Title
Structure-Aware Representation Learning and Its Application to Medical Imaging

Author
Wang, Jiayun

Publication Date
2023

Peer reviewed|Thesis/dissertation

Structure-Aware Representation Learning and Its Medical Applications

By

Jiayun Wang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Vision Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Stella X. Yu, Co-chair
Professor Meng C. Lin, Co-chair
Professor Bruno Olshausen
Professor David Whitney

Spring 2023

Structure-Aware Representation Learning and Its Medical Applications

Abstract

Structure-Aware Representation Learning and Its Medical Applications

by

Jiayun Wang

Doctor of Philosophy in Vision Science

University of California, Berkeley

Professor Stella X. Yu, Co-chair

Professor Meng C. Lin, Co-chair

Deep learning models have made remarkable progress in understanding object-centric visual data by focusing on one or more objects present in the scene. However, they may not perform well for scene-centric visual data that includes multiple objects, backgrounds, and other elements in the scene. Furthermore, the low interpretability of deep learning models due to their high complexity may hinder their trustworthiness and practical deployment in the real world.

In this dissertation, we present a structure-aware representation learning method to address these challenges. Rather than solely relying on end-to-end supervised learning, we first use deep learning to learn multi-level representations that explicitly model the structures of individual objects, such as their shape and geometry, and then map the representations to the final prediction with a shallow classifier like a single-layer perceptron. This approach achieves more robust performance for non-object-centric visual data as it explicitly extracts structure-aware representations from the input. Additionally, the method disentangles representation learning and classification. Thus, analyzing the shallow classifier can provide quantitative interpretation of why a prediction is made. We demonstrate the effectiveness of our approach on several standard computer vision benchmark datasets, as well as real-world medical applications like dry eye disease diagnosis, and 3D dental casting.

Part I of the dissertation describes a segmentation-based method to learn instance-level representations that enable us to understand the individual characteristics of objects. We demonstrate its effectiveness with an application to multi-level gland morphology quantification from medical images for disease diagnosis purposes.

Part II presents approaches for learning geometric shape representations from visual data and how such representations can be used for reconstructing 3D shapes. We provide a medical

application of 3D dental casting and jaw reconstruction.

In Part III, based on the instance-level and shape-aware representations from previous parts, we map the representations to the final prediction with a shallow model and show how it could be analyzed and interpreted. We demonstrate an application to demographics prediction from medical images, where we can identify the most relevant features that inform the model's decision and improve its reliability.

To make the proposed method suitable for deployment for practical uses, Part IV introduces how we improve the efficiency of deep learning models using constrained neural optimization. We provide its special cases including orthogonal convolutional neural networks and recurrent parameter generators.

To My Family

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my sincere gratitude to all the individuals and organizations who supported me throughout my PhD journey.

First and foremost, I am deeply grateful to my advisors Stella Yu and Meng Lin for their guidance and unwavering support. They provided me with an exceptional opportunity to work at the intersection of machine learning, computer science, and healthcare, and challenged me with cutting-edge technical innovations that have expanded my horizons beyond what I thought was possible. I cannot thank them enough for their mentorship and trust.

I would also like to thank my dissertation and qualifying committee members, including Bruno Olshausen, David Whitney, Susana Chung, Yi Ma and Ken Polse for their guidance and support throughout this process.

I also want to express my heartfelt appreciation to Yubei Chen for his invaluable insights and encouragement. His guidance has helped me grow both as a researcher and a person. I would also like to thank Andrew Graham for teaching me how to interact and communicate effectively with people from different backgrounds.

My deepest gratitude goes to my lab mates who have been a constant source of inspiration, motivation, and support. Their invaluable feedback, insights, and discussions have significantly enriched my research. Special thanks to Pat Virtue, Ziwei Liu, Zhirong Wu, Qian Yu, Rudrasis Chakraborty, Thao Yeh, Runtao Liu, Zhongqi Miao, Daniel Yeh, Utkarsh Singh, Nils-Steffen Worzyk, Peter Ren, Nils-Steffen Worzyk, Dong-Jin Kim, Sangryul Jeon, Tsung-Wei Ke, Jyh-Jing Hwang, Sacha Hornauer, Ke Wang, Baladitya Yellapragada, Yunhui Guo, Xudong Wang, Tony Lian and Zilin Wang.

I am also grateful for the opportunity to work with exceptional undergraduate students Tejasvi Kothpalli, Wayne Li, Martin Zhai, and Jasmine Li, as well as industry folks Himanshu Arora, Amin Kheradmand, Brian Zhang, Sergei Azernikov, and Chih-Chun Lin. Their diverse perspectives and insights have been invaluable to my research.

I would also like to acknowledge the outstanding team of annotators who provided high-quality annotations for my work on meibography and dry eye diseases: Dorothy Ng, Jessica Vu, Jasper Cheng, Kristin Kiang, Megan Tsiu, Fozia Khan-Ram, April Myers, Shawn Tran, Michelle Hoang, and Zoya Razzak. Their contributions were instrumental in the success of this project.

I am also grateful to Daniel Zeng, Jesse Livezey, Haoran Guo, Ryan Zarcone, Pratik Sachdeva, Jasmine Collins and Kaichun Mo for their help and insightful discussions that have improved my research.

Finally, I would like to acknowledge the funding support from the National Institute of Health, National Science Foundation, Berkeley AI Research Lab, Berkeley Deep Drive, Berkeley Clinical Research Center, Amazon, Sony, Glidewell, and the Roberta Smith Research Fund.

Thank you all for your invaluable contributions and support. My PhD journey would not have been possible without you.

Chapter 1

Introduction

Deep learning has achieved remarkable success in numerous computer vision applications, such as image classification and object detection [113, 307, 200, 183]. Such models now achieve performance that is comparable to or surpasses human abilities on benchmark datasets such as ImageNet [67], where only a dominant object is the primary focus of each image. However, deep learning models trained on ImageNet are still far from achieving human-level performance and cannot generalize well on scene-centric images, where multiple objects exist and the overall scene is the focus.

Additionally, deep learning models are getting larger and larger: the past few years have witnessed the exponential growth of the training data, the size and the computation of such models. For example, foundation models [31], a large deep learning model trained on a vast quantity of data at scale, can be adapted to a wide range of downstream tasks with superior performance at the cost of the gigantic size of the model. For example, an NLP foundation model known as GPT-3 [36] has astonishing 175 billion parameters. With the increased size and complexity, it is also getting increasingly challenging to interpret deep learning models. The difficulty in interpreting and understanding deep learning models may lead practitioners and users to question the model's reliability and trustworthiness, making it challenging to deploy them in real-world settings. A lot of efforts have been techniques are available to interpret deep learning models, such as visualization tools that show how the model processes input data [262, 301] or feature attribution methods that identify the most critical features or components of the input data for making a prediction [27]. However, these techniques typically provide qualitative visualizations to aid model interpretation, and may not be efficient when we need quantitative conclusions or a ranking of the most important decision indicators.

As deep learning models continue to grow in size, there is a growing concern that they may not be accessible to the general public with limited data or computational resources. While these large models can achieve state-of-the-art performance on many tasks, they may be impractical for use in real-world applications that require efficient and timely processing. To address this issue, it is necessary to develop techniques for making these models more efficient, particularly for edge computation and scenarios that require real-time feedback.

However, improving the efficiency of these models without sacrificing performance remains a challenging task.

To address these challenges, we present a structure-aware representation learning method in this dissertation. We learn from domain experts on the specific structures of the data they focus on when performing the targeted task, and incorporate learning representations to reflect such structures in our framework. As specific examples of structure-aware representations, we extract representations at the individual object level from the visual data with segmentation. For each individual object, we make the representations aware of its shape and 3D geometry. The representations augment the texture-focused, categorical representations which are commonly used by existing methods, and make the deep learning methods robust and more generalizable. After obtaining multi-level structure-aware representations, we utilize a shallow classifier, such as a single-layer perception [89], to map these representations to the final prediction. This approach allows us to separate representation learning and classification, enabling a detailed analysis of the decision-making process. Specifically, by examining the shallow classifier, we can obtain a quantitative interpretation of the model's predictions, including the most significant indicators and their corresponding certainty levels. For instance, we can identify the key features of the data that contribute to a particular prediction and assess the model's confidence in its prediction.

We further improve the efficiency of the proposed representation learning methods. We show how we can make models more efficient, including reducing training data and model training time, to get them ready for practical use [304, 303, 44]. Specifically, we propose to use constrained neural optimization to improve the efficiency of deep neural networks. This approach includes several specialized cases, such as orthogonal convolutional neural networks and recurrent parameter generators, which achieve higher efficiency on less training data compared to existing methods.

We demonstrate the effectiveness of our approach on standard computer vision benchmark datasets, as well as several real-world medical applications, including the diagnosis of dry eye diseases and 3D dental casting.

In the remaining part of the introduction, we first explain the important concept of representation learning and how it helps achieve robust performance and interpretability. We then give an overview of the dissertation, where the details of the proposed structure-aware representation learning method are introduced.

1.1 Representation Learning

What is representation? David Marr [1] defined it as a "formal system for making explicit certain entities or types of information, together with a specification of how the system does this". We are interested in representation learning, which could bring the representations or the structures of the data explicit.

For example, in the context of image processing, according to Marr's theory [195], representation learning involves learning multi-level representations of the image, starting from

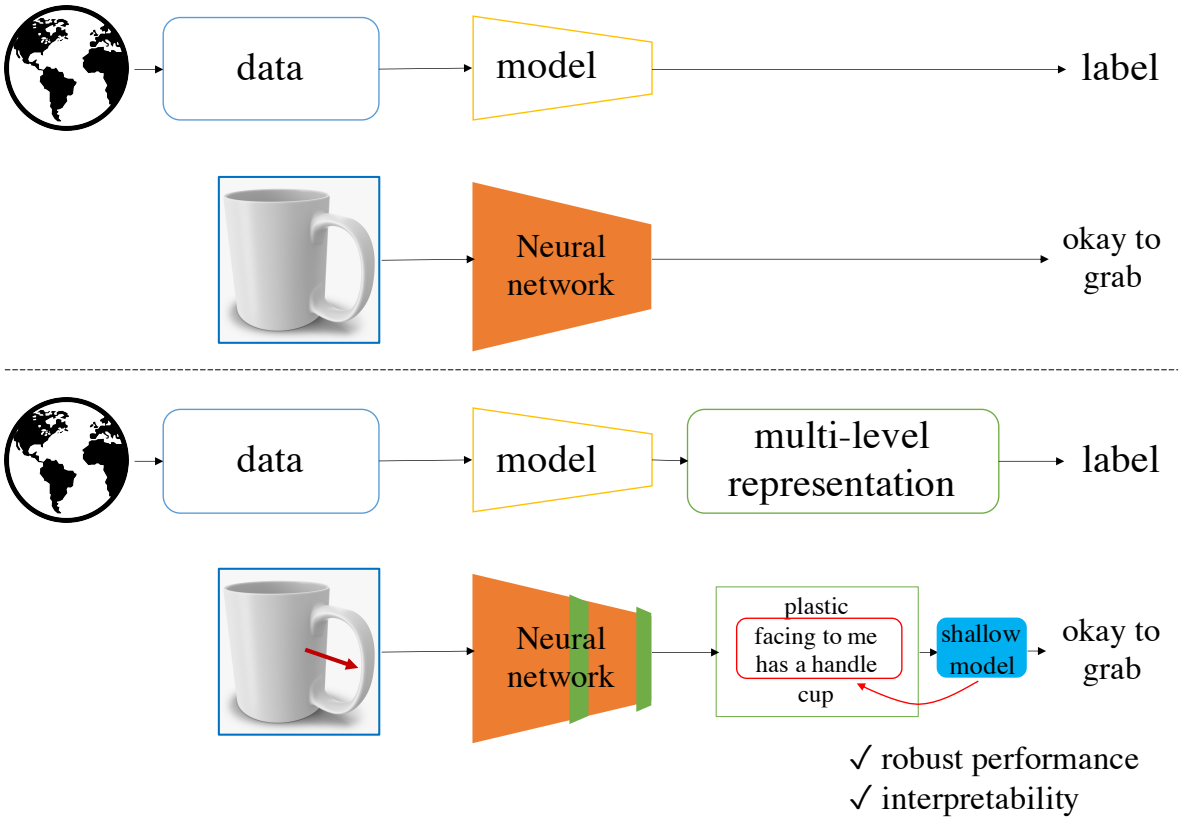


Figure 1.1: The comparison of end-to-end learning (**Upper**) and multi-level representation learning (**Lower**). While end-to-end learning also extracts representations from the data, they are only used to predict the final label, and thus such methods suffer issues like fragile performance and cannot generalize well. The multi-level representation learning extracts multi-level information from different layers of the network and achieves robust performance. We then map the representations to the final label with a shallow classifier. Analyzing the classifier could provide insights on why the prediction is made.

low-level features such as edges, and progressing to mid-level features such as geometry and object segmentation, and finally to the high-level categorical information, for example, the foreground object is a cup, and the function is to grab the cup. By extracting multi-level features, representation learning makes it easier for machine learning algorithms to make accurate predictions.

One may question why we do not rely on labels and perform end-to-end learning. We compare the end-to-end learning and representation learning in Fig. 1.1. End-to-end learning involves collecting data from the world, working on data curation and annotation, curating and annotating the data, and feeding it to a machine learning model, such as a deep neural network. Final task labels are the only output. While representations exist as the output of

each layer of the network, they are used to predict the sole final label and thus may not be robust and generalizable.

However, in multi-level representation learning, or multi-level feature learning, rather than learning the final labels only, we can extract multi-level representations from different layers of the network, such as the low-level material, mid-level pose and geometric structure like it has a handle, and high-level semantic category. We then use a small shallow model, like a single-layer perceptron, to map the representations to the final prediction. Analyzing the shallow model reveals the most important indicator of why the decision is made. Thus, the multi-level representation learning method could achieve robust performance and high interpretability for practical tasks.

In this dissertation, we use an application of quantitative and interpretable dry eye diagnosis to guide the study of multi-level interpretable representation learning. We provide a summary of the proposed method for the case study with comparisons to end-to-end learning in Fig. 1.2.

1.2 Dissertation Overview

We summarize the dissertation in Fig. 1.3 with details below:

Part I: Learning instance-level representation with segmentation. Existing methods focus on learning image-level representations. In addition to high-level semantics, we, however, also learn the mid-level organization of objects to understand the individual characteristics of objects. In Chapters 2 and 3, we move from learning coarse, image-level representations to fine-grained, instance-level representations for meibomian glands depicted in meibography images, which plays a significant role in automated dry eye disease diagnosis. Specifically, to learn instance-level representations, we use an instance-segmentation that leverages the power of pixel representations, rather than commonly-used detect-then-segment methods like Mask-RCNN [115], which fail to segment glands at an instance level. We show the method based on pixel representations has robust segmentation performance as it adapts to the dense and tilted structures of glands.

Part II: Learning 3D-geometry-aware representation. Chapters 4 and 5 describe how we learn equivariant representations from 3D visual data that disentangle semantics and geometric shapes, as well as how they can be used for reconstructing 3D shapes. In Chapter 6, we show a healthcare application in 3D dental casting and jaw reconstruction.

Part III: Interpretable models from structure-aware representations with applications to demographics prediction. Building on the learned representations described in the previous parts (i.e. individual-object-aware and 3D-geometry-aware representations), we can map these representations to the final output with a shallow model and gain insights into why a particular prediction was made (Fig. 1.1**L**). This interpretability is crucial for understanding the model's decision-making process and for building trust in the model's outputs. By analyzing the learned representations and their corresponding contributions to the final prediction, we can identify the most relevant features that inform the model's

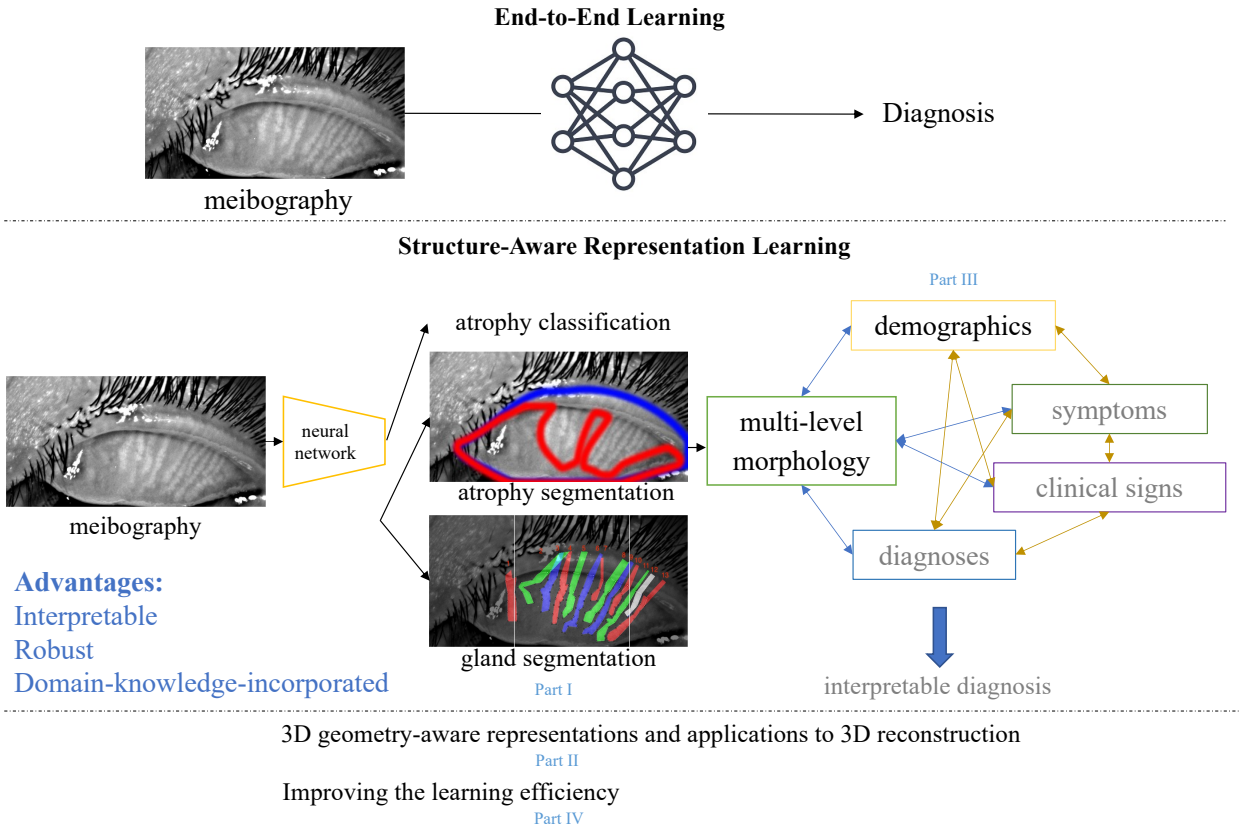


Figure 1.2: As a case study, we apply the proposed structure-aware representation methods to quantitative and interpretable dry eye diagnosis. We compare end-to-end learning (**Upper**) with the proposed structure-aware representation learning (**Middle**), and provide possible improvements as detailed in other parts of the paper (**Lower**). Our method differs from end-to-end learning which has low interpretability. We incorporate domain knowledge by learning multi-level representations that matter for interpretability. Our method also achieves more robust performance as we explicitly extract multi-level representations instead of solely relying on image-level representations. This enables an interpretable understanding of the disease as well as diagnosis. Note that this dissertation only includes interpretable demographics prediction, with the work of other predictions being in press.

decision, and improve the model's accuracy and reliability. In Chapter 7, we demonstrate an application of the interpretable model to predict demographics from meibography and to quantitatively identify the most important indicators for each demographic prediction.

Part IV: Improving learning efficiency. In this part, we focus on improving the efficiency of deep representation learning models. Specifically, we present constrained neural optimization (details in Fig. 1.4) with its specific cases of orthogonal convolutional neural networks in Chapter 8 and recurrent parameter generators in Chapter 9. We show how

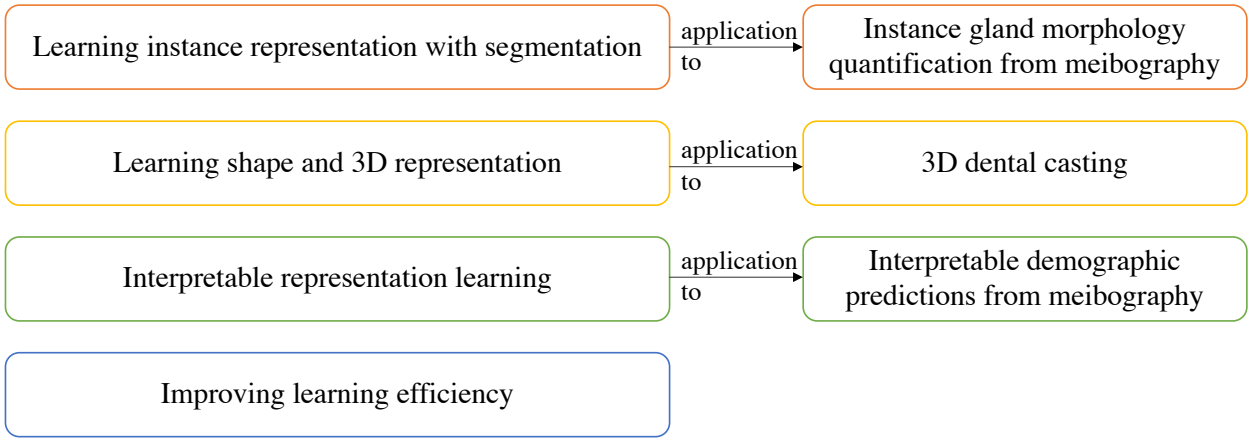


Figure 1.3: Overview of the dissertation

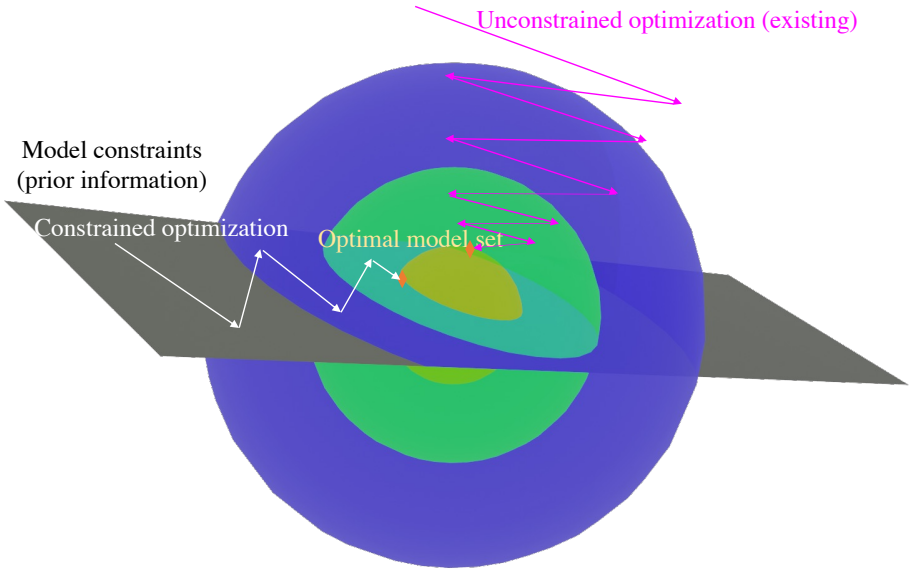


Figure 1.4: We achieve efficient learning via constrained neural optimization. Common and existing method use unconstrained optimization and search in the whole space, which may take quite a few steps to optimize. We consider constraints for models, or prior information, and search in a subspace. With constrained optimization, it takes fewer steps to achieve optimality as the search space is smaller, thus we could have faster training time. Additionally, as the number of model parameters is smaller with the constraint, the evaluation or inference time is also reduced. We consider two different constraints under this framework: orthogonality (Chapter 8) and recurrence (Chapter 9).

we can improve learning efficiency by reducing training data and time with minimal loss of performance.

Summary. Finally, in Chapter 10, we summarize the contributions of this dissertation and discuss several future directions.

Part I

Learning Instance-Level Representation

Chapter 2

Learning Coarse, Image-Level Representations of Meibomian Glands

2.1 Introduction

Dry eye disease is a multifactorial ocular surface disorder and is very common among adults. Meibomian glands (MGs) are believed to play a critical role in ocular surface health by secreting lipids into the tear film to slow down the rate of aqueous evaporation and minimize symptoms of dry eye. Dysfunction of MGs is the most frequent cause of dry eyes [28]. The ability to visualize the glands and to monitor their changes with time or treatment is important for evaluating the risk of meibomian gland dysfunction (MGD) and dry eye diseases. Meibography, which is a photo documentation of MGs in the eyelids using either trans-illumination or infrared light, is commonly used in dry eye clinics for the diagnosis, treatment, and management of MGD.

The measurement of glandular loss is of significant clinical impact for the diagnosis of MGD [13, 227]. Percent MG atrophy, the ratio of gland loss area to the total eyelid area, may be an important clinical factor for assessing MGD severity. Using a standardized MG atrophy grading scale, the percent atrophy can be classified based on severity [12, 235]. In this paper, a previously published clinical-grading system [231] (i.e., meibosocre) was applied (Table 2.1). Figure 2.1 depicts some sample meibography images with varying percent atrophy and corresponding meiboscores to help readers gain insights on the relationship between the percent atrophy and meiboscore. Currently, clinicians subjectively estimate the degree of MG atrophy severity. They assign a severity score after grossly estimating the relative ratio between MG atrophy area and total eyelid area. This subjective assessment has several limitations: (1) it may have high inter- and intra-observer variability and low repeatability [215, 236]; (2) it is based on qualitative judgments, therefore lacking quantitative evaluations to accurately track longitudinal changes; and (3) it may take a longer time and more costly to manually process a large number of images.

Recent advances in deep learning [162, 218, 47], a particular form of artificial intelligence

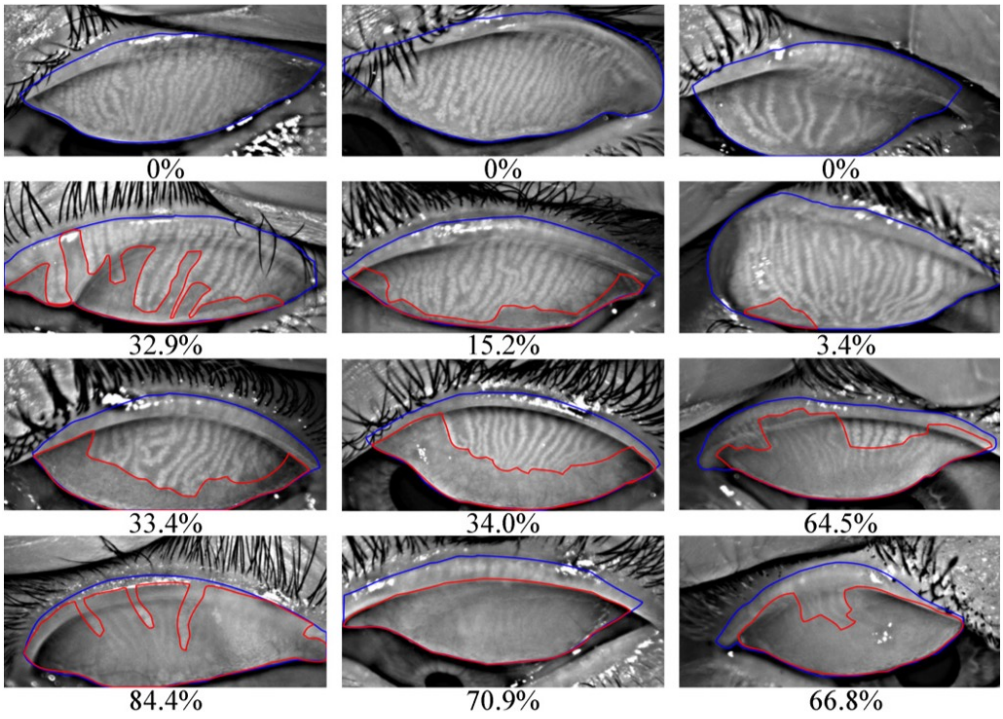


Figure 2.1: Meibography images with ground-truth percent atrophies (%) and ground-truth meiboscores. Rows 1-4 refer to images with meiboscores of grade 0-3, respectively. Given a meibography image, the area of gland atrophy (marked in red) and eyelid (marked in blue) are compared by our deep learning algorithm to estimate the percent atrophy A (Equation 2.1), and are further converted to meiboscore according to the criteria in Table 2.1.

Table 2.1: Percent MG atrophy to meiboscore conversion criteria

Percent MG atrophy / %	Meiboscore
0	0
0 – 33%	1
33% - 66%	2
66%	3

(AI), show the ability of deep neural networks to learn predictive features directly from a large dataset of labeled images, without explicitly specifying rules or features. Additionally, deep learning has shown great success in medical imaging, such as diabetic retinopathy [288, 211, 346], breast cancer [183, 76], melanoma [78], and others [180, 54]. It is of interest to use deep learning methods to benefit the process of evaluating atrophy in meibography

images. Specifically, clinicians can use such methods to automatically segment the eyelid and atrophy areas in meibography images, and then compute the percent atrophy, for the purpose of evaluating MG atrophy. Therefore, an automated method will potentially provide valuable and timely information on MG atrophy.

This study aimed primarily to develop and validate an automated deep learning system for evaluating MG atrophy severity from meibography images with clinician-verified annotations of eyelid and atrophy areas. Additionally, the performance of human clinicians and deep learning algorithm in determining the atrophy severity in meibography images were compared.

2.2 Methods

2.2.1 Development and Evaluation Dataset

This study was based on the utilization of a meibography image dataset with clinician-verified annotations of eyelid and atrophy areas, for deep learning algorithm development and evaluation.

Subject recruitment and demographics. Adult human subjects (ages 18 year or greater) were recruited from the University of California, Berkeley (UCB) campus and surrounding community for a single-visit ocular surface evaluation during the period from 2012 to 2017. During the visits, meibography images of the upper and lower eyelids for both eyes were captured with the OCULUS Keratograph 5M (OCULUS, Arlington, WA), a clinical instrument that uses an infrared light with wavelength 880 nm for MG imaging [347]. During image captures, the same testing conditions were kept (i.e., the ambient light was off with the subject's head positioned on the chin rest and forehead strap). [340] showed that MG contrast in meibography captured using the same instrument was repeatable and invariant to ambient light conditions and head poses. Only upper eyelid images were used in this study. A total of 775 images were collected and pre-screened to rule out images that did not capture the entire upper eyelid (69 images or 8.90%). Examining clinicians assigned an MG atrophy severity score during the exam using the meiboscore scale in Table 2.1, which was previously defined [231]. The meiboscores assigned during the examination were referred to as "clinical meiboscore" and were assigned by multiple clinicians. All clinicians were masked from the subject's ocular surface health status. The most experienced clinician (TNY) also provided a separate set of the clinical meiboscore data for evaluation purposes (i.e., comparing scores among group clinicians, the lead clinical investigator (LCI) and our machine learning algorithm against the ground-truth data generated by the machine algorithms). Subject demographics can be found in Table 2.2.

Data annotations. A team of trained individuals labeled and measured the total eyelid and atrophy regions using the polygon tool in Fiji (ImageJ version 2.0.0-rc-59)[246]. For labeling the total eyelid region, the upper border was defined at the MG orifices, the lower border was set at the edge of proximal tarsal plate, and the horizontal borders were where

Table 2.2: Subject demographics and meiboscores of the meibography image datasets

	Development		Evaluation
	Train	Validation	
Images (No.)	398	99	209
Patient Demographics			
Unique individuals (No.)	308	77	191
Age (average ± SD)	25.5 ± 10.9	27.0 ± 12.6	26.4 ± 11.6
Female/ total patients	63.5%	66.6%	68.3%

Atrophy Severity Distribution	Development				Evaluation	
	Train		Validation			
	No.	%	No.	%	No.	%
Meiboscore 0	73	18.3%	18	18.2%	38	18.20%
Meiboscore 1	267	67.1%	67	67.7%	142	67.90%
Meiboscore 2	53	13.3%	13	13.1%	27	12.90%
Meiboscore 3	5	1.3%	1	1.0%	2	1.0%

the top and bottom borders intersected. For labeling the total atrophy region, the upper border was drawn at the proximal ends of normal glands, the lower border was at the edge of proximal tarsal plate, and the horizontal borders were where the upper and lower borders intersected. Portions of glands that appeared atrophied (e.g., fainter, thinner) compared to other glands on the same eyelid were included in the atrophy region. The areas measured for the regions of interest were captured by selecting "Analyze¿Measure" when the regions of interest were active. Final annotations were verified by an LCI (TNY) before they were made available for the machine learning algorithm to minimize variability in the ground-truth data. From the fine-grained MG atrophy and total eyelid area annotation masks, these ground-truth annotated data were used to calculate the percent atrophy, which was then converted to meiboscore according to Table 2.1, for generating both "ground-truth percent atrophy" and the "ground-truth meiboscore". Figure 2.1 depicts examples of atrophy- and eyelid-area annotations, along with corresponding ground-truth percent atrophy and ground-truth meiboscores. Algorithms were considered to achieve 100% accuracy if they predicted results exactly the same as the ground-truth annotations. Note that machine learning systems can "predict" the MG atrophy region and percent atrophy from an meibography image not seen in the training phase. Machine predictions are different from medical predictions, which usually refer to predicting the future status of a disease or condition.

Data allocations. All meibography images were randomly allocated into two sub-datasets: development and evaluation datasets. The former was used for developing the deep learning algorithm, while the latter was for evaluating the performance of the algorithm. The percent atrophy distributions of the two datasets were very similar as shown in Figure

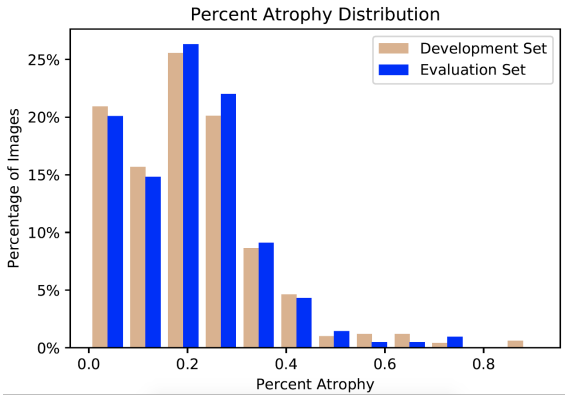


Figure 2.2: Percent atrophy distribution of the development and evaluation datasets. The distributions of the two sets are similar, indicating the scenario differences between training and evaluation are minimized.

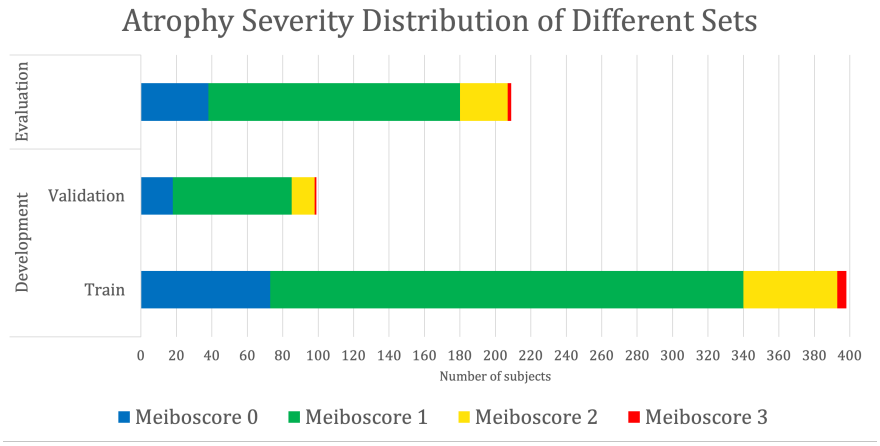


Figure 2.3: Data composition by atrophy severity. The meiboscore distributions of the train, validation and evaluation sets are similar, indicating the scenario differences among training, validation and evaluation are minimized.

2.2. This minimized the scenario differences between training and evaluation. For algorithm development, the development dataset was further divided randomly into 2 subsets: a train set and a validation set. The images in the train set were used to train the deep learning model, while the validation set was used for tuning the model hyperparameters (e.g. network architectures, learning rate, etc.). The evaluation dataset, which did not have any overlapped image with the development dataset, was evaluated using the model that achieved the best performance from the validation set. The patient demographics and atrophy severity of the development and evaluation datasets can be found in Table 2.2 and Figure 2.3.

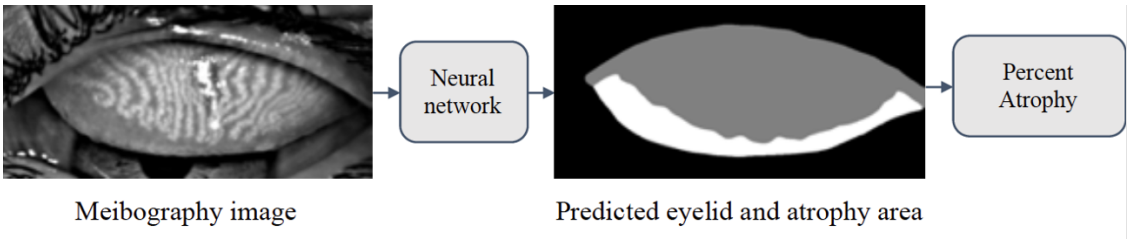


Figure 2.4: The pipeline to evaluating meibography atrophy. The network aims to predict the atrophy area (white part in the right image) and the total eyelid area (white and grey part in the right image). Based on the predicted mask, the percent atrophy can be calculated.

2.2.2 Algorithm Design and Training

In computer vision, image segmentation is the process of partitioning an image into multiple segments [223, 191]. Earlier methods first extracted hand-engineered features of the image, and then used the features to classify pixels independently [264, 132, 276]. More recently, deep learning methods incorporated feature extraction and classification together into a unified framework, and achieved the state-of-the-art in image segmentation [47, 20]. A deep learning algorithm was built upon the pyramid scene parsing network (PSPNet) [352] to segment the atrophy and eyelid region of a given meibography image, and then the percent atrophy was calculated (Figure 2.4).

The input to the neural network was a meibography image. The network could be considered as several stages of computations, parameterized by millions of parameters, and mapped the input image to the output segmentation masks of MG atrophy and eyelid area. Additionally, the network generated a vector indicating if MG atrophy existed in a given meibography image. The auxiliary output helped improve the segmentation performance by providing additional information. The final optimization goal was to have both correct segmentation and atrophy existence vector prediction.

Parameters of a neural network were determined by training the network on the development dataset. The network was repeatedly given images with known ground truth (the segmentation masks of atrophy and eyelid area, as well as atrophy existence vectors in our model). The model predicted the segmentation masks and vector of the given meibography image, and adjusted its parameters over the training process to make the predictions increasingly more similar to the ground truth. The parameters of the model were optimized using stochastic gradient descent [32] and the model performance was evaluated by the validation set every epoch. When atrophy area and eyelid area were predicted, the percent atrophy $\mathcal{A}$ using the following equation can be calculated as follows:

$$\mathcal{A} = \frac{\text{area of atrophy}}{\text{area of eyelid}} \tag{2.1}$$

A hyperparameter search was performed on the network architectures, data augmentation techniques, learning rate, auxiliary loss ratio and learning rate decreasing policy. The

hyperparameters, which attained the best performance over the validation set, was selected. The convolutional neural networks were developed using the PyTorch[229] deep learning framework. The proposed networks were repeatedly trained and evaluated on two NVIDIA GeForce GTX 1080 GPUs with NVIDIA CUDA v8.0 and NVIDIA cuDNN v5.1 acceleration.

2.2.3 Evaluation Protocol

Evaluating the performance of the trained deep model was necessary. First, when performing the extensive tuning of hyperparameters on the training set, the model was evaluated on the validation set to select the hyperparameters that achieved the best performance. Additionally, once the best-performance model was obtained, the performance on the evaluation set was further evaluated to obtain the final performance. The algorithmic performance including segmentation, percent atrophy and meiboscore grading performance was also examined.

Atrophy segmentation. Two evaluation metrics were adopted to comprehensively assess the similarity between the predicted MG atrophy region and the ground-truth atrophy region.

Accuracy (or pixel accuracy): In atrophy segmentation example, the atrophy region was referred to as our region of interest (ROI). To evaluate how similar the network predictions were compared to the ground truth, the label of ROI (region of interest) was denoted as class i, while the rest as class j. n_{ij} was denoted as the number of pixels of class i predicted to belong to class j, n_{ii} as the number of correctly classified ROI pixels (true positives, TP), n_{ij} as the number of pixels wrongly classified as ROI (false positives, FP), n_{jj} as true negative (TN), and n_{ji} as false negative (FN). If the total pixel of the input image was n, $n = n_{ij} + n_{ii} + n_{jj} + n_{ji}$. Thus, accuracy (abbreviated as ACC) was defined as follows:

$$\text{ACC} = \frac{n_{ii} + n_{jj}}{n} = \frac{n_{ii} + n_{jj}}{n_{ij} + n_{ii} + n_{jj} + n_{ji}} = \frac{\text{TP+TN}}{\text{FP+TP+TN+FN}} \tag{2.2}$$

Similarly, in total eyelid area segmentation example, the eyelid region would be the ROI. ACC reported the percentage of pixels in the image that were correctly classified. However, this metric sometimes provided misleading results when ROI was small, as the measure would be biased by mainly reporting how well non-ROI cases were identified.

Mean IU: Mean intersection over union (mean IU, or Jaccard index) quantified the percent overlap between the target mask and our prediction output. It measured the number of pixels common between the target and prediction masks divided by the total number of pixels present across both masks. Denoting ground truth ROI segmentation mask as GT, network predicted segmentation mask as prediction, mean IU [186] could be written as following:

$$\text{mean IU} = \frac{\text{GT} \cap \text{prediction}}{\text{GT} \cup \text{prediction}} = \frac{n_{ii}}{n_{ij} + n_{ji} + n_{ii}} = \frac{\text{TP}}{\text{FP+FN+TP}} \tag{2.3}$$

Intuitively, Equation 2.3 was analogous to harmonic average of the precision and recall, F1-score, which was defined as 2TP/(FP+FN+2TP), and provided a more "comprehensive"

evaluation - the method considered both precision and recall. For segmentation tasks, higher mean IU value indicated higher alignment of the algorithm prediction with the ground-truth.

In summary, ACC reflected the performance of pixel-wise classification accuracy, while mean IU for how the predicted ROI segmentation mask overlapped with the real counterpart. Mean IU was considered as a stricter evaluation metric than ACC in terms of segmentation. Both evaluation metrics for each image in the evaluation dataset and the calculated average score of all images were reported.

Percent atrophy. Percent atrophy from the atrophy masks (Equation 1) can be calculated, and the percent atrophy performance of algorithm prediction can also be evaluated against the ground-truth. One standard way is to compute the atrophy ratio difference over the evaluation set and compute the root-mean-square deviation (RMSD) as follows:

$$\text{RMSD}(\mathcal{A}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{\mathcal{A}}_\rangle - \mathcal{A}_i)^2} \tag{2.4}$$

where $\mathcal{A}_i$) is the real atrophy ratio of i^{th} image defined in Equation 2.1, $\hat{\mathcal{A}}_\rangle$ represent the i^{th} atrophy ratio predicted by the neural network. N is the total number of images in evaluation set. Intuitively, Equation 2.4 reflects the mean difference between predicted percent atrophy and ground truth over all the images in the evaluation set. RMSD could also be considered as the error made by the algorithm when predicting the percent atrophy of an image.

Converting to meiboscore. Percent atrophy, as a numerical indicator, provided substantial information on the MG atrophy severity of a meibography image. In order to compare with human clinicians' performance, the numerical ratios were converted back to the meiboscore.

From the meibography images with different percent atrophy in Figure 2.1, images near the grading transition limits (0, 33%, 66%) were very similar and difficult to classify. A tolerance threshold near the grading transition limit was necessary. The converting criteria in Table 2.1 was applied with a relaxed standard. As illustrated in Figure 2.5, the tolerance threshold was set at 0.25%, so classifying images with percent atrophy 0-0.25%, 32.75% - 33.25%, and 65.75% - 66.25% either to its ground-truth or adjacent meiboscores were both considered as correct prediction. The same relaxed converting criteria for both human clinicians and algorithm in the experiments were followed for fair comparison.

2.3 Results

2.3.1 Network Training Details

Each meibography image and its corresponding segmentation mask(s) were resized to the size of 420×420 pixels. During training, 400×400 pixels were randomly cropped out of a given meibography image and corresponding annotations in every training epoch for data augmentation. A center crop of 400×400 pixels was made to a given meibography image and

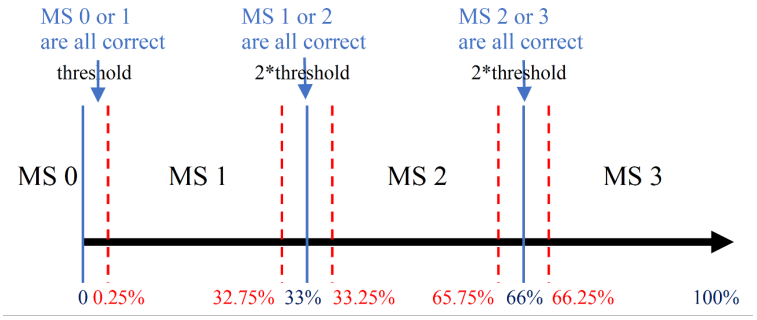


Figure 2.5: Relaxed meiboscore conversion rule with the tolerance threshold. The percent atrophy to the meiboscore conversion criteria is relaxed with tolerance threshold near the grading transition limits (0, 33%, 66%). The threshold is set to be 0.25%. Therefore, when percent atrophy falls in 0 - 0.25%, 32.75% - 33.25% or 65.75 – 66.25%, the correct prediction can be either the ground-truth meiboscore or the adjacent meiboscore.

corresponding annotations during the evaluation process for both validation and evaluation datasets.

Different network architectures (SqueezeNet, resnet18, resnet34, resnet50), auxiliary loss ratio (for 0 to 1.0 with grid of 0.1), learning rate, and learning rate decreasing policy were carefully assessed to obtain the best performance of the network on the validation dataset[1]. The algorithm performance of the model on the evaluation dataset is reported.

2.3.2 Algorithm Performance

The baseline characteristics of the training and evaluation dataset were described in Table 2.2. Development and evaluation dataset had similar characteristics regarding patient demographics and MG atrophy severity.

Table 2.3 shows the ACC, mean IU and RMSD of meibography images with different meiboscores. Note that images with meiboscore of grade 0 do not have atrophy so there are no corresponding ACC and mean IU. The performance of meiboscore of grade 0 can however be measured by RMSD and meiboscore grading accuracy. Regarding atrophy segmentation, ACC was higher than mean IU. The instance average ACC values were 97.6% and 95.5% for eyelid and atrophy respectively, while mean IU values were 95.4% and 66.7% for eyelid and atrophy respectively. Regarding mean IU for different gland atrophy severity, mean IU was the lowest for meiboscore of grade 3 samples, which only included 5 images in the training set and only 2 in evaluation set. Regarding percent atrophy prediction performance, the instance average error was 6.7%. Although images with different meiboscores had relatively similar RMSD, the RMSD value was the highest at 9.0% for meiboscore of grade 1 and

[1]The best performance of the network over the validation set was resnet50, auxiliary loss ratio 0.1, learning rate 1e-3, 200 epochs in total, with learning rate decrease at 100, 150, and 180 respectively.

Table 2.3: Performance of the eyelid and atrophy segmentation algorithm

	Eyelid Area		Atrophy Area		Percent Atrophy
	ACC	mean IU	ACC	mean IU	RMSD
Meiboscore 0	97.9%	96.1%	/	/	9.0%
Meiboscore 1	97.5%	95.5%	95.7%	64.6%	6.2%
Meiboscore 2	97.6%	95.6%	94.5%	78.3%	5.7%
Meiboscore 3	91.9%	82.1%	86.8%	63.7%	6.9%
Class average accuracy	96.3%	92.3%	92.4%	68.8%	7.0%
Instance average accuracy	97.6%	95.5%	95.4%	66.7%	6.7%

Table 2.4: Performance of the eyelid and atrophy segmentation algorithm under 10-fold cross validation

	Eyelid Area		Atrophy Area		Percent Atrophy
	ACC	mean IU	ACC	mean IU	RMSD
Meiboscore 0	98.0 ± 0.4%	96.0 ± 0.5%	/	/	9.8 ± 0.8%
Meiboscore 1	97.6 ± 0.3%	95.4 ± 0.6%	96.1 ± 1.7%	65.9 ± 1.8%	5.6 ± 0.6%
Meiboscore 2	97.9 ± 0.5%	96.0 ± 0.9%	94.6 ± 1.9%	77.3 ± 1.8%	7.4 ± 1.6%
Meiboscore 3	92.4 ± 2.0%	83.8 ± 4.0%	85.1 ± 3.9%	61.6 ± 4.5%	7.3 ± 2.0%
Class average accuracy	96.4 ± 0.8%	92.8 ± 1.5%	91.9 ± 2.5%	68.3 ± 2.7%	7.5 ± 1.3%
Instance average accuracy	97.6 ± 0.4%	95.5 ± 0.7%	95.7 ± 1.8%	67.6 ± 1.8%	6.6 ± 0.8%

the lowest at 5.7% for meiboscore of grade 2. Although the intention was to capture each subject for one single visit, there were 3 subjects (providing a total of 10 images) from 467 subjects (0.6% subjects) returned to the research facility for two visits within at least 2-year time lapse. Therefore, a 10-fold cross validation was performed to confirm that the images obtained from the repeated visits would not bias the study results. The results (e.g., mean and standard deviation of ACC, mean IU and RMSD) were reported in Table 2.4. Specifically, the development and evaluation sets were randomly split 10 times according to the number of images of each meiboscore category presented in Table 2.2. The algorithm was trained on different development sets and evaluated on the corresponding evaluation sets for 10 times. Figure 2.6 plots the predicted percent atrophy versus ground-truth percent atrophy. While most of the points fall on the ideal line (percent atrophy prediction equals to ground-truth), the deep learning algorithm tends to give higher percent atrophy for some cases of meiboscore of grade 0. This is because the algorithm has been trained to be sensitive to even small atrophy, which might have been ignored by clinicians. The errors are greatly reduced when converting to meiboscores using the relaxed criteria as described in the "evaluation protocols" section.

Figure 2.7 visualizes the atrophy region and eyelid region segmentation results of clin-

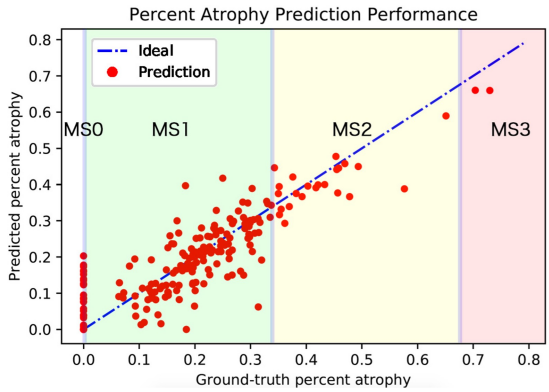


Figure 2.6: Algorithm-predicted percent atrophy versus ground-truth percent atrophy. The average root-mean-square deviation (RMSD) of the predicted percent atrophy is 6.7%. While most of the points fall on the ideal line (percent atrophy prediction equals to the ground-truth), more percent atrophy errors occurred for meiboscore of grade 0. Note that the errors are greatly eliminated when applying relaxed meiboscore conversion criteria. The thresholds near grading transition limits are marked in blue.

Table 2.5: Confusion matrix of LCI and clinical meiboscore

			LCI Meiboscore			
	Number		Meibo 0	Meibo 1	Meibo 2	Meibo 3
Clinical Meiboscore	96	Meibo 0	45%	51%	4%	0%
	84	Meibo 1	11%	69%	18%	2%
	22	Meibo 2	0%	32%	50%	18%
	5	Meibo 3	0%	20%	40%	40%

ician team and computer. From the visualization, the human segmentation and computer segmentation appear to be very similar, especially for eyelid region segmentations.

2.3.3 Human Clinician's Performance

The confusion matrix is a specific table layout that allows visualizations of how two identities perform the same classification task. Table 2.5 shows the confusion matrices of study clinician and single clinician. The highest agreement percentage was 69% for meiboscore of grade 1, while the lowest was 40% for meiboscore of grade 3. The kappa score is 0.324 for clinical team and the LCI, which led to a fair agreement according to Landis et al.[164] In other words, clinicians' ratings have high variability.

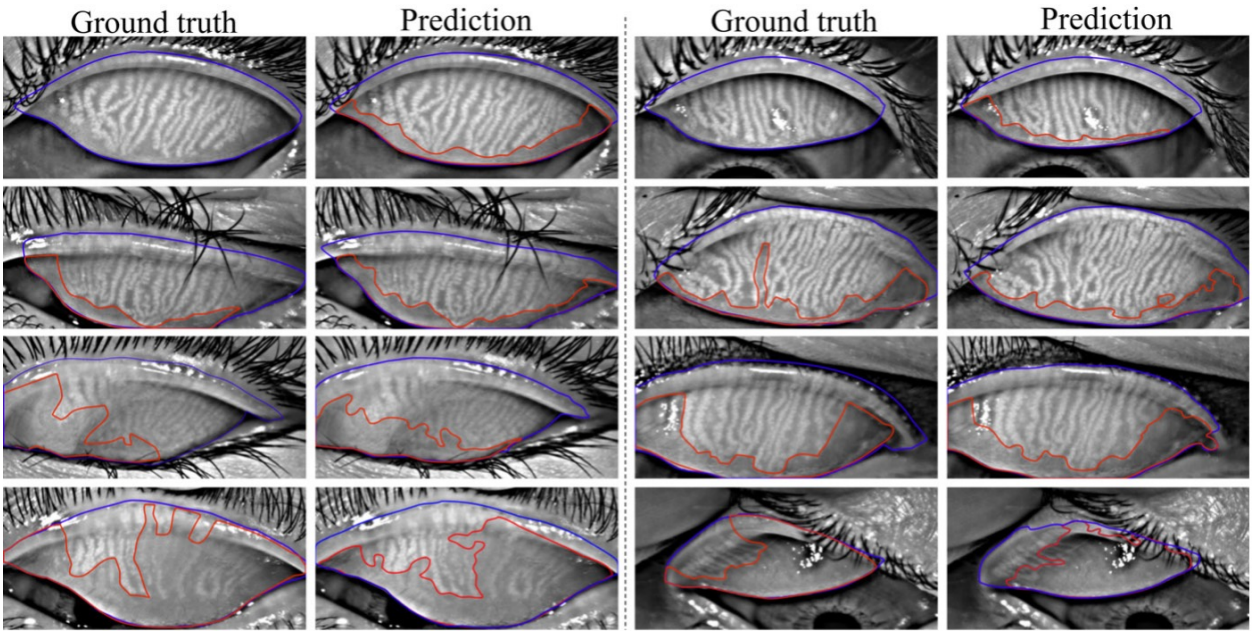


Figure 2.7: Eyelid (outlined in blue) and atrophy (outlined in red) segmentations from the deep learning algorithm versus the ground-truth. Rows 1-4 refer to meiboscores of grade 0-3, respectively. The first and third columns are ground-truth images from the annotators, while the second and fourth columns are algorithm predictions. The algorithm predictions shared high visual similarity with the ground-truth, especially for eyelid region segmentation.

2.3.4 Comparing Meiboscores

The meiboscore grading performance of the algorithm was compared against the ground-truth meiboscores. The ground-truth meiboscores were obtained from the percent atrophy (calculated from human-annotated segmentation masks) using Table 2.1. Table 2.6 and Figure 2.8 show the meiboscore grading performance by the algorithm, clinical team (clinical meiboscore), and the LCI meiboscore. The algorithm achieves 95.6% overall grading accuracy, which outperforms the LCI meiboscore by 16.0% and clinical team meiboscore by 40.6%. For each meiboscore, the algorithm also largely outperforms human clinicians. Figure 2.9 depicts some failure cases in meiboscore grading of human clinicians and our algorithm. Failure cases appear for both human-assigned and algorithm meiboscores when percent atrophy is near the meiboscore grading transition limits.

2.4 Discussion

The present work introduces a deep learning approach to automatically predict the MG atrophy region and compute percent atrophy in the meibography image. The proposed method

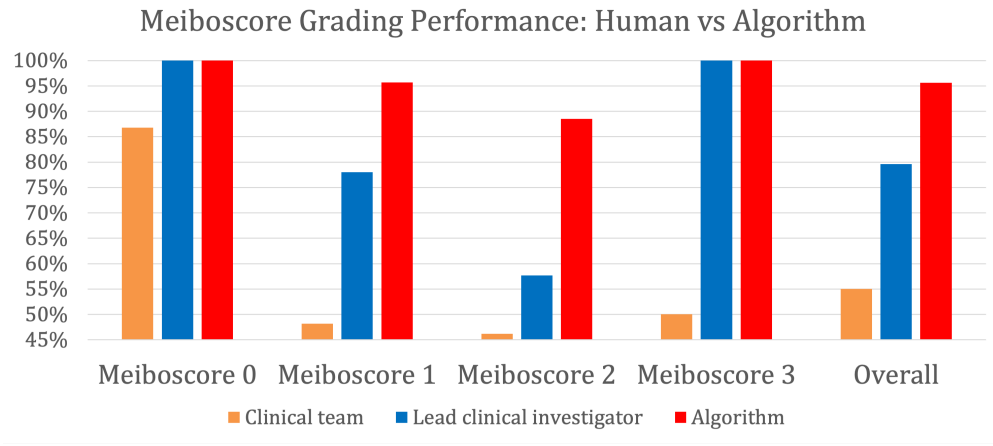


Figure 2.8: Meiboscore grading performance of the clinical team, the lead clinical investigator and the proposed algorithm. The algorithm outperforms the LCI meiboscore by 16.0% and clinical team meiboscore by 40.6%.

Table 2.6: Meiboscore grading performance of clinicians and algorithm

	Clinical team	Lead clinical investigator	Algorithm
Meiboscore 0	86.8%	100%	100%
Meiboscore 1	48.2%	78.0%	95.7%
Meiboscore 2	46.2%	57.7%	88.5%
Meiboscore 3	50.0%	100%	100%
Class average accuracy	57.8%	83.9%	96.1%
Instance average accuracy	55.0%	79.6%	95.6%

has three advantages: (1) low variability and high repeatability (test–retest reliability); (2) output quantitative result rather than qualitative description, e.g., specific gland atrophy region and numerical percent atrophy prediction; (3) efficient and low cost. The average processing time per meibography image was approximately 0.29 seconds[2]. This means that more than 1000 unprocessed or raw meibography images can be evaluated for atrophy severity in 5 minutes without additional human resource needed.

The algorithm also has very high performance. The accuracies of eyelid area and atrophy area achieve 97.6% and 95.4% respectively, and the overall mean IUs are 95.5% and 66.7% respectively. Our algorithm achieves a 95.6% overall grading accuracy and outperforms the LCI meiboscore grading accuracy by 16%. From the visualization of the predicted eyelid and atrophy segmentation, the algorithm predictions have high visual similarity with the

[2]Experiments performed on one NVIDIA GeForce GTX 1080 GPU.

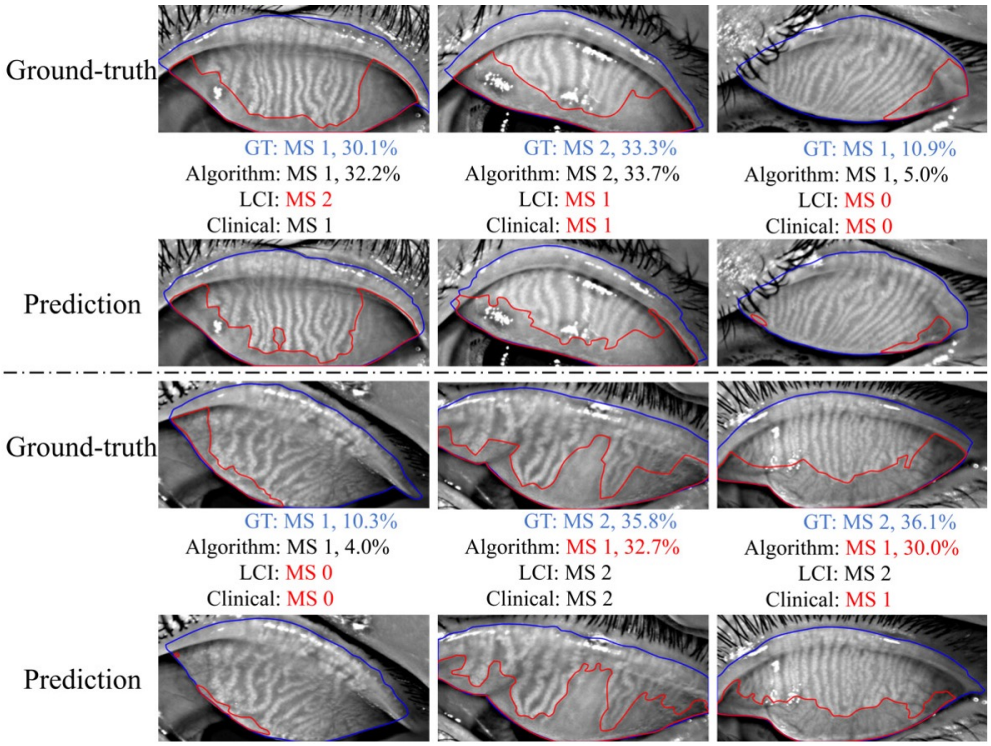


Figure 2.9: Examples of failure cases in meiboscore (MS) grading: human clinicians vs machine algorithm. GT refers to the ground-truth. Failure cases occur for both clinicians and the algorithm for the images with percent atrophy near the meiboscore grading boundaries.

ground-truth annotations. In all, the proposed algorithm achieves very high performance with low variability and high repeatability in evaluating MG atrophy from meibography images. Additionally, the proposed algorithm could potentially be applied to other similar image segmentation tasks in the clinical community.

Furthermore, regarding MG atrophy evaluation, numerical-percent-atrophy reporting is a better assessment than simple meiboscore grading. It is challenging to distinguish different meiboscores when the percent atrophy is near the grading transition limits (0, 33%, 66%), since no distinct changes observed for meibography images in these regions. Forcing a strict meiboscore near the boundary can lead to grading inconsistency and variability. When applying numerical percent atrophy, however, clinicians would not need to worry about the images with percent atrophy near these grading transition limits. Additionally, there are several different grading scales for gland atrophy [12, 235]. Conversion between different grading system is impossible as the percentage information already got lost. Numerical percent atrophy adopted in this study overcomes the above-mentioned problems.

In conclusion, a deep learning approach to automatically evaluate the MG atrophy in meibography images has been developed. The system has high accuracy, repeatability and

low variability, as well as outperforming human clinicians by a significant margin. The quantitative outputs (specific atrophy region and percent atrophy) provide valuable information of MG atrophy severity of the meibography image. In the present work, our deep learning system could only predict MG atrophy region, but not individual MG morphology. Future work can explore how deep learning can automatically analyze MG morphological characteristics (e.g., gland number, width, intensity and tortuosity), which can be potentially important for advancing the efficiency of MGD treatment and management. Capturing human expert knowledge with data-driven, deep-learning systems is the future of image-based medical diagnosis. The possibility of using deep learning methods for clinical diagnosis in ocular surface diseases is shown. New surveys can be enabled by automatic and quantitative evaluations, opening up many exciting opportunities for targeted medical treatment and drug discoveries.

Chapter 3

Learning Fine-Grained, Instance-Level Representations of Meibomian Glands

3.1 Introduction

Meibomian glands (MGs), located in the upper and lower eyelids with orifices at the eyelid margins, are believed to play a critical role in ocular surface health by secreting lipids into the tears, which form a polar lipid film that serves to retard aqueous evaporation. Meibomian gland dysfunction (MGD) can result in an unstable tear film and excessive evaporation of tears from the ocular surface, and is the most frequent cause of dry eye symptoms [28]. The ability to visualize MGs and to monitor their temporal changes is important for evaluating the risk of MGD, identifying active pathology, and following treatment outcomes. Meibography, which is the photodocumentation of MGs in vivo using either trans-illumination or infrared imaging, is now commonly used in clinics for the diagnosis, treatment, and management of MGD [194, 235].

The relative size of areas of MG atrophy, or gland loss area, is an important clinical measure for assessing MGD severity. Currently, clinicians estimate the degree of MG atrophy subjectively by comparing the area of glandular loss to the total eyelid area [235, 237]. Though commonly used, the method only evaluates the overall severity of gland atrophy, not detailed individual MG morphological features. Recent studies have shown that MG morphological features, such as length, curvature or tortuosity, and local contrast (average gland region intensity normalized by its surrounding intensity) may also be indicative of MGD severity and related to ocular surface disease [197, 134, 340, 63]. Additionally, the presence of glands that were once patent but now contain little or no meibum, referred to as ghost glands, is also thought to be associated with MGD. These glands have a faint or "ghostly" appearance in meibography images due to decreased local contrast of the glandular structures [63, 306].

To accurately measure individual MG morphological features from meibography images would be prohibitively time consuming in a patient care setting and would require exten-

sive training and standardization across the field to be accurate and repeatable for research purposes. To date, there has been little progress made in understanding the roles played by various MG morphological features in MGD and various related pathologies (e.g., dry eye, tear film instability). The lack of quantification tools for individual gland morphology hinders such progress. Fortunately, recent advances in deep learning [162, 218, 47], a form of artificial intelligence, demonstrate the ability of deep neural networks to learn to perform quantification tasks directly from a large dataset of images without explicitly specifying rules. Recent papers [302, 233] have proposed deep-learning-based approaches to analyzing MG morphology from meibography images and some have outperformed trained human observers. However, such methods are only capable of evaluating global MG morphology rather than the fine-grained morphology of individual glands. Therefore, employing deep learning approaches to partitioning individual MG regions from meibography images, a process known as image segmentation, quantitatively measuring gland morphological features, and identifying the presence of ghost glands is of timely interest to many clinicians.

This study aims to develop an automated approach based on deep learning that segments MGs in meibography images, identifies ghost glands, and quantifies individual MG morphological features. The health and morphology of the MGs are likely to be related to the quality of the lipid layer and the stability of the tear film. If such detailed morphological features could be obtained and quantified quickly and accurately, this would provide researchers with a powerful tool to quantitively explore their etiological significance in ocular surface disease. In the longer term, the proposed tool could make a significant contribution to improving the diagnosis, treatment, and long-term management of tear film instability and evaporative dry eye.

3.2 Methods

3.2.1 Data Collection and Annotation

Subject recruitment and imaging. Adult human subjects, mean (SD) age 27.8 (12.9) years, were recruited from the University of California, Berkeley campus and surrounding community for single-visit ocular surface evaluations during the period from 2012 to 2017. Eligible subjects were 18 years of age or older with no history of ocular injury or surgery, and not taking prescription ocular medications or systemic medications with effects on the ocular surface or tear film. Contact lens wearers were required to discontinue wear at least 24 hours prior to the study visit. Meibography images of the upper and lower eyelids of both eyes illuminated with 800 nm infrared light were captured with the OCULUS Keratograph 5M (R) (OCULUS, Arlington, WA) [194]. During image capture for all subjects, the ambient light was off with the subject's head positioned in a chin rest and forehead strap apparatus.

Informed consent was obtained from all subjects after a thorough description of the goals, procedures, risks and potential benefits of the study. This study conformed to the tenets of the Declaration of Helsinki, and was approved by the U.C. Berkeley Committee for

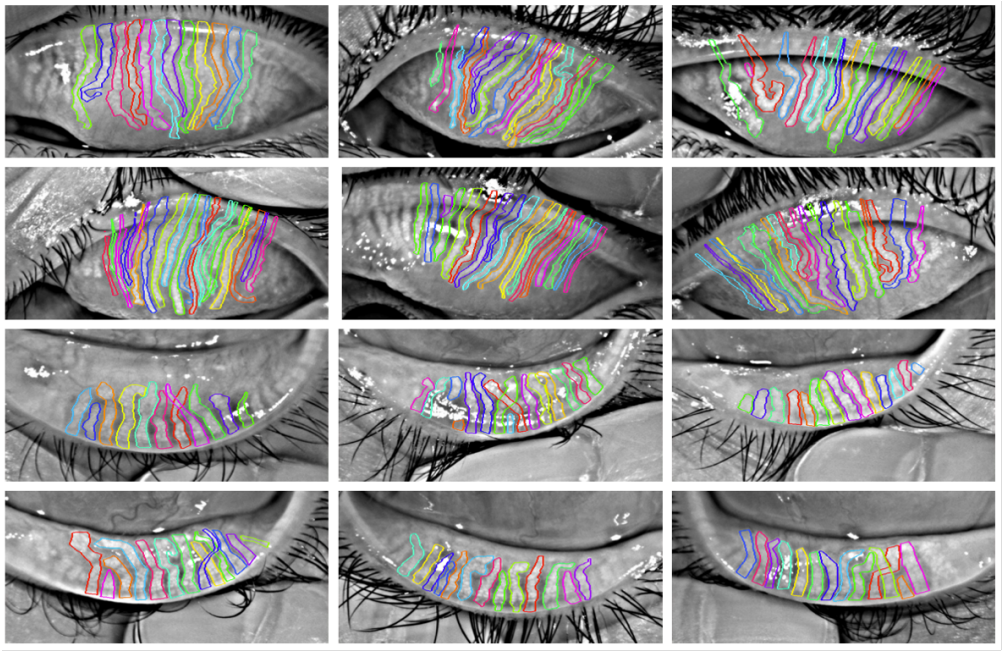


Figure 3.1: Meibography images with ground truth MG regions (outlined in color). The first two rows refer to upper eyelids while the last two refer to lower eyelids. Different colors correspond to different glands. Only MGs overlapping with the central region with width equaling to 50% eyelid width were considered since the far temporal and far nasal regions are out of focus.

Protection of Human Subjects.

Data annotations. A total of 1550 images were collected and pre-screened to rule out images that did not capture the entire eyelid (107 images or 6.9%), thus 1443 images were used for the study. This study utilized a meibography image dataset from a previous paper [302], in which the percent area of MG atrophy was estimated in upper eyelid meibography images. In the current paper, to 689 of these upper eyelid images were added 754 lower eyelid images from the same subjects. Both upper and lower eyelid images then had the total eyelid area, any regions of MG atrophy, and each individual MG contour traced, or "annotated", by one of 10 trained observers, using the polygon tool in Fiji (ImageJ version 2.0.0-rc-59)[259]. For lower eyelid images, only the individual MGs were annotated, not the overall lid and atrophy regions, because the border of the lower tarsal plate could not be easily defined due to inter-subject variation in lower eyelid eversion. Far temporal and nasal gland contours were not annotated because these regions are generally out of focus in meibography images due to the anterior curvature of the eye. All visible glands within approximately the central 50% of the eyelid area of each meibography image were annotated. Every gland was also identified as a ghost or non-ghost gland. Note that ghost gland regions and MG atrophy regions are mutually exclusive because ghost glands are faint but still visible, while atrophy regions have

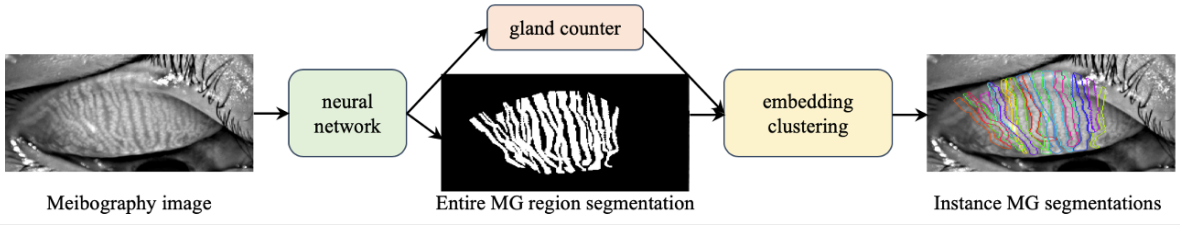


Figure 3.2: The MG segmentation network. The gland segmentation network takes a meibography image as an input and produces two intermediate outputs: the entire MG region segmentation (along with pixel-wise feature embedding, see the supplementary materials for details) and the number of glands. The predicted MG contour/embedding and gland number are further exploited by the network to predict instance MG region segmentation.

no discernable glands for annotation. Final annotations were verified by a single lead clinical investigator (TNY) before they were made available to the machine learning algorithm, in order to minimize variability in the ground truth data (i.e., the manual annotations from trained observers) that would arise from using multiple unsupervised observers. Figure 3.1 depicts several examples of MG region annotations. Detailed statistics can be found in Table 3.1. Algorithms were considered to achieve 100% accuracy if they predicted results identical to the ground truth annotations.

Data allocations. Meibography images were allocated into two subsets according to collection time. The development set contained meibography images collected from the years 2015 to 2017, while the evaluation set contained those collected from the years 2012 to 2013. The development set was used for developing the deep learning algorithm, while the evaluation set was used for evaluating the performance of the algorithm. For algorithm development, the development set was further divided randomly into 2 subsets for training and validation. The images in the training set were used to train the deep learning model, while the validation set was used for tuning the model hyperparameters (e.g., network architectures, learning rate). The evaluation set, which did not have any overlapping images/subjects with the development dataset, was evaluated using the model that achieved the best performance from the validation set.

3.2.2 Algorithm Design and Training

Data processing. Raw MG annotations were preprocessed to a standardized format before feeding to the deep learning model. Detailed procedures are available in the supplementary materials.

Gland segmentation. In computer vision, image segmentation is the process of partitioning an image into multiple regions [223, 191]. Instance segmentation, one important task of image segmentation, aims to identify the contour of each object at the detailed pixel level [252, 182]. Instance segmentation techniques are suitable for segmenting MGs since

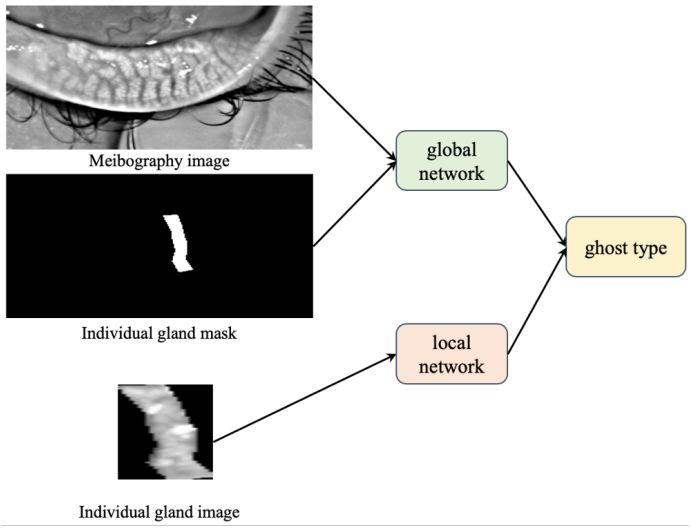


Figure 3.3: Ghost gland classification network. The neural network has three inputs: an entire meibography image, a specific gland mask (the region within the predicted contour) and a corresponding cropped gland image. Inputs consist of a global view of the meibography image and a local view of individual gland, and thus two subnetworks are used to learn to identify ghost glands. The global network learns a representation from the entire meibography image and the individual gland mask, while the local network learns a representation from the individual gland image. Two representations are concatenated to predict ghost glands. In this example, the network outputs "0", indicating it is a non-ghost gland.

contours of individual MGs are desired outputs. Recently, deep learning approaches have outperformed all other methods in instance segmentation [182, 80, 65]. A deep learning algorithm was built upon an instance segmentation network with discriminative loss[65] to predict individual MG regions in meibography images not previously seen in the training phase. Figure 3.2 depicts the architecture of the proposed MG segmentation network, which takes a meibography image as input and produces the contour for each gland as output. Details on the network design and training can be found in the supplementary materials.

Ghost gland identification. The deep learning model was also designed to identify ghost glands (Figure 3.3) at the individual MG level from a meibography image. Details on the network design and training can be found in the supplementary materials.

3.2.3 Evaluation Metrics

After finishing the training of deep learning models and identifying the model with the best performance on the tuning set of images, its performance with respect to MG instance segmentation and ghost gland identification was evaluated on the evaluation dataset.

Gland segmentation. To quantify the similarity between the predicted MG region (i.e., the entire MG region within the predicted contour) and the ground truth, the mean intersection over union (mean IU) was used. This metric was also used by a previous paper evaluating predicted MG atrophy segmentation [302]. Mean IU, or the Jaccard index, quantifies the percent overlap between the target region and the prediction output. It measures the number of pixels in common between the target and prediction regions divided by the total number of pixels present across both regions. Denoting ground-truth gland region segmentation as GT, network predicted segmentation as P, the mean IU[186] can be written as follows

$$\text{mean IU} = \frac{\text{GT} \cap \text{P}}{\text{GT} \cup \text{P}} \tag{3.1}$$

Intuitively, mean IU is analogous to the harmonic average of the precision and recall (F1-score) and provides a fair evaluation of the segmentation algorithm by considering both precision and recall. For segmentation tasks, a higher mean IU value indicates a better alignment of the algorithm prediction with the ground truth.

Gland detection. In addition to evaluating the similarity between the predicted region and the ground truth, it is also necessary to evaluate how well the model detects objects for instance segmentation. Since there are multiple glands presented in an image, a high detection performance means the model does not miss or wrongly identify individual glands. The evaluation metric is available in the supplementary materials.

Ghost gland identification. Confusion matrices and receiver operating characteristic (ROC) curves were used to evaluate ghost gland classification performance. A confusion matrix is a table that is used to describe the performance of a classification model on a set of test data for which the true class labels are known. Each row of the confusion matrix represents a true class label while each column represents each a model-predicted label. Sensitivity and specificity (with 50% probability threshold) were calculated from the confusion matrix. A ROC curve[107] graphically depicts the diagnostic ability of a binary classifier system as its threshold is varied.

3.2.4 Morphology Analysis

A meibography image reveals the morphology of multiple glands. Individual gland features as well as global morphology are both important. We thus aimed to quantify both local and global morphological features.

Local morphological features. Based on individual MG segmentations, morphological features such as average gland local contrast, gland length, width and tortuosity can be quantitively measured. Details on measuring each feature from a meibography image are available in the supplementary materials.

Visualizations of gland local contrast, length and tortuosity extraction are shown in Figure 3.4, Figure 3.5 and Figure 3.6, respectively. Specifically, local contrast of a gland region represents its pixel intensity normalized by the surrounding pixel intensity (gray

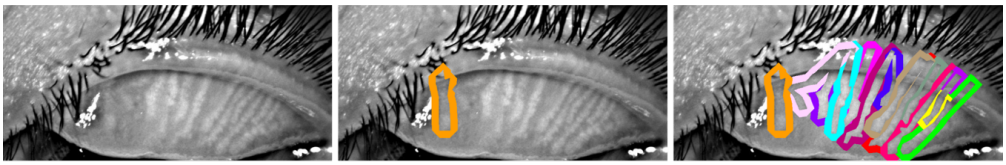


Figure 3.4: Local contrast calculations of MGs. To "normalize" the individual gland intensity, the average gland local contrast is defined as the difference of average intensity of the gland region and surrounding area with negative values set to 0. Specifically, given an input image (left), instance glands are segmented. For a segmented gland region (middle), its surrounding area (outlined in orange) can be obtained by binary dilation operation (see the supplementary materials for details). The difference in intensity is thus calculated. The surrounding areas of all glands are also shown (right).

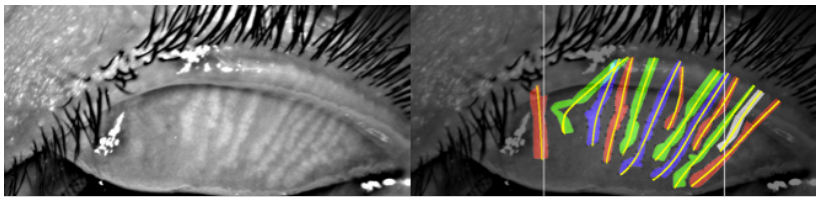


Figure 3.5: Principal curves (marked in yellow) fit for gland length calculation. Gland length is defined as the number of pixels in each principal curve.

scale: 0-255). Glands containing little or no meibum have a faint, ghost-like appearance[306] and low contrast in the meibography images. Additionally, [340] showed that gland local contrast in meibography captured using the same instrument was repeatable and invariant to ambient light conditions. Therefore, it is of interest to include local contrast of a gland as a morphological feature in the present study.

Association of local morphological features with ghost glands. One goal of analyzing MG morphology is to understand which morphological features are most associated with ghost glands. A linear support vector machine (SVM) for classifying MGs as ghost glands or non-ghost glands was constructed with the four aforementioned morphological features as explanatory variables (inputs). The model could be considered to be using the weighted sum of input morphological features as the probability of classifying a gland as a ghost gland. Weights are learned by training the model. Feature coefficients, or model weights, thus reflect the relative importance of different features for the ghost gland decisions of the model.

Global morphological features. Several global morphological features can also be measured quantitively with this deep learning approach. Global features that can be extracted include number of glands per image, gland density (the ratio of the gland area to the

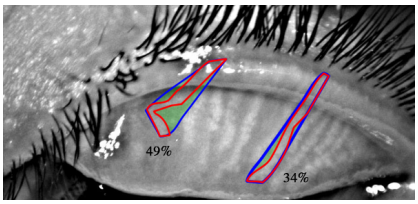


Figure 3.6: Two sample MGs with tortuosity marked below. For each MG (outlined in red), the convex hull (outlined in blue) of the shape is extracted. Gland tortuosity is defined as the convexity defect of a gland, the ratio of the cavity area (green region) to the convex hull area. More tortuous glands have greater gland tortuosity values. The left gland is more tortuous.

eyelid area), and percent atrophy (the ratio of atrophy area to the eyelid area).The number of glands was calculated for both upper and lower eyelids while gland density and percent atrophy were calculated over upper eyelids only due to the inability to annotate the full eyelid and atrophy regions in lower eyelid images as explained above.

3.3 Results

The algorithm development and evaluation were performed for the whole dataset with both upper and lower eyelid images for local morphological features, and with upper eyelid images for global morphological features. Normalized histograms of global morphological features (total number of glands, gland density, and percent atrophy) are shown in Figures 7 (a)- (c) for upper eyelid images, and of local morphological features (local contrast, length, width, tortuosity, and identification of ghost glands) in Figures 7 (d)-(h).

3.3.1 Dataset and Training Details

Subject demographics stratified on the development and evaluation datasets can be found in Table 3.1. Details of data preprocessing and network training are available in the supplementary materials.

3.3.2 Gland Segmentation Performance

Table 3.2 reports the performance of the MG instance segmentation algorithm. For gland segmentation, mean IUs were 58.4% and 68.0%. Gland detection performance is available in the supplementary materials. Figure 3.7 and Figure 3.8 provide visualizations of the ground truth and predicted MG segmentations for upper and lower eyelids, respectively, with the last row in each figure presenting illustrative examples of failure cases.

Table 3.1: Subject Demographics of the Meibography Image Dataset

Upper eyelid	**Development**		**Evaluation**
	Train	**Tune**	
Images (No.)	389	97	203
Patient Demographics			
Unique individuals (No.)	260	94	109
Age (average ± SD)	27.8 ± 13.1	27.0 ± 11.5	27.9 ± 12.7
Female/ total patients	69.60%	66.00%	69.40%

Lower eyelid	**Development**		**Evaluation**
	Train	**Tune**	
Images (No.)	445	108	201
Patient Demographics			
Unique individuals (No.)	272	97	106
Age (average ± SD)	27.9 ± 13.1	28.2 ± 13.4	27.7 ± 12.6
Female/ total patients	68.40%	63.90%	67.60%

Table 3.2: Gland Segmentation Performance

	Upper eyelid	Lower eyelid
Precision	62.70%	76.90%
Recall	53.90%	74.10%
Mean IU	58.40%	68.00%

3.3.3 Ghost Gland Identification Performance

Table 3.3 reports confusion matrices of ghost gland classification for the upper and lower eyelids. The deep learning algorithm achieved 84.7% sensitivity and 72.5% specificity for ghost gland classification in upper eyelid images, and 84.1% sensitivity and 70.8% specificity for ghost gland classification in lower eyelid images. ROC curves for ghost gland classifications for the upper and lower eyelids are shown in Figure 3.9.

3.3.4 Morphological Features and Associations with Ghost Glands

To give readers a sense of extracted features, individual MG morphological features from an example meibography image are shown in Figure 3.11. Detailed morphological features including local contrast, gland length, width, tortuosity and ghost gland identification of the

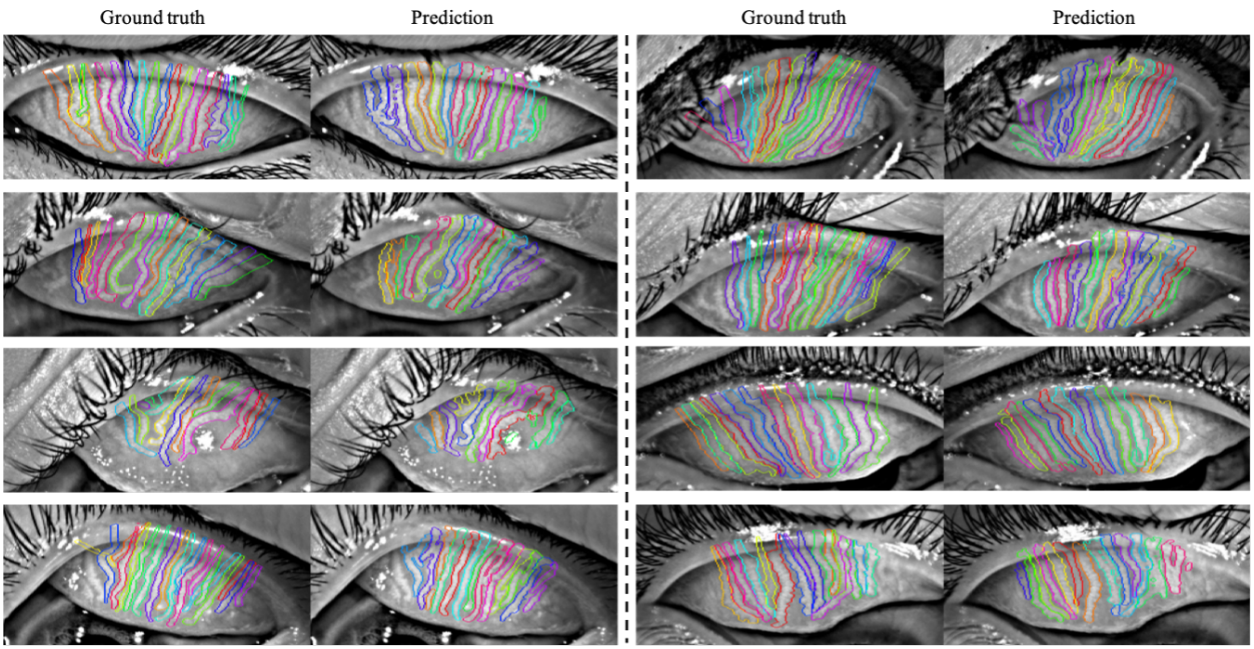


Figure 3.7: Upper eyelid MG segmentation ground truths versus predictions. Different colors refer to different MGs. The predictions were visually similar to the ground truths. The last row depicts some failure cases.

Table 3.3: Performance of Ghost Gland Classification

Upper eyelid	Predicted: Ghost	Predicted: Non-ghost
Actual: Ghost	188 (84.7%)	34 (15.3%)
Actual: Non-ghost	789 (27.5%)	2085 (72.5%)

Lower eyelid	Predicted: Ghost	Predicted: Non-ghost
Actual: Ghost	74 (84.1%)	14 (15.9%)
Actual: Non-ghost	699 (29.2%)	1693 (70.8%)

13 visible glands in that image are shown. Figure 3.10 depicts the normalized histograms of local and global morphological features for the whole meibography image dataset. To determine the important morphological gland features associated with ghost glands, the feature coefficient of an SVM model constructed on MG features was analyzed. The feature coefficient of the SVM model reflects the importance of different input features. An SVM model achieving 57% sensitivity and 65% specificity was used. The corresponding feature coefficient of gland local contrast was approximately 73% , gland length was 69%, gland width was 14% and gland tortuosity was 30%. Gland local contrast and length had comparably

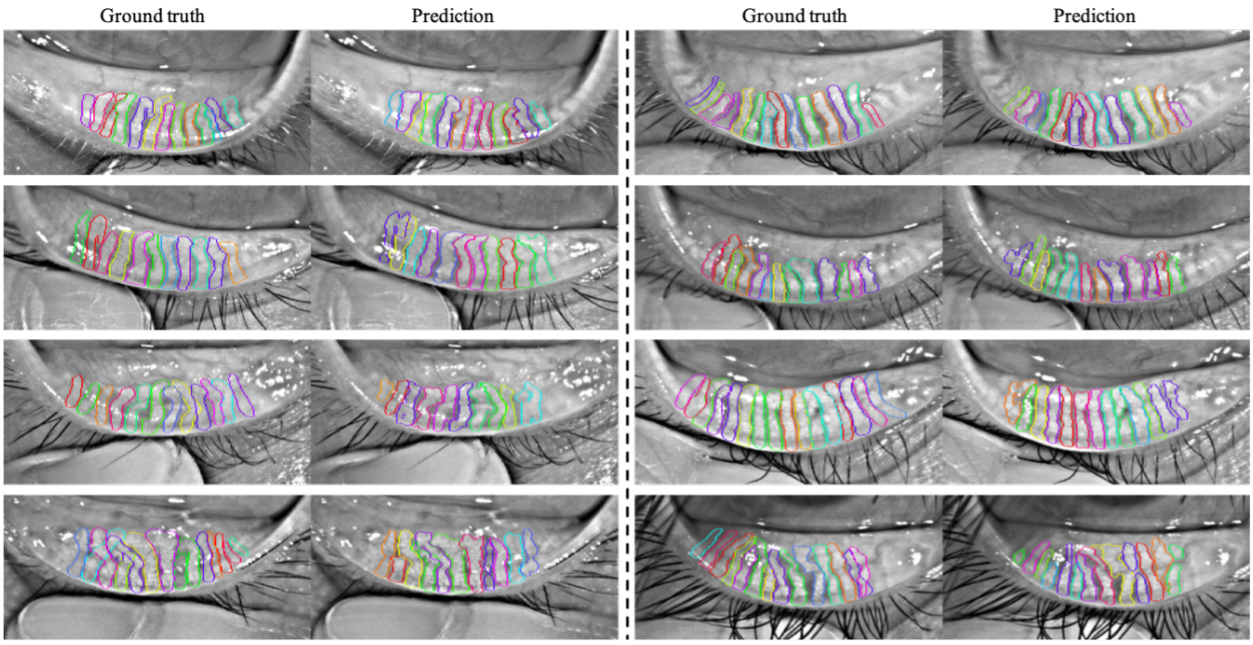


Figure 3.8: Lower eyelid MG segmentation ground truths versus predictions. Different colors refer to different MGs. The predictions were visually similar to the ground truths. The last row depicts some failure cases.

large coefficient values.

3.4 Discussion

This work presents a deep learning-based approach to automatically analyzing MG morphology by segmenting individual glands from meibography images. This approach introduces the following three innovations: 1) Instance gland regions are automatically segmented from the meibography image; 2) Ghost glands are automatically identified; 3) Morphological features at the individual MG level are analyzed. The proposed approach to quantifying gland morphological features makes it possible to further study the quantitative relationships between gland morphology and clinical signs, to improve the understanding of MG morphology and pathology, and could ultimately provide quick and accurate diagnostic information in the clinical setting.

Previous works have explored the plausibility of automatically segmenting glands from meibography images with the goal of MG morphological analysis, and have achieved satisfactory results for certain images [233, 43]. However, such methods only segment entire MG regions instead of individual MG regions, which is a serious limitation for further morphological analysis. While additional post-processing could break the entire region into several

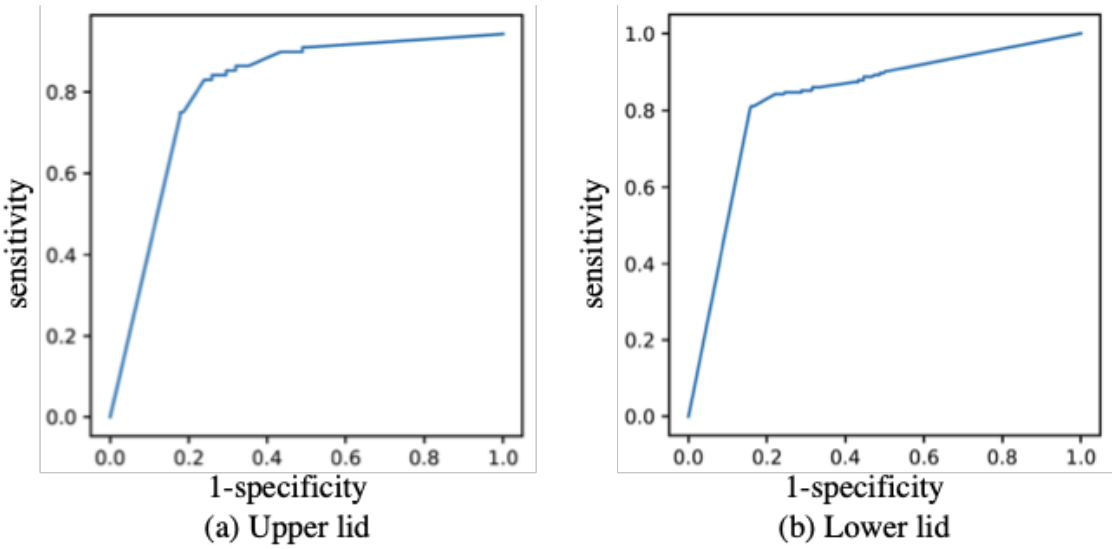


Figure 3.9: ROC curves for ghost gland classification.

smaller parts, such methods are unlikely to achieve adequate results as MGs are usually dense, close to or even overlapping with each other. The proposed method, in contrast, is able to segment instance glands even when they are close or slightly overlapping, which makes it more appropriate for further morphological analysis. Empirically, a mean IU with value of 50% or higher suggests a good match between the prediction and the ground truth in computer vision. Thus, both the numerical performance evaluation metric (58% and 68% for upper and lower eyelid, respectively) and the visual examples reported in the previous section validated the acceptable performance of the proposed algorithm.

The proposed approach is able to identify ghost glands, which have been shown previously to be indicative of MGD [63, 306]. Based on gland segmentation and quantified morphological features, further analysis of the association between individual gland morphological features and ghost glands suggested that low MG local contrast was most associated with the presence of ghost glands. This is consistent with the definition of ghost glands, which have low contrast to their surroundings in images since they contain little or no meibum. The coefficient of gland length is the second largest, meaning that longer glands are more likely to be ghost glands. Future studies of the links between gland health, pathology and morphological features such as gland local contrast and length are warranted to verify and further understand this finding.

The average processing time (including segmentation, classification and morphological analysis) per image was approximately 0.32 seconds. Experiments were performed on a single graphics processing unit (GeForce RTX 2080 Ti. NVIDIA, Santa Clara, CA.). This means that more than 900 unprocessed or raw meibography images can be evaluated for atrophy severity in 5 minutes without additional human intervention. The processing speed renders

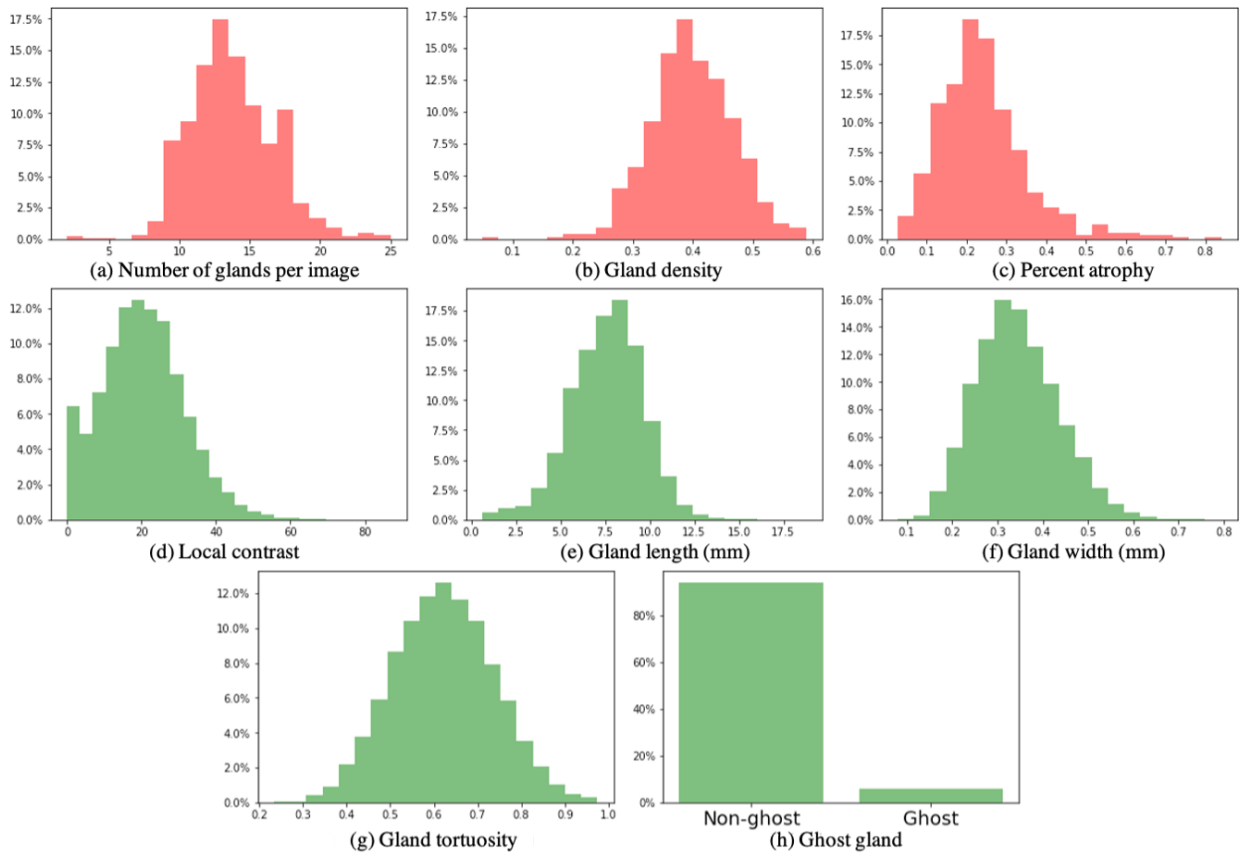
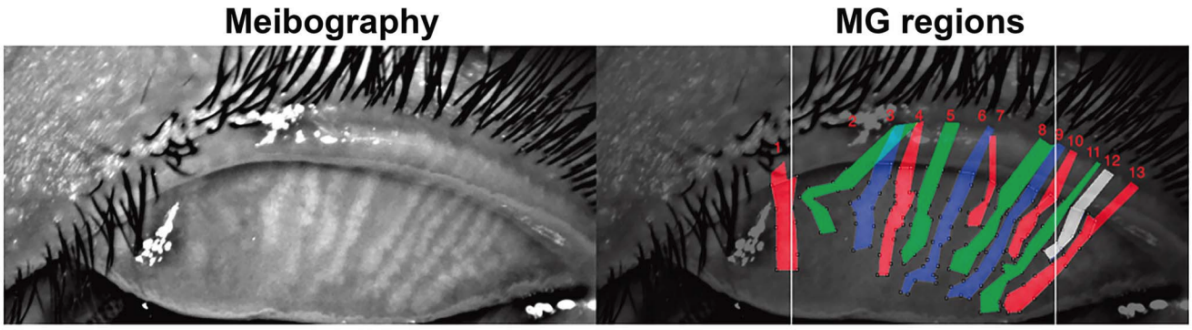


Figure 3.10: Histograms (in percentage) of various morphological features of our meibography image dataset.

it possible to automatically analyze a large number of meibography images within a short time and greatly improve efficiency in the clinical and research settings. Such evaluations currently would be prohibitively time consuming and without standardization for large-scale clinical trials or on-site ocular health screenings with large numbers of patients.

As with any emerging methodology, the proposed approach has certain limitations. Only glands in the central eyelid region were analyzed. Because the entire eyelid is a curved surface, imaging the central region of the tarsal plate with an optimal focus causes defocus of the peripheral glands. Additionally, incomplete eyelid eversion may also occlude far peripheral glands, making accurate identification of MG morphological features challenging. In terms of global morphological features, gland density and percent atrophy were not analyzed for the lower eyelids in this study due to the inability to accurately annotate lower eyelid boundaries for many subjects. The tarsal plate of the lower eyelid is about 5 mm in length [151], which is approximately half the length of the upper eyelid. Variability among patients in the eyelid area exposed by eversion of the lower lid makes it difficult to consistently determine its



	1	2	3	4	5	6	7	8	9	10	11	12	13
contrast	12.1	11.8	20.5	17.6	18.4	23.4	7.4	14.1	16.7	12.4	17.5	14.7	17.2
length (mm)	4.27	9.74	6.24	6.97	6.71	8.94	4.11	7.84	9.29	5.80	9.13	5.02	8.96
width (mm)	0.608	0.300	0.427	0.378	0.397	0.340	0.232	0.404	0.317	0.303	0.279	0.319	0.322
tortuosity (%)	15.9	49.0	25.4	20.9	22.1	34.9	44.9	26.3	34.0	26.9	34.2	24.0	32.5
ghost gland	1	1	0	1	0	0	1	0	0	0	0	0	0

Figure 3.11: Individual MG morphological features. Based on MG regions (upper right), individual MG morphological features (lower) can be measured. In the last row, 1 indicates a ghost gland while 0 indicates a non-ghost gland.

boundaries. It is also important to note that deep learning model performance is measured against, and ultimately depends upon, human expert annotations of the original meibography images. It is thus not possible to eliminate completely the subjective nature of the expertise needed to train the algorithm [279]. Finally, it should be noted that infrared imaging of the eyelids may or may not provide the best images of all MGs, as some researchers have found significant differences in detecting gland dropout between infrared imaging and other methods such as optical coherence tomography [341].

The scope of this study is restricted to reporting a new approach to evaluating morphological features in meibography without providing clinical understanding of what the variability of these features may indicate. Few other studies to date have attempted to investigate the associations among MG morphological features, especially local features, and basic physiological or ocular health outcomes. Some researchers have used the 0-3 (per eyelid) discrete ordinal meiboscore proposed by Arita et al.[12], or variations thereof [83]. These scores are generally based on global features such as percent area of the eyelid affected, and do not have fine resolution or consider local morphological features of individual MGs. With the proposed method, a great deal more detailed and accurate information about both the overall status and the fine-grained morphology of the MG array in the eyelid can be made available to researchers. Future work is warranted using the proposed methodology to study the relationships among MG morphological features, gland health and pathology, the quality and thickness of the tear lipid layer, tear film instability and dry eye.

In conclusion, a deep learning model to automatically segment individual MGs, analyze both global and local MG morphology and predict the presence of ghost glands in meibogra-

phy images has been developed. Given its accuracy and efficiency, the quantitative outputs (morphological features at the individual gland level) can potentially be helpful in furthering our understanding of the interplay between MG features and clinical signs and symptoms by analyzing large-scale image datasets [63]. Future work is warranted to improve MG segmentation performance for highly overlapping glands and peripheral glands in the nasal and temporal regions of the tarsal plate, and to expand our understanding of the relationships between MG morphology and various ocular pathologies.

Part II

Learning 3D-Geometry-Aware Representation

Chapter 4

Learning to Geometry-Aware Representations with Spatial Transformer for 3D Point Clouds

4.1 Introduction

3D computer vision has been on the rise with more advanced 3D sensors and computational algorithms. Depth cameras and LiDAR sensors output 3D point clouds, which become key components in several 3D computer vision tasks including but not limited to virtual / augmented reality [178, 244], 3D scene understanding [291, 295, 59], and autonomous driving [50, 174, 325].

On the algorithmic side, convolutional neural networks (CNNs) have achieved great success in many computer vision tasks [67, 161]. However, the concept of convolution cannot be directly applied to a point cloud, as 3D points are not pixels on a regular grid with regular neighbourhoods. One line of approaches is to convert the 3D point cloud into a representation where CNNs are readily applicable, e.g., a regular voxel representation [327, 250, 313] or 2D view projections [277, 239, 143, 355].

Another line of approaches is to develop network architectures that can directly process point clouds [238, 241, 247, 87]. Analogous to convolution on 2D pixels, convolution on 3D points needs to first identify *local neighborhoods* around individual input points. This step is achieved by computing the so-called *point affinity matrix*, i.e., the adjacency matrix of a dense graph constructed from the point cloud. These neighborhoods are then used for extracting features with point-wise convolutions. By stacking basic point convolution layers, a neural network can extract information from the point cloud with an increasing level of abstraction.

However, unlike images where 2D pixels are laid out on a regular grid with simple and well-defined local neighborhoods, local neighborhoods of 3D points are ill-defined and subject to various geometric transformations of 3D shapes. Most methods [238, 241, 317, 176]

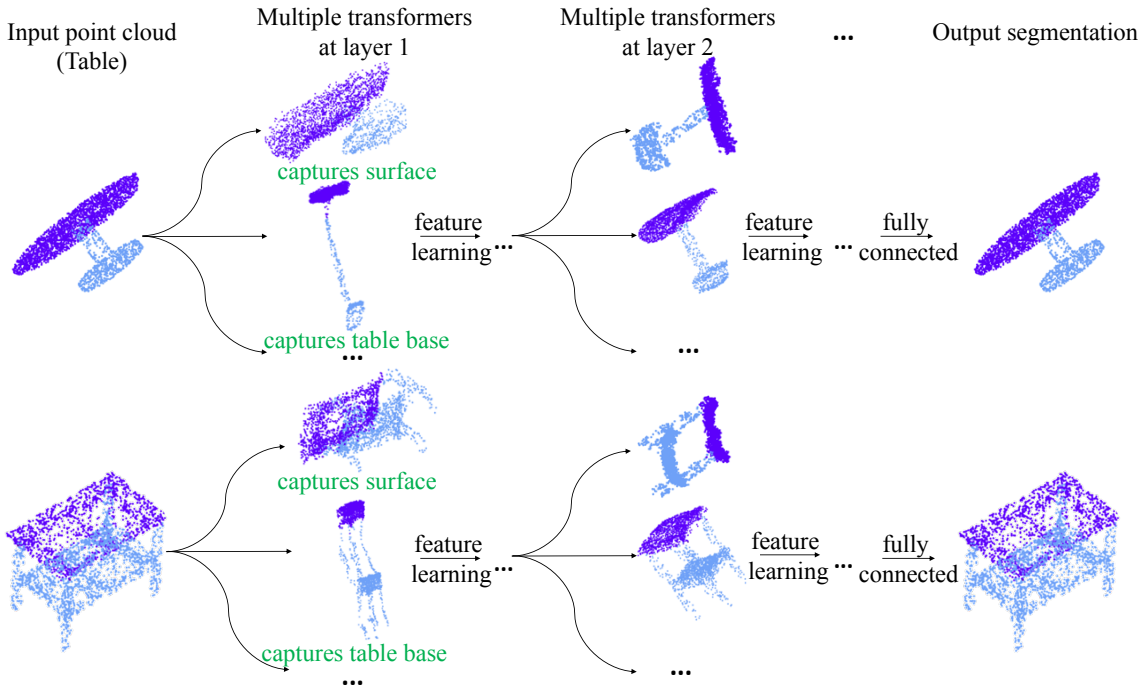


Figure 4.1: Spatial transformers learn several *global transformations* at each layer to obtain different local point neighborhoods. We show transformed point clouds at different layers learned by spatial transformers for different instances of a category (e.g. tables). Compared with previous works adopting fixed local neighborhoods, dynamic point neighborhoods make the network more powerful in learning semantics from point clouds. For example, corresponding geometric transformations capture similar semantic information even high intra-class spatial variations exist. The second transformation at layer 1 deforms different tables to be more semantically similar, and makes parsing the part of table base easier. Furthermore, the proposed transformer is a stand-alone module and can be easily added to existing point cloud processing networks.

define local neighborhoods as nearest neighbors in the Euclidean space of the input 3D point coordinates.

This common practice of defining a nearest neighbor graph according to the Euclidean distances on the fixed input 3D point coordinates may be simple but not optimal. First, such distances may not be able to efficiently encode semantics of 3D shapes, e.g., semantically or topologically far points might be spatially close in terms of the Euclidean distances. Secondly, fixed neighborhoods throughout the network may reduce the model's learning capacity as different layers capture information at different levels of abstraction, e.g., objects have a natural hierarchy and in order to segment out their parts, it would be more efficient to provide different layers the ability to parse them at different spatial scales.

We propose to address these fixed point neighbourhood restrictions by dynamically learning local neighborhoods and transforming the input point cloud at different layers. We use

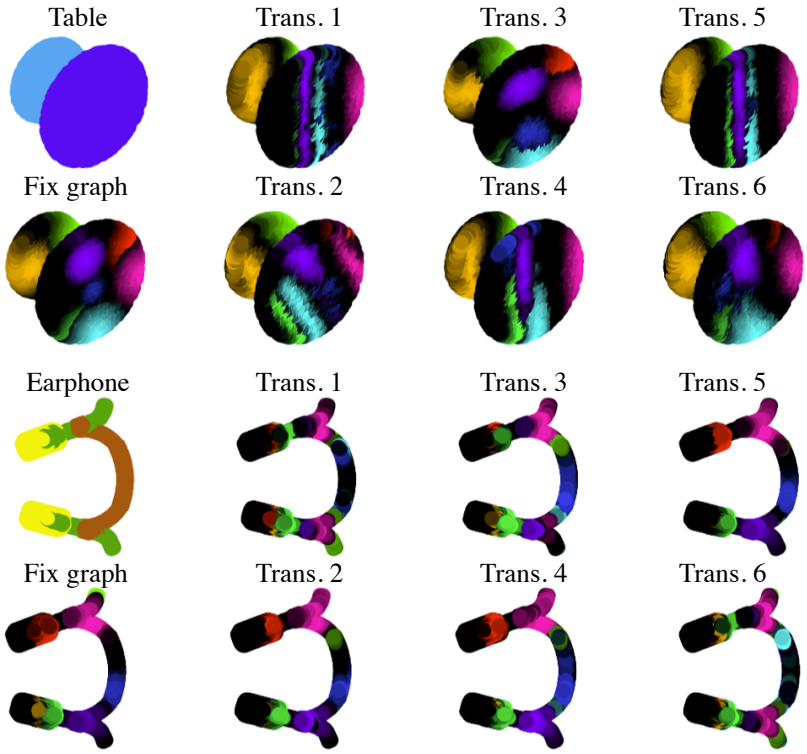


Figure 4.2: At each layer, we apply multiple spatial transformers to deform the input point cloud for learning different neighborhoods. We show local neighborhoods of input point cloud examples with and without deformable transformers. Different colors indicate different neighborhoods, and intensities indicate distances to the central point. The dynamic neighborhood enhance the network capacity to learn from objects with large spatial variations. Rotating table and earphone for better visualizations.

a parametric model that takes both point coordinates and learned features as inputs to learn the point affinity matrix. At different layers of the network, we learn several different transformations (dubbed as *spatial transformers* or *transformers* hereafter, Fig.4.1) and corresponding point local neighborhoods (Fig.4.2). Spatial transformers allow the network to adaptively learn point features covering different spatial extensions at each depth layer.

To spatially transform a point cloud, we learn a function, Φ, that generates transformed point coordinates from the input point coordinates and feature maps. However, it is nontrivial to learn Φ without smoothness constraints. Since any isometric (e.g. rigid) transformation cannot change the distance metric, we consider non-rigid transformations, both linear and non-linear families. That is, our *spatial transformers* are parameterized functions conditioned on the input point coordinates P and feature map F; they are subsequently used to transform the point coordinates, resulting in a new point affinity matrix for obtaining dynamic local neighborhoods.

We consider three families of spatial transformers. **1)** Affine transformation $P \mapsto AP$, where A is an affine matrix. **2)** Projective transformer $\widetilde{P} \mapsto B\widetilde{P}$, where $\widetilde{P}$ is 3D points expressed in homogeneous coordinates. **3)** Deformable transformer $P \mapsto CP + DF$, where C, D are respective transformation matrices of point coordinates and features F. The transformation depends on both the input point coordinates and the features the points assume.

Our work makes the following contributions.

- We propose to learn linear (affine) and non-linear (projective, deformable) spatial transformers for new point affinity matrices and thus dynamic local neighborhoods throughout the neural network.
- We demonstrate that our spatial transformers can be easily added to existing point cloud networks for a variety of tasks: classification, detection, and segmentation.
- We apply spatial transformers to various point cloud processing networks, with point-based and sampling-based metrics for point neighborhoods, and observe performance gains of dynamic graphs over fixed graphs.

4.2 Background

We discuss related works that motivate the necessity of our proposed spatial transformers.

View-based methods project 3D shapes to 2D planes and use images from multiple views as representations. Taking advantages of the power of CNNs in image processing [81, 105, 239, 277], view-based methods achieve reasonable 3D processing performance. However, certain information about 3D structures gets lost when 3D points are projected to 2D image planes; occluded surfaces and density variations are thus often troublesome for these methods.

Voxel-based methods represent 3D shapes as volumetric data on a regular 3D grid, and proceed with 3D convolution [327, 198, 283]. Their caveates are quantization artifacts, inefficient usage of 3D voxels, and low spatial resolutions due to a large memory requirement. In addition, 3D convolutions are not biased towards surface property extraction and thus cannot capture geometrical and semantic information efficiently. Recent works that apply different partition strategies [154, 250, 283, 313] relieve these issues but depend heavily on bounding volume subdivision instead of local geometric shapes. In contract, our method works directly on the 3D point cloud, minimizing geometric information loss and maximizing processing efficiency.

Point cloud processing methods take a point cloud as the input and extract semantic information by point convolutions. PointNet [238] directly learns the embedding of every 3D point in isolation and gather that information by pooling point features later on. Although it achieves good performance at the time, PointNet does not learn any 3D local shape information since each local neighborhood contains only one point. PointNet++ [241] addresses this caveate by adopting a hierarchical application of isolated 3D point feature learning to

multiple subsets of a point cloud. Many other works also explore different strategies for leveraging local structure learning from point clouds [317, 176]. Instead of finding neighbors of each point, SplatNet [278] encodes local structures from the sampling perspective: it groups points based on permutohedral lattices [3], and then applies bilateral convolution [141] for feature learning. Super-point graphs [165] partition a point cloud into super-points and learn the 3D point geometric organization. Many works focus on designing novel point convolutions given 3D point local neighborhoods [241, 278, 176], ignoring how the local neighborhoods should be formed.

Unlike pixels in a 2D image, points in a 3D point cloud are un-ordered, with irregular and heterogeneous neighborhoods; regular convolution operations thus cannot be applied. Many works [317, 176, 314, 284, 351, 172] aim to design point convolution operations that resemble regular 2D convolutions. Fixed input point coordinates are used to define local neighborhoods for the point convolution, resulting in the same local neighbourhoods at different layers that limit the model's processing power. In contrast, our work uses spatial transformers at each layer to learn dynamic local neighborhoods in a more adaptive, flexible, and efficient way.

Spatial Transformations. The idea of enabling spatial transformation in neural networks has been explored for 2D image understanding [139]. It is natural to extend the idea to 3D point clouds. PointNet [238] adopts a rigid transformation module on the input point cloud to factor out the object pose and improve classification accuracy. KPConv [287] applies *local deformation* in the neighborhood of point convolution to enhance its learning capacity. In contrast, our work learns several different global transformations to apply on the input point cloud at each layer for dynamic neighborhoods.

4.3 Methods

We first briefly review different geometric transformation methods and their influence on the affinity matrix of point cloud data, then describe the design of our three *spatial transformers*, namely, **(a)** affine, **(b)** projective and **(c)** deformable. We apply the *spatial transformer block*, consisting of multiple spatial transformers, to each layer of a network for altering local neighborhoods for better point feature learning. We conclude the section by introducing how the transformers can be added to existing point cloud processing networks and the relevance to other works.

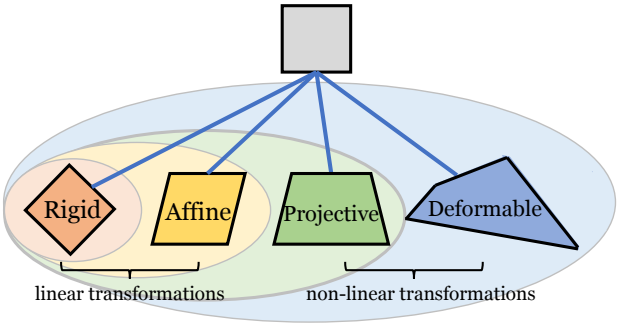


Figure 4.3: Geometric transformations. We illustrate how a grey square transforms after rigid, affine, projective and deformable transformations.

4.3.1 Geometric Transformations

We propose to learn transformations on the input point cloud to *deform* its geometric shape, and alter local neighborhoods with new point affinity matrices. The hypothesis behind the usage of geometric transformation is as follows:

Hypothesis 1. *Let $P = \{\mathbf{p}_i\}$ be the input point cloud and let $\mathcal{N}_i$ be the local neighborhood around $\mathbf{p}_i \in \mathbf{R}^3$ from which we extract local features. Let $\mathcal{N} = \{\mathcal{N}_i\}$ be the set of local neighborhoods. Assume $\widetilde{\mathcal{N}} = \left\{\widetilde{\mathcal{N}}_i\right\}$ be the optimal neighborhood for learning local features, then $\exists$(smooth) $\Phi : \mathcal{N}_i \to \widetilde{\mathcal{N}}_i$ for all $\mathbf{p}_i$.*

Essentially we are going to use different types of geometric transformations to approximate Φ. The new learned affinity matrix will dynamically alter local neighborhoods to allow better feature learning.

Illustrated in Fig.4.3, transformations can be categorized into rigid and non-rigid transformations, and the latter can be further categorized into linear and non-linear transformations. We now discuss different spatial transformations.

Rigid Transformations. The group of rigid transformations consist of translations and rotations. However, rigid transformations are isometric (in ℓ_2 distance) and therefore preserves the point affinity matrix. Thus, local neighborhoods are *invariant* to rigid transformations in terms of k-NN graphs. Hence, we do not consider rigid transformations.

Affine Transformations. Affine transformations belong to non-rigid linear transformations. Consider a 3D point cloud $P = \{\mathbf{p}_i\}_{i=1}^N \subset \mathbf{R}^3$ consisting of N three-dimensional vectors $\mathbf{p}_i \in \mathbf{R}^3$. Then, an affine transformation can be parameterized by an invertible matrix $A \in \mathbf{R}^{3\times3}$ and a translation vector $\mathbf{b} \in \mathbf{R}^3$. Given $A, \mathbf{b}$, the affine transformed coordinates $\mathbf{p}_i$ can be written as $\mathbf{p}_i \mapsto A\mathbf{p}_i + \mathbf{b}$. Note that translation $\mathbf{b}$ does not change the point affinity matrix and point neighborhoods. Recall that an affine transformation preserves collinearity, parallelism, and convexity.

Projective Transformations. Projective transformations are non-rigid non-linear transformations. We first map the 3D point cloud P to the homogeneous space and get $\widetilde{P}$, by appending one-vectors to the last dimension. The projective transformation is parameterized by $A \in \mathbf{R}^{4\times4}$ and the transformed point $\widetilde{\mathbf{p}}_i \mapsto A\widetilde{\mathbf{p}}_i$. Compared to the affine transformations, projective transformations have more degrees of freedom but cannot preserve parallelism. Projective transformations preserve collinearity and incidence, hence fail to capture all possible deformations. For example, points lying on the same line will always be mapped to a line, and this constraint may be overly restrictive. It is of interest to be able to break this constraint if these points are from different semantic categories. A more general transformation that covers various deformations may be more effective.

Deformable Transformations. When all the points have the freedom to move without much constraint, the 3D shape can deform freely. We refer to this general spatial transformation as a *deformable transformation*. It has more degrees of freedom and does not necessarily preserve the topology.

Learning new per-point offsets would be computationally hard and costly, we thus use a parametric offset model instead. Taking both point coordinates and features as inputs, the model would learn offsets dependent upon both spatial and feature representations of the input point cloud.

4.3.2 Spatial Transformers for 3D Point Clouds

Our so-called *spatial transformer* method applies a geometric transformation to the input point cloud to obtain different local neighborhoods for feature learning. It can be applied to existing point cloud processing networks as spatial transformers only alter local neighborhoods.

Suppose at layer t, the spatial transformer block contains $k^{(t)}$ transformers. Each transformer learns a transformation to apply to the input point coordinates. We refer to the transformed points as nodes of a *sub-graph* and their feature on it the corresponding sub-feature. We then concatenate all sub-features from these transformers to form the final output of the learning block. Suppose that the i^{th} spatial transformer at the t^{th} layer takes as input the original point cloud $P \in \mathbf{R}^{3\times N}$ and previous feature map $\mathcal{F}^{(t-1)} \in \mathbf{R}^{f^{(t-1)}\times N}$.
Affine. We form $k^{(t)}$ new transformed point from $\mathbf{p}_j$ as:

$$\mathbf{g}_{i,j}^{(t)} = A_i^{(t)}\mathbf{p}_j + \mathbf{b}_i^{(t)}, \quad i = 1, 2, ..., k^{(t)}. \tag{4.1}$$

Since the point affinity matrix is invariant under uniform scaling and translation, we set $\|A_i\|_F = 1, b = 0$, for all i. Thus, with $G_i^{(t)} = \left\{\mathbf{g}_{i,j}^{(t)}\right\}_j$, we simplify Equation 4.1 as:

$$G_i^{(t)} = A_i^{(t)}P, \quad i = 1, 2, \cdots, k^{(t)}. \tag{4.2}$$

We compute the k nearest neighbours of each transformed point $G_i^{(t)}$ and obtain the point affinity matrix $S_i^{(t)}$, based on which we define local neighborhoods and apply point convolutions on previous point cloud feature map $\mathcal{F}^{(t-1)}$. We get the point cloud feature $F_i^{(t)} \in \mathbf{R}^{f_i^{(t)}\times N}$ of the sub-graph from i-th transformation and its altered neighborhoods:

$$F_i^{(t)} = \mathsf{CONV}(\mathcal{F}^{(t-1)}, S_i^{(t)}, k), \quad i = 1, 2, ..., k^{(t)}, \tag{4.3}$$

where CONV denotes the point convolution: It takes **(a)** previous point cloud features, **(b)** the affinity matrix (for defining local neighborhoods of every point) and **(c)** the number of neighbors (for defining the size of neighborhoods) as inputs.

In point convolutions such as [317], the point affinity matrix changes the input feature in a non-differentiable way. Therefore, we append the transformed point cloud $P_i^{(t)}$ to the input feature for the sake of back-propagating the transformation matrix A. In sampling-based convolutions such as bilateral convolution [278], the point affinity matrix changes the input feature in a differentiable way; no additional operation is needed.

For all the $k^{(t)}$ sub-graph in layer/ block t, we learn $k^{(t)}$ point cloud features $F_i^{(t)}$. The output of this module is the concatenation of all the sub-graph point cloud features:

$$\mathcal{F}^{(t)} = \mathsf{CONCAT}(F_1^{(t)}, F_2^{(t)}, ..., F_{k^{(t)}}^{(t)}), \tag{4.4}$$

where $F_i^{(t)} \in \mathbf{R}^{f_i^{(t)} \times N}$ and $\mathfrak{f}^{(t)} = \sum_i^{k^{(t)}} f_i^{(t)}$, $\mathcal{F}^{(t)} \in \mathbf{R}^{\mathfrak{f}^{(t)} \times N}$. In our implementation, we randomly initialize A from the standard normal distribution $\mathcal{N}(0, 1)$. Before computing the coordinates of the transformed point cloud, we normalize A by its norm $\|A\|_F$, as the point affinity matrix is invariant under uniform scaling.

Projective. Analogous to the affine spatial transformer, for the i^{th} graph at t^{th} layer, we apply a projective transformation to the point cloud $\widetilde{P}$ in homogeneous coordinates and get the transformed point cloud as:

$$\widetilde{G}_i^{(t)} = B_i^{(t)} \widetilde{P}, \;\; i = 1, 2, \cdots, k^{(t)}, \tag{4.5}$$

where $B_i^{(t)} \in \mathbf{R}^{4\times 4}$ is the transformation matrix in homogeneous coordinates. We then follow the same procedure as in Equations 4.3 and 4.4 to get the output feature $\mathcal{F}^t$.

Deformable. Affine and projective transformations can transform the input point cloud, alter the point affinity matrix, and provide learnable local neighborhoods for point convolutions at different layers. However, they are limited as affine transformations are linear and projective transformations map lines to lines only. We define a non-linear deformable spatial transformer at the t^{th} layer and i^{th} sub-graph as

$$G_i^{(t)} = A_i^{(t)} P + D_i^{(t)}, \tag{4.6}$$

where $A_i^{(t)} P$ is the affine transformation component and $D_i^{(t)} \in \mathbf{R}^{3\times N}$ gives every point additional freedom to move, so the point cloud has the flexibility to deform its shape. Note that the translation vector $\mathbf{b}$ in Equation 4.1 is a special case of the deformation matrix $D_i^{(t)}$. In general, the deformation matrix $D_i^{(t)}$ can significantly change local neighborhoods.

The spatial transformer parameters are learned in an end-to-end fashion from both point cloud coordinates and features. Since affine transformation $A_i^{(t)} P$ is dependent on spatial locations, we let the deformation matrix $D_i^{(t)}$ depend on the features: $D_i^{(t)} = \mathcal{C}_i^{(t)} \mathcal{F}^{(t-1)}$, where $\mathcal{C}_i^{(t)} \in \mathbf{R}^{3\times \mathfrak{f}}$ transforms the previous layer feature $\mathcal{F}^{(t-1)} \in \mathbf{R}^{\mathfrak{f}\times N}$ from $\mathbf{R}^{\mathfrak{f}}$ to $\mathbf{R}^3$. Hence, the deformable transformation in Equation 4.6 can thus be simplified as:

$$G_i^{(t)} = \begin{bmatrix} A_i^{(t)} & \mathcal{C}_i^{(t)} \end{bmatrix} \begin{bmatrix} P \\ \mathcal{F}^{(t-1)} \end{bmatrix} = C_i^{(t)} \begin{bmatrix} P \\ \mathcal{F}^{(t-1)} \end{bmatrix}, \tag{4.7}$$

where $C_i^{(t)} \in \mathbf{R}^{3\times(3+f^{(t-1)})}$ is the concatenation of affine and deformable transformation matrix that captures both point cloud coordinates and features.

After we compute the transformed point coordinates $G^{(t)}$, we follow Equations 4.3 and 4.4 to learn the feature of each transformed sub-graph and concatenate them as the final output feature of layer t.

Our deformable spatial transformer has two parts: $A_i^{(t)}P$ and $\mathcal{C}_i^{(t)}\mathcal{F}^{(t-1)}$, for a linear transformation of 3D spatial coordinates and a nonlinear transformation of point features (which reflect semantics) respectively. In Section 4.4.5, we provide empirical analysis of these two components.

4.3.3 Spatial Transformer Networks

We spatially transform the input point cloud in order to obtain dynamic local neighborhoods for point convolutions. The transformer can be easily added to existing point cloud processing networks. We first describe the procedure and then provide three applications with several networks.

Point Cloud Networks with Spatial Transformers. Consider segmenting N 3D points into C classes as an example. Fig.4.4 depicts a general network architecture for point cloud segmentation, where several spatial transformers are used at different layers. At layer t, we learn $k^{(t)}$ transformation matrices $\{A_i^{(t)}\}_{i=1}^{k^{(t)}}$, apply each to the input point coordinates P, and then compute the point affinity matrices $\{S_i^{(t)}\}_{i=1}^{k^{(t)}}$, e.g., based on k-NN graphs for the edge convolution [317].

For each sub-transformation, we learn a feature $F_i^{(t)}$ of dimension $N \times f_i^{(t)}$. We then concatenate all $k^{(t)}$ features at this layer to form an output feature $\mathcal{F}^t$ of dimension $N \times \mathfrak{f}^{(t)}$, where $\mathfrak{f}^{(t)} = \sum_i^{k^{(t)}} f_i^{(t)}$. The output feature serves as the input to the next layer for further feature learning.

Note that affine or projective transformation matrices are applied to the original point cloud coordinates P, since each layer has not just one but multiple spatial transformers. However, the deformable transformation matrix $\mathcal{C}_i^{(t)}$ is applied to the previous feature map, the feature transformation component is thus progressively learned.

By stacking several such transformation learning blocks and finally a fully connected layer of dimension C, we can map the input point cloud to the segmentation map of dimension $C \times N$, or downsample to a vector of dimension C for classification tasks. For the spatial transformer block in a point cloud detection network (Fig.4.5), C is the dimension of the output feature. We train the network end-to-end.

Classification Networks. A point cloud classifier [241, 176] takes 3D points, learns features from their local neighborhoods, and outputs C classification scores, where C is the number of classes. We add spatial transformers at each layer to obtain different local neighborhoods for feature learning.

Point-based Segmentation Networks. These networks [241, 238, 176, 317] take 3D points and compute their point affinity matrices and local neighborhoods from the point coordinates. Features are learned by applying convolution operators on the points and their local neighborhoods.

We use the *edge convolution* in [317] as our baseline, which takes relative point coordinates as inputs and achieves the state-of-the-art performance. Specifically, we retain their learning

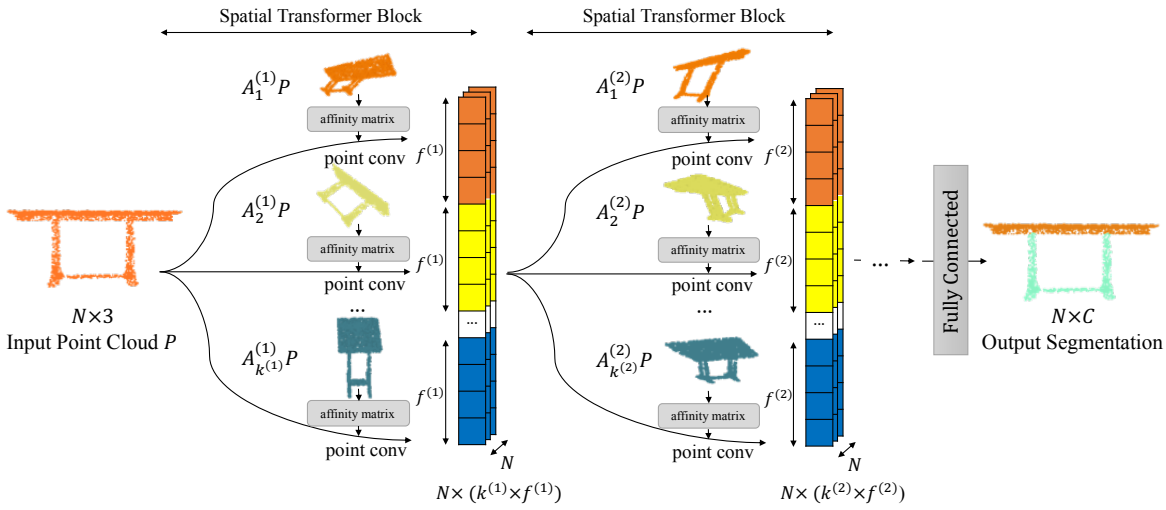


Figure 4.4: The point cloud segmentation network with spatial transformers. Our network consists of several spatial transformers. At each layer, we learn k transformation matrices A to apply to the input point cloud P, and compute the corresponding point affinity matrices based on their k-NN graphs. For each sub transformation, we can learn a sub-feature, and then concatenate all features to form an output feature of dimension $f \times N$. The output feature will be used for the next spatial transformer block for feature learning. By stacking several such transformation learning blocks and finally a fully connected layer of dimension C (the number of class), we can map the input point cloud to the $C \times N$ segmentation map.

settings and simply insert spatial transformers to generate new local neighborhoods for the edge convolutions.

Sampling-based Segmentation Networks. To demonstrate the general applicability of our spatial transformers, we consider point affinity matrices on transformed point clouds as defined in sampling-based networks such as SplatNet [278].

SplatNet groups 3D points onto a permutohedral lattice [3] and applies bilateral filters [141] on the grouped points to get features. The permutohedral lattice defines the local neighborhoods of every point and makes the bilateral convolution possible. We add spatial transformers to deform the point cloud and form various new lattices. The local neighborhoods can dynamically configure for learning point cloud semantics. We keep all the other settings of SplatNet.

Detection Networks. Detecting objects in a 3D point cloud generated from e.g. LiDAR sensors is important for autonomous navigation, housekeeping robots, and AR/ VR. These 3D points are often sparse and imbalanced across semantic classes. Our spatial transformers can be added to a detection network and improve feature learning efficiency and task performance with dynamic local neighborhoods.

Our baseline is VoxelNet [357], the state-of-the-art 3D object detector for autonomous driving data. We adopt all its settings, and add spatial transformers on the raw point cloud

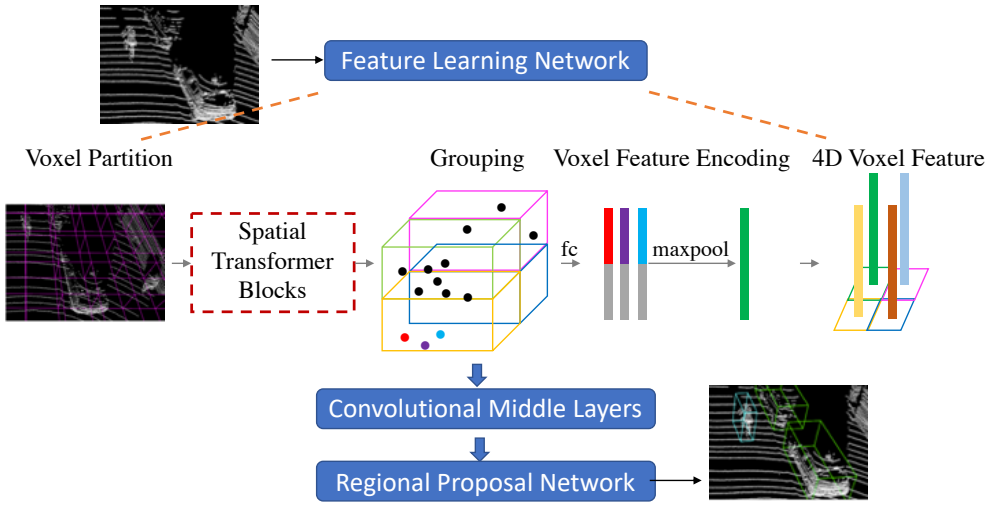


Figure 4.5: The object detection network. We add spatial transformers to the the point feature learning network of [357] for obtaining dynamic local neighborhoods. Transformers only affect feature learning but not point coordinates for grouping.

data, before point grouping (Fig.4.5). To demonstrate that spatial transformers enhance feature learning for point cloud processing, we let transformers only affect point features but not point coordinates for grouping. With spatial transformers, point coordinates could also be transformed at the grouping stage, which would lead to non-cuboid 3D detection boxes. Although interesting, we do not explore this variation and deem it beyond the scope of this paper.

4.3.4 Relevance to Other Works

We review related works on deformable convolutions [61, 287] and DGCNN [317].

Deformable Convolutions. Deformable convolutional networks [61] learn dynamic local neighborhoods for 2D images. Specifically, at each location $\mathbf{p_0}$ of the output feature map Y, deformable convolutions modify the regular grid R with offsets $\{\Delta\mathbf{p_n}\}_{n=1}^{N}$, where $N = |R|$. The output on input X by convolution with weight $\mathbf{w}$ becomes:

$$Y(\mathbf{p_0}) = \sum_{\mathbf{p_n} \in \mathbf{R}^3} w(\mathbf{p_n}) X(\mathbf{p_0} + \mathbf{p_n} + \Delta\mathbf{p_n}) \tag{4.8}$$

Note that KPConv [287] directly adapts this formula to point clouds as deformable point convolutions. Although also achieving dynamic local neighborhoods, our spatial transformers alter neighborhoods differently:

1. Deformable point convolutions learn to alter each neighborhood with an offset to the regular grid R. We learn global transformations on the input point cloud and the metric of

defining local neighborhoods changes. Global transformations like affine transformations can retain the global geometric properties such as collinearity and parallelism, while local transformations has no such constraints as only local neighborhoods are available.

2. Offsets of deformable point convolutions are dependent upon feature values, while transformation matrices of our spatial transformers are dependent upon point coordinates for affine and projective transformations, or both point coordinates and feature values for deformable transformations. Access to point coordinates provides additional information and regularzation.

Dynamic Graph CNN. Dynamic local neighborhoods have also been explored in DGCNN [317] for point cloud processing. It has three main differences with our work.

1. How neighborhoods are defined is different. DGCNN uses high-dimensional feature maps to construct the point affinity matrix and generate local neighborhoods. Our local neighborhoods are from transformed point clouds. Reusing point features for defining neighborhoods may be straightforward, but reduce the distinction between spatial and semantic information and hurt generalization.

2. It is computationally costly to build dense nearest neighbor graphs in a high-dimensional feature space.

3. DGCNN [317] uses only one nearest neighbor graph at different layers, whereas we have multiple graphs at each layer for capturing different geometric transformations.

With less computational cost and more flexibility in geometric transformations, we achieve better empirical performance on semantic segmentation (Table 4.2 and Table 4.3).

4.4 Experiments

We conduct comprehensive experiments to verify the effectiveness of our spatial transformers. We benchmark with two types of networks, point-based and sampling-based metrics for defining point neighborhoods, on four point cloud processing tasks: classification, part segmentation, semantic segmentation and detection. We conduct ablation studies on deformable spatial transformers. We further provide visualization, analysis and insights of our method.

4.4.1 Classification

We benchmark on ModelNet40 3D shape classification [327]. We add transformers to two baselines [317, 278] and adopt the same network architecture, experimental setting and evaluation protocols. Table 4.1 and Fig.4.6 show that adding spatial transformers to point-based and sampling-based method gives 1% and 2% gain.

Table 4.1: Spatial transformers improves ModelNet40 classification accuracy. We report ModelNet40 classification accuracy of different baselines, without and with spatial transformers, with or without random rotations. *Point-based* refers to the baseline method adopting Euclidean-distance based affinity matrices [317]. *Sampling-based* refers to the baseline method adopting permutohedral-lattice based affinity matrices [278]. We observe accuracy gains for different baseline networks with spatial transformers. Transformer gains are invariant to input rotations.

			Point-based				Point-based with rand. input rotations		Samplin-based			
	PointNet [238]	DGCNN [317]	[317] (fixed)	Affine	Proj.	Deformable	[317] (fixed)	Deformable	SplatNet[278]	Affine	Proj.	Deformable
Avg.	86.2	89.2	88.8	89.3	89.2	**89.9**	85.7	**88.3**	86.3	87.4	87.1	**88.6**

In addition, our performance gain over [317], which builds one per-layer dynamic neighborhood graphs with high-dimensional point features, demonstrates the advantages of our method of building multiple dynamic neighborhood graphs with transformed 3D point coordinates.

Fig.4.12 shows that spatial transformers align the 3D shape better according to its semantics. We augment training and testing data with random rotations, and observe that spatial transformers gain 3% over its fixed graph counterpart.

4.4.2 Part Segmentation

We benchmark on ShapeNet part segmentation [45], where the goal is to assign a part category label (e.g. chair leg, cup handle) to each 3D point. The dataset contains 16,881 shapes from 16 categories, annotated with 50 parts in total, and the number of parts per category ranges from 2 to 6.

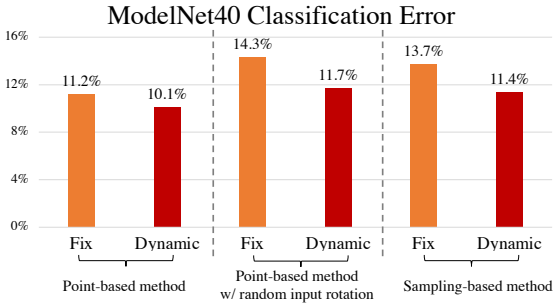


Figure 4.6: Spatial transformers lead to higher accuracy and more rotation invariance on ModelNet40. We report classification errors for different baselines, without and with spatial transformers, with or without random rotations. Transformers consistently lead to the lower errors than fixed graph baselines, and the improvement is larger upon random rotations.

Point-based Method

Network Architectures. *Point-based* methods construct neighborhoods based on point coordinate operations such as *edge convolution* for our baseline DGCNN [317]. We follow the same network architecture and evaluation protocols of [317]. The network has 3 convolutional layers; the output feature dimension is 64. To capture information at different levels, all the convolutional features are concatenated and fed through several fully connected layers to output the segmentation.

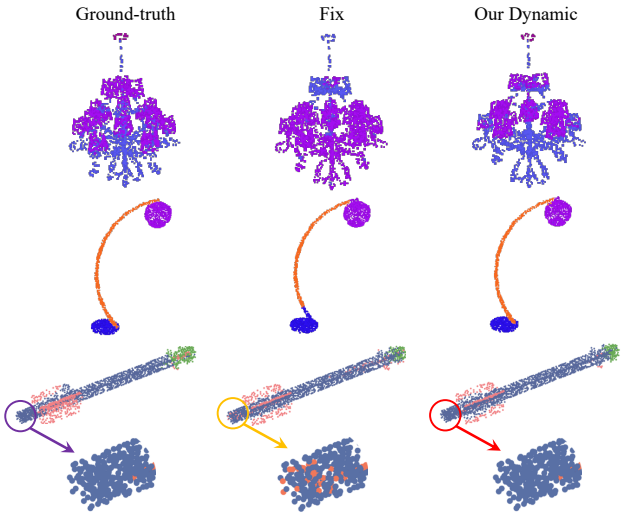


Figure 4.7: Spatial transformers improve the part segmentation performance. We show part segmentation results of different baselines, where different parts are marked with different colors. With spatial transformers, part segmentation for objects with less rigid and more complicated structures improves (1st and 2nd row, lamp). The segmentation consistency within each part also improves (3rd row, rocket).

As a *fixed graph* baseline, we use the same input point coordinates as the metric to define fixed local neighborhoods. We insert spatial transformers to alter the metric for defining point neighborhoods for edge convolutions. There are point-based *affine*, *projective* and *deformable* networks when inserting different spatial transformers (Section 4.3.2). As for classification, [317] directly uses learned features to build point affinity matrices for dynamic neighborhoods.

We follow [317] and use three edge convolution layers. At each layer, we keep the number of graphs k and sub-graph feature dimension f the same, and search for the best architecture. We report results of affine, projective and deformable networks with $k=4$, $f=!32$. For fair comparisons, we increase the number of channels of baselines so all the methods have the same number of parameters.

Results and Analyses. In Table 4.2, we report the instance average mIOU (mean intersection over union), as well as the mIOU of some representative categories in ShapeNet. Compared with the fixed graph baseline, the affine, projective and deformable spatial transformers achieve 0.5%, 0.2% and 1.1% improvement respectively and beats the fixed graph baseline methods in most categories. Specifically, we observe 8.0%, 8.3% and 4.7% performance boost with spatial transformers over the fixed graph baseline. Our deformable spatial transformers gain 4.0% over [317].

We also beat other state-of-the-art methods [238, 241, 247] by a significant margin. Adding deformable spatial transformers to PointCNN [176] gains 6% (4%) on motorbike (bag) and 1% on average. We observe that categories with fewer samples are more likely to

Table 4.2: Spatial transformers improve part segmentation performance. We report mIoU(%) on ShapeNet PartSeg dataset. Compared with several other methods, deformable spatial transformers achieve the SOTA in average mIoU.

	Avg.	aero	bag	cap	car	chair	earphone	guitar	knife	lamp	laptop	motorbike	mug	pistol	rocket	skateboard	table
# shapes		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
3DCNN [238]	79.4	75.1	72.8	73.3	70.0	87.2	63.5	88.4	79.6	74.4	93.9	58.7	91.8	76.4	51.2	65.3	77.1
PointNet[238]	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PointNet++ [241]	85.0	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	**82.6**
FCPN [247]	81.3	84.0	82.8	86.4	**88.3**	83.3	73.6	**93.4**	87.4	77.4	**97.7**	**81.4**	95.8	**87.7**	**68.4**	**83.6**	73.4
DGCNN [317]	81.3	84.0	82.8	86.4	78.0	**90.9**	76.8	91.1	87.4	83.0	95.7	66.2	94.7	80.3	58.7	74.2	80.1
Point-based [317] fixed graph	84.2	83.7	82.4	84.0	78.2	**90.9**	69.9	91.3	86.6	82.5	95.8	66.5	94.0	80.8	56.0	73.8	79.8
Point-based affine	84.7	84.1	83.5	86.9	79.6	**90.9**	72.5	91.6	88.2	83.3	96.1	68.9	95.3	83.3	60.9	75.2	79.7
Point-based projective	84.4	84.3	84.2	88.5	77.9	90.4	72.8	91.2	86.6	81.7	96.0	66.6	94.8	81.3	61.6	72.1	80.5
Point-based deformable	85.3	**84.6**	83.3	88.7	79.4	**90.9**	77.9	91.7	87.6	83.5	96.0	68.8	95.2	82.4	64.3	76.3	81.5
Point-based deformable random	84.7	84.3	84.4	83.2	78.9	90.8	75.6	91.4	87.1	83.0	95.9	66.8	94.8	82.1	62.3	75.7	80.4
PointCNN [176]	84.9	82.7	82.8	82.5	80.0	90.1	75.8	91.3	**87.8**	82.6	95.7	69.8	93.6	81.1	61.5	80.1	81.9
PointCNN deformable	**85.8**	83.4	**86.6**	85.5	79.1	90.3	**78.5**	91.6	**87.8**	84.2	95.8	75.3	94.6	83.3	65.0	80.7	81.7
Sampling-based baseline [278]	84.6	81.9	83.9	88.6	79.5	90.1	73.5	91.3	84.7	84.5	96.3	69.7	95.0	81.7	59.2	70.4	81.3
Sampling-based projective	84.4	82.1	84.0	**89.1**	77.9	89.6	73.7	91.1	83.3	83.0	96.3	67.2	94.5	79.8	60.0	68.8	82.1
Sampling-based deformable	85.2	82.9	83.8	87.6	79.6	90.6	73.0	92.2	86.1	**85.7**	96.3	72.7	**95.8**	83.1	65.1	76.5	81.3

gain possibly due to regularization by transformers. Fig.4.7 shows that deformable spatial transformers make more smooth predictions and achieve better performance than the fixed graph baseline.

From affine to deformable transformations, the performance increases as the degree of freedom increases for the transformer. Projective transformers, however, perform slightly worse than affine transformers. The performance drop could result from geometrical distortion caused by mapping 3D points with homogeneous coordinates. Furthermore, for deformable transformers, when removing the constraint that the transformed points should be similar to the input point cloud (Fig.4.8, feature only $G = CF$), the performance also drops, indicating the necessity of the proposed similar-to-input constraint on spatial transformers.

Sampling-based Method

Network Architectures. *Sampling-based* methods construct neighborhoods are based on sampling operations on point coordinates. SplatNet [278] groups points on permutohedral lattices and applies learned bilateral filters [141] on naturally defined local neighbors to extract features. We follow the same architecture as SplatNet [278]. The network starts with a single 1×1 regular convolutional layer, followed by 5 bilateral convolution layers (BCL). The output of all BCL are concatenated and fed to a final 1×1 regular convolutional layer to get the segmentation output. Since each BCL directly takes raw point locations, we consider it as a fixed graph baseline. We add deformable spatial transformers to the network and feed transformed point graphs to BCL to construct permutohedral lattices. With gradients on the permutohedral lattice grid, we can make the transformation matrix learned end-to-end. Note that we increase the channel of convolution layers for fair comparisons.

Results and Analyses. Table 4.2 shows that our deformable spatial transformers (with

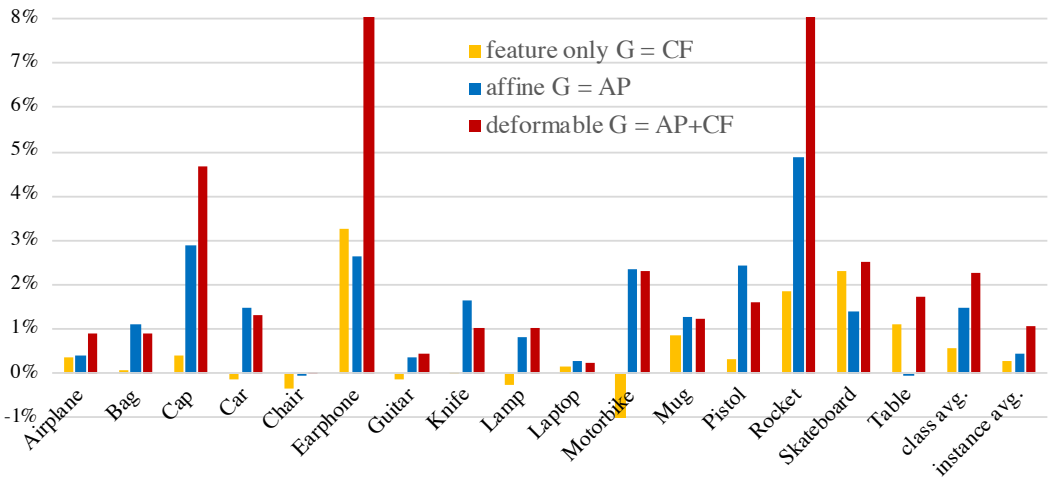


Figure 4.8: Transforming both point cloud coordinates and features for dynamic local neighborhoods leads to the largest gain. We report different parts of deformable transformers' performance gain over fixed local neighborhood baseline on ShapeNet part segmentation. 0% means achieving the same accuracy as fixed local neighborhood baseline and negative value means achieving worse accuracy than fixed neighborhood baseline. Compared with preserving either affine part AP or feature part CF, deformable spatial transformers ($AP + CF$) achieves largest gains on every category, specifically 8% gains on earphone and rocket.

$k = 1$ at all BCLs) gains over the sampling-based fixed graph baseline [278] in most categories with 0.6% on average and 5.9% for the rocket category. It also beats other state-of-the-art baselines.

4.4.3 Semantic Segmentation

We benchmark on the Stanford 3D semantic parsing dataset [15]. It contains 3D scans by Matterport covering 6 areas and 271 rooms. Each point is annotated into one of 13 categories such as *chair, table, floor, clutter*. We follow the data processing procedure of [238]: We first split points by room, and then sample rooms into several 1m $\times$1m blocks. When training, 4096 points are sampled from the block on the fly. We train our network to predict the point class in each block, where each point is represented by 9 values: XYZ, RGB and its [0,1]-normalized location with respect to the room.

Network Architectures. We adopt DGCNN [317] as Section 4.4.2, with $C = 13$, the number of semantic categories.

Results and Analyses. In terms of average mIoU, Table 4.3 shows that affine and deformable spatial transformers gain 0.9% and 1.2% respectively over the fixed graph baseline. Deformable transformers also gain 1.1% over [317] and beat all other state-of-the-art methods. Likewise for sampling-based methods [278], we observe 1.4% gain.

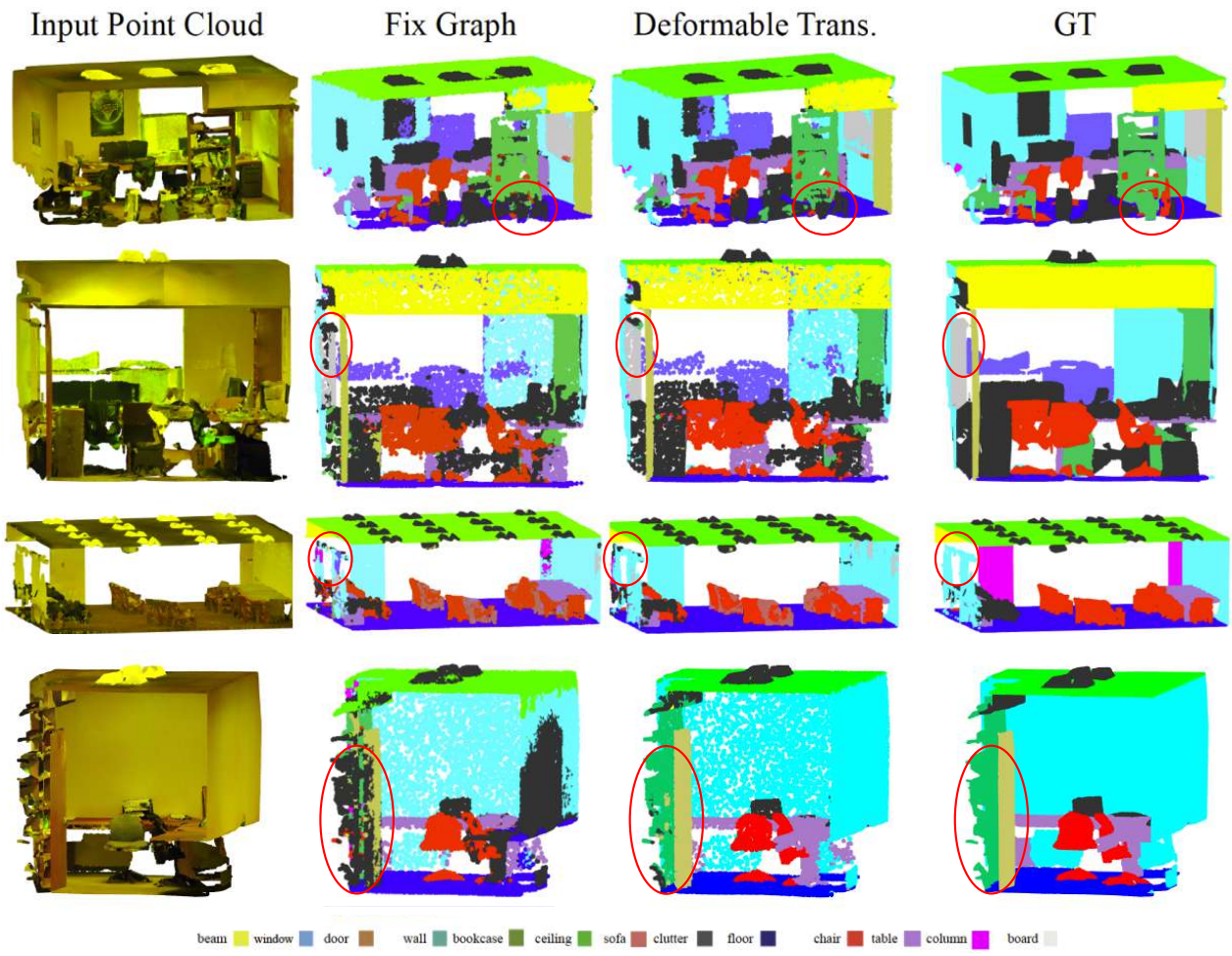


Figure 4.9: Spatial transformers improve semantic segmentation results. We show qualitative visualizations for semantic segmentation of deformable spatial transformers and the fixed local neighborhood baseline. The first column is the input point cloud, the second and the third column shows the fixed graph and our spatial transformer results, and the last column is the ground truth. Points belonging to different semantic regions are colored differently. We observe better and more consistent segmentation result with our spatial transformer, specifically for the areas circled in red.

As for part segmentation, semantic segmentation performance improves when point clouds are given more freedom to deform (from affine to deformable spatial transformers) based on transformation of original locations and feature projections. Projective transformers give least performance gain, suggesting that mapping 3D points via homogeneous coordinates may not be most efficient.

Fig.4.9 shows that semantic segmentation results are smoother and more robust to missing points and occlusions with our deformable transformers.

Table 4.3: Spatial transformers improve semantic segmentation performance. We report mIoU(%) on S3DIS semantic segmentation dataset. Adding spatial transformers to [317] and [278] improves the performance.

	PointNet[238]	DGCNN[317]	[317](FIXED)	[317]+AFF	[317]+DEF	SplatNet [278]	[278]+DEF
	47.7	56.1	56.0	56.9	**57.2**	54.1	55.5
	ceiling	floor	wall	beam	column	window	clutter
[317](FIXED)	92.5	93.1	76.1	51.0	**41.7**	**49.6**	46.8
[317]+AFF	92.7	**93.6**	76.7	52.6	41.2	48.7	47.8
[317]+PROJ	92.5	93.5	76.7	52.7	40.7	48.5	**48.0**
[317]+DEF	**92.8**	**93.6**	**76.8**	**52.9**	41.1	49.0	**48.0**
	door	table	chair	sofa	bookcase	board	
[317](FIXED)	63.4	61.8	43.1	23.3	**42.0**	43.5	
[317]+AFF	**63.7**	63.4	45.1	27.0	41.3	44.8	
[317]+PROJ	**63.5**	62.3	44.8	27.0	41.5	44.9	
[317]+DEF	63.5	**64.2**	**45.2**	**28.1**	41.7	**46.1**	

Table 4.4: Spatial transformers improve object detection performance. We report car detection AP(%) on KITTI validation set. Adding spatial transformers leads to 2% performance gain.

	birds' eye			3D		
	Easy	Medium	Hard	Easy	Medium	Hard
VoxelNet[357]	77.3	59.6	51.6	43.8	32.6	27.9
VoxelNet + fixed graph	84.3	67.2	59.0	45.7	34.5	32.4
VoxelNet + deformable	**85.3**	**69.1**	**60.9**	**46.1**	**35.9**	**34.0**

4.4.4 3D Object Detection

We benchmark on KITTI 3D object detection [91]. It contains 7,481 training images / point clouds and 7,518 test images / point clouds, covering three categories: Car, Pedestrian, and Cyclist. For each class, detection outcomes are evaluated based on three difficulty levels: easy, moderate, and hard, according to the object size, occlusion state and truncation level. We follow the evaluation protocol in VoxelNet [357] and report car detection results on the validation set.

Network Architectures.. Shown in Fig.4.5, the network first partitions raw 3D points into voxels. We add deformable spatial transformers; points in each voxel are represented with point features. There are two deformable feature learning layers, each layer having 2 sub-graphs with 16-dimensional outputs. Note that the voxel partition is based on the input point coordinates. As in VoxelNet, the point features in each voxel are fed to 2 voxel feature encoding layers with channel 32 and 128 to get sparse 4D tensors representing the space. The middle convolutional layers process 4D tensors to further aggregate spatial contexts. Finally a Region Proposal Network (RPN) generates the 3D detection.

Table 4.5: Performance of different number of deformable transformation modules. Metric is average mIOU (%).

	fixed graph	1 graph	2 graphs	4 graphs
$f_i^{(t)} = 32$	84.2	84.9	85.2	85.3
$f_i^{(t)} k_i^{(t)} = 64$	84.2	85.3	85.2	83.5

In the first row, the output feature of each sub-graph is of dim. 32, while the number of subgraphs changes; the second row limits the multiplication of number of sub-graphs and sub-feature dim. to be 64.

We report the performance of 3 networks: **(1)** VoxelNet baseline [357]; **(2)** the fixed graph baseline, where we used the original point cloud location to learn the point feature at the place of spatial transformer blocks; **(3)** deformable spatial transformer networks as discussed above.

Results and Analyses. Table 4.4 reports car detection results on KITTI validation set.[1] Compared with baseline, having a point feature learning module improves the performance by 7.3% and 2.8% for birds' eye view and 3D detection performance on average, respectively. The deformable module further improves 8.9% and 3.9% respectively over VoxelNet.

4.4.5 Ablation Studies

We conduct ablation studies to understand how many spatial transformers may be sufficient to achieve satisfactory performance. We also study transformations of point coordinates and features of deformable spatial transformers. The influences of updating transformation matrices and transformers at different layers are investigated.

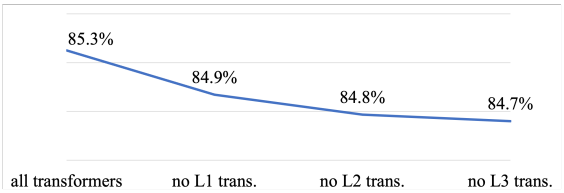


Figure 4.10: Part segmentation performance (average mIOU) of deformable transformers at different layers. When applying all transformers at three layers, the performance is highest. Removing transformers at different layer lead to performance drop. Removing transformers at layer 3 gives the most performance drop.

The Number of Transformers. Table 4.5 shows that for the fixed sub-feature dimension, the more graphs in each layer, the higher the performance. With the fixed complexity, (i.e., the product of the number of sub-graphs and the sub-feature dimension fixed at 64), the best performance is achieved at $k = 1, f = 64$ and $k = 2, f = 32$.

Two Components in Deformable transformers. A deformable spatial transformer has two components (Equation 4.7): affine transformation on point coordinates, AP, and three-

[1]The authors did not provide code. We use the implementation by [129] and obtain lower performance than the original paper.

dimensional projection of high-dimensional feature, CF. Fig.4.8 shows that both affine and feature only spatial transformers also improve performance, but the combination of both leads to the largest gain.

Updating Transformation Matrices. The transformation matrices are updated in an end-to-end fashion with the ultimate goal of increasing the task performance. It is of interest to understand if updating transformation matrix boosts the performance. Specifically, we randomly initialize transformation matrices of deformable spatial transformers and keep them not updated during training. The performance is 0.5% better than fixed graphs, indicating that adding more transformation graphs at different layers helps; however, it is 0.6% worse than updating transformation matrices, indicating learning to update transformation matrices in an end-to-end fashion is helpful.

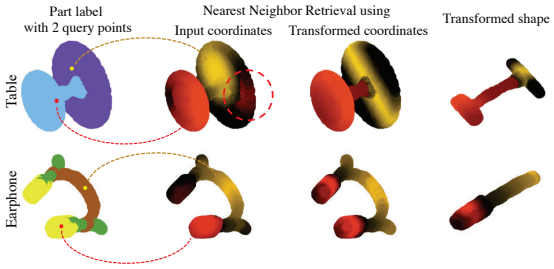


Figure 4.11: Local neighborhoods of two query points (red and yellow) using (transformed) 3D coordinates with nearest neighbor retrieval. Neighborhoods of transformed point clouds makes semantic information extraction more efficient: the neighborhood inside the dashed circle adapts to table base part. View rotating version for better visualization.

Transformers at Different Layers. We start with all deformable transformers effective at three layers, and remove transformers one layer a time. In Fig.4.10, the part segmentation performance is best with all transformers, whereas removing transformers at layer 3 gives the largest performance drop, suggesting that transformers at every layer help and those at the last layer are most important.

4.4.6 Time and Space Complexity

With spatial transformers, the model size changes little and the inference takes slightly more time (Table 4.6). Note that for fair comparisons, we increase the number of channels in the fixed graph baseline model for all the experiments. Even without increasing the number of

Table 4.6: Model size and test time on ShapeNet part segmentation. Spatial transformers slightly increase the inference time.

	Sampling-based		Point-based	
	[278]	[278] + transformer	[317] (fixed)	[317] + transformer
# Params.	2,738K	2,738K	2,174K	2,174K
Inference time (s/shape)	0.352	0.379	0.291	0.315

parameters of baselines (not shown in Table 4.6), adding spatial transformers only increases the number of parameters by 0.1%, as the number of parameters of spatial transformers (only transformation matrices) is very small.

4.4.7 Visualization and Analysis

We visualize the change in local neighborhoods when applying spatial transformers. We also visualize the transformed 3D points globally and locally.

Dynamic Neighborhood Visualization. To illustrate how our spatial transformers learn diverse neighborhoods for 3D shapes, we show the nearest neighbors of two query points and use corresponding colors to indicate corresponding neighborhoods. **(1)** Fig.4.11 shows that neighborhoods retrieved from deformed shapes encode additional semantic information, compared to neighborhoods from 3D coordinates. **(2)** Fig.4.2 shows that for table and earphone, different graphs enable the network to learn from diverse neighborhoods without incurring additional computational cost.

Global Visualization of Deformable Transformations. Fig.4.12 depicts some examples of learned deformable transformations in ShapeNet part segmentation. Each graph at a certain layer aligns the input 3D shape with similar semantic geometric transformations. For example, regardless of the shape of the rocket, graph 2 at layer 2 always captures the rocket wing information.

Local Distributions after Deformable Transformations. 3D Points often do not have balanced sampling, which makes point convolution challenging, as the k-NN graph does not accurately represents the exact neighborhood and 3D structure information. Our deformable spatial transformer gives every point flexibility and finds better neighborhoods.

We wonder if transformers make the point cloud closer to balanced sampling. We normalize the point coordinates for fair comparisons. Fig.4.13 visualizes the local distribution around a sample point on skateboard: After deformable transformation, the points are moved to a more uniform distribution. We analyze the standard deviation of raw and transformed point cloud coordinates in ShapeNet data. The standard deviation of point coordinates decreases 50.2% over all categories after spatial transformations, indicating a more balanced distribution of transformed points.

We check if the point coordinates are statistically different before and after the application of transformers. We perform t-test on the original and transformed point clouds. The t-score is 7.15 over all categories with p-value smaller than 1e-9. The transformed point cloud distribution is thus statistically different from the input point cloud distribution.

4.5 Summary

We propose novel spatial transformers for 3D point clouds that can be easily added onto to existing point cloud processing networks. They can dynamically alter local point neighborhoods for better feature learning.

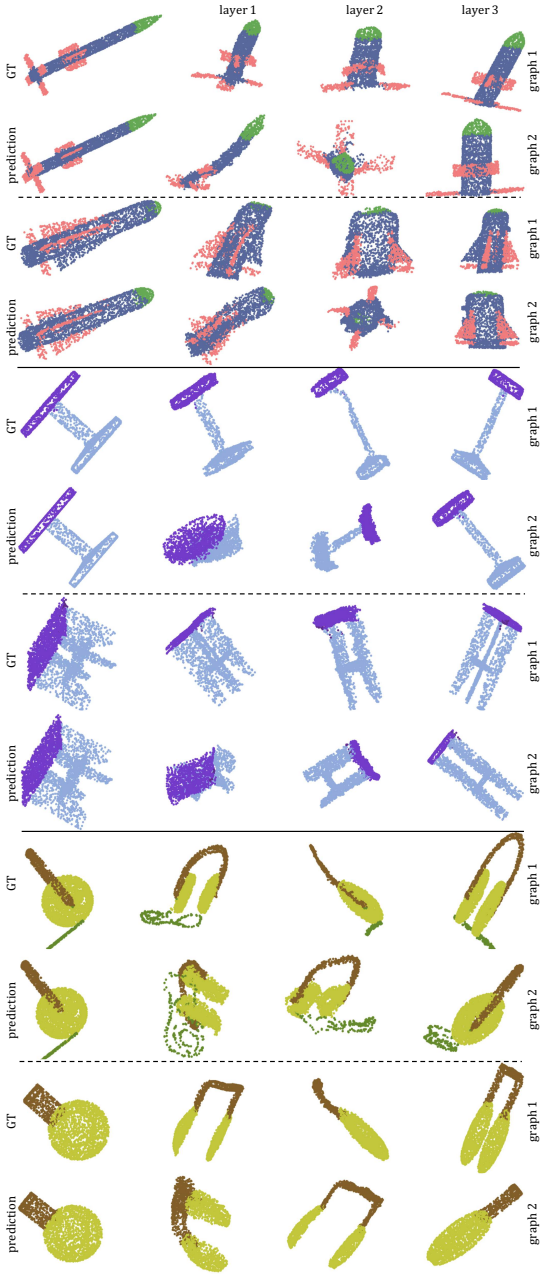


Figure 4.12: Examples of learned deformable transformations in ShapeNet part segmentation. 3D shapes include rocket, table and earphone (from up to bottom). Every two rows depict an instance with learned transformations. We observe that each transformation at certain layer aligns input 3D shape with similar semantic geometric transformation, e.g., graph 2 at layer 2 in rocket examples captures rocket wings. Graph 2 at layer 1 in table examples captures table surfaces.

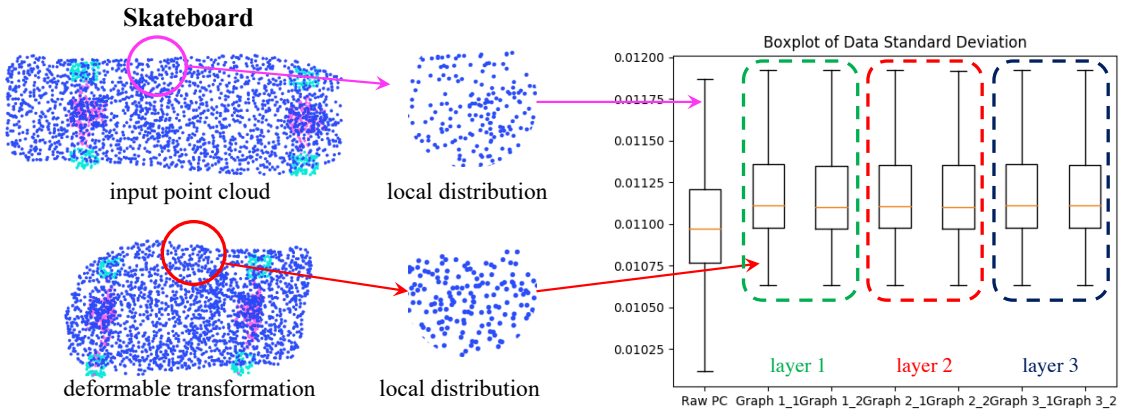


Figure 4.13: Spatial transformers improve the point cloud processing efficiency by improving local distributions of points. We show local distributions of a point cloud without and with transformers. The standard deviation of the transformed point cloud is smaller, enhancing the local neighborhood grouping (e.g. when using k-NN for affinity matrices, more balanced point distributions make feature learning in each neighborhood suffer less variations and outliers) and feature learning efficiency.

We study one linear (affine) transformer and two non-linear (projective and deformable) transformers. We benchmark them on point-based [317, 176] and sampling-based [278] point cloud networks and on three large-scale 3D point cloud processing tasks (part segmentation, semantic segmentation and object detection). Our spatial transformers outperform the fix graph counterpart for state-of-the-art methods.

There are some limitations of our spatial transformers. First, there are not many constraints on deformable spatial transformers to capture the geometry of the 3D point clouds. More complex non-linear spatial transformers may further improve the performance. On the other hand, spatial transformers learn global transformations of 3D point clouds for altering local neighborhoods. It is unclear if combining both global and local transformations [61, 287] would further improve the learning capacity and task performance.

Chapter 5

3D Shape Reconstruction from Free-Hand Sketches

5.1 Introduction

Human free-hand sketches are the most abstract 2D representations for 3D visual perception. Although a sketch may consist of only a few colorless strokes and exhibit various deformation and abstractions, humans can effortlessly envision the corresponding real-world 3D object from it. It is of interest to develop a computer vision model that can replicate this ability. Although sketches and 3D representations have drawn great interest from researchers in recent years, these two modalities have been studied relatively independently. We explore the plausibility of bridging the gap between sketches and 3D, and build a computer vision model to recover 3D shapes from sketches. Such a model will unleash many applications, like interactive CAD design and VR/AR games.

With the development of new devices and sensors, sketches and 3D shapes, as representations of real-world objects beyond natural images, become increasingly important. The popularity of touch-screen devices makes sketching not a privilege to professionals anymore and increasingly popular. Researchers have applied sketch in tasks like image retrieval [90, 343, 256, 169, 309, 333] and image synthesis [90, 348, 230, 311, 181, 92] to leverage its power in expression. Furthermore, as depth sensors, such as structured light device, LiDAR, and TOF cameras, become more ubiquitous, 3D data become an emerging modality in computer vision. 3D reconstruction, the process of capturing the shape and appearance of real objects, is an essential topic in 3D computer vision. 3D reconstruction from multi-view images has been studied for many years [85, 55, 7, 222]. Recent works [248, 68, 79] have further explored 3D reconstruction from a single image.

Despite these trends and progress, there are limited works connecting 3D and sketches. We argue that sketches are abstract 2D representations of 3D perception, and it is of great significance to study sketches in a 3D-aware perspective and build connections between two modalities. Researchers have explored the potential of *distortion-free* line drawings (e.g.

edge maps) for 3D modeling [170, 332, 171]. These works are based on *distortion-free* line drawings and generalize poorly to free-hand sketches (Fig.5.1**L**). Furthermore, the role of line drawings in such works is to provide geometrical information for the subsequent 3D modeling. Some other works [66, 189] employ neural networks to reconstruct 3D shapes directly from line drawings. However, their decent reconstructions come with two major limitations: **a)** they use distortion-free line drawings as training data, which makes such models hard to generalize to free-hand sketches; **b)** they usually require inputs depicting the object from multi-views to achieve satisfactory outcomes. Therefore, such methods cannot reconstruct the 3D shape from a single-view free-hand sketch well, as we show later in the experiment section. Other works such as [300, 116] tackle 3D retrieval instead of 3D shape reconstruction from sketches. Retrieved shapes come from the pre-collected gallery set and may not resembles novel sketches well. Overall, reconstructing a 3D shape from a single *free-hand* sketch is still left not well explored.

We explore single-view free-hand sketch-based 3D reconstruction (Fig.5.1**C**). A free-hand sketch is defined as a line drawing created without any additional tool. As an abstract and concise representation, it is different from distortion-free line drawings (e.g. edge maps) since it commonly has some spatial distortions, but it can still reflect the essential geometric shape. 3D reconstruction from sketch is challenging due to the following reasons: **a)** Data insufficiency. Paired sketch-3D datasets are rare although there exist several large-scale sketch datasets and 3D shape datasets, respectively. Furthermore, collecting sketch-3D pairs can be very time-consuming and expensive than collecting sketch-image pairs, as each 3D shape could be sketched from various viewing angles. **b)** Misalignment between two representations. A sketch depicts an object from a certain view while a 3D shape can be viewed from multiple angles due to the encoded depth information. **c)** Due to the nature of hand drawing, a sketch is usually geometrically imprecise with a individual style compared to the real object. Thus a sketch can only provide suggestive shape and structural information. In contrast, a 3D shape is faithful to its corresponding real-world object with no geometric deformation.

To address these challenges, we propose a single-view sketch-to-3D shape reconstruction framework. Specifically, it takes a sketch from an *arbitrary* angle as input and reconstructs a 3D point cloud. Our model cascades a sketch standardization module U and a reconstruction module G. U handles various sketching styles/distortions and transfers inputs to standardized sketches while G takes a standardized sketch to reconstruct the 3D shape (point cloud) *regardless of the object category*. The key novelty lies in the mechanisms we propose to tackle the data insufficiency issue. Specifically, we first train an photo-to-sketch model on unpaired large-scale datasets. Based on the model, sketch-3D pairs can be automatically generated from 2D renderings of 3D shapes. Together with the standardization module U which unifies input sketch styles, the synthesized sketches provide sufficient information to train the reconstruction model G. We conduct extensive experiments on a composed sketch-3D dataset, spanning 13 classes, where sketches are synthesized and 3D objects come from the ShapeNet dataset [46]. Furthermore, we collect an evaluation set, which consists of 390 real sketch-3D pairs. Results demonstrate that our model can reconstruct 3D shapes with certain geomet-

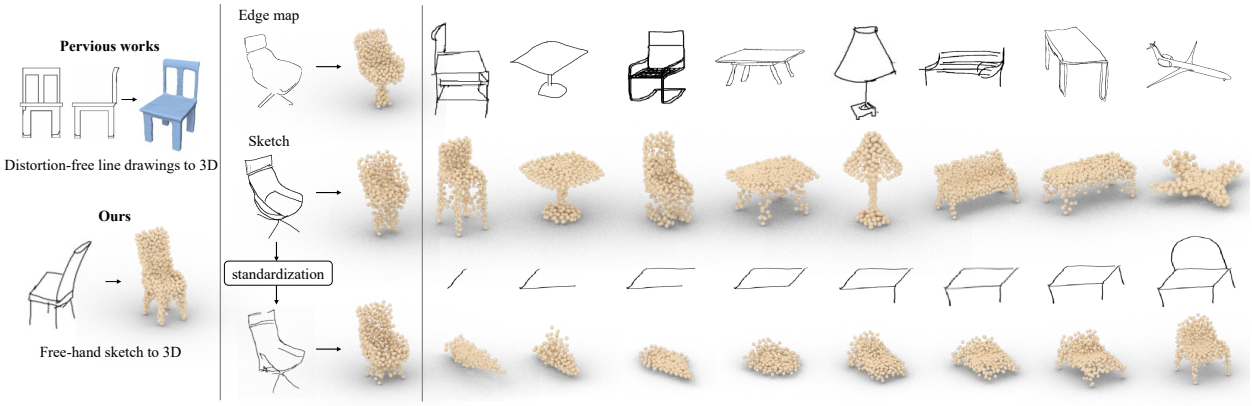


Figure 5.1: **Left**: We study 3D reconstruction from a single-view free-hand sketch, differing from previous works [66, 300, 189] which use multi-view distortion-free line-drawings as training data. **Center**: While previous works [66, 300] employ distortion-free line drawings (e.g. edge-maps) as proxies for sketches, our model trained on synthesized sketches can generalize better to free-hand sketches. Further, the proposed sketch standardization module makes the method generalizes well to free-hand sketches by standardizing different sketching styles and distortion levels. **Right**: Our model unleashes many practical applications such as real-time 3D modeling with sketches. A demo is here.

ric details from real sketches under different styles, stroke line-widths, and object categories. Our model also enables practical applications such as real-time 3D modeling with sketches (Fig.5.1**R**).

To summarize our contributions: **a)** We are one of the pioneers to study the plausibility of reconstructing 3D shapes from single-view free-hand sketches. **b)** We propose a novel framework for this task and explore various design choices. **c)** To handle data insufficiency, we propose to train on synthetic sketches. Moreover, sketch standardization is introduced to make the model generalize to free-hand sketches better. It is a general method for zero-shot domain translation, and we show applications on zero-shot image translation tasks.

5.2 Background

3D Reconstruction from Images. While SfM [222] and SLAM [85] achieve success in handling multi-view 3D reconstructions in various real-world scenarios, their reconstructions can be limited by insufficient input viewpoints and 3D scanning data. Deep-learning-based methods have been proposed to further improve reconstructions by completing 3D shapes with occluded or hollowed-out areas [336, 55, 144]. In general, recovering the 3D shape from a single-view image is an ill-posed problem. Attempts to tackle the problem include 3D shape reconstructions from silhouettes [68], shading [248], and texture [323]. However, these methods need strong presumptions and expertise in natural images [350], limiting their

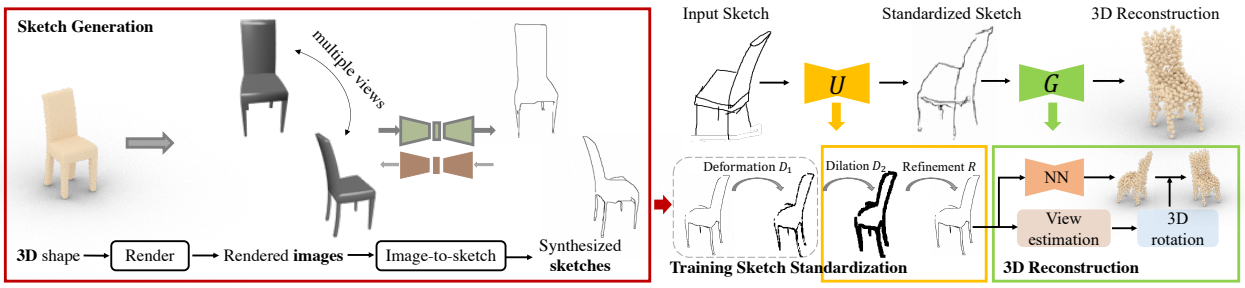


Figure 5.2: Model overview. The model consists of three major components: sketch generation, sketch standardization, and 3D reconstruction. To generate synthesized sketches, we first render 2D images for a 3D shape from multiple viewpoints. We then employ an image-to-sketch translation model to generate sketches of corresponding views. The standardization module standardize sketches with different styles and distortions. *Deformation* D_2 is only used in training for augmentation such that the model is robust to geometric distortions of sketches. For inference, sketches are dilated (D_1) and refined (R) so their style matches that of training sketches. For 3D reconstruction, a view estimation module is adopted to align the output's view and the ground-truth 3D shape.

usage in real-world scenarios. Generative adversarial networks (GANs) [96] and variational autoencoders (VAEs) [153] have achieved success in image synthesis and enabled [326] 3D shape reconstruction from a single-view image. Fan *et al.* [79] further adopt point clouds as 3D representation, enabling models to reconstruct certain geometric details from an image. They may not directly work on sketches as many visual cues are missing.

3D reconstruction networks are designed differently depending on the output 3D representation. 3D voxel reconstruction networks [283, 106, 330] benefit from many image processing networks as convolutions are appropriate for voxels. They are usually constrained to low resolution due to the computational overhead. Mesh reconstruction networks [312, 157] are able to directly learn from meshes, where they suffer from topology issues and heavy computation [224]. We adopt point cloud representation as it can capture certain 3D geometric details with low computational overhead. Reconstructing 3D point clouds from images has been shown to benefit from well-designed network architectures [79, 192], latent embedding matching [193], additional image supervision [210], etc.

Sketch-Based 3D Retrievals/Reconstructions. Free-hand sketches are used for 3D shape retrieval [300, 116] given their power in expression. However, retrieval methods are significantly constrained by the gallery dataset. Precise sketching is also studied in the computer graphics community for 3D shape modeling or procedural modeling [170, 127, 171]. These works are designed for professionals and require additional information for shape modeling, e.g., surface-normal, procedural model parameters. Delanoy *et al.* [66] first employ neural networks to learn 3D voxels from line-drawings. While it achieves impressive performance, this model has several limitations: **a)** The model uses distortion-free edge map as training data. While working on some sketches with small distortions, it cannot

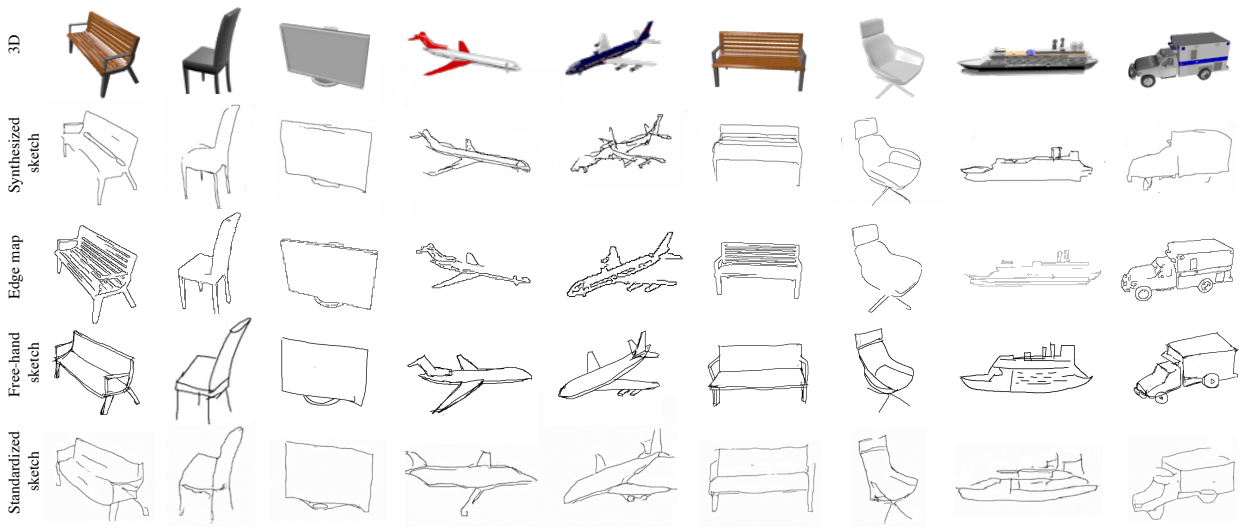


Figure 5.3: Synthesized sketches are visually more similar to free-hand sketches than edge maps as they contain distortions and emphasize perceptually significant contours. After standardization, the free-hand sketches share a uniform style similar to training data.

generalize to general free-hand sketches. **b)** The model requires multiple inputs from different viewpoints for a satisfactory result. These limitations prevent the model from generalizing to real free-hand sketches. Recent works also explore reconstructing 3D models from sketches with direct shape optimization [104], shape contours [98], differential renderer [349], and unsupervised learning [308]. Compared to existing works, the proposed method in this work reconstructs the 3D point cloud based on a single-view free-hand sketch. Our model may make 3D reconstruction and its applications more accessible to the general public.

5.3 3D Reconstruction from Sketches

The proposed framework has three modules (Fig.5.2). To deal with data insufficiency, we first synthesize sketches as the training set. The module U transfers an input sketch to a standardized sketch. Then, the module G takes the standardized sketch to reconstruct a 3D shape (point clouds). We also present details of a new sketch-3D dataset, which is collected for evaluating the proposed model.

5.3.1 Synthetic Sketch Generation

To the best of our knowledge, there exists no paired sketch-3D dataset. While it is possible to resort to edge maps [66], edge maps are different from sketches (as shown in the 3rd and 4th rows of Fig.5.3). We show that the reconstruction model trained on edge maps cannot generalize well to real free-hand sketches in Sec. 5.4.4. Thus it is crucial to find an

efficient and reliable way to synthesize sketches for 3D shapes. Inspired by [181], we employ a generative model to synthesize sketches from rendered images of 3D shapes. Fig.5.2**L** depicts the procedure. Specifically, we first render m images for each 3D shape, where each image corresponds to a particular view of a 3D shape. We then adopt the model introduced in [181] to synthesize gray-scale sketches images, denoted as $\{S_i | S_i \in \mathbb{R}^{W \times H}\}$, as our training data. W, H refer to the width and height of a sketch image.

5.3.2 Sketch Standardization

Sketches usually have strong individual styles and geometric distortions. Due to the gap between the free-hand sketches and the synthesized sketches, directly using the synthesized sketches as training data would not lead to a robust model. The main issues are that the synthesized sketches have a uniform style and they do not contain enough geometric distortions. Rather, the synthesized sketches can be treated as an intermediate representation if we can find a way to project a free-hand sketch to the synthesized sketch domain. We propose a zero-shot domain translation technique, the sketch standardization module, to achieve this domain adaption goal without using the free-hand sketches as the training data. The training of the sketch standardization module only involves synthesized sketches. The general idea is to project a distorted synthesized sketch to the original synthesized sketch. The training consists of two parts: **a)** since the free-hand sketches usually have geometric distortions, we apply predefined distortion augmentation to the input synthesized sketches first. **b)** A geometrically distorted synthesized sketch still has a different style and line style compared to the free-hand sketches. Thus, the first stage of the standardization is to apply a dilation operation. The dilation operation would project distorted synthesized sketches and the free-hand sketches to the same domain. Then, a refinement network follows to project the dilated sketch back to the synthesized sketch domain.

In summary, as in Fig.5.2, the standardization module U first applies a dilation operator D_2 to the input sketch, which is followed by a refinement operator R to transfer to the standardized synthesized-sketch style (or training-sketch style) $\widetilde{S}_i$, i.e. $U = R \circ D_2$. R is implemented as an image translation network. During training, a synthesized sketch $\widetilde{S}_i$ is first augmented by the deformation operator D_1 to mimic the drawing distortion, and then U aims to project it back to $\widetilde{S}_i$. Please note that D_1 would not be used during the testing. We illustrate the standardization process in Fig.5.2**R** with more details in the following.

Deformation. When training U, each synthesized sketch is deformed with moving least squares [258] for random, local and rigid distortion. Specifically, we randomly sample a set of control points on sketch strokes and denote them as p, and denote the corresponding deformed point set as q. Following moving least squares, we solve for the best affine transformation $l_v(x)$ such that: $\min \sum_i w_i |l_v(p_i) - q_i|^2$, where p_i and q_i are row vectors and weights $w_i = \frac{1}{|p_i - v|^{2\alpha}}$. Affine transformation can be written as $l_v(p_i) = p_i M + T$. We add constraint $M^T M = I$ to make the deformation is rigid to avoid too much distortion. Details can be found in [258].

Style Translation. Adapting to unknown input free-hand sketch style during inference can be considered as a zero-shot domain translation problem, which is challenging. Inspired by [338], we first dilate the augmented training sketch strokes with 4 pixels and then use image-to-image translation network Pix2Pix [137] to translate the dilated sketches to the undistorted synthesized sketches. During inference, we also dilate the free-hand sketches and apply the trained Pix2Pix model such that the style of an input free-hand sketch could be adapted to the synthesized sketch style during training. The dilation step can be considered as introducing uncertainty for the style adaption. Further, we show in Section 5.4.5 that the proposed style standardization module could be used as a general zero-shot domain translation technique, which generalizes to more applications such as sketch classification and zero-shot image-to-image translation.

A More General Message: Zero-Shot Domain Translation. We illustrate in Fig.5.4 a more general message of the standardization module: it can be considered as a general method for zero-shot domain translation. Consider the following problem: we would like to build a model to transfer domain X to domain Z but we do not have any training data from domain X. We propose a general idea to solve this problem is to build an intermediate domain Y as a bridge such that: **a)** we can translate data from domain X to domain Y and **2)** we can further translate data from domain Y to domain Z. We give two examples in the caption of Fig.5.4 and provided experimental results in Section 5.4.5.

5.3.3 Sketch-Based 3D Reconstruction

Our 3D reconstruction network G (pipeline in Fig.5.2**R**) consists of several components. Given a standardized sketch $\widetilde{S}_i$, the view estimation module first estimates its viewpoint. $\widetilde{S}_i$ is then fed to the sketch-to-3D module to generate a point cloud $P_{i,pre}$, whose pose aligns with the sketch viewpoint. A 3D rotation corresponding to the viewpoint is then applied to $P_{i,pre}$ to output the canonically-posed point cloud P_i. The objective of G is to minimize distances between reconstructed point cloud P_i and the ground-truth point cloud $P_{i,gt}$.

View Estimation Module. The view estimation module g_1 aims to determine the 3D pose from an input sketch $\widetilde{S}$. Similar to the input transformation module of the PointNet [240], g_1 estimates a 3D rotation matrix A from a sketch $\widetilde{S}$, i.e., $A = g_1(\widetilde{S})$. A regularization loss $L_{\text{orth}} = \|I - AA^T\|_F^2$ is applied to ensure A is a rotation (orthogonal) matrix. The rotation matrix A rotates a point cloud from the viewpoint pose to a canonical pose, which matches the ground truth.

3D Reconstruction Module. The reconstruction network g_2 learns to reconstruct a 3D point cloud P_{pre} from a sketch $\widetilde{S}$, i.e., $P_{pre} = g_2(\widetilde{S})$. P_{pre} is further transformed by the corresponding rotation matrix A to P so that P aligns with the ground-truth 3D point cloud P_{gt}'s canonical pose. Overall, we have $P = g_1(\widetilde{S}) \cdot g_2(\widetilde{S})$. To train G, we penalize the distance between an output point cloud P and the ground-truth point cloud P_{gt}. We employ

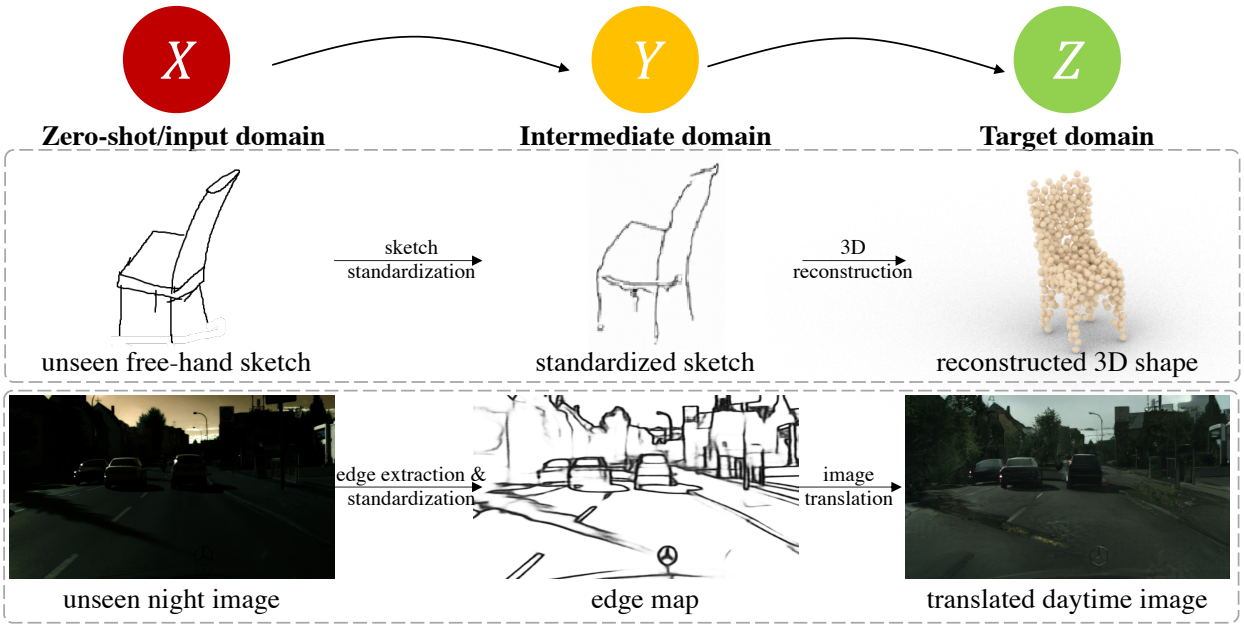


Figure 5.4: Sketch standardization can be considered as a general zero-shot domain translation method. Given a sample from a zero-shot (input) domain X, we first translate it to a universal intermediate domain Y, and finally to the target domain Z. **1st example** (2nd row): the input domain is an unseen free-hand sketch. With sketch standardization, it is translated to an intermediate domain: standardized sketch, which shares similar style as synthesized sketch for training. With 3D reconstruction, the standardized sketch can be translated to the target domain: 3D point clouds. **2nd example** (last row): the input domain is an unseen nighttime image. With edge extraction, it gets translated to an intermediate domain: edge map. With the image-to-image translation model, the standardized edge map can be translated to the target domain: daytime image.

the Chamfer distance (CD) between two point clouds $P, P_{gt} \subset \mathbb{R}^3$:

$$d_{CD}(P \| P_{gt}) = \sum_{\mathbf{p} \in P} \min_{\mathbf{q} \in P_{gt}} \|\mathbf{p} - \mathbf{q}\|_2^2 + \sum_{\mathbf{q} \in P_{gt}} \min_{\mathbf{p} \in P} \|\mathbf{p} - \mathbf{q}\|_2^2 \tag{5.1}$$

The final loss of the entire network is

$$L = \sum_i d_{CD}\left(G \circ U(S_i) \| P_{i,gt}\right) + \lambda L_{\text{orth}} \tag{5.2}$$

$$= \sum_i d_{CD}\left(A_i \cdot P_{i,pre} \| P_{i,gt}\right) + \lambda L_{\text{orth}} \tag{5.3}$$

$$= \sum_i d_{CD}\left(g1(\widetilde{S}_i) \cdot g_2(\widetilde{S}_i) \| P_{i,gt}\right) + \lambda L_{\text{orth}} \tag{5.4}$$

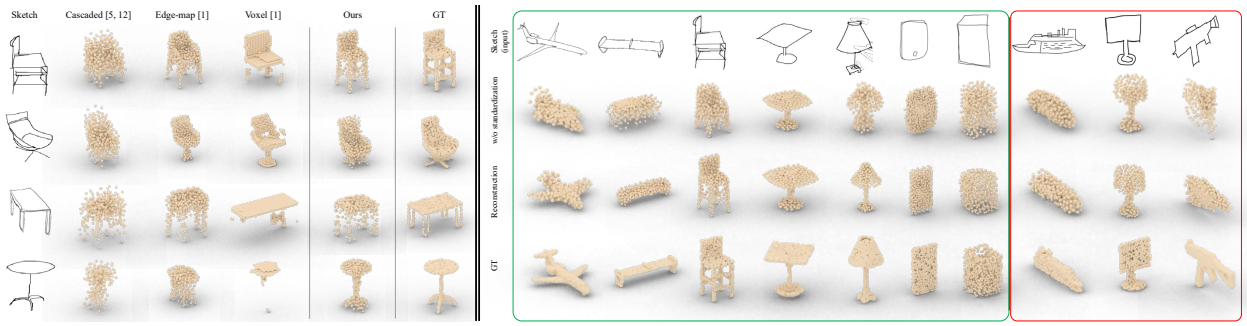


Figure 5.5: **Left**: Performance on free-hand sketches with different design choices. The design pool includes the model with a cascaded two-stage structure (2nd column), the model trained on edge maps (3rd column), the model whose 3D output is represented by voxel (4th column), and the proposed model (5th column). Overall, the proposed method achieves better performance and keeps more fine-grained details, e.g., the legs of chairs. **Right**: 3D reconstructions on our newly-collected free-hand sketch evaluation dataset. *Outlined in green:* Examples of some good reconstruction results. Our model reconstructs 3D shapes with fine geometric fidelity of multiple categories *unconditionally*. *Outlined in red:* Examples of failure cases. Our model may not handle detailed structures well (e.g., *watercraft*), recognize the wrong category (e.g., *display* as a *lamp*) due to the ambiguity of the sketch, as well as not able to generate 3D shape from very abstract sketches where few geometric information is available (e.g., rifle).

where λ is the weight of the orthogonal regularization loss and $\widetilde{S}_i = R \circ D_2 \circ D_1(S_i)$ is the standardized sketch from S_i. We employ CD rather than EMD (Section 5.4.2) to penalize the difference between the reconstruction and the ground-truth point clouds because CD emphasizes the geometric outline of point clouds and leads to reconstructions with better geometric details. EMD, however, emphasizes the point cloud distribution and may not preserve the geometric details well at locations with low point density.

5.4 Experimental Results

We first present the datasets, training and evaluation details, followed by qualitative and quantitative results. Then, we provide comparisons with some state-of-the-art methods. We also conduct ablation studies to understand each module.

5.4.1 3D Sketching Dataset

To evaluate the performance of our method, we collected a real-world evaluation set containing paired sketch-3D data. Specifically, we randomly choose ten 3D shapes from each of the 13 categories of the ShapeNet dataset [46]. Then we randomly render 3 images from different viewpoints for each 3D shape. Totally, there are 130 different 3D shapes and 390

rendered images. We recruited 11 volunteers to draw the sketches for the rendered images. Final sketches are reviewed for quality control. We present several examples in Fig.5.3.

5.4.2 Training Details and Evaluation Metrics

Training. The proposed model is trained on a subset of ShapeNet [46] dataset, following settings of [330]. The dataset consists of 43,783 3D shapes spanning 13 categories, including car, chair, table, etc. For each category, we randomly select 80% 3D shapes for training and the rest for evaluation. As mentioned in Section 5.3.1, corresponding sketches of rendered images from 24 viewpoints of each 3D shape of ShapeNet are synthesized with our synthetic sketch generation module.

Evaluation. To evaluate our method's 3D reconstruction performance on free-hand sketches, we use our proposed sketch-3D datasets (Section 5.4.1). To evaluate the generalizability of our model, we also evaluate on three additional free-hand sketch datasets, including the Sketchy dataset [256], the TU-Berlin dataset [77], and the QuickDraw dataset [97]. For these additional datasets, only sketches from categories that overlap with the ShapeNet dataset are considered.

Following the previous works [79, 193, 337], we adopt two evaluation metrics to measure the similarity between the reconstructed 3D point cloud P and the ground-truth point cloud P_{gt}. The first one is the Chamfer Distance (Eqn.5.1), and another one is the Earth Mover's Distance (EMD): $d_{EMD}(P, P_{gt}) = \min_{\phi:P\mapsto P_{gt}} \sum_{x\in P} \|x - \phi(x)\|$, where P, P_{gt} has the same size $|P| = |P_{gt}|$ and $\phi : P \mapsto P_{gt}$ is a bijection. CD and EMD evaluate the similarity between two point clouds from two different perspectives (more details can be found in [79]).

5.4.3 Implementation Details

Sketch Generation. We utilize an off-the-shelf sketch-image translation model [181] to synthesize sketches for training. Given the appropriate quality of the generated sketches on the ShapeNet dataset (with some samples depicted in Fig.5.3), we directly use the model without any fine-tuning.

Data Augmentation. During training, to improve the model's generalizability and robustness, we perform data augmentation for synthetic sketches before feeding them to the standardization module. Specifically, we apply image spatial translation (up to ± 10 pixels) and rotation (up to $\pm 10^\circ$) on each input sketch.

Sketch Standardization. Each input sketch S_i is first randomly deformed with moving least squares [258] both globally and locally (D_1), and then binarized and dilated five times iteratively (D_2) to obtain a rough sketch S_r. The rough sketch S_r is then used to train a Pix2Pix model [137], R, to reconstruct the input sketch S_i. The network is trained for 100 epochs with an initial learning rate of 2e-4. Adam optimizer [152] is used for the parameter optimization. During evaluation, random deformation D_1 is discarded.

3D Reconstruction. The 3D reconstruction network is based on [79]'s framework with hourglass network architecture [213]. We compare several different network architectures

Table 5.1: Ours outperforms baselines for 3D reconstruction. [66] uses edge-maps rather than sketches as input. [330] uses voxels rather than point clouds as output. [217] represents the 3D shapewith multi-view depth maps. "cas." refers to the two-stage cascaded training following [181, 79]. CD and EMD measure distances between reconstructions and ground-truths from different perspectives (see text for details). The lower, the better.

error	**C**hamfer **D**istance ($\times 10^{-4}$)							**E**arth **M**over's **D**istance ($\times 10^{-2}$)						
	points	edge [66]	voxel [330]	cas.	[217]	retrieval[300]	ours	points	edge [66]	voxel [330]	cas.	[217]	retrieval[300]	ours
airplane	11.4	7.8	35.1	71.7	8.0	11.2	**6.1**	8.5	7.3	10.8	12.7	8.5	11.9	**6.5**
bench	29.2	16.7	202.8	414.1	16.8	14.5	**13.0**	11.1	8.7	22.0	25.8	10.0	8.6	**7.8**
cabinet	61.7	50.4	59.1	354.5	51.5	45.3	**39.2**	17.6	17.8	17.0	29.6	18.4	17.2	**16.0**
car	20.8	13.3	173.2	114.2	14.1	14.2	**10.4**	**8.9**	20.0	25.2	20.0	21.6	21.2	18.0
chair	41.8	36.4	108.6	237.1	36.1	33.0	**26.9**	15.1	15.6	19.4	22.8	16.1	15.3	**13.0**
display	68.6	48.3	**33.1**	340.2	49.3	38.2	37.7	15.5	15.1	**13.1**	27.9	16.4	14.6	14.4
lamp	63.3	59.4	107.0	214.0	60.2	63.5	**46.3**	21.3	22.6	21.2	24.9	22.3	22.6	**20.4**
speaker	88.2	79.7	203.2	406.4	81.2	72.3	**62.1**	19.4	19.2	23.8	28.0	21.8	20.0	**17.9**
rifle	17.0	12.1	170.1	15.4	12.3	14.2	**10.1**	**11.2**	13.8	23.7	15.4	15.2	17.6	12.4
sofa	32.8	20.9	141.2	482.4	22.3	20.3	**16.3**	11.1	8.5	18.6	25.4	9.1	8.6	**7.7**
table	55.2	49.4	134.7	469.5	50.5	49.1	**40.7**	19.1	17.7	18.5	26.5	18.2	18.2	**17.3**
telephone	30.7	27.3	26.9	259.8	27.1	27.4	**21.3**	13.4	13.6	15.1	27.2	15.1	15.3	**12.3**
watercraft	32.9	26.0	129.1	53.8	26.0	27.3	**20.3**	12.5	11.1	23.1	17.8	12.2	12.7	**10.6**
avg.	42.6	34.4	117.2	264.1	35.0	33.1	**26.9**	14.2	14.7	19.3	23.4	15.8	15.7	**13.4**
free-hand sketch	87.1	89.0	162.5	334.2	91.8	89.2	**86.1**	18.6	16.4	22.9	26.1	17.0	16.8	**16.0**

(simple encoder-decoder architecture, two-prediction-branch architecture, etc.) and find that hourglass network architecture gives the best performance. This may be due to its ability to extract key points from images [213, 40]. We train the network for 260 epochs with an initial learning rate of 3e-5. The weight λ of the orthogonal loss is 1e-3. To enhance the performance on every category, all categories of 3D shapes are trained together. The class-aware mini-batch sampling [263] is adopted to ensure a balanced category-wise distribution for each mini-batch. We choose Adam optimizer [152] for the parameter optimization. 3D point clouds are visualized with the rendering tool from [205].

5.4.4 Results and Comparisons

We first present our model's 3D shape reconstruction performance, along with the comparisons with various baseline methods. Then we present the results on sketches from different viewpoints and of different categories, as well the results on other free-hand sketch datasets. Note that unless specifically mentioned, all evaluations are on the free-hand sketches rather than synthesized sketches.

Baseline Methods. Our 3D reconstruction network is a one-stage model where the input sketch is treated as an image, and point clouds represent the output 3D shape. As conducting the first work for single-view sketch-based 3D reconstruction, we explore different design options adopted by previous works on distortion-free line drawings and/or 3D reconstruction, including architectures, representation of sketches and 3D shapes. We compare with different variants to demonstrate the effectiveness of each choice of our model.

1) Model design: end-to-end vs. two-stage. Although the task of reconstructing 3D shapes from free-hand sketches is new, sketch-to-image synthesize and 3D shape reconstruction from images have been studied before [181, 330, 79]. Is a straight combination of the two models, instead of an end-to-end model, enough to perform well for the task? To compare these two architectures' performance, We implement a cascaded model by composing a sketch-to-image model [358] and an image-to-3D model [79] to reconstruct 3D shapes.
2) Sketch: point-based vs. image-based. Considering a sketch is relatively sparse in pixel space and consists of colorless strokes, we can employ 2D point clouds to represent a sketch. Specifically, 512 points are randomly sampled from strokes of each binarized sketch, and we use a point-to-point network architecture (adapted from PointNet [240]) to reconstruct 3D shapes from the 2D point clouds.
3) Sketch: Using edge maps as proxy. We compare with a previous work [66]. Our proposed model uses synthetic sketch for training. However, an alternative option is using edge maps as a proxy of the free-hand sketch. As edge maps can be generated automatically (we use the Canny edge detector in implementation), the comparison helps us understand if our proposed synthesizing method is necessary.
4) 3D shape: voxel vs. point cloud. We compare with a previous work [330]. In this variant, we follow their settings and represent a 3D shape with voxels. As the voxel representation is adopted from the previous method, the comparison helps to understand if representing 3D shapes with point clouds has benefits.
5) 3D shape: depth map vs. point cloud. In this variant, we exactly follow a previous work [217] and represent the 3D shape with multi-view depth maps.
Comparison and Results. Table 5.1 and Fig.5.5**L** present quantitative and qualitative results of our method and different design variants. Specifically for quantitative comparisons (Table 5.1), we report 3D shape reconstruction performance on both synthesized (evaluation set) and free-hand sketches. This is due to that the collected free-hand sketch dataset is relatively small and together they provide a more comprehensive evaluation. We have the following observations: **a)** Representing sketches as images outperforms representing them as 2D point clouds (points vs. ours). **b)** The model trained on synthesized sketches performs better on real free-hand sketches than the model trained on edge maps (89.0 vs. 86.1 on CD, 16.4 vs. 16.0 on EMD). Training with edge maps could reconstruct okay overall coarse shape. However, the unsatisfactory performance on geometric details reveals such methods are hard to generalize to free-hand sketches with distortions. It also shows the necessity of the proposed sketch generation and standardization modules. **c)** For model design, the end-to-end model outperforms the two-stage model by a large margin (cas. vs. ours). **d)** For 3D shape representation, while the voxel representation can reconstruct the general shape well, the fine-grained details are frequently missing due to its low resolution ($32 \times 32 \times 32$). Thus, point clouds outperform voxels. The proposed method also outperforms a previous work that uses depth maps as 3D shape representation [217] . Note that the resolution of voxels can hardly improve much due to the complexity and computational overhead. However, we show that increasing the number of points improves the reconstruction quality (details in supplementary).

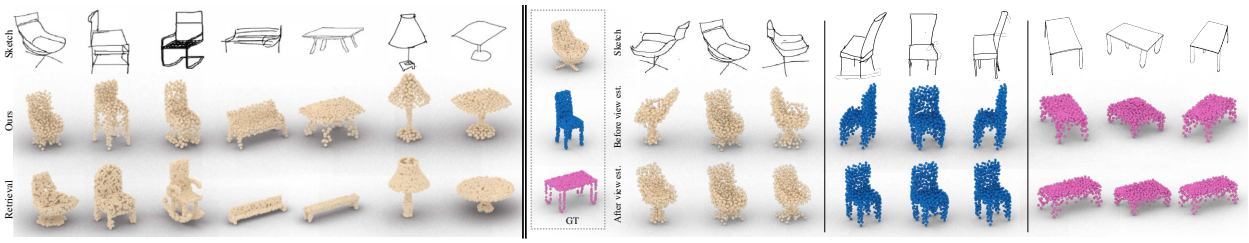


Figure 5.6: **Left**: Ours (2nd row) versus nearest-neighbor retrieval results (last row) of given sketches. Our model generalizes to unseen 3D shapes better and has higher geometry fidelity. **Right**: 3D reconstructions of sketches from different viewpoints. Before the view estimation module, the reconstructed 3D shape aligns with the input sketch's viewpoint. The module transforms the pose of the output 3D shape to align with canonical pose, i.e. the pose of the ground-truth 3D shape.

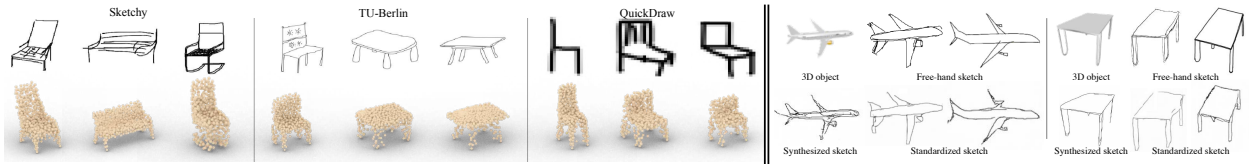


Figure 5.7: **Left**: Our approach trained on ShapeNet can be directly applied to other unseen sketch datasets [256, 77, 97] and generalize well. Our model is able to reconstruct 3D shapes from sketches with different styles and line-widths, and even low-resolution data. **Right**: Standardized sketches converted from different individual styles (by different volunteers). For each rendered image of a 3D object, we show free-hand sketches from two volunteers and the standardized sketches from these free-hand sketches. Contents are preserved after the standardization process, and standardized sketches share the style similar to the synthesized ones.

Retrieval Results. We compare with nearest-neighbor retrievals, following methods and settings of [300] (Fig.5.6**L**). We could generalize to unseen 3D shapes and reconstructs with higher geometry fidelity (e.g., stand of the lamp).

Reconstruction with Different Categories and Views. Fig.5.5**R** shows 3D reconstruction results with sketches from different object categories. Our model reconstructs 3D shapes of multiple categories *unconditionally*. There are some failure cases that the model may not handle well.

Fig.5.6**R** depicts reconstructions with sketches from different views. Our model can reconstruct 3D shapes from different views even if certain parts are occluded (e.g. legs of the table). Slight variations in details exist for different views.

Evaluation on Other Free-Hand Sketch Datasets. We also evaluate on three other free-hand sketch datasets [256, 77, 97]. Our model can reconstruct 3D shapes from sketches with different styles, line-widths, and levels of distortions even at low resolution (Fig.5.7**L**).

Table 5.2: Ablation studies of standardization and view estimation module. CD is enlarged by 10^4, and EMD by 10^2. **(a)** 3D shape reconstruction errors of ablation studies of standardization and view estimation module. Having both standardization and view estimation module gives the highest performance. The lower, the better. **(b)** Reconstruction error with different components of the standardization module: deformation and style translation. Having both parts gives the highest performance. **(c)** The sketch standardization module improves cross-dataset sketch classification accuracy. A ResNet-50 model is trained on TU-Berlin [77] and evaluated Sketchy dataset [256]. The sketch standardization module gives 3 percentage points gain.

(a)

error	no standard.	no view est.	ours
CD	92.6	86.8	**86.1**
EMD	18.2	16.2	**16.0**

(b)

deform.	trans.	CD	EMD
×	×	92.6	18.2
×	✓	87.2	16.3
✓	×	90.1	17.4
✓	✓	**86.1**	**16.0**

(c)

(%)	acc.
w/o std.	75.1
w/ std.	**78.1**

5.4.5 Sketch Standardization Module

Visualization. The standardization module can be considered as a domain translation module designed for sketches. We show the standardized sketches of these free-hand sketches and compare them to the synthesized ones Fig.5.7**R**. With the standardization module, sketches share a style similar to synthesized sketches which are used as training data. Thus, standardization diminishes the domain gap of sketches with various styles and enhances the generalizability.

Ablation Studies of the Entire Module. The sketch standardization module is introduced to handle various drawing styles of humans. We thus verify this module's effectiveness on real sketches, both quantitatively (Table 5.2**a**) and qualitatively (Fig. 5.5**R**). As shown in Table 5.2**a**, the reconstruction performance has a significant drop when removing the standardization module. Its effect is also proved in visualizations. In Fig. 5.5**R**, we can observe that our full model equipping with the standardization module can produce 3D shapes with higher quality, being more similar to GT shapes, e.g., the airplane and the lamp.

Ablation Studies of Different Components. The standardization module consists of two components: sketch deformation and style translation. We study each module's performance and report in Table 5.2**b**. We observe that the style transformation part improves the reconstruction performance better compared with the deformation part, while having both parts gives the highest performance.

Additional Applications. We show the effectiveness of the proposed sketch standardization with two more applications. The first applications is on cross-dataset sketch classification. We identify the common 98 categories of TU-Berlin sketch dataset [77] and Sketchy dataset [256]. Then we train on TU-Berlin and evaluate on Sketchy. As reported in Table **5.2c**, adding additional sketch standardization module, the classification accuracy improves 3 percentage points.

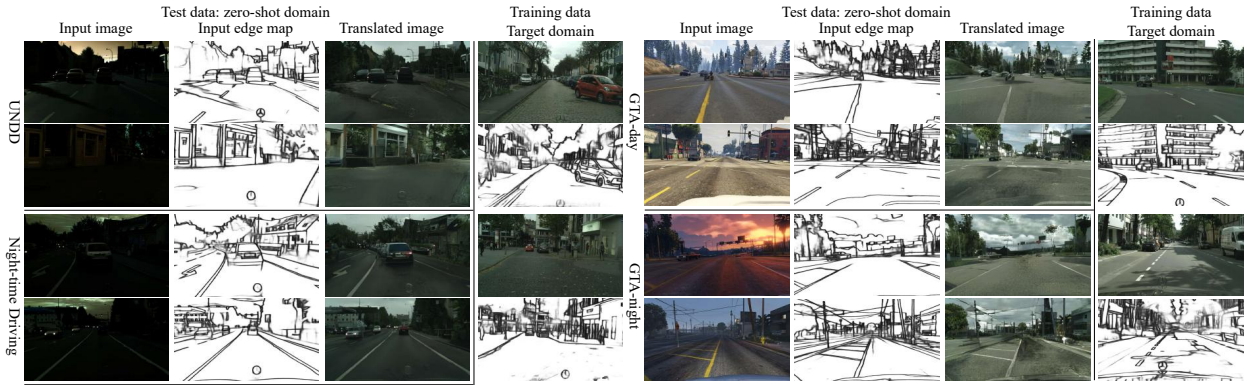


Figure 5.8: Zero-shot domain translation results. We aim to translate zero-shot images (**Left**) to the training data domain (**Right**). Specifically, we evaluate the proposed zero-shot domain translation performance on three new datasets: UNDD [209], Night-Time Driving [60] and GTA [249]. The novel domains of night-time and simulated images can be translated to the target domain of daytime and real-world images by leveraging the synthetic edge map domain as a bridge. The target domain is CityScapes dataset [56]. We extract corresponding edge maps and train an image-to-image translation model [137] to translate edge maps to the corresponding RGB images. 1st, 3rd, 5th, 7th rows of column 1 depict some sample training RGB images and 2nd, 4th, 6th, 8th rows of column 1 depict the corresponding edge maps respectively.

The second application corresponds to the second example depicted in Fig.5.4. The target domain is CityScapes dataset [56], where the training data comes from. We extract corresponding edge maps with a deep learning approach [232] and train an image-to-image translation model [137] to translate edge maps to the corresponding RGB images. We evaluate the zero-shot domain translation performance on three new datasets: UNDD [209] (night images), Night-Time Driving [60] (night images) and GTA [249] (synthetic images; screenshots taken from simulated environment). The novel domains of night and simulated images can be translated to the target domain of daytime and real-world images. We visualize the results in Fig.5.8.

5.4.6 View Estimation Module

Removing the view estimation module leads to a performance drop of CD and EMD (Table 5.2**a**). For qualitative results (Fig.5.6**R**), without the 3D rotation, the reconstructed 3D shape has the pose aligned with the input sketch. With the 3D rotation, the 3D shape is aligned to the ground truth's canonical pose.

5.5 Summary

We study 3D shape reconstruction from a single-view free-hand sketch. The major novelty is that we use synthesized sketches as training data and introduce a sketch standardization module, in order to tackle the data insufficiency and sketch style variation issues. Extensive experimental results shows that the proposed method is able to successfully reconstruct 3D shapes from single-view free-hand sketches *unconditioned* on viewpoints and categories. The work may unleash more potentials of the sketch in applications such as sketch-based 3D design/games, making them more accessible to the general public.

Chapter 6

3D Dental Casting from a Single Image

6.1 Introduction

In dentistry, dental casts (gypsum teeth models) play a significant role in clinics, including the diagnosis for various syndromes, giving support to plastic and orthognathic surgery, detecting normal and abnormal growth, and providing information for planning and evaluating medical procedures and treatments [6, 148]. Accurate and reliable 3D dental cast reconstruction helps clinicians plan effective treatments.

Several non-contact methods such as laser scanning, magnetic resonance imaging, Moiré stripes, and stereophotogrammetry have been employed for 3D dental cast reconstruction [82,

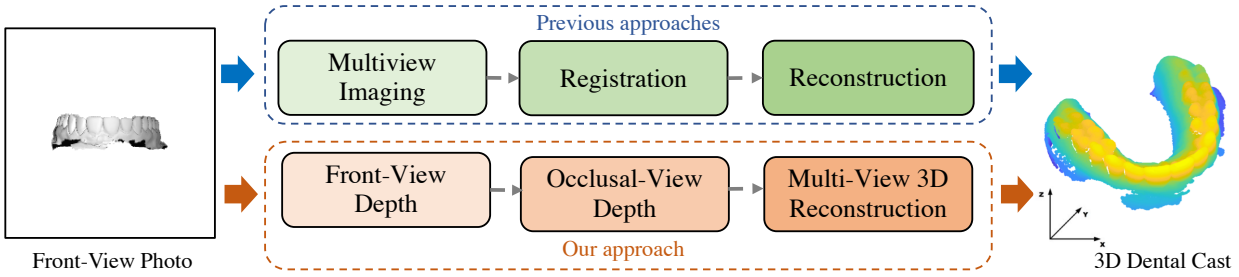


Figure 6.1: 3D dental cast reconstruction. To reconstruct a 3D dental cast, previous approaches require the full teeth model and perform multiview imaging, registration and finally multi-view 3D reconstruction, which are expensive and time-consuming (top). We study the plausibility of dental cast reconstruction from a single image. Specifically, from a front-view photo of a dental cast, we first predict the front-view depth map, and then synthesize the occlusal-view depth map, and finally project depth maps back to the 3D space and perform multi-view reconstruction to obtain the 3D dental cast (bottom). Our method relies on minimum input and is more efficient.

275, 123]. Though successful, these methods are complicated, time-consuming and requires a certain level of human labor. Additionally, these model-based methods overfit to a specific dental cast and has little generalizability, meaning the same procedure need to be done for every dental cast [123, 275] We seek a more efficient, inexpensive and generalizable approach for dental cast reconstruction.

Great progress has been made in 3D shape reconstruction from a single image in the computer vision community recently. Methods based on voxel reconstruction network [106, 330], point processing network [79, 240] and graph-based neural network [312, 157] are able to reconstruct satisfactory 3D shapes from a photo. These methods take a very small amount of time to reconstruct 3D shapes in the inference stage and has a very high generalizability for a certain category of objects. We resort to neural-network-based methods for 3D reconstruction.

We pioneer to study the plausibility of full 3D dental cast reconstruction from a single photo (Fig.6.1). The major challenge lies in how detailed structures of individual tooth can be reconstructed. Though state-of-the-art 3D reconstruction algorithms is able to reconstruct coarse 3D shapes from a single-view image, they hardly preserve fine details due to computational overhead and lack of sufficient information [330, 312, 79]. We aim to reconstruct dental casts with both coarse and fine geometrical fidelity. While it seems very challenging to reconstruct it from a single photo, it is not inaccessible as the oral cavity has certain structures, strong regularity and similarity. We learn proper priors to guide the dental cast reconstruction from the data.

State-of-the-art 3D reconstruction approaches are good at reconstructing global shapes but not fine-grained details, partially due to 3D data are usually represented in an irregular grid (meshes, point clouds) and strong generative models based on convolutions are not directly applicable. Convolutions are applicable to 3D representations in a regular grid (voxels, some signed distance functions), but the computational overhead to reach fine-grained details are usually very high. On the other hand, generative adversarial networks (GANs) achieves great success in synthesizing many realistic photos with fine-grained details [137, 316] but may not have satisfactory geometric shape [214, 208]. We propose a 3D reconstruction model that reconstructs shapes with good geometry and fine-grained details.

Specifically, for a photo of a dental cast from a viewpoint, we first predict a corresponding depth map, and then synthesize a novel view of the depth map with GANs. With all the synthesized depth maps, we finally project them back to the 3D space and obtain the full 3D dental cast.

To summarize, our work makes following contributions: **1)** We are the first to reconstruct the 3D dental cast with both coarse and fine geometrical fidelity from a single image. **2)** We improve state-of-the-art single-view 3D reconstruction approaches by adding fine geometric loss. Qualitative and quantitative results demonstrate that our model an reconstruct 3D shapes with fine-grained details without much computational overhead.

6.2 Background

Novel-View Synthesis. Object-centric view synthesis approaches aim to either synthesize novel views of the same instance [201, 268] or novel instances in the same category [282, 267, 356] by direct pixel prediction, pixel flow prediction, or 3D modeling by way of differentiable rendering. While not trained to generate specific views, generative adversarial networks that can successfully disentangle pose from identity [49, 214, 266] can be used to synthesize new views of objects in the same category. These methods show a high generalizability for objects within the category.

Multi-View 3D Reconstruction. Multi-view 3D reconstruction is also a problem that has long been studied with classical computer vision techniques [110, 190] like multi-view stereo and visual hull reconstruction. SfM [222] and SLAM [85] achieve success in handling multi-view 3D reconstructions in various real-world scenarios. Their reconstructions can be limited by insufficient input viewpoints and 3D scanning data. Deep learning based methods have been proposed to further improve reconstructions by completing 3D shapes with occluded or hollowed-out areas [336, 55, 144].

Methods that require full shape supervision in the multi-view [55, 144, 330] case are typically trained using ShapeNet [46]. There are other approaches that use more natural forms of multi-view supervision such as images, depth maps, and silhouettes [144, 292, 321, 334], with known cameras.

Single-View 3D Reconstruction. In general, recovering the 3D shape from a single-view image is an ill-posed problem. Attempts to tackle the problem include 3D shape reconstructions from silhouettes [68], shading [248], and texture [323]. However, these methods need strong presumptions and expertise in natural images [350], which limit their usage in the real-world scenarios. Generative adversarial networks (GANs) [96] and variational autoencoders (VAEs) [153] have achieved success in image synthesis and enable [326] 3D shape reconstruction from a single-view image. Fan *et al.* [79] further adopt point clouds as 3D shape representation, which enables models to reconstruct fine-grained details from a single-view image.

3D reconstruction networks are designed differently depending on the output 3D representation. 3D voxel reconstruction networks [283, 106, 330] benefit from many image processing networks as convolutions are appropriate for voxels. They are usually constrained to low resolution due to the computational overhead. Mesh reconstruction networks [312, 157] are able to directly learn from meshes, where they suffer from topology issues and heavy computation [224]. Point cloud representation capture fine-grained 3D geometric details with low computational overhead. Reconstructing 3D point clouds from images has been shown to benefit from well-designed network architectures [79, 192], latent embedding matching [193], additional image supervision [210], etc. However, the above-mentioned methods fail to reconstruct 3D shapes with fine-grained details.

As we aim to reconstruct 3D shapes with both coarse and fine geometrical fidelity, we focus on image/depth-map based 3D reconstruction approaches. Soltani et al. [17] proposed to synthesize multiple depth maps and silhouettes from given single-view or multi-view depth

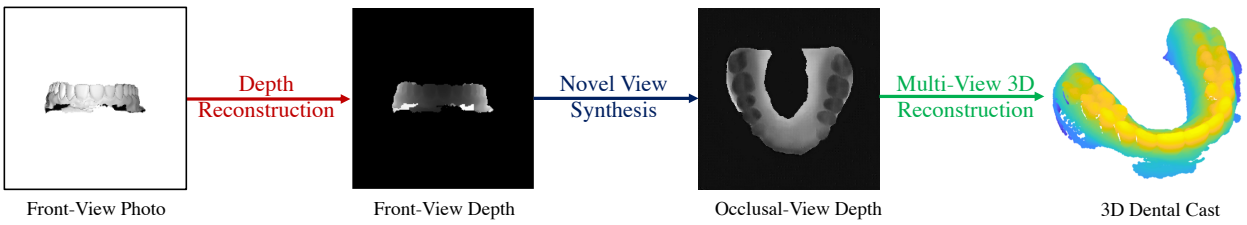


Figure 6.2: Pipeline of the proposed approach. For a photo of a dental cast from a viewpoint (we use front-view photo as an example), we first reconstruct a corresponding depth map (depth reconstruction). Then, we synthesize novel views of the depth map with GANs (novel view synthesis). Depth maps are finally projected back b to the 3D space and combined to obtain the full 3D dental cast (multi-view 3D reconstruction).

maps and silhouettes, and perform multi-view reconstruction for the final 3D shape. As showed in Section 6.4.6, this approach reconstruct 3D shapes with satisfactory coarse geometrical fidelity, but fail to reconstruct good local geometric structures and suffers from alignment issues when combining multi-view depth maps.

6.3 Approach

For a photo of a dental cast from a viewpoint, we first predict a corresponding depth map, and then synthesize a novel view of the depth map with GANs. With synthesized depth maps, we finally project them to back the 3D space to obtain the full 3D dental cast (Fig.6.2). We introduce details of each component of our approach as follows.

6.3.1 Data Collection and Prepossessing

We collect 2,427 3D dental casts for the study. The dataset originates from full jaw gypsum models. For simplicity without loss of generality, only upper jaws were used in this paper. The casts are digitized through a precise optical scanning process and serialized into triangular mesh representation [289].

A cusp-based algorithm was used to properly align all dental casts to a canonical pose. The algorithm fits a parabolic curve with detected cusps of the teeth to determine the occlusal and anterior directions, which are used for the preliminary data alignment.

6.3.2 Rendering and Depth Prediction from Photo

To build our training dataset, we first render a grayscale front-view photo and a depth map for each 3D model. We normalize the 3D mesh and use a single fixed point light source during rendering for simplicity and consistency. The grayscale photo and depth map pair is then used to train the depth reconstruction module. Specifically, we employ pix2pix [137]

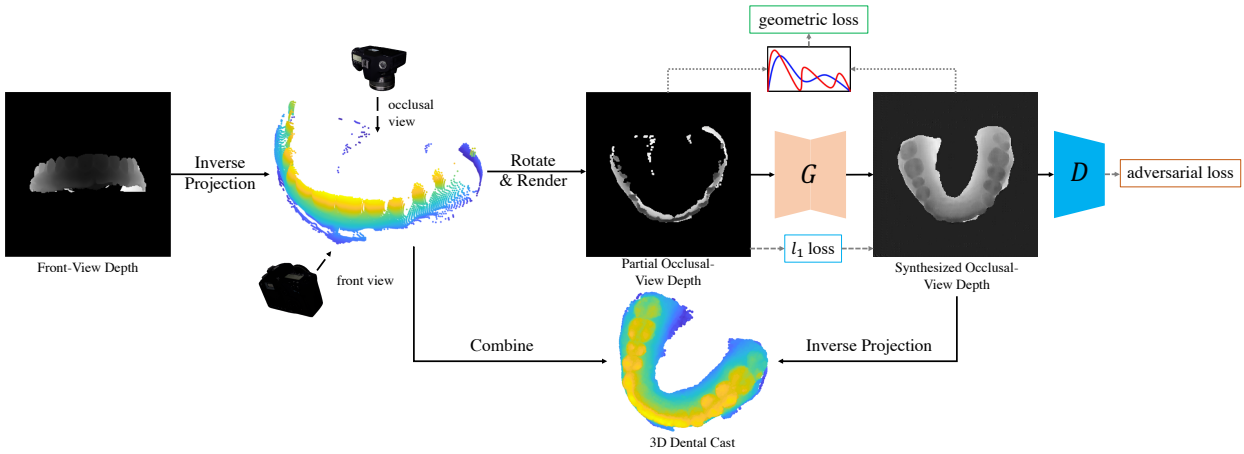


Figure 6.3: Novel view synthesis. We first inversely project a given depth map back to the 3D space to obtain a 3D point cloud, and then rotate the point cloud to the desired novel viewpoint. It is further rendered to a partial depth map x. We then use GANs to synthesize a full novel-view depth map y from the partial depth map. Adversarial loss, ℓ_1 loss and fine geometric fidelity loss are used for reconstructing a dental cast with coarse and fine geometric structures.

for depth reconstruction. Note that reconstructing high-quality depth maps from greyscale photos is not very challenging due to the consistency of rendering settings (such as light source).

6.3.3 Novel View Synthesis for Depth Maps

State-of-the-art models are able to synthesize novel views with fine details within a relatively small angle difference. We utilize the property of the depth map and synthesize a new view with a larger angle difference (90°).

Specifically, we first inversely project a given depth map back to the 3D space, and then rotate it to the desired novel viewpoint. It is further rendered to a partial depth map x. We then use GANs to synthesize a full novel-view depth map y from the partial depth map. Fig.6.3 depicts the pipeline of the novel view synthesis module.

Three loss terms are used to train the novel view synthesis module. First a GAN loss aims at training the generator G and the discriminator D adversarially:

$$L_{GAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_x[\log(1 - D(x, G(x)))] \tag{6.1}$$

where G tries to minimize this objective against an adversarial D that tries to maximize it.

The second term is the ℓ_1 loss to encourage synthesized depth map $G(x)$ and the ground-truth depth map to be pixel-wise similar:

$$L_{\ell_1} = \mathbb{E}_{x,y}[\|y - G(x)\|_1] \tag{6.2}$$

The last loss term, fine geometric fidelity loss, aims to preserve the 3D geometry of the synthesized and the ground-truth depth map. Specifically, we adopt contextual loss [199], and compare the performance of alternative choices in Section 6.4.6. Rather than comparing the pixel-wise difference, contextual loss handles distribution difference and thus can penalize the geometric difference between the synthesized depth map and the ground-truth depth map. Contextual loss can also be considered as an updated version of chamfer loss (a detailed comparison between contextual loss and chamfer loss can be found in [199]), which can be used for measuring the distance between two point clouds. Specifically, first the cosine distances $d(G(x)_i, y_j)$ are computed between all pairs $G(x)_i, y_j$. The distances are then normalized

$$\widetilde{d}_{ij} = \frac{d(G(x)_i, y_j)}{\min_k d(G(x)_i, y_k)} \tag{6.3}$$

Then, the pairwise affinities

$$A_{i,j} = \frac{\exp(1 - \frac{\widetilde{d}_{ij}}{h})}{\sum_l \exp(1 - \frac{\widetilde{d}_{il}}{h})} \in [0, 1] \tag{6.4}$$

can be computed. Note that h is a scalar bandwidth parameter and we set $h = 0.1$ in our paper. Then, the contextual similarity between images is defined as:

$$\mathrm{CX}(G(x), y) = \frac{1}{N} \sum_j \max_i A_{i,j} \tag{6.5}$$

where N is the number of pixels.

The contextual loss can thus be defined as:

$$L_{CX}(G(x), y) = -\log(\mathrm{CX}(G(x), y)) \tag{6.6}$$

Our final objective is a combination of the three loss terms controlled with corresponding loss weight λ:

$$G^* = \arg\min_G \max_D L_{GAN}(G, D) + \lambda_{\ell_1} L_{\ell_1}(G) + \lambda_{CX} L_{CX}(G) \tag{6.7}$$

Note that a major novelty of the paper is that we introduce a fine geometric fidelity loss for the novel view synthesis. We show that with the geometric loss, the reconstructed 3D shape avoids misalignment issues and has better fine geometric details.

6.3.4 3D Reconstruction

We project multiple synthesized depth maps back to 3D space to create the final rendering. We reconstruct 3D shapes from multi-view depth maps by first generating a 3D point cloud from each depth image with its corresponding camera setting. The union of these point

clouds from all views can be seen as an initial estimation of the shape. We also create silhouettes/masks from the depth maps by binarizing depths with a certain threshold. We then refine it by applying silhouettes to filter out noise points. A point will be kept only if all of its multi-view 2D projections are valid in the silhouettes. Supervision on the camera angles or distance from the 3D shape centroid is not explicitly provided to the models. The point clouds can be further converted to meshes based on triangulation for clinical use.

6.4 Experimental Results

We first present details of our dataset, evaluation metrics and implementation, followed by qualitative and quantitative results of each module of our model. Along the way, we provide comparisons with a state-of-the-art method [17]. We also conduct ablation studies and hyper-parameters analysis to understand the benefits of different modules.

6.4.1 Dataset and Evaluation Metrics

The proposed model is trained on a dataset consisting of 2,427 3D dental casts in the mesh format. We randomly select 80% (1,941) for training and 20% (486) for evaluation. Each 3D dental cast is aligned to a canonical pose and normalized for consistency and simplicity. Each mesh model is rendered to obtain the grayscale photo and the depth image with Pyrender.

To evaluate the performance of the proposed method, we use peak signal-to-noise ratio (PSNR) to measure the difference between the reconstruction and the ground-truth. Note that outputs for both the depth reconstruction and novel view synthesis are depth maps, so we evaluate the difference on the reconstructed and ground-truth depth map in the pixel space.

6.4.2 Implementation Details

Depth Reconstruction. We follow pix2pix [137] settings with 9-block ResNet backbone model. The weight of the ℓ_1 loss $\lambda_{\ell_1} = 100$. The network is trained for 200 epochs with initial learning rate 2e-4. Adam optimizer [152] is used for the parameter optimization.

Novel View Synthesis. We follow pix2pix [137] settings with 9-block ResNet backbone model. The weight of the ℓ_1 loss $\lambda_{\ell_1} = 100$ and the weight of the contextual loss $\lambda_{CX} = 10$. The network is trained for 200 epochs with initial learning rate 2e-4. Adam optimizer [152] is used for the parameter optimization.

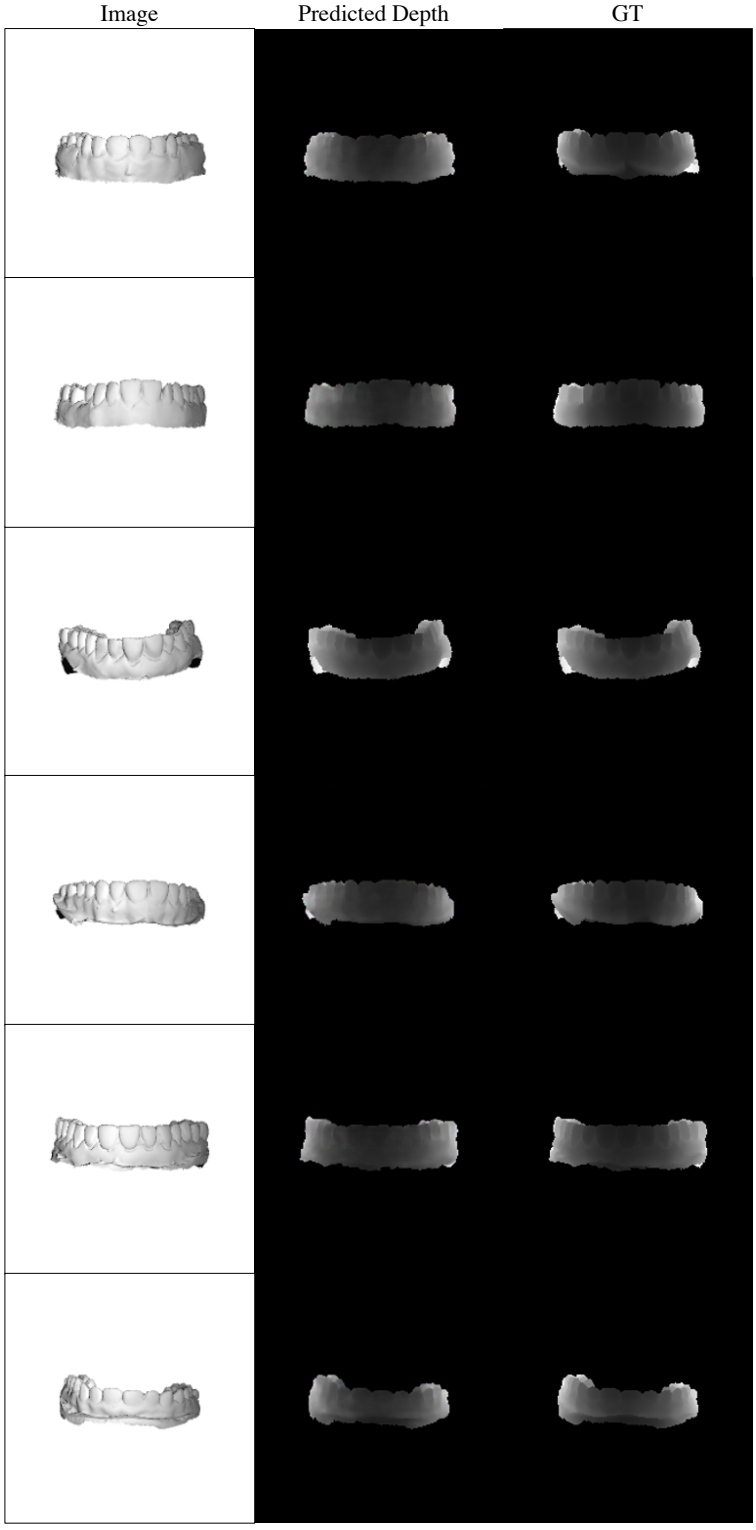


Figure 6.4: Depth prediction from photo results. Each row depicts an example. For each example, the column from left to right depicts: input photo, predicted depth map, and ground-truth depth map. The predicted and the ground-truth depth are visually similar, and the PSNR is 22.7 ± 2.0.

3D Reconstruction. This part is standard multi-view 3D reconstruction with no learning-based method involved. Specifically we create silhouettes/masks from the depth maps by binarizing depths with a certain threshold 0.25 and apply silhouettes to filter out noise points. In other words, for pixels with depth value smaller than 0.25 will be removed. The reconstructions are dense point clouds.

6.4.3 Depth Reconstruction

Fig.6.4 depicts the depth prediction from photo results. Specifically, we take front-view depth prediction as an example, although the model also works on other viewpoints. Each row shows an example, and for each example, the column from left to right depicts: input grayscale image, reconstructed depth map, and ground-truth depth map. Visually the reconstruction and the ground-truth depth maps are very similar. The PSNR between the reconstruction and the ground truth is 22.7 ± 2.0.

6.4.4 Novel-View Synthesis

Fig.6.5 depicts results of the novel view synthesis of depth maps. For consistency, we synthesize the occlusal view from the front view in this paper. The model also works on different input and output viewpoints. Synthesized and ground-truth depth maps are visually similar. Specifically, each tooth are well separated and the structure is relatively clear. Our model can handle dental casts that are not well-aligned to the canonical pose, although the reconstruction is slightly worse than the one with the canonical pose. The PSNR between the synthesized depth map and the ground truth is 16.3 ± 2.0.

6.4.5 3D Dental Cast Reconstruction

We project the depth maps back to the 3D space to obtain the 3D model. Fig.6.6 depicts the 3D reconstructions results. Dental cast with both coarse and fine geometrical fidelity are reconstructed, although geometric structures near the boundary have slightly worse details.

We find that reconstructing from front-view and occlusal view only already gives satisfactory 3D dental cast so only use two views for reconstruction in this paper. Our method can also synthesize more novel views from or to other viewpoints (for example, side-views) and perform multi-view 3D reconstruction from them.

6.4.6 Ablation Studies

As mentioned before, there are alternative choices for the fine geometrical fidelity loss. Fig.6.7 and Table 6.1 show the qualitative and quantitative results of novel view synthesis with no geometric loss [17], with chamfer loss and with contextual loss.

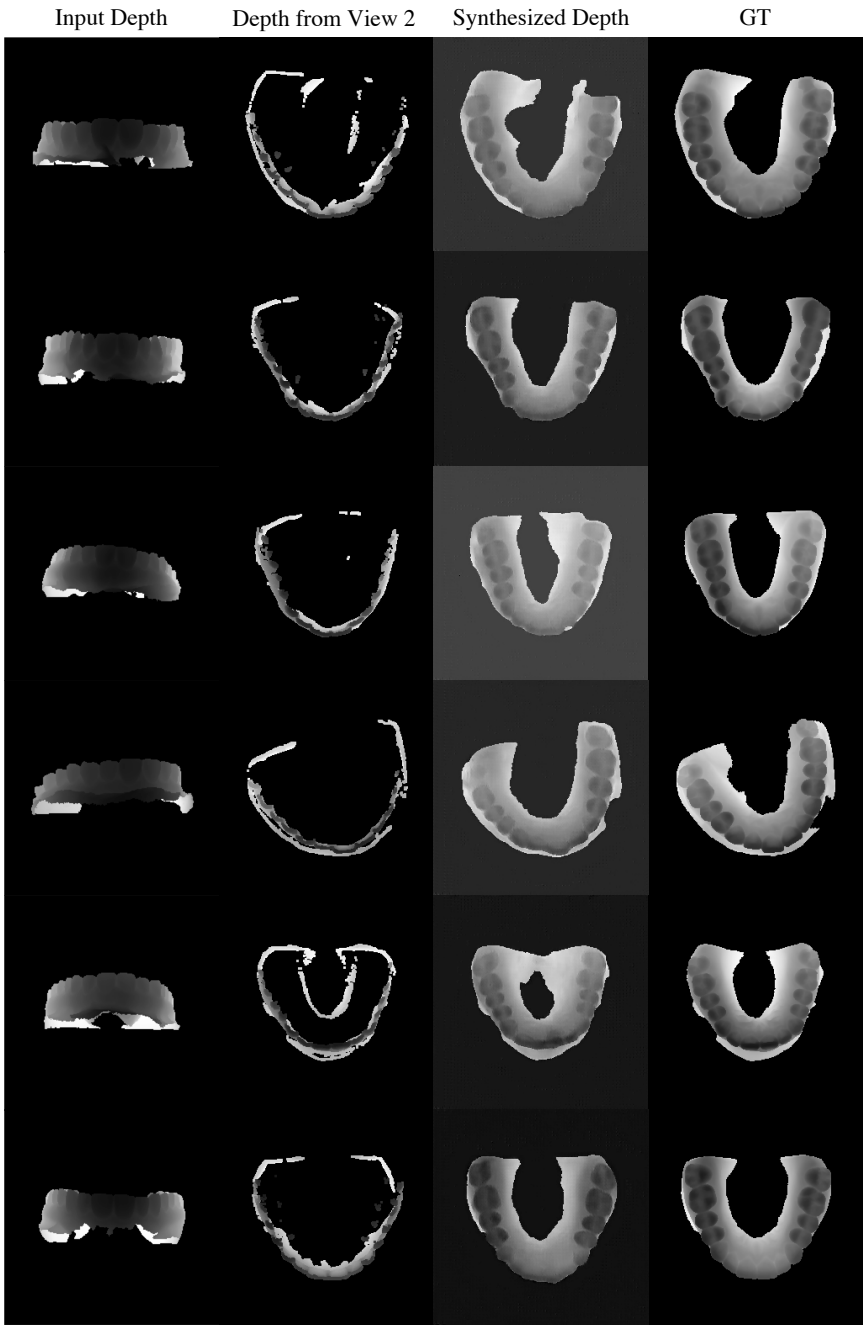


Figure 6.5: Novel view synthesis results. Each row depicts an example. For each example, the column from left to right depicts: input depth map, depth map from a different viewpoint (occlusal-view), synthesized depth map, and ground-truth depth map. The PSNR between the synthesized depth map and the ground truth is 16.3 ± 2.0.

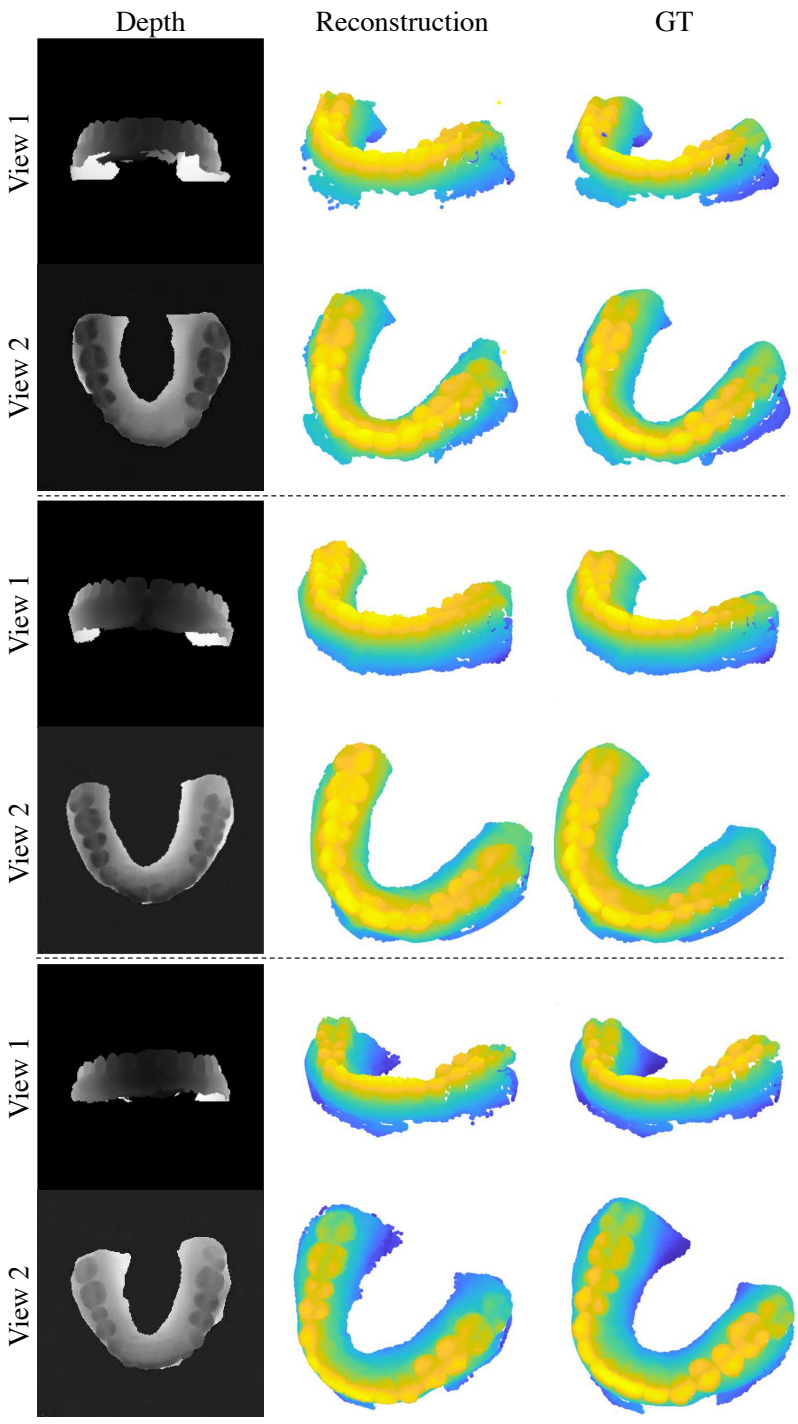


Figure 6.6: 3D reconstruction results. View 1 (front-view) is the input to the view synthesis module (and originally predicted from the front-view photo) while view 2 (occlusal view) is the synthesized depth map from view 1. Dental cast with both coarse and fine geometrical fidelity are reconstructed, although geometric structures near the boundary have slightly worse details.

Figure 6.7: Novel view synthesis with no geometric loss, with chamfer loss and with contextual loss. Without geometric loss, the reconstructed 3D shapes suffers misalignment issues and different teeth has worse local geometric structures. Adding chamfer loss alleviate the misalignment issue, but local structures (especially for teeth near the boundary) can still improve. Contextual loss solves the misalignment issue and reconstructs the 3D dental cast with both good global and local geometric details.

Table 6.1: PSNR performance of different choices of fine geometric fidelity loss. The contextual loss achieves the best result.

	No geometric loss [17]	**Chamfer loss & Contextual loss**	
PSNR	19.2 ± 2.5	17.1 ± 2.5	16.3 ± 2.0

The contextual loss achieves the highest PSNR at 16.3, outperforming [17] and chamfer loss by 2.9 and 0.8, respectively. Visually the contextual loss outperforms alternative choices and reconstructs 3D shapes with coarse and fine geometrical fidelity.

6.4.7 Hyper-Parameter Analysis

Table 6.2: PSNR performance of different weight λ_{ℓ_1} of the contextual loss. $\lambda_{\ell_1} = 10$ gives the best result.

	0.1	**1**	**10**	**100**
PSNR	17.0 ± 2.3	16.5 ± 2.0	16.3 ± 2.0	16.6 ± 2.1

We study the influence of the hyper-parameter, weight λ_{ℓ_1} of the contextual loss (Table 6.2). While changing weight λ_{ℓ_1} does not affect the performance much, setting $\lambda_{\ell_1} = 10$ gives the best result

6.5 Summary

We propose a novel approach to reconstructing a 3D dental cast from a singe image. With GANs and fine geometric fidelity loss to achieve both coarse and fine geometrical fidelity, our approach learns priors of teeth structures and reconstructs a dental cast with both good global and local geometric details. The efficient and low-cost method can be of significance in dental clinics. The proposed approach also improves state-of-the-art single-view 3D reconstruction algorithms by reconstructing 3D shapes with both coarse and fine geometrical fidelity with limited additional computational overhead.

Part III

Interpretable Models from Structure-Aware Representations

Chapter 7

Interpretable Demographics Prediction from Meibography Images

7.1 Introduction

The Meibomian glands of the human eyelid secrete lipid-rich meibum that during blinking forms a thin film on the surface of the tears [155, 38] that serves to inhibit evaporation of the tears and stabilize the tear film by reducing surface tension [35, 75]. Dysfunction of the Meibomian glands leading to insufficient or poor quality lipids is a primary cause of dry eye (DE) [125, 286], a globally impactful and highly prevalent ocular surface disease [225]. Infrared meibography is the biomedical imaging of the glands, exposed by everting the eyelids, using a thermographic camera. Meibography has been increasingly used in recent years for clinical diagnosis and treatment of Meibomian gland dysfunction (MGD) as well as in clinical research on MGD and DE; however, there are few studies that have examined how the detailed morphological structure of these glands relates to the signs and symptoms of MGD and DE [18, 4, 58]. Furthermore, it is unknown how the detailed morphology of the Meibomian glands relates to subject demographic characteristics such as age, gender, and ethnicity – all of which are well-documented factors in the prevalence and severity of MGD and DE [12, 150, 290, 310, 142].

There are a number of studies to date that have employed traditional statistical techniques to examine broad, global assessments of the Meibomian glands such as the overall percent area of gland atrophy (e.g., the 4-level meiboscore of Arita [11], the 5-level score and software-based atrophy area of Pult [236]) and their relationships to various subject characteristics. With few exceptions, however, the detailed, local morphological characteristics of the glands in meibography images (e.g., length, width, tortuosity, local contrast) have not been studied with respect to subject characteristics or clinical outcomes. A major impediment to detailed morphological analysis of meibography images and to its use in both research and clinical eyecare has been the technical difficulty and time-consuming nature of quantifying local meibography features [63]. In a previous work we developed a deep learning

model that proved capable of quickly and automatically identifying and quantifying eight different metrics describing both global and local morphological features in novel meibography images with good accuracy [306]. In this study we will build on this work by training a supervised machine learning model to identify and quantify the morphological features observed in meibography images and to then use these images and image-derived metrics to predict the demographic characteristics of the subjects who provided them.

The significance of attempting to predict subject demographics from meibography images is twofold. First, it offers an alternative approach to traditional associative statistical modeling for determining whether the morphology of the Meibomian glands differs depending on age, gender, or ethnicity – all known factors in MGD and DE. Rather than testing a null hypothesis under certain assumptions, we train the machine learning model not only to learn to use meibography image features to predict subject demographic characteristics but to reveal what the most highly weighted image features were in contributing to this prediction. This could shed further light on the etiology of MGD at a more detailed level, and possibly reveal novel relationships.

Second, de-identified biomedical imaging is not currently considered Protected Health Information (PHI), and is therefore not subject to the strict regulations on its use, sharing, storage and transmission [187, 212, 305]. These regulations, however, are in active debate and are likely to evolve rapidly, as ocular features such as retinal vein patterns, eye movements, and iris patterns have been proven to provide unique biometric "fingerprints" that can be used to identify individuals with a high level of accuracy [140, 253, 196]. It seems reasonable, given the highly detailed morphology revealed in meibography images, that meibography could be developed into a biometric identifier as well, thus requiring far greater patient safety and privacy protections – even for de-identified images – than it is subject to today. The field of artificial intelligence and machine learning is evolving rapidly and its capabilities ever-expanding. If we can train a machine learning model now to take de-identified meibography images as input, and based solely on the detailed morphology revealed in those images, reconstruct some characteristics of the subjects that provided those images, it is very easy to imagine with larger training datasets and improved models, being able in the near future to reveal a patient's individual identity.

7.2 Methods

7.2.1 Development and Evaluation Dataset

This study utilized a meibography image dataset from a previous paper [306] along with corresponding subject demographic information for deep learning algorithm development and evaluation.

Subject recruitment and Imaging. Adult human subjects (age $\geq$ 18 years) were recruited from the University of California, Berkeley campus and surrounding community for single-visit ocular surface evaluations during the period from 2012 to 2017. Eligible

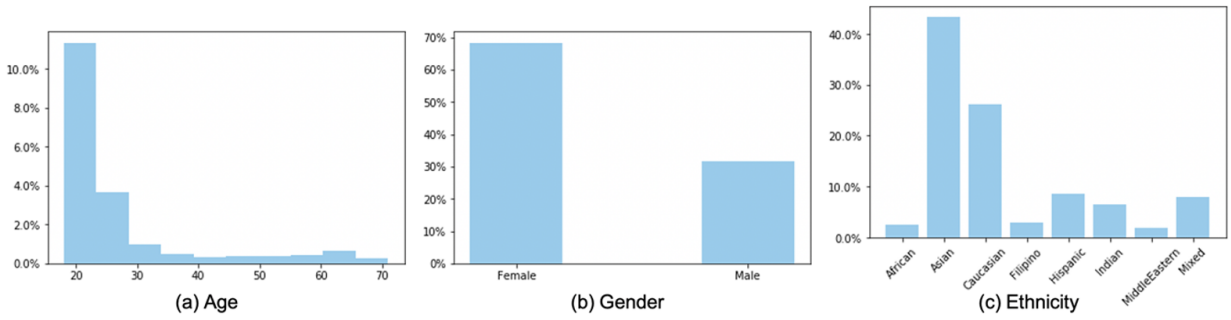


Figure 7.1: Histogram (in percentage) of demographic features of our meibography image dataset

subjects were free of any eye conditions contraindicating meibography, not currently taking medications with effects on the anterior eye or adnexa, and with no history of ocular surgery. Meibography images of the upper eyelids for both eyes were captured with the OCULUS Keratograph 5M (OCULUS, Arlington, WA), a clinical instrument that uses an infrared light with wavelength 880 nm for Meibomian gland imaging [347]. During image capture, the ambient light was off with the subject's head positioned on a chin rest and forehead strap. A total of 750 images were collected and prescreened to rule out images that did not capture the entire upper eyelid (61 images or 8.90%); the remaining 689 images were used in the analysis.

Demographics. Subject demographics were documented during the visit. Three demographic characteristics were studied in this paper, namely age, gender and ethnicity. Histograms depicting the distributions of these demographic features are presented in Figure 7.1. Specifically for ethnicity, only Caucasians and Asians were considered due to lack of sufficient subjects of other ethnicities. The total number of images used for ethnicity prediction is thus 421 after ruling out subjects with other ethnicities, while the numbers of images used for age and gender prediction both stay 689.

Morphological features. The development of an interpretable deep learning model for predicting demographic characteristics requires morphological features such as gland length and tortuosity as data sources. Eight morphological features were quantified for each meibography image: number of glands, gland density, percent area of gland atrophy, gland local contrast, gland length (mm), gland width (mm), gland tortuosity and percentage of ghost glands. Histograms of these morphological features are presented in Figure 7.2.

Data partitioning. Meibography images were partitioned into two non-overlapping subsets for training and evaluating the deep learning model. The partition was based on image collection time. Specifically, images collected from years ranging from 2015 to 2017 were combined to be the development set, while those collected from years ranging from 2012 to 2013 were combined to be the evaluation set. All images were taken with the same instrument under the same protocol. The development set was further divided randomly

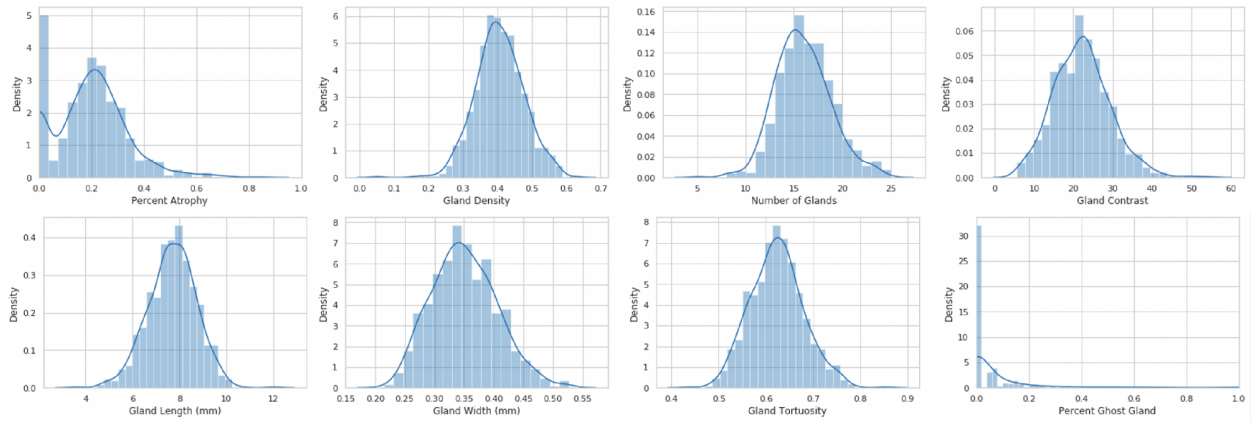


Figure 7.2: Histogram and density plots of learned morphological features from meibography images

Table 7.1: Subject demographics of the meibography image dataset used in the study

	Development		
	Train	Tune	Evaluation
Images (n)	389	97	203
Subject Demographics			
Subjects (n)	260	94	109
Mean (SD) Age (yrs)	27.8 (13.1)	27.0 (11.5)	27.9 (12.7)
% Female Subjects	69.6	66	69.4

into 2 subsets for training and validating the model. Specifically, the validation set was used to find the best hyperparameters (e.g., model learning rate) for the model that was trained on the training set. The evaluation set was for evaluating or testing the performance of the model. Subject demographics stratified on development and evaluation datasets can be found in Table 1. Different subsets had similar demographic feature distributions, so that the distributional shift between the training and evaluation sets was minimized.

7.2.2 Algorithm Design and Training

The overall goal is to design an interpretable deep learning model that can predict the demographic characteristics of a subject. Interpretability requires the model to be able to identify the most highly weighted morphological features used by the algorithm to predict the demographic characteristics of a subject directly from their meibography image. A two-stage model was designed with a first stage attribute learning model to identify and quantify

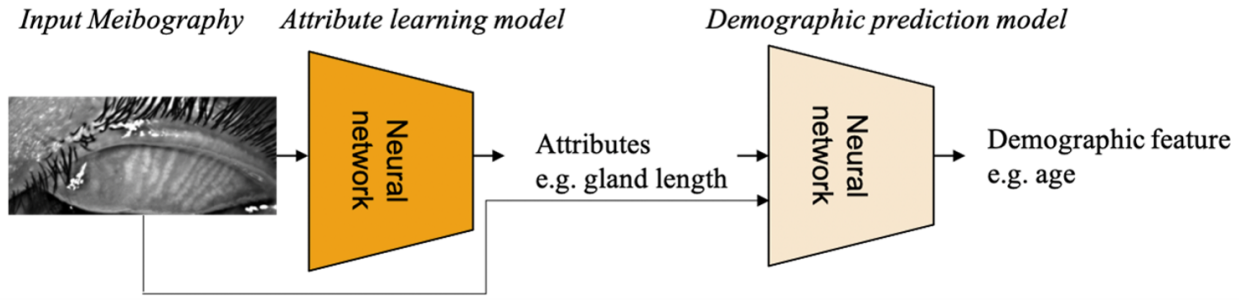


Figure 7.3: Overall pipeline of the proposed interpretable deep learning model for predicting demographics from images

Table 7.2: Means and standard deviations of morphological features

	% atrophy	Gland density	# glands	Gland local contrast[1]	Gland length (mm)	Gland width (mm)	Gland tortuosity	% ghost gland[2]
Mean	19.70%	40.70%	15.9	22.2	7.7	0.35	63.60%	7.80%
Standard deviation	13.80%	7.10%	2.8	7.2	1	0.05	10.60%	16.90%

morphological features from input meibography images, and a second stage demographic prediction model to predict subject demographic features from meibography images and corresponding first-stage morphological features. Figure 7.3 depicts the overall pipeline.

Deep attribute learning. In the first stage, a deep learning model was developed to predict and quantify the morphological features of a given meibography image (first part of Figure 7.3). The primary goal of the attribute learning model is to provide value ranges rather than exact values of morphological features for the final demographic predictions. There are two underlying reasons for this: 1) Predicting coarser value ranges is easier than predicting precise values for the deep learning model, especially since the dataset (689 images in total) was not sufficiently large-scale to learn precise morphological feature values. 2) Morphological attribute prediction was an intermediate result, with the major purpose of interpreting relationships between demographic features and morphological features. Predicting value ranges is adequate for the purpose. For example, it would be acceptable for predicting gender to find that females exhibit a high probability of having > 15 glands rather than a high probability of having exactly 16 glands. Therefore, our first stage deep learning model predicts morphological features to fall within ordinal ranges (or, in the case of ghost gland percentage, binary classes).

The morphological attribute learning model specifically predicts a ternary level rather than an exact numerical value for each morphological feature. As depicted in Figure 7.4, the model predicts each morphological feature value to fall below $\mu - \sigma$ (level 1), between $\mu - \sigma$

[1]Measured in pixel intensity of the meibography image. The lowest intensity the sensor could detect was 0 and the highest was 255.

[2]For the percentage of ghost glands prediction, a binary classification was used ($= 0$, > 0).

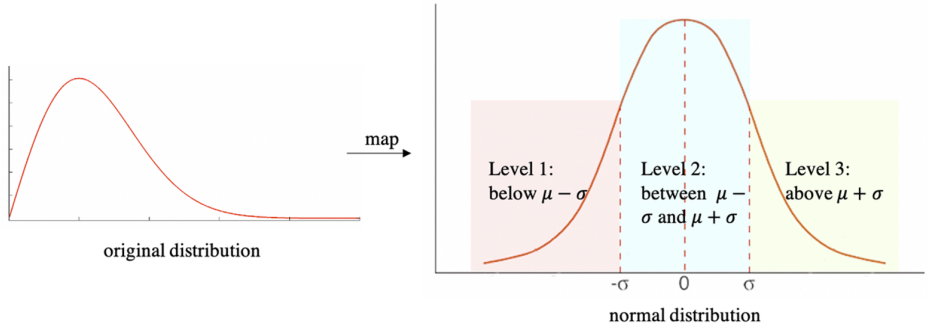


Figure 7.4: The original distribution of a morphological features is mapped to a normal distribution. The deep attribute learning model predicts if a morphological feature value is below $\mu - \sigma$, between $\mu - \sigma$ and $\mu + \sigma$, or above $\mu + \sigma$, where μ, σ refer to the mean and standard deviation of the original morphological feature value distribution.

and $\mu+\sigma$ (level 2), or above $\mu+\sigma$ (level 3), where μ and σ refer to the mean and standard deviation of the morphological feature predicted value distribution. Table 2 provides the μ and σ for all morphological features investigated. Specifically for the percentage of ghost glands, 77.1% images (or 531 images) have 0 ghost glands. Therefore, for the percentage prediction, a binary class was used (percentage of ghost glands $=0$ or > 0).

Demographic feature prediction. In the second stage, a deep learning model was developed to predict demographic features from both meibography images and corresponding attributes from the attribute learning model in stage one (second part of Figure 7.3). Specifically, a given image was input to ResNet18 [113] (a residual neural network of 18 convolution layers) to obtain a 64-dimensional vector. The vector can be considered as an embedding that encodes information of the image. The vector was combined with 8 predicted vectors of morphological features from the deep attribute learning model. All vectors are of the same dimension. The combined 9 vectors were fed to a fully convolutional layer for predicting the demographic features.

Among three demographic features to be predicted, gender and ethnicity are categorical, while age is continuous numerical. Following Dana et al. [62] based on dry eye prevalence, subject age has been stratified into 3 categories: 1) $\leq$39 years old, 2) $> 39, <$ 50 years old, and 3) $\geq$50 years old. The final output of the demographic prediction model can be interpreted by analyzing the learned coefficients of the morphologic features used to predict the demographic characteristics. Higher coefficient values indicate a stronger effect of a morphological feature in predicting a demographic feature.

7.2.3 Evaluation Metrics

The model is trained on the training set with varying hyperparameters (e.g., different learning rates) and the highest performance model on the validation set was selected for the final

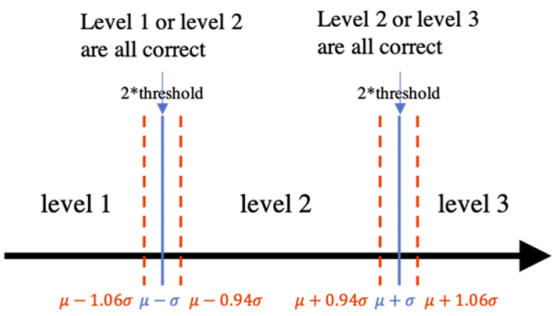


Figure 7.5: The relaxed morphological feature classification evaluation rule with the tolerance threshold. The tolerance threshold was set at 0.06σ, and classifying images with percent atrophy $\mu - 1.06\sigma$ to $\mu - 0.94\sigma$, $\mu - 0.06\sigma$ to $\mu + 0.06\sigma$, and $\mu + 0.94\sigma$ to $\mu + 1.06\sigma$ either to its ground-truth or adjacent level were both considered as correct prediction.

evaluation on the evaluation set. The highest performance models were selected for attribute learning and demographic prediction, respectively. Both were evaluated with classification accuracy.

Classification evaluation with tolerance threshold. The evaluation technique was used for evaluating deep attribute learning performance. As described in the previous section, the deep attribute learning model predicts the trinary level of each morphological feature. However, near the transition limits of different levels ($\mu - \sigma$ and $\mu+\sigma$), the morphological features may be very similar and difficult to classify. A similar technique described in Wang et al. [302] was applied here. A tolerance threshold near the grading transition limit was necessary. As illustrated in Figure 7.5, the tolerance threshold was set at 0.03σ, and classifying morphological feature values within $\mu - 1.06\sigma$ to $\mu - 0.94\sigma$, and $\mu+0.94\sigma$ to $\mu + 1.06\sigma$ either to their ground-truth or adjacent level were both considered as correct predictions. Note that the tolerance threshold does not apply to predictions of the percentage of ghost glands as that is a binary classification.

Five-fold cross validation. For evaluating both attribute learning and demographic prediction performance, in additional to reporting classification accuracy on the evaluation set with the best performing model on the validation set, five-fold cross validation accuracy is also reported. First, the entire dataset (including both development and evaluation subsets) was randomly split into 5 folds. Second, 5 iterations of training and evaluation were conducted. At each iteration, 4 folds were used for training and the remaining fold for evaluation. The mean and standard deviation of the classification accuracy on each fold were reported as five-fold cross validation accuracy.

Table 7.3: Classification accuracy for morphological feature prediction of the attribute learning model. Note that the second row reports accuracies on the evaluation set and the last two rows reports means and standard deviations of accuracies on the 5-fold cross validation set.

	Percent atrophy	Gland density	Number of glands	Gland contrast	Gland length	Gland width	Gland tortuosity	Percent ghost gland	Average
Evaluation accuracy (%)	73.1	75.2	82.1	78.5	76.6	76.1	78.4	72.1	76.5
Cross-validation accuracy - mean (%)	73	76.7	80.1	79.1	77.7	76	78.3	71.7	76.6
SD (%)	3.9	4.1	4	4.5	3.8	4	4.3	3.8	4

7.3 Results

7.3.1 Attribute Prediction Performance

Table 3 reports the performance of the attribute prediction accuracy for morphological features. Specifically, model accuracies on the evaluation set and 5-fold cross validation are reported. For attribute prediction (including percent atrophy, gland density, number of glands, gland local contrast, gland length, gland width, gland tortuosity and percentage of ghost glands), the model achieved 76.5% accuracy on the evaluation set, and 76.6% average accuracy for five-fold cross validation. The average standard deviation of the 5-fold cross validation is 4.0%.

7.3.2 Demographic Prediction Performance

For each subject demographic feature, classification accuracy of the demographic prediction model is reported. Similarly, accuracies on the evaluation set and 5-fold cross validation are reported. Additionally, the coefficients for morphological features were also analyzed by ranking coefficients from largest to smallest, thereby identifying the most highly weighted morphological features for demographic prediction.

It is also of interest to understand how demographic features change with morphological features. One way is to compare average values of morphological features of different demographic groups. For each split in the five-fold cross validation, morphological features ranked from most highly weighted to least were recorded. For both age and ethnicity, the top 2 most important features were observed to be consistent for every split. Average values for the top 2 most important morphological features sorted by demographic groups are reported below.

Age. The 5-fold mean (SD) classification accuracy for age was 75.7 (4.5)% (Table 4 upper). Coefficient analysis showed the two most important morphological features for determining age were percent area of gland atrophy and percentage of ghost glands (Table 4 lower). For the three age groups from youngest to oldest, the average percent area of atrophy increases from 18.1% to 25.2% to 33.6%, while the average percentage of ghost glands increases from 5.6% to 14.2% to 28.7%. Older subjects tend to have a higher percent area

Table 7.4: Upper: Classification accuracy for subject age prediction of the demographic feature prediction model. Note that the second row reports accuracies on the evaluation set and the last two rows reports means and standard deviations of accuracies on the 5-fold cross validation set. Lower: Average values for the top 2 most highly weighted (largest model coefficients) morphological features stratified on age group.

	Age ≤ 39	**Age <39, >50**	**Age ≥ 50**	**Average**
Evaluation accuracy (%)	86.1	70.1	71.1	75.8
Cross-validation accuracy - mean (%)	85.9	71.1	70.2	75.7
SD (%)	4.4	4.5	4.5	4.5

	Age ≤ 39	**Age <39, >50**	**Age ≥ 50**
Avg. % atrophy	18.1	25.2	33.6
Avg. % ghost glands	5.6	14.2	28.7

Table 7.5: Classification accuracy for subject ethnicity prediction of the demographic feature prediction model. Note that the second row reports accuracies on the evaluation set and the last two rows reports means and standard deviations of accuracies on the 5-fold cross validation set. Lower: Average values for the top 2 most highly weighted (largest model coefficients) morphological features stratified on ethnic group.

	Caucasian	**Asian**	**Average**
Evaluation accuracy (%)	85.1	86.4	85.8
Cross-validation accuracy – mean (%)	84.6	86.8	85.7
SD (%)	4.9	4	4.5

	Caucasian	**Asian**
Avg. gland density (%)	39.2	42
Avg. % ghost glands	10.5	7.9

of atrophy and a higher percentage of ghost glands compared with younger subjects.

Gender. The 5-fold mean (SD) classification accuracy for gender was 56.5 (5.0)% which is close to a random guess accuracy of 50%. Therefore, gender could not be accurately predicted from meibography images with the proposed deep learning model, nor could important morphological differences be identified.

Ethnicity. As shown in the histogram of ethnicity (Figure 7.1c), although the entire dataset was comprised of subjects of eight different ethnicities, most ethnicities did not have sufficient sample sizes for trustworthy predictions to be obtained. Therefore, predictions were only conducted for Asian and Caucasian subjects (75% of all subjects). The 5-fold mean (SD) classification accuracy for ethnicity (predicting if the subject is Asian or Caucasian) was

85.7 (4.5)% (Table 5 upper). Coefficient analysis showed the two most important Meibomian gland morphological features for determining ethnicity were gland density and percentage of ghost glands (Table 5 lower). Asian subjects exhibited approximately 2.8% greater gland density than Caucasians. Asian subjects exhibited 2.6% fewer ghost glands than Caucasians.

7.4 Discussion

The work presents an interpretable deep learning model to predict demographics from meibography images. The proposed approach makes the following two contributions: 1) Using deep learning models to find morphological features predictive of demographics offers an alternative to traditional associative modeling and may reveal new relationships; 2) The proposed approach investigates an early stage of the technology that could be used to develop meibography images into a biometric fingerprint capable of identifying individuals.

Previous studies have explored the associations between subject demographics and the symptoms of MGD and DE [290, 310, 142]. In this study we explored associations between subject demographics and specific Meibomian gland morphological features. The results showed that older subjects had a higher percent area of gland atrophy and a higher percentage of ghost glands. Previous works [8, 133] have identified age-related changes Meibomian gland structure and function, including changes to the acini, loss of progenitor stem cells, abnormal meibum secretion, and MGD. The proposed method also identified age-related changes with a new focus on changes in gland morphology which was quantitatively analyzed directly from meibography images.

As for ethnicity, Caucasians subjects exhibited a higher percentage of ghost glands and lower gland density compared with Asian subjects. Previous work [150] observed ethnic differences (specifically between Asians and Caucasians) in Meibomian gland morphological patterns in the pediatric population by qualitatively identifying MG morphoclimatic changes such as presence of gland shortening and gland tortuosity. Major differences compared to this study lie in the efficiency and extensibility. Each subject need to be manually annotated for morphological features in their study, while this study used a deep-learning-based model to automatically analyze morphological features from subject meibography. This work thus studied 10 times as many as subjects in their study (70), and no additional manual annotation is needed for new subjects.

The proposed deep learning model was unable to predict gender with high accuracy. Previous works [111] however identified changes in the anatomic lid margin and gland morphology in different gender groups. Hypotheses for the discrepancies include: 1) Number of subjects in this study was not large enough to identify intergroup differences, 2) The proposed deep learning method was unable to identify intergroup differences, 3) Meibography image only is unable to predict the subject gender, and clinical signs such as tear film breakup time is also necessary for the prediction. Future work is warranted to verify the hypotheses and study if meibography images encode enough gland morphological information for revealing subject gender.

In the United States, health care providers and researchers are held accountable for the privacy and security of protected health information (PHI) and individually identifiable health information [2]. However, the frequency and magnitude of health care data breaches continues to climb [261], with significant impacts for patients in the areas of identity theft, faulty treatment, insurance coverage, job security, financial well-being, and mental health [234, 57]. While the majority of a person's medical and health information is considered PHI and thus subject to strict regulations on its use, storage, and dissemination, this is currently not the case for de-identified medical imaging [187, 212, 305]. This is already an area of active debate, as biometric identifiers of unique individuals continue to be developed, and many types of medical images are highly unlikely to be exactly homomorphic between individuals, particularly with today's high resolution imaging and sophisticated image processing software. In the case of the eye, retinal vein patterns, iris patterns, and idiosyncratic eye movement patterns have all been shown to be accurate biometrics [140, 253, 196]. Meibography, given the highly detailed morphology of the Meibomian glands, could also serve as a biometric identifier with some further development of current technology. In this study, we have shown that a relatively straightforward deep learning algorithm trained on a fairly small dataset was capable of extracting some of the demographic characteristics of the subjects that provided the de-identified meibography images. Further development of the models and larger training datasets will certainly gain in accuracy and specificity, with individually identifiable meibography images on the horizon. While there is a potential for meibography's use as a highly accurate biometric identifier, there is also concern with the lack of regulation and enforcement of the privacy and security of such images (assuming they are de-identified). Considering that combining data from multiple sources (e.g., multi-modal biometrics) significantly increases the likelihood of accurate individual identification [9, 156], it seems clear that urgent updates are needed in the regulations and enforcement governing the security of all biomedical imaging of patients and research subjects, regardless of whether it has been anonymized.

The study has certain limitations. Only Meibomian gland morphology in the central upper eyelid was analyzed as imaging the central region of the tarsal plate with an optimal focus causes defocus of the peripheral glands. This work thus only presents results on demographic features and central tarsal plate morphological features as morphological features of peripheral glands were not used in this analysis. For ethnicity, due to limited and imbalanced samples for some races, only Asian and Caucasian meibography images were distinguished using this deep learning approach. This leaves morphological features for other ethnic groups undiscovered.

In conclusion, an interpretable deep learning model to predict demographic characteristics from meibography images was developed. The model could be helpful in furthering the understanding of the relationships between local features of Meibomian gland morphology and subject demographics. Future work will extend the model to investigate detailed aspects of Meibomian gland morphology and the signs and symptoms of MGD and DE. Finally, this work suggests that de-identified meibography images – currently not considered PHI and thus not subject to strict regulations on their use and dissemination – could in the future

be developed into biometric identifiers of individuals with the rapidly evolving capabilities of artificial intelligence.

Part IV

Improving the Learning Efficiency

Chapter 8

Orthogonal Convolutional Neural Networks

8.1 Introduction

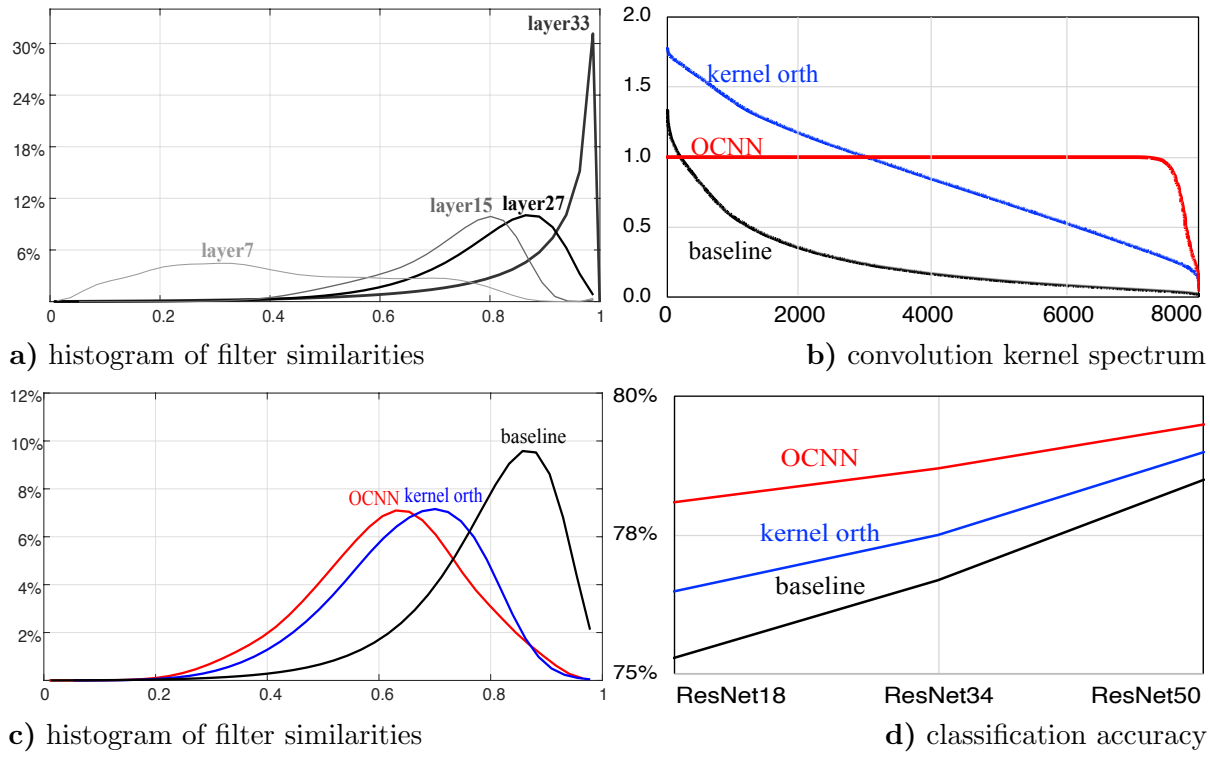
While convolutional neural networks (CNNs) are widely successful [161, 67, 265], several challenges still exist: over parameterization or under utilization of model capacity [101, 52], exploding or vanishing gradients [29, 94], growth in saddle points [64], and shifts in feature statistics [136]. Through our analysis to solve these issues, we observe that convolutional filters learned in deeper layers are not only highly correlated and thus redundant (Fig.8.1a), but that each layer also has a long-tailed spectrum as a linear operator (Fig.8.1b), contributing to unstable training performance from exploding or vanishing gradients.

We propose *orthogonal CNN* (OCNN), where a convolutional layer is regularized with orthogonality constraints during training. When filters are learned to be as orthogonal as possible, they become de-correlated. Their filter responses are much less redundant. Therefore, the model capacity is better utilized, which improves the feature expressiveness and consequently the task performance.

Specifically, we show that simply by regularizing convolutions with our orthogonality loss during training, networks produce more uniform spectra (Fig.8.1b) and more diverse features (Fig.8.1c), delivering consistent performance gains with various network architectures (Fig.8.1d) on various tasks, e.g. image classification/retrieval, image inpainting, image generation, and adversarial attacks (Table 8.1).

Many works have proposed the orthogonality of linear operations as a type of regularization in training deep neural networks. Such a regularization improves the stability and performance of CNNs [25, 329, 24, 23], since it can preserve energy, make spectra uniform [354], stabilize the activation distribution in different network layers [251], and remedy the exploding or vanishing gradient issues [14].

Existing works impose orthogonality constraints as kernel orthogonality, whereas ours directly implements orthogonal convolutions, based on an entirely different formulation of a



a) histogram of filter similarities **b)** convolution kernel spectrum

c) histogram of filter similarities **d)** classification accuracy

Figure 8.1: Our *OCNN* can remove correlations among filters and result in consistent performance gain over standard convolution *baseline* and alternative kernel orthogonality baseline (*kernel orth*) during testing. **a)** Normalized histograms of pairwise filter similarities of ResNet34 for ImageNet classification show increasing correlation among standard convolutional filters with depth. **b)** A standard convolutional layer has a long-tailed spectrum. While kernel orthogonality widens the spectrum, our OCNN can produce a more ideal uniform spectrum. **c)** Filter similarity (for layer 27 in **a**) is reduced most with our OCNN. **d**) Classification accuracy on CIFAR100 always increases the most with our OCNN.

convolutional layer as a linear operator.

Orthogonality for a convolutional layer $Y = \text{Conv}(K, X)$ can be introduced in two different forms (Fig.8.2).

1. **Kernel orthogonality** methods [329, 24, 23] view convolution as multiplication between the kernel matrix K and the *im2col* [335, 121] matrix $\widetilde{X}$, *i.e.* $Y = K\widetilde{X}$. The orthogonality is enforced by penalizing the disparity between the Gram matrix of kernel K and the identity matrix, *i.e.* $\|KK^T - I\|$. However, the construction of $\widetilde{X}$ from input X is also a linear operation $\widetilde{X} = QX$, and Q has a highly nonuniform spectrum.
2. **Orthogonal convolution** keeps the input X and the output Y intact by connecting them with a doubly block-Toeplitz (DBT) matrix $\mathcal{K}$ of filter K, i.e. $Y = \mathcal{K}X$

Table 8.1: Summary of experiments and OCNN gains

Task		Metric	Gain
Image Classification	CIFAR100	classification accuracy	3%
	ImageNet	classification accuracy	1%
	semi-supervised learning	classification accuracy	3%
Feature Quality	fine-grained image retrieval	kNN classification accuracy	3%
	unsupervised image inpainting	PSNR	4.3
	image generation	FID	1.3
	deep metric learning	NMI	1.2
Robustness	black box attack	attack time	7x less

and enforces the orthogonality of $\mathcal{K}$ directly. We can thus directly analyze the linear transformation properties between the input X and the output Y.

Existing works on CNNs adopt kernel orthogonality, due to its direct filter representation.

We prove that kernel orthogonality is in fact only necessary but not sufficient for orthogonal convolutions. Consequently, the spectrum of a convolutional layer is still non-uniform and exhibits a wide variation even when the kernel matrix K itself is orthogonal (Fig.8.1b).

More recent works propose to improve the kernel orthogonality by normalizing spectral norms [203], regularizing mutual coherence [25], and penalizing off-diagonal elements [33]. Despite the improved stability and performance, the orthogonality of K is insufficient to make a linear convolutional layer orthogonal among its filters. In contrast, we adopt the DBT matrix form, and regularize $\|\text{Conv}(K, K) - I_r\|$ instead. While the kernel K is indirectly represented in the DBT matrix $\mathcal{K}$, the representation of input X and output Y is intact and thus the orthogonality property of their transformation can be directly enforced.

We show that our regularization enforces orthogonal convolutions more effectively than kernel orthogonality methods, and we further develop an efficient approach for our OCNN regularization.

To summarize, we make the following contributions.

1. We provide an equivalence condition for orthogonal convolutions and develop efficient algorithms to implement orthogonal convolutions for CNNs.
2. With no additional parameters and little computational overhead, our OCNN consistently outperforms other orthogonal regularizers on image classification, generation, retrieval, and inpainting under supervised, semi-supervised, and unsupervised settings.

Better feature expressiveness, reduced feature correlation, more uniform spectrum, and enhanced adversarial robustness may underlie our performance gain.

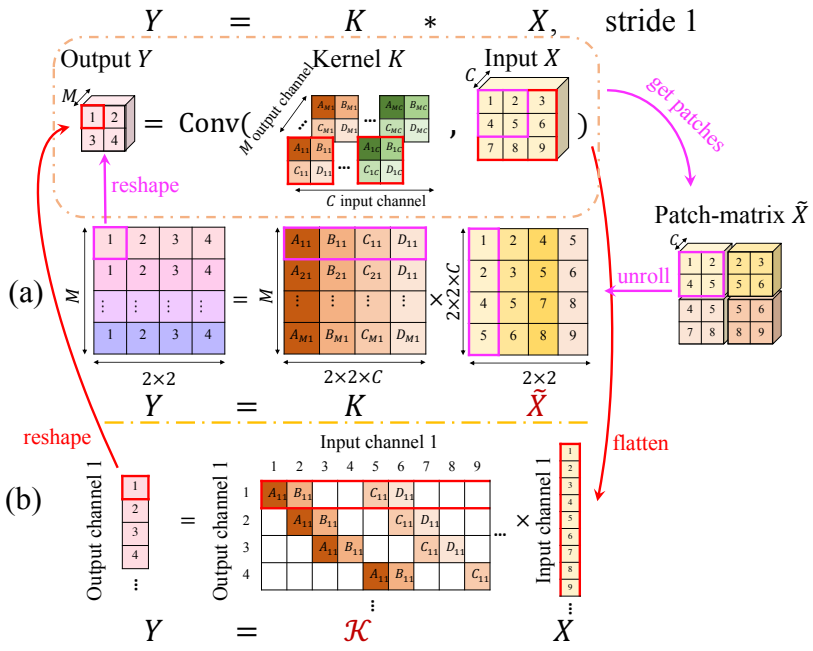


Figure 8.2: Basic idea of our OCNN. A convolutional layer $Y = \text{Conv}(K, X)$ can be formulated as matrix multiplications in two ways: **a)** *im2col* methods [335, 121] retain kernel K and convert input X to patch-matrix $\widetilde{X}$. **b)** We retain input X and convert K to a doubly block-Toeplitz matrix $\mathcal{K}$. With X and Y intact, we directly analyze the transformation from the input to the output. We further propose an efficient algorithm for regularizing $\mathcal{K}$ towards orthogonal convolutions and observe improved feature expressiveness, task performance and uniformity in $\mathcal{K}$'s spectrum (Fig.8.1b).

8.2 Background

***Im2col*-Based Convolutions**. The *im2col* method [335, 121] has been widely used in deep learning as it enables efficient GPU computation. It transforms the convolution into a General Matrix to Matrix Multiplication (GEMM) problem.

Fig.8.2a illustrates the procedure. **a)** Given an input X, we first construct a new input-patch-matrix $\widetilde{X} \in \mathbf{R}^{Ck^2 \times H'W'}$ by copying patches from the input and unrolling them into columns of this intermediate matrix. **b)** The kernel-patch-matrix $K \in \mathbf{R}^{M \times Ck^2}$ can then be constructed by reshaping the original kernel tensor. Here we use the same notation for simplicity. **c)** We can calculate the output $Y = K\widetilde{X}$ where we reshape Y back to the tensor of size $M \times H \times W$ – the desired output of the convolution.

The orthogonal kernel regularization enforces the kernel $K \in \mathbf{R}^{M \times Ck^2}$ to be orthogonal. Specifically, if $M \leq Ck^2$, the row orthogonal regularizer is $L_{\text{korth-row}} = \|KK^T - I\|_F$ where I is the identity matrix. Otherwise, column orthogonal may be achieved by $L_{\text{korth-col}} = \|K^TK - I\|_F$.

Kernel Orthogonality in Neural Networks. Orthogonal kernels help alleviate gradient

vanishing or exploding problems in recurrent neural networks (RNNs) [72, 322, 42, 14, 297, 228]. The effect of soft versus hard orthogonal constraints on the performance of RNNs is discussed in [297]. A cheap orthogonal constraint based on a parameterization from exponential maps is proposed in [42].

Orthogonal kernels are also shown to stabilize the training of CNNs [251] and make more efficient optimizations [25]. Orthogonal weight initialization is proposed in [257, 202]; utilizing the norm-preserving property of orthogonal matrices, it is similar to the effect of batch normalization [136]. However, the orthogonality may not sustain as the training proceeds [257]. To ensure the orthogonality through the whole training, Stiefel manifold-based optimization methods are used in [109, 220, 128] and are further extended to convolutional layers in [220].

Recent works relax and extend the exact orthogonal weights in CNNs. Xie et al. enforce the Gram matrix of the weight matrix to be close to identity under Frobenius norm [329] . Bansal et al. further utilize mutual coherence and the restricted isometry property [25]. Orthogonal regularization has also been observed to help improve the performance of image generation in generative adversarial networks (GANs) [33, 34, 203].

All the aforementioned works adopt kernel orthogonality for convolutions. Sedghi et al. utilize the DBT matrix to analyze singular values of convolutional layers but do not consider orthogonality [260].

Feature Redundancy. Optimized CNNs are known to have significant redundancy between different filters and feature channels [138, 126]. Many works use the redundancy to compress or speed up networks [103, 119, 126]. The highly nonuniform spectra may contribute to the redundancy in CNNs. To overcome the redundancy by improving feature diversity, multi-attention [353], diversity loss [175], and orthogonality regularization [51] have been proposed.

Other Ways to Stabilize CNN Training. To address unstable gradient and co-variate shift problems, various methods have been proposed: Initialize each layer with near-constant variances [94, 114]; Use batch normalization to reduce internal covariate shifts [136]; Reparameterize the weight vectors and decouple their lengths from their directions [254]; Use layer normalization with the mean and variance computed from all of the summed inputs to the neurons [19]; Use a gradient norm clipping strategy to deal with exploding gradients and a soft constraint for vanishing gradients [228].

8.3 Orthogonal Convolution

As we mentioned earlier, convolution can be viewed as an efficient matrix-vector multiplication, where matrix $\mathcal{K}$ is generated by a kernel K. In order to stabilize the spectrum of $\mathcal{K}$, we add convolutional orthogonality regularization to CNNs, which is a stronger condition than kernel orthogonality. First, we discuss the view of convolution as a matrix-vector multiplication in detail. Then, fast algorithms for constraining row and column orthogonality

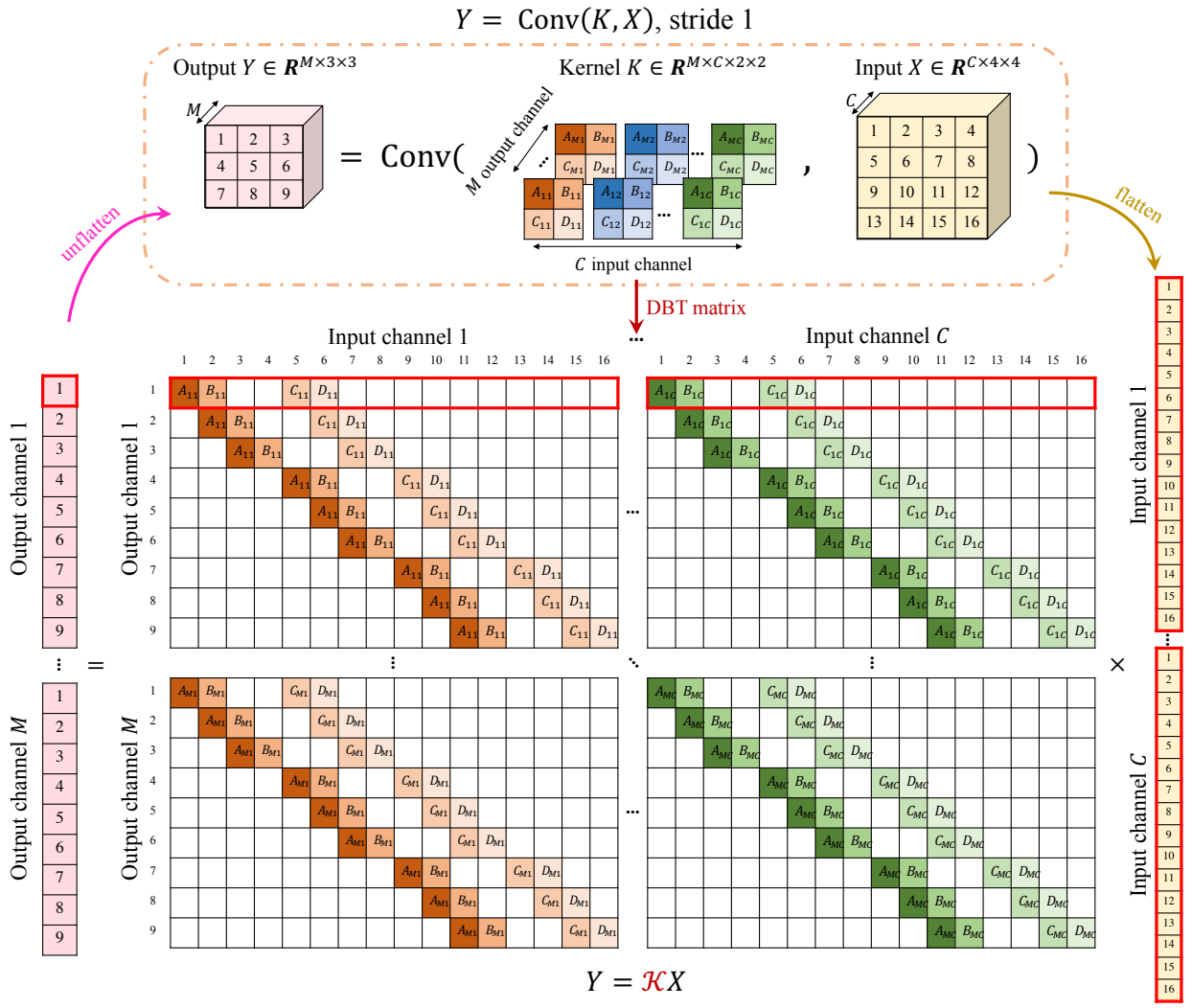


Figure 8.3: Convolution based on the doubly block-Toeplitz (DBT) matrix. We first flatten X to a vector $\mathbf{x}$, and then convert weight tensor $K \in \mathbf{R}^{M\times C\times k\times k}$ as DBT matrix $\mathcal{K} \in \mathbf{R}^{(MH'W')\times(CHW)}$. The output $\mathbf{y} = \mathcal{K}\mathbf{x}$. We can obtain the desired output $Y \in \mathbf{R}^{M\times H'\times W'}$ by reshaping $\mathbf{y}$. The example has input size $C \times 4 \times 4$, kernel size $M \times C \times 2 \times 2$ and stride 1.

in convolutions are proposed. Condition 8.3 summarizes the orthogonality. In this work, we focus on the 2D convolution case, but concepts and conditions generalize to other cases.

8.3.1 Convolution as a Matrix-Vector Multiplication

For a convolutional layer with input tensor $X \in \mathbf{R}^{C\times H\times W}$ and kernel $K \in \mathbf{R}^{M\times C\times k\times k}$, we denote the convolution's output tensor $Y = \text{Conv}(K, X)$, where $Y \in \mathbf{R}^{M\times H'\times W'}$. We can further view K as M different filters, $\{K_i \in \mathbf{R}^{C\times k\times k}\}$. Since convolution is linear, we can

rewrite $\text{Conv}(K, X)$ in a matrix-vector form:

$$Y = \text{Conv}(K, X) \Leftrightarrow \mathbf{y} = \mathcal{K}\mathbf{x} \tag{8.1}$$

where $\mathbf{x}$ is X flattened to a vector. Note that we adopt rigorous notations here while $\mathbf{x}$ and X are not distinguished previously. Each row of $\mathcal{K}$ has non-zero entries corresponding to a particular filter K_i at a particular spatial location. As a result, $\mathcal{K}$ can be constructed as a doubly block-Toeplitz (DBT) matrix $\mathcal{K} \in \mathbf{R}^{(MH'W')\times(CHW)}$ from kernel tensor $K \in \mathbf{R}^{M\times C\times k\times k}$.

We can obtain the output tensor Y by reshaping vector $\mathbf{y}$ back to the tensor form. Fig.8.3 depicts an example of a convolution based on DBT matrix, where we have input size of $C \times 4 \times 4$, kernel size of $M \times C \times 2 \times 2$ and stride 1.

8.3.2 Convolutional Orthogonality

Depending on the configuration of each layer, the corresponding matrix $\mathcal{K} \in \mathbf{R}^{(MH'W')\times(CHW)}$ may be a fat matrix ($MH'W' \leq CHW$) or a tall matrix ($MH'W' > CHW$). In either case, we want to regularize the spectrum of $\mathcal{K}$ to be uniform. In the fat matrix case, the uniform spectrum requires a row orthogonal convolution, while the tall matrix case requires a column orthogonal convolution, where $\mathcal{K}$ is a normalized frame [158] and preserves the norm.

In theory, we can implement the doubly block-Toeplitz matrix $\mathcal{K}$ and enforce the orthogonality condition in a brute force fashion. However, since $\mathcal{K}$ is highly structured and sparse, a much more efficient algorithm exists. In the following, we show the equivalent conditions to the row and column orthogonality, which can be easily computed.

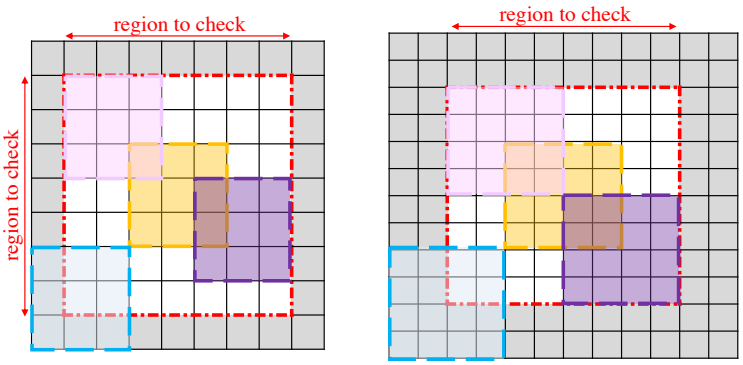
Row Orthogonality. As we mentioned earlier, each row of $\mathcal{K}$ corresponds to a filter K_i at a particular spatial location (h', w') flattened to a vector, denoted as $\mathcal{K}_{ih'w',\bullet} \in \mathbf{R}^{CHW}$. The row orthogonality condition is:

$$\langle \mathcal{K}_{ih'_1w'_1,\bullet}, \mathcal{K}_{jh'_2w'_2,\bullet} \rangle = \begin{cases} 1, (i, h'_1, w'_1) = (j, h'_2, w'_2) \\ 0, \text{otherwise} \end{cases} \tag{8.2}$$

In practice, we do not need to check pairs when the corresponding filter patches do not overlap. It is clear that $\langle \mathcal{K}_{ih'_1w'_1,\bullet}, \mathcal{K}_{jh'_2w'_2,\bullet} \rangle = 0$ if either $|h_1 - h_2| \geq k$ or $|w_1 - w_2| \geq k$, since the two flattened vectors have no support overlap and thus have a zero inner product. Thus, we only need to check Condition 8.2 where $|h_1 - h_2|, |w_1 - w_2| < k$. Due to the spatial symmetry, we can choose fixed h_1, w_1 and only vary i, j, h_2, w_2, where $|h_1 - h_2|, |w_1 - w_2| < k$.

Fig.8.4 shows examples of regions of overlapping filter patches. For a convolution with the kernel size k and the stride S, the region to check orthogonality can be realized by the original convolution with padding $P = \lfloor \frac{k-1}{S} \rfloor \cdot S$. Now we have an equivalent condition to Condition 8.2 as the following self-convolution:

$$\text{Conv}(K, K, \text{padding} = P, \text{stride} = S) = I_{r0} \tag{8.3}$$



(a) kernel size=3, stride=1 (b) kernel size=4, stride=2

Figure 8.4: The spatial region to check for row orthogonality. It is only necessary to check overlapping filter patches for the row orthogonality condition. We show two example cases: stride $S = 1$ with kernel size $k = 3$ and stride $S = 2$ with kernel size $k = 4$. In both examples, the orange patch is the center patch, and the red border is the region of overlapping patches. For example, pink and purple patches fall into the red region and overlap with the center region; blue patches are not fully inside the red region and they do not overlap with the orange ones. We can use padding to obtain the overlapping regions.

where $I_{r0} \in \mathbf{R}^{M\times M\times(2P/S+1)\times(2P/S+1)}$ is a tensor, which has zeros entries except for center $M \times M$ entries as an identity matrix. Minimizing the difference between $Z = \text{Conv}(K, K, \text{padding} = P, \text{stride} = S)$ and I_{r0} gives us a near row-orthogonal convolution in terms of DBT matrix $\mathcal{K}$.

Column Orthogonality. We use tensor $E_{i,h,w} \in \mathbf{R}^{C\times H\times W}$ to denote an input tensor, which has all zeros except a 1 entry at the i^{th} input channel, spatial location (h, w). Denoting $\mathbf{e_{ihw}} \in \mathbf{R}^{CHW}$ as the flattened vector of $E_{i,h,w}$, we can obtain a column $\mathcal{K}_{\bullet,ihw}$ of $\mathcal{K}$ by multiply $\mathcal{K}$ and vector $\mathbf{e_{ihw}}$:

$$\mathcal{K}_{\bullet,ihw} = \mathcal{K}\mathbf{e_{ihw}} = \text{Conv}(K, E_{i,h,w}) \tag{8.4}$$

Here, we slightly abuse the equality notation as the reshaping is easily understood. The column orthogonality condition is:

$$\langle \mathcal{K}_{\bullet,ih_1w_1}, \mathcal{K}_{\bullet,jh_2w_2} \rangle = \begin{cases} 1, (i, h_1, w_1) = (j, h_2, w_2) \\ 0, \text{otherwise} \end{cases} \tag{8.5}$$

Similar to the row orthogonality, since the spatial size of K is only k, Condition 8.5 only needs to be checked in a local region where there is spatial overlap between $\mathcal{K}_{\bullet,ih_1w_1}$ and $\mathcal{K}_{\bullet,jh_2w_2}$. For the stride 1 convolution case, there exists a simpler condition equivalent to Condition 8.5:

$$\text{Conv}(K^T, K^T, \text{padding} = k - 1, \text{stride} = 1) = I_{c0} \tag{8.6}$$

where K^T is the input-output transposed K, i.e. $K^T \in \mathbf{R}^{C\times M\times k\times k}$. $I_{c0} \in \mathbf{R}^{C\times C\times(2k-1)\times(2k-1)}$ has all zeros except for the center $C \times C$ entries as an identity matrix.

Comparison to Kernel Orthogonality. The kernel row- and column-orthogonality conditions can be written in the following convolution form respectively:

$$\begin{cases} \text{Conv}(K, K, \text{padding} = 0) = I_{r0} \\ \text{Conv}(K^T, K^T, \text{padding} = 0) = I_{c0} \end{cases} \tag{8.7}$$

where tensor $I_{r0} \in \mathbf{R}^{M\times M\times 1\times 1}$, $I_{c0} \in \mathbf{R}^{C\times C\times 1\times 1}$ are both equivalent to identity matrices[1].

Obviously, the kernel orthogonality conditions 8.7 are necessary but not sufficient conditions for the orthogonal convolution conditions 8.3,8.6 in general. For the special case when convolution stride is k, they are equivalent.

Row-Column Orthogonality Equivalence. The lemma below unifies the row orthogonality condition 8.2 and column orthogonality condition 8.5. This lemma [166] gives a uniform convolution orthogonality independent of the actual shape of $\mathcal{K}$ and provides a unique regularization: $\min_K L_{\text{orth}} = \|Z - I_{r0}\|_F^2$, which only depends on Condition 8.3.

Lemma 1. *The row orthogonality and column orthogonality are equivalent in the MSE sense, i.e.* $\|\mathcal{K}\mathcal{K}^T - I\|_F^2 = \|\mathcal{K}^T\mathcal{K} - I'\|_F^2 + U$, *where* U *is a constant.*

We leave the proof to Section B.4 of supplementary materials.

Orthogonal Regularization in CNNs. We add an additional soft orthogonal convolution regularization loss to the final loss of CNNs, so that the task objective and orthogonality regularization can be simultaneously achieved. Denoting $\lambda > 0$ as the weight of the orthogonal regularization loss, the final loss is:

$$L = L_{\text{task}} + \lambda L_{\text{orth}} \tag{8.8}$$

where L_{task} is the task loss, e.g. softmax loss for image classification, and L_{orth} is the orthogonal regularization loss.

8.4 Experiments

We conduct 3 sets of experiments to evaluate OCNNs. The first set benchmarks our approach on image classification datasets CIFAR100 and ImageNet. The second set benchmarks the performance under semi-supervised settings and focuses on qualities of learned features. For high-level visual feature qualities, we experiment on the fine-grained bird image retrieval. For low-level visual features, we experiment on unsupervised image inpainting. Additionally, we compare visual feature qualities in image generation tasks. The third set of experiments focuses on the robustness of OCNN under adversarial attacks. We analyze OCNNs in terms of DBT matrix $\mathcal{K}$'s spectrum, feature similarity, hyperparameter tuning, and space/time complexity.

[1]Since there is only 1 spatial location.

8.4.1 Classification on CIFAR100

The key novelty of our approach is the orthogonal regularization term on convolutional layers. We compare both conv-orthogonal and kernel-orthogonal regularizers on CIFAR-100 [160] and evaluate the image classification performance using ResNet [113] and WideResNet [344] as backbone networks. The kernel-orthogonality and our conv-orthogonality are added as additional regularization terms, without modifying the network architecture. Hence, the number of parameters of the network does not change.

ResNet and Row Orthogonality. Though we have derived a unified orthogonal convolution regularizer, we benchmark its effectiveness with two different settings. Convolutional layers in ResNet [113] usually preserve or reduce the dimension from input to output, *i.e.* a DBT matrix $\mathcal{K}$ would be a square or fat matrix. In this case, our regularizer leads to the row orthogonality condition. Table 8.2 shows top-1 classification accuracies on CIFAR100. Our approach achieves 78.1%, 78.7%, and 79.5% image classification accuracies with ResNet18, ResNet34 and ResNet50, respectively. For 3 backbone models, OCNNs outperform plain baselines by 3%, 2%, and 1%, as well as kernel orthogonal regularizers by 2%, 1%, and 1%.

WideResNet and Column Orthogonality. Unlike ResNet, WideResNet [344] has more channels and some tall DBT matrices $\mathcal{K}$. When the corresponding DBT matrix $\mathcal{K}$ of a convolutional layer increases dimensionality from the input to the output, our OCNN leads to the column orthogonality condition. Table 8.3 reports the performance of column orthogonal regularizers with backbone model of WideResNet28 on CIFAR100. Our OCNNs achieve 3% and 1% gain over plain baselines and kernel orthogonal regularizers.

Table 8.2: Top-1 accuracies on CIFAR100. Our OCNN outperforms baselines and the SOTA orthogonal regularizations.

	ResNet18	ResNet34	ResNet50
baseline [113]	75.3	76.7	78.5
kernel orthogonality [329]	76.5	77.5	78.8
OCNN (ours)	**78.1**	**78.7**	**79.5**

Table 8.3: WideResNet [344] performance. We observe improved performance of OCNNs.

	WideResNet [344]	Kernel orth [329]	OCNN
Acc.	77.0	79.3	**80.1**

8.4.2 Classification on ImageNet

We add conv-orthogonal regularizers to the backbone model ResNet34 on ImageNet [67], and compare OCNNs with state-of-the-art orthogonal regularization methods.

Experimental Settings. We follow the standard training and evaluation protocols of ResNet34. In particular, the total epoch of the training is 90. We start the learning rate at 0.1, decreasing by 0.1 every 30 epochs and weight decay 1e-4. The weight λ of the regularization loss is 0.01, the model is trained using SGD with momentum 0.9, and the batch size is 256.

Comparisons. Our method is compared with hard orthogonality OMDSM [128], kernel orthogonality [329] and spectral restricted isometry property regularization [25]. Table 8.4 shows the Top-1 and Top-5 accuracies on ImageNet. Without additional modification to the backbone model, OCNN achieves 25.87% top-5 and 7.89% top-1 error. The proposed method outperforms the plain baseline, as well as other orthogonal regularizations by 1%.

Table 8.4: Top-1 and Top-5 errors on ImageNet [67] with ResNet34 [113]. Our conv-orthogonal regularization outperforms baselines and SOTA orthogonal regularizations.

	Top-1 error	Top-5 error
ResNet34 (baseline) [113]	26.70	8.58
OMDSM [128]	26.88	8.89
kernel orthogonality [329]	26.68	8.43
SRIP [25]	26.10	8.32
OCNN (ours)	**25.87**	**7.89**

8.4.3 Semi-Supervised Learning

A general regularizer should provide benefit to a variety of tasks. A common scenario that benefits from regularization is semi-supervised learning, where we have a large amount of data with limited labels. We randomly sample a subset of CIFAR100 as labeled and treat the rest as unlabeled. The orthogonal regularization is added to the baseline model ResNet18 without any additional modifications. The classification performance is evaluated on the entire validation set for all different labeled subsets.

We compare OCNN with kernel-orthogonal regularization while varying the proportion of labeled data from 10% to 80% of the entire dataset (Table 8.5). OCNN constantly outperforms the baseline by 2% - 3% under different fractions of labeled data.

8.4.4 Fine-grained Image Retrieval

We conduct fine-grained image retrieval experiments on CUB-200 bird dataset [320] to understand high-level visual feature qualities of OCNNs. Specifically, we directly use the

Table 8.5: Top-1 accuracies on CIFAR100 with different fractions of labeled data. OCNNs are consistently better.

% of training data	10%	20%	40%	60%	80%	100%
ResNet18 [113]	31.2	47.9	60.9	66.6	69.1	75.3
kernel orthogonality [329]	33.7	50.5	63.0	68.8	70.9	76.5
Conv-orthogonality	**34.5**	**51.0**	**63.5**	**69.2**	**71.5**	**78.1**
Our gain	3.3	3.1	2.6	2.6	2.4	2.8

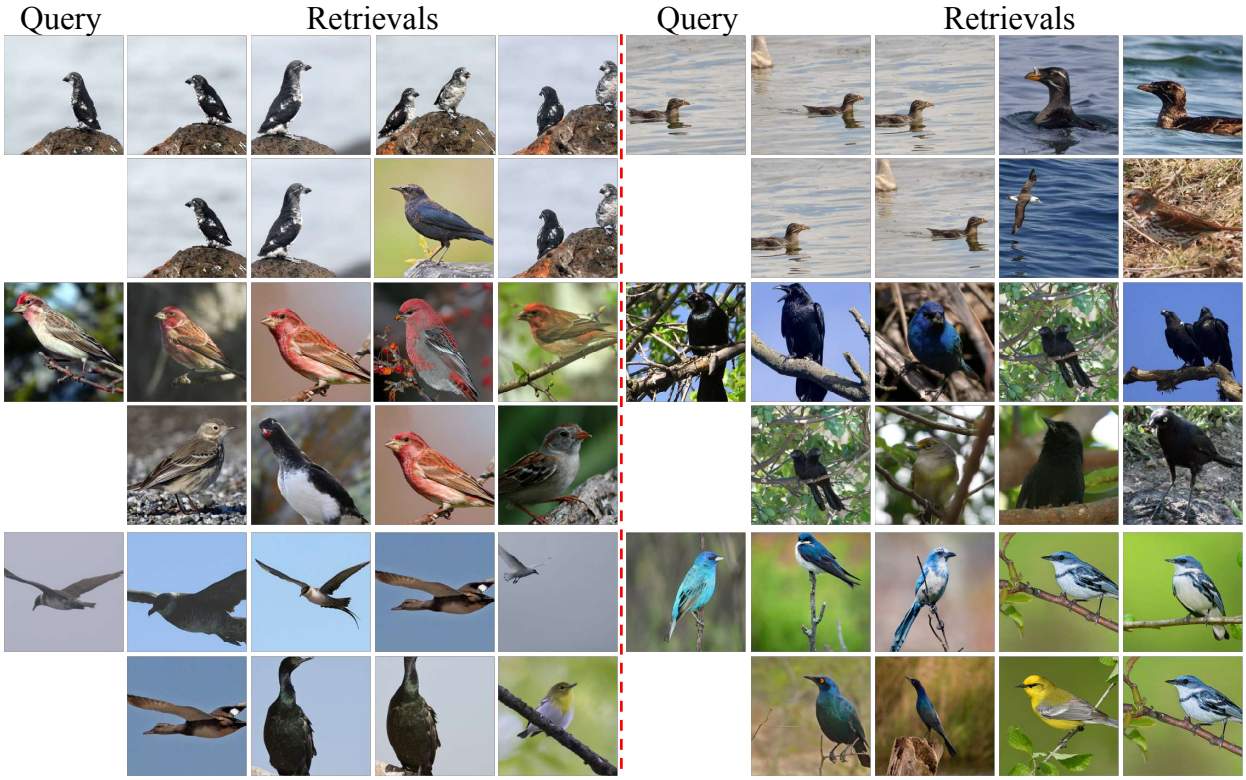


Figure 8.5: Image retrieval results on CUB-200 Birds Dataset. The model (ResNet34) is trained on ImageNet only. First row shows our OCNN results, while the second row shows the baseline model results. Ours achieves 2% and 3% top-1 and top-5 k-nearest neighbor classification gain.

ResNet34 model trained on ImageNet (from Section 8.4.2) to obtain features of images in CUB-200, without further training on the dataset. We observed improved results with OCNNs (Fig.8.5). With conv-orthogonal regularizers, the top-1 k-nearest-neighbor classification accuracy improves from 25.1% to 27.0%, and top-5 k-nearest-neighbor classification accuracy improves from 39.4% to 42.3%.

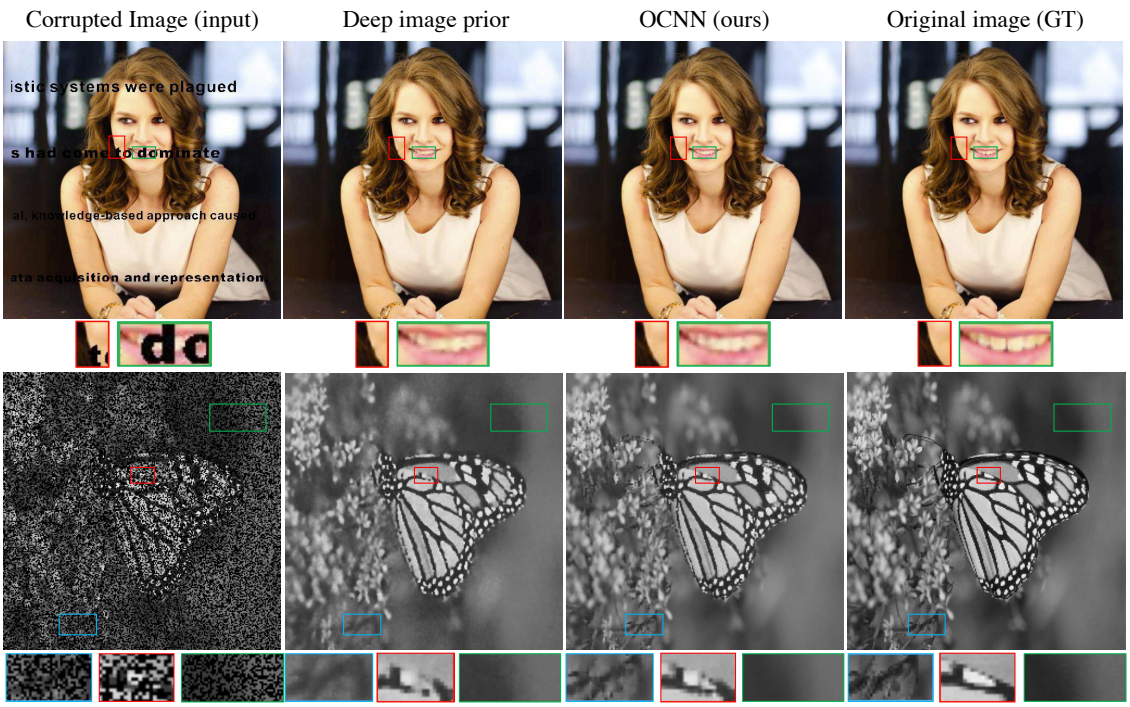


Figure 8.6: Image inpainting results compared with deep image prior [293]. Top – comparison on text inpainting example. Bottom – comparison on inpainting 50% of missing pixels. In both cases, our approach outperforms previous methods.

8.4.5 Unsupervised Image Inpainting

Table 8.6: Quantitative comparisons on the standard inpainting dataset [120]. Our conv-orthogonality outperforms the SOTA methods.

	Barbara	Boat	House	Lena	Peppers	C.man	Couple	Finger	Hill	Man	Montage
Convolutional dictionary learning [226]	28.14	31.44	34.58	35.04	31.11	27.90	31.18	31.34	32.35	31.92	28.05
Deep image prior (DIP) [293]	32.22	33.06	39.16	36.16	33.05	29.8	32.52	32.84	32.77	32.20	34.54
DIP + kernel orthogonality [329]	34.88	34.93	38.53	37.66	34.58	33.18	33.71	34.40	35.98	32.93	36.99
DIP + conv-orthogonality (ours)	**38.12**	**35.15**	**41.73**	**39.76**	**37.75**	**38.21**	**35.88**	**36.87**	**39.89**	**33.57**	**38.48**

To further assess the generalization capacity of OCNNs, we add the regularization term to the new task of unsupervised inpainting. In image inpainting, one is given an image X_0 with missing pixels in correspondence of a binary mask $M \in \{0,1\}^{C\times H\times W}$ of the same size of the image. The goal is to reconstruct the original image X by recovering missing pixels:

$$\min E(X; X_0) = \min \|(X - X_0) \odot M\|_F^2 \tag{8.9}$$

Deep image prior (DIP) [293] proposed to use the prior implicitly captured by the choice of a particular generator network f_θ with parameter θ. Specifically, given a code vector/ tensor $\mathbf{z}$, DIP used CNNs as a parameterization $X = f_\theta(\mathbf{z})$. The reconstruction goal in Eqn.8.9 can be written as:

$$\min_\theta \|(f_\theta(\mathbf{z}) - X_0) \odot M\|_F^2 \tag{8.10}$$

The network can be optimized without training data to recover X. We further add our conv-orthogonal regularization as an additional prior to the reconstruction goal, to validate if the proposed regularization helps the inpainting:

$$\min_\theta \|(f_\theta(\mathbf{z}) - X_0) \odot M\|_F^2 + \lambda L_{\text{orth}}(\theta) \tag{8.11}$$

In the first example (Fig.8.6, top), the inpainting is used to remove text overlaid on an image. Compared with DIP [293], our orthogonal regularization leads to improved reconstruction result of details, especially for the smoothed face outline and finer teeth reconstruction.

The second example (Fig.8.6, bottom) considers inpainting with masks randomly sampled according to a binary Bernoulli distribution. Following the procedure in [226, 293], we sample a mask to randomly drop 50% of pixels. For a fair comparison, all the methods adopt the same mask. We observe improved background quality, as well as finer reconstruction of the texture of butterfly wings.

We report quantitative PSNR comparisons on the standard data set [121] in Table 8.6. OCNN outperforms previous state-of-the-art DIP [293] and convolutional sparse coding [226]. We also observe performance gains compared to kernel orthogonal regularizations.

8.4.6 Image Generation

Orthogonal regularizers have shown great success in improving the stability and performance of GANs [34, 203, 33]. We analyze the influence of convolutional orthogonal regularizers on GANs with the best architecture reported in [95]. Training takes 320 epochs with OCNN regularizer applied to both the generator and discriminator. The regularizer loss λ is set to 0.01 while other settings are retained as default.

The reported model is evaluated 5 times with 50k images each. We achieve an inception score (IS) of 8.63 ± 0.007 and Fréchet inception distance (FID) of 11.75 ± 0.04 (Table 8.7), outperforming the baseline and achieving the state-of-the-art performance. Additionally, we observe faster convergence of GANs with our regularizer (Fig.8.7).

8.4.7 Robustness under Attack

The uniform spectrum of $\mathcal{K}$ makes each convolutional layer approximately a 1-Lipschitz function. Given a small perturbation to the input, Δx, the change of the output Δy is

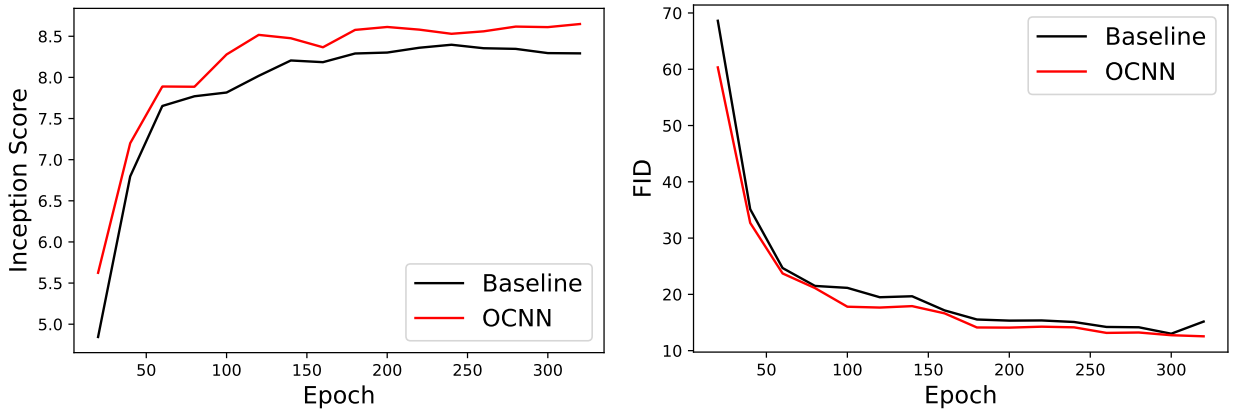


Figure 8.7: OCNNs have faster convergence for GANs. For IS (left) and FID (right), OCNNs consistently outperforms CNNs [95] at every epoch.

Table 8.7: Inception Score and Fréchet Inception Distance comparison on CIFAR10. Our OCNN outperforms the baseline [95] by 0.3 IS and 1.3 FID.

	IS	FID
PixelCNN [294]	4.60	65.93
PixelIQN [219]	5.29	49.46
EBM [74]	6.78	38.20
SNGAN [203]	8.22	21.70
BigGAN [33]	**9.22**	14.73
AutoGAN [95]	8.32	13.01
OCNN (ours)	8.63	**11.75**

Table 8.8: Attack time and number of necessary attack queries needed for 90% successful attack rate

	Attack time/s	# necessary attack queries
ResNet18 [113]	19.3	27k
OCNN (ours)	136.7	46k

bounded to be low. Therefore, the model enjoys robustness under attack. Our experiments demonstrate that it is much harder to search for adversarial examples.

We adopt the simple black box attack [99] to evaluate the robustness of baseline and OCNN with ResNet18 [113] backbone architecture trained on CIFAR100. The attack samples around the input image and finds a "direction" to rapidly decrease the classification

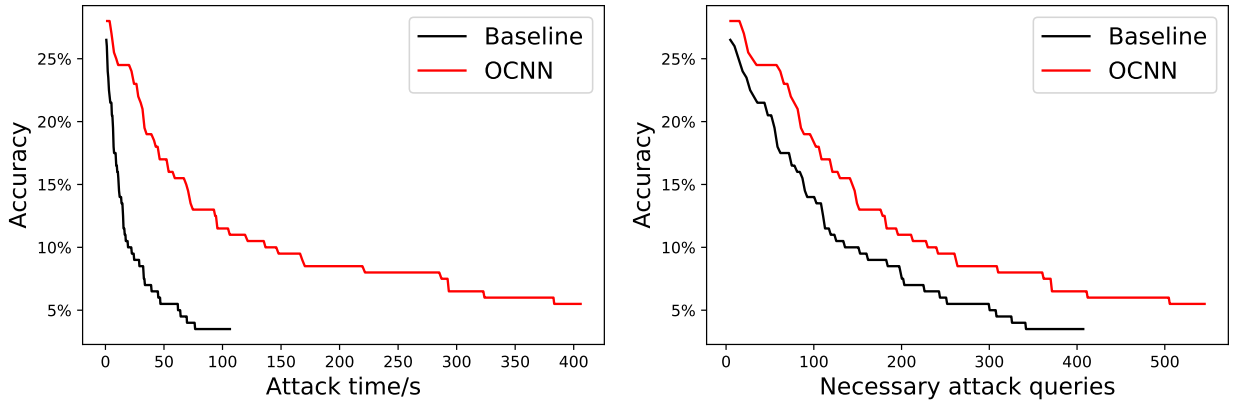


Figure 8.8: Model accuracy v.s. attack time and necessary attack queries. With our conv-orthogonal regularizer, it takes 7x time and 1.7x necessary attack queries to achieve 90% successful attack rate. Note that baseline ends at accuracy 3.5% while ours ends at 5.5% with the same iteration.

confidence of the network by manipulating the input. We only evaluate on the correctly classified test images. The maximum iteration is 10,000 with pixel attack. All other settings are retained. We report the attack time and number of necessary attack queries for a specific attack successful rate.

It takes approximately 7x time and 1.7x attack queries to attack OCNN, compared with the baseline (Table 8.8 and Fig.8.8). Additionally, after the same iterations of the attack, our model outperforms the baseline by 2%.

To achieve the same attack rate, baseline models need more necessary attack queries, and searching for such queries is nontrivial and time consuming. This may account for the longer attack time of the OCNN.

8.4.8 Analysis

To understand how the conv-orthogonal regularization help improve the performance of CNNs, we analyze several aspects of OCNNs. First, we analyze the spectrum of the DBT matrix $\mathcal{K}$ to understand how it helps relieve gradient vanishing/exploding. We then analyze the filter similarity of networks over different layers, followed by the influence of the weight λ of the regularization term. Finally, we analyze the time and space complexity of OCNNs.
Spectrum of the DBT Kernel Matrix $\mathcal{K}$. For a convolution $Y = \mathcal{K}X$, we analyze the spectrum of $\mathcal{K}$ to understand the properties of the convolution. We analyze the spectrum of $K \in \mathbf{R}^{64\times128\times3\times3}$ of the first convolutional layer of the third convolutional block of ResNet18 network trained on CIFAR100. For fast computation, we use input of size $64 \times 16 \times 16$, and solve all the singular values of $\mathcal{K}$.

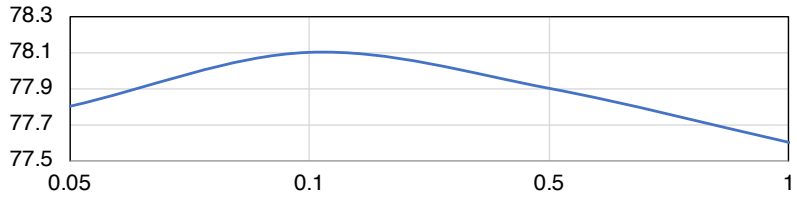


Figure 8.9: CIFAR100 classification accuracy (%) with different weight λ of the regularization loss. With backbone model ResNet18, we achieve the highest performance at $\lambda = 0.1$.

As in Fig.5.1(b), the spectrum of plain models vanishes rapidly, and may cause gradient vanishing problems. Kernel orthogonality helps the spectrum decrease slower. With our conv-orthogonal regularization, the spectrum almost always stays at 1. The uniform spectrum perserves norm and information between convolutional layers.

Filter Similarity. Orthogonality makes off-diagonal elements become 0. This means that for any two channels of a convolutional layer, correlations should be relatively small. This can reduce the filter similarity and feature redundancy across different channels.

We use guided back-propagation patterns [270] on images from the validation set of the ImageNet [67] dataset to investigate filter similarities. Guided back-propagation patterns visualize the gradient of a particular neuron with respect to the input image $X \in \mathbf{R}^{C\times H\times W}$. Specifically for a layer of M channels, the combination of flattened guided back-propagation patterns is denoted as $G \in \mathbf{R}^{M\times CWH}$. The correlation matrix $\text{corr}(G)$ over different channels of this layer is $(\text{diag}(K_{GG}))^{-\frac{1}{2}} K_{GG} (\text{diag}(K_{GG}))^{-\frac{1}{2}}$, where $K_{GG} = \frac{1}{M}[(G - \mathbb{E}[G])(G - \mathbb{E}[G])^T]$ is the covariance matrix. We plot the histogram of off-diagonal elements of $\text{corr}(G)$ of all validation images.

Fig.5.1(a) depicts the normalized histogram of pairwise filter similarities of plain ResNet34. As the number of channels increases with depth from 128 to 512, the curve shifts right and becomes far narrower, i.e., more filters become similar. Fig.5.1(c) depicts the histogram of filter similarities at layer 27 of ResNet34 with different regularizers. OCNNs make the curve shift left and become wider, indicating it can enhance filter diversity and decrease feature redundancy.

Hyper-Parameter Analysis. We analyze the influence of the weight λ of the orthogonality loss. As discussed earlier, we achieve the "soft orthogonality" by adding additional loss with weight λ to the network. Fig.8.9 plots the image classification performance of CIFAR100 with backbone model ResNet18 under λ ranging from 0.05 to 1.0. Our approach achieves the highest accuracy when $\lambda = 0.1$.

Space and Time Complexity. We analyze the space and time complexity in Table 8.9. The ResNet34 [113] backbone model is tested on ImageNet [67] with a single NVIDIA GeForce GTX 1080 Ti GPU and batch size 256.

The number of parameters and the test time of the CNN do not change since the regularizer is an additional loss term only used during training. With kernel orthogonal regularizers,

the training time increases 3%; with conv-orthogonal regularizers, the training time increases 9%.

Table 8.9: Model size and training/ test time on ImageNet [67].

	ResNet34 [113]	kernel-orth [329]	OCNN
# Params.	21.8M	same	same
Training time (min/epoch)	49.5	51.0	54.1
Test time (min/epoch)	1.5	same	same

8.5 Summary

We develop an efficient OCNN approach to impose a filter orthogonality condition on a convolutional layer based on the doubly block-Toeplitz matrix representation of the convolutional kernel, as opposed to the commonly adopted kernel orthogonality approaches. We show that kernel orthogonality [25, 128] is necessary but not sufficient for ensuring orthogonal convolutions.

Our OCNN requires no additional parameters and little computational overhead, consistently outperforming the state-of-the-art alternatives on a wide range of tasks such as image classification and inpainting under supervised, semi-supervised and unsupervised settings. It learns more diverse and expressive features with better training stability, robustness, and generalization.

Chapter 9

Recurrent Parameter Generator

9.1 Introduction

Deep neural networks as general optimization tools have achieved great success with increasingly more training data, deeper and larger neural networks: A recently developed NLP model, GPT-3 [36], has astonishing 175 billion parameters! While the model performance generally scales with the number of parameters [122], with parameters outnumbering training data, the model is significantly over-parameterized.

Many approaches have been proposed to remove redundancy in trained large models: neural network pruning [167, 102, 184], efficient network design spaces [126, 135, 255], parameter regularization [299, 304, 271, 216], model quantization [130, 245, 188], neural architecture search [359, 39, 298], recurrent models [21, 22, 318], multi-task feature encoding [243, 108], etc. Pruning-based model compression dates back to the late 80s [207, 167] and has enjoyed recent resurgence [102, 30]. They remove unimportant parameters from a pre-trained model and can achieve significant model compression.

Our work is a departure from mainstream approaches towards model optimization and parameter reduction: rather than compressing a large model, we directly optimize a lean model with a small set of free parameters (number of free parameters equal to degree of freedom of the model, or DoF), which can be linearly unpacked to a large model. Training the large model can be viewed as solving a neural optimization with a set of predefined linear constraints. One benefit of constrained neural optimization we observe is that it leads to a faster convergence rate (Section 9.5.6). Specifically, we define different layers in a neural network based on a fixed amount of DoF, which we call *recurrent parameter generator* (RPG). That is, we differentiate the number of model parameters and DoF. Traditionally, model parameters are treated independently of each other; the total number of parameters equals DoF. However, by tapping into how a core set of free parameters can be assigned to the neural network model, we can develop a large model of many parameters, which are linearly constrained by the small set of free parameters.

There is excess capacity in neural networks independent of how and where the parameters

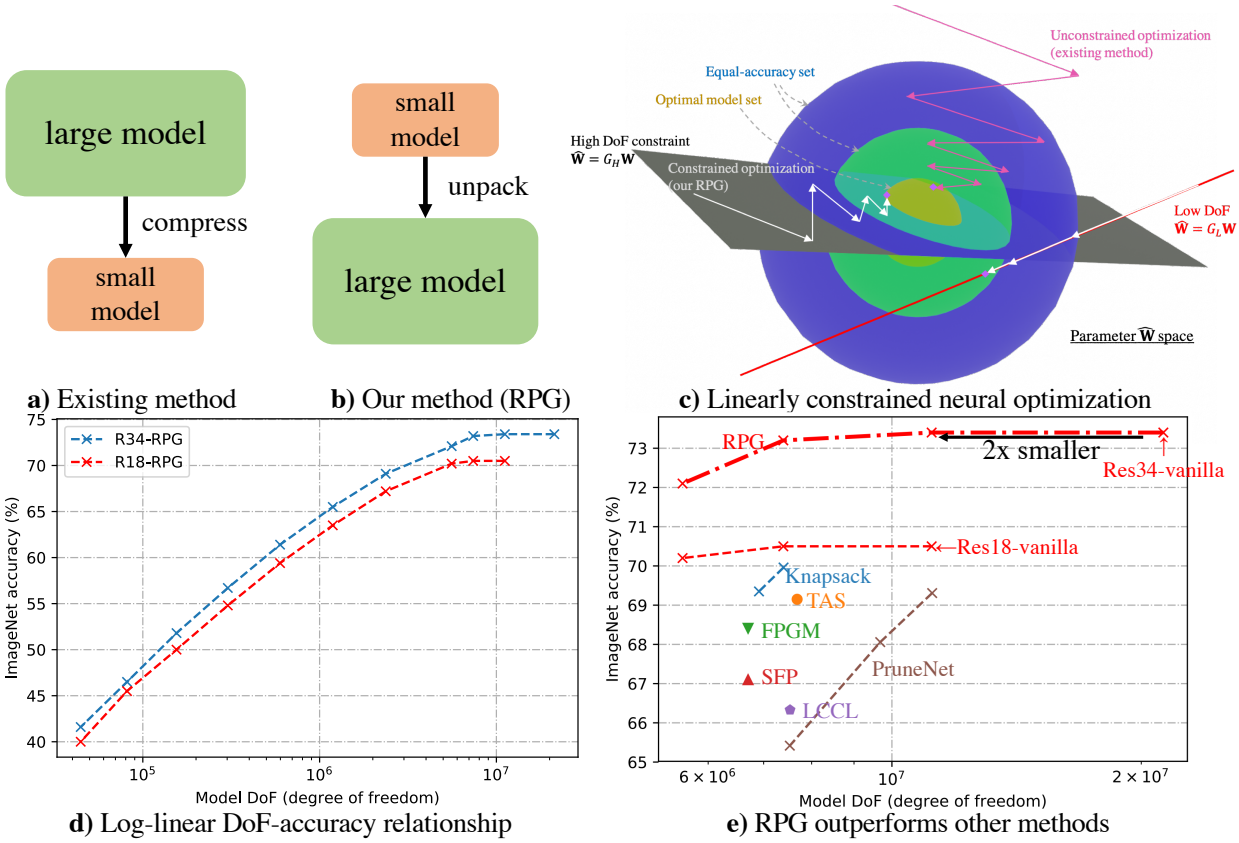


Figure 9.1: We propose a novel approach to compact and optimal deep learning by decoupling model DoF and model parameters. **a**) Existing methods first finds the optimal in a large model space and then compress it for practical deployment. **b**) We propose to start with a small (DoF) model of free parameters, use recurrent parameter generator (RPG) to unpack them onto a large model with predefined random linear projections. **c**) Gradient descent finds the optimal model of a small DoF under these linear constraints with faster converge than training the large unpacked model itself (Fig.5.5**b**). If the DoF is too small, the optimal large model may fall out of the constrained subpsace. However, at a sufficiently large DoF, RPG gets rid of redundancy and often finds a model with little loss in accuracy. **d**) RPG reveals a log-linear relationship between model DoF and accuracy. **e**) RPG achieves the same ImageNet accuracy with half of the ResNet-vanilla DoF. RPG also outperforms other state-of-the-art compression approaches.

are used in the network, even at the level of individual scalar values. Surprisingly, backpropagation training of a deep network is able to cope with that the same parameter can be assigned to multiple random locations in the network without significantly impacting model performance. Our extensive experiments show that a large neural network does not need to be over-parameterized to achieve competitive performance. Particularly, a ResNet18 can be implemented with DoF equivalent to one convolution layer in a ResNet18-vanilla (4.72×

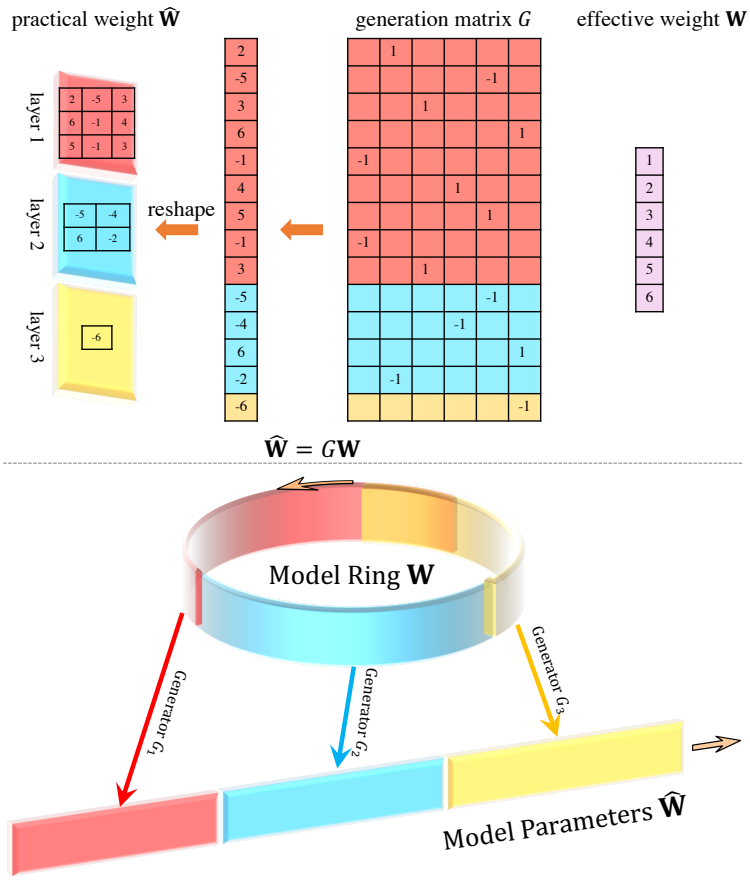


Figure 9.2: **Upper:** Networks are optimized with a linear constraint $\hat{\mathbf{W}} = \boldsymbol{G}\mathbf{W}$, where the constrained parameter $\hat{\mathbf{W}}$ of each network layer was generated by the generating matrix $\boldsymbol{G}$ from the free parameter $\mathbf{W}$, which is directly optimized. $\hat{\mathbf{W}}$ is unpacked large model parameter while the size of $\mathbf{W}$ is the model DoF. **Lower:** This paper discusses a specific format of parameter generation, recurrent parameter generator (RPG). RPG shares a fixed set of parameters in a ring and uses them to generate parameters of different parts of a neural network, whereas in the standard neural network, all the parameters are independent of each other, so the model gets bigger as it gets deeper. The third section of the model starts to overlap with the first section in the model ring, and all later layers share generating parameters for possibly multiple times.

DoF reduction) and still achieves 67.2% ImageNet top-1 accuracy. The proposed method is also extremely flexible in reducing model DoF. In some sense, the proposed RPG method can be viewed as an automatic model DoF reduction technique, which explores the optimal accuracy-parameter trade-off. When we reduce the model DoF, RPG demonstrates graceful performance degradation, and its compression results are frequently on par with the SOTA pruning methods besides the flexibility. Even if we reduce the Res18 backbone DoF to 36K, which is about 300× reduction, ResNet18 can still achieve 40% ImageNet top-1 accuracy.

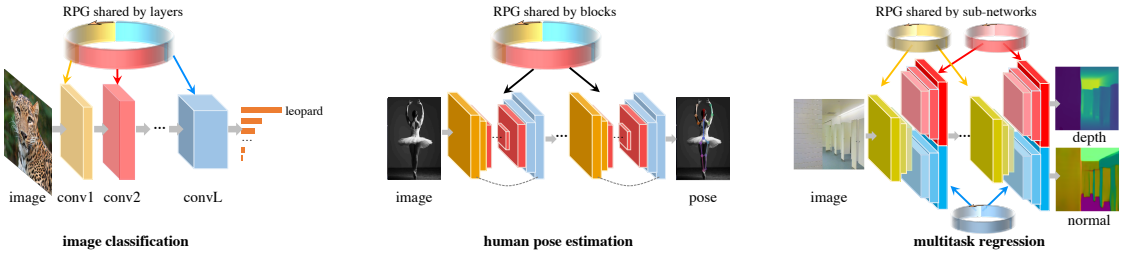


Figure 9.3: We demonstrate the effectiveness of RPG on various applications including image classification (**Left**), human pose estimation (**Middle**), and multitask regression (**Right**). RPGs are shared at multiple scales: a network can either have a global RPG or multiple local RPGs that are shared within blocks or sub-networks.

Further, we show RPG can be quantized and pruned to improve FLOPs and runtime with relatively mild accuracy drops.

To summarize, we make three contributions: **1)** We provide a new perspective towards automatic model size reduction: we define a neural network with certain DoF with random linear constraints. We discover that gradient descent can automatically solve constrained optimization for the best model with a faster convergence rate. This constrained neural optimization perspective is likely to benefit many other applications. **2)** We propose the recurrent parameter generator (RPG), which decouples the network architecture and the network DoF. We can flexibly choose any desired DoF to construct the network given a specific neural network architecture. **3)** By separating network architectures from parameters, RPG becomes a tool to understand the relationship between the model DoF and the network performance. We observe an empirical log-linear DoF-Accuracy relationship.

9.2 Background

Many works study model DoF reduction or compression. We discuss each one and its relationship to our work.

Model Pruning, Neural Architecture Search, and Quantization. Model pruning seeks to remove unimportant parameters in a trained model. Recently, it's proposed to use neural architecture search as coarse-grained model pruning [342, 70]. Another related effort is network quantization [130, 245, 188], which seeks to reduce the bits used for each parameter and can frequently reduce the model size by 4× with minimal accuracy drop. More recently, [69] presents a framework for analyzing model scaling strategies that consider network properties such as FLOPs and activations.

Parameter Regularization and Priors. Regularization has been widely used to reduce model redundancy [163, 216], alleviate overfitting [271, 299], and ensure desired mathematical regularity [304]. RPG can be viewed as a parameter regularization in the sense that weight sharing poses many equality constraints to weights and regularizes weights to a low-

dimensional space. HyperNeat [274] and CPPNs [273] use networks to determine the weight between two neurons as a function of their positions. [146, 145] introduced a similar idea by providing a hierarchical prior for network parameters.

Recurrent Networks and Deep Equilibrium Models. Recurrence and feedback have been shown in psychology and neuroscience to act as modulators or competitive inhibitors to aid feature grouping [93], figure-ground segregation [131] and object recognition [328]. Recurrence-inspired mechanisms also achieve success in feed-forward models. There are two main types of employing recurrence based on if weights are shared across recurrent modules. ResNet [113], a representative of reusing similar structures without weight sharing, introduces parallel residual connections and achieves better performance by going deeper in networks. Similarly, some works [280, 272] also suggest iteratively injecting thus-far representations to the feed-forward network useful. Stacked inference methods [242, 324, 319] are also related while they consider each output in isolation. Some find sharing weights across recurrent modules valuable. They demonstrate applications in temporal modelling [319, 331, 147], spatial attention [204, 37], pose estimation [318, 41], and so on [173, 345]. Such methods usually shine in modeling long-term dependencies. In this work, we recurrently share weights across different layers of a feedback network to reduce network redundancy.

Given stacking weight-shared modules improve the performance, researchers consider running even infinite depth of such modules by making the sequential modules converge to a fixed point [168, 21]. Employing such *equilibrium* models to existing networks, they show improved performance in many natural language processing [21] and computer vision tasks [22, 315]. One issue with deep equilibrium models is that the forward and backward propagation usually takes much more iterations than explicit feed-forward networks. Some work [86] improves the efficiency by making the backward propagation Jacobian free. Another issue is that *infinite* depth and fixed point may not be necessary or even too strict for some tasks. Instead of achieving infinite depth, our model shares parameters to a certain level. We empirically compare with equilibrium models in Section 9.5.

Efficient Network Space and Matrix Factorization. Convolution is an efficient and structured matrix-vector multiplication. Arguably, the most fundamental idea in building efficient linear systems is matrix factorization. Given the redundancy in deep convolutional neural network parameters, one can leverage the matrix factorization concept, e.g., factorized convolutions, and design more efficient network classes [126, 135, 281, 255].

9.3 Recurrent Parameter Generator

Linearly Constrained Neural Optimization. Consider optimizing a network with input data $\mathbf{X}$, parameters $\hat{\mathbf{W}}$ and loss function L. The optimization can be written as:

$$\min L(\mathbf{X}; \hat{\mathbf{W}}) \quad \text{s.t. } \hat{\mathbf{W}} = \boldsymbol{G}\mathbf{W} (\text{or equally } \boldsymbol{R}\hat{\mathbf{W}} = 0) \tag{9.1}$$

where $\hat{\mathbf{W}} = \boldsymbol{G}\mathbf{W}$ refers to a set of linear constraints, where $\boldsymbol{G} \in \Re^{N \times M}$ is a full-rank tall matrix (i.e. $N \geq M$). Here we refer to $\hat{\mathbf{W}}$ as the constrained parameters and $\mathbf{W}$ as the

free parameters. This constraint is a change of variable, i.e., the constrained parameter $\hat{\mathbf{W}}$ is linearly generated from the free parameter $\mathbf{W}$ by generating matrix $\boldsymbol{G}$. We can consider $\mathbf{W}$ as a compressed model, which is unpacked into $\hat{\mathbf{W}}$ to construct the large neural network. $\mathbf{W}$ is directly optimized via gradient descent and free to update. In this linearly constrained neural optimization, the model DoF is equivalent to M, which is the dimension of $\mathbf{W}$. An equivalent form of the constraint $\hat{\mathbf{W}} = \boldsymbol{G}\mathbf{W}$ is $\boldsymbol{R}\hat{\mathbf{W}} = 0$, where $\boldsymbol{R} \in \Re^{(N-M)\times N}$ can be derived from SVD of $\boldsymbol{G}$.

Recurrent Parameter Generator. Let's assume that we construct a deep convolutional neural network containing L different convolution layers. Let $\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_L$ be the corresponding L convolutional kernels[1]. Rather than using separate sets of parameters for different convolution layers, we create a single set of parameters $\mathbf{W} \in \Re^M$ and use it to generate the corresponding parameters $\hat{\mathbf{W}} = \left[\mathbf{K}_1^T, \mathbf{K}_2^T, \ldots, \mathbf{K}_L^T\right]^T \in \Re^N$ for each convolution layer:

$$\mathbf{K}_i = \boldsymbol{G}_i \cdot \mathbf{W}, i \in \{1, \ldots, L\} \tag{9.2}$$

where $\boldsymbol{G}_i$ is a fixed predefined generating matrix, which is used to generate $\mathbf{K}_i$ from $\mathbf{W}$. We call $\boldsymbol{G} = \left[\boldsymbol{G}_1^T, \ldots, \boldsymbol{G}_L^T\right]^T$ and $\mathbf{W}$ the *recurrent parameter generator* (RPG). In this work, we always assume that the size of $\mathbf{W}$ is not larger than the total parameters of the model, i.e., $|\mathbf{W}| \leq \sum_i |\mathbf{K}_i|$. This means an element of $\mathbf{W}$ will generally be used in more than one layer of a neural network. Additionally, the gradient of $\mathbf{W}$ is a linear superposition of the gradients from each convolution layer. During the neural network training, let's assume convolution kernel $\mathbf{K}_i$ receives gradient $\frac{\partial \ell}{\partial \mathbf{K}_i}$, where ℓ is the loss function. Based on the chain rule, it is clear that the gradient of $\mathbf{W}$ is:

$$\frac{\partial \ell}{\partial \mathbf{W}} = \sum_{i=1}^{L} \boldsymbol{G}_i^T \cdot \frac{\partial \ell}{\partial \mathbf{K}_i} \tag{9.3}$$

Generating Matrices and Destructive Weight Sharing. There are various ways to create the generating matrices $\{\boldsymbol{G}_i\}$. While in general $\boldsymbol{G}$ can be any full-rank tall matrix, this paper focuses on the destructive generating matrices, which are random orthogonal matrices and could prevent different kernels from sharing the representation during weight sharing. Random generating matrices empirically improve the model capacity when the model DoF is fixed. We provide an intuitive theoretical explanation of how random orthogonal matrices prevent representation sharing as follows.

For easier discussion, let us consider a special case, where all of the convolutional kernels have the same size and are used in the same shape in the corresponding convolution layers. The dimension of $\mathbf{W}$ is equal to that of one convolutional layer kernel. In other words, $\{\boldsymbol{G}_i\}$ are square matrices, and the spatial sizes of all of the convolutional kernels have the same size, $d_{in} \times d_{out} \times w \times h$, and the input channel dimension d_{in} is always equal to the output channel dimension d_{out}. In this case, a filter $\mathbf{f}$ in a kernel can be treated as a vector

[1] A kernel contains all the filters of one layer. In this paper, we treat each convolutional kernel as a vector. When the kernel is used to do the convolution, it will be reshaped into the corresponding shape.

in $\Re^{dwh}$. Further, we choose $\boldsymbol{G}_i$ to be a block-diagonal matrix $\boldsymbol{G}_i = \text{diag}\{\boldsymbol{A}_i, \boldsymbol{A}_i, \ldots, \boldsymbol{A}_i\}$, where $\boldsymbol{A}_i \in O(dwh)$ is an orthogonal matrix that generates each filter of the kernel $\mathbf{K}_i$ from $\mathbf{W}$, and $O(\cdot)$ denotes the orthogonal group. Similar to the Proposition 2 in [53], we show in the Appendix C.3 that: if $\boldsymbol{A}_i$, $\boldsymbol{A}_j$ are sampled from the $O(dwh)$ Haar distribution and $\mathbf{f}_i$, $\mathbf{f}_j$ are the corresponding filters (generated by $\boldsymbol{G}_i$, $\boldsymbol{G}_j$ respectively from the same set of entries of $\mathbf{W}$) from $\mathbf{K}_i$, $\mathbf{K}_j$ respectively, then we have $\mathbb{E}\left[\langle \mathbf{f}_i, \mathbf{f}_j \rangle\right] = 0$ and $\mathbb{E}\left[\langle \frac{\mathbf{f}_i}{\|\mathbf{f}_i\|}, \frac{\mathbf{f}_j}{\|\mathbf{f}_j\|} \rangle^2\right] = \frac{1}{dwh}$. Since dwh is usually large, the corresponding filters from $\mathbf{K}_i$, $\mathbf{K}_j$ are close to orthogonal and generally dissimilar. This shows that even when $\{\mathbf{K}_i\}$ are generated from the same entries of $\mathbf{W}$, they are prevented from sharing the representation.

Though $\{\boldsymbol{G}_i\}$ are not updated during training, the size of $\boldsymbol{G}_i$ can be quite large in general, which can create additional computation and storage overhead. In practice, we can use permutation and element-wise random sign reflection to construct a subset of the orthogonal group as permutations and sign reflections could be implemented with high simplicity and negligible cost. A simple demonstration of $\{\boldsymbol{G}_i\}$ is demonstrated in Fig.9.2**U**[2]. Since pseudo-random numbers are used, it takes only two random seeds to store a random permutation and an element-wise random sign reflection.

Even Parameter Sampling and Model Ring. While it is easy to randomly sample elements from $\mathbf{W}$ when generating parameters for each layer, it may not be optimal as some elements in $\mathbf{W}$ may not be evenly used, and some elements in $\mathbf{W}$ used at all due to sampling fluctuation. A simple equalization technique can be used to guarantee all elements of $\mathbf{W}$ are evenly sampled. Suppose the size of $\mathbf{W}$ is M, and the size of parameter $\hat{\mathbf{W}}$ of the model to be generated is N, $N > M$. As we mentioned earlier, there are L layers and they require $\{\|K_1\|, \ldots, \|K_L\|\}$ parameters respectively. As $N > M$, we can use W as a ring: we first draw the first $\|K_1\|$ parameters from $\hat{\mathbf{W}}$ followed by a pre-generated random permutation p_1 and a pre-generated random element-wise sign flipping b_1 to construct layer-1 kernel $\mathbf{K}_1$. Then we draw the next $\|K_2\|$ parameters from $\hat{\mathbf{W}}$ followed by pre-generated random permutation p_2 and a pre-generated random element-wise sign flipping b_2. We continue this process and wrap around when there is not enough entries left from $\hat{\mathbf{W}}$. We refer to $\hat{\mathbf{W}}$ together with this sampling strategy as *model rings* since the free parameters are recurrently used in a loop. We illustrate the general parameter generator in Fig.9.2**U** and RPG in Fig.9.2**L**. This For data saving efficiency, we just need to save several random seed numbers instead of saving the pre-generated permutations $\{p_1, \ldots, p_L\}$ and sign flipping operations $\{b_1, \ldots, b_L\}$.

Batch Normalization. Model performance is relatively sensitive to the batch normalization parameters. For better performance, each convolution layer needs to have its own batch normalization parameters. In general, however, the size of batch normalization is relatively negligible. Yet when $\mathbf{W}$ is extremely small (e.g., 36K parameters), the size of batch normalization should be considered.

[2]Permutations and element-wise random sign reflection conceptually are subgroups from the orthogonal group, but we shall never use them in the matrix form for the obvious efficiency purpose.

9.4 RPG at Multiple Scales

We discuss the general idea of parameter generators where only one RPG is shared globally across all layers previously. We could also create several local RPGs, each of which is shared at certain scales, such as blocks and sub-networks. Such RPGs may be useful for certain applications such as recurrent modeling.

RPGs at Block-Level. Many existing network architectures reuse the same design of network blocks multiple times for higher learning capacity, as discussed in the related work. Instead of using one global RPG for the entire network, we could alternatively create several RPGs that are shared within certain network blocks. We take Res18 [113] as a concrete example.Res18 has four building blocks. Every block has 2 residual convolution modules. We create four local RPGs for Res18. Each RPG is shared within the corresponding building block, where the size of the RPG is flexible and can be determined by users. Fig.9.3**M**) illustrates how RPGs can be shared at the block-level.

RPGs at Sub-Network-Level. Reusing sub-networks, or recurrent networks, has achieved success in many tasks as they iteratively refine and improve the prediction. Parameters are often shared when reusing the sub-networks. This may not be optimal as sub-networks at different stages iteratively improve the prediction, and shared parameters may limit the learning capacity at different stages. However, not sharing parameters at all greatly increases the model size. RPG can be created for each sub-network. Such design leads to a much smaller DoF, while parameters of different sub-networks are orthogonal by undergoing destructive changes.We show applications of sub-network-level RPGs for pose estimation and multitask regression (Section 9.5.3 and 9.5.4). Fig.9.3**R**) illustrates sub-network-level RPGs.

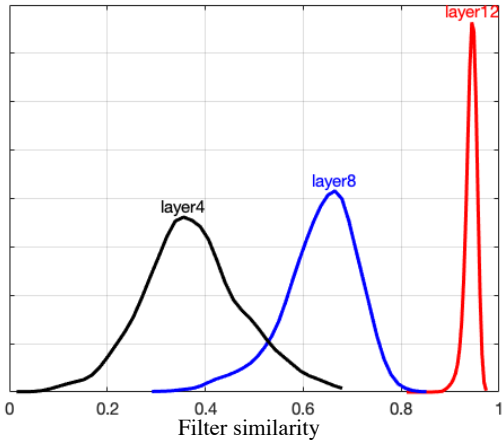
9.5 Experimental Results

We evaluate the performance of RPG with various tasks illustrated in Fig.9.3. For classification, RPG was used for the entire network except for the last fully-connected layer. We discuss performance with regard to *backbone DoF*, the actual number of parameters of the backbone. For example, Res18 has 11M backbone parameters and 512K fc parameters, and RPG was applied to reduce 11M backbone DoF only.

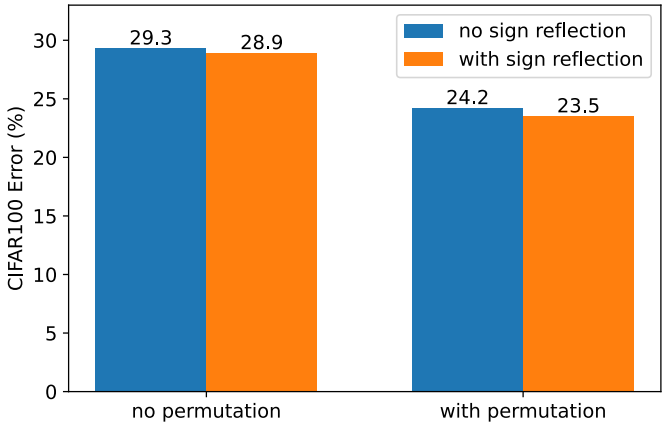
9.5.1 CIFAR Classification

Implementation Details. CIFAR experiments use 128 batch size, 5e-4 weight decay, initial learning rate of 0.1 with gamma of 0.1 at epoch 60, 120 and 160. We use Kaiming initialization [114] with adaptive scaling. Shared parameters are initialized with a particular variance and scale the parameters for each layer to make it match the Kaiming initialization.

Compared to Deep Equilibrium Models. As a representative of implicit models, deep equilibrium models [21] reduce model DoF by finding fix points via additional optimizations. We compare the image classification accuracy on CIFAR10 and CIFAR100, as well as the



a) Large models have high redundancy



b) Ablation studies of permutation and sign reflection

Figure 9.4: **a)** Large models are known to have high redundancy and low degree of freedom (DoF). They could be pruned to small models, e.g. high filter similarity of different layers in VGG16 is observed. **b)** Ablation studies of permutation and sign reflection of Res34-RPG. Having both matrices gives the highest performance.

inference time on CIFAR100 (Table 9.1). Following the settings of MDEQ [22], an image was sequentially fed into the initial convolutional block, the multi-scale deep equilibrium block (dubbed as *MS* block), and the classification head. MDEQ [22] achieves *infinite* MS blocks by finding the fixed point of the MS block. We reuse the MS block two to four times without increasing the model DoF. RPG achieves 3% - 6% gain on CIFAR10 and 3% - 6% gain on CIFAR100. RPG inference time is 15 - 25 times smaller than MDEQ since MDEQ needs additional time to solve equilibrium during training.

Global RPG with Varying Model DoF. We create one global RPG to generate param-

Table 9.1: RPG compared with multiscale deep equilibrium models (MDEQ) [22] on CIFAR10 and CIFAR100 classification. At the same number of model DoF, RPG achieves 3% - 6% performance gain with 15 - 25x less inference time. Inference time is measured by milliseconds per image.

Accuracy (%)	MDEQ	Our RPG (same DoF)		
		2x MS blk	3x MS blk	4x MS blk
CIFAR10	85.1	88.5	90.1	**90.9**
CIFAR100	59.8	62.8	64.7	**65.7**
Inference time (ms)	3.15	**0.12**	0.18	0.22

Table 9.2: ResNet-RPG outperforms existing DoF reduction methods [102, 48, 339] on CIFAR100. Additionally, a global RPG outperforms block-wise local RPGs.

	DoF	Acc. (%)
R18-vanilla	11M	77.5
R34-RPG.blk	11M	78.5
R34-RPG	11M	**78.9**
R34-random weight share	11M	74.9
R34-DeepCompression [102]	11M	72.2
R34-Hash [48]	11M	75.6
R34-Lego [339]	11M	78.4
R34-vanilla	21M	**79.1**

eters for convolution layers of ResNet and refer to it as *ResNet-RPG*. We report CIFAR100 top-1 accuracy of ResNet-RPG18 and ResNet-RPG34 at different model DoF (Table 9.3 and Fig.C.1 in Appendix C.2). Compared to ResNet, ResNet-RPG achieves higher accuracy at the same model DoF. Specifically, we achieve 36% CIFAR100 accuracy with only 8K backbone DoF. Further, ResNet34-RPG achieves higher accuracy than ResNet18-RPG, indicating increasing time complexity gives performance gain. We observe log-linear DoF-accuracy relationship, with details in *Power Law* of the following subsection.

Local RPGs at the Block-Level. In the previous Res-RPG experiments, we use one global RPG for the entire network. We also evaluate the performance when RPGs are shared locally at a block level, as discussed in Section 9.5.4. In Table 9.2, compared to plain ResNet18 at the same DoF, our block-level RPG network gives 1.0% gain. In contrast, our ResNet-RPG (parameters are evenly distributed) gives a 1.4% gain. Using one global RPG where parameters of each layer are evenly distributed is 0.4% higher than multiple RPGs.

Comparison to Baselines. Table 9.2 compares RPG and other model DoF reduction methods including random weight sharing, weight sharing with the deep compression [102], hashing trick [48] and weight sharing with Lego filters [339]. We also compare with HyperNetworks [100] in Appendix C.4. At the same model DoF, RPG outperforms all other baselines, demonstrating the effectiveness of the proposed method.

RPG for Transformers. We apply RPG for a vision transformer ViT [73] and report results in Fig.9.5**a**. Specifically, the ViT-tiny model with 6 transformer layers, 4 attention heads and 64 embedding dimensions, is used as a baseline. A log-linear relationship is also identified in ViT-RPG.

9.5.2 ImageNet Classification

Implementation Details. All ImageNet experiments use batchsize of 256, weight decay of 3e-5, and an initial learning rate of 0.3 with gamma of 0.1 every 75 epochs and 225 epochs in total. Our schedule is different from the standard schedule as the weight-sharing mechanism requires different training dynamics. We tried a few settings and found this one to be the best for RPG.

RPG with Varying Model DoF. We use RPG with different DoF for ResNet and report the top-1 accuracy (Table 9.3 and Fig.9.1**e**)). ResNet-RPGs consistently achieve higher performance than ResNets under the same model DoF. Specifically, ResNet-RPG34 achieves the same accuracy 73.4% as ResNet34 with only half of ResNet34 backbone DoF. ResNet-RPG18 also achieves the same accuracy as ResNet18 with only half of ResNet18 backbone DoF. Further, RPG networks have higher generalizability (Section 9.5.6).

Power Law. Empirically, accuracy and model DoF follow a power law, when RPG DoF is lower than 50% ResNet-vanilla DoF (Fig.9.1**d**). The exponents of the power laws are the same for ResNet18-RPG and ResNet34-RPG on ImageNet. The scaling law may be useful for estimating the network accuracy without training the network. Similarly, [122] also identifies a power law for accuracy and model DoF of transformers. The proposed RPG

Table 9.3: ResNet-RPG consistently achieves higher performance at the same model DoF. We report ImageNet and CIFAR100 top-1 accuracy and backbone DoF for ResNet-vanilla and ResNet-RPG.

Acc. (%)	**R18-RPG**			**R18-vanilla**	**R34-RPG**			**R34-vanilla**
ImageNet	40.0	67.2	70.5	70.5	41.6	69.1	**73.4**	**73.4**
CIFAR100	60.2	75.6	77.6	77.6	61.7	76.5	**78.9**	**79.1**
Model DoF	45K	2M	5.5M	11M	45K	2M	11M	21M

enables *under-parameterized* models for large-scale datasets such as ImageNet, which may unleash more new studies and findings.

9.5.3 Pose Estimation

Implementation Details. We superpose sub-networks for pose estimation with a globally shared RPG. Hourglass networks [213] are used as the backbone. An input image is first fed to an initial convolution block to obtain a feature map, which is then fed to multiple stacked pose estimation sub-networks. Each sub-network outputs a pose estimation prediction, which is penalized by the pose estimation loss. Convolutional pose machine (CPM) [318] share all sub-networks weights. We create one global RPG to generate parameters for each sub-network. Our model size is set to the same as CPM. We also compare with larger models where parameters of sub-networks are not shared.

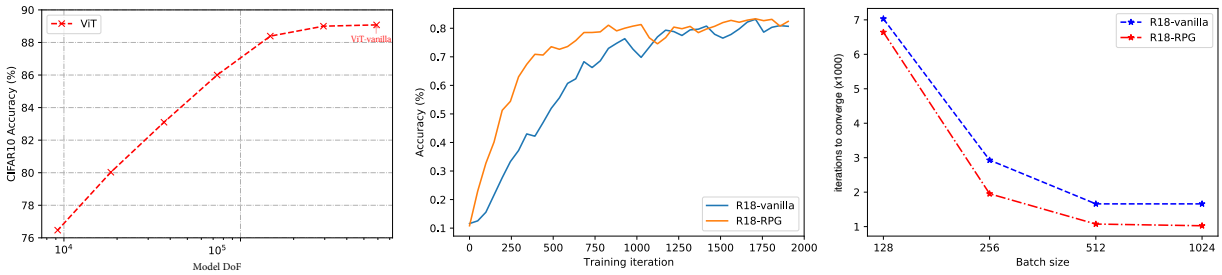
We evaluate on MPII Human Pose dataset [10], a benchmark for articulated human pose estimation, which consists of over 28K training samples over 40K people with annotated body joints. We use the hourglass network [213] as backbone and follow all their settings.
Results and Analysis. We report the Percentage of Correct Key-points at 50% threshold (PCK@0.5) of different methods in Table 9.4. CPM [318] share all parameters for different sub-networks. We use one RPG that is shared globally at the same size as CPM. For

Table 9.4: RPG outperforms CPM [318] at the same DoF. We report pose estimation performance (model DoF) on MPII human pose compared with CPM [318]. The metric is PCKh@0.5.

Acc. (DoF)	**CPM** [318]	**RPG**	**No shared w.**
1x sub-net	84.7 (3.3M)		
2x sub-nets	86.1 (3.3M)	86.5 (3.3M)	87.1 (6.7M)
4x sub-nets	86.5 (3.3M)	87.3 (3.3M)	88.0 (13.3M)

Table 9.5: RPG achieves the best accuracy without sharing batch normalize parameters and with permutation and sign reflection. We report multitask regression errors on S3DIS with sub-net architecture as [243]. Lower is better. All methods share the same DoF. Sub-net is reused once.

RMSE (%)	**Depth**	**Normal**
Vanilla model	25.5	41.0
RPG with shared BN	24.7	40.3
Reuse & new BN	24.0	39.4
Reuse & new BN & perm. and reflect.	**22.8**	**39.1**



a) Log-linear DoF-accuracy relationship for ViT-RPG b) RPG converges faster than the vanilla model c) RPG converges faster for different batch sizes

Figure 9.5: **a)** A log-linear DoF-accuracy relationship exists for RPGs applied to vision transformer ViT [73]. **b)** RPG converges faster than the vanilla model. We plot the CIFAR10 accuracy (smoothed by moving average) versus training iterations for Res18-vanilla and Res18-RPG. RPG converges at 1k iterations while the vanilla model converges at 1.7k. **c)** RPG consistently converges faster. The reduction becomes substantial with the increasing batchsize, e.g., at batchsize 1024, RPG takes 41% less iterations to converge. Denote final accuracy as P_f, the convergence iteration is defined when current smoothed accuracy (by moving average) is within 5% range of P_f.

reference, we also compare with the no-sharing model as the performance ceiling. Adding the number of recurrences leads to performance gain for all methods. At the same model size, RPG achieves higher PCK@0.5 compared to CPM. Increasing the number of parameters by not sharing sub-network parameters also leads to some performance gain.

9.5.4 Multi-Task Regression

Implementation Details. We superpose sub-networks for multi-task regression with multiple RPGs at the building-block level. We focus on predicting depth and normal maps from a given image. We stack multiple SharpNet [243], a network for monocular depth and normal estimation. Specifically, we create multiple RPGs at the SharpNet building-block level. That is, parameters of corresponding blocks of different sub-networks are generated from the same RPG.

We evaluate the monocular depth and normal prediction performance on a 3D indoor scene dataset [16], which contains over 70K images with corresponding depths and normals covering over 6,000 m^2 indoor area. We follow all settings of SharpNet [243], a SOTA monocular depth and normal estimation method.

Results and Analysis. We report the mean square errors for depth and normal estimation in Table 9.5. Compared to one-time inference without recurrence, our RPG network gives 3% and 2% gain for depth and normal estimation, respectively. Directly sharing weights but using new batch normalization layers decrease the performance by 1.2% and 0.3% for depth and normal. Sharing weights and normalization layers further decrease the performance by 0.7% and 0.9% for depth and normal.

Table 9.6: RPG achieves higher post-pruning CIFAR10 accuracy and similar post-pruning accuracy drops as SOTA fine-grained pruning approach IMP [84]. Fine-grained pruning is used for reducing DoF.

	acc before	acc after ↓ DoF	acc drop	model DoF
R18-IMP [84]	92.3	90.5	1.8	274k
R18-RPG	95.0	93.0	2.0	274k

Table 9.7: RPG achieves similar post-pruning ImageNet performance as SOTA coarse-grained apporach Knapsack [5] at the same FLOPs. Coarse-grained pruning is used for reducing RPG FLOPs.

	DoF before pruning	Pruned acc.	FLOPs
R18-Knapsack	11.2M	69.35%	1.09e9
Pruned R18-RPG	5.6M	69.10%	1.09e9

Table 9.8: RPG increases the model generalizability. **(a)** ResNet-RPG has lower training-validation accuracy gap on ImageNet classification. The metric is training accuracy minus validation accuracy. Lower is better. **(b)** Using RPG for pose estimation also decreases the training and validation performance GAP. The metric is training PCK@0.5 minus validation PCK@0.5. Lower is better. **(c)** ResNet with RPG has higher performance on out-of-distribution dataset ObjectNet [26]. The model is trained on ImageNet only and directly evaluated on ObjectNet.

(a) IN train-val gap

Acc gap (%)	vanilla	RPG
R18	-0.7	**-2.7**
R34	1.1	**-2.3**

(b) Pose train-val gap

Acc gap (%)	no shared w	shared w	RPG
2x sub-nets	1.15	1.13	**0.64**
4x sub-nets	1.98	1.70	**1.15**

(c) OOD on ObjectNet

	R18	R34-RPG	R34
DoF	11M	11M	21M
Acc. (%)	13.4	**16.5**	16.0

9.5.5 Pruning RPG

Fine-Grained Pruning. Fine-grained pruning methods aim to reduce the model DoF by sparsifying weight matrices. Such methods usually do not reduce the inference speed, although custom algorithms [88] may improve the speed. At the same model DoF, RPG outperforms state-of-the-art fine-grained pruning method IMP [84]. Accuracy drops of RPG and IMP are similar, both around 2% (Table 9.6). It is worth noting that although IMP has no run time improvement in regular settings, it could save inference time with customized sparse GPU kernels [88].

Coarse-Grained Pruning. While RPG is not designed to reduce FLOPs, it can be combined with coarse-grained pruning to reduce FLOPs. We prune RPG filters with the lowest ℓ_1 norms. Table 9.7 shows that the pruned RPG achieves on-par performance as state-of-the-art coarse-grained pruning method Knapsack [5] at the same FLOPs.

9.5.6 Analysis

Convergence rate. Compared with the vanilla model, RPG optimizes in a parameter subspace $\hat{\mathbf{W}} = \boldsymbol{G}\mathbf{W}$ with fewer DoF. Would such constrained optimization lead to a faster convergence rate? We analyze the convergence rate of Res18-vanilla and Res18-RPG (DoF is 5.5M, 50% of the vanilla model) with different batchsizes. All models are trained with multi-step SGD optimizer and they all reach $> 94.1\%$ final CIFAR10 accuracy. For simplicity, we

analyze the first optimization stage where learning rate has not decayed.

Fig.9.5**b** plots the accuracy (smoothed with moving averages) v.s. training iterations with batchsize 1024. RPG has a faster convergence rate than vanilla models. We also analyze the smoothed accuracy and identify the convergence iteration versus batchsize in Fig.9.5**c**. RPG consistently converges faster than the vanilla model, and the reduction becomes substantial with the increasing batchsize.

Comparison to Model Compression Methods. We report ResNet-RPG performance with different model DoF and existing compression methods on ImageNet (Fig.9.1**e**). RPG networks outperform SOTA methods such as [5, 70, 117, 118, 71, 149]. For example, at the same model DoF, our RPG network has 0.6% gain over the knapsack pruning [5], a SOTA method of ImageNet pruning.

Storage. RPG models only need to save the effective parameter $\mathbf{W}$, which has the size of the model DoF, since the generation matrix G is saved as a random seed at no cost. The storage space of the model file can be diminished to satisfy a smaller storage limit for inference and a faster model file transfer. Empirically on PyTorch platform, ResNet18-vanilla model file is 45MB. With no accuracy loss, ResNet18-RPG model save file size is 23MB ($\downarrow$ 49%). With 2 percentage point accuracy loss, RPG save file size is 9.5MB ($\downarrow$ 79%).

Generalizability. We report the performance gap between training and validation set on ImageNet (Table 9.8(a)) and MPII pose estimation (Table 9.8(b)). CPM [318] serves as the baseline pose estimation method. RPG models consistently achieve lower gaps between training and validation sets, indicating the RPG model suffers less from over-fitting.

We also report the out-of-distribution performance of RPG models. ObjectNet [26] contains 50k images with 113 classes overlapping with ImageNet. Existing models are reported to have a large performance drop on ObjectNet. We directly evaluate the performance of ImageNet-trained model on ObjectNet without any fine-tuning (Table 9.8(c)). With the same backbone DoF, R18-RPG achieves a 3% gain compared to R18-vanilla. With the same network architecture design, R34-RPG achieves 0.5% gain compared to R34. This indicates RPG networks have higher out-of-distribution performance even with smaller model DoF.

Quantization. Network quantization can reduce model size with minimal accuracy drop. It is of interest to study if RPG models, whose parameters have been shrunk, can be quantized. After 8-bit quantization, the accuracy of ResNet18-RPG (5.6M DoF) only drop 0.1 percentage point on ImageNet, indicating RPG can be quantized for further model size reduction. Details are in Appendix C.1.

Security. Permutation matrices generated by the random seed can be considered as security keys to decode the model. Further, only random seeds to generate generating matrix G need to be saved and transferred at negligible cost.

9.5.7 Ablation Studies

We conduct ablation studies on CIFAR100 to analyze functions of permutation and reflection matrices (Fig.9.4**b**. We evaluate ResNet-RPG34 with 2M backbone DoF. Permutation and sign reflection together achieves 76.5% accuracy, while permutation only achieves 75.8%,

and sign reflection only achieves 71.1%. Training with neither permutation nor reflection matrices achieves 70.7%. This suggests permutation and sign reflection matrices increase RPG performance.

9.6 Discussion

The common practice in neural network compression is to prune weights from a trained large model with many parameters or degrees of freedom (DoF). Our key insight is that a direct and drastically different approach might work faster and better: We start from a lean model with a small DoF, which can be linearly unpacked into a large model with many parameters. Then we can let the gradient descent automatically find the best model under the linear constraints. Our work is a departure from mainstream approaches towards model optimization and parameter reduction. We show how the model DoF and actual parameter size can be decoupled: we can define an arbitrary network of an arbitrary DoF.

We limit our scope to optimization with random linear constraints, termed destructive weight sharing. However, in general, there might also exist nonlinear RPGs and efficient nonlinear generation functions to create convolutional kernels from a shared model ring $\mathbf{W}$. Further, although RPG focuses on reducing model DoF, it can be quantized and pruned to further reduce the FLOPs and runtime.

To sum up, we develop an efficient approach to build an arbitrarily complex neural network with any amount of DoF via a recurrent parameter generator. On a wide range of applications, including classification, pose estimation and multitask regression, we show RPG consistently achieves higher performance at the same model DoF. Further, we show such networks converge faster, are less likely to overfit and have higher performance on out-of-distribution data.

RPG can be added to any existing network flexibly with any amount of DoF at the user's discretion. It provides new perspectives for recurrent models, equilibrium models, and model compression. It also serves as a tool for understanding relationships between network properties and network DoF by factoring out the network architecture.

Chapter 10

Summary

In conclusion, this dissertation presents data-driven approaches for learning structure-aware representations from visual data, with applications to several medical problems. By learning individual-object-aware and 3D-geometry-aware representations, we demonstrate robust performance, as well as improved generalizability, interpretability and reliability of the model. Furthermore, we introduce a constrained neural optimization method to improve the efficiency of deep learning models, which is demonstrated in orthogonal convolutional neural networks and recurrent parameter generators.

Our approach is effective in various applications, as demonstrated by our experiments on standard computer vision benchmark datasets and real-world medical scenarios, including the diagnosis of dry eye diseases and 3D dental casting. Overall, our work contributes to the field of visual representation learning and enables new applications in medicine and beyond.

The current structure-aware representation learning algorithms focus solely on passive learning, where patterns are learned from existing data without much consideration for actions and their impact on the world. This approach may not be optimal, as the world is constantly changing, and neglecting actions and world states could result in inaccurate predictions.

To overcome this limitation, we must shift towards active learning, where a real-time multi-modality decision system can quickly comprehend the situation (the world state) and act accordingly. This approach requires us to build on the structure-aware representations and develop effective actionable representations from dynamic scenes and multi-modality data. With such representations, we can create a world model that understands world states based on different actions. The method may unleash wide applications ranging from healthcare to sciences.

Bibliography

[1] Malcolm Acock. "Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. By David Marr". In: *The Modern Schoolman* 62.2 (1985), pp. 141–142.

[2] Accountability Act. "Health insurance portability and accountability act of 1996". In: *Public law* 104 (1996), p. 191.

[3] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. "Fast high-dimensional filtering using the permutohedral lattice". In: *Computer Graphics Forum*. Vol. 29. 2. 2010, pp. 753–762.

[4] Muhammed Yasin Adil et al. "Meibomian gland morphology is a sensitive early indicator of meibomian gland dysfunction". In: *American journal of ophthalmology* 200 (2019), pp. 16–25.

[5] Yonathan Aflalo et al. "Knapsack pruning with inner distillation". In: *arXiv preprint arXiv:2002.08258* (2020).

[6] Kristina Aldridge et al. "Precision and error of three-dimensional phenotypic measures acquired from 3dMD photogrammetric images". In: *American journal of medical genetics Part A* 138.3 (2005), pp. 247–253.

[7] Dimitrios S Alexiadis et al. "An integrated platform for live 3D human reconstruction and motion capturing". In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.4 (2016), pp. 798–813.

[8] Yousef A Alghamdi et al. "The epidemiology of meibomian gland dysfunction in an elderly population". In: *Cornea* 35.6 (2016), p. 731.

[9] Basma Ammour et al. "Face–iris multimodal biometric identification system". In: *Electronics* 9.1 (2020), p. 85.

[10] Mykhaylo Andriluka et al. "2d human pose estimation: New benchmark and state of the art analysis". In: *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*. 2014, pp. 3686–3693.

[11] Reiko Arita et al. "Contact lens wear is associated with decrease of meibomian glands". In: *Ophthalmology* 116.3 (2009), pp. 379–384.

[12] Reiko Arita et al. "Noncontact infrared meibography to document age-related changes of the meibomian glands in a normal population". In: *Ophthalmology* 115.5 (2008), pp. 911–915.

[13] Reiko Arita et al. "Proposed diagnostic criteria for obstructive meibomian gland dysfunction". In: *Ophthalmology* 116.11 (2009), pp. 2058–2063.

[14] Martin Arjovsky, Amar Shah, and Yoshua Bengio. "Unitary evolution recurrent neural networks". In: *ICML*. 2016, pp. 1120–1128.

[15] Iro Armeni et al. "3d semantic parsing of large-scale indoor spaces". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2016, pp. 1534–1543.

[16] Iro Armeni et al. "Joint 2d-3d-semantic data for indoor scene understanding". In: *arXiv preprint arXiv:1702.01105* (2017).

[17] Amir Arsalan Soltani et al. "Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1511–1519.

[18] Penny A Asbell et al. "Dry eye assessment and management (DREAM©) study: study design and baseline characteristics". In: *Contemporary Clinical Trials* 71 (2018), pp. 70–79.

[19] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[20] V Badrinarayanan, A Kendall, and R Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 12, pp. 2481-2495, Dec. 1* (2017).

[21] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. "Deep Equilibrium Models". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 690–701.

[22] Shaojie Bai, Vladlen Koltun, and J Zico Kolter. "Multiscale Deep Equilibrium Models". In: *Advances in Neural Information Processing Systems* 33 (2020).

[23] Randall Balestriero et al. "A spline theory of deep networks". In: *ICML*. 2018, pp. 383–392.

[24] Randall Balestriero and Richard Baraniuk. "Mad max: Affine spline insights into deep learning". In: *arXiv preprint arXiv:1805.06576* (2018).

[25] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. "Can we gain more from orthogonality regularizations in training deep CNNs?" In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018, pp. 4266–4276.

[26] Andrei Barbu et al. "Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models". In: *Advances in neural information processing systems* 32 (2019), pp. 9453–9463.

[27] David Bau et al. "Network dissection: Quantifying interpretability of deep visual representations". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6541–6549.

[28] C Baudouin et al. "Boboridis KG, Merayo-Lloves J., Rolando M., Labetoulle M. Revisiting the vicious circle of dry eye disease: a focus on the pathophysiology of meibomian gland dysfunction". In: *Br J Ophthalmol* 100.3 (2016), pp. 300–306.

[29] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166.

[30] Davis Blalock et al. "What is the state of neural network pruning?" In: *Proceedings of Machine Learning and Systems*. 2020.

[31] Rishi Bommasani et al. "On the opportunities and risks of foundation models". In: *arXiv preprint arXiv:2108.07258* (2021).

[32] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer. 2010, pp. 177–186.

[33] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *ICLR*. 2019.

[34] Andrew Brock et al. "Neural Photo Editing with Introspective Adversarial Networks". In: *ICLR*. 2017.

[35] AJ Bron et al. "Functional aspects of the tear film lipid layer". In: *Experimental eye research* 78.3 (2004), pp. 347–360.

[36] Tom B Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[37] Nicholas J Butko and Javier R Movellan. "Optimal scanning for faster object detection". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 2751–2758.

[38] Igor A Butovich. "Meibomian glands, meibum, and meibogenesis". In: *Experimental eye research* 163 (2017), pp. 2–16.

[39] Han Cai, Ligeng Zhu, and Song Han. "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware". In: *International Conference on Learning Representations*. 2018.

[40] Zhe Cao et al. "Realtime multi-person 2d pose estimation using part affinity fields". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[41] Joao Carreira et al. "Human pose estimation with iterative error feedback". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4733–4742.

[42] Mario Lezcano Casado and David Martinez-Rubio. "Cheap Orthogonal Constraints in Neural Networks: A Simple Parametrization of the Orthogonal and Unitary Group". In: *ICML*. 2019, pp. 3794–3803.

[43] Turgay Celik et al. "Bioimage informatics approach to automated meibomian gland analysis in infrared images of meibography". In: *Journal of optometry* 6.4 (2013), pp. 194–204.

[44] Rudrasis Chakraborty, Jiayun Wang, and Stella X Yu. "Sur-real: Frechet mean and distance transform for complex-valued deep learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[45] Angel X Chang et al. "Shapenet: An information-rich 3d model repository". In: *arXiv:1512.03012* (2015).

[46] Angel X Chang et al. "Shapenet: An information-rich 3d model repository". In: *arXiv preprint arXiv:1512.03012* (2015).

[47] Liang-Chieh Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[48] Wenlin Chen et al. "Compressing neural networks with the hashing trick". In: *International conference on machine learning*. PMLR. 2015, pp. 2285–2294.

[49] Xi Chen et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets". In: *Advances in neural information processing systems*. 2016, pp. 2172–2180.

[50] Yiping Chen et al. "LiDAR-Video Driving Dataset: Learning Driving Policies Effectively". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2018, pp. 5870–5878.

[51] Yunpeng Chen et al. "Training Group Orthogonal Neural Networks with Privileged Information". In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2017, pp. 1532–1538.

[52] Brian Cheung et al. "Superposition of many models into one". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019, pp. 10867–10876.

[53] Brian Cheung et al. "Superposition of many models into one". In: *Advances in neural information processing systems*. 2019.

[54] T Ching et al. *Opportunities and obstacles for deep learning in biology and medicine. JR Soc. Interface 15*. 2017.

[55] Christopher B Choy et al. "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction". In: *European conference on computer vision*. Springer. 2016.

[56] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[57] Lynne Coventry and Dawn Branley. "Cybersecurity in healthcare: A narrative review of trends, threats and ways forward". In: *Maturitas* 113 (2018), pp. 48–52.

[58] Ricaurte Ramiro Crespo-Treviño et al. "Comparative of meibomian gland morphology in patients with evaporative dry eye disease versus non-dry eye disease". In: *Scientific Reports* 11.1 (2021), p. 20729.

[59] Angela Dai et al. "ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans". In: *Proc. Conf. Comput. Vis. Pat. Recog.* Vol. 1. 2018, p. 2.

[60] Dengxin Dai and Luc Van Gool. "Dark model adaptation: Semantic image segmentation from daytime to nighttime". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 3819–3824.

[61] Jifeng Dai et al. "Deformable convolutional networks". In: *Proc. Int. Conf. Comput. Vis.* 2017, pp. 764–773.

[62] Reza Dana et al. "Estimated prevalence and incidence of dry eye disease based on coding analysis of a large, all-age United States health care system". In: *American Journal of Ophthalmology* 202 (2019), pp. 47–54.

[63] Ebenezer Daniel et al. "Grading and baseline characteristics of meibomian glands in meibography images and their clinical associations in the Dry Eye Assessment and Management (DREAM) study". In: *The ocular surface* 17.3 (2019), pp. 491–501.

[64] Yann N Dauphin et al. "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization". In: *Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 2933–2941.

[65] Bert De Brabandere, Davy Neven, and Luc Van Gool. "Semantic instance segmentation with a discriminative loss function". In: *arXiv preprint arXiv:1708.02551* (2017).

[66] Johanna Delanoy et al. "3d sketching using multi-view deep volumetric prediction". In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.1 (2018), pp. 1–22.

[67] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *Proc. Conf. Comput. Vis. Pat. Recog.* Ieee. 2009, pp. 248–255.

[68] Endri Dibra et al. "Human shape from silhouettes using generative hks descriptors and cross-modal neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[69] Piotr Dollár, Mannat Singh, and Ross Girshick. "Fast and accurate model scaling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 924–932.

[70] Xuanyi Dong and Yi Yang. "Network Pruning via Transformable Architecture Search". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[71] Xuanyi Dong et al. "More is less: A more complicated network with less inference complexity". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5840–5848.

[72] Victor Dorobantu, Per Andre Stromhaug, and Jess Renteria. "Dizzyrnn: Reparameterizing recurrent neural networks for norm-preserving backpropagation". In: *arXiv preprint arXiv:1612.04035* (2016).

[73] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[74] Yilun Du and Igor Mordatch. "Implicit generation and generalization in energy-based models". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[75] Thomas J Dursch et al. "Tear-film evaporation rate from simultaneous ocular-surface temperature and tear-breakup area". In: *Optometry and Vision Science* 95.1 (2018), pp. 5–12.

[76] B Ehteshami Bejnordi, M Veta, and PJ Diest. "van, Ginneken B, van, Karssemeijer N, Litjens G, et al. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer". In: *JAMA* 318 (2017), pp. 2199–210.

[77] Mathias Eitz, James Hays, and Marc Alexa. "How Do Humans Sketch Objects?" In: *ACM Transactions on Graphics* 31.4 (2012), 44:1–44:10.

[78] A Esteva et al. "Dermatologist-level classification of skin cancer with deep neural networks". In: *Dermatologist-level classification of skin cancer with deep neural networks. nature* 542.7639 (), pp. 115–118.

[79] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. "A Point Set Generation Network for 3D Object Reconstruction from a Single Image". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[80] Alireza Fathi et al. "Semantic instance segmentation via deep metric learning". In: *arXiv preprint arXiv:1703.10277* (2017).

[81] Yifan Feng et al. "GVCNN: Group-view convolutional neural networks for 3D shape recognition". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2018, pp. 264–272.

[82] Virgilio F Ferrario et al. "Three-dimensional computerized anthropometry of the nose: landmark representation compared to surface analysis". In: *The Cleft palate-craniofacial journal* 44.3 (2007), pp. 278–285.

[83] David Finis et al. "Evaluation of meibomian gland dysfunction and local distribution of meibomian gland atrophy by non-contact infrared meibography". In: *Current eye research* 40.10 (2015), pp. 982–989.

[84] Jonathan Frankle et al. "Stabilizing the lottery ticket hypothesis". In: *arXiv preprint arXiv:1903.01611* (2019).

[85] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. "Visual simultaneous localization and mapping: a survey". In: *Artificial intelligence review* 43.1 (2015), pp. 55–81.

[86] Samy Wu Fung et al. "Fixed point networks: Implicit depth models with jacobian-free backprop". In: *arXiv preprint arXiv:2103.12803* (2021).

[87] Matheus Gadelha, Rui Wang, and Subhransu Maji. "Multiresolution tree networks for 3d point cloud processing". In: *Proc. EUR Conf. Comput. Vis.* 2018, pp. 103–118.

[88] Trevor Gale et al. "Sparse GPU Kernels for Deep Learning". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020*. 2020.

[89] Stephen I Gallant et al. "Perceptron-based learning algorithms". In: *IEEE Transactions on neural networks* 1.2 (1990), pp. 179–191.

[90] Xinbo Gao et al. "Face sketch–photo synthesis and retrieval using sparse representation". In: *IEEE Transactions on circuits and systems for video technology* 22.8 (2012), pp. 1213–1226.

[91] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2012, pp. 3354–3361.

[92] Arnab Ghosh et al. "Interactive sketch & fill: Multiclass sketch-to-image translation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

[93] Charles D Gilbert and Mariano Sigman. "Brain states: top-down influences in sensory processing". In: *Neuron* 54.5 (2007), pp. 677–696.

[94] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2010, pp. 249–256.

[95] Xinyu Gong et al. "AutoGAN: Neural Architecture Search for Generative Adversarial Networks". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2019, pp. 3224–3234.

[96] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014.

[97] Google. "The Quick, Draw! Dataset". In: (2017). `https://quickdraw.withgoogle.com/data`.

[98] Benoit Guillard et al. "Sketch2Mesh: Reconstructing and Editing 3D Shapes from Sketches". In: *arXiv preprint arXiv:2104.00482* (2021).

[99] Chuan Guo et al. "Simple Black-box Adversarial Attacks". In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2019, pp. 2484–2493.

[100] David Ha, Andrew Dai, and Quoc V Le. "Hypernetworks". In: *arXiv preprint arXiv:1609.09106* (2016).

[101] Song Han, Huizi Mao, and William J Dally. "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding". In: *ICLR* (2016).

[102] Song Han, Huizi Mao, and William J Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding". In: *Proceedings of the International Conference on Learning Representations*. 2016.

[103] Song Han et al. "EIE: efficient inference engine on compressed deep neural network". In: *IEEE Annual International Symposium on Computer Architecture (ISCA)*. 2016, pp. 243–254.

[104] Zhizhong Han et al. "Reconstructing 3D shapes from multiple sketches using direct shape optimization". In: *IEEE Transactions on Image Processing* 29 (2020), pp. 8721–8734.

[105] Zhizhong Han et al. "SeqViews2SeqLabels: Learning 3D Global Features via Aggregating Sequential Views by RNN With Attention". In: *IEEE Transactions on Image Processing* 28.2 (2019), pp. 658–672.

[106] Christian Häne, Shubham Tulsiani, and Jitendra Malik. "Hierarchical surface prediction for 3d object reconstruction". In: *2017 International Conference on 3D Vision*. IEEE. 2017.

[107] James A Hanley and Barbara J McNeil. "The meaning and use of the area under a receiver operating characteristic (ROC) curve." In: *Radiology* 143.1 (1982), pp. 29–36.

[108] Yongchang Hao et al. "Multi-Task Learning with Shared Encoder for Non-Autoregressive Machine Translation". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021, pp. 3989–3996.

[109] Mehrtash Harandi and Basura Fernando. "Generalized backpropagation, Étude de cas: Orthogonality". In: *arXiv preprint arXiv:1611.05927* (2016).

[110] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[111] Samira Hassanzadeh et al. "Global prevalence of meibomian gland dysfunction: a systematic review and meta-analysis". In: *Ocular immunology and inflammation* 29.1 (2021), pp. 66–75.

[112] Trevor Hastie and Werner Stuetzle. "Principal curves". In: *Journal of the American Statistical Association* 84.406 (1989), pp. 502–516.

[113] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[114] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*. 2015, pp. 1026–1034.

[115] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[116] Xinwei He et al. "Triplet-center loss for multi-view 3d object retrieval". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[117] Yang He et al. "Filter pruning via geometric median for deep convolutional neural networks acceleration". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4340–4349.

[118] Yang He et al. "Soft filter pruning for accelerating deep convolutional neural networks". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2018, pp. 2234–2240.

[119] Yihui He, Xiangyu Zhang, and Jian Sun. "Channel pruning for accelerating very deep neural networks". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 1389–1397.

[120] Felix Heide, Wolfgang Heidrich, and Gordon Wetzstein. "Fast and Flexible Convolutional Sparse Coding". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[121] Felix Heide, Wolfgang Heidrich, and Gordon Wetzstein. "Fast and flexible convolutional sparse coding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 5135–5143.

[122] Tom Henighan et al. "Scaling laws for autoregressive generative modeling". In: *arXiv preprint arXiv:2010.14701* (2020).

[123] Yasushi Hirogaki et al. "Complete 3-D reconstruction of dental cast shape using perceptual grouping". In: *IEEE transactions on medical imaging* 20.10 (2001), pp. 1093–1101.

[124] Elad Hoffer and Nir Ailon. "Deep metric learning using triplet network". In: *International Workshop on Similarity-Based Pattern Recognition*. Springer. 2015, pp. 84–92.

[125] Jutta Horwath-Winter et al. "Evaluation of the clinical course of dry eye syndrome". In: *Archives of ophthalmology* 121.10 (2003), pp. 1364–1368.

[126] Andrew G Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[127] Haibin Huang et al. "Shape synthesis from sketches via procedural models and convolutional networks". In: *IEEE transactions on visualization and computer graphics* 23.8 (2016), pp. 2003–2013.

[128] Lei Huang et al. "Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks". In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2018.

[129] Qiangui Huang. *VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection - Code*. `https://github.com/qianguih/voxelnet`. 2019.

[130] Itay Hubara et al. "Quantized neural networks: Training neural networks with low precision weights and activations". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6869–6898.

[131] JM Hupé et al. "Cortical feedback improves discrimination between figure and background by V1, V2 and V3 neurons". In: *Nature* 394.6695 (1998), pp. 784–787.

[132] D Hutchison et al. "What, Where and How Many? Combining Object Detectors and CRFs". In: *Proc. of the Eur. Conf. on Comp. Vis.* 2010, pp. 424–437.

[133] Ho Sik Hwang et al. "Meibocyte differentiation and renewal: Insights into novel mechanisms of meibomian gland dysfunction (MGD)". In: *Experimental eye research* 163 (2017), pp. 37–45.

[134] PG Hykin and AJ Bron. "Age-related morphological changes in lid margin and meibomian gland anatomy". In: *Cornea* 11.4 (1992), pp. 334–342.

[135] Forrest N Iandola et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡ 0.5 MB model size". In: *arXiv preprint arXiv:1602.07360* (2016).

[136] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *ICML*. 2015, pp. 448–456.

[137] Phillip Isola et al. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[138] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. "Speeding up Convolutional Neural Networks with Low Rank Expansions". In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2014.

[139] Max Jaderberg et al. "Spatial transformer networks". In: *Proc. Conf. NeruIPS*. 2015, pp. 2017–2025.

[140] Lena A Jäger et al. "Deep eyedentification: Biometric identification using micromovements of the eye". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part II*. Springer. 2020, pp. 299–314.

[141] Varun Jampani, Martin Kiefel, and Peter V Gehler. "Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2016, pp. 4452–4461.

[142] Y Jie et al. "Prevalence of dry eye among adult Chinese in the Beijing Eye Study". In: *Eye* 23.3 (2009), pp. 688–693.

[143] Evangelos Kalogerakis et al. "3D shape segmentation with projective convolutional networks". In: *Proc. Conf. Comput. Vis. Pat. Recog.* Vol. 1. 2. 2017, p. 8.

[144] Abhishek Kar, Christian Häne, and Jitendra Malik. "Learning a multi-view stereo machine". In: *Advances in neural information processing systems*. 2017.

[145] Theofanis Karaletsos and Thang D. Bui. "Hierarchical Gaussian Process Priors for Bayesian Neural Network Weights". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Ed. by Hugo Larochelle et al. 2020.

[146] Theofanis Karaletsos, Peter Dayan, and Zoubin Ghahramani. "Probabilistic meta-representations of neural networks". In: *arXiv preprint arXiv:1810.00555* (2018).

[147] Andrej Karpathy and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3128–3137.

[148] Chung How Kau et al. "Three-dimensional surface acquisition systems for the study of facial morphology and their application to maxillofacial surgery." In: *The international journal of medical robotics+ computer assisted surgery: MRCAS* 3.2 (2007), p. 97.

[149] Ashish Khetan and Zohar Karnin. "PruneNet: Channel Pruning via Global Importance". In: *arXiv preprint arXiv:2005.11282* (2020).

[150] Ji Soo Kim, Michael TM Wang, and Jennifer P Craig. "Exploring the Asian ethnic predisposition to dry eye disease in a pediatric population". In: *The Ocular Surface* 17.1 (2019), pp. 70–77.

[151] Yeon Soo Kim and Kun Hwang. "Shape and height of tarsal plates". In: *Journal of Craniofacial Surgery* 27.2 (2016), pp. 496–497.

[152] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[153] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[154] Roman Klokov and Victor Lempitsky. "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models". In: *Proc. Int. Conf. Comput. Vis.* 2017, pp. 863–872.

[155] Erich Knop et al. "The international workshop on meibomian gland dysfunction: report of the subcommittee on anatomy, physiology, and pathophysiology of the meibomian gland". In: *Investigative ophthalmology & visual science* 52.4 (2011), pp. 1938–1978.

[156] T. Ko. "Multimodal biometric identification for large user population using fingerprint, face and iris recognition". In: *34th Applied Imagery and Pattern Recognition Workshop (AIPR'05)*. 2005, 6 pp.–223. DOI: 10.1109/AIPR.2005.35.

[157] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. "Convolutional mesh regression for single-image human shape reconstruction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[158] Jelena Kovačević, Amina Chebira, et al. "An introduction to frames". In: *Foundations and Trends in Signal Processing* 2.1 (2008), pp. 1–94.

[159] Jonathan Krause et al. "3D Object Representations for Fine-Grained Categorization". In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia, 2013.

[160] Alex Krizhevsky, Geoffrey Hinton, et al. *Learning multiple layers of features from tiny images*. Tech. rep. University of Toronto, Canada, 2009.

[161] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Proc. Conf. NeruIPS*. 2012, pp. 1097–1105.

[162] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[163] Anders Krogh and John A Hertz. "A simple weight decay can improve generalization". In: *Advances in neural information processing systems*. 1992, pp. 950–957.

[164] J Richard Landis and Gary G Koch. "The measurement of observer agreement for categorical data". In: *biometrics* (1977), pp. 159–174.

[165] Loic Landrieu and Martin Simonovsky. "Large-scale point cloud semantic segmentation with superpoint graphs". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2018, pp. 4558–4567.

[166] Quoc V Le et al. "ICA with reconstruction cost for efficient overcomplete feature learning". In: *Advances in Neural Information Processing Systems (NIPS)*. 2011, pp. 1017–1025.

[167] Yann LeCun, John S Denker, and Sara A Solla. "Optimal brain damage". In: *Advances in neural information processing systems*. 1990, pp. 598–605.

[168] Yann LeCun et al. "A theoretical framework for back-propagation". In: *Proceedings of the 1988 connectionist models summer school*. Vol. 1. 1988, pp. 21–28.

[169] Jianjun Lei et al. "Semi-heterogeneous three-way joint embedding network for sketch-based image retrieval". In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.9 (2019), pp. 3226–3237.

[170] Changjian Li et al. "BendSketch: modeling freeform surfaces through 2D sketching". In: *ACM Transactions on Graphics* 36.4 (2017), pp. 1–14.

[171] Changjian Li et al. "Robust flow-guided neural prediction for sketch-based freeform surface modeling". In: *ACM Transactions on Graphics (TOG)* 37.6 (2018), pp. 1–12.

[172] Jiaxin Li, Ben M Chen, and Gim Hee Lee. "So-net: Self-organizing network for point cloud analysis". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2018, pp. 9397–9406.

[173] Ke Li, Bharath Hariharan, and Jitendra Malik. "Iterative instance segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3659–3667.

[174] Peiliang Li, Tong Qin, et al. "Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving". In: *Proc. EUR Conf. Comput. Vis.* 2018, pp. 646–661.

[175] Shuang Li et al. "Diversity regularized spatiotemporal attention for video-based person re-identification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 369–378.

[176] Yangyan Li et al. "PointCNN: Convolution on X-transformed points". In: *Proc. Conf. NeurIPS*. 2018, pp. 820–830.

[177] JI Liang, Jim Piper, and Jing-Yan Tang. "Erosion and dilation of binary images by arbitrary structuring elements using interval coding". In: *Pattern Recognition Letters* 9.3 (1989), pp. 201–209.

[178] Cheng-Hung Lin et al. "A novel campus navigation APP with augmented reality and deep learning". In: *Proc. Int. Conf. App. Sys. Invent.* 2018, pp. 1075–1077.

[179] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755.

[180] Geert Litjens et al. "A survey on deep learning in medical image analysis". In: *Medical image analysis* 42 (2017), pp. 60–88.

[181] Runtao Liu, Qian Yu, and Stella Yu. "An unpaired sketch-to-photo translation model". In: *arXiv preprint arXiv:1909.08313* (2019).

[182] Shu Liu et al. "Path aggregation network for instance segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8759–8768.

[183] Yun Liu et al. "Detecting cancer metastases on gigapixel pathology images". In: *arXiv preprint arXiv:1703.02442* (2017).

[184] Zhuang Liu et al. "Rethinking the Value of Network Pruning". In: *International Conference on Learning Representations*. 2018.

[185] Ziwei Liu et al. "Large-scale long-tailed recognition in an open world". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 2537–2546.

[186] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[187] Eyal Lotan et al. "Medical imaging and privacy in the era of artificial intelligence: myth, fallacy, and the future". In: *Journal of the American College of Radiology* 17.9 (2020), pp. 1159–1162.

[188] C Louizos et al. "Relaxed quantization for discretized neural networks". In: *International Conference on Learning Representations*. International Conference on Learning Representations, ICLR. 2019.

[189] Zhaoliang Lun et al. "3d shape reconstruction from sketches via multi-view convolutional networks". In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 67–77.

[190] Yi Ma et al. *An invitation to 3-d vision: from images to geometric models*. Vol. 26. Springer Science & Business Media, 2012.

[191] Jitendra Malik et al. "The three R's of computer vision: Recognition, reconstruction and reorganization". In: *Pattern Recognition Letters* 72 (2016), pp. 4–14.

[192] Priyanka Mandikal and Venkatesh Babu Radhakrishnan. "Dense 3d point cloud reconstruction using a deep pyramid network". In: *2019 IEEE Winter Conference on Applications of Computer Vision*. IEEE. 2019.

[193] Priyanka Mandikal et al. "3D-LMNet: Latent embedding matching for accurate and diverse 3D point cloud reconstruction from a single image". In: *arXiv preprint arXiv:1807.07796* (2018).

[194] Maria Markoulli et al. "Imaging the tear film: a comparison between the subjective keeler tearscope-plus™ and the objective oculus® keratograph 5M and LipiView® interferometer". In: *Current eye research* 43.2 (2018), pp. 155–162.

[195] David Marr. "Vision: A computational investigation into the human representation and processing of visual information, henry holt and co". In: *Inc., New York, NY* 2.4.2 (1982).

[196] Libor Masek et al. "Recognition of human iris patterns for biometric identification". In: (2003).

[197] William D Mathers et al. "Meibomian gland morphology and tear osmolarity: changes with Accutane therapy". In: *Cornea* 10.4 (1991), pp. 286–290.

[198] Daniel Maturana and Sebastian Scherer. "Voxnet: A 3d convolutional neural network for real-time object recognition". In: *Proc. Int. Conf. Intell. Rob. Sys.* 2015, pp. 922–928.

[199] Roey Mechrez et al. "Maintaining natural image statistics with the contextual loss". In: *Asian Conference on Computer Vision*. Springer. 2018, pp. 427–443.

[200] Zhongqi Miao et al. "Insights and approaches using deep learning to classify wildlife". In: *Scientific reports* 9.1 (2019), pp. 1–9.

[201] Ben Mildenhall et al. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *arXiv preprint arXiv:2003.08934* (2020).

[202] Dmytro Mishkin and Jiri Matas. "All you need is a good init". In: *ICLR*. 2016.

[203] Takeru Miyato et al. "Spectral Normalization for Generative Adversarial Networks". In: *ICLR*. 2018.

[204] Volodymyr Mnih et al. "Recurrent Models of Visual Attention". In: *Advances in Neural Information Processing Systems*. 2014.

[205] Kaichun Mo et al. "Structurenet: Hierarchical graph networks for 3d shape generation". In: *ACM Transactions on Graphics* 38.6 (2019).

[206] Yair Movshovitz-Attias et al. "No fuss distance metric learning using proxies". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 360–368.

[207] Michael C Mozer and Paul Smolensky. "Using relevance to reduce network size automatically". In: *Connection Science* 1.1 (1989), pp. 3–16.

[208] Siva Karthik Mustikovela et al. "Self-Supervised Viewpoint Learning From Image Collections". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 3971–3981.

[209] Sauradip Nag, Saptakatha Adak, and Sukhendu Das. "What's there in the dark". In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 2996–3000.

[210] KL Navaneet et al. "CAPNet: Continuous approximation projection for 3D point cloud reconstruction using 2D supervision". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 8819–8826.

[211] Jagadish Nayak et al. "Automated identification of diabetic retinopathy stages using digital fundus images". In: *Journal of medical systems* 32 (2008), pp. 107–115.

[212] John F Nettrour, M Benjamin Burch, and B Sonny Bal. "Patients, pictures, and privacy: managing clinical photographs in the smartphone era". In: *Arthroplasty Today* 5.1 (2019), pp. 57–60.

[213] Alejandro Newell, Kaiyu Yang, and Jia Deng. "Stacked hourglass networks for human pose estimation". In: *European conference on computer vision*. Springer. 2016, pp. 483–499.

[214] Thu Nguyen-Phuoc et al. "Hologan: Unsupervised learning of 3d representations from natural images". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 7588–7597.

[215] Jason J Nichols et al. "An assessment of grading scales for meibography images". In: *Cornea* 24.4 (2005), pp. 382–388.

[216] Steven J. Nowlan and Geoffrey E. Hinton. "Simplifying Neural Networks by Soft Weight-Sharing". In: *Neural Computation* 4.4 (1992), pp. 473–493.

[217] Naoki Nozawa et al. "Single Sketch Image based 3D Car Shape Reconstruction with Deep Learning and Lazy Learning." In: *VISIGRAPP (1: GRAPP)*. 2020, pp. 179–190.

[218] TB Olaf Ronneberger, Philipp Fischer, and Computer. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: ().

[219] Georg Ostrovski, Will Dabney, and Remi Munos. "Autoregressive Quantile Networks for Generative Modeling". In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2018, pp. 3936–3945.

[220] Mete Ozay and Takayuki Okatani. "Optimization on submanifolds of convolution kernels in cnns". In: *arXiv preprint arXiv:1610.07008* (2016).

[221] Umut Ozertem and Deniz Erdogmus. "Locally defined principal curves and surfaces". In: *The Journal of Machine Learning Research* 12 (2011), pp. 1249–1286.

[222] Onur Özyeşil et al. "A survey of structure from motion*." In: *Acta Numerica* 26 (2017), pp. 305–364.

[223] Nikhil R Pal and Sankar K Pal. "A review on image segmentation techniques". In: *Pattern recognition* 26.9 (1993), pp. 1277–1294.

[224] Junyi Pan et al. "Deep Mesh Reconstruction from Single RGB Images via Topology Modification Networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

[225] Eric B Papas. "The global prevalence of dry eye disease: A Bayesian view". In: *Ophthalmic and Physiological Optics* 41.6 (2021), pp. 1254–1266.

[226] Vardan Papyan et al. "Convolutional Dictionary Learning via Local Processing". In: *ICCV*. 2017.

[227] E Akçan Parlaz et al. "Ergenlik dönemi: fiziksel büyüme, psikolojik ve sosyal gelişim süreci". In: *Turkish Family Physician* 3.2 (2012), pp. 10–16.

[228] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: *ICML*. 2013, pp. 1310–1318.

[229] Adam Paszke et al. "Automatic differentiation in pytorch". In: (2017).

[230] Chunlei Peng et al. "Superpixel-based face sketch–photo synthesis". In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.2 (2015), pp. 288–299.

[231] Stephen C Pflugfelder et al. "Evaluation of subjective assessments and objective diagnostic tests for diagnosing tear-film disorders known to cause ocular irritation". In: *Cornea* 17.1 (1998), p. 38.

[232] Xavier Soria Poma, Edgar Riba, and Angel Sappa. "Dense extreme inception network: Towards a robust cnn model for edge detection". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 1923–1932.

[233] Sahana M Prabhu et al. "Deep learning segmentation and quantification of Meibomian glands". In: *Biomedical signal processing and control* 57 (2020), p. 101776.

[234] W Nicholson Price and I Glenn Cohen. "Privacy in the age of medical big data". In: *Nature medicine* 25.1 (2019), pp. 37–43.

[235] Heiko Pult and Jason J Nichols. "A review of meibography". In: *Optometry and Vision Science* 89.5 (2012), E760–E769.

[236] Heiko Pult and Britta Riede-Pult. "Comparison of subjective grading and objective assessment in meibography". In: *Contact Lens and Anterior Eye* 36.1 (2013), pp. 22–27.

[237] Heiko Pult and Britta H Riede-Pult. "Non-contact meibography in diagnosis and treatment of non-obvious meibomian gland dysfunction". In: *Journal of Optometry* 5.1 (2012), pp. 2–5.

[238] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 1.2 (2017), p. 4.

[239] Charles R Qi et al. "Volumetric and multi-view cnns for object classification on 3d data". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2016, pp. 5648–5656.

[240] Charles Ruizhongtai Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[241] Charles Ruizhongtai Qi et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Proc. Conf. NeurIPS*. 2017, pp. 5099–5108.

[242] Varun Ramakrishna et al. "Pose machines: Articulated pose estimation via inference machines". In: *European Conference on Computer Vision*. Springer. 2014, pp. 33–47.

[243] Michael Ramamonjisoa and Vincent Lepetit. "SharpNet: Fast and Accurate Recovery of Occluding Contours in Monocular Depth Estimation". In: *The IEEE International Conference on Computer Vision (ICCV) Workshops* (2019).

[244] Jason R Rambach et al. "Learning to fuse: A deep learning approach to visual-inertial camera pose estimation". In: *Proc. Int. Symp. Mix. & Aug. Real.* 2016, pp. 71–76.

[245] Mohammad Rastegari et al. "Xnor-net: Imagenet classification using binary convolutional neural networks". In: *European conference on computer vision*. Springer. 2016, pp. 525–542.

[246] Christopher D Reinkemeier, Gemma Estrada Girona, and Edward A Lemke. "Designer membraneless organelles enable codon reassignment of selected mRNAs in eukaryotes". In: *Science* 363.6434 (2019), eaaw2644.

[247] Dario Rethage et al. "Fully-convolutional point networks for large-scale point clouds". In: *Proc. EUR Conf. Comput. Vis.* 2018, pp. 596–611.

[248] Stephan R Richter and Stefan Roth. "Discriminative shape from shading in uncalibrated illumination". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[249] Stephan R Richter et al. "Playing for data: Ground truth from computer games". In: *European conference on computer vision*. Springer. 2016, pp. 102–118.

[250] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. "Octnet: Learning deep 3d representations at high resolutions". In: *Proc. Conf. Comput. Vis. Pat. Recog.* Vol. 3. 2017.

[251] Pau Rodriguez et al. "Regularizing CNNs with Locally Constrained Decorrelations". In: *ICLR*. 2017.

[252] Bernardino Romera-Paredes and Philip Hilaire Sean Torr. "Recurrent instance segmentation". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*. Springer. 2016, pp. 312–329.

[253] Fahreddin Sadikoglu and Selin Uzelaltinbulat. "Biometric retina identification based on neural network". In: *Procedia Computer Science* 102 (2016), pp. 26–33.

[254] Tim Salimans and Durk P Kingma. "Weight normalization: A simple reparameterization to accelerate training of deep neural networks". In: *Advances in Neural Information Processing Systems (NIPS)*. 2016, pp. 901–909.

[255] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[256] Patsorn Sangkloy et al. "The sketchy database: learning to retrieve badly drawn bunnies". In: *ACM Transactions on Graphics* 35.4 (2016), pp. 1–12.

[257] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks". In: *ICLR*. 2014.

[258] Scott Schaefer, Travis McPhail, and Joe Warren. "Image deformation using moving least squares". In: *ACM SIGGRAPH 2006 Papers*. 2006, pp. 533–540.

[259] Johannes Schindelin et al. "Fiji: an open-source platform for biological-image analysis". In: *Nature methods* 9.7 (2012), pp. 676–682.

[260] Hanie Sedghi, Vineet Gupta, and Philip M. Long. "The Singular Values of Convolutional Layers". In: *ICLR*. 2019.

[261] Adil Hussain Seh et al. "Healthcare data breaches: insights and implications". In: *Healthcare*. Vol. 8. 2. MDPI. 2020, p. 133.

[262] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[263] Li Shen, Zhouchen Lin, and Qingming Huang. "Relay backpropagation for effective learning of deep convolutional neural networks". In: *European conference on computer vision*. Springer. 2016.

[264] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. "Semantic texton forests for image categorization and segmentation". In: *2008 IEEE conference on computer vision and pattern recognition*. IEEE. 2008, pp. 1–8.

[265] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *ICLR*. 2015.

[266] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. "Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6490–6499.

[267] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. "Scene representation networks: Continuous 3d-structure-aware neural scene representations". In: *Advances in Neural Information Processing Systems*. 2019, pp. 1121–1132.

[268] Vincent Sitzmann et al. "Deepvoxels: Learning persistent 3d feature embeddings". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2437–2446.

[269] Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning". In: *Advances in neural information processing systems* 30 (2017).

[270] J Springenberg et al. "Striving for Simplicity: The All Convolutional Net". In: *ICLR (workshop track)*. 2015.

[271] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[272] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway networks". In: *arXiv preprint arXiv:1505.00387* (2015).

[273] Kenneth O Stanley. "Compositional pattern producing networks: A novel abstraction of development". In: *Genetic programming and evolvable machines* 8.2 (2007), pp. 131–162.

[274] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. "A hypercube-based encoding for evolving large-scale neural networks". In: *Artificial life* 15.2 (2009), pp. 185–212.

[275] Vitor T Stuani et al. "Photogrammetry as an alternative for acquiring digital dental models: A proof of concept". In: *Medical hypotheses* 128 (2019), pp. 43–49.

[276] Paul Sturgess et al. "Combining appearance and structure from motion features for road scene understanding". In: *BMVC-British Machine Vision Conference*. BMVA. 2009.

[277] Hang Su et al. "Multi-view convolutional neural networks for 3d shape recognition". In: *Proc. Int. Conf. Comput. Vis.* 2015, pp. 945–953.

[278] Hang Su et al. "Splatnet: Sparse lattice networks for point cloud processing". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2018, pp. 2530–2539.

[279] Wenlong Sun, Olfa Nasraoui, and Patrick Shafto. "Evolution and impact of bias in human and machine learning algorithm interaction". In: *Plos one* 15.8 (2020), e0235502.

[280] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[281] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International Conference on Machine Learning*. 2019, pp. 6105–6114.

[282] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. "Multi-view 3d models from single images with a convolutional network". In: *European Conference on Computer Vision*. Springer. 2016, pp. 322–337.

[283] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. "Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs". In: *Proc. Int. Conf. Comput. Vis.* 2017, pp. 2088–2096.

[284] Maxim Tatarchenko et al. "Tangent convolutions for dense prediction in 3d". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2018, pp. 3887–3896.

[285] WM Team. "Learning Deep Convolutional Neural Networks for Places2 Scene Recognition". In: ().

[286] Calesta Hui Yi Teo et al. "Meibomian gland dysfunction is the primary determinant of dry eye symptoms: Analysis of 2346 patients". In: *The Ocular Surface* 18.4 (2020), pp. 604–612.

[287] Hugues Thomas et al. "Kpconv: Flexible and deformable convolution for point clouds". In: *Proc. Int. Conf. Comput. Vis.* 2019, pp. 6411–6420.

[288] Cheung CY TingDSW, G Lim, et al. "Developmentand validationofadeeplearningsystemfordiabeticretinopathy andrelatedeyediseasesusingretinalimagesfrom multiethnic populationswithdiabetes". In: *Jama* 318.22 (2017), pp. 2211–2223.

[289] Gabriella Tognola et al. "3-D acquisition and quantitative measurements of anatomical parts by optical scanning and image reconstruction from unorganized range data". In: *IEEE Transactions on Instrumentation and measurement* 52.5 (2003), pp. 1665–1673.

[290] Nina Tran, Andrew D Graham, and Meng C Lin. "Ethnic differences in dry eye symptoms: effects of corneal staining and length of contact lens wear". In: *Contact Lens and Anterior Eye* 36.6 (2013), pp. 281–288.

[291] Shubham Tulsiani et al. "Factoring shape, pose, and layout from the 2D image of a 3D scene". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2018, pp. 302–310.

[292] Shubham Tulsiani et al. "Multi-view supervision for single-view reconstruction via differentiable ray consistency". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2626–2634.

[293] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Deep Image Prior". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[294] Aaron Van den Oord et al. "Conditional image generation with pixelcnn decoders". In: *Advances in Neural Information Processing Systems (NIPS)*. 2016, pp. 4790–4798.

[295] Subeesh Vasu, Mahesh Mohan MR, and AN Rajagopalan. "Occlusion-Aware Rolling Shutter Rectification of 3D Scenes". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2018, pp. 636–645.

[296] Francesco Visin et al. "Reseg: A recurrent neural network-based model for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2016, pp. 41–48.

[297] Eugene Vorontsov et al. "On orthogonality and learning recurrent networks with long term dependencies". In: *ICML*. 2017, pp. 3570–3578.

[298] Alvin Wan et al. "Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 12965–12974.

[299] Li Wan et al. "Regularization of neural networks using dropconnect". In: *International conference on machine learning*. PMLR. 2013, pp. 1058–1066.

[300] Fang Wang, Le Kang, and Yi Li. "Sketch-based 3d shape retrieval using convolutional neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[301] Haofan Wang et al. "Score-CAM: Score-weighted visual explanations for convolutional neural networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 24–25.

[302] Jiayun Wang et al. "A deep learning approach for meibomian gland atrophy evaluation in meibography images". In: *Translational vision science & technology* 8.6 (2019), pp. 37–37.

[303] Jiayun Wang et al. "Compact and Optimal Deep Learning with Recurrent Parameter Generators". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2023).

[304] Jiayun Wang et al. "Orthogonal convolutional neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11505–11515.

[305] Jiayun Wang et al. "Predicting demographics from meibography using deep learning". In: *Scientific Reports* 12.1 (2022), p. 15701.

[306] Jiayun Wang et al. "Quantifying meibomian gland morphology using artificial intelligence". In: *Optometry and Vision Science* 98.9 (2021), pp. 1094–1103.

[307] Jiayun Wang et al. "Unsupervised scene sketch to photo synthesis". In: *Proceedings of the European Conference on Computer Vision Workshop*. 2022.

[308] Lingjing Wang et al. "Unsupervised learning of 3d model reconstruction from hand-drawn sketches". In: *Proceedings of the 26th ACM international conference on Multimedia*. 2018, pp. 1820–1828.

[309] Luo Wang et al. "Sketch-based image retrieval with multi-clustering re-ranking". In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.12 (2019), pp. 4929–4943.

[310] Michael TM Wang and Jennifer P Craig. "Natural history of dry eye disease: Perspectives from inter-ethnic comparison studies". In: *The Ocular Surface* 17.3 (2019), pp. 424–433.

[311] Nannan Wang et al. "Anchored neighborhood index for face sketch synthesis". In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.9 (2017), pp. 2154–2163.

[312] Nanyang Wang et al. "Pixel2mesh: Generating 3d mesh models from single rgb images". In: *Proceedings of the European Conference on Computer Vision*. 2018.

[313] Peng-Shuai Wang et al. "O-cnn: Octree-based convolutional neural networks for 3d shape analysis". In: *ACM Transactions on Graphics* 36.4 (2017), p. 72.

[314] Shenlong Wang et al. "Deep parametric continuous convolutional neural networks". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2018, pp. 2589–2597.

[315] Tiancai Wang, Xiangyu Zhang, and Jian Sun. "Implicit Feature Pyramid Network for Object Detection". In: *arXiv preprint arXiv:2012.13563* (2020).

[316] Ting-Chun Wang et al. "High-resolution image synthesis and semantic manipulation with conditional gans". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8798–8807.

[317] Yue Wang et al. "Dynamic graph cnn for learning on point clouds". In: *ACM Transactions On Graphics* 38.5 (2019), pp. 1–12.

[318] Shih-En Wei et al. "Convolutional pose machines". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2016, pp. 4724–4732.

[319] David Weiss and Benjamin Taskar. "Structured prediction cascades". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 916–923.

[320] P. Welinder et al. *Caltech-UCSD Birds 200*. Tech. rep. CNS-TR-2010-001. California Institute of Technology, 2010.

[321] Olivia Wiles and Andrew Zisserman. "Silnet: Single-and multi-view reconstruction by learning from silhouettes". In: *arXiv preprint arXiv:1711.07888* (2017).

[322] Scott Wisdom et al. "Full-capacity unitary recurrent neural networks". In: *Advances in Neural Information Processing Systems (NIPS)*. 2016, pp. 4880–4888.

[323] Andrew P Witkin. "Recovering surface shape and orientation from texture". In: *Artificial intelligence* 17.1-3 (1981), pp. 17–45.

[324] David H Wolpert. "Stacked generalization". In: *Neural networks* 5.2 (1992), pp. 241–259.

[325] Bichen Wu et al. "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud". In: *Proc. Int. Conf. Rob. & Auto.* 2018, pp. 1887–1893.

[326] Jiajun Wu et al. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling". In: *Advances in neural information processing systems*. 2016.

[327] Zhirong Wu et al. "3d shapenets: A deep representation for volumetric shapes". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2015, pp. 1912–1920.

[328] Dean Wyatte, Tim Curran, and Randall O'Reilly. "The limits of feedforward vision: Recurrent processing promotes robust object recognition when objects are degraded". In: *Journal of Cognitive Neuroscience* 24.11 (2012), pp. 2248–2261.

[329] Di Xie, Jiang Xiong, and Shiliang Pu. "All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6176–6185.

[330] Haozhe Xie et al. "Pix2Vox: Context-aware 3D reconstruction from single and multi-view images". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

[331] Shi Xingjian et al. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting". In: *Advances in neural information processing systems*. 2015, pp. 802–810.

[332] Baoxuan Xu et al. "True2Form: 3D Curve Networks from 2D Sketches via Selective Regularization". In: *Transactions on Graphics* 33.4 (2014). DOI: 2601097.2601128.

[333] Peng Xu et al. "Fine-grained instance-level sketch-based video retrieval". In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.5 (2020), pp. 1995–2007.

[334] Xinchen Yan et al. "Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision". In: *Advances in neural information processing systems*. 2016, pp. 1696–1704.

[335] Keiji Yanai, Ryosuke Tanno, and Koichi Okamoto. "Efficient mobile implementation of a cnn-based object recognition system". In: *Proceedings of the International Conference on Multimedia (ICM)*. 2016, pp. 362–366.

[336] Bo Yang et al. "Dense 3D object reconstruction from a single depth view". In: *IEEE transactions on pattern analysis and machine intelligence* 41.12 (2018), pp. 2820–2834.

[337] Guandao Yang et al. "Pointflow: 3d point cloud generation with continuous normalizing flows". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

[338] Shuai Yang et al. "Deep Plastic Surgery: Robust and Controllable Image Editing with Human-Drawn Sketches". In: *arXiv preprint arXiv:2001.02890* (2020).

[339] Zhaohui Yang et al. "Legonet: Efficient convolutional neural networks with lego filters". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7005–7014.

[340] Thao N Yeh and Meng C Lin. "Repeatability of meibomian gland contrast, a potential indicator of meibomian gland function". In: *Cornea* 38.2 (2019), p. 256.

[341] Young-Sik Yoo et al. "Examination of gland dropout detected on infrared meibography by using optical coherence tomography meibography". In: *The ocular surface* 15.1 (2017), pp. 130–138.

[342] Jiahui Yu et al. "Slimmable Neural Networks". In: *International Conference on Learning Representations*. 2018.

[343] Qian Yu et al. "Sketch me that shoe". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[344] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks". In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2016.

[345] Amir R Zamir et al. "Feedback networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1308–1317.

[346] E Zhang and G Cazugue. "B. lLay et al.,"Feedback on a publicly distributed image database: the messidor database,"". In: *Image Analysis and Stereology* 33.3 (2014), p. 231.

[347] Meiqin Zhang et al. "TNFSF15 polymorphisms are associated with Graves' disease and Graves' ophthalmopathy in a Han Chinese population". In: *Current Eye Research* 45.7 (2020), pp. 888–895.

[348] Shengchuan Zhang et al. "Face sketch synthesis from a single photo–sketch pair". In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.2 (2015), pp. 275–287.

[349] Song-Hai Zhang, Yuan-Chen Guo, and Qing-Wen Gu. "Sketch2Model: View-Aware 3D Modeling from Single Free-Hand Sketches". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 6012–6021.

[350] Yang Zhang et al. "RealPoint3D: An efficient generation network for 3D object reconstruction from a single image". In: *IEEE Access* 7 (2019), pp. 57539–57549.

[351] Hengshuang Zhao et al. "Pointweb: Enhancing local neighborhood features for point cloud processing". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2019, pp. 5565–5573.

[352] Hengshuang Zhao et al. "Pyramid scene parsing network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2881–2890.

[353] Heliang Zheng et al. "Learning multi-attention convolutional neural network for fine-grained image recognition". In: *ICCV*. 2017, pp. 5209–5217.

[354] Jianping Zhou, Minh N Do, and Jelena Kovacevic. "Special paraunitary matrices, Cayley transform, and multidimensional orthogonal filter banks". In: *IEEE Transactions on Image Processing* 15.2 (2006), pp. 511–519.

[355] Lei Zhou et al. "Learning and matching multi-view descriptors for registration of point clouds". In: *Proc. EUR Conf. Comput. Vis.* 2018, pp. 505–522.

[356] Tinghui Zhou et al. "View synthesis by appearance flow". In: *European conference on computer vision*. Springer. 2016, pp. 286–301.

[357] Yin Zhou and Oncel Tuzel. "Voxelnet: End-to-end learning for point cloud based 3d object detection". In: *Proc. Conf. Comput. Vis. Pat. Recog.* 2018, pp. 4490–4499.

[358] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017.

[359] Barret Zoph and Quoc V Le. "Neural architecture search with reinforcement learning". In: *ICLR*. 2017.

Appendix A

Supplementary Materials of Meibomian Gland Morphology Quantification

We present detailed data processing procedures, network design, loss functions and training details of the proposed model, Meibomian gland (MG) morphological feature measuring methods, and MG detection evaluation details.

A.1 Data Processing

Upper eyelid total area. To minimize the annotation variability across different annotators, only the visible part of MGs in each image were annotated. As occluded parts of MGs hidden behind the tarsal plate should also be considered, algorithms were designed to extend MG annotations in upper eyelids. Specifically, a principal curve[112] was fit to each (visible) annotated MG region to obtain its linear direction. MG annotations were then extended along the linear principal curve direction until they intersected with the upper borders of annotated eyelid regions, which were defined at the MG orifices. Figure A.1 shows some examples of upper eyelid gland completion.

Center crop. Meibography imaging projects the three-dimensional eyelid surface to a two-dimensional image, with more distortion in the far nasal and temporal image regions. Furthermore, far temporal and nasal glands were less distinguishable and were difficult to annotate accurately due to the inter-observer differences in eyelid eversion techniques that result in MGs not being imaged at all or imaged MGs appearing blurry. Therefore, based on the eyelid region annotations, all glands located within or partially overlapping with the central 50% of the upper eyelid were retained for analysis. The boundaries of the cropped central region are outlined in white in Figure A.1 for visualization. Processed images with annotations and center cropping were verified by a clinical investigator (MCL).

The gland segmentation network took a meibography image as an input and produced

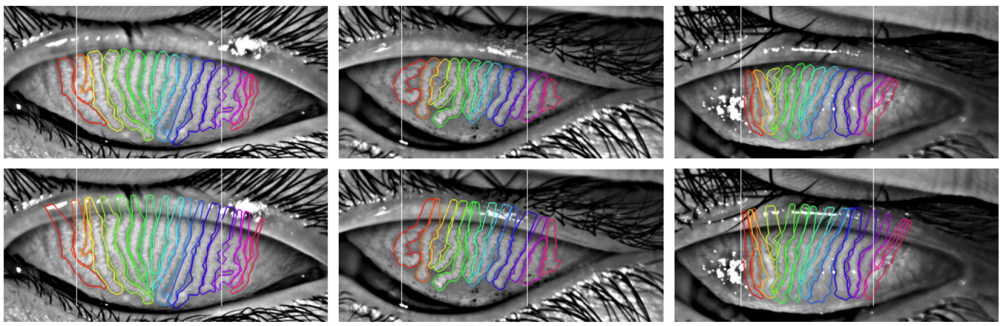


Figure A.1: Processing of MG region annotations: Upper eyelid gland annotation completion and center crop. The first and the second row correspond to original MG region annotations and MG regions after extension, respectively (see the supplementary for details). Each column corresponds to an individual example eyelid. The boundaries of the cropped regions are marked in white. Only imaged MGs falling into the cropped region were retained although all MG region annotations are shown. (a) To avoid annotation variability, only the visible part of each MG was originally annotated. A principal curve was fit to the (visible) annotated MG region to obtain its linear direction. Annotated MG regions were then extended along the linear principal curve direction to the upper border of the annotated eyelid region, which was defined at the MG orifice. (b) Far nasal and temporal gland annotations were not considered in this study due to their ambiguity. Based on the eyelid region annotations, imaged glands falling in the center region (outlined in white) of width equaling to 50% eyelid width were retained while rest were discarded. Glands falling on the boundaries were also retained.

two intermediate outputs: the entire MG region contour (along with pixel-wise feature embedding) and the number of glands. The predicted MG contour, embedding, and gland number were further used by the network to predict MG region instance segmentation.

A.2 Network Design Details

Instance segmentation. MG instance segmentations were generated differently in the training and evaluation phases, since ground-truth segmentation (MG region contours) were not available in the evaluation phase. In the training phase, ground-truth segmentation (with pixel-wise feature embedding) for individual gland regions was available. Loss functions were employed to penalize the difference between network predictions and corresponding ground truths during training, so that networks could learn from the annotations to segment MGs accurately. Further details of the design and implementation of different loss terms can be found in a previous paper [65].

In the evaluation phase, intermediate outputs including pixel-wise embedding and gland number were extracted. K-means clustering was then performed on the embeddings to form different gland clusters. Different clusters corresponded to individual gland instance segmentations.

Ghost gland identification. A deep learning model was designed to identify ghost glands (Figure 3.3). The neural network had three inputs: a whole meibography image, a specific gland region contour, and a corresponding cropped gland image. Inputs consisted of a global view of the meibography image and a local view of individual glands; thus, two subnetworks were used to learn to identify ghost glands. The global network learned a representation from the entire meibography image and the individual gland region contour, while the local network learned a representation from the individual gland image. The two representations were combined to predict ghost glands. A cross-entropy loss was utilized to penalize wrong predictions. Specifically, the numbers of ghost and non-ghost glands were highly imbalanced (Figure 3.10(h) of the paper). The imbalanced data distribution can result in the model having high bias and a focus on learning from the majority. The imbalance in the data distribution leads to inequality in category-wise performance as the model ignores properties of the minority (i.e., ghost glands). To avoid imbalanced category-wise performance [269, 285], class-aware balanced sampling [185] was employed to ensure that each training data batch had the same number of ghost and non-ghost glands and similar classification performance across two categories, so that the model learned from different categories equally.

A.3 Loss Functions for Training the Segmentation Model

To train the MG segmentation model, four loss terms were used: 1) A cross-entropy loss was used to generate an entire MG region contour similar to the ground truth; 2) A regression loss was employed to minimize the difference between the predicted gland number and the actual gland number; 3) A contrastive loss was utilized to reduce intra-gland pixel embedding distances and enlarge inter-gland pixel embedding distances; 4) A regularization loss was incorporated to penalize clusters too far away from the origin to keep the activations bounded. The four loss terms cooperatively produced reasonable MG instance segmentations. Further details of the design and implementation of different loss terms can be found in a previous paper[65].

A.4 Model Training Details

MG instance segmentation. Each meibography image and corresponding MG region segmentations were resized to 262×262 pixels. During training, 256×256 pixels were randomly cropped out of a given meibography image along with the corresponding annotations in every training epoch for data augmentation. A center crop of 256×256 pixels was made to a given meibography image and corresponding annotations during the evaluation process for both validation and evaluation datasets.

Different network architectures (ReSeg networks [296] and stacked recurrent hourglass networks [213]), weights of different loss terms, learning rate, and learning-rate decreasing policy were carefully assessed to obtain the best performance of the network on the validation dataset. The best performance of the network over the validation set was the ReSeg network backbone, cross-entropy loss weight 1.0, regression loss weight 1.0, contrastive loss weight 1.0, regularization loss weight 1e-3, initial learning rate 1.0, 200 epochs in total, with learning rate decrease when the validation loss was not decreasing for 20 epochs. The algorithm performance of the model on the evaluation dataset was reported.

Ghost gland classification. As mentioned above, each gland in a meibography image was augmented to three images: the entire meibography image, the individual gland region contour and individual gland images. The first two images were fed to a global network while the last image was fed to a local network. Two representations from both networks were concatenated and a linear layer was employed to obtain a final vector indicating whether the gland was a ghost gland or not.

Different network architectures (SqueezeNet [135], resnet18, resnet34, resnet50 [113]), balanced sampling strategies, learning rate, and learning-rate decreasing policy were carefully assessed to obtain the best performance of the network on the validation dataset. The best performance of the network over the validation set employed the resnet18 backbone, class-aware balanced sampling strategy [185], initial learning rate 1e-3, 90 epochs in total, with learning rate decrease when the validation loss was not decreasing for 20 epochs.

A.5 Local Morphological Features

Based on individual MG segmentations, local morphological features were quantitively measured. Morphological features included average gland local contrast, gland length, width and tortuosity.

Gland local contrast. Adopting an approach used in previous research [340], the surrounding region of each MG in an image was obtained by extending the gland region border by 10 pixels using binary dilation [177]. Surrounding regions are highlighted in color in Figure 3.4 of the paper for better visualizations. Gland pixels were also excluded if they overlapped with surrounding pixels. The average gland local contrast was defined as the difference in the average intensity of a gland region in the image and its surrounding area with negative values set to 0. Figure 3.4 of the paper demonstrates the process of MG local contrast calculation with examples.

Gland length. Gland length was calculated from the gland region using locally defined principal curves [221]. It is important to note that these curves are different from the linear principal curves used for MG completion where only linear direction was desired. Locally defined principal curves utilize density estimation with gaussian mixture models to represent MG regions, and to appropriately represent the region geometry. For each gland region, corresponding principal curves are shown in Figure 3.5 of the paper. They usually pass through the middle of the gland and fit the geometric shape of each MG. Gland length was

defined as the product of millimeters-per-pixel and the number of pixels of each principal curve. Millimeters-per-pixel resolution in this set of images was 0.0235, which was measured with the image of a ruler captured by the imaging instrument.

Gland width. The average gland width was defined as the gland area (the number of pixels in a MG region) divided by the gland length. The number of pixels was also multiplied by the millimeters-per-pixel conversion factor to obtain gland width in millimeters.

Gland tortuosity. Convexity defect was used to represent the overall tortuosity of a gland. As in Figure 3.6 of the paper, a convex hull (outlined in blue) of a MG region (outlined in red) was found for each MG region in the image. The cavity (marked in green) of a gland is defined as the area belonging to the convex hull area but not the gland region area. The convexity defect is thus defined as the ratio of the cavity area to the convex hull area. A larger convexity defect indicates a more tortuous gland. Convexity defect is referred to as the more commonly used clinical term "tortuosity" for convenience in this study.

A.6 Gland Detection: Evaluation Metrics and Results

Evaluation Metrics. Detection performance is a key indicator of gland instance segmentation performance. Precision and recall were used as a standard metric for gland detection in this study. Specifically, an attempted match was made between each predicted gland region and a ground truth gland region. A match between the prediction and the ground truth was established for the highest mean intersection over union (IU), provided that the mean IU surpassed a threshold of 50%. Unmatched predicted glands were considered to be false positives while unmatched ground truth glands were considered to be false negatives, thereby permitting calculation of precision and recall. More details about evaluation metrics are available in the documentation for the Microsoft Common Objects in Context dataset [179].

Appendix B

Supplementary Materials of Orthogonal Convolutional Networks

The supplementary materials provide intuitive explanations of our approach (Section B.1), network dissection results to understand the change in feature redundancy/expressiveness (Section B.2), deep metric learning performance to understand the generalizability (Section B.3), proof of Lemma 1 (Section B.4) and visualizations of filter similarities (Section B.5).

B.1 Intuitive Explanations of our Approach

We analyze a convolutional layer which transforms input X to output Y with a learnable kernel K: $Y = \text{Conv}(K, X)$ in CNNs. Writing in linear matrix-vector multiplication form $Y = \mathcal{K}X$ (Fig.8.2b of the paper), we simplify the analysis from the perspective of linear systems. We do not use *im2col* form $Y = K\widetilde{X}$ (Fig.8.2a of the paper) as there is an additional structured linear transform from X to $\tilde{X}$ and this additional linear transform makes the analysis indirect. As we mentioned earlier, the kernel orthogonality does not lead to a uniform spectrum.

The spectrum of $\mathcal{K}$ reflects the scaling property of the corresponding convolutional layer: different input X (such as cat, dog, and house images) would scale up by $\eta = \frac{\|Y\|}{\|X\|}$. The scaling factor η also reflects the scaling of the gradient. A typical CNN has highly non-uniform convolution spectrum (Fig.5.1b of the paper): for some inputs, it scales up to 2; for others, it scales by 0.1. For a deep network, these irregular spectra are multiplied together and can potentially lead to gradient exploding and vanishing issues.

Features learned by CNNs are also more redundant due to the non-uniform spectrum issues (Fig.5.1a of the paper). This comes from the diverse learning ability for different images and a uniform spectrum distribution could alleviate the problem. In order to achieve a uniform spectrum, We make convolutions orthogonal by enforcing DBT kernel matrix $\mathcal{K}$ to be orthogonal. The orthogonal convolution regularizer leads to uniform $\mathcal{K}$ spectra as expected. It further reduces feature redundancy and improves the performance (Fig.5.1b,c,d

of the paper).

Besides classification performance improvements, we also observe improved visual feature qualities, both in high-level (image retrieval) and low-level (image inpainting) tasks. Our OCNNs also generate realistic images (Section 8.4.6) and are more robust to adversarial attacks (Section 8.4.7).

B.2 Network Dissection

We demonstrate in Section 5.4 that OCNNs reduce the feature redundancy by decorrelating different feature channels and enhancing the feature expressiveness with improved performance in image retrieval, inpainting, and generation. Network dissection [27] is utilized to further evaluate the feature expressiveness across different channels.

Network dissection [27] is a framework that quantifies the interpretability of latent representations of CNNs by evaluating the alignment between individual hidden units and a set of semantic concepts. Specifically, we evaluate the baseline and our OCNN with backbone architecture ResNet34 [113] trained on ImageNet. Models are evaluated on the Broden [27] dataset, where each image is annotated with spatial regions of different concepts, including cat, dog, house, etc. The concepts are further grouped into 6 categories: scene, object, part, material, texture, and color. Network dissection framework compares the mean intersection over union (mIoU) between network channel-wise activation of each layer and ground-truth annotations. The units/feature channels are considered as "effective" when mIoU$\geq$ 0.04. They are denoted as "unique detectors".

OCNNs have more unique detectors over different layers of the network (Table B.1 & Fig.B.1). Additionally, OCNNs have more uniform distributions of 6 concept categories (Fig.B.2). These imply that orthogonal convolutions reduce feature redundancy and enhance the feature expressiveness.

Table B.1: Number of unique detectors (feature channels with mIoU $\geq$ 0.04) comparisons on ImageNet [67].

	conv2	conv3	conv4	conv5
ResNet34 [113]	6	13	47	346
OCNN (ours)	6	14	57	365

B.3 Deep Metric Learning

We evaluate the generalizability and performance of our our orthogonal regularizer in deep metric learning tasks. Specifically, following the training/evaluation settings in [206], we

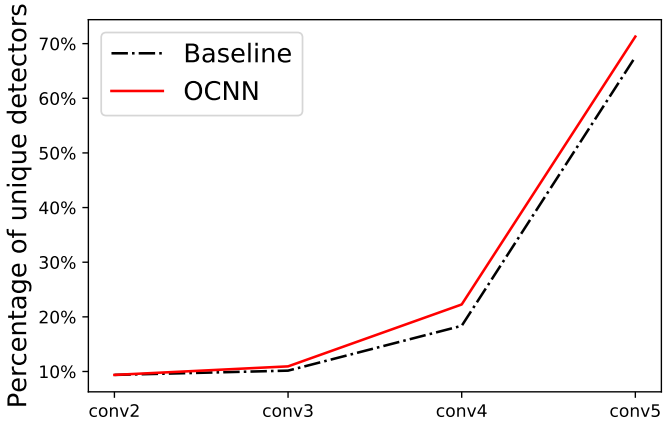


Figure B.1: Percentage of unique detectors (mIoU ≥ 0.04) over different layers. Our OCNN has more unique detectors compared to plain baseline ResNet34 [113] at each layer.

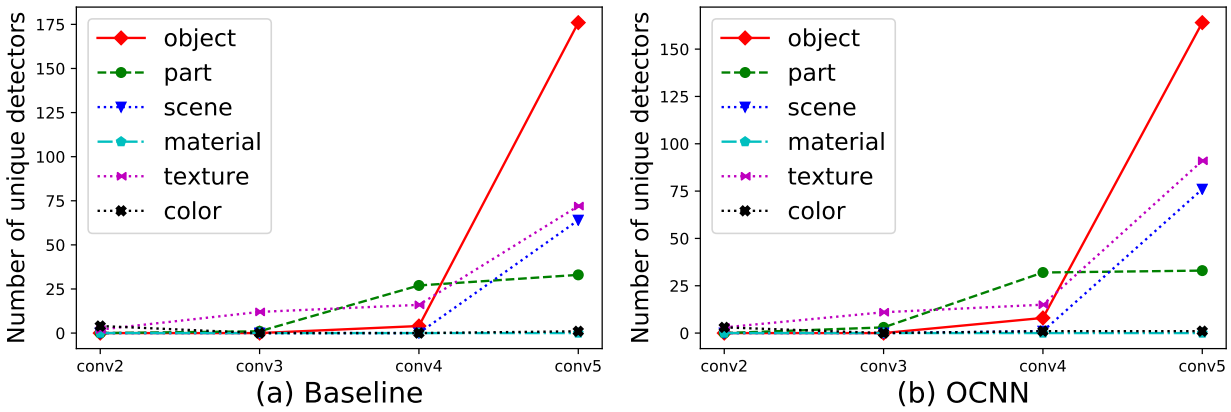


Figure B.2: Distribution of concepts of unique detectors of different layers. Our OCNN has more uniform concept distribution compared to plain baseline ResNet34 [113].

perform retrieval and clustering on Cars196 dataset [159] and summarize the results in Table B.2. We observe performance gains when the orthogonal regularizer is added.

B.4 Proof of the Orthogonality Equivalence

Here we provide a proof for the lemma 1: The row orthogonality and column orthogonality are equivalent in the MSE sense, i.e. $\|\mathcal{K}\mathcal{K}^T - I\|_F^2 = \|\mathcal{K}^T\mathcal{K} - I'\|_F^2 + U$, where U is a constant. A simple motivation for this proof is that when $\mathcal{K}$ is a square matrix, then $\mathcal{K}\mathcal{K}^T = I \iff \mathcal{K}^T\mathcal{K} = I'$. So we can hope to generalize this result and provide a more

Table B.2: Retrieval/clustering performance on Cars196 (%)

	NMI	F1	Recall@1	@2	@4	@8
Triplet loss [124]	61.9	27.1	61.4	73.5	83.1	89.9
ProxyNCA [206]	62.4	29.2	67.9	78.2	85.6	90.6
[206]+Kernel orth	63.1	29.6	67.6	78.4	86.2	91.2
[206]+OCNN	**63.6**	**30.2**	**68.8**	**79.0**	**87.4**	**92.0**

convenient algorithm. The following short proof is provided in the supplementary material of [166]. We would like to present it here for the reader's convenience.

Proof. It's sufficient to prove the general result, where we choose $\mathcal{K} \in \mathbf{R}^{M\times N}$ to be an arbitrary matrix[1]. We denote $\|\mathcal{K}\mathcal{K}^T - I_M\|^2$ as L_r and $\|\mathcal{K}^T\mathcal{K} - I_N\|^2$ as L_c.

$$\begin{aligned}
L_r &= \|\mathcal{K}\mathcal{K}^T - I_M\|_F^2 \\
&= tr\left[(\mathcal{K}\mathcal{K}^T - I_M)^T(\mathcal{K}\mathcal{K}^T - I_M)\right] \\
&= tr(\mathcal{K}\mathcal{K}^T\mathcal{K}\mathcal{K}^T) - 2tr(\mathcal{K}\mathcal{K}^T) + tr(I_M) \\
&= tr(\mathcal{K}^T\mathcal{K}\mathcal{K}^T\mathcal{K}) - 2tr(\mathcal{K}^T\mathcal{K}) + tr(I_N) + M - N \\
&= tr\left[\mathcal{K}^T\mathcal{K}\mathcal{K}^T\mathcal{K} - 2\mathcal{K}^T\mathcal{K} + I_N\right] + M - N \\
&= tr\left[(\mathcal{K}^T\mathcal{K} - I_N)(\mathcal{K}^T\mathcal{K} - I_N)\right] + M - N \\
&= \|\mathcal{K}^T\mathcal{K} - I_N\|_F^2 + M - N \\
&= L_c + U
\end{aligned}$$

where $U = M - N$.

□

B.5 Filter Similarity Visualizations

As shown in Fig.5.1, filter similarity increases with depth of the network. We visualize the guided back-propagation patterns to understand this phenomenon.

For the ResNet34 trained on ImageNet, we plot guided back-propagation patterns of an image in Fig.B.3. The first row depicts patterns of the first 3 channels from layer 7, while the second row depicts patterns of the first 3 channels from layer 33. Patterns of different channels from earlier layers are more diverse, while patterns of different channels from later layers usually focus on certain regions. The filter similarity increases with depth.

[1]Here M and N are just some constant, different from the the ones used in the main text.

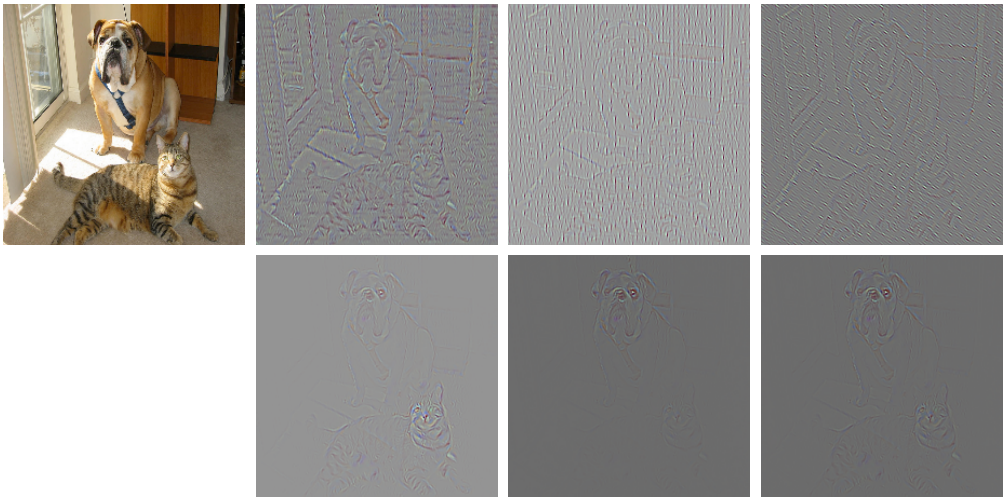


Figure B.3: Guided back-propagation patterns of the input image (first column) with a ResNet34 model. The first row depicts patterns of the first 3 channels from layer 7, while the second row depicts patterns of the first 3 channels from layer 33. The filter similarity increases with network depth.

Appendix C

Supplementary Materials of Recurrent Parameter Generator

We first show RPG networks could be quantized with minimal accuracy drop for compression purpose in Section C.1. We then provide a figure revealing log-linear DoF-accuracy relationship in Section C.2. We also provide proof for the orthogonal proposition in the main paper (Section C.3). Finally, we provide detailed comparison and discussion to a closely related work HyperNetworks [100] in Section C.4.

Additionally, we provide the most important code to reproduce the layer superposition experiments on ImageNet in supplementary as a tgz file. The rest of code is also ready for release, and will be released after additional internal review.

C.1 Quantize RPG

Quantization refers to techniques for performing computations and storing tensors at lower bitwidths than floating point precision. Quantization can reduce model size with tiny accuracy drop. Table C.1 shows that with 8-bit quantization, ResNet18-vanilla has an accuracy drop of 0.3 percentage point, while our ResNet18-RPG has an accuracy drop of 0.1 percentage point. RPG models can be quantized for further model size reduction with a negligible accuracy drop.

Table C.1: RPG model can be quantized with very tiny accuracy drop. With 8-bit quantization on ImageNet, ResNet18-vanilla has an accuracy drop of 0.3 percentage point, while our ResNet18-RPG has an accuracy drop of 0.1 percentage point.

	# Params	Acc before	Acc after ↓ quantization	Acc drop
R18-vanilla	11M	69.8	69.5	0.3
R18-RPG	5.6M	70.2	70.1	0.1

C.2 CIFAR100 Accuracy versus DoF

Fig.C.1 plots CIFAR100 classification accuracy versus model DoF. We observe a similar log-linear relationship as in ImageNet.

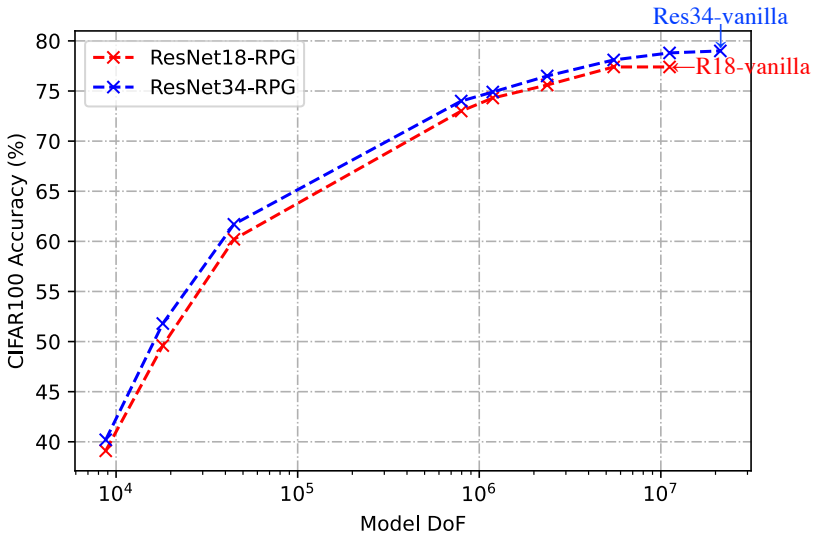


Figure C.1: Log-linear DoF-accuracy relationship of CIFAR100 accuracy and model DoF on CIFAR100. RPG achieves the same accuracy as vanilla ResNet with 50% DoF.

C.3 Proof to the Orthogonal Proposition

We provide proofs to the orthogonal proposition mentioned in Section 9.3 of the main paper. Suppose we have two vectors $\mathbf{f}_i = \boldsymbol{A}_i\mathbf{f}, \mathbf{f}_j = \boldsymbol{A}_i\mathbf{f}$, where $\boldsymbol{A}_i$, $\boldsymbol{A}_j$ are sampled from the $O(M)$ Haar distribution.

Proposition 1. $\mathbb{E}\left[\langle \mathbf{f}_i, \mathbf{f}_j \rangle\right] = 0$.

Proof.

$$\begin{aligned}
\mathbb{E}\left[\langle \mathbf{f}_i, \mathbf{f}_j \rangle\right] &= \mathbb{E}\left[\langle \mathbf{f}_i, \mathbf{f}_j \rangle\right] \\
&= \mathbb{E}\left[\langle \boldsymbol{A}_i\mathbf{f}, \boldsymbol{A}_j\mathbf{f} \rangle\right] \\
&= \mathbb{E}\left[\langle \mathbf{f}, \boldsymbol{A}_i^T\boldsymbol{A}_j\mathbf{f} \rangle\right] \\
&= \mathbf{f}^T\,\mathbb{E}\left[\boldsymbol{A}_i^T\boldsymbol{A}_j\right]\mathbf{f} \\
&= 0
\end{aligned}$$

where $\boldsymbol{A}_i^T\boldsymbol{A}_j$ is equivalently a random sample from $O(M)$ Haar distribution and its expectation is clearly 0. □

Proposition 2. $\mathbb{E}\left[\langle \frac{\mathbf{f}_i}{\|\mathbf{f}_i\|}, \frac{\mathbf{f}_j}{\|\mathbf{f}_j\|}\rangle^2\right] = \frac{1}{M}$.

Proof.

$$\begin{aligned}\mathbb{E}\left[\langle \frac{\mathbf{f}_i}{\|\mathbf{f}_i\|}, \frac{\mathbf{f}_j}{\|\mathbf{f}_j\|}\rangle^2\right] &= \frac{\mathbb{E}\left[\langle \boldsymbol{A}_i\mathbf{f}, \boldsymbol{A}_j\mathbf{f}\rangle^2\right]}{\|\mathbf{f}\|_2^2\|\mathbf{f}\|_2^2} \\ &= \mathbb{E}\left[\langle \boldsymbol{A}\frac{\mathbf{f}}{\|\mathbf{f}\|}, \frac{\mathbf{f}}{\|\mathbf{f}\|}\rangle^2\right], \\ &\quad \text{where } \boldsymbol{A} = \boldsymbol{A}_i^T\boldsymbol{A}_j \sim O(M) \text{ Haar distribution}\end{aligned}$$

Due to the symmetry,

$$= \mathbb{E}\left[\langle \boldsymbol{A}\frac{\mathbf{f}}{\|\mathbf{f}\|}, (1,0,0,\ldots,0)^T\rangle^2\right]$$

Let $\mathbf{g} = \boldsymbol{A}\frac{\mathbf{f}}{\|\mathbf{f}\|}$,

$$\begin{aligned}&= \mathbb{E}\left[g_1^2\right] \\ &= \frac{1}{M}\end{aligned}$$

since $\mathbf{g}$ is a random unit vector and $\mathbb{E}\left[\sum_{k=1}^M g_k^2\right] = \sum_{k=1}^M \mathbb{E}\left[g_k^2\right] = 1$. □

C.4 Comparison to HyperNetworks

HyperNetworks [100] share similarity with RPG as both methods reduce model DoF. Specifically, HyperNetworks rely on learnable modules to generate network parameters. We compare with them and report results in Table C.2. On CIFAR100 with the embedding dimension of 64 and the same model size, HyperNetworks has 68x FLOPs as our RPG, yet 10 percentage points lower than RPG in accuracy.

Table C.2: RPG outperforms HyperNetworks [100] with same DoF on CIFAR100. HyperNetworks has 68x FLOPs as our RPG, yet 10 percentage points lower than RPG in accuracy.

	model DoF	**FLOPs**	**CIFAR100 Acc.**
HyperNet [100]	632k	2.49G	61.3%
RPG	632k	36.7M	71.6%

RPG can be considered as an extreme and minimal version of HyperNetworks, one without a network. However, RPG's unique design and implementation delivers the following advantages over HyperNetworks:

1. HyperNetworks add substantial FLOPs to the network and render it less practical. Given a network architecture, RPG adds minimal to no additional computation, as

the permutation and sign reflection can be efficiently implemented. However, HyperNetworks use a weight generation network to generate the primary network weights. A hypernet mainly uses matrix multiplication and introduces substantial FLOPs. In the table below, we analyze FLOPs of HyperNetwork for ResNet18 with the embedding dimension of 64. FLOPs of a vanilla-Res18 for ImageNet (224 input size) and CIFAR100 (32 input size) are 1.8G and 36.7M, whereas the weight generation part of the HyperNet-Res18 takes 2.45G FLOPs. This means the weight generation FLOPs are 1.4 times of vanilla-Res18 for ImageNet and 67 times of that of CIFAR100. Empirically, we find the training and inference time HyperNet-Res18 is around 70x larger than vanilla-Res18.

2. HyperNetworks do not have an arbitrary DoF (number of reduced parameters). RPG uses a model ring of a size (model DoF) that can be arbitrarily determined. In HyperNetworks, the weight generation network uses the same hyper-weight and requires embedding to be of a certain size so that the matrix multiplication can be used for generating primary network weights. Therefore, the model DoF or reduced number of parameters cannot be arbitrarily determined. In other words, RPG decouples the model DoF (actual parameters) and the network architecture, while HyperNetworks have model DoF and architecture tightly coupled together, a highly restrictive limitation.

3. Weights generated by HyperNetworks may be coupled and not optimized for different layers. HyperNetworks use only one weight generation network parameterized by hyper-weight to generate all primary network weights. This may not be optimal as different layers of the primary network may need different weight generation networks. Additionally, matrix multiplication is used for generating weights, and the generated primary network weights may be coupled. On the other hand, RPG has destructive weight sharing, which improves the network performance by decoupling cross-layer network weights. We will add these results and discussions in the revision to clarify the differences between RPG and HyperNetworks.