disc shows the weights in a row of P by coloring the landmarks according to the corresponding value in that row of P. The color scale for each row is individually normalized to emphasize its structure. The pyramidal arrangement of rows is chosen to highlight their strong resemblance to the Zernike polynomials. B) (Top) The classic manifold embedding perspective allows for low-dimensional data visualization using P_C, which in this case is given by the second and third rows of P (shown in dashed box in panel A). Each blue dot shows the 2D projection of a landmark using P_C. Boundary effects cause the landmarks to cluster toward the perimeter. (Bottom) A 1-sparse function is recoverable when projected to the embedding space by P_C. C) (Top) A 4-sparse function (red arrows) and its discrete k-sparse approximation, α (black arrows) on the unit disc. (Bottom) The recovery, α_{REC}, (black arrows) is computed by solving the optimization problem in equation (3.1). The estimate of the underlying function (red arrows) was computed by taking a normalized local mean of the recovered k-sparse approximations for a visualization purpose.

Recovering more than a one-sparse function requires using additional rows of P with higher spatial-frequencies on the manifold, which together provide higher sensing capacity. Figure 3.2C demonstrates recovery of an underlying 4-sparse function on the manifold using all 21 functionals, from p'_1 to p'_{21}. From this representation, we can recover an estimate of α with positive-only sparse inference:

$$\alpha_{\text{REC}} = g(\beta) \equiv \underset{\alpha}{\operatorname{argmin}} \|\beta - P\alpha\|_F^2 + \lambda z^T \alpha, \text{ s.t. } \alpha \geq 0, \tag{3.1}$$

where $z = [\|p_1\|_2, \cdots, \|p_N\|_2]^T$ and $p_j \in \mathbb{R}^{21}$ is the j^{th} column of P. Note that although α_{REC} is not an exact recovery of α, the 4-sparse structure is still well preserved, up to a local shift in the locations of the delta functions. We conjecture this will lead to a recovery that is perceptually similar for an image signal.

The functional embedding concept can be generalized beyond functionals defined on a single manifold and will still apply when the underlying geometrical domain is a union of several different manifolds. A thorough analysis of the capacity of this sensing method is beyond the scope of this paper, although we recognize it as an interesting research topic for model-based compressive sensing.

3.2 The Formulation of the SMT

The Sparse Manifold Transform (SMT) consists of a non-linear sparse coding expansion followed by a linear manifold sensing compression (dimension reduction). The manifold sensing step acts to linearly pool the sparse codes, α, with a matrix, P, that is learned using the functional embedding concept (sec. 3.1) in order to straighten trajectories arising from video (or other dynamical) data as shown in Figure 3.3.

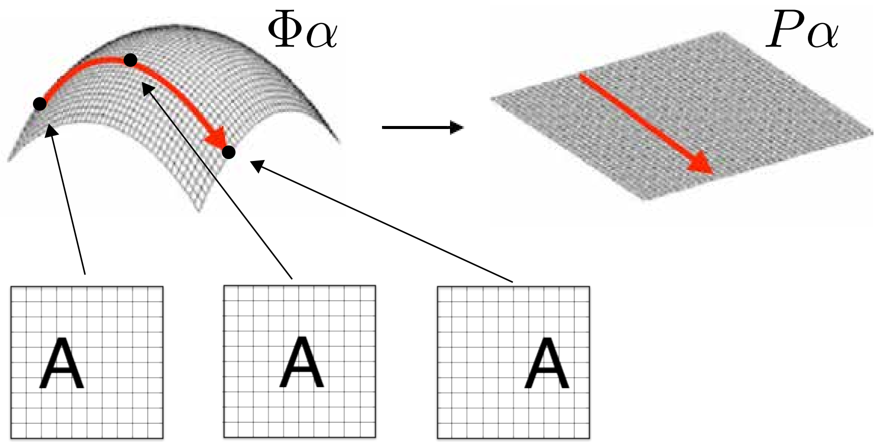

Figure 3.3: While a natural transformation trajectory in the signal space is highly curved, we would like it to be straightened in the embedding space.

The SMT framework makes three basic assumptions:

1. The dictionary Φ learned by sparse coding has an organization that is a discrete sampling of a low-dimensional, smooth manifold, M (Fig. 3.1).

2. The resulting sparse code α is a discrete k-sparse approximation of an underlying h-sparse function defined on M. There exists a functional manifold embedding, $\tau : \Phi \hookrightarrow P$,

that maps each of the dictionary elements to a new vector, $p_j = \tau(\phi_j)$, where p_j is the j^{th} column of P s.t. both the topology of M and h-sparse function's structure are preserved.

3. A continuous temporal transformation in the input (e.g., from natural movies) lead to a linear flow on M and also in the geometrical embedding space.

In an image, the elements of the underlying h-sparse function correspond to discrete components such as edges, corners, blobs or other features that are undergoing some simple set of transformations. Since there are only a finite number of learned dictionary elements tiling the underlying manifold, they must cooperate (or 'steer') to represent each of these components as they appear along a continuum.

The desired property of linear flow in the geometric embedding space may be stated mathematically as

$$P\alpha_t \approx \tfrac{1}{2}P\alpha_{t-1} + \tfrac{1}{2}P\alpha_{t+1}. \tag{3.2}$$

where α_t denotes the sparse coefficient vector at time t. Here we exploit the temporal continuity inherent in the data to solve the otherwise cumbersome nearest-neighbor search required of LLE or LLL. The embedding matrix P satisfying (3.2) may be derived by minimizing an objective function that encourages the second-order temporal derivative of $P\,\alpha$ to be zero:

$$\min_P \|PAD\|_F^2, \text{ s.t. } PVP^T = I \tag{3.3}$$

where A is the coefficient matrix whose columns are the coefficient vectors, α_t, in temporal order, and D is the second-order differential operator matrix, with $D_{t-1,t} = -0.5, D_{t,t} = 1, D_{t+1,t} = -0.5$ and $D_{\tau,t} = 0$ otherwise. V is a positive-definite matrix for normalization, I is the identity matrix and $\| \bullet \|_F$ indicates the matrix Frobenius norm. We choose V to be the covariance matrix of α and thus the optimization constraint makes the rows of P orthogonal in whitened sparse coefficient vector space. Note that this formulation is qualitatively similar to applying SFA to sparse coefficients, but using the second-order derivative instead of the first-order derivative.

The solution to this generalized eigen-decomposition problem is given (Vladymyrov & Carreira-Perpinán 2013) by $P = V^{-\frac{1}{2}}U$, where U is a matrix of f trailing eigenvectors (i.e. eigenvectors with the smallest eigenvalues) of the matrix $V^{-\frac{1}{2}}ADD^TA^TV^{-\frac{1}{2}}$. Some drawbacks of this analytic solution are that: 1) there is an unnecessary ordering among different dimensions, 2) the learned functional embedding tends to be global, which has support as large as the whole manifold and 3) the solution is not online and does not allow other constraints to be posed. In order to solve these issues, we modify the formulation slightly with a sparse regularization term on P and develop an online SGD (Stochastic Gradient Descent) solution, which is detailed in the Supplement A.1.

To summarize, the SMT is performed on an input signal x by first computing a higher-dimensional representation α via sparse inference with a learned dictionary, Φ, and second computing a contracted code by sensing a manifold representation, $\beta = P\,\alpha$ with a learned pooling matrix, P.

3.3 Results of the SMT Trained with Natural Videos

Straightening of video sequences. We applied the SMT optimization procedure on sequences of whitened 20×20 pixel image patches extracted from natural videos. We first learned a $10\times$ overcomplete spatial dictionary $\Phi \in \mathbb{R}^{400\times4000}$ and coded each frame x_t as a 4000-dimensional sparse coefficient vector α_t. We then derived an embedding matrix $P \in \mathbb{R}^{200\times4000}$ by solving equation 3.3. Figure 3.4 shows that while the sparse code α_t exhibits high variability from frame to frame, the embedded representation $\beta_t = P\,\alpha_t$ changes in a more linear or smooth manner. It should be emphasized that finding such a smooth linear projection (embedding) is highly non-trivial, and is possible if and only if the sparse codes change in a locally linear manner in response to smooth transformations in the image. If the sparse code were to change in an erratic or random manner under these transformations, any linear projection would be non-smooth in time. Furthermore, we show that this embedding does not constitute a trivial temporal smoothing, as we can recover a good approximation of the image sequence via $\hat{x}_t = \Phi\, g(\beta_t)$, where $g(\beta)$ is the inverse embedding function (3.1). We can also use the functional embedding to regularize sparse inference, as detailed in Section 3.4, which further increases the smoothness of both α and β.

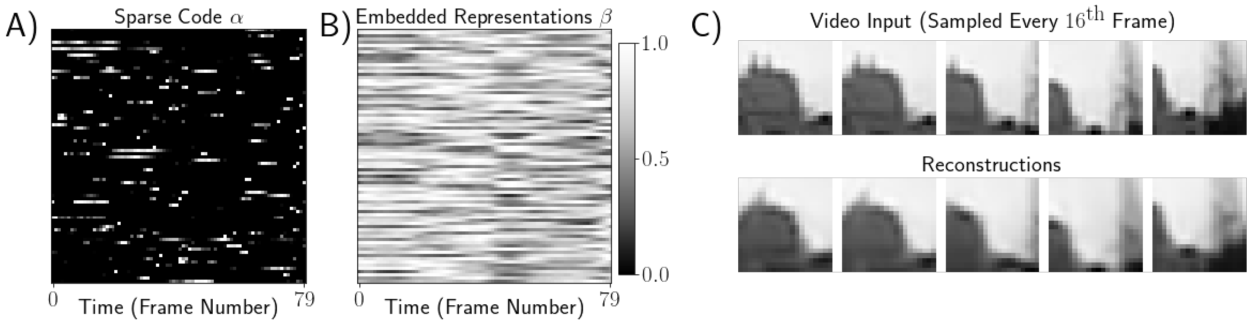

Figure 3.4: SMT encoding of a 80 frame image sequence. A) Rescaled activations for 80 randomly selected α units. Each row depicts the temporal sequence of a different unit. B) The activity of 80 randomly selected β units. C) Frame samples from the 90fps video input (top) and reconstructions computed from the α_{REC} recovered from the sequence of β values (bottom).

Affinity Groups and Dictionary Topology. Once a functional embedding is learned for the dictionary elements, we can compute the cosine similarity between their embedding vectors, $\cos(p_j, p_k) = \frac{p_j^T p_k}{\|p_j\|_2 \|p_k\|_2}$, to find the neighbors, or affinity group, of each dictionary element in the embedding space. In Figure 3.6A we show the affinity groups for a set of randomly sampled elements from the overcomplete dictionary learned from natural videos. This figure utilized a visualization technique that depicts each dictionary element as a needle whose position, orientation and length indicate those properties of the element. The

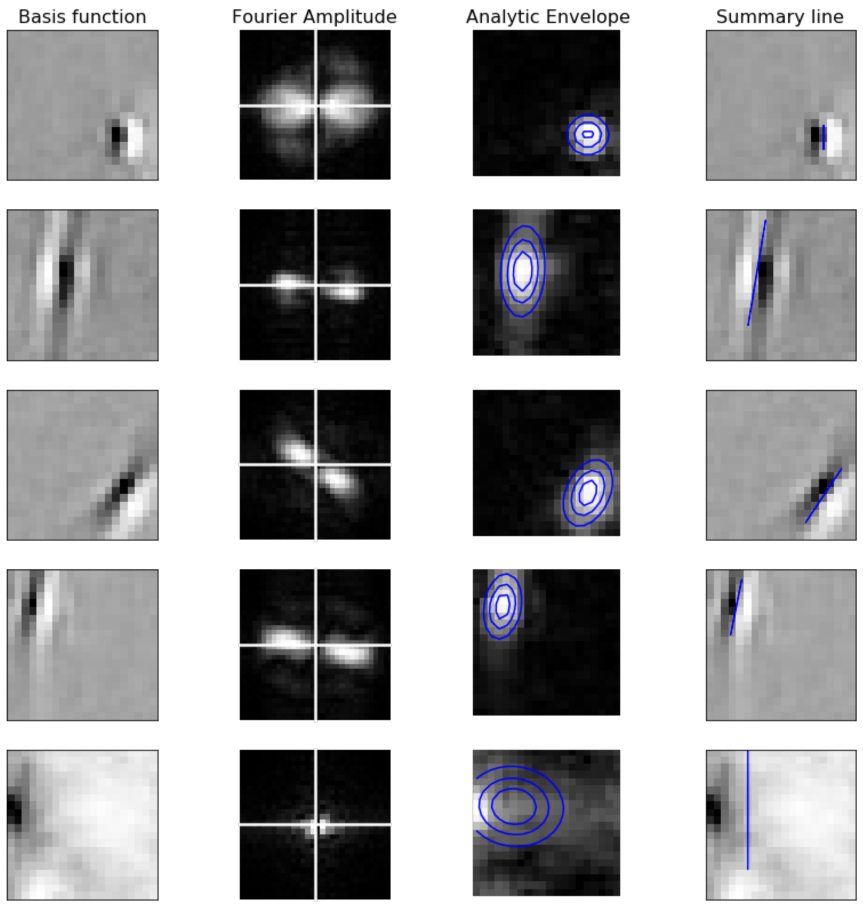

Figure 3.5: Needle plot fitting procedure

parameters of the needles were computed from a combination of the analytic signal envelope and the Fourier transform of each dictionary element. The envelope was fitted with a 2D Gaussian, and its mean was used to determine the center location of the dictionary element. The primary eigenvector of the Gaussian covariance matrix was used to determine the spatial extent of the primary axis of variation, which we indicated by the bar length. The orientation was computed from the peak in the Fourier amplitude map of the dictionary element. Figure 3.5 gives an illustration of the fitting process. A subset of elements were not well summarized by this methodology because they do not fit a standard Gaussian profile, which can be seen in the last row of Figure 3.5. However, the majority appear to be well characterized by this method. Each box in the main-text figure 3.6 indicates a single affinity group. The color indicates the normalized similarity between the dictionary elements in the embedding space.

As one can see in Figure 3.6A, the topology of the embedding learned from the SMT reflects the structural similarity of the dictionary elements according to the properties of position, orientation, and scale. Figure 3.6B shows that the nearest neighbors of each dictionary element in the embedding space are more 'semantically similar' than the nearest neighbors of the element in the pixel space. To measure the similarity, we chose the top 500 most well-fit dictionary elements and computed their lengths and orientations. For each of these elements, we find the top 9 nearest neighbors in both the embedding space and in pixel space and then compute the average difference in length (Δ Length) and orientation (Δ Angle). The results confirm that the embedding space is succeeding in grouping dictionary elements according to their structural similarity, presumably due to the continuous geometric transformations occurring in image sequences.

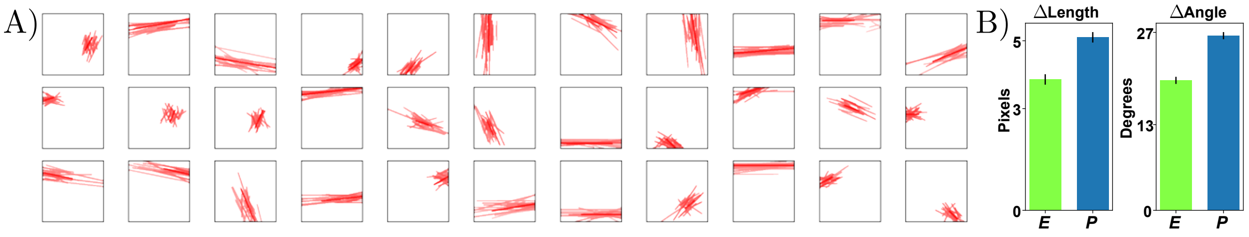

Figure 3.6: A) Affinity groups learned using the SMT reveal the topological ordering of a sparse coding dictionary. Each box depicts as a needle plot the affinity group of a randomly selected dictionary element and its top 40 affinity neighbors. The length, position, and orientation of each needle reflect those properties of the dictionary element in the affinity group. The color shade indicates the normalized strength of the cosine similarity between the dictionary elements. B) The properties of length and orientation (angle) are more similar among nearest neighbors in the embedding space (E) as compared to the pixel space (P).

Computing the cosine similarity can be thought of as a hypersphere normalization on the embedding matrix P. In other words, if the embedding is normalized to be approximately on a hypersphere, the cosine distance is almost equivalent to the Gramian matrix, $P^T P$. Taking this perspective, the learned geometric embedding and affinity groups can explain the dictionary grouping results shown in previous work (Hosoya & Hyvärinen 2016). In that work, the layer 1 outputs are pooled by an affinity matrix given by $P = E^T E$, where E is the eigenvector matrix computed from the correlations among layer 1 outputs. This PCA-based method can be considered an embedding that uses only spatial correlation information, while the SMT model uses both spatial correlation and temporal interpolation information.

Hierarchical Composition. A SMT layer is composed of two sublayers: a sparse coding sublayer that models sparse discreteness, and a manifold embedding sublayer that models simple geometrical transforms. It is possible to stack multiple SMT layers to form a hierarchical architecture, which addresses the third pattern from Mumford's theory: hierarchical composition. It also provides a way to progressively flatten image manifolds, as proposed by

DiCarlo & Cox (DiCarlo & Cox 2007). Here we demonstrate this process with a two-layer SMT model (Figure 3.7A) and we visualize the learned representations. The network is trained in a layer-by-layer fashion on a natural video dataset as above.

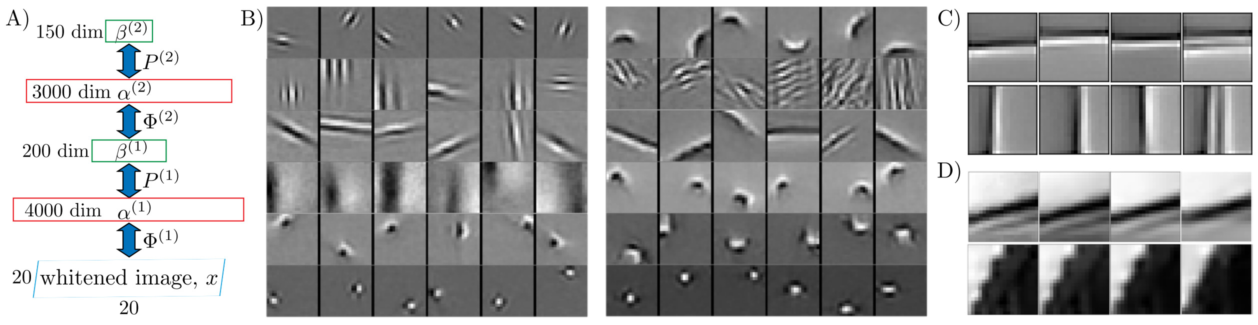

Figure 3.7: SMT layers can be stacked to learn a hierarchical representation. A) The network architecture. Each layer contains a sparse coding sublayer (red) and a manifold sensing sublayer (green). B) Example dictionary element groups for $\Phi^{(1)}$ (left) and $\Phi^{(2)}$ (right). C) Each row shows an example of interpolation by combining layer 3 dictionary elements. From left to right, the first two columns are visualizations of two different layer-3 dictionary elements, each obtained by setting a single element of $\alpha^{(3)}$ to one and the rest to zero. The third column is an image generated by setting both elements of $\alpha^{(3)}$ to 0.5 simultaneously. The fourth column is a linear interpolation in image space between the first two images, for comparison. D) Information is approximately preserved at higher layers. From left to right: The input image and the reconstructions from $\alpha^{(1)}$, $\alpha^{(2)}$ and $\alpha^{(3)}$, respectively. The rows in C) and D) are unique examples. See section 3.1 for visualization details.

We can produce reconstructions and dictionary visualizations from any layer by repeatedly using the inverse operator, $g(\beta)$. Formally, we define $\alpha_{\text{REC}}^{(l)} = g^{(l)}(\beta^{(l)})$, where l is the layer number. For example, the inverse transform from $\alpha^{(2)}$ to the image space will be $x_{\text{REC}} = C\Phi^{(1)}g^{(1)}(\Phi^{(2)}\alpha^{(2)})$, where C is an unwhitening matrix. We can use this inverse transform to visualize any single dictionary element by setting $\alpha^{(l)}$ to a 1-hot vector. Using this method of visualization, Figure 3.7B shows a comparison of some of the dictionary elements learned at layers 1 and 2. We can see that lower layer elements combine together to form more global and abstract dictionary elements in higher layers, e.g. layer-2 units tend to be more curved, many of them are corners, textures or larger blobs.

Another important property that emerges at higher levels of the network is that dictionary elements are steerable over a larger range, since they are learned from progressively more linearized representations. To demonstrate this, we trained a three-layer network and performed linear interpolation between two third-layer dictionary elements, resulting in a non-linear interpolation in the image space that shifts features far beyond what simple linear interpolation in the image space would accomplish (Figure 3.7C). A thorough visualization of the dictionary elements and groups is provided below.

Dictionary Visualization Here we provide a more complete visualization of the dictionary elements in the two-layer SMT model. In Figure 3.8, we show 19 representative dictionary elements and their top 5 affinity group neighbors in both layer 1 (Figure 3.8A) and layer 2 (Figure 3.8B). For each row, we choose a representative dictionary element (on the left side) and compute its cosine similarity to the rest of the dictionary elements in the embedding space. The closest 5 elements in the embedding space are shown on its right. So each row can be treated as a small affinity group. Figures 3.9 and 3.10 show a large random sample of the dictionary elements in layers 1 and 2, respectively.

3.4 Sparse Coefficient Inference with Manifold Embedding Regularization

Functional manifold embedding can be used to regularize sparse inference for video sequences. We assume that in the embedding space the representation has a locally linear trajectory. Then we can change the formulation from equation 2.4 to the following:

$$\begin{aligned} \min_{\alpha_t} \quad & \|x_t - \Phi\alpha_t\|_2^2 + \lambda\|\alpha_t\|_1 + \gamma_0\|P(\alpha_t - \widetilde{\alpha_t})\|_2^2 \\ \text{s.t.} \quad & \alpha_t \geq 0,\ \widetilde{\alpha_t} = \alpha_{t-1} + (\alpha_{t-1} - \alpha_{t-2}), \end{aligned} \tag{3.4}$$

where $\widetilde{\alpha_t}$ is the casual linear prediction from the previous two steps α_{t-1}, α_{t-2}. Our preliminary experiments indicate that this can significantly increase the linearity of the embedding $\beta = P\alpha$.

3.5 Dictionary Embedding Without Time

As we mentioned earlier in Section 2.4, neither sparse coding nor manifold learning requires temporal sequences. Using temporal to guide the embedding is rather a natural choice, which can potentially increase the learning efficiency. So the sparse manifold transform, as a unification of sparse coding and manifold learning, does not require temporal sequences per se. We will revisit the formulation of the SMT in this chapter to make this point more clear since Equation (3.2) and Optimization (3.3) may deceptively imply the necessity of temporal information.

In both Section 2.4 and Section 3.2, we have discussed that the formulation used in the SMT utilizes the temporal information because the temporal neighbors provides a more efficient solution than the general nearest neighbor search used in the LLE and other manifold learning algorithm, specifically eq. (2.9) is used as a surrogate to eq. (2.6) to capture the local geometry among data points once temporal information is available. With the i.i.d. data, we can relax the third assumption in Section 3.2 to the following:

3*. The local geometry in the input space is preserved in the geometric embedding space. Here we can choose to preserve any geometric property used by a manifold learning technique, e.g. nearest neighbor interpolation relationship in LLE, geodesic distance used by ISOMAP, the weighted distance used in Laplacian eigenmaps etc. Let's first take the local linear interpolation as our example.

In some sense, this is redundant given the second assumption. With this more general formulation, eq. (3.2) would be replaced by the following equation:

$$P\alpha_i \approx \sum_{j \in NN(i)} \gamma_j P\alpha_j. \tag{3.5}$$

where $NN(i)$ is the set of top K nearest neighbor of i^{th} data point and $\{\gamma_j | j \in NN(i)\}$ are corresponding the local nearest neighbor interpolation coefficients solved in eq. 2.6.

Overall, this relaxation to non-temporal data still leads to the same optimization:

$$\min_P \|PAD^*\|_F^2, \text{ s.t. } PVP^T = I \tag{3.6}$$

the only difference is that D^* is constructed from data local linear interpolation matrix Γ that $D^* = I - \Gamma$. $D*$ is similar to the one used in LLL (Vladymyrov & Carreira-Perpinán 2013) rather D in Optimization (3.3), which is the second-order temporal differential operator matrix. If we view $D*$ as a general Laplacian operator, D can be viewed as the second-order line derivative operator.

Let's look at another example where we choose to build an embedding of our dictionary with the property used in Laplacian eigenmaps (Belkin & Niyogi 2002). We further assume we already build the weight W matrix between the data points. Then the optimization to be solved is the following:

$$\min_P \operatorname{Tr} PADA^T P^T, \text{s.t. } PA\Lambda A^T P^T = I \tag{3.7}$$

where $D = \Lambda - W$, W_{ij} is the weight between the i^{th} data point and the j^{th} data point and Λ is the degree matrix of the data points[2].

We would like to emphasis again that in the SMT framework, it's a user's freedom to choose which geometric property to preserved. This property could be sequence straightening (Optimization (3.3)), local linear interpolation (Optimization (3.6)), to bring neighbors in data space close in the embedding space (Optimization (3.7)) or the data points in the same category (if label is available) close in the embedding space.

3.6 Hierarchical Representation of Digits

In this section, we apply the general SMT formulation to affNIST dataset and build a hierarchical representation of hand-written digits. We do not perform whitening in the first

[2]If Λ is the identity matrix, then $PA\Lambda A^T P^T = PVP^T$.

layer and the dictionary elements are required to be positive-only. This choice also provides a better visualization since the digits images are strictly non-negative and negative values makes a dictionary element look quite different from the input signal.

We adopt Optimization (3.6) to learn a two-layer SMT representation, where each dictionary layer has 4096 dictionary elements. The layer visualization is performed as shown in Section 3.2. In Figure 3.11, we show the learning result: while the first layer have learned the basic strokes, the second layer has learned much more global structures. Further, we can visualize the affinity groups (introduced in Section 3.3), which is shown in Figure 3.12. As we can see that the dictionary elements in the affinity groups are all semantically similar and the embedding space successfully captures the geometry of the dictionary.

3.7 Discussion

A key new perspective introduced in this work is to view both the signals (such as images) and their sparse representations as functions defined on a manifold domain. A gray-scale image is a function defined on a 2D plane, tiled by pixels. Here we propose that the dictionary elements should be viewed as the new 'pixels' and their coefficients are the corresponding new 'pixel values'. The pooling functions can be viewed as low pass filters defined on this new manifold domain. This perspective is strongly connected to the recent development in both signal processing on irregular domains (Shuman *et al.* 2013) and geometric deep learning (Bronstein *et al.* 2017).

Previous approaches have learned the group structure of dictionary elements mainly from a statistical perspective (Hyvärinen & P. Hoyer 2000; Hyvärinen, P. O. Hoyer, *et al.* 2001; Osindero *et al.* 2006; Köster & Hyvärinen 2010; Lyu & Simoncelli 2008; Malo & Gutiérrez 2006). Additional unsupervised learning models (Shan & Cottrell 2013; Paiton *et al.* 2016; Le *et al.* 2012; Zeiler, Taylor, *et al.* 2011) combine sparse discreteness with hierarchical structure, but do not explicitly model the low-dimensional manifold structure of inputs. Our contribution here is to approach the problem from a geometric perspective to learn a topological embedding of the dictionary elements.

The functional embedding framework provides a new perspective on the pooling functions commonly used in convnets. In particular, it provides a principled framework for learning the pooling operators at each stage of representation based on the underlying geometry of the data, rather than being imposed in a 2D topology *a priori* as was done previously to learn linearized representations from video (Goroshin, Mathieu, *et al.* 2015). This could facilitate the learning of higher-order invariances, as well as equivariant representations (Sabour *et al.* 2017), at higher stages. In addition, since the pooling is approximately invertible due to the underlying sparsity, it is possible to have bidirectional flow of information between stages of representation to allow for hierarchical inference (T. S. Lee & Mumford 2003). The invertibility of SMT is due to the underlying sparsity of the signal, and is related to prior works on the invertibility of deep networks (Gilbert *et al.* 2017; Bruna *et al.* 2013; Zeiler &

Fergus 2014; Dosovitskiy & Brox 2016). Understanding this relationship may bring further insights to these models.

Finally, though the SMT framework does not require convolution, incorporating convolution into the SMT may help to scale up the modeling to larger image modeling. We also leave this as a future direction.

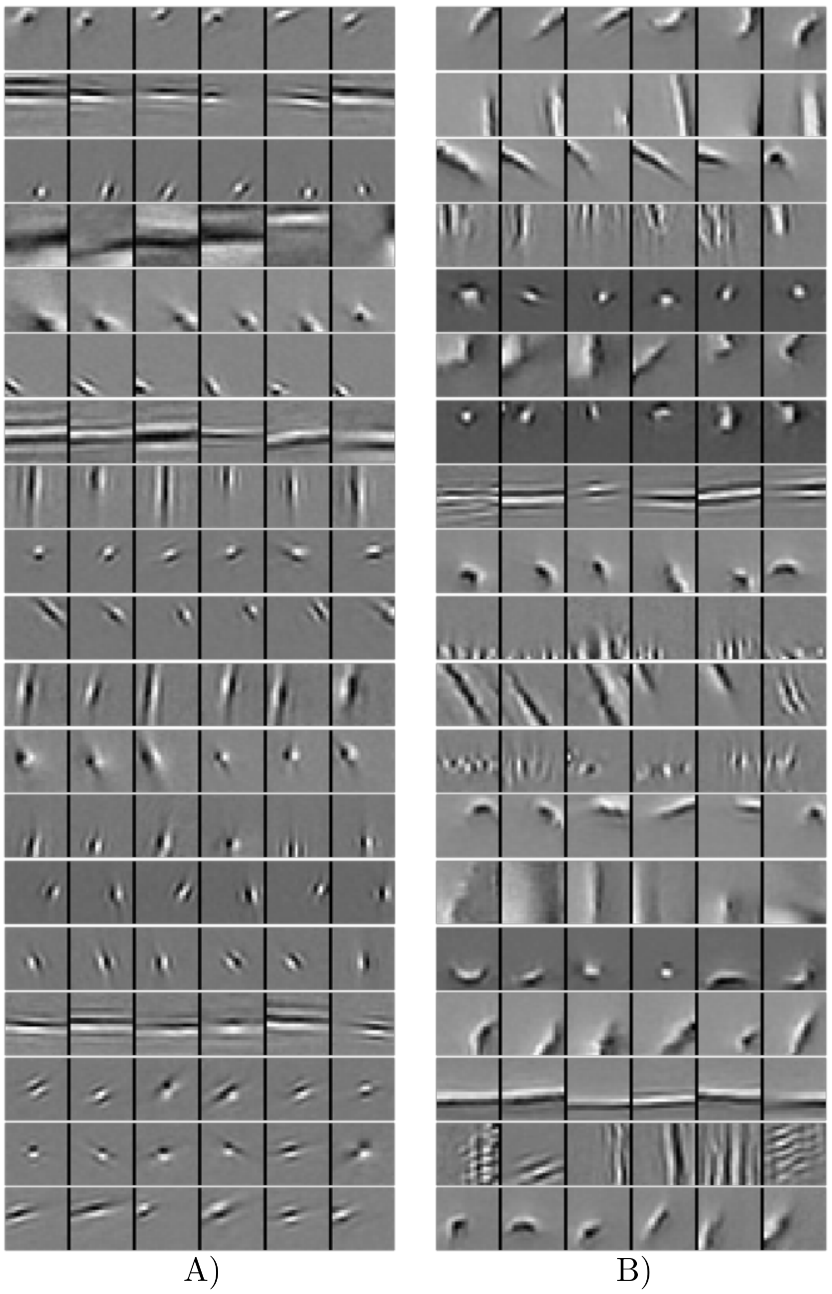

Figure 3.8: Representative dictionary elements and their top 5 affinity group neighbors in the embedding space. A) Dictionary elements from layer 1. B) Dictionary elements from layer 2. We encourage the reader to compare the affinity neighbors to the results found in (Zeiler & Fergus 2014).

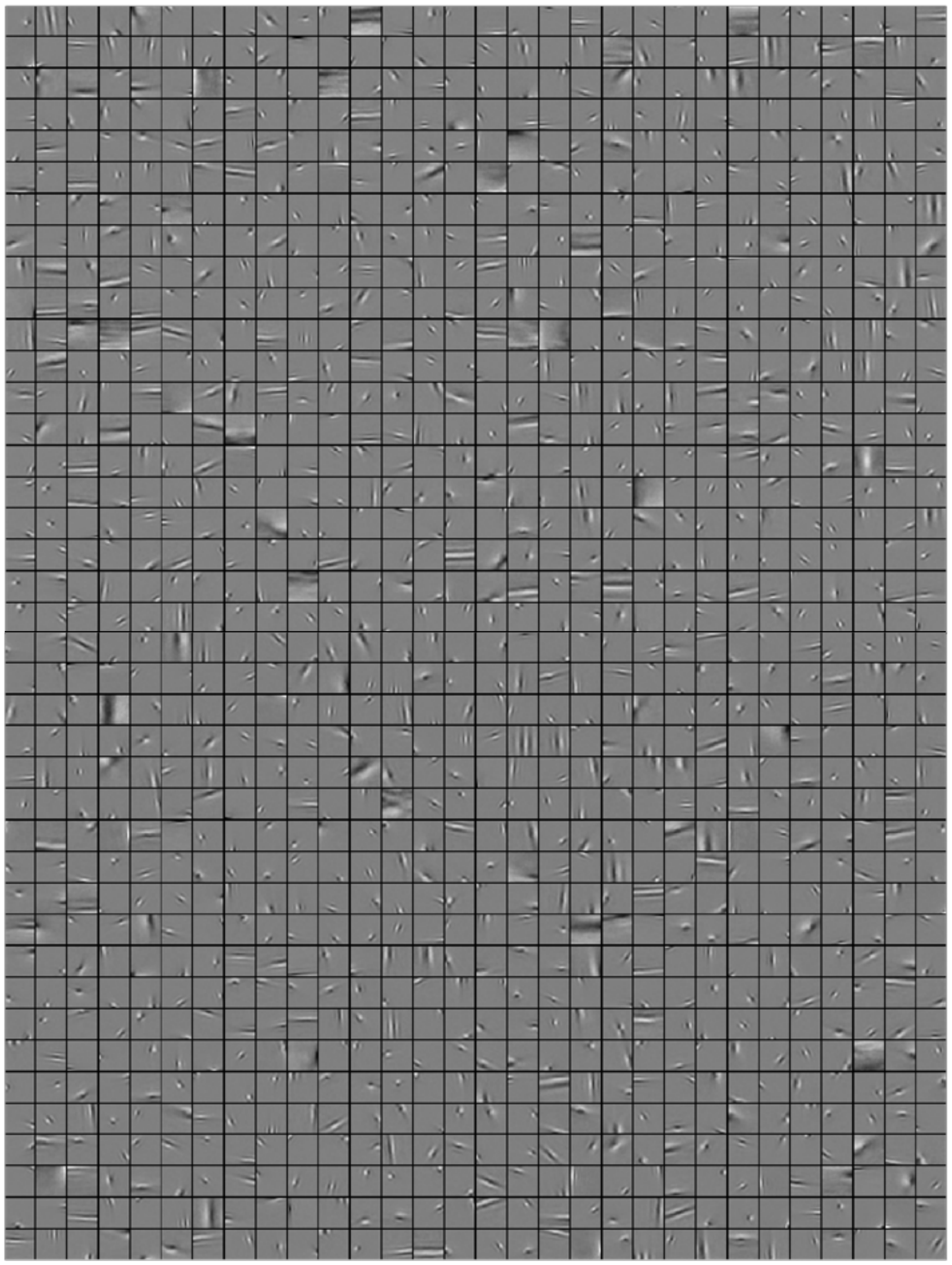

Figure 3.9: A random sample of 1200 dictionary elements from layer 1 (out of 4000 units). Details are better discerned by zoomed in on a monitor.

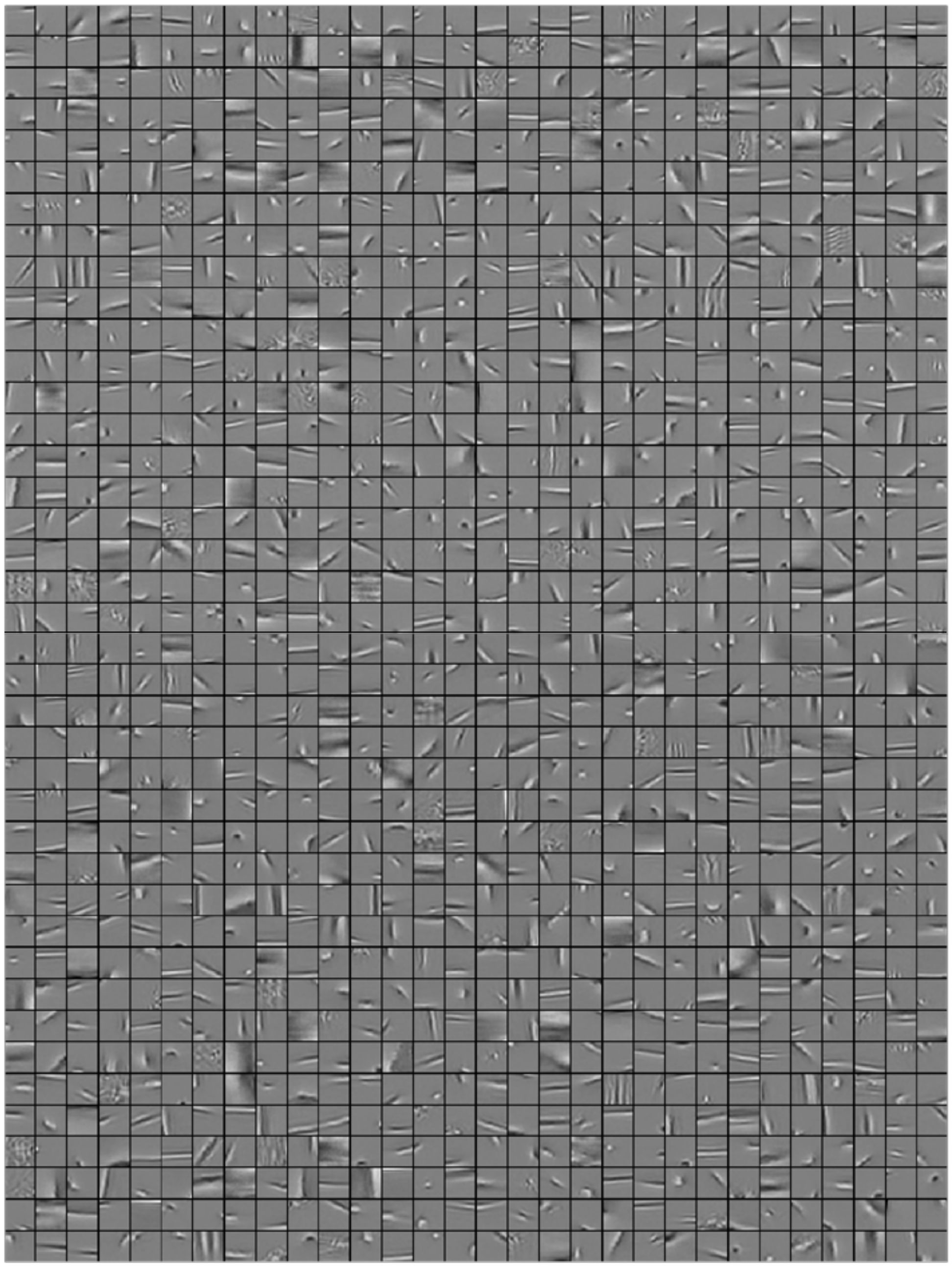

Figure 3.10: A random sample of 1200 dictionary elements from layer 2 (out of 3000 units). Details are better discerned by zoomed in on a monitor.

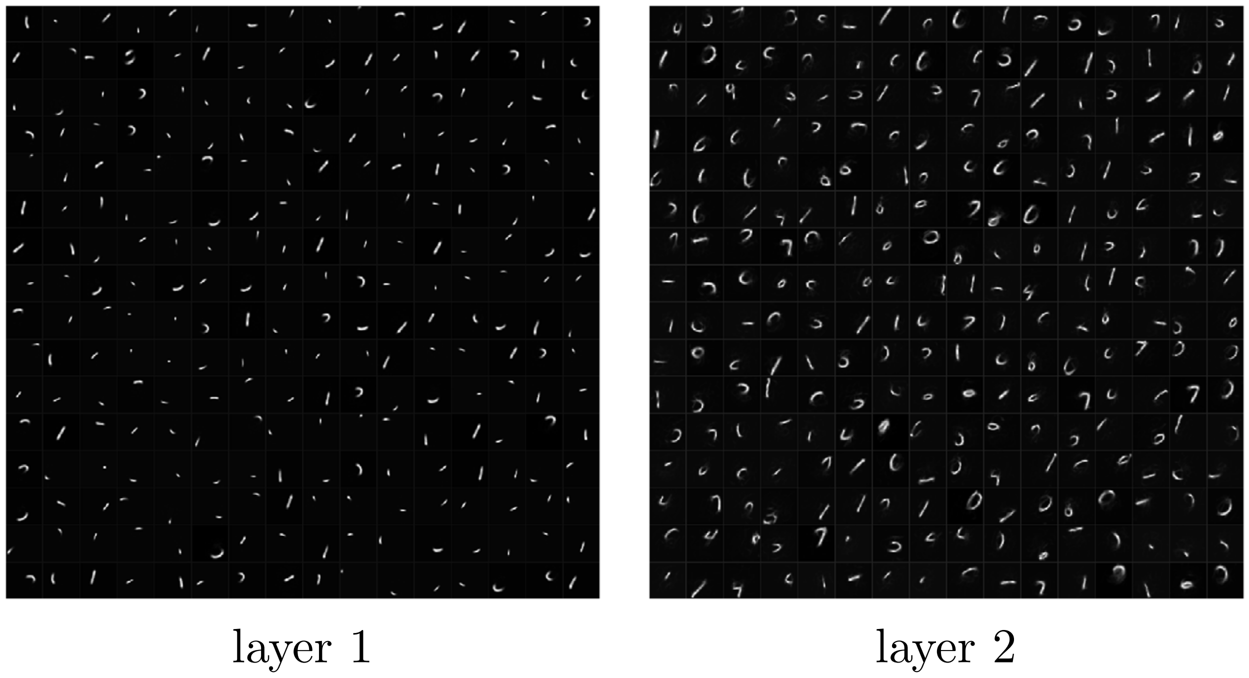

Figure 3.11: When we apply SMT to the hand-written digits, the high layers learn much more global structures than the lower layers. So we can stack SMT and build a hierarchical representation of the input dataset.

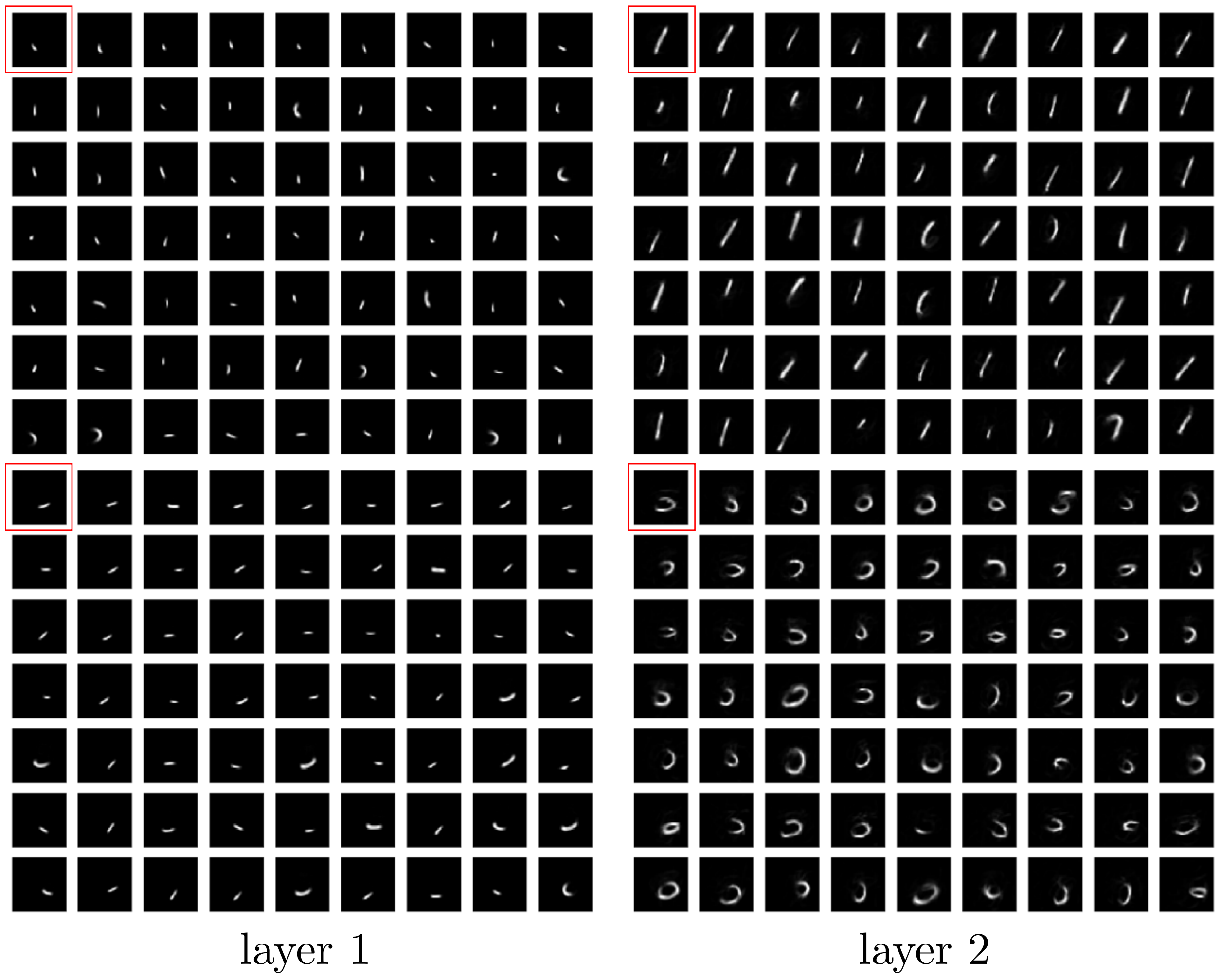

Figure 3.12: The left panel shows two affinity groups from layer 1 and the right panel shows two from layer 2. Each of the affinity groups is the embedding geometric nearest neighbors (by cosine similarity) of the corresponding dictionary element in the red box.

Chapter 4

A Geometric Theory for Complex Cells

4.1 Introduction

A fundamental role of the visual cortex is to represent structure in natural scenes, including local features and smooth transformations of these features. It is widely believed that *simple cells* learn and recognize these local features, which can be modeled using unsupervised dictionary learning (Olshausen & Field 1996). Previous approaches for understanding complex cells fall into two major classes: 1) those motivated by invariant image processing (pooling simple cell features) (Hubel & Wiesel 1962; Riesenhuber & Poggio 1999) and 2) those motivated by modeling the multivariate higher-order statistics of neural responses (Hyvärinen, Hurri, *et al.* 2003; Hyvärinen & Köster 2007; Karklin & Lewicki 2009; Hosoya & Hyvärinen 2016).

Here we provide a unifying geometric perspective on the role of simple and complex cells based on the a version of the SMT in which neurons in the two layers correspond to simple and complex cells. This provides a new functional explanation for these cell types: Simple cells can be viewed as representing a discrete sampling of a smooth manifold in the sensor space, while complex cells can be viewed as representing localized smooth linear functions on the manifold. While each individual complex cell pools from a local region, together they tile (Sengupta *et al.* 2018) the manifold and build an untangled population representation (DiCarlo & Cox 2007), which tends to preserve the identity of the signal while straightening the transformations (Henaff *et al.* 2017). In the localized SMT, the complex cell layer is learned in an unsupervised manner based on a diffusion process. Our results demonstrate that simple and complex cells are emergent properties in a neural system that is optimized to learn the manifold structure of dynamic sensory inputs and is subject to sparse connectivity constraints.

4.2 Methods

We use highly overcomplete positive-only sparse coding to model simple cells as described in Section 2.2. For each input image patch x, we use FISTA (Beck & Teboulle 2009) to solve the positive only sparse coefficient inference:

$$\min_{\alpha} \tfrac{1}{2}\|x - \Phi\alpha\|_2^2 + \lambda\|\alpha\|_1, \text{ s.t. } \alpha \geq 0, \tag{4.1}$$

where $\Phi \in \mathbb{R}^{n\times m}$ is the learned dictionary matrix with dictionary elements ϕ_i in each of its columns, and $\alpha \in \mathbb{R}^m$ is a positive-only vector of coefficients.

The responses of simple cells are linearly pooled by the complex cells such that $\beta = P\alpha$, where each of the rows in P is a learned pooling unit. The pooling units can be learned by solving a variation of the following optimization problem:

$$\begin{aligned} \min_{P} \quad & \langle \|PV^{-\frac{1}{2}}\ddot{\alpha}\|_2^2 \rangle + \gamma\|P\|_1 \\ \text{s.t.} \quad & \|p'_j\|_2 = c, \ \|p_i\|_2 = 1, \ P \geq 0 \end{aligned} \tag{4.2}$$

where $V^{-\frac{1}{2}}$ is the whitening matrix for α, where $V = (\alpha - \langle\alpha\rangle)(\alpha - \langle\alpha\rangle)^T$, p'_j is a row of P and p_i is a column. The constraints are for row and column normalization. The angular bracket $\langle\cdot\rangle$ stands for temporal average and $\ddot{\alpha}$ is the second-order temporal derivative of α. It is helpful to reexpress the first term in the objective above as:

$$\langle \|PV^{-\frac{1}{2}}\ddot{\alpha}\|_2^2 \rangle = \operatorname{Tr} PUP^T, U = V^{-\frac{1}{2}}\langle\ddot{\alpha}\ddot{\alpha}^T\rangle V^{-\frac{1}{2}} \tag{4.3}$$

which has a gradient given by PU. Thus to solve (4.2), we can use an iterative proximal gradient method: 1) gradient descent via $P = (1-\eta)P + \eta P(I - U)$, where η is the step size, 2) sparse shrinkage and rectification $P = relu(P - \eta\gamma)$, 3) row and column normalization. Note that Step 1) is very similar to a diffusion process, where the diffusion kernel is $I - U$. To better condition the optimization, we replace $I - U$ by a sparse diffusion kernel K^*, where K^* only preserves the largest 10 absolute values in each column of $I - U$ and each column is normalized to unit L1 norm. Note that if the temporal sequence is randomly shuffled, both $I - U$ and K^* are the identity matrix. Thus K^* only contains how the activations of α flow to the geometric neighbors under transformations.

The optimization (4.2) can be viewed as a variation of the Optimization (3.3) , which leads to spectral and thus *global* pooling functions rather than localized pooling functions. However, both solutions lead to smooth pooling functions defined on an underlying simple cell manifold, which linearize the response from the simple cells after pooling.

4.3 Experiments

Our dataset contains 53 minutes of 128x128 90fps whitened natural video taken on the UC Berkeley campus. We extract 20x20 pixel random image patches and train a 10x overcomplete sparse coding model. Once the dictionary converges, we compute the covariance matrix

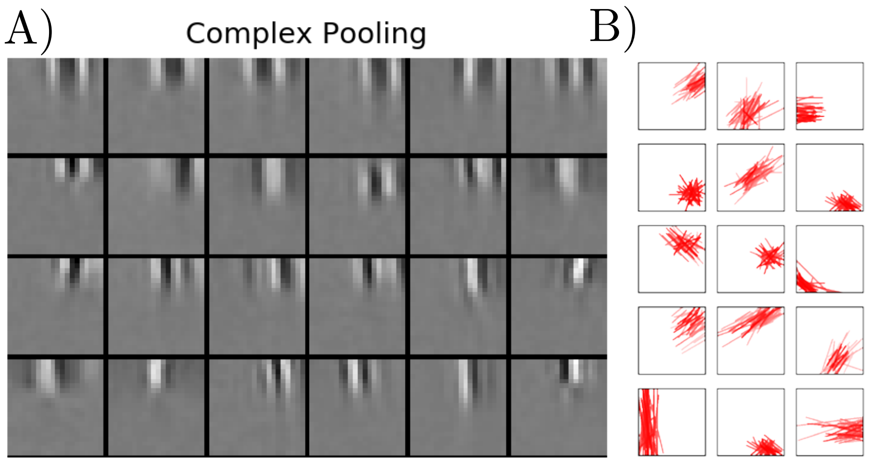

Figure 4.1: A) A group of simple cells pooled from one complex cell. B) A random sample of 15 complex cells, whose pooling fields are shown as needle plots (described in Section 3.3), with each bar representing a simple cell fit with a Gabor function.

V and the whitening matrix $V^{-\frac{1}{2}}$ of the sparse coefficient vectors, extract patch sequences to estimate the matrix U, construct the diffusion kernel K^* from $I - U$, and train 512 pooling units following the sparse regularized diffusion process described in the method section. The number of pooling units is not critical and the result changes gracefully with respect to variation of the pooling dimension.

In Figure 4.1, we show that learned pooling units are relatively 'localized' in the sense that they pool simple cells with similar shapes. To emphasize, 'local' is not defined in the pixel domain but rather in the abstract configuration space, e.g. spatial location, orientation and scale. The distance in this space is determined by the natural transformations in natural signals. Fig 4.2A shows that a complex cell in our model has phase invariance while the simple cells it pools from tile different phases. Fig 4.2B shows the learned complex cells maintain orientation selectivity while gaining phase invariance. Fig 4.2C displays the F1/F0 ratio (Skottun *et al.* 1991) of complex and simple cells in the model, revealing a clear bimodal distribution matching experimental data (De Valois *et al.* 1982; Skottun *et al.* 1991).

4.4 Discussion

In this work, we use the insights from SMT to view simple cells as a discrete sampling of an underlying smooth manifold and complex cells as a smooth tiling of the manifold to build a locally equivariant representation. We show that localized complex pooling cells can be learned from temporal signal dynamics. Multiple sensitivity tests show these units admit similar properties to complex cells in V1. The model predicts that populations of complex cells represent the structure in time-varying images in terms of smooth trajectories that reflect the underlying continuous transformations in images, as opposed to forming invariant representations of objects per se.

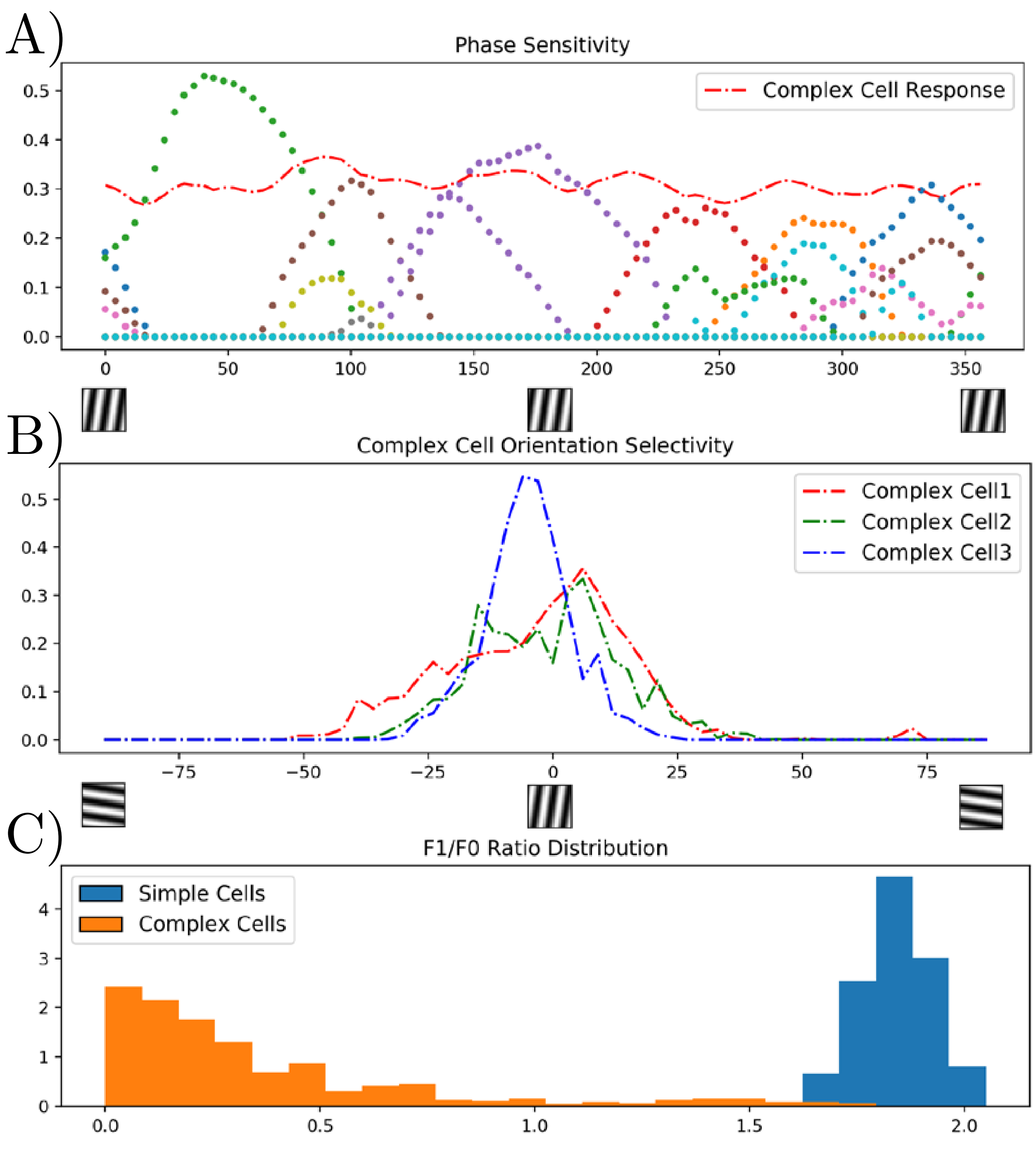

Figure 4.2: A) A phase sweep of the optimal grating stimuli of a complex cell. Dotted lines are the responses from the simple cells it pools from. The invariant red dashed line is the complex cell's response. B) Orientation selectivity of 3 randomly sampled complex cells. C) F1/F0 ratio bimodal distribution.

Chapter 5

Word Embedding Visualization via Word Factors

5.1 Introduction

Several recent works (Mikolov, K. Chen, *et al.* 2013; Pennington, Socher & C. Manning 2014; Bojanowski *et al.* 2017; Peters *et al.* 2018) show that co-occurrence statistics can be used to efficiently learn high-quality continuous vector representations of words from a large corpus of text. The learned word vectors encode rich semantic and syntactic information, which is demonstrated with word analogies. As shown by (Mikolov, Yih, *et al.* 2013), vector arithmetic operations such as: 'king' - 'queen' = 'man' - 'woman' reflects the semantic analogy of what 'king' is to 'queen' as what 'man' is to 'woman'. Thanks to the competitive performance of these models, these methods have become fundamental tools used in the study of natural language processing.

(Allen & Hospedales 2019; Ethayarajh *et al.* 2019; Levy & Goldberg 2014) explain and understand these word embeddings from a theoretical perspective. Empirically, visualizing these embeddings can be difficult since individual dimensions of most word embeddings are not semantically meaningful.

In the absence of tools to visualize and gain further insight about the learned word vectors, we have little hope of improving the existing performance on downstream tasks like word analogy (Mikolov, K. Chen, *et al.* 2013). There are two major ways for visualizing word vectors:

- Nearest neighbor approach: we can use either Euclidean distance or cosine similarity to search for each word vector's nearest neighbors to find its relevant words (Peters *et al.* 2018; Bojanowski *et al.* 2017; Wieting *et al.* n.d.). This method only provides a single scalar number of relatedness information while two words may exhibit much more intricate relationships than just a relatedness (Pennington, Socher & C. D. Manning n.d.). For example, *man* and *woman* both describe human beings and yet are usually considered opposite in gender.

- t-SNE approach (Maaten & G. Hinton 2008; G. E. Hinton & Roweis 2003): This approach nonlinearly reduces word vectors to a very low dimensional (most likely 2) space. While such a global method reveals some interesting separation between word groups, it often distorts (S. Liu *et al.* 2017) important word vector linear structures and does not exhibit more delicate components in each word.

- Subset PCA approach (Mikolov, Sutskever, *et al.* 2013): 1) Select a subset of word pairs, which have certain relations, e.g. city-country, currency-country, comparative etc. 2) Perform PCA on the selected subset and visualize the subset with the first two principle components, as shown in Figure 5.3. The relationship is frequently encoded by a vector in this subspace. However, performing PCA with all the word vectors makes this information entirely opaque. This method needs manually selected sets of words which requires human intervention. Despite this, such a visualization method does capture important semantic meaning for word vectors.

The linear substructures generated using subset PCA approach and semantic content of arithmetic operations provide strong motivation to automatically discover the factors which these underlying word vectors are composed of.

The key insight of this work is that the relationships visualized in the human selected subsets represent more elementary word factors and a word vector is a linear combination of a sparse subset of these factors. Dictionary learning (Olshausen & Field 1997; Olshausen & Field 1996; Bell & Sejnowski 1995) is a useful tool to extract elementary factors from different modalities. (Murphy *et al.* 2012; Fyshe *et al.* 2014; Faruqui *et al.* 2015; Subramanian *et al.* 2018) shows sparsity help to improve the dimension interpretability. Specifically, in (Faruqui *et al.* 2015), the authors apply non-negative sparse coding (NSC) (P. O. Hoyer 2002; D. D. Lee & Seung 1999) with binary coefficients to word vectors and suggest to use the resulted sparse vectors as word vector representation of words. Then (Subramanian *et al.* 2018) followed the idea and applied k-sparse autoencoder to further improve the sparse word vector representation. In this work, we thoroughly explore this idea from a visualization perspective. Since a word vector may involve a different number of factors with a different strength, neither binary coefficients or a k-sparse setting would be ideal for such a purpose. In section 5.2, we reformulate the NSC problem and also introduce the spectral clustering algorithm to further handle group sparsity. Once NSC has been train on word vectors from different word embedding methods, in section 5.3, we demonstrate reliable word factors with very clear semantic meanings, which is consistent with but not limited to the existing prior knowledge. With these reliable word factors, we then open up the word vectors and visualize them in many different ways. Through these visualizations, we show many interesting compositions:

$$\begin{aligned}\text{apple} =& 0.09\text{“dessert”} + 0.11\text{“organism”} + 0.16\text{“fruit”}\\ &+ 0.22\text{“mobile\&IT”} + 0.42\text{“other”}\end{aligned}$$

Many new word analogy tasks can be easily developed based on these learned word factors. Different embedding models and text corpus bias can be diagnosed, i.e. we find that a factor proportion might change depend on which corpus we use to train the word vector embedding.

Since several learned word factors may encode a similar meaning and frequently work together, we can use spectral clustering (Ng *et al.* 2002; Von Luxburg 2007) to group word factors with similar meanings. Each group can provide robust semantic meaning for each factor and identify key semantic differences in a word analogy task. We show in Section 5.4 that these groups help to improve the word analogy tasks significantly in almost every subcategory irrespective to which word embedding technique we use. Our simple and reliable discovery provides a new venue to understand the elementary factors in existing word embedding models. In Section 5.5, we discuss a few interesting questions and point out some potential directions for future work.

5.2 Word Factors Learning and Spectral Grouping

Similar to Section 2.2, We use non-negative *overcomplete* sparse coding to learn word factors from word vectors. Given a set of word vectors $\{x_j \in \mathbb{R}^n\}$ ($n = 300$ is used in this work as a convention), we assume each of them is a sparse and linear superposition of word factors ϕ_i:

$$\boldsymbol{x} = \Phi\,\boldsymbol{\alpha} + \boldsymbol{\epsilon},\ \text{s.t. } \boldsymbol{\alpha} \geq 0 \tag{5.1}$$

where $\Phi \in \mathbb{R}^{n\times d}$ is a matrix with columns $\Phi_{:,i}$, $\boldsymbol{\alpha} \in \mathbb{R}^d$ is a sparse vector of coefficients and $\boldsymbol{\epsilon}$ is a vector containing independent Gaussian noise samples, which are assumed to be small relative to $\boldsymbol{x}$. Typically $d > n$ so that the representation is *overcomplete*. This inverse problem can be efficiently solved by FISTA algorithm (Beck & Teboulle 2009). A word vector $\boldsymbol{x}_i$ is sampled with respect to the responding frequency f_i of the word i in the corpus. Once the sparse coefficients is solved, we can then update the word factors to better reconstruct the word vectors. Through this iterative optimization, the word factors can be learned. We provide more details of the algorithm and configurations in Appendix C.2.

Though overcomplete sparse coding tends to extract more accurate elementary factors and thus approximate signal vectors at better accuracy given a fixed sparsity level, several learned factors may be corresponding to a similar semantic meanings. (Y. Chen *et al.* 2018) proposes to model the relationships by using a manifold embedding, which is essentially a spectral method. In this work, we use spectral clustering (Ng *et al.* 2002) to group the learned word factors into groups.

Since word factors with similar semantic meaning tends to co-activate to decompose a word vector, we calculate a normalized covariance matrix W of word factor coefficients with the unit diagonal removed:

$$W = \sum_i f_i \hat{\boldsymbol{\alpha}}_i \hat{\boldsymbol{\alpha}}_i^T - I \tag{5.2}$$

where f_i is the frequency of the ith word in the corpus, $\hat{\boldsymbol{\alpha}}_i$ is the normalized sparse coefficients by each dimensions standard deviation such that $\hat{\alpha}_{ij} = \alpha_{ij}/\sigma_j$, and $\sigma_j = \left[\sum_i f_i(\alpha_{ij})^2\right]^{\frac{1}{2}}$, $j = 1 \ldots n$.

This matrix captures the similarity between the word factors. To better perform a spectral clustering, we first make the normalized covariance matrix W sparse by selecting k largest values in each row of W:

$$W_{sp} = f_k(W, dim = 0) \tag{5.3}$$

where f_k stands for keeping the k largest values unchanged in the given dimension, while setting all the other entities to 0. Then we obtain a symmetric sparse matrix:

$$W_{adj} = W_{sp} + W_{sp}^T \tag{5.4}$$

where $W_{adj} \in \mathbb{R}^{d\times d}$ is the adjacency matrix used in spectral clustering (Ng *et al.* 2002; Von Luxburg 2007). We use the implementation of the algorithm in Scipy (Jones *et al.* n.d.). Using the notation from (Von Luxburg 2007), we first compute a normalized Laplacian matrix L_{sym}:

$$L_{sym} = I - D^{-1/2} W_{adj} D^{-1/2} \tag{5.5}$$

where D is a diagonal matrix with the sum of each row of the symmetric W_{adj} on its diagonal. Suppose we set the number of clusters to k, then the first k eigenvectors of L_{sym} form the columns of matrix $V \in \mathbb{R}^{\mathrm{d}\times\mathrm{k}}$:

$$V = [v_1, v_2, ..., v_k] \tag{5.6}$$

And we normalize each row of V to get a new matrix U:

$$U = [u_1, u_2, ..., u_d]^T \tag{5.7}$$

where each row $u_i = V_{i,:}/\|V_{i,:}\|_2$. $\|V_{i,:}\|_2$ indicates the L2 norm of the ith row of V. Finally, a k-means algorithm is performed on the d rows of U to get the clusters.

5.3 Visualization

The word factors learned for different word embedding models are qualitatively similar. For simplicity, we show the results for the 300 dimensional GloVe word vectors(Pennington, Socher & C. Manning 2014) pretrained on CommonCrawl (Pennington, Socher & C. D. Manning n.d.). We shall discuss the difference across different embedding models at the end in this section.

Once word factors have been learned and each word vector's sparse decomposition $\boldsymbol{\alpha}$ has been inferred by Equation 5.1, we can denote all the decompositions in the matrix form as the following:

$$X \approx \Phi A, \text{ s.t. } A \geq 0 \tag{5.8}$$

where $X_{:,i} = \boldsymbol{x}_i$ and $A_{:,i} = \boldsymbol{\alpha}_i$, $X \in \mathbb{R}^{n\times N}$ and $A \in \mathbb{R}^{d\times N}$. N is the size of the vocabulary. We do not require the dictionary Φ to be non-negative.

We can visualize a *word factor* $\Phi_{:,j}$ by examining its corresponding row $A_{j,:}$ in A and visualize a word vector $x_{:,i}$ by examining the corresponding column α_i of A. Since word factors are learned in an unsupervised fashion, the explicit meaning of each factor is unknown in advance. To help understand the meaning of a specific factor, we print out the words that have high coefficients for this factor, some examples are illustrated in Table 5.1. We first present the visualization of word factors and give each factor a semantic name. Then we decompose word vectors into linear combinations of word factors. Furthermore, we demonstrate factor groups and discuss the difference across multiple word embedding models.

Word Factor Visualization

Word factors can be visualized through the sparse coding coefficients of each word vector. We refer to these coefficients as activations as they describe how much a factor is turned on for a specific word. In Table 5.1, we demonstrate a set of factors with their top-activation words. Usually the top-activation words for each factor share an obvious semantic or syntactic meaning. Based on the top-activation words, a semantic name can be given to each factor as a guide. For example, for factor 59, we can call it "medical" since most of the words have activation on it are related to medical purpose. In the Appendix C.1, we discuss this naming procedure in more detail.

Reliability. The factors we discovered exhibit strong reliability. For instance, a female factor is illustrated in Figure 5.1. Clearly, the activations remain all 0s for the male words, but have high values for the female words. Similarly strong word factors are also found to capture syntactic meanings of words. Figure 5.4 shows activations on a factor representing the superlative information, i.e. the superlative factor, where the superlative forms of adjectives have relatively high coefficients. The significance is obvious from the sharp contrast in the heights of the bars.

The Learned Factors Match the Prior. We empirically find that for each of the 14 word analogy tasks, there are always a few corresponding factors capture the key semantic difference, e.g. the "female", "superlative", "country" factors shown in Table 5.1. Given the learned word factors closely matched the 14 word analogy tasks chosen based on human priors, we can expect the rest majority of the learned factors may provide an automatic method to select and construct the word analogy tasks. For instance, Figure 5.2 shows a factor corresponding to professions. For words such as "entertain", "poem", "law" and so on, it has 0 or very small activation, whereas for "entertainer", "poet" and "lawyer", the activations are clearly large. In Appendix C.3, we provide more of such generated word analogy tasks by the learned factors.

Unclear Factors. While most of the factors have a strong and clear semantic meaning, there are also about less than 10% of them that we can not identify. Some of these factors have relatively dense activations that they may activate on more than 10% of the whole vocabulary while the activations are relatively low. Some of the factors seem to cluster

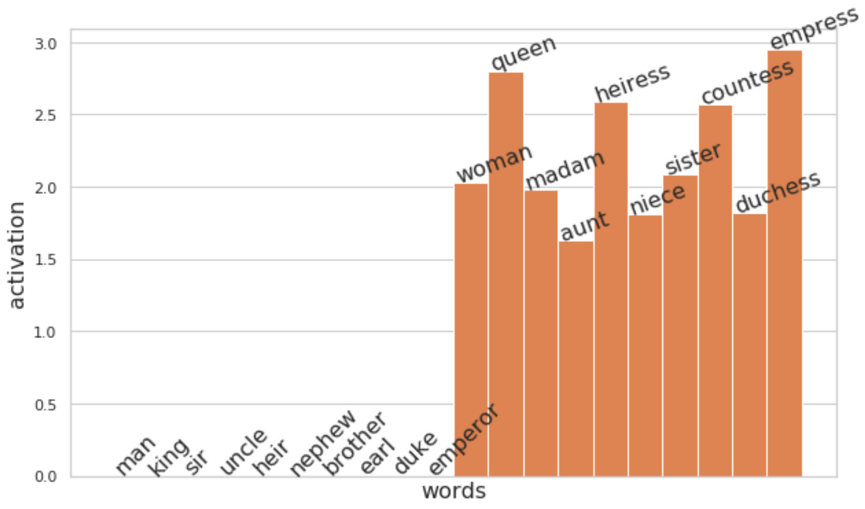

Figure 5.1: "Female" factor's activation w.r.t. a selected set of words contain both male-related words and female related words.

either high or low frequency words regardless of the semantics, e.g. a factor's top-activation words are all rare words. We feel that some of these unidentifiable factors might actually have semantic meaning with more summarization effort and the rest might be due to an optimization choice, e.g. we sampled word embedding in proportion to the words' frequency during optimization, so that high frequency words got more exposure than low frequency ones.

Factor Groups. Different factors may correspond to similar semantic meaning and in a particular word vector, they co-activate or only one of them activate. But in general similar factors tend to have a relatively higher co-activation. Based on the co-activation strength, we can cluster word factors in to groups, each provides a more reliable semantic and syntactic meaning detection. In Appendix C.4, we show the co-activation patterns in more details.

Word Vector Visualization

As a result of sparse coding, every word vector can now be expressed as a linear combination of a limited number of word factors. This makes it possible for us to open up continuous word vectors and see different aspects of meanings through the component factors. In Figure 5.5, we show several word vectors as a combinations of highly activated factors.

Polysemy Separation. Words like "apple" contain multiple senses of meanings but are encoded into one continuous vector. By visualization through word factors, different senses are separated. As is shown in figure 5.5, the vector of "apple" contains 4 major factors: "tech-

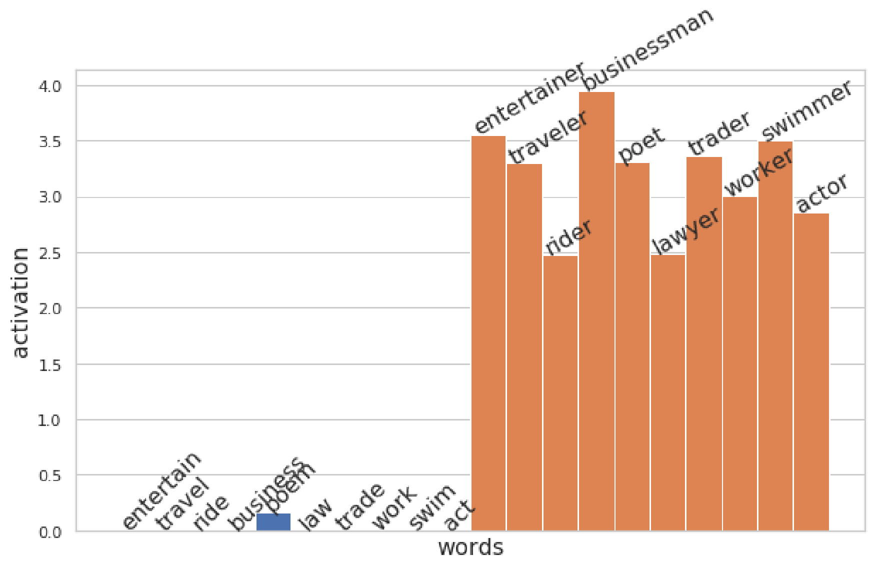

Figure 5.2: "Profession" factor's activation w.r.t. a selected set of words contain action-related words and their profession form.

nology", "fruit", "dessert" and "organism". This combination coincides with our knowledge that "apple" is a fruit, a food ingredient, a living creature and a well-known tech company. Another polysemous example is the word "China": the presence of factor "ware" makes sense as the training corpus is not case sensitive. We further notice the "country" factors and the "Chinese" factor, which is closely related to specific Chinese nouns such as the names of its provinces and cities. In fact a combination of the "country" factor and a "country-specific" factor shows up as a common combination in the names of countries.

Semantic + Syntactic. Besides polysemy, we also find that words are opened up as combinations of both semantic and syntactic factors: "big" has both "size" and "comparative" factor; "won" has "match", "award", "sports" and of course "past tense".

Unexpected Meaning. Sometimes a word may have an unexpected factor, e.g. in the visualization of "so", we find a "German" factor, of which all the top activated words are German word pieces like "doch", "aber", "voll", "schön" and "ich". The possible explanation for this is that the training data of the embedding model covers German corpus, and "so" is actually also used in German.

Word Vectors Manipulations. Prior work (Mikolov, Yih, *et al.* 2013) has demonstrated that linear operations between continuous word vectors can capture linguistic regularities. Now given such factors with clear and strong semantic meanings, it is natural to think of some manipulations. An interesting question is: if we manually add in or subtract out a

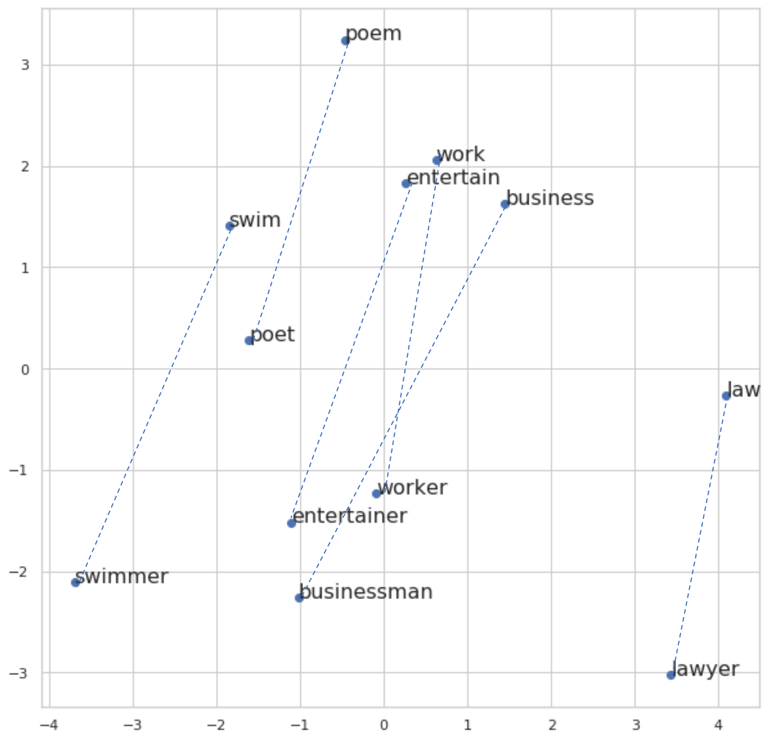

Figure 5.3: PCA visualization of a new word analogy task: "profession", which are automatically generated by the "profession" word factor.

certain factor from a word vector, would the new word vector be consistent with the semantic relations entailed by the manipulation? To validate this, we manipulate a vector with some factors, and see what is the nearest word in the embedding space. Examples are listed in Table 5.2. Since the average norm of the word vectors we use is about 7.2, while the factors are of unit norm, we give a constant coefficient of 4 to the factors so that their lengths become comparable but still shorter than the word vectors, therefore can be appropriately regarded as components of word vectors. Results show that both syntactic and semantic meanings including part of speech, gender, sentiment and so on are successfully modified in the desired way. This interesting experiment shows the potential of word factors. Given their explicit meanings, now we are no longer limited to operations between word vectors, but can also conduct operations between word vectors and factors.

Comparison Across Different Models

We conducted experiments with several mainstream word embedding models, including GloVe, Fasttext, Word2vec CBoW and Word2vec skipgram, all of 300 dimension. For GloVe,

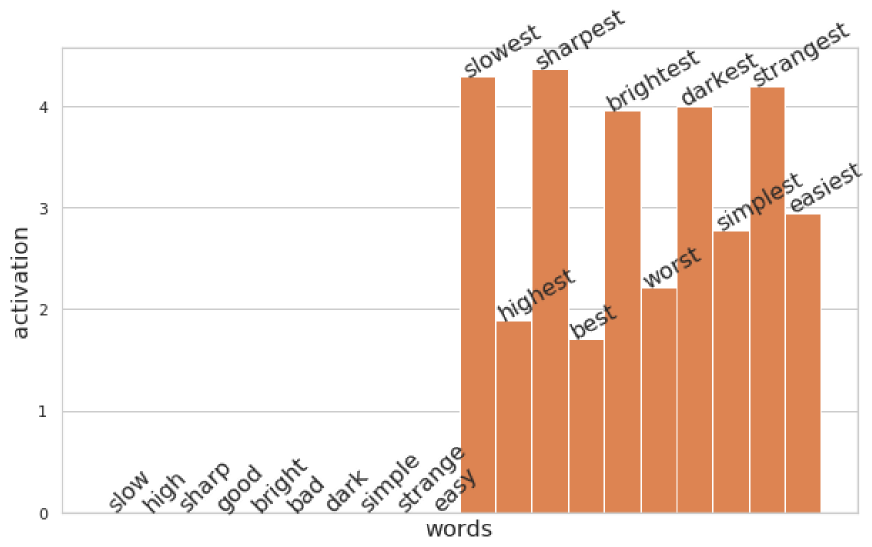

Figure 5.4: "Superlative" factor's activation w.r.t. a selected set of words contain words and their superlative forms.

we download model pretrained on CommonCrawl (Pennington, Socher & C. D. Manning n.d.). For fasttext, we download model pretrained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset (Mikolov, Grave, *et al.* n.d.). And for Word2vec models, we trained them on 3B token wikipedia dump (Wikimedia 2019). Although the results are similar between different embedding models, we also notice some interesting differences that can provide understanding of the models. In the fastText embeddings, word "sing" has a abnormally high activation on the factor "present tense", as is shown in Figure 5.6. This is because the algorithm trains word embeddings based on subword n-grams (Bojanowski *et al.* 2017; Joulin *et al.* 2016), in this way word "sing" is considered as if a word in present tense because it contains a three-gram "ing". This shows that despite the advantages of using subword n-grams to embed words, such as tackling out-of-vocabulary issue and encode strong syntactic information, it may also lead to problems. Such a visualization can be used to diagnose and provide insights to improve the existing methods.

There are also differences that reflect bias in different corpus. For example, we compared the pretrained GloVe and a GloVe model trained only on Wikipedia, and refer to them as "GloVe Crawl" and "GloVe Wiki". As a result, we discovered the factor "bedding" in the visualization of vector "king" in the "GloVe Crawl", while it is missing in "GloVe Wiki". The only difference between the two models is the training data. The presence of factor "bedding" actually makes sense because "king" is frequently used to describe the size of beds and bedclothes. The reason why it is missing in "GloVe Wiki" is likely due to the difference in training data. Which is to say such usage of "king" is much less frequent

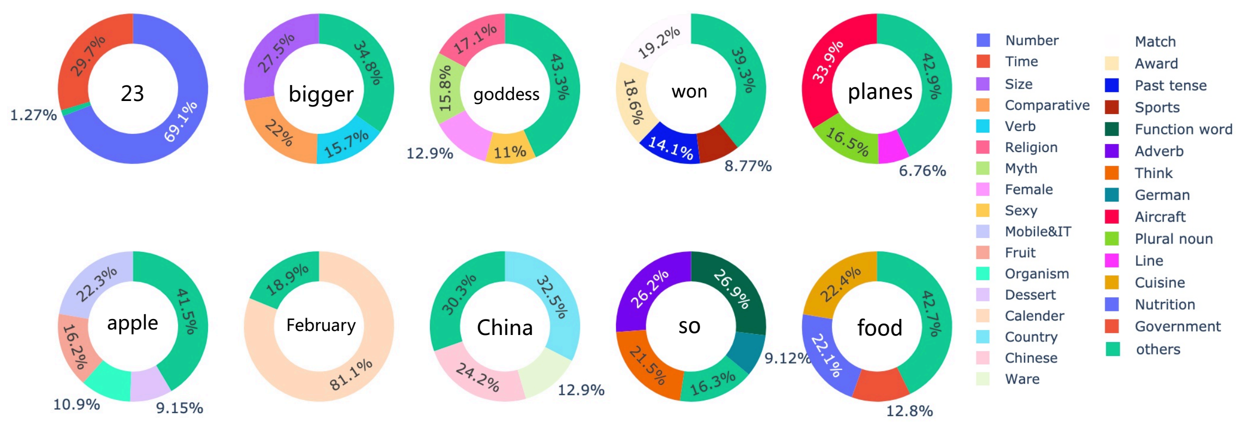

Figure 5.5: Word vectors can be decomposed into a sparse linear combination of word factors. Due to a space limit, we only show the major factors and leave the rest as "others".

in Wikipedia than in CommonCrawl, so it is possible that this aspect of meaning is not significant on Wikipedia. In order to verify this, we examined the average co-occurrence statistics in Wikipedia between "king" and top 100 activated words of factor "royal" and "bedding". The fact that the former is more than 30 times larger than the latter supports the assumption that "king" appears rather rarely in the "bedding" context. Thus a factor of "bedding" is hard to get given the corpus from Wikipedia. This comparison shows that the difference in data is captured by embedding models and displayed by our factors.

5.4 Improvement in Semantic and Syntactic Tasks with Word Factors

Word analogy task is a classic task for evaluating the quality of word embeddings. Proposed first by (Mikolov, K. Chen, *et al.* 2013), it measures the accuracy of answering the question : A is to B as C is to D, such as 'king' is to 'queen' as 'man' is to 'woman' and 'slow' is to 'slower' as 'good' is to 'better'. Given word A, B and C, a model must try to predict D. The conventional approach taken by the word embeddings (Mikolov, K. Chen, *et al.* 2013; Pennington, Socher & C. Manning 2014; Bojanowski *et al.* 2017) is a vector arithmetic: calculate $B - A + C$, and find its nearest neighbor in the embedding space as the predicted word. While this approach leads to good results, several failure modes are shown in Table 5.5. Although vector arithmetic returns a word very close to the ground truth in terms of meaning, it fails to identify the key semantic relationship implied in the question.

However, if semantic meaning captured by the factors are reliable enough, they should be able to identify the semantic difference and therefore correct such mistakes very easily. To

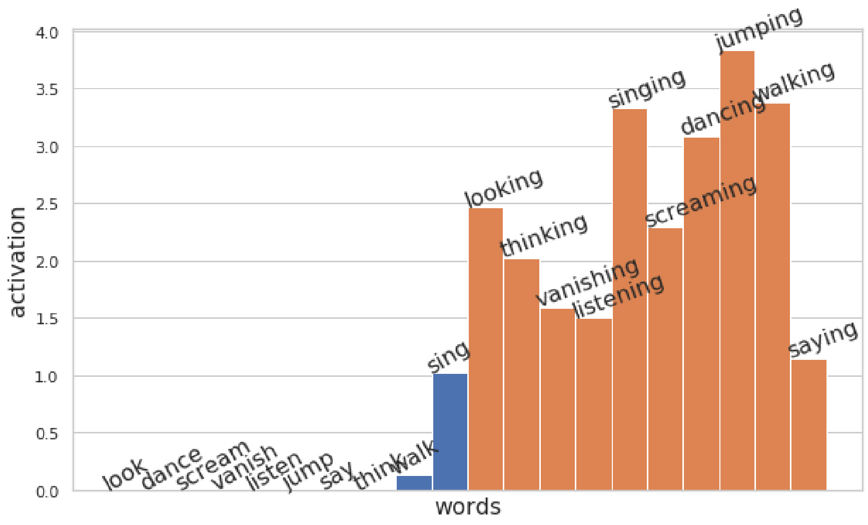

Figure 5.6: The word vector "sing" learned by fastText has a high activation on the "present tense" factor, due to the subword structure 'ing'.

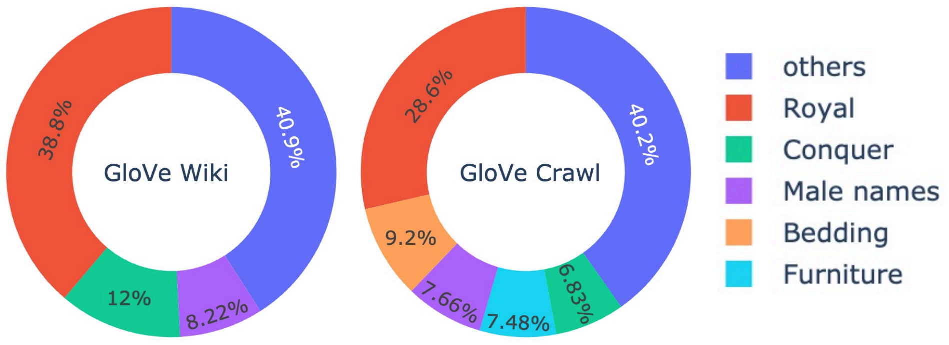

Figure 5.7: Difference between embedding models. The visualization of "king" has significant "bedding" factor in "GloVe Crawl", but it is not found in "GloVe Wiki".

do this, we propose a simple factor group selection approach, where we require the predicted word to not only be the closest in the embedding space, but also have an higher activation on the specific factor groups than that of the words in the other category. For example, to be selected as an answer, "queen"'s activation on the "female" factor group must be higher than those male words in the subtask. By simply applying this factor group selection approach, we achieve consistent improvement for every embedding algorithm on almost every subtask. For many subtasks, the improvement is quite significant. Three experiments in Table 5.3 get a decreased performance, most likely due to the correct factors are not grouped together during spectral clustering.

Besides validating the previously identified relationships, as mentioned in Section 5.3, we are also able to find many new ones using the factors, such as "ideology" (Figure C.2), "profession" (Figure 5.3) and "adj-to-verb" (Figure C.1). The questions are constructed in the same way as word analogy tasks. Here are examples from each new task:

$$V_{collectivist} - V_{collectivism} = V_{liberal} - V_{liberalism}$$

$$V_{entertain} - V_{entertainer} = V_{poem} - V_{poet}$$

$$V_{sensational} - V_{sensationalize} = V_{marginal} - V_{marginalize}$$

Performance of each embedding method on the new tasks is shown in Table 5.4, which has the same behavior as shown in Table 5.3. Consistent improvements are obtained once we apply the simple factor group selection method. This further demonstrates that word factors can capture reliable semantic meanings and the phenomenon is not only constraint to the previously proposed ones in the word analogy task.

5.5 Discussion

In this work, we show that dictionary learning can extract elementary factors from continuous word vectors. By using the learned factors, we can meaningfully decompose word vectors and visualize them in many interesting ways. Further, we demonstrate with these factors, we can consistently improve many word embedding methods in word analogy tasks. The word factors may provide an convenient mechanism to develop new word analogy tasks beyond the existing 14 ones. Further examination of existing word embedding models may leads to further improvements.

A fundamental question that remains to be answered is why word factors can be combined in such linear fashion? (Levy & Goldberg 2014) provides one possible explanation: with sparse word representation, which explicitly encode each word's context statistics, one can also construct equally good word vectors. If we see a word vector as an explicit statistic based on a word's surrounding context, then this context may fall into sub-categories of words. Our guess is that the learned word factors reflects each of these sub-context categories. This suggests an interesting future direction of our work. A limit of this work is that all the word vectors we visualize are trained from methods which ignore the context of a word used in a specific *instance*. Applying dictionary learning to attention-based methods (Vaswani *et al.* 2017; Devlin *et al.* 2019) is another interesting future direction. Finally, the existence of more elementary meaning than words is a debatable argument in linguistic study. The learned word factors may also provide insights and verification to the sememe thoery (Bazell 1966; Niu *et al.* 2017; Xie *et al.* 2017). We leave this to the future work.

factors	top activation words
59 “medical”	hospital, medical, physician, physicians, nurse, doctor, hospitals, doctors, nurses, patient, nursing, medicine, care, healthcare, psychiatric, clinic, psychiatry, ambulance, pediatric
116 “vehicle”	vehicles, vehicle, driving, drivers, cars, car, driver, buses, truck, trucks, taxi, parked, automobile, fleet, bus, taxis, passenger, van, automobiles, accidents, motorcycle, mph
193 “ware”	pottery, bowl, bowls, porcelain, ware, vase, teapot, china, saucer, denby, vases, saucers, ceramic, glass, plates, earthenware, pitcher, wedgwood, pots, plate, tureen, jug, pot, jar
296 “mobile&IT”	ipad, iphone, ios, itunes, apple, android, app, ipod, airplay, 3g, 4s, apps, ipads, htc, tablet galaxy, jailbreak, iphones, netflix, mac, os, touch, nook, skyfire, dock, siri, eris, 4g, tablets
337 “superlative”	strongest, funniest, largest, longest, oldest, fastest, wettest, tallest, heaviest, driest, sexiest, scariest, coldest, hardest, richest, biggest, happiest, smallest, toughest, warmest, most
461 “country”	venezuela, germany, paraguay, uruguay, norway, russia, lithuania, ecuador, netherlands, estonia, korea, brazil, argentina, albania, denmark, poland, europe, sweden, colombia
470 “bedding”	mattress, pillow, bed, mattresses, beds, pillows, queen, ottoman, simmons, cushion, bedding, topper, foam, plush, sleeper, sofa, comforter, couch, futon, seat, bolster, pad, sleeping
493 “royal”	king, royal, throne, prince, monarch, emperor, duke, queen, reign, coronation, kings, empress, regent, dynasty, palace, monarchs, ruler, crown, heir, monarchy, kingdom, sultan, consort
582 “fruit”	fruit, fruits, pears, oranges, apples, peaches, grapes, apple, ripe, plums, bananas, mandarin, grapefruit, peach, berries, tomatoes, kiwi, watermelon, berry, lemons, mango, canning, kiwis
635 “Chinese”	china, fujian, zhejiang, guangdong, hangzhou, shandong, shanghai, qingdao, beijing, chongqing, guangzhou, sichuan, jiangsu, hainan, hebei, luoyang, shenzhen, nanjing, henan
781 “national”	croatian, american, lithuanian, norwegian, vietnamese, chinese, romanian, bulgarian, indonesian, armenian serbian, turkish, hungarian, korean, malaysian, italian, austrian
886 “female”	her, queen, herself, she, actress, feminist, heroine, princess, empress, sisters, woman, dowager, lady, sister, mother, goddess, women, daughter, diva, maiden, girl, née, feminism, heroines

Table 5.1: In this figure we show a set of learned factors with its top-activation words. Based on the common aspect of the first 20% of the top-activation words (usually around 100 words), we can give each of the factors a semantic name.

Table 5.2: Factor-vector manipulations and factor-factor manipulations. The word vectors' average norm is 7.2. The learned word factors all have unit norm.

Manipulations	**Nearest Neighbors**
$V_{good} + 4F_{superlative}$	V_{best}
$V_{king} + 4F_{female}$	V_{queen}
$V_{Italy} + 4F_{national}$	$V_{Italian}$
$V_{unwise} + 4F_{negative\ prefix}$	$V_{prudent}$
$V_{hospital} + 4F_{vehicle}$	$V_{ambulance}$
$V_{soldier} - 4F_{military}$	V_{man}
$V_{dancers} - 4F_{profession}$	V_{dance}
$V_{kindle} - 4F_{book}$	V_{ipad}

Table 5.3: Performance on word analogy task for different word embedding models

	W2V sg ori	W2V sg group	Fasttext ori	Fasttext group	W2V cb ori	W2V cb group	GloVe ori	GloVe group
0	93.87	93.87	56.92	**58.10**	88.34	**89.13**	80.04	83.99
1	89.86	**90.52**	40.01	**40.79**	87.48	**87.97**	79.86	**83.52**
2	11.06	**12.02**	36.54	**37.26**	16.35	**18.27**	20.19	**22.84**
3	72.15	**75.76**	23.89	**24.11**	68.22	**68.79**	65.46	**66.11**
4	86.17	**87.15**	89.13	**89.92**	89.33	**90.12**	96.44	**98.02**
5	32.16	**44.56**	75.00	**76.61**	34.38	**38.81**	43.15	42.24
6	44.46	**47.91**	68.97	**71.18**	40.39	**43.47**	35.10	**42.61**
7	83.93	**86.71**	97.15	97.15	89.56	**90.24**	87.69	**91.67**
8	62.59	**73.86**	99.15	99.15	63.83	**69.79**	92.23	**95.55**
9	66.38	**87.41**	97.73	**100.00**	69.98	**86.74**	82.77	**96.12**
10	90.31	**90.43**	85.05	**85.74**	88.12	**88.68**	69.36	**72.67**
11	61.99	**63.08**	84.94	**90.71**	67.05	**71.60**	64.10	**83.72**
12	76.43	71.55	96.70	**97.15**	80.26	**82.36**	95.95	91.44
13	71.26	**82.99**	95.63	**98.74**	63.79	**72.07**	67.24	**88.51**
Sem	79.53	**81.16**	40.81	**41.42**	77.22	**77.86**	72.84	**75.31**
Syn	67.95	**73.47**	89.38	**91.19**	69.32	**74.01**	72.59	**79.80**
Tot	72.70	**76.63**	74.42	**75.86**	72.56	**75.59**	72.69	**77.96**

Table 5.4: Word analogy performance on new tasks for different word embedding models

	W2V sg ori	W2V sg group	Fasttext ori	Fasttext group	W2V cb ori	W2V cb group	GloVe ori	GloVe group
Ideology	56.00	56.00	93.33	93.33	56.50	**58.67**	82.00	**85.50**
Profession	27.78	**38.30**	65.79	**77.19**	36.55	**45.03**	36.26	**53.80**
Adj-to-verb	8.97	**13.46**	62.82	**65.38**	16.03	**16.67**	24.36	**26.28**
Total	40.53	**44.44**	80.42	**84.34**	44.54	**48.45**	59.56	**67.21**

Table 5.5: A few examples to show the typical errors in word analogy tasks using word vectors.

Vector arithmetic	Ground truth	Prediction
unreasonable - reasonable + competitive	uncompetitive	competition
worse - bad + cheap	cheaper	cheapest
greece - athens + beijing	china	shanghai
danced - dancing + enhancing	enhanced	enhance
wife - husband + policeman	policewoman	policemen

Chapter 6

Learning Energy-based Models with Multiscale Denoising Score Matching

6.1 Introduction

Treating data as stochastic samples from a probability distribution and developing models that can learn such distributions is at the core for solving a large variety of application problems, such as error correction/denoising (Vincent, Larochelle, Lajoie, *et al.* 2010), outlier/novelty detection (Zhai *et al.* 2016; Choi & Jang 2018), sample generation (Nijkamp *et al.* 2019; Du & Mordatch 2019), invariant pattern recognition, Bayesian reasoning (Welling & Teh 2011) which relies on good data priors, and many others.

Energy-Based Models (EBMs) (LeCun *et al.* 2006; Ngiam *et al.* 2011) assign an energy $E(\boldsymbol{x})$ to each data point $\boldsymbol{x}$ which implicitly defines a probability by the Boltzmann distribution $p_m(\boldsymbol{x}) = e^{-E(\boldsymbol{x})}/Z$. Sampling from this distribution can be used as a generative process that yield plausible samples of $\boldsymbol{x}$. Compared to other generative models, like GANs (Goodfellow *et al.* 2014), flow-based models (Dinh *et al.* 2015; Kingma & Dhariwal 2018), or auto-regressive models (van den Oord *et al.* 2016; Ostrovski *et al.* 2018), energy-based models have significant advantages. First, they provide explicit (unnormalized) density information, compositionality (G. E. Hinton 1999; Haarnoja *et al.* 2017), better mode coverage (Kumar *et al.* 2019) and flexibility (Du & Mordatch 2019). Further, they do not require special model architecture, unlike auto-regressive and flow-based models. Recently, Energy-based models has been successfully trained with maximum likelihood (Nijkamp *et al.* 2019; Du & Mordatch 2019), but training can be very computationally demanding due to the need of sampling model distribution. Variants with a truncated sampling procedure have been proposed, such as contrastive divergence (G. E. Hinton 2002). Such models learn much faster with the draw back of not exploring the state space thoroughly (Tieleman 2008).

Score Matching, Denoising Score Matching and Deep Energy Estimators

Score matching (SM) (Hyvärinen 2005) circumvents the requirement of sampling the model distribution. In score matching, the score function is defined to be the gradient of log-density or the negative energy function. The expected $L2$ norm of difference between the model score function and the data score function are minimized. One convenient way of using score matching is learning the energy function corresponding to a Gaussian kernel Parzen density estimator (Parzen 1962) of the data: $p_{\sigma_0}(\tilde{\boldsymbol{x}}) = \int q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}$. Though hard to evaluate, the data score is well defined: $s_d(\tilde{\boldsymbol{x}}) = \nabla_{\tilde{\boldsymbol{x}}} \log(p_{\sigma_0}(\tilde{\boldsymbol{x}}))$, and the corresponding objective is:

$$L_{SM}(\theta) = \mathbb{E}_{p_{\sigma 0}(\tilde{\boldsymbol{x}})} \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(p_{\sigma_0}(\tilde{\boldsymbol{x}})) + \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 \tag{6.1}$$

(Vincent 2011) studied the connection between denoising auto-encoder and score matching, and proved the remarkable result that the following objective, named *Denoising Score Matching* (DSM), is equivalent to the objective above:

$$L_{DSM}(\theta) = \mathbb{E}_{p_{\sigma_0}(\tilde{\boldsymbol{x}},\boldsymbol{x})} \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(q_{\sigma 0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})) + \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 \tag{6.2}$$

Note that in (6.2) the Parzen density score is replaced by the derivative of log density of the single noise kernel $\nabla_{\tilde{\boldsymbol{x}}} \log(q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x}))$, which is much easier to evaluate. In the particular case of Gaussian noise, $\log(q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})) = -\frac{(\tilde{\boldsymbol{x}}-\boldsymbol{x})^2}{2\sigma_0^2} + C$, and therefore:

$$L_{DSM}(\theta) = \mathbb{E}_{p_{\sigma 0}(\tilde{\boldsymbol{x}},\boldsymbol{x})} \parallel \boldsymbol{x} - \tilde{\boldsymbol{x}} + {\sigma_0}^2 \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 \tag{6.3}$$

The interpretation of objective (6.3) is simple, it forces the energy gradient to align with the vector pointing from the noisy sample to the clean data sample. Essentially, this requires the derivative of the energy function to be a denoising autoencoder (Vincent, Larochelle, Bengio, *et al.* 2008), therefore it is named denoising score matching. To optimize an objective involving the derivative of a function defined by a neural network, (Kingma & LeCun 2010) proposed the use of double backpropagation (Drucker & Le Cun 1991). *Deep energy estimator networks* (Saremi, Mehrjou, *et al.* 2018) first applied this technique to learn an energy function defined by a deep neural network. In this work and similarly in (Saremi & Hyvarinen 2019), an energy-based model was trained to match a Parzen density estimator of data with a certain noise magnitude. The previous models were able to perform denoising task, but they were unable to generate high-quality data samples from a random input initialization. Recently, (Song & Ermon 2019) trained an excellent generative model by fitting a series of score estimators coupled together in a single neural network, each matching the score of a Parzen estimator with a different noise magnitude.

The questions we address here is why learning energy-based models with single noise level does not permit high-quality sample generation and what can be done to improve energy based models. Our work builds on key ideas from (Saremi, Mehrjou, *et al.* 2018; Saremi & Hyvarinen 2019; Song & Ermon 2019). Section 6.2, provides a geometric view of the learning problem in denoising score matching and provides a theoretical explanation why training with one noise level is insufficient if the data dimension is high. Section 6.3 presents a novel method for training energy based model, *Multiscale Denoising Score Matching* (MDSM). Section 6.4 describes empirical results of the MDSM model and comparisons with other models.

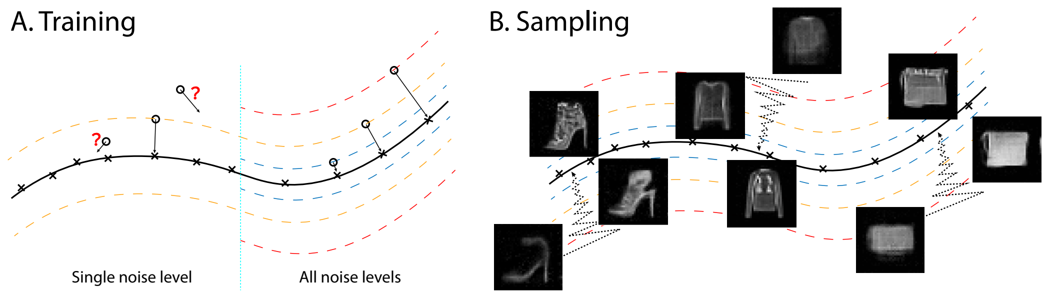

Figure 6.1: Illustration of anneal denoising score matching. A. During training, derivative of log-likelihood is forced to point toward data manifold, establishing energy difference between points within manifold and points outside. Note that energy is negative log-likelihood therefore energy is higher for point further away from data manifold. B. During annealed Langevin sampling, sample travel from outside data manifold to data manifold. Shown are singled step denoised sample during sampling of an energy function trained with MDSM on Fashion-MNIST (see text for details).

6.2 A Geometric view of Denoising Score Matching

(Song & Ermon 2019) used denoising score matching with a range of noise levels, achieving great empirical results. The authors explained that large noise perturbation are required to enable the learning of the score in low-data density regions. But it is still unclear why a series of different noise levels are necessary, rather than one single large noise level. Following (Saremi & Hyvarinen 2019), we analyze the learning process in denoising score matching based on measure concentration properties of high-dimensional random vectors.

We adopt the common assumption that the data distribution to be learned is high-dimensional, but only has support around a relatively low-dimensional manifold (Tenenbaum *et al.* 2000; Roweis & Saul 2000; Lawrence 2005). If the assumption holds, it causes a problem for score matching: The density, or the gradient of the density is then undefined outside the manifold, making it difficult to train a valid density model for the data distribution defined on the entire space. (Saremi & Hyvarinen 2019) and (Song & Ermon 2019) discussed this problem and proposed to smooth the data distribution with a Gaussian kernel to alleviate the issue.

To further understand the learning in denoising score matching when the data lie on a manifold $\mathcal{X}$ and the data dimension is high, two elementary properties of random Gaussian vectors in high-dimensional spaces are helpful: First, the length distribution of random vectors becomes concentrated at $\sqrt{d}\sigma$ (Vershynin 2018), where σ^2 is the variance of a single dimension. Second, a random vector is always close to orthogonal to a fixed vector (Tao 2012). With these premises one can visualize the configuration of noisy and noiseless data points that enter the learning process: A data point $\boldsymbol{x}$ sampled from $\mathcal{X}$ and its noisy version $\tilde{\boldsymbol{x}}$ always lie on a line which is almost perpendicular to the tangent space $T_{\boldsymbol{x}}\mathcal{X}$ and intersects $\mathcal{X}$ at $\boldsymbol{x}$. Further, the distance vectors between $(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ pairs all have similar length $\sqrt{d}\sigma$. As a consequence, the set of noisy data points concentrate on a set $\tilde{\mathcal{X}}_{\sqrt{d}\sigma,\epsilon}$ that has a distance with $(\sqrt{d}\sigma - \epsilon, \sqrt{d}\sigma + \epsilon)$ from the data manifold $\mathcal{X}$, where $\epsilon \ll \sqrt{d}\sigma$.

Therefore, performing denoising score matching learning with $(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ pairs generated with a fixed noise level σ, which is the approach taken previously except in (Song & Ermon 2019), will match the score in the set $\tilde{\mathcal{X}}_{\sqrt{d}\sigma,\epsilon}$ and enable denoising of noisy points in the same set. However, the learning provides little information about the density outside this set, farther or closer to the data manifold, as noisy samples outside $\tilde{\mathcal{X}}_{\sqrt{d}\sigma,\epsilon}$ rarely appear in the training process. An illustration is presented in Figure 6.1A.

Let $\tilde{\mathcal{X}}^C_{\sqrt{d}\sigma,\epsilon}$ denote the complement of the set $\tilde{\mathcal{X}}_{\sqrt{d}\sigma,\epsilon}$. Even if $p_{\sigma_0}(\tilde{\boldsymbol{x}} \in \tilde{\mathcal{X}}^C_{\sqrt{d}\sigma,\epsilon})$ is very small in high-dimensional space, the score in $\tilde{\mathcal{X}}^C_{\sqrt{d}\sigma,\epsilon}$ still plays a critical role in sampling from random initialization. This analysis may explain why models based on denoising score matching, trained with a single noise level encounter difficulties in generating data samples when initialized at random. For an empirical support of this explanation, see our experiments with models trained with single noise magnitudes (Appendix B.2). To remedy this problem, one has to apply a learning procedure of the sort proposed in (Song & Ermon 2019), in which samples with different noise levels are used. Depending on the dimension of the data, the different noise levels have to be spaced narrowly enough to avoid empty regions in the data space. In the following, we will use Gaussian noise and employ a Gaussian scale mixture to produce the noisy data samples for the training (for details, See Section 6.3 and Appendix B.1).

Another interesting property of denoising score matching was suggested in the denoising auto-encoder literature (Vincent, Larochelle, Lajoie, *et al.* 2010; Karklin & Simoncelli 2011). With increasing noise level, the learned features tend to have larger spatial scale. In our experiment we observe similar phenomenon when training model with denoising score matching with single noise scale. If one compare samples in Figure B.1, Appendix B.2, it is evident that noise level of 0.3 produced a model that learned short range correlation that spans only a few pixels, noise level of 0.6 learns longer stroke structure without coherent overall structure, and noise level of 1 learns more coherent long range structure without details such as stroke width variations. This suggests that training with single noise level in denoising score matching is not sufficient for learning a model capable of high-quality sample synthesis, as such a model have to capture data structure of all scales.

6.3 Learning Energy-based model with Multiscale Denoising score matching

Multiscale Denoising Score matching

Motivated by the analysis in section 6.2, we strive to develop an EBM based on denoising score matching that can be trained with noisy samples in which the noise level is not fixed but drawn from a distribution. The model should approximate the Parzen density estimator of the data $p_{\sigma_0}(\tilde{\boldsymbol{x}}) = \int q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})p(\boldsymbol{x})dx$. Specifically, the learning should minimize the difference between the derivative of the energy and the score of p_{σ_0} under the expectation $\mathbb{E}_{p_M(\tilde{\boldsymbol{x}})}$ rather than $\mathbb{E}_{p_{\sigma_0}(\tilde{\boldsymbol{x}})}$, the expectation taken in standard denoising score matching. Here $p_M(\tilde{\boldsymbol{x}}) = \int q_M(\tilde{\boldsymbol{x}}|\boldsymbol{x})p(\boldsymbol{x})dx$ is chosen to cover the signal space more evenly to avoid the measure concentration issue described above. The resulting *Multiscale Score Matching* (MSM) objective is:

$$L_{MSM}(\theta) = \mathbb{E}_{p_M(\tilde{\boldsymbol{x}})} \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(p_{\sigma_0}(\tilde{\boldsymbol{x}})) + \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 \tag{6.4}$$

Compared to the objective of denoising score matching (6.1), the only change in the new objective (6.4) is the expectation. Both objectives are consistent, if $p_M(\tilde{\boldsymbol{x}})$ and $p_{\sigma_0}(\tilde{\boldsymbol{x}})$ have the same support, as shown formally in Proposition 1 of Appendix B.1. In Proposition 2, we prove that Equation 6.4 is equivalent to the following denoising score matching objective:

$$L_{MDSM^*} = \mathbb{E}_{p_M(\tilde{\boldsymbol{x}})q_{\sigma_0}(\boldsymbol{x}|\tilde{\boldsymbol{x}})} \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(q_{\sigma 0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})) + \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 \tag{6.5}$$

The above results hold for any noise kernel $q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})$, but Equation 6.5 contains the reversed expectation, which is difficult to evaluate in general. To proceed, we choose $q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ to be Gaussian, and also choose $q_M(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ to be a Gaussian scale mixture: $q_M(\tilde{\boldsymbol{x}}|\boldsymbol{x}) = \int q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})p(\sigma)d\sigma$ and $q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}, \sigma^2 I_d)$. After algebraic manipulation and one approximation (see the derivation following Proposition 2 in Appendix B.1), we can transform Equation 6.5 into a more convenient form, which we call *Multiscale Denoising Score Matching* (MDSM):

$$L_{MDSM} = \mathbb{E}_{p(\sigma)q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})p(\boldsymbol{x})} \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(q_{\sigma 0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})) + \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 \tag{6.6}$$

The square loss term evaluated at noisy points $\tilde{\boldsymbol{x}}$ at larger distances from the true data points $\boldsymbol{x}$ will have larger magnitude. Therefore, in practice it is convenient to add a monotonically decreasing term $l(\sigma)$ for balancing the different noise scales, e.g. $l(\sigma) = \frac{1}{\sigma^2}$. Ideally, we want our model to learn the correct gradient everywhere, so we would need to add noise of all levels. However, learning denoising score matching at very large or very small noise levels is useless. At very large noise levels the information of the original sample is completely lost. Conversely, in the limit of small noise, the noisy sample is virtually indistinguishable from real data. In neither case one can learn a gradient which is informative about the data structure. Thus, the noise range needs only to be broad enough to encourage learning of data features over all scales. Particularly, we do not sample σ but instead choose a series of fixed σ values $\sigma_1 \cdots \sigma_K$. Further, substituting $\log(q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})) = -\frac{(\tilde{\boldsymbol{x}}-\boldsymbol{x})^2}{2\sigma_0^2} + C$ into Equation 6.4, we arrive at the final objective:

$$L(\theta) = \sum_{\sigma \in \{\sigma_1 \cdots \sigma_K\}} \mathbb{E}_{q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})p(\boldsymbol{x})} l(\sigma) \parallel \boldsymbol{x} - \tilde{\boldsymbol{x}} + \sigma_0^2 \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}}; \theta) \parallel^2 \tag{6.7}$$

It may seem that σ_0 is an important hyperparameter to our model, but after our approximation σ_0 become just a scaling factor in front of the energy function, and can be simply set to one as long as the temperature range during sampling is scaled accordingly (See Section 6.3). Therefore the only hyper-parameter is the rang of noise levels used during training.

On the surface, objective (6.7) looks similar to the one in (Song & Ermon 2019). The important difference is that Equation 6.7 approximates a *single* distribution, namely $p_{\sigma_0}(\tilde{\boldsymbol{x}})$, the data smoothed with one fixed kernel $q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})$. In contrast, (Song & Ermon 2019) approximate the score of *multiple* distributions, the family of distributions $\{p_{\sigma_i}(\tilde{\boldsymbol{x}}) : i = 1, ..., n\}$, resulting from the data smoothed by kernels of different widths σ_i. Because our model learns only a single target distribution, it does not require noise magnitude as input.

Sampling by Annealed Langevin Dynamics

Langevin dynamics has been used to sample from neural network energy functions (Du & Mordatch 2019; Nijkamp *et al.* 2019). However, the studies described difficulties with mode exploration unless very large number of sampling steps is used. To improve mode exploration, we propose incorporating simulated annealing in the Langevin dynamics. Simulated annealing (Kirkpatrick *et al.* 1983; Neal 2001) improves mode exploration by sampling first at high temperature and then cooling down gradually. This has been successfully applied to challenging computational problems, such as combinatorial optimization.

To apply simulated annealing to Langevin dynamics. Note that in a model of Brownian motion of a physical particle, the temperature in the Langevin equation enters as a factor $\sqrt{T}$ in front of the noise term, some literature uses $\sqrt{\beta^{-1}}$ where $\beta = 1/T$ (Jordan *et al.* 1998). Adopting the $\sqrt{T}$ convention, the Langevin sampling process (Bellec *et al.* 2017) is given by:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \frac{\epsilon^2}{2} \nabla_{\boldsymbol{x}} E(\boldsymbol{x}_t; \theta) + \epsilon \sqrt{T_t} \mathcal{N}(0, I_d) \tag{6.8}$$

where T_t follows some annealing schedule, and ϵ denotes step length, which is fixed. During sampling, samples behave very much like physical particles under Brownian motion in a potential field. Because the particles have average energies close to the their current thermic energy, they explore the state space at different distances from data manifold depending on temperature. Eventually, they settle somewhere on the data manifold. The behavior of the particle's energy value during a typical annealing process is depicted in Appendix Figure B.7B.

If the obtained sample is still slightly noisy, we can apply a single step gradient denoising jump (Saremi & Hyvarinen 2019) to improve sample quality:

$$\boldsymbol{x}_{clean} = \boldsymbol{x}_{noisy} - \sigma_0^2 \nabla_{\boldsymbol{x}} E(\boldsymbol{x}_{noisy}; \theta) \tag{6.9}$$

This denoising procedure can be applied to noisy sample with any level of Gaussian noise because in our model the gradient automatically has the right magnitude to denoise the sample. This process is justified by the Empirical Bayes interpretation of this denoising process, as studied in (Saremi & Hyvarinen 2019).

(Song & Ermon 2019) also call their sample generation process annealed Langevin dynamics. It should be noted that their sampling process does not coincide with Equation 6.8. Their sampling procedure is best understood as sequentially sampling a series of distributions corresponding to data distribution corrupted by different levels of noise.

6.4 Image Modeling Results

Training and Sampling Details. The proposed energy-based model is trained on standard image datasets, specifically MNIST, Fashion MNIST, CelebA (Z. Liu *et al.* 2015) and CIFAR-10 (Krizhevsky, G. Hinton, *et al.* 2009). During training we set $\sigma_0 = 0.1$ and train over a noise range of $\sigma \in [0.05, 1.2]$, with the different noise uniformly spaced on the batch dimension. For MNIST and Fashion MNIST we used geometrically distributed noise in the range $[0.1, 3]$. The weighting factor $l(\sigma)$ is always set to $1/\sigma^2$ to make the square term roughly independent of σ. We fix the batch size at 128 and use the Adam optimizer with a learning rate of 5×10^{-5}. For MNIST and Fashion MNIST, we use a 12-Layer ResNet with 64 filters, for the CelebA and CIFAT-10 data sets we used a 18-Layer ResNet with 128 filters (He *et al.* 2016a; He *et al.* 2016b). No normalization layer was used in any of the networks. We designed the output layer of all networks to take a generalized quadratic form (Fan *et al.* 2018). Because the energy function is anticipated to be approximately quadratic with respect to the noise level, this modification was able to boost the performance significantly. For more detail on training and model architecture, see Appendix B.4. One notable result is that since our training method does not involve sampling, we achieved a speed up of roughly an order of magnitude compared to the maximum-likelihood training using Langevin dynamics [1]. Our method thus enables the training of energy-based models even when limited computational resources prohibit maximum likelihood methods.

We found that the choice of the maximum noise level has little effect on learning as long as it is large enough to encourage learning of the longest range features in the data. However, as expected, learning with too small or too large noise levels is not beneficial and can even destabilize the training process. Further, our method appeared to be relatively insensitive to how the noise levels are distributed over a chosen range. Geometrically spaced noise as in (Song & Ermon 2019) and linearly spaced noise both work, although in our case learning with linearly spaced noise was somewhat more robust.

For sampling the learned energy function we used annealed Langevin dynamics with an empirically optimized annealing schedule,see Figure B.7B for the particular shape of

[1]For example, on a single GPU, training MNIST with a 12-layer Resnet takes 0.3s per batch with our method, while maximum likelihood training with a modest 30 Langevin step per weight update takes 3s per batch. Both methods need similar number of weight updates to train.

annealing schedule we used. In contrast, annealing schedules with theoretical guaranteed convergence property takes extremely long (S. Geman & D. Geman 1984). The range of temperatures to use in the sampling process depends on the choice of σ_0, as the equilibrium distribution is roughly images with Gaussian noise of magnitude $\sqrt{T}\sigma_0$ added on top. To ease traveling between modes far apart and ensure even sampling, the initial temperature needs to be high enough to inject noise of sufficient magnitude. A choice of $T = 100$, which corresponds to added noise of magnitude $\sqrt{100} * 0.1 = 1$, seems to be sufficient starting point. For step length ϵ we generally used 0.02, although any value within the range $[0.015, 0.05]$ seemed to work fine. After the annealing process we performed a single step denoising to further enhance sample quality.

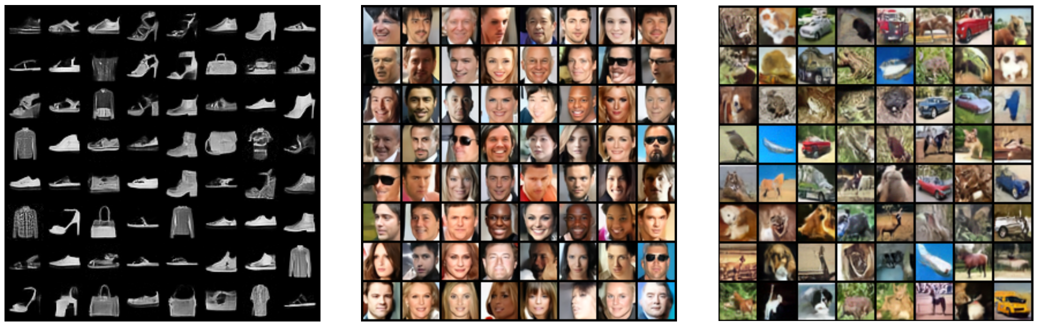

Figure 6.2: Samples from our model trained on Fashion MNIST, CelebA and CIFAR-10. See Figure B.5 and Figure B.6 in Appendix for more samples and comparison with training data.

Unconditional Image Generation. We demonstrate the generative ability of our model by displaying samples obtained by annealed Langevin sampling and single step denoising jump. We evaluated 50k sampled images after training on CIFAR-10 with two performance scores, Inception (Salimans *et al.* 2016) and FID (Heusel *et al.* 2017). We achieved Inception Score of 8.31 and FID of 31.7, comparable to modern GAN approaches. Scores for CelebA dataset are not reported here as they are not commonly reported and may depend on the specific pre-processing used. More samples and training images are provided in Appendix for visual inspection. We believe that visual assessment is still essential because of the possible issues with the Inception score (Barratt & R. Sharma 2018). Indeed, we also found that the visually impressive samples were not necessarily the one achieving the highest Inception Score.

Although overfitting is not a common concern for generative models, we still tested our model for overfitting. We found no indication for overfitting by comparing model samples with their nearest neighbors in the data set, see Figure B.2 in Appendix.

[2]Author reported difficulties evaluating Likelihood

[3]Upper Bound obtained by Reverse AIS

Table 6.1: Unconditional Inception score, FID scores and Likelihoods for CIFAR-10

Model	**IS ↑**	**FID ↓**	**Likelihood**	**NNL (bits/dim) ↓**
iResNet (Behrmann *et al.* 2019)	-	65.01	Yes	3.45
PixelCNN (van den Oord *et al.* 2016)	4.60	65.93	Yes	**3.14**
PixelIQN (Ostrovski *et al.* 2018)	5.29	49.46	Yes	-
Residual Flow (R. T. Chen *et al.* 2019)	-	46.37	Yes	3.28
GLOW (Kingma & Dhariwal 2018)	-	46.90	Yes	3.35
EBM (ensemble) (Du & Mordatch 2019)	6.78	38.2	Yes	- [2]
MDSM (Ours)	8.31	31.7	Yes	7.04 [3]
SNGAN (Miyato *et al.* 2018)	8.22	**21.7**	No	-
NCSN (Song & Ermon 2019)	**8.91**	25.32	No	-

Mode Coverage. We repeated with our model the 3 channel MNIST mode coverage experiment similar to the one in (Kumar *et al.* 2019). An energy-based model was trained on 3-channel data where each channel is a random MNIST digit. Then 8000 samples were taken from the model and each channel was classified using a small MNIST classifier network. We obtained results of the 966 modes, comparable to GAN approaches. Training was successful and our model assigned low energy to all the learned modes, but some modes were not accessed during sampling, likely due to the Langevin Dynamics failing to explore these modes. A better sampling technique such as HMC (Neal *et al.* 2011) or a Maximum Entropy Generator (Kumar *et al.* 2019) could improve this result.

Image Inpainting. Image impainting can be achieved with our model by clamping a part of the image to ground truth and performing the same annealed Langevin and Jump sampling procedure on the missing part of the image. Noise appropriate to the sampling temperature need to be added to the clamped inputs. The quality of inpainting results of our model trained on CelebA and CIFAR-10 can be assessed in Figure 6.3. For CIFAR-10 inpainting results we used the test set.

Log likelihood estimation. For energy-based models, the log density can be obtained after estimating the partition function with Annealed Importance Sampling (AIS) (Salakhutdinov & Murray 2008) or Reverse AIS (Burda *et al.* 2015). In our experiment on CIFAR-10 model, similar to reports in (Du & Mordatch 2019), there is still a substantial gap between AIS and Reverse AIS estimation, even after very substantial computational effort. In Table 6.1, we report result from Reverse AIS, as it tends to over-estimate the partition function thus underestimate the density. Note that although density values and likelihood values are not directly comparable, we list them together due to the sheer lack of a density model for CIFAR-10.

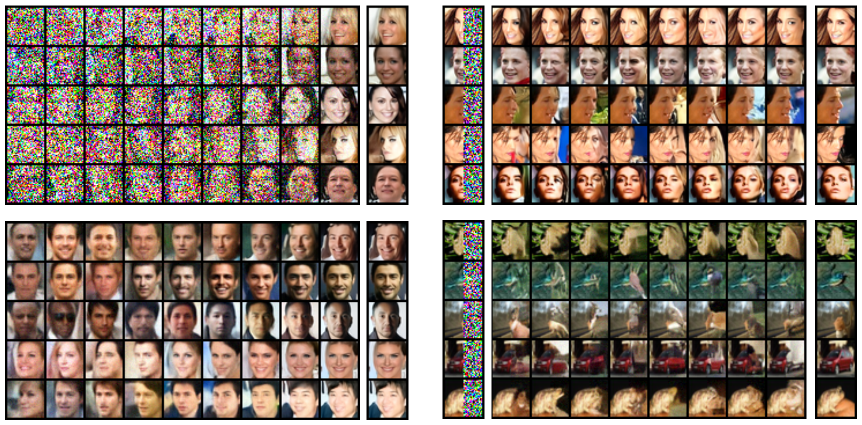

Figure 6.3: Demonstration of the sampling process (left), and image inpainting (right). The sampling process is shown with Gaussian noise (top left), and denoised by single step gradient jump (lower left). The column next to sampling process shows samples after the last denoising jump at the end of sampling. Inpainting results are shown next to initial image (left column) and the ground truth image (right column).

We also report a density of 1.21 bits/dim on MNIST dataset, and we refer readers to (Du & Mordatch 2019) for comparison to other models on this dataset. Note that this number follows a different convention than the number for CIFAR-10, which consider images pixal values to lie between $[0, 255]$. More details on this experiment is provided in the Appendix.

Outlier Detection. (Choi & Jang 2018) and (Nalisnick *et al.* 2019) have reported intriguing behavior of high dimensional density models on out of distribution samples. Specifically, they showed that a lot of models assign higher likelihood to out of distribution samples than real data samples. We investigated whether our model behaves similarly.

Our energy function is only trained outside the data manifold where samples are noisy, so the energy value at clean data points may not always be well behaved. Therefore, we added noise with magnitude σ_0 before measuring the energy value. We find that our network behaves similarly to previous likelihood models, it assigns lower energy, thus higher density, to some OOD samples. We show one example of this phenomenon in Appendix Figure B.7A.

We also attempted to use the denoising performance, or the objective function to perform outlier detection. Intriguingly, the results are similar as using the energy value. Denoising performance seems to correlate more with the variance of the original image than the content of the image.

6.5 Discussion

In this work we provided analyses and empirical results for understanding the limitations of learning the structure of high-dimensional data with denoising score matching. We found that the objective function confines learning to a small set due to the measure concentration phenomenon in random vectors. Therefore, sampling the learned distribution outside the set where the gradient is learned does not produce good result. One remedy to learn meaningful gradients in the entire space is to use samples during learning that are corrupted by different amounts of noise. Indeed, (Song & Ermon 2019) applied this strategy very successfully.

The central contribution of our paper is to investigate how to use a similar learning strategy in EBMs. Specifically, we proposed a novel EBM model, the *Multiscale Denoising Score Matching* (MDSM) model. The new model is capable of denoising, producing high-quality samples from random noise, and performing image inpainting. While also providing density information, our model learns an order of magnitude faster than models based on maximum likelihood.

Our approach is conceptually similar to the idea of combining denoising autoencoder and annealing (Geras & Sutton 2015; Chandra & R. K. Sharma 2014; Q. Zhang & L. Zhang 2018) though this idea was proposed in the context of pre-training neural networks for classification applications. Previous efforts of learning energy-based models with score matching (Kingma & LeCun 2010; Song, Garg, *et al.* 2019) were either computationally intensive or unable to produce high-quality samples comparable to those obtained by other generative models such as GANs. (Saremi, Mehrjou, *et al.* 2018) and (Saremi & Hyvarinen 2019) trained energy-based model with the denoising score matching objective but the resulting models cannot perform sample synthesis from random noise initialization.

Recently, (Song & Ermon 2019) proposed the NCSN model, capable of high-quality sample synthesis. This model approximates the score of a family of distributions obtained by smoothing the data by kernels of different widths. The sampling in the NCSN model starts with sampling the distribution obtained with the coarsest kernel and successively switches to distributions obtained with finer kernels. Unlike NCSN, our method learns an energy-based model corresponding to $p_{\sigma_0}(\tilde{\boldsymbol{x}})$ for a fixed σ_0. This method improves score matching in high-dimensional space by matching the gradient of an energy function to the score of $p_{\sigma_0}(\tilde{\boldsymbol{x}})$ in a set that avoids measure concentration issue.

All told, we offer a novel EBM model that achieves high-quality sample synthesis, which among other EBM approaches provides a new state-of-the art. Compared to the NCSN model, our model is more parsimonious than NCSN and can support single step denoising without prior knowledge of the noise magnitude. But our model performs sightly worse than the NCSN model, which could have several reasons. First, the derivation of Equation 6.6 requires an approximation to keep the training procedure tractable, which could reduce the performance. Second, the NCSN's output is a vector that, at least during optimization, does not always have to be the derivative of a scalar function. In contrast, in our model the network output is a scalar function. Thus it is possible that the NCSN model performs better because it explores a larger set of functions during optimization.

Appendices

Appendix A

Appendix for the SMT

A.1 Online SGD Solution

Some minor drawbacks of the analytic generalized eigendecomposition solution are that: 1) there is an unnecessary ordering among different embedding dimensions, 2) the learned functional embedding tends to be global, which has support as large as the whole manifold and 3) the solution is not online and does not allow other constraints to be posed. In order to solve these issues, we modify the formulation of equation 3.3 slightly with a sparse regularization term on P and develop an online SGD (Stochastic Gradient Descent) solution:

$$\min_P \gamma_0 \|PAD\|_F^2 + \gamma_1 \|PW\|_1, \text{ s.t. } PVP^T = I, \tag{A.1}$$

where γ_0 and γ_1 are both positive parameters, $W = \text{diag}(\langle\alpha\rangle)$ and $\| \bullet \|_1$ is the $L_{1,1}$ norm, V is the covariance matrix of α. The sparse regularization term encourages the functionals to be localized on the manifold.

The iterative update rule for solving equation (A.1) to learn the manifold embedding consists of:

1. One step along the whitened gradient computed on a mini-batch: $P \mathrel{+}= -2\gamma_0\eta_P PADD^TA^TV^{-1}$, where V^{-1} serves to whiten the gradient. η_P is the learning rate of P.

2. Weight regularization: $P = \text{Shrinkage}_{\langle\alpha\rangle,\gamma 1}(P)$, which shrinks each entry in the j^{th} column of P by $\gamma_1\langle\alpha_j\rangle$.

3. Parallel orthogonalization: $(PVP^T)^{-\frac{1}{2}}P$, which is a slight variation to an orthogonalization method introduced by (Karhunen *et al.* 1997).

A.2 Dataset and Preprocessing

The natural videos were captured with a head-mounted GoPro 5 session camera at 90fps. The video dataset we used contains about 53 minutes (about 300,000 frames) of 1080p video while the cameraperson is walking on a college campus. For each video frame, we select the center 1024x1024 section and down-sample it to 128x128 by using a steerable pyramid (Simoncelli & Freeman 1995) to decrease the range of motion in order to avoid temporal aliasing. For each 128x128 video frame, we then apply an approximate whitening in the Fourier domain: For each frame, 1) take a Fourier transform, 2) modulate the amplitude by a whitening mask function $w(u,v)$, 3) take an inverse Fourier transform. Since the Fourier amplitude map of natural images in general follows a $1/f$ law(Field 1987), the whitening mask function is chosen to be the product: $w(u,v) = w_1(u,v)w_2(u,v)$, where $w_1(u,v) = r$ is a linear ramp function of the frequency radius $r(u,v) = (u^2 + v^2)^{\frac{1}{2}}$ and w_2 is a low-pass windowing function in the frequency domain $w_2(u,v) = e^{-(\frac{r}{r_0})^4}$, with $r_0 = 48$. The resulting amplitude modulation function is shown in Supplementary Figure A.1. For a more thorough discussion on this method, please see (Atick & Redlich 1990; Atick & Redlich 1992; Olshausen & Field 1997). We found that the exact parameters for whitening were not critical. We also implemented a ZCA version and found that the results were qualitatively similar.

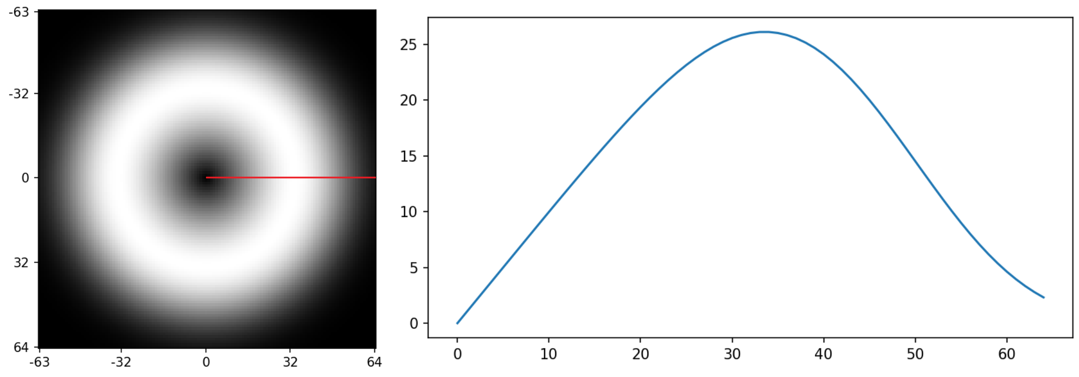

Figure A.1: In this figure we show the whitening mask function $w(u,v)$. Since w is radial symmetric, we show its value with respect to the frequency radius r as a 1-D curve. As we are using 128x128 images, the frequency radius we show is from 0 to 64, which is the Nyquist frequency.

Appendix B

Appendix for Multiscale Denoising Score Matching

B.1 MDSM Objective

In this section, we provide a formal discussion of the MDSM objective and suggest it as an improved score matching formulation in high-dimensional space.

(Vincent 2011) illustrated the connection between the model score $-\nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta)$ with the score of Parzen window density estimator $\nabla_{\tilde{\boldsymbol{x}}} \log(p_{\sigma_0}(\tilde{\boldsymbol{x}}))$. Specifically, the objective is Equation 6.1 which we restate here:

$$L_{SM}(\theta) = \mathbb{E}_{p_{\sigma 0}(\tilde{\boldsymbol{x}})} \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(p_{\sigma_0}(\tilde{\boldsymbol{x}})) + \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 \tag{B.1}$$

Our key observation is: in high-dimensional space, due the concentration of measure, the expectation w.r.t. $p_{\sigma 0}(\tilde{\boldsymbol{x}})$ over weighs a thin shell at roughly distance $\sqrt{d}\sigma$ to the empirical distribution $p(x)$. Though in theory this is not a problem, in practice this leads to results that the score are only well matched on this shell. Based on this observation, we suggest to replace the expectation w.r.t. $p_{\sigma 0}(\tilde{\boldsymbol{x}})$ with a distribution $p_{\sigma'}(\tilde{\boldsymbol{x}})$ that has the same support as $p_{\sigma 0}(\tilde{\boldsymbol{x}})$ but can avoid the measure concentration problem. We call this *multiscale score matching* and the objective is the following:

$$L_{MSM}(\theta) = \mathbb{E}_{p_M(\tilde{\boldsymbol{x}})} \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(p_{\sigma_0}(\tilde{\boldsymbol{x}})) + \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 \tag{B.2}$$

Proposition 1. $L_{MSM}(\theta) = 0 \iff L_{SM}(\theta) = 0 \iff \theta = \theta^*$.

Given that $p_M(\tilde{\boldsymbol{x}})$ and $p_{\sigma 0}(\tilde{\boldsymbol{x}})$ has the same support, it's clear that $L_{MSM} = 0$ would be equivalent to $L_{SM} = 0$. Due to the proof of the Theorem 2 in (Hyvärinen 2005), we have $L_{SM}(\theta) \iff \theta = \theta^*$. Thus, $L_{MSM}(\theta) = 0 \iff \theta = \theta^*$.

□

Proposition 2. $L_{MSM}(\theta) \smile L_{MDSM^*} = \mathbb{E}_{p_M(\tilde{\boldsymbol{x}})q_{\sigma 0}(\boldsymbol{x}|\tilde{\boldsymbol{x}})} \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(q_{\sigma 0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})) + \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2$.

We follow the same procedure as in (Vincent 2011) to prove this result.

$$\begin{aligned}J_{MSM}(\theta) &= \mathbb{E}_{p_M(\tilde{\boldsymbol{x}})} \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(p_{\sigma_0}(\tilde{\boldsymbol{x}})) + \nabla_{\tilde{x}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 \\ &= \mathbb{E}_{p_M(\tilde{\boldsymbol{x}})} \parallel \nabla_{\tilde{x}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 + 2S(\theta) + C\end{aligned}$$

$$\begin{aligned}S(\theta) &= \mathbb{E}_{p_M(\tilde{\boldsymbol{x}})} \langle \nabla_{\tilde{\boldsymbol{x}}} \log(p_{\sigma_0}(\tilde{\boldsymbol{x}})), \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta)\rangle \\
&= \int_{\tilde{\boldsymbol{x}}} p_M(\tilde{\boldsymbol{x}}) \langle \nabla_{\tilde{\boldsymbol{x}}} \log(p_{\sigma_0}(\tilde{\boldsymbol{x}})), \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta)\rangle \, d\tilde{\boldsymbol{x}} \\
&= \int_{\tilde{\boldsymbol{x}}} p_M(\tilde{\boldsymbol{x}}) \langle \frac{\nabla_{\tilde{\boldsymbol{x}}} p_{\sigma_0}(\tilde{\boldsymbol{x}})}{p_{\sigma_0}(\tilde{\boldsymbol{x}})}, \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta)\rangle \, d\tilde{\boldsymbol{x}} \\
&= \int_{\tilde{\boldsymbol{x}}} \frac{p_M(\tilde{\boldsymbol{x}})}{p_{\sigma_0}(\tilde{\boldsymbol{x}})} \langle \nabla_{\tilde{\boldsymbol{x}}} p_{\sigma_0}(\tilde{\boldsymbol{x}}), \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta)\rangle \, d\tilde{\boldsymbol{x}} \\
&= \int_{\tilde{\boldsymbol{x}}} \frac{p_M(\tilde{\boldsymbol{x}})}{p_{\sigma_0}(\tilde{\boldsymbol{x}})} \langle \nabla_{\tilde{\boldsymbol{x}}} \int_{\boldsymbol{x}} p(\boldsymbol{x}) q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x}) d\boldsymbol{x}, \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta)\rangle \, d\tilde{\boldsymbol{x}} \\
&= \int_{\tilde{\boldsymbol{x}}} \frac{p_M(\tilde{\boldsymbol{x}})}{p_{\sigma_0}(\tilde{\boldsymbol{x}})} \langle \int_{\boldsymbol{x}} p(\boldsymbol{x}) \nabla_{\tilde{\boldsymbol{x}}} q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x}) d\boldsymbol{x}, \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta)\rangle \, d\tilde{\boldsymbol{x}} \\
&= \int_{\tilde{\boldsymbol{x}}} \frac{p_M(\tilde{\boldsymbol{x}})}{p_{\sigma_0}(\tilde{\boldsymbol{x}})} \langle \int_{\boldsymbol{x}} p(\boldsymbol{x}) q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x}) \nabla_{\tilde{\boldsymbol{x}}} \log q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x}) d\boldsymbol{x}, \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta)\rangle \, d\tilde{\boldsymbol{x}} \\
&= \int_{\tilde{\boldsymbol{x}}} \int_{\boldsymbol{x}} \frac{p_M(\tilde{\boldsymbol{x}})}{p_{\sigma_0}(\tilde{\boldsymbol{x}})} p(\boldsymbol{x}) q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x}) \langle \nabla_{\tilde{\boldsymbol{x}}} \log q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x}), \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta)\rangle \, d\tilde{\boldsymbol{x}} d\boldsymbol{x} \\
&= \int_{\tilde{\boldsymbol{x}}} \int_{\boldsymbol{x}} \frac{p_M(\tilde{\boldsymbol{x}})}{p_{\sigma_0}(\tilde{\boldsymbol{x}})} p_{\sigma_0}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) \langle \nabla_{\tilde{\boldsymbol{x}}} \log q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x}), \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta)\rangle \, d\tilde{\boldsymbol{x}} d\boldsymbol{x} \\
&= \int_{\tilde{\boldsymbol{x}}} \int_{\boldsymbol{x}} p_M(\tilde{\boldsymbol{x}}) q_{\sigma_0}(\boldsymbol{x}|\tilde{\boldsymbol{x}}) \langle \nabla_{\tilde{\boldsymbol{x}}} \log q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x}), \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta)\rangle \, d\tilde{\boldsymbol{x}} d\boldsymbol{x}\end{aligned}$$

Thus we have:

$$\begin{aligned}L_{MSM}(\theta) &= \mathbb{E}_{p_M(\tilde{\boldsymbol{x}})} \parallel \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 + 2S(\theta) + C \\
&= \mathbb{E}_{p_M(\tilde{\boldsymbol{x}}) q_{\sigma_0}(\boldsymbol{x}|\tilde{\boldsymbol{x}})} \parallel \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 + 2\mathbb{E}_{p_M(\tilde{\boldsymbol{x}}) q_{\sigma_0}(\boldsymbol{x}|\tilde{\boldsymbol{x}})} \langle \nabla_{\tilde{\boldsymbol{x}}} \log q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x}), \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta)\rangle + C \\
&= \mathbb{E}_{p_M(\tilde{\boldsymbol{x}}) q_{\sigma 0}(\boldsymbol{x}|\tilde{\boldsymbol{x}})} \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(q_{\sigma 0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})) + \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 + C'\end{aligned}$$

So $L_{MSM}(\theta) \smile L_{MDSM^*}$.

□

The above analysis applies to any noise distribution, not limited to Gaussian. but L_{MDSM^*} has a reversed expectation form that is not easy to work with. To proceed further we study the case where $q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ is Gaussian and choose $q_M(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ as a Gaussian scale mixture (Wainwright & Simoncelli 2000) and $p_M(\tilde{\boldsymbol{x}}) = \int q_M(\tilde{\boldsymbol{x}}|\boldsymbol{x}) p(\boldsymbol{x}) dx$. By Proposition 1 and Proposition 2, we have the following form to optimize:

$$\begin{aligned}
L_{MDSM^*}(\theta) &= \int_{\tilde{\boldsymbol{x}}}\int_{\boldsymbol{x}} p_M(\tilde{\boldsymbol{x}}) q_{\sigma_0}(\boldsymbol{x}|\tilde{\boldsymbol{x}}) \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(q_{\sigma 0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})) + \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 d\tilde{\boldsymbol{x}} d\boldsymbol{x} \\
&= \int_{\tilde{\boldsymbol{x}}}\int_{\boldsymbol{x}} \frac{q_{\sigma_0}(\boldsymbol{x}|\tilde{\boldsymbol{x}})}{q_M(\boldsymbol{x}|\tilde{\boldsymbol{x}})} p_M(\tilde{\boldsymbol{x}}) q_M(\boldsymbol{x}|\tilde{\boldsymbol{x}}) \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(q_{\sigma 0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})) + \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 d\tilde{\boldsymbol{x}} d\boldsymbol{x} \\
&= \int_{\tilde{\boldsymbol{x}}}\int_{\boldsymbol{x}} \frac{q_{\sigma_0}(\boldsymbol{x}|\tilde{\boldsymbol{x}})}{q_M(\boldsymbol{x}|\tilde{\boldsymbol{x}})} p_M(\boldsymbol{x},\tilde{\boldsymbol{x}}) \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(q_{\sigma 0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})) + \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 d\tilde{\boldsymbol{x}} d\boldsymbol{x} \\
&= \int_{\tilde{\boldsymbol{x}}}\int_{\boldsymbol{x}} \frac{q_{\sigma_0}(\boldsymbol{x}|\tilde{\boldsymbol{x}})}{q_M(\boldsymbol{x}|\tilde{\boldsymbol{x}})} q_M(\tilde{\boldsymbol{x}}|\boldsymbol{x}) p(\boldsymbol{x}) \parallel \nabla_{\tilde{\boldsymbol{x}}} \log(q_{\sigma 0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})) + \nabla_{\tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta) \parallel^2 d\tilde{\boldsymbol{x}} d\boldsymbol{x} \qquad (*) \\
&\approx L_{MDSM}(\theta)
\end{aligned}$$

To minimize Equation (*), we can use the following importance sampling procedure (Russell & Norvig 2016): we can sample from the empirical distribution $p(\boldsymbol{x})$, then sample the Gaussian scale mixture $q_M(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ and finally weight the sample by $\frac{q_{\sigma_0}(\boldsymbol{x}|\tilde{\boldsymbol{x}})}{q_M(\boldsymbol{x}|\tilde{\boldsymbol{x}})}$. We expect the ratio to be close to 1 for the following reasons: Using Bayes rule, $q_{\sigma_0}(\boldsymbol{x}|\tilde{\boldsymbol{x}}) = \frac{p(\boldsymbol{x})q_{\sigma_0}(\tilde{\boldsymbol{x}}|\boldsymbol{x})}{p_{\sigma_0}(\tilde{\boldsymbol{x}})}$ we can see that $q_{\sigma_0}(\boldsymbol{x}|\tilde{\boldsymbol{x}})$ only has support on discret data points $\boldsymbol{x}$, same thing holds for $q_M(\boldsymbol{x}|\tilde{\boldsymbol{x}})$. because in $\tilde{\boldsymbol{x}}$ is generated by adding Gaussian noise to real data sample, both estimators should give results highly concentrated on the original sample point $\boldsymbol{x}$. Therefore, in practice, we ignore the weighting factor and use Equation 6.6. Improving upon this approximation is left for future work.

B.2 Problem with single noise denoising score matching

To compare with previous method, we trained energy-based model with denoising score matching using one noise level on MNIST, initialized the sampling with Gaussian noise of the same level, and sampled with Langevin dynamics at $T = 1$ for 1000 steps and perform one denoise jump to recover the model's best estimate of the clean sample, see Figure B.1. We used the same 12-layer ResNet as other MNIST experiments. Models were trained for 100000 steps before sampling.

B.3 Overfitting test

We demonstrate that the model does not simply memorize training examples by comparing model samples with their nearest neighbors in the training set. We use Fashion MNIST for this demonstration because overfitting can occur there easier than on more complicated datasets, see Figure B.2.

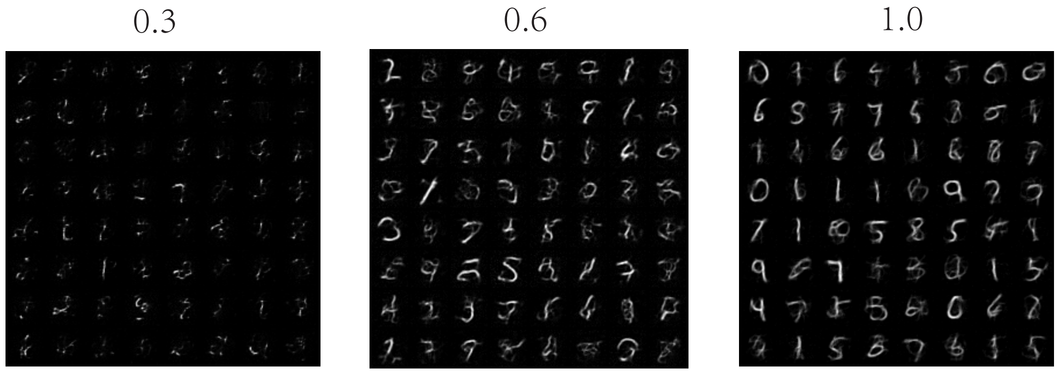

Figure B.1: Denoised samples from energy-based model trained with denoising score matching with single magnitude Gaussian noise on MNIST. Noise magnitude used in training is shown above samples.

B.4 Details on Training and Sampling

We used a custom designed ResNet architecture for all experiments. For MNIST and Fashion-MNIST we used a 12-layer ResNet with 64 filters on first layer, while for CelebA and CIFAR dataset we used a 18-layer ResNet with 128 filters on the first layer. All network used the ELU activation function. We did not use any normalization in the ResBlocks and the filer number is doubled at each downsampling block. Details about the structure of our networks used can be found in our code release. All mentioned models can be trained on 2 GPUs within 2 days.

Since the gradient of our energy model scales linearly with the noise, we expected our energy function to scale quadratically with noise magnitude. Therefore, we modified the standard energy-based network output layer to take a flexible quadratic form (Fan *et al.* 2018):

$$E_{out} = (\sum_i a_i h_i + b_1)(\sum_i c_i h_i + b_2) + \sum_i d_i h_i^2 + b_3 \tag{B.3}$$

where a_i, c_i, d_i and b_1, b_2, b_3 are learnable parameters, and h_i is the (flattened) output of last residual block. We found this modification to significantly improve performance compared to using a simple linear last layer.

For CIFAR and CelebA results we trained for 300k weight updates, saving a checkpoint every 5000 updates. We then took 1000 samples from each saved networks and used the network with the lowest FID score. For MNIST and fashion MNIST we simply trained for 100k updates and used the last checkpoint. During training we pad MNIST and Fashion MNIST to 32*32 for convenience and randomly flipped CelebA images. No other modification was performed. We only constrained the gradient of the energy function, the energy value itself could in principle be unbounded. However, we observed that they naturally stabilize

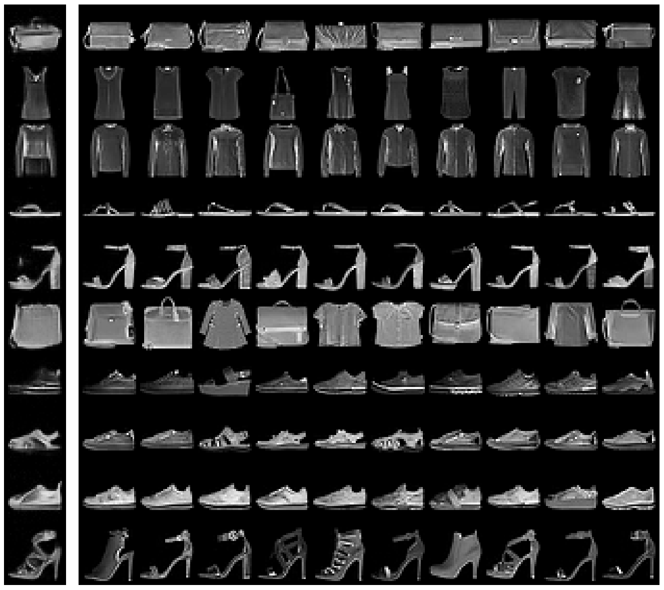

Figure B.2: Samples from energy-based model trained on Fashion MNIST (Left column) next to 10 (L2) nearest neighbors in the training set.

so we did not explicitly regularize them. The annealing sampling schedule is optimized to improve sample quality for CIFAR-10 dataset, and consist of a total of 2700 steps. For other datasets the shape has less effect on sample quality, see Figure B.7 B for the shape of annealing schedule used.

For the Log likelihood estimation we initialized reverse chain on test images, then sample 10000 intermediate distribution using 10 steps HMC updates each. Temperature schedule is roughly exponential shaped and the reference distribution is an isotropic Gaussian. The variance of estimation was generally less than 10% on the log scale. Due to the high variance of results, and to avoid getting dominated by a single outlier, we report average of the log density instead of log of average density.

B.5 Extended Samples and Inpainting results

We provide more inpainting examples and further demonstrate the mixing during sampling process in Figure B.3. We also provide more samples for readers to visually judge the quality of our sample generation in Figure B.4, B.5 and B.6. All samples are randomly selected.

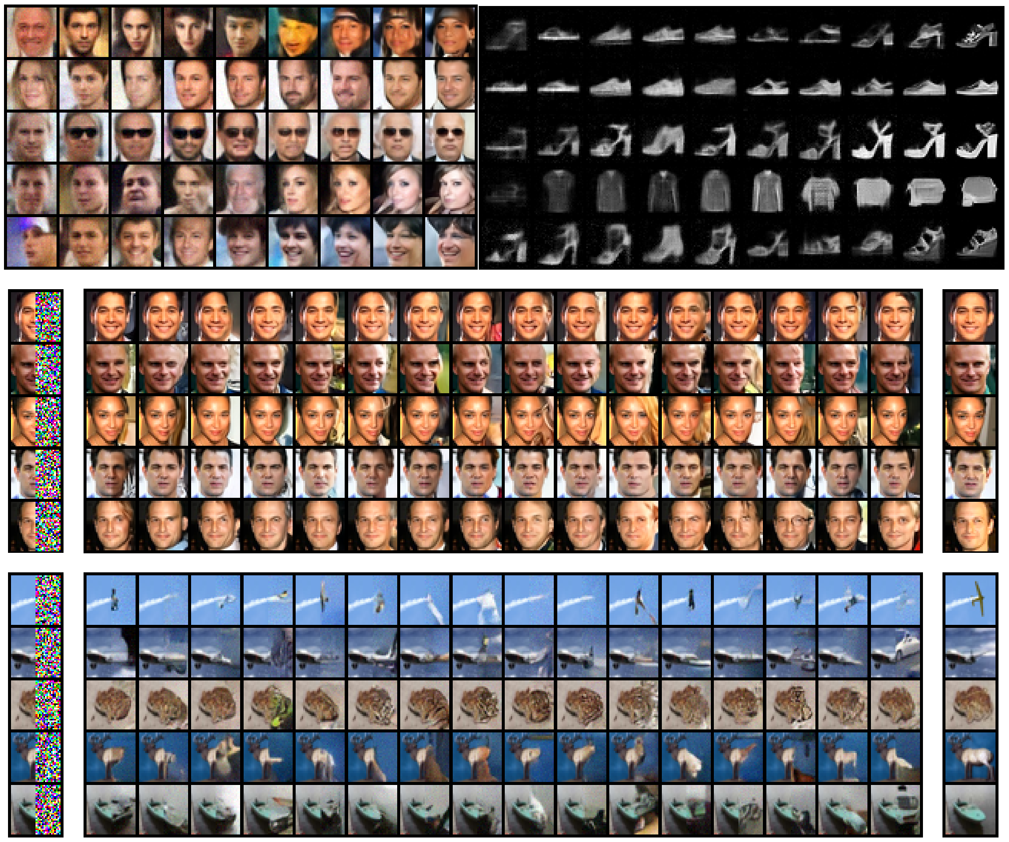

Figure B.3: Denoised Sampling process and inpainting results. Sampling process is from left to right.

B.6 Sampling process and Energy value comparisons

Here we show how the average energy of samples behaves vs the sampling temperature. We also show an example of our model making out of distribution error that is common in most other likelihood based models (Nalisnick *et al.* 2019) Figure B.7.

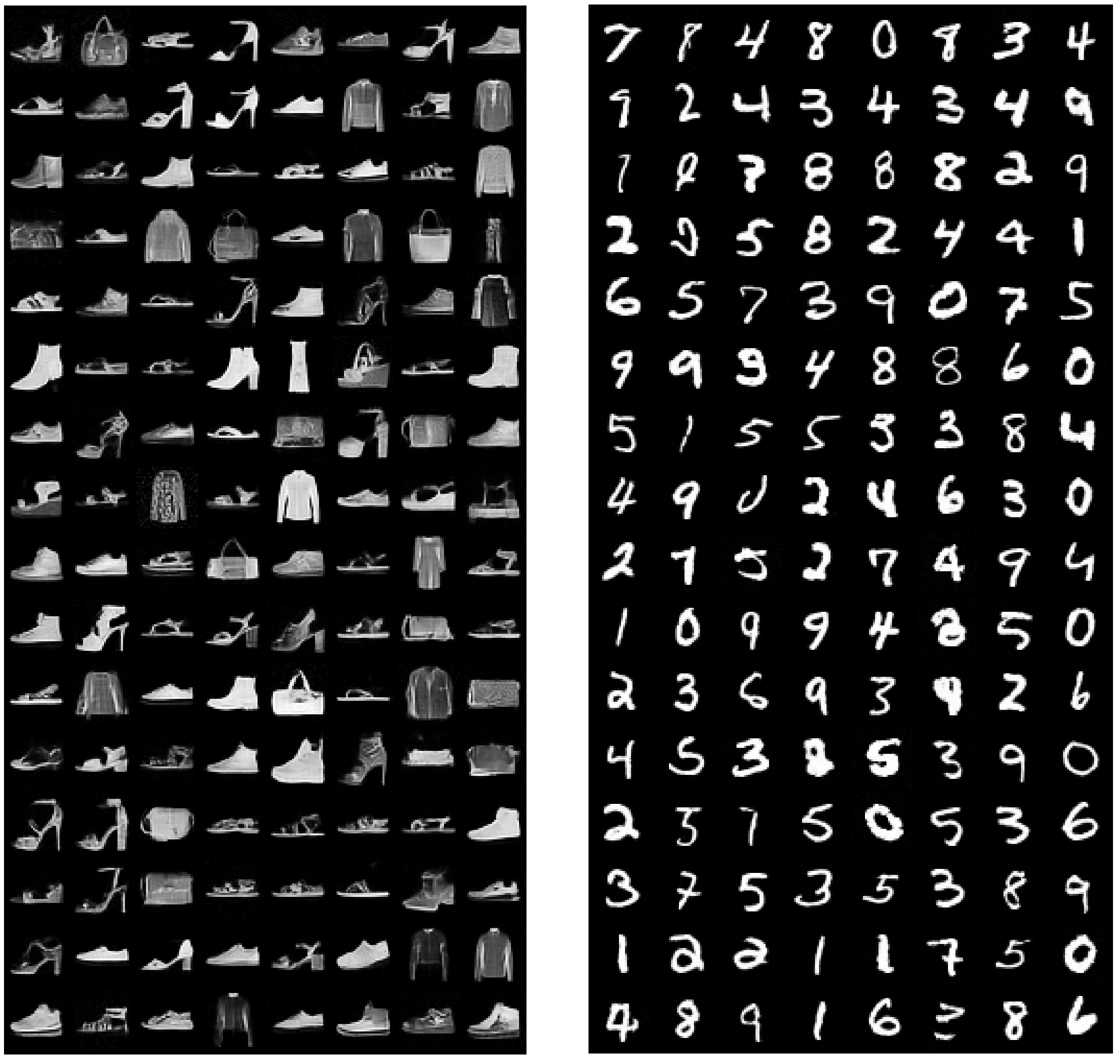

Figure B.4: Extended Fashion MNIST and MNIST samples

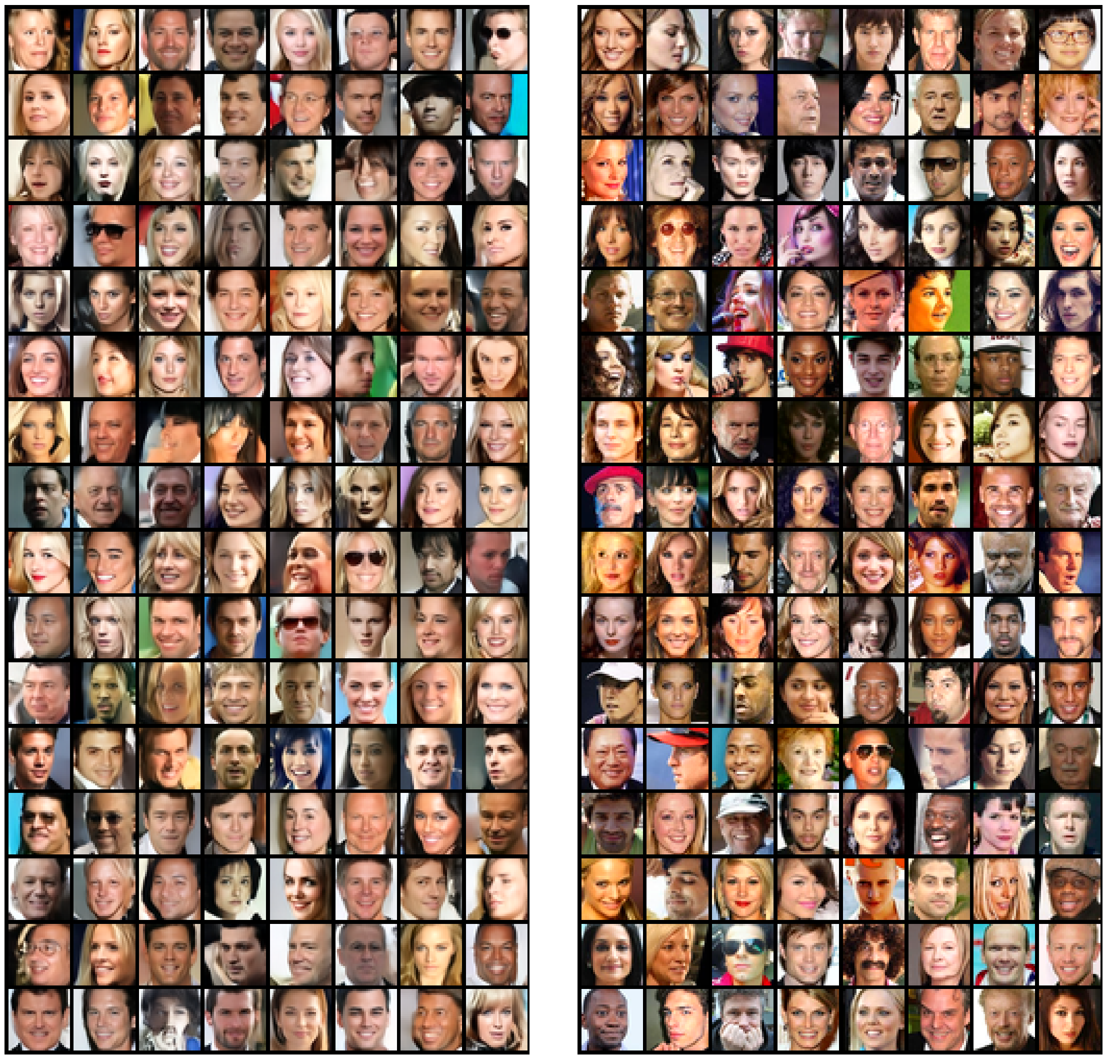

Figure B.5: Samples (left panel) from network trained on CelebA, and training examples from the dataset (right panel).

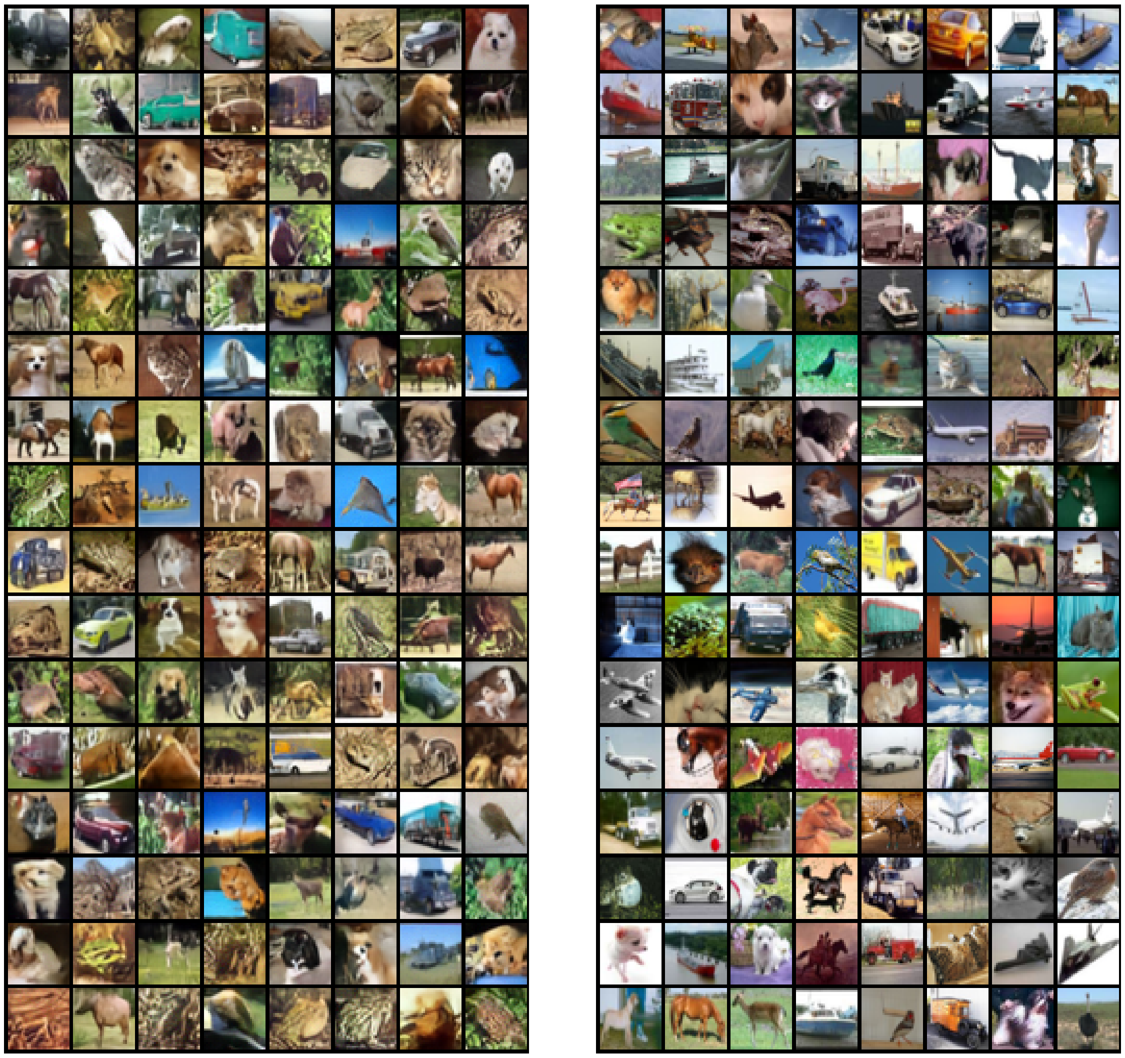

Figure B.6: Samples (left panel) from energy-based model trained on CIFAR-10 next to training examples (right panel).

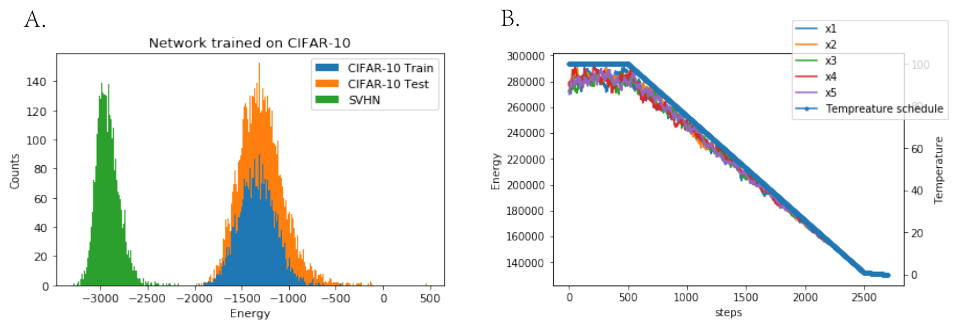

Figure B.7: A. Energy values for CIFAR-10 train, CIFAR-10 test and SVHN datasets for a network trained on CIFAR-10 images. Note that the network does not over fit to the training set, but just like most deep likelihood model, it assigns lower energy to SVHN images than its own training data. B. Annealing schedule and a typical energy trace for a sample during Annealed Langevin Sampling. The energy of the sample is proportional to the temperature, indicating sampling is close to a quasi-static process.

Appendix C

Appendix for Word Factors

C.1 The Word Factor Naming Procedure

In this section we illustrate how the factors are named. A factor is named based on the common aspect of its top-activation words. Specifically, for every factor, we use the word frequency to weight the factor's activation on each word, and take the top words that totally contributing 20% of the total weighted activation. The idea is that a factor should be better represented by words that have strong and obvious activation and show up frequently as well. Usually we get up to 200 words but the number varies from factor to factor. We demonstrate four factors: "national", "mobile&IT", "superlative", "ideology", among which the first two are semantic factors and the latter two are syntactic factors.

"national" Factor. The top 20% activation of the No.781 factor contains about 80 words. They are enumerated as the following:

croatian, american, lithuanian, norwegian, vietnamese, chinese, romanian, bulgarian, indonesian, armenian, serbian, turkish, hungarian, korean, malaysian, italian, austrian, portuguese, mexican, macedonian, german, scottish, albanian, cambodian, bosnian, rican, filipino, lebanese, swedish, estonian, irish, venezuelan, dutch, pakistani, haitian, iranian, peruvian, argentine, malay, colombian, danish, ethiopian, australian, european, chilean, brazilian, israeli, japanese, indian, finnish, singaporean, african, british, nigerian, argentinian, belgian, hispanic, french, cypriot, guatemalan, latvian, russian, welsh, algerian, bolivian, egyptian, moroccan, belarusian, jamaican, icelandic, samoan, uruguayan, georgian, ukrainian, jordanian, flemish, muslim, yugoslav, greek, jewish

By looking into these words, we can easily identify that almost every one of them is related to a specific national meaning, thus this factor can be named "national".

"mobile&IT" Factor. The top 20% activation of the No.296 factor contains about 130 words. They are enumerated as the following:

ipad, iphone, ios, itunes, apple, android, app, ipod, airplay, 3g, 4s, apps, ipads, htc, tablet, *macbook, kindle, galaxy, jailbreak, iphones, netflix, mac, os, touch, nook, skyfire, dock, siri, eris, 4g, thunderbolt, tablets, google, nexus, ipa, barnes, blackberry, sync, devices, hulu, device, ota, amazon, spotify, macworld, retina, wifi, g1, gb, facebook, nfc, syncing, downgrade, protector, iplayer, ipods, 2g, tethered, zune, lte, instagram, s3, kinect, usd, itv, mackintosh, tethering, rooted, tether, shuffle, sansa, garageband, nano, wwdc, smartphones, downgraded, dsi, hotspot, jailbreaking, gps, drm, icloud, smartphone, playbook, casemate, twitter, 3gs, droid, gen., ics, snapchat, multitasking, fuze, stylus, docking, pandora, docks, gadgets, rhapsody, powerbook, tv2, synced, fw, appstore, skype, armband, hd, macbooks, ipo, ssd, evo, aggregator, eyetv, macintosh, g5, folios, steve, sd, gestures, lumia, gen, keynote, shazam, 5g, jellybean, androids, ipcc, cases, magicjack, aria*

By looking into these words, we can easily identify that almost every one of them is related to mobile devices and IT technology, such as apps, brands and etc. Thus we name factor No.296 as “mobile&IT”.

While for the semantic factor, it requires more human knowledge to name a factor given the top activation words, the naming procedure is less subjective if a factor captures syntactic meaning.

“superlative” Factor. For instance, the top 20% activation of the No.337 factor contains about 70 words:

strongest, funniest, largest, longest, oldest, fastest, wettest, tallest, heaviest, driest, sexiest, scariest, coldest, hardest, richest, biggest, happiest, smallest, toughest, warmest, most, brightest, loudest, shortest, costliest, coolest, smartest, darkest, slowest, weakest, greatest, lightest, deadliest, thickest, craziest, sunniest, deepest, quickest, busiest, best, cleanest, saddest, worst, ugliest, densest, sweetest, nicest, wealthiest, hottest, weirdest, dumbest, dullest, poorest, highest, bloodiest, prettiest, grandest, safest, meanest, bravest, strangest, catchiest, dirtiest, proudest, cleverest, purest, quietest, fairest, youngest, sharpest

It’s clear that this factor captures the “superlative” form of different words.

“ideology” Factor. Finally, we demonstrate the top 20% activating about 120 words of the No.674 factor:

nationalism, liberalism, socialism, individualism, capitalism, communism, fascism, anarchism, materialism, humanism, secularism, feudalism, republicanism, modernism, conservatism, rationalism, imperialism, totalitarianism, militarism, multiculturalism, feminism, marxism, racism, ideology, consumerism, pacifism, modernity, romanticism, utilitarianism, fundamentalism, positivism, democracy, authoritarianism, patriotism, unionism, politics, environmentalism, internationalism, paganism, absolutism, nazism, radicalism, commercialism, pluralism, naturalism, colonialism, protestantism, relativism, idealism, egalitarianism, patriarchy, sexism, spiritualism, libertarianism, regionalism, atheism, mysticism, populism, collectivism, ideologies, pragmatism, universalism, isolationism, anarchy, paternalism, antisemitism, protectionism, federalism, transcendentalism, deism, religiosity, elitism, deter-

minism, neoclassicism, postmodernism, centralism, orthodoxy, empiricism, industrialization, catholicism, puritanism, monasticism, separatism, promoted, realism, classicism, altruism, zionism, nihilism, bolshevism, globalization, sectarianism, progressivism, expressionism, orientalism, morality, modernization, barbarism, christianity, occultism, expansionism, slavery, interventionism, traditionalism, tyranny, monogamy, surrealism, abolitionism, primitivism, hedonism, vegetarianism, historicism, chauvinism, humanitarianism, asceticism, dualism, doctrine, unitarianism, misogyny, extremism

The idea that it reflects ideology forms of different concepts is quite obvious once we see the words. So the factor summarized as "ideology".

C.2 The Details of the Non-negative Sparse Coding Optimization

As a convention, all the word vectors used in this word is 300 dimensional and we choose our dictionary to have 1000 word factors.[1] To learn the word factors, we use a typical iterative optimization procedure:

$$\min_A \tfrac{1}{2}\|X - \Phi A\|_F^2 + \lambda \sum_i \|\alpha_i\|_1, \text{ s.t. } \alpha_i \geq 0, \tag{C.1}$$

$$\min_\Phi \tfrac{1}{2}\|X - \Phi A\|_F^2, \ \|\Phi_j\|_2 \leq 1. \tag{C.2}$$

These two optimizations are both convex, we solve them iteratively to learn the word factors: In practice, we use minibatches contains 100 word vectors as X. Optimization C.1 can converge in 500 steps using the FISTA algorithm. We experimented with different λ values from 0.3 to 1, and choose $\lambda = 0.5$ to give results presented in this paper. Once the sparse coefficients have been inferred, we update our dictionary Φ based on Optimization C.2 by one step using an approximate second-order method, where the Hessian is approximated by its diagonal to achieve an efficient inverse (Duchi *et al.* 2011). The second-order parameter update method usually leads to much faster convergence of the word factors. Empirically, we train 200k steps and it takes about 2-3 hours on a Nvidia 1080 Ti GPU.

C.3 The New Word Analogy Tasks Generated

In this section, we would like to demonstrate further that the word factors are more elementary structures than word vectors with Figure C.1 and Figure C.2.

[1] We also experimented other settings and they all lead to qualitatively similar result and discussing the difference is beyond the scope of this work.

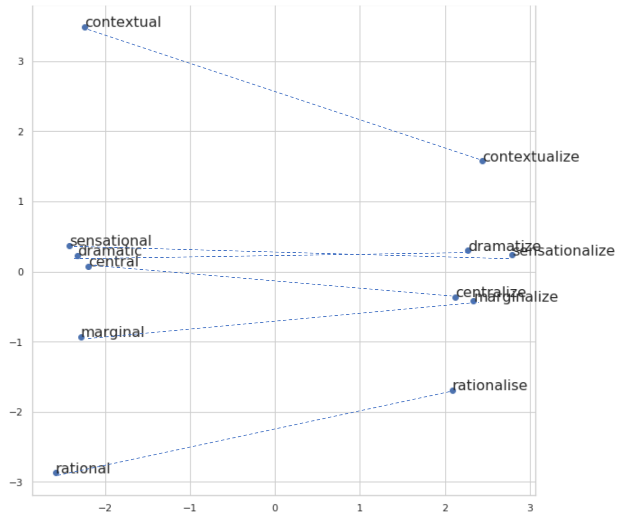

Figure C.1: PCA of the generated adj-to-verb examples.

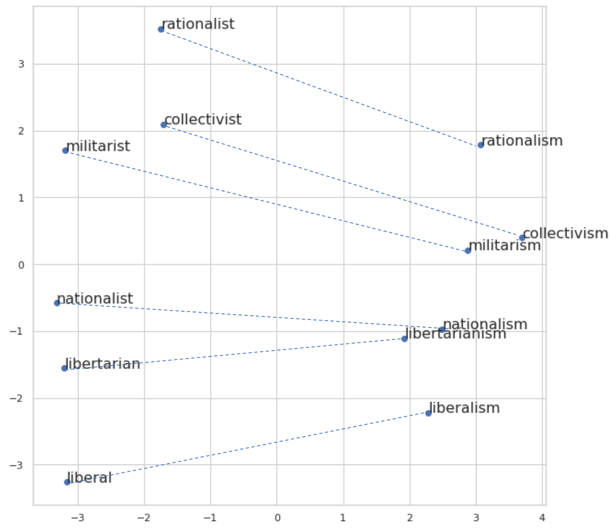

Figure C.2: PCA visualization of a new word analogy task: "ideology", which are automatically generated by the "ideology" word factor.

C.4 Factor Group Co-activation

In Figure C.3 and C.4 we use heat maps to visualize the activations of factors within a group. A heat map shows a fraction of the activation matrix A in Equation 5.8, with each row corresponds to a factor, each column to a word. Therefore, a bright block indicates a high activation on the given word and the dark background means 0 values. It is very clear that factors within a group are often activated together on the same words, forming parallel bright bands across the heat maps.

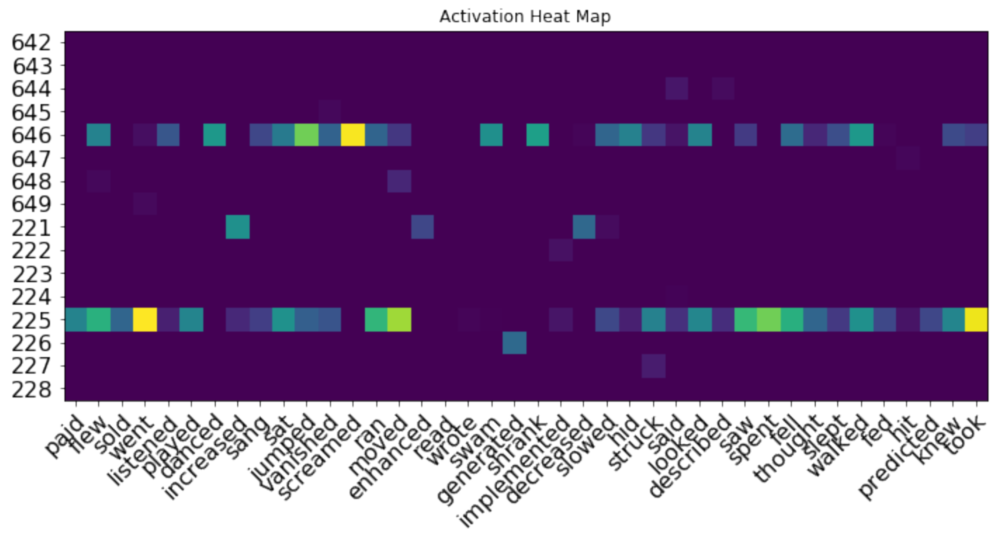

Figure C.3: This figure shows the co-activation pattern of factors in the “past tense” factor group.

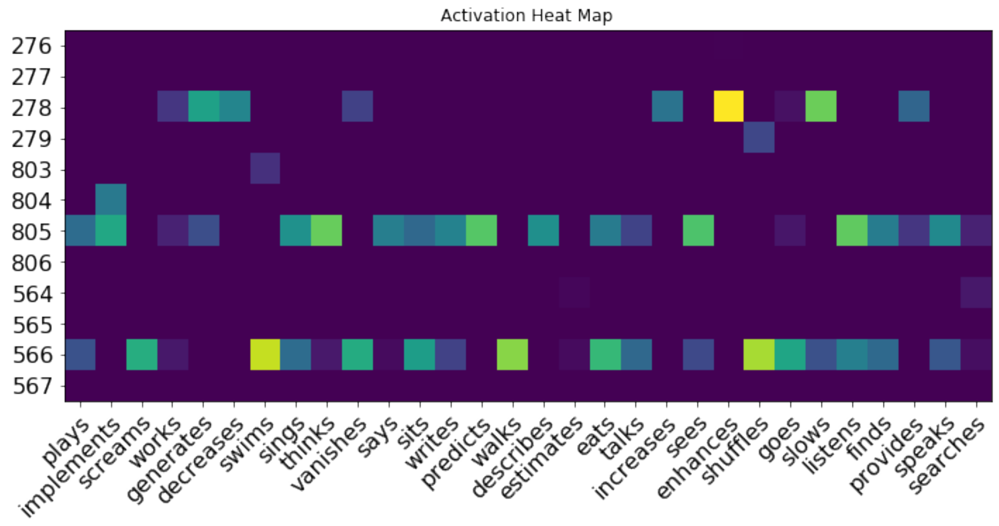

Figure C.4: This figure shows the co-activation pattern of factors in the “singular form” factor group.

Bibliography

1. Allen, C. & Hospedales, T. M. *Analogies Explained: Towards Understanding Word Embeddings* in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA* (2019), 223–231.
2. Andolina, I. M., Jones, H. E., Wang, W. & Sillito, A. M. Corticothalamic feedback enhances stimulus response precision in the visual system. *Proceedings of the National Academy of Sciences* **104,** 1685–1690 (2007).
3. Angelucci, A. & Bullier, J. Reaching beyond the classical receptive field of V1 neurons: horizontal or feedback axons? *Journal of Physiology-Paris* **97,** 141–154 (2003).
4. Atick, J. J. & Redlich, A. N. Towards a theory of early visual processing. *Neural Computation* **2,** 308–320 (1990).
5. Atick, J. J. & Redlich, A. N. What does the retina know about natural scenes? *Neural computation* **4,** 196–210 (1992).
6. Ballé, J., Laparra, V. & Simoncelli, E. P. Density modeling of images using a generalized normalization transformation. *arXiv preprint arXiv:1511.06281* (2015).
7. Barratt, S. & Sharma, R. A note on the inception score. *arXiv preprint arXiv:1801.01973* (2018).
8. Bazell, C. E. The sememe. *Readings in linguistics II,* 329–40 (1966).
9. Beck, A. & Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences* **2,** 183–202 (2009).
10. Behrmann, J., Grathwohl, W., Chen, R. T. Q., Duvenaud, D. & Jacobsen, J. *Invertible Residual Networks* in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA* (2019), 573–582.
11. Belkin, M. & Niyogi, P. *Laplacian eigenmaps and spectral techniques for embedding and clustering* in *Advances in neural information processing systems* (2002), 585–591.
12. Bell, A. J. & Sejnowski, T. J. An information-maximization approach to blind separation and blind deconvolution. *Neural computation* **7,** 1129–1159 (1995).
13. Bellec, G., Kappel, D., Maass, W. & Legenstein, R. Deep rewiring: Training very sparse deep networks. *arXiv preprint arXiv:1711.05136* (2017).

14. Berkes, P., Turner, R. E. & Sahani, M. A structured model of video reproduces primary visual cortical organisation. *PLoS computational biology* **5,** e1000495 (2009).
15. Bojanowski, P., Grave, E., Joulin, A. & Mikolov, T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* **5,** 135–146 (2017).
16. Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A. & Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* **34,** 18–42 (2017).
17. Bruna, J., Szlam, A. & LeCun, Y. Signal recovery from pooling representations. *arXiv preprint arXiv:1311.4025* (2013).
18. Burda, Y., Grosse, R. & Salakhutdinov, R. *Accurate and conservative estimates of MRF log-likelihood using reverse annealing* in *Artificial Intelligence and Statistics* (2015), 102–110.
19. Cadieu, C. F. & Olshausen, B. A. Learning intermediate-level representations of form and motion from natural movies. *Neural computation* **24,** 827–866 (2012).
20. Carlsson, G., Ishkhanov, T., De Silva, V. & Zomorodian, A. On the local behavior of spaces of natural images. *International journal of computer vision* **76,** 1–12 (2008).
21. Chandra, B. & Sharma, R. K. *Adaptive noise schedule for denoising autoencoder* in *International conference on neural information processing* (2014), 535–542.
22. Chen, R. T., Behrmann, J., Duvenaud, D. & Jacobsen, J. Residual Flows for Invertible Generative Modeling. *arXiv preprint arXiv:1906.02735* (2019).
23. Chen, Y., Paiton, D. M. & Olshausen, B. A. The Sparse Manifold Transform. *arXiv preprint arXiv:1806.08887* (2018).
24. Choi, H. & Jang, E. Generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392* (2018).
25. De Silva, V. & Carlsson, G. E. Topological estimation using witness complexes. *SPBG* **4,** 157–166 (2004).
26. De Silva, V. & Tenenbaum, J. B. *Sparse multidimensional scaling using landmark points* tech. rep. (Technical report, Stanford University, 2004).
27. De Valois, R. L., Albrecht, D. G. & Thorell, L. G. Spatial frequency selectivity of cells in macaque visual cortex. *Vision research* **22,** 545–559 (1982).
28. Denton, E. & Birodkar, V. *Unsupervised learning of disentangled representations from video* in *Advances in Neural Information Processing Systems* (2017), 4417–4426.
29. Devlin, J., Chang, M., Lee, K. & Toutanova, K. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* in *NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1* (2019), 4171–4186.

30. DiCarlo, J. J. & Cox, D. D. Untangling invariant object recognition. *Trends in cognitive sciences* **11,** 333–341 (2007).

31. Dinh, L., Krueger, D. & Bengio, Y. *NICE: Non-linear Independent Components Estimation* in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings* (2015).

32. Donoho, D. L. Compressed sensing. *IEEE Transactions on information theory* **52,** 1289–1306 (2006).

33. Dosovitskiy, A. & Brox, T. *Inverting visual representations with convolutional networks* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), 4829–4837.

34. Drucker, H. & Le Cun, Y. *Double backpropagation increasing generalization performance* in *IJCNN-91-Seattle International Joint Conference on Neural Networks* **2** (1991), 145–150.

35. Du, Y. & Mordatch, I. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689* (2019).

36. Duchi, J., Hazan, E. & Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* **12,** 2121–2159 (2011).

37. Elhamifar, E. & Vidal, R. *Sparse manifold clustering and embedding* in *Advances in neural information processing systems* (2011), 55–63.

38. Ethayarajh, K., Duvenaud, D. & Hirst, G. *Towards Understanding Linear Word Analogies* in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (Association for Computational Linguistics, Florence, Italy, July 2019), 3253–3262.

39. Fan, F., Cong, W. & Wang, G. A new type of neurons for machine learning. *International journal for numerical methods in biomedical engineering* **34,** e2920 (2018).

40. Faruqui, M., Tsvetkov, Y., Yogatama, D., Dyer, C. & Smith, N. A. *Sparse Overcomplete Word Vector Representations* in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (Association for Computational Linguistics, Beijing, China, July 2015), 1491–1500.

41. Felzenszwalb, P. F., Girshick, R. B., McAllester, D. & Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence* **32,** 1627–1645 (2010).

42. Field, D. J. Relations between the statistics of natural images and the response properties of cortical cells. *Josa a* **4,** 2379–2394 (1987).

43. Földiák, P. Learning invariance from transformation sequences. *Neural Computation* **3,** 194–200 (1991).

44. Freeman, W. T., Adelson, E. H., *et al.* The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence* **13,** 891–906 (1991).

45. Fyshe, A., Talukdar, P. P., Murphy, B. & Mitchell, T. M. *Interpretable semantic vectors from a joint model of brain-and text-based meaning* in *Proceedings of the conference. Association for Computational Linguistics. Meeting* **2014** (2014), 489.

46. Geman, S. & Geman, D. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence,* 721–741 (1984).

47. Geras, K. J. & Sutton, C. A. *Scheduled denoising autoencoders* in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (2015).

48. Ghahramani, Z. *Unsupervised learning* in *Summer School on Machine Learning* (2003), 72–112.

49. Gilbert, A. C., Zhang, Y., Lee, K., Zhang, Y. & Lee, H. Towards understanding the invertibility of convolutional neural networks. *arXiv preprint arXiv:1705.08664* (2017).

50. Goodfellow, I. *et al. Generative adversarial nets* in *Advances in neural information processing systems* (2014), 2672–2680.

51. Goroshin, R., Bruna, J., Tompson, J., Eigen, D. & LeCun, Y. *Unsupervised learning of spatiotemporally coherent metrics* in *Proceedings of the IEEE international conference on computer vision* (2015), 4086–4093.

52. Goroshin, R., Mathieu, M. F. & LeCun, Y. *Learning to linearize under uncertainty* in *Advances in Neural Information Processing Systems* (2015), 1234–1242.

53. Haarnoja, T., Tang, H., Abbeel, P. & Levine, S. *Reinforcement learning with deep energy-based policies* in *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), 1352–1361.

54. He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 770–778.

55. He, K., Zhang, X., Ren, S. & Sun, J. *Identity mappings in deep residual networks* in *European conference on computer vision* (2016), 630–645.

56. Hénaff, O. J., Goris, R. L. & Simoncelli, E. P. Perceptual straightening of natural videos. *Nature neuroscience* **22,** 984 (2019).

57. Henaff, O., Goris, R. & Simoncelli, E. Perceptual straightening of natural video trajectories. *Journal of Vision* **17,** 402–402 (2017).

58. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. & Hochreiter, S. *Gans trained by a two time-scale update rule converge to a local nash equilibrium* in *Advances in Neural Information Processing Systems* (2017), 6626–6637.

59. Hinton, G. E. Products of experts (1999).
60. Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation* **14,** 1771–1800 (2002).
61. Hinton, G. E. & Roweis, S. T. *Stochastic neighbor embedding* in *Advances in neural information processing systems* (2003), 857–864.
62. Hosoya, H. & Hyvärinen, A. Learning visual spatial pooling by strong PCA dimension reduction. *Neural computation* (2016).
63. Hoyer, P. O. *Non-negative sparse coding* in *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing* (2002), 557–565.
64. Hubel, D. H. & Wiesel, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology* **160,** 106–154 (1962).
65. Huo, X., Ni, X. & Smith, A. K. A survey of manifold-based learning methods. *Recent advances in data mining of enterprise data,* 691–745 (2007).
66. Hupe, J.-M. *et al.* Feedback connections act on the early part of the responses in monkey visual cortex. *Journal of neurophysiology* **85,** 134–145 (2001).
67. Hyvärinen, A. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research* **6,** 695–709 (2005).
68. Hyvärinen, A. & Hoyer, P. Emergence of phase-and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural computation* **12,** 1705–1720 (2000).
69. Hyvärinen, A., Hoyer, P. O. & Inki, M. Topographic independent component analysis. *Neural computation* **13,** 1527–1558 (2001).
70. Hyvärinen, A., Hurri, J. & Väyrynen, J. Bubbles: a unifying framework for low-level statistical properties of natural image sequences. *JOSA A* **20,** 1237–1252 (2003).
71. Hyvärinen, A. & Köster, U. Complex cell pooling and the statistics of natural images. *Network: Computation in Neural Systems* **18,** 81–100 (2007).
72. Hyvärinen, A. & Oja, E. Independent component analysis: algorithms and applications. *Neural networks* **13,** 411–430 (2000).
73. Jones, E., Oliphant, T., Peterson, P., *et al. SciPy: Open source scientific tools for Python* `http://www.scipy.org/`, last accessed on 07/10/2019.
74. Jordan, R., Kinderlehrer, D. & Otto, F. The variational formulation of the Fokker–Planck equation. *SIAM journal on mathematical analysis* **29,** 1–17 (1998).
75. Joulin, A., Grave, E., Bojanowski, P. & Mikolov, T. *Bag of Tricks for Efficient Text Classification* in *EACL* (2016).
76. Karhunen, J., Oja, E., Wang, L., Vigario, R. & Joutsensalo, J. A class of neural networks for independent component analysis. *IEEE Transactions on Neural Networks* **8,** 486–504 (1997).

77. Karklin, Y. & Lewicki, M. S. Emergence of complex cell properties by learning to generalize in natural scenes. *Nature* **457,** 83 (2009).

78. Karklin, Y. & Simoncelli, E. P. *Efficient coding of natural images with a population of noisy linear-nonlinear neurons* in *Advances in neural information processing systems* (2011), 999–1007.

79. Kingma, D. P. & LeCun, Y. *Regularized estimation of image statistics by Score Matching* in *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.* (2010), 1126–1134.

80. Kingma, D. P. & Dhariwal, P. *Glow: Generative flow with invertible 1x1 convolutions* in *Advances in Neural Information Processing Systems* (2018), 10215–10224.

81. Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. *science* **220,** 671–680 (1983).

82. Köster, U. & Hyvärinen, A. A two-layer model of natural stimuli estimated with score matching. *Neural Computation* **22,** 2308–2333 (2010).

83. Krizhevsky, A., Hinton, G., *et al. Learning multiple layers of features from tiny images* tech. rep. (Citeseer, 2009).

84. Krizhevsky, A., Sutskever, I. & Hinton, G. E. *Imagenet classification with deep convolutional neural networks* in *Advances in neural information processing systems* (2012), 1097–1105.

85. Kumar, R., Goyal, A., Courville, A. & Bengio, Y. Maximum Entropy Generators for Energy-Based Models. *arXiv preprint arXiv:1901.08508* (2019).

86. Lawrence, N. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of machine learning research* **6,** 1783–1816 (2005).

87. Le, Q. V. *et al. Building high-level features using large scale unsupervised learning* in *Proceedings of the 29th International Coference on International Conference on Machine Learning* (2012), 507–514.

88. LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M. & Huang, F. A tutorial on energy-based learning. *Predicting structured data* **1** (2006).

89. Lee, A. B., Pedersen, K. S. & Mumford, D. The nonlinear statistics of high-contrast patches in natural images. *International Journal of Computer Vision* **54,** 83–103 (2003).

90. Lee, D. D. & Seung, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature* **401,** 788 (1999).

91. Lee, J. *Introduction to smooth manifolds* ISBN: 978-1-4419-9982-5 (Springer, New York London, 2012).

92. Lee, T. S. & Mumford, D. Hierarchical Bayesian inference in the visual cortex. *JOSA A* **20,** 1434–1448 (2003).

93. Levy, O. & Goldberg, Y. *Linguistic regularities in sparse and explicit word representations* in *Proceedings of the eighteenth conference on computational natural language learning* (2014), 171–180.

94. Liu, S. *et al.* Visual exploration of semantic relationships in neural word embeddings. *IEEE transactions on visualization and computer graphics* **24,** 553–562 (2017).

95. Liu, Z., Luo, P., Wang, X. & Tang, X. *Deep learning face attributes in the wild* in *Proceedings of the IEEE international conference on computer vision* (2015), 3730–3738.

96. Lyu, S. & Simoncelli, E. P. *Nonlinear image representation using divisive normalization* in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (2008), 1–8.

97. Maaten, L. v. d. & Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research* **9,** 2579–2605 (2008).

98. Malo, J. & Gutiérrez, J. V1 non-linear properties emerge from local-to-global non-linear ICA. *Network: Computation in Neural Systems* **17,** 85–102 (2006).

99. Mikolov, T., Chen, K., Corrado, G. & Dean, J. *Efficient Estimation of Word Representations in Vector Space* in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings* (2013).

100. Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C. & Joulin, A. *Pretrained FastText English Word Vectors* `https://fasttext.cc/docs/en/english-vectors.html`, last accessed on 07/10/2019.

101. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. *Distributed Representations of Words and Phrases and their Compositionality* in *NIPS* (2013).

102. Mikolov, T., Yih, W.-t. & Zweig, G. *Linguistic regularities in continuous space word representations* in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2013), 746–751.

103. Miyato, T., Kataoka, T., Koyama, M. & Yoshida, Y. *Spectral Normalization for Generative Adversarial Networks* in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (2018).

104. Morris, P. *Animal Eyes (Oxford Animal Biology Series) by Michael F. Land & Dan-Eric Nilsson* 2012.

105. Mumford, D. & Desolneux, A. *Pattern theory: the stochastic analysis of real-world signals* (CRC Press, 2010).

106. Murphy, B., Talukdar, P. & Mitchell, T. *Learning effective and interpretable semantic models using non-negative sparse embedding* in *Proceedings of COLING 2012* (2012), 1933–1950.

107. Nalisnick, E. T., Matsukawa, A., Teh, Y. W., Görür, D. & Lakshminarayanan, B. *Do Deep Generative Models Know What They Don't Know?* in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* (2019).

108. Neal, R. M. Annealed importance sampling. *Statistics and computing* **11,** 125–139 (2001).

109. Neal, R. M. *et al.* MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo* **2,** 2 (2011).

110. Ng, A. Y., Jordan, M. I. & Weiss, Y. *On spectral clustering: Analysis and an algorithm* in *Advances in neural information processing systems* (2002), 849–856.

111. Ngiam, J., Chen, Z., Koh, P. W. & Ng, A. Y. *Learning deep energy models* in *Proceedings of the 28th international conference on machine learning (ICML-11)* (2011), 1105–1112.

112. Nijkamp, E., Hill, M., Han, T., Zhu, S.-C. & Wu, Y. N. On the Anatomy of MCMC-based Maximum Likelihood Learning of Energy-Based Models. *arXiv preprint arXiv:1903.12370* (2019).

113. Niu, Y., Xie, R., Liu, Z. & Sun, M. *Improved Word Representation Learning with Sememes* in *ACL* (2017).

114. Olshausen, B. A. in *20 Years of Computational Neuroscience* 243–270 (Springer, 2013).

115. Olshausen, B. A. *Highly overcomplete sparse coding* in *Human Vision and Electronic Imaging XVIII* **8651** (2013), 86510S.

116. Olshausen, B. A. *Learning sparse, overcomplete representations of time-varying natural images* in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on* **1** (2003), I–41.

117. Olshausen, B. A. & Field, D. J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* **381,** 607 (1996).

118. Olshausen, B. A. & Field, D. J. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision research* **37,** 3311–3325 (1997).

119. Osindero, S., Welling, M. & Hinton, G. E. Topographic product models applied to natural scene statistics. *Neural Computation* **18,** 381–414 (2006).

120. Ostrovski, G., Dabney, W. & Munos, R. *Autoregressive Quantile Networks for Generative Modeling* in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018* (2018), 3933–3942.

121. Paiton, D. M. *et al. A Deconvolutional Competitive Algorithm for Building Sparse Hierarchical Representations* in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies* (2016), 535–542.

122. Parzen, E. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics* **33,** 1065–1076 (1962).

123. Pennington, J., Socher, R. & Manning, C. *Glove: Global vectors for word representation* in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), 1532–1543.

124. Pennington, J., Socher, R. & Manning, C. D. *Pretrained GloVe Word Embedding* `https://nlp.stanford.edu/projects/glove/`, last accessed on 07/10/2019.

125. Perona, P. Deformable kernels for early vision. *IEEE Transactions on pattern analysis and machine intelligence* **17,** 488–499 (1995).

126. Peters, M. *et al. Deep Contextualized Word Representations* in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (Association for Computational Linguistics, New Orleans, Louisiana, June 2018), 2227–2237.

127. Riesenhuber, M. & Poggio, T. Hierarchical models of object recognition in cortex. *Nature neuroscience* **2,** 1019 (1999).

128. Roweis, S. T. & Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *science* **290,** 2323–2326 (2000).

129. Rozell, C. J., Johnson, D. H., Baraniuk, R. G. & Olshausen, B. A. Sparse coding via thresholding and local competition in neural circuits. *Neural computation* **20,** 2526–2563 (2008).

130. Russell, S. J. & Norvig, P. *Artificial intelligence: a modern approach* (Malaysia; Pearson Education Limited, 2016).

131. Sabour, S., Frosst, N. & Hinton, G. E. *Dynamic routing between capsules* in *Advances in Neural Information Processing Systems* (2017), 3859–3869.

132. Salakhutdinov, R. & Murray, I. *On the quantitative analysis of deep belief networks* in *Proceedings of the 25th international conference on Machine learning* (2008), 872–879.

133. Salimans, T. *et al. Improved techniques for training gans* in *Advances in neural information processing systems* (2016), 2234–2242.

134. Saremi, S. & Hyvarinen, A. Neural Empirical Bayes. *arXiv preprint arXiv:1903.02334* (2019).

135. Saremi, S., Mehrjou, A., Schölkopf, B. & Hyvärinen, A. Deep energy estimator networks. *arXiv preprint arXiv:1805.08306* (2018).

136. Sengupta, A., Tepper, M., Pehlevan, C., Genkin, A. & Chklovskii, D. Manifold-tiling Localized Receptive Fields are Optimal in Similarity-preserving Neural Networks. *bioRxiv,* 338947 (2018).

137. Shan, H. & Cottrell, G. Efficient visual coding: From retina to V2. *arXiv preprint arXiv:1312.6077* (2013).

138. Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A. & Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* **30,** 83–98 (2013).

139. Silva, J., Marques, J. & Lemos, J. *Selecting landmark points for sparse manifold learning* in *Advances in neural information processing systems* (2006), 1241–1248.

140. Simoncelli, E. P. & Freeman, W. T. *The steerable pyramid: A flexible architecture for multi-scale derivative computation* in *Proceedings of the International Conference on Image Processing* **3** (1995), 444–447.

141. Simoncelli, E. P., Freeman, W. T., Adelson, E. H. & Heeger, D. J. Shiftable multiscale transforms. *IEEE transactions on Information Theory* **38,** 587–607 (1992).

142. Skottun, B. C. *et al.* Classifying simple and complex cells on the basis of response modulation. *Vision research* **31,** 1078–1086 (1991).

143. Song, Y. & Ermon, S. Generative Modeling by Estimating Gradients of the Data Distribution. *arXiv preprint arXiv:1907.05600* (2019).

144. Song, Y., Garg, S., Shi, J. & Ermon, S. *Sliced Score Matching: A Scalable Approach to Density and Score Estimation* in *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019* (2019), 204.

145. Subramanian, A., Pruthi, D., Jhamtani, H., Berg-Kirkpatrick, T. & Hovy, E. *Spine: Sparse interpretable neural embeddings* in *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).

146. Tao, T. *Topics in random matrix theory* (American Mathematical Soc., 2012).

147. Tenenbaum, J. B., De Silva, V. & Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *science* **290,** 2319–2323 (2000).

148. Tieleman, T. *Training restricted Boltzmann machines using approximations to the likelihood gradient* in *Proceedings of the 25th international conference on Machine learning* (2008), 1064–1071.

149. Ulyanov, D., Vedaldi, A. & Lempitsky, V. *Deep image prior* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), 9446–9454.

150. Van den Oord, A., Kalchbrenner, N. & Kavukcuoglu, K. *Pixel Recurrent Neural Networks* in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016* (2016), 1747–1756.

151. Van Hateren, J. H. & Ruderman, D. L. Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. *Proceedings of the Royal Society of London B: Biological Sciences* **265,** 2315–2320 (1998).

152. Vaswani, A. *et al. Attention is all you need* in *Advances in neural information processing systems* (2017), 5998–6008.

153. Vershynin, R. *High-dimensional probability: An introduction with applications in data science* (Cambridge University Press, 2018).

154. Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation* **23,** 1661–1674 (2011).

155. Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. *Extracting and composing robust features with denoising autoencoders* in *Proceedings of the 25th international conference on Machine learning* (2008), 1096–1103.

156. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. & Manzagol, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research* **11,** 3371–3408 (2010).

157. Vladymyrov, M. & Carreira-Perpinán, M. Á. *Locally linear landmarks for large-scale manifold learning* in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2013), 256–271.

158. Von Luxburg, U. A tutorial on spectral clustering. *Statistics and computing* **17,** 395–416 (2007).

159. Wainwright, M. J. & Simoncelli, E. P. *Scale mixtures of Gaussians and the statistics of natural images* in *Advances in neural information processing systems* (2000), 855–861.

160. Welling, M. & Teh, Y. W. *Bayesian learning via stochastic gradient Langevin dynamics* in *Proceedings of the 28th international conference on machine learning (ICML-11)* (2011), 681–688.

161. Wieting, J., Bansal, M., Gimpel, K. & Livescu, K. *Towards Universal Paraphrastic Sentence Embeddings* in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016* ().

162. Wikimedia. *Wikimedia Downloads* `https://dumps.wikimedia.org` (2019).

163. Wiskott, L. & Sejnowski, T. J. Slow feature analysis: Unsupervised learning of invariances. *Neural computation* **14,** 715–770 (2002).

164. Wu, Y. N., Si, Z., Gong, H. & Zhu, S.-C. Learning active basis model for object detection and recognition. *International journal of computer vision* **90,** 198–235 (2010).

165. Xie, R., Yuan, X., Liu, Z. & Sun, M. *Lexical Sememe Prediction via Word Embeddings and Matrix Factorization* in *IJCAI* (2017).

166. Zeiler, M. D. & Fergus, R. *Visualizing and understanding convolutional networks* in *European conference on computer vision* (2014), 818–833.

167. Zeiler, M. D., Taylor, G. W. & Fergus, R. *Adaptive deconvolutional networks for mid and high level feature learning* in *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), 2018–2025.

168. Zhai, S., Cheng, Y., Lu, W. & Zhang, Z. *Deep Structured Energy Based Models for Anomaly Detection* in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016* (2016), 1100–1109.

169. Zhang, Q. & Zhang, L. Convolutional adaptive denoising autoencoders for hierarchical feature extraction. *Frontiers of Computer Science* **12,** 1140–1148 (2018).