UNIVERSITY OF CALIFORNIA SAN DIEGO

Exploring Visual Structures in Deep Representation Learning

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

in

Computer Science

by

Weijian Xu

2022

University of California San Diego

2022

DEDICATION

To my family.

EPIGRAPH

We must know. We shall know.

—David Hilbert

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

It has been a fantastic journey to pursue Ph.D. in the past five years. First and foremost, I would acknowledge and give my sincere thanks to my advisor, Professor Zhuowen Tu, whose guidance, encouragement, and patience are invaluable. His pursuit of fundamental and prominent research has significantly influenced me, and his advice is always constructive and enlightening. I feel incredibly fortunate to have Professor Zhuowen Tu as my advisor.

I would show great gratitude to my co-advisor, Professor Hao Su, for his inspiring discussions on research. Professor Su's thoughts on vision, robotics, and artificial general intelligence are very mind-expanding and deeply impress me. In addition, I would thank Professor Lawrence Saul, especially for his exceptional teaching. It is Professor Saul's probabilistic reasoning and decision-making course that leads me into the marvelous world of machine learning. His academic writing class also provided significant help when I was writing this dissertation. Furthermore, I would thank Professor Xiaolong Wang and Professor Virginia De Sa for their thought-provoking questions and insightful advice. At last, I would thank all my dissertation committee members for their substantial time and efforts.

I feel fortunate to have several internships at Microsoft and Facebook. I have learned a lot in finding unsolved problems, establishing new tasks, and conducting meaningful research from the perspective of industry labs. I want to thank my mentors and managers, Jingdong Wang, Tamara Berg, Alexander Berg, Baoyuan Wang, Shuang Ma, and Ashish Kapoor, for their tremendous support and extensive help during the past summers.

I would thank my colleagues in the mlPC lab, Yifan Xu, Kwonjoon Lee, Justin Lazarow, Gaurav Parmar, Huaijin Wang, Tyler Chang, David Cheung, Fan Fan, for their fruitful collaboration. I would thank Sainan Liu, Zheng Ding, Xiang Zhang, Wenlong Huang, Ke Li, Shijie Wang, Yuan Gao, for their insightful discussions. I would also thank Chen-Yu Lee, Saining Xie, Long Jin, Zeyu Chen, for their substantial help in my research and career. Outside the lab, I would thank my friends at UC San Diego, Zhichao Li, Qiaojun Feng, Pengluo Wang, Songting Xu, for their kindness and assistance, especially during the COVID-19 pandemic.

Finally, I would like to thank my parents for their unconditional love. Whenever I experience downs, they always believe in me, encourage me, and push me back to the right track. My accomplishment and success cannot be achieved without their unwavering support and belief.

Chapter 2 is based on the material "Geometry-Aware End-to-End Skeleton Detection" by Weijian Xu, Gaurav Parmar, and Zhuowen Tu, which appears in British Machine Vision Conference (BMVC) 2019. The dissertation author is the primary investigator and author of this material.

Chapter 3 is based on the material "Attentional Constellation Nets for Few-Shot Learning" by Weijian Xu*, Yifan Xu*, Huaijin Wang*, and Zhuowen Tu, which appears in International Conference on Learning Representations (ICLR) 2021. The dissertation author is the co-primary investigator and author of this material.

Chapter 4 is based on the material "Co-Scale Conv-Attentional Image Transformers" by Weijian Xu*, Yifan Xu*, Tyler Chang, and Zhuowen Tu, which appears in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) 2021. The dissertation author is the co-primary investigator and author of this material.

Chapter 5 is based on the material "Line Segment Detection Using Transformers without Edges" by Yifan Xu*, Weijian Xu*, David Cheung, and Zhuowen Tu, which appears in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2021. The dissertation author is the co-primary investigator and author of this material.

VITA

2016	B. E. in Computer Science, Beihang University
2018	M. S. in Computer Science, University of California San Diego
2022	Ph. D. in Computer Science, University of California San Diego

PUBLICATIONS

Weijian Xu*, Yifan Xu*, Tyler Chang and Zhuowen Tu, "Co-Scale Conv-Attentional Image Transformers", In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. (*equal contribution)

Yifan Xu*, **Weijian Xu***, David Cheung and Zhuowen Tu, "Line Segment Detection Using Transformers without Edges", In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. (*equal contribution)

Weijian Xu*, Yifan Xu*, Huaijin Wang* and Zhuowen Tu, "Attentional Constellation Nets for Few-Shot Learning", *International Conference on Learning Representations (ICLR)*, 2021. (*equal contribution)

Weijian Xu, Gaurav Parmar and Zhuowen Tu, "Geometry-Aware End-to-End Skeleton Detection", *British Machine Vision Conference (BMVC)*, 2019.

ABSTRACT OF THE DISSERTATION

Exploring Visual Structures in Deep Representation Learning

by

Weijian Xu

Doctor of Philosophy in Computer Science

University of California San Diego, 2022

Professor Zhuowen Tu, Chair
Professor Hao Su, Co-Chair

Deep representation learning has dominated almost every task in computer vision and achieves superior performance. In deep representation learning, deep neural networks are trained on massive data to provide rich visual representations. However, general-purpose neural networks are not fully aware of visual structures, which limit their generalizability on specific vision tasks (e.g., skeleton detection and line segment detection) or under particular scenarios (e.g., few-shot settings). To tackle the aforementioned limitation, we find it natural and essential to enhance the deep representation with visual structures.

This dissertation concentrates on three visual structures: geometric structure, part structure, and multi-scale structure. We then present four examples to study these visual structures in deep representation learning. First, we focus on object skeleton detection and introduce geometric structure in objective function design. Second, we encode part structure in a convolutional neural network for the few-shot image classification. Third, we build a generic vision Transformer with a co-scale structure for image recognition and instance-level prediction. Finally, we present a Transformer model with a multi-scale structure in line segment detection.

Chapter 1

Introduction

As humans, our brain receives abundant information from our perception of the world. Notably, the visual perception system in our brain provides the most complex signals: numerous photoreceptor cells on the retina transform lights into electrical impulses, which are subsequently processed by the visual cortex. Being the largest system in human brain, the visual cortex decodes the dispersed impulses into a series of complex and hierachical *visual structures*. The visual structures range from low-level visual cues such as corners and edges to high-level visual concepts like objects [Vis22]. The decoded visual structures are then summarized and eventually enable us to understand the world. As the bridge between raw impulses and advanced visual understanding, visual structures play an extremely crucial role in a vision system.

Following the understanding of human visual perception, computer vision has witnessed rapid progress in recent decades. Inspired by human vision, researchers in computer vision have developed delicate hand-crafted feature descriptors including SIFT [Low04] and HoG [DT05]. These feature descriptors establish concrete visual structures such as edge directions and keypoints before building more complex visual structures with histograms. In addition, the vision models are empowered with better flexibility by the growing progress in machine learning. As a result, the feature descriptors and machine learning techniques lead to a classical learning-based pipeline for vision tasks. This pipeline first extracts rich features from images using hand-craft feature descriptors; it then applies a relatively light model (e.g., logistic regression) for task prediction.

However, the classical learning-based pipeline has limitations. First, feature descriptors require a careful design with abundant domain knowledge, which is difficult. Second, the hand-crafted feature descriptors may not generalize well in extremely diverse scenes (e.g., scenes for image recognition and object detection). Third, the machine learning model in the pipeline has limited capacity in modeling complex feature distributions.

In recent years, *deep representation learning* has gradually addressed the limitations above and dominated the field of computer vision. This learning paradigm aims to obtain learned feature representation within deep neural networks using large-scale visual datasets, which is of great capacity and flexibility to cope with diverse scenarios. However, general-purpose neural networks (e.g., multi-layer perceptrons) typically do not generalize well on vision tasks. It is commonly believed that we still need to incorporate specific domain knowledge into the deep representation learning pipeline design. Such domain knowledge can guide the representation towards our expectations and occasionally mitigate optimization difficulties. This dissertation narrows the domain knowledge to visual structures, which reveals crucial structural information from the visual data. With the visual structures embedded, we can establish more interpretable and better-performing deep learning pipelines. We hope that our results can shed some light on building practical learning-based models for vision applications.

In the following sections, we first introduce a typical deep representation pipeline for vision tasks. We then demonstrate two mainstream categories of deep neural networks on which this dissertation mainly works. Furthermore, we show examples of visual structures to embed into deep representation learning. Finally, we present the overview of this dissertation.

1.1 Deep Representation Learning Pipeline

Deep representation learning usually consists of three main components: Model, data, and objectives. The deep representation is learned within the model by optimizing the objectives on model predictions from *training* data. We further discuss these three components in the following paragraphs.

Model. The most distinctive part of the deep representation learning pipeline lies in its model, that is, deep neural networks. Deep neural networks consist of a stack of linear and non-linear operations, which possesses many parameters and enables the modeling of learnable hierarchical features. In vision tasks, two variants of deep neural networks, convolutional neural networks (CNNs) [LBBH98] and vision Transformers [DBK$^+$21], are widely used. For a specific vision task, we can tailor the deep neural networks to enhance awareness of designated visual structures and achieve better representation.

Data. Deep representation learning usually requires large-scale datasets. For example, image recognition datasets such as ImageNet [DDS$^+$09] have more than 1 million annotated images; object detection and instance segmentation datasets such as COCO [KFF15] include more than 200 thousand data points. The large datasets and their corresponding annotations allow the deep models to learn rich and generalizable features. It is noteworthy that it might be challenging to collect massive data in some fields (e.g., medical imaging). Thus, learning deep representation under data-limited scenarios (e.g. few-shot and zero-shot settings) has also received increasing attention. These data-limited scenarios could benefit more from embedded visual structures in the pipeline since the prior knowledge in visual structures can serve as regularization and mitigate overfitting issues.

Objectives. The objectives in deep representation learning guide the update of predictions and parameters from the model during optimization. Specifically, in supervised scenarios, the objectives typically measure a certain distance between model predictions and ground-truth annotations. For example, the cross-entropy loss for classification tasks implies a Kullback–Leibler divergence; the L_1 loss for regression tasks measures an absolute deviation. In vision tasks, we may further strengthen the objectives by introducing statistical properties or visual structures: focal loss [LGG+17] adapts the standard cross-entropy loss to focus on hard misclassified examples; weighted cross-entropy [XT15] bridges the large gap between the edge and background pixels; weighted Hausdorff distance [RGCD19] reflects the geometric difference between sets of points.

1.2 Deep Neural Networks

1.2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) [LBBH98] are prevalent variants of deep neural networks in computer vision. In CNNs, convolutional layers replace linear layers to preserve translation equivariance and reduce parameters. The translation equivariance is a favorable property for visual data like images, while the parameter reduction eases the optimization of CNN models.

The recent resurgence of CNNs starts from the seminal work AlexNet [KSH12], which achieves unbelievable performance on image classification. Following AlexNet, there is a rapid development of CNN architecture design for classification task, including VGG [SZ15], GoogLeNet [SLJ+15], and ResNet [HZRS16]. It is noteworthy that the learned network from image classification can provide a general visual representation for other tasks. In the meanwhile, the powerful CNNs are gradually applied to a broad range of vision tasks, such as edge detection [XT15], object detection [GDDM14, Gir15, RHGS15, LGG+17], semantic segmentation [LSD15, RFB15, CPK+17], and instance segmentation [HGDG17].

1.2.2 Transformers

Transformers [VSP$^+$17] has achieved great success in natural language processing. Transformers have a stack of encoders and decoders, which operate on sequences of tokens and transform the sequences using attention mechanisms. Specifically, attention mechanisms aggregate all tokens from a sequence with dynamic weights. Thus, Transformers have the superb capability in modeling rich global relations among tokens, which is a key factor of their success in many language tasks [VSP$^+$17, RNSS18, DCLT19].

Lately, Transformers has been employed in computer vision and caught great attention. There are two main directions to adopt Transformers for vision tasks. The first direction applies the Transformer encoders to patch-level image tokens. The Transformer encoders can obtain generic visual feature presentation with better non-local information. Representative works in this direction are ViT [DBK$^+$21], PVT [WXL$^+$21], and Swin [LLC$^+$21]. The second direction primarily uses the Transformer decoders to learn instance-level (query) tokens. Each query token progressively gathers the features of a specific instance during the decoding procedure and finally predicts the properties of this instance. This token-to-instance paradigm significantly simplifies the traditional feature-to-instance pipeline [**?**] that has heavy post-processing steps. This paradigm was firstly introduced to object detection by DETR [CMS$^+$20], and then employed in many other tasks including video instance segmentation [WXW$^+$21], line segment detection [XXCT21], and human pose estimation [LWZ$^+$21].

1.3 Examples of Visual Structures in Deep Representation

This dissertation concentrates on three types of visual structures in deep representation: geometric structures, part structures, and multi-scale structures.

Geometric structures in visual representation refer to geometric objects (e.g., points and lines) and their geometric properties (e.g., the distance between points and shape of objects). The mainline of research using geometric structures focuses on direct geometric object extraction from images. For example, line segment detection [XBW$^+$19], wireframe detection [ZQM19, XBW$^+$19], and vanishing point detection [ZQHM19]. Another line of research introduces awareness of geometric properties for better features or objectives, such as geometry-aware layer in 3D point completion [YRW$^+$21], and weighted Hausdorff distance in object localization [RGCD19]. This second line of research receives less attention, but it is still worth exploring.

Part structures disassemble an object to spatially exclusive components, in which each component is called a *part*. Prior to deep learning era, part structures are broadly used in object detection and recognition [FPZ03, FH05, FGMR09]. In deep representation learning, part structures are extensively studied in fine-grained classification [SR15, PHZ17, GLY19, QLL19] In addition, part structures are increasingly explored in part segmentation [HJL$^+$19], object detection [ZZW$^+$17], video object segmentation [CTH$^+$18], and human pose estimation [KSJ$^+$20].

Multi-scale structures create a hierarchical set of features with decreasing spatial resolutions and expanding contexts. In convolutional neural networks, multi-scale structures have already deeply rooted in backbone architectures [KSH12, SZ15, SLJ$^+$15, HZRS16]. Furthermore, the fusion of multi-scale features provides additional contexts that benefit dense prediction tasks, including edge detection [XT15], object detection [LDG$^+$17], semantic segmentation [CPK$^+$17], and pose estimation [WSC$^+$20]. In contrast, early Transformers in vision [DBK$^+$21, CMS$^+$20, TCD$^+$21] stick to a single-scale design. Recent works [WXL$^+$21, LLC$^+$21] have shifted to multi-scale structures and demonstrated extraordinary performance on multiple vision tasks.

1.4 Overview of This Dissertation

This dissertation consists of two parts to study visual structures in deep representation learning. The first part investigates the geometric and part structures in convolutional neural networks (chapter 2 and 3). The second part investigates the multi-scale structures in Transformers (chapter 4 and 5). We present a brief introduction for each chapter as follows.

Chapter 2 presents an objective function reflecting geometric structures in object skeleton detection. Recent approaches in skeleton detection are based primarily on the holistically-nested edge detector (HED) that is learned in a fundamentally bottom-up fashion by minimizing a pixel-wise cross-entropy loss. Here, we introduce a new objective function inspired by the Hausdorff distance that carries both global and local shape information. This new objective function is made differentiable and employed in an end-to-end neural network framework. On several widely adopted skeleton benchmarks, our method achieves state-of-the-art results under the standard F-measure. This sheds some light towards directly incorporating shape and geometric constraints in an end-to-end fashion for image segmentation and detection problems — a viewpoint that has been mostly neglected in the past.

Chapter 3 introduces a part structure in few-shot image recognition. Specifically, we make an effort to enhance structured features by expanding CNNs with a constellation model, which performs cell feature clustering and encoding with a dense part representation; the relationships among the cell features are further modeled by an attention mechanism. With the additional constellation branch to increase the awareness of object parts, our method is able to attain the advantages of the CNNs while making the overall internal representations more robust in the few-shot learning setting. Our approach attains a significant improvement over the existing methods in few-shot learning on the CIFAR-FS, FC100, and *mini*-ImageNet benchmarks.

Chapter 4 demonstrates an improved multi-scale structure — co-scale mechanism — for vision Transformers. In this work, we present Co-scale conv-attentional image Transformers (CoaT), a Transformer-based image classifier equipped with co-scale and conv-attentional mechanisms. First, the co-scale mechanism maintains the integrity of Transformers' encoder branches at individual scales, while allowing representations learned at different scales to effectively communicate with each other; we design a series of serial and parallel blocks to realize the co-scale mechanism. Second, we devise a conv-attentional mechanism by realizing a relative position embedding formulation in the factorized attention module with an efficient convolution-like implementation. CoaT empowers image Transformers with enriched multi-scale and contextual modeling capabilities. On ImageNet, relatively small CoaT models attain superior classification results compared with similar-sized convolutional neural networks and image/vision Transformers. The effectiveness of CoaT's backbone is also illustrated on object detection and instance segmentation, demonstrating its applicability to downstream computer vision tasks.

Chapter 5 builds a Transformer model with the multi-scale structure in line segment detection. Our method takes advantages of having integrated tokenized queries, a self-attention mechanism, and encoding-decoding strategy within Transformers by skipping standard heuristic designs for the edge element detection and perceptual grouping processes. We equip Transformers with a multi-scale encoder/decoder strategy to perform fine-grained line segment detection under a direct endpoint distance loss. This loss term is particularly suitable for detecting geometric structures such as line segments that are not conveniently represented by the standard bounding box representations. The Transformers learn to gradually refine line segments through layers of self-attention. In our experiments, we show state-of-the-art results on Wireframe and YorkUrban benchmarks.

Chapter 6 concludes the dissertation and provides further discussions.

Part I

Geometric and Part Structures in Convolutional Neural Networks

Chapter 2

Geometry-Aware Skeleton Detection

2.1 Introduction

An object skeleton is a compact visual representation that captures the centerline and the symmetric axis of an object [Blu73]. A wide range of computer vision applications have adopted the skeleton representation in their systems, including pose estimation [WZX17, WRKS16, SFC$^+$11], object segmentation [JCLY17, LGY98], scene text detection [CWRL19], and character recognition [WL18]. It is generally observed that an object skeleton exhibits both global (*e.g.* medial axis [SSDZ99, ZY96, SP08]) and local (*e.g.* continuity, local symmetry, and junctions [EG02, Lin98b]) geometric properties.

Object skeleton extraction is a long standing problem in computer vision. Early approaches [Lin98a, LSD13, JH01, YB04] are based on classic image processing techniques using morphological operators. Learning-based approaches, particularly the ones based on deep convolutional neural networks (CNNs), have become increasingly popular. A number of recent methods [SZJ$^+$16, SZJ$^+$17, KCJ$^+$17, ZSG$^+$18, LKQY18, WXT$^+$19] build their skeleton extraction systems on top of the holistically-nested edge detector (HED) [XT15], making use of a balanced cross-entropy loss within a deeply-supervised [LXG$^+$15] fully convolutional neural networks (FCN) [LSD15] framework. These skeleton extraction algorithms demonstrate various properties focusing on different aspects, *e.g.* scale separation and aggregation [SZJ+16, SZJ+17], structural architecture design [KCJ+17, ZSG+18, LKQY18], and rich intermediate representation [WXT+19].

HED was originally designed to perform end-to-end edge detection. It adopts an image-to-image prediction framework by learning a family of nested edge features beyond image gradients. The weighted cross-entropy loss in HED is customized for edge detection, creating a performance gap when applied to skeleton detection. The pixel-wise cross-entropy loss is most effective for the semantic labeling [CPK+17] and edge detection task [XT15], making dense pixel-wise prediction based primarily on local image contents. A skeleton, however, observes strong geometric properties with structural information capturing the long-range contextual shape information (*e.g.* symmetry). Figure 2.1 shows a typical example where the result by a standard HED-based skeleton detector [ZSG+18] outputs a blurry skeleton for the main body of the elephant. In addition, a pixel-wise loss makes an independence assumption for each pixel, rendering violations to the global and local geometric constraints that are commonly observed for object skeletons. HED-based models learn rich hierarchical edge features, but the learning process is not made geometry-aware explicitly. Existing methods in this domain instead rely on a separate post-processing step [ZSG+18] which often has a limited scope of success and cannot correct large mistakes. Figure 2.1 again shows an failure case where significant false positive and false negative are present, which is difficult to be fixed by a standard post-processing algorithm.

Motivated by the above observations, we develop a new convolutional neural network based skeleton detector (named as *GeoSkeletonNet*) by introducing a geometry-aware objective. Specifically, the training objective (still learned end-to-end) consists of a weighted Hausdorff distance and a geometrically weighted cross-entropy loss, providing the global and local geometric constraints. In addition, an extra patch-based point loss is added to the overall objective in order to employ the local geometric constraints. Our proposed algorithm mitigates the limitation

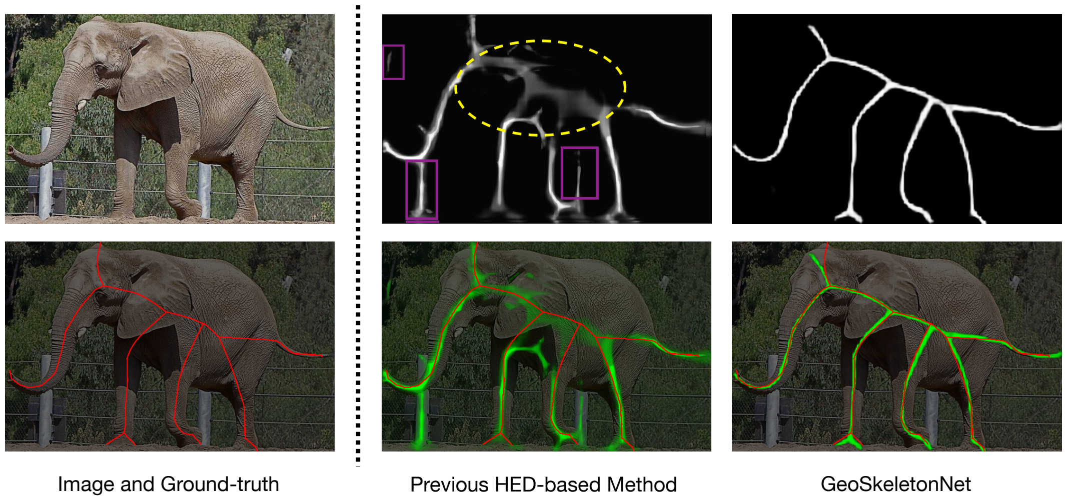


Figure 2.1: Qualitative comparison between a HED-based skeleton detection method [ZSG$^+$18] and our proposed method. Left: an input image and its ground-truth skeleton. Middle: predicted skeleton map (the first row) and a superimposed version with the ground-truth in the input image (the second row); the yellow dashed circle indicates an unsatisfactory area with a blurry effect on elephant's back; purple rectangles show some false positives. Right: predicted skeleton map (the first row) by our method and a superimposed version with the ground-truth in the input image (the second row).

in the existing skeleton detection methods [SZJ+16, SZJ+17, ZSG+18, LKQY18] that do not take into account explicit geometric constraints in training. Figure 2.1 shows the advantage of our geometry-aware framework when compared with a standard learning-based object skeleton extraction system.

The main properties of our GeoSkeletonNet are summarized as follows: (1) we propose to incorporate a geometry-aware objective property within an end-to-end the skeleton detection framework, (2) we adopt a weighted Hausdorff distance and design a geometrically weighted cross-entropy loss, while utilizing a patch-based point loss for local constraints, (3) GeoSkeletonNet demonstrates state-of-the-art results on five skeleton detection benchmarks, outperforming recent competing methods in this domain [SZJ+16, SZJ+17, ZSG+18, LKQY18, KCJ+17, WXT+19].

2.2 Related Work

The task of skeleton detection has been long studied [Blu73], both in computer vision [Lin98a] and medical imaging [SP08]. We refer to a recent paper [WXT+19] for a relative comprehensive discussion about the literature in this subject. Here, we primarily discuss some recent deep learning based skeleton detection algorithms.

Existing Skeleton Detection Methods. Most recent skeleton detection algorithms [SZJ+16, KCJ+17, LKQY18, ZSG+18, WXT+19] are built on top of the holistically-nested edge detector (HED) [XT15]. Shen *et al.* [SZJ+16] propose to enforce the side output of deep supervision with a specific receptive field to match the skeleton at the corresponding scale. Ke *et al.* [KCJ+17] attempt to apply bi-directional residual learning to side outputs, aiming at a larger receptive field. Liu *et al.* [LKQY18] generalize the residual unit in [KCJ+17] by employing the linear span unit, which improves the expressiveness of side outputs. Zhao *et al.* [ZSG+18] design a hierarchical fusion architecture in the deep supervision to further enrich the representation of side outputs. However, all above approaches merely modify the network structure, especially the deeply supervised part, yet still suffer from the side effects of the pixel-wise loss. Recently, Wang *et al.* [WXT$^+$19] propose to change the skeleton prediction from the probability-based map to the flux-based vector field. The flux representation encodes the local geometric relationship between image pixels and skeletal points, but this representation is difficult to learn accurately, which leads to many discontinuities in the predicted skeleton map after post-processing. In contrast, our approach keeps the probability-based skeleton map, while injecting the local and global geometric relation between the prediction and the ground-truth into the objective function, which boosts the overall performance and maintain the visual continuity.

Geometry-Aware Distances in Vision. In computer vision, geometry-aware distances has been widely adopted, especially in object matching [DJ94], face recognition [JKF01] and image retrieval [GWJ$^+$14]. In the deep learning era, geometry-aware distances have been employed in tasks such as 3D object reconstruction [FSG17] and object localization [RGCD19]: Fan *et al.* attempts to build a shape reconstruction framework based on point cloud representation, which minimizes the Chamfer distance and the Earth-mover distance between the prediction and the ground-truth. Ribera *et al.* [RGCD19] proposes to relax the Hausdorff distance and optimize it on the location probability map in the object localization task. Inspired by [RGCD19], we adopt the weighted Hausdorff distance in the objective function for skeleton detection. Besides, we augment the objective with a geometrically weighted cross-entropy loss and a patch-based point loss, which provides additional global and local geometric constraints.

2.3 Method

2.3.1 Problem Formulation

Consider a training dataset $\{(\mathrm{X}_k, \Gamma_k), k = 1, 2, ..., K\}$, where X_k and Γ_k respectively refer to the k-th input image and its corresponding ground-truth skeleton. Note that Γ_k is an explicit vectorized representation encoding the object centerline. K is the total number of training images. In the literature, Γ was represented in various forms, *e.g.* the medial axis [SP08] or the shock graph [SSDZ99] representation. The simplest form of Γ_k can be in a parametric representation $\Gamma_k = (i(s), j(s) : \ s \in [0,1])$ where $(i(s), j(s))$ indicates each 2D skeleton point $(i(s), j(s))$ parameterized by s. For notational simplicity, we drop k by considering only one image X in the training set, $\{(\mathrm{X}, \Gamma)\}$.

To facilitate our training process, skeleton Γ is converted into a label map:
$\mathrm{Y} = (y_{(i,j)}; (i,j) \in \Lambda)$, where $y_{(i,j)} = 1$ if pixel (i,j) is on the skeleton, and $y_{(i,j)} = 0$ otherwise. Thus, our training set is simplified to $\{(\mathrm{X}, \mathrm{Y})\}$, where X refers to the input image and Y denotes the corresponding ground-truth label map. For X, its image lattice that includes all the pixels is defined as $\Lambda = \{(i,j), i = 1..Height, j = 1..Width\}$ where $Height$ and $Width$ refer to the height and width of image X respectively.

2.3.2 Geometry-Aware Objective Function

Revisit the Weighted Cross-Entropy Loss

Given a training image X together with its ground-truth label map Y, our goal is to learn, in an end-to-end fashion, a neural network model to predict $\hat{\mathrm{Y}} = (\hat{y}_{(i,j)}; (i,j) \in \Lambda)$ where $\hat{y}_{(i,j)} \in \{0,1\}$ that is as faithful as possible to the ground-truth $\mathrm{Y} = (y_{(i,j)}; (i,j) \in \Lambda)$. We further define a positive set $Y_+ = \{(i,j); y_{(i,j)} = 1\}$ and a negative set $Y_- = \{(i,j); y_{(i,j)} = 0\}$, to have separate notations for the skeleton and background pixels.

To make the learning process differentiable, the hard prediction map $\hat{\mathrm{Y}}$ is relaxed by a soft probability map $\mathrm{P} = (p_{(i,j)}; (i,j) \in \Lambda)$. Typically, a neural network model can be learned through a pixel-wise cross-entropy loss between the predicted probability map P and the ground-truth Y. Specifically, in [SZJ$^+$16, SZJ$^+$17, KCJ$^+$17, ZSG$^+$18, LKQY18], a weighted cross-entropy

(WC) proposed by [XT15] is used to tackle the problem of an imbalanced dataset:

$$\mathcal{L}_{\mathrm{WC}} = -\beta \sum_{a \in Y_+} \log p_a - (1-\beta) \sum_{a \in Y_-} \log(1-p_a), \tag{2.1}$$

where $\beta = |Y_-|/|\Lambda|$ and $1-\beta = |Y_+|/|\Lambda|$. However, the pixel-wise loss in Equation 2.1 basically evaluates all pixels independently and is absent of explicit geometric constraints, which are important prior for tasks like skeleton extraction. A result obtained by such a loss is illustrated in Figure 2.1, showing problems in localization, precision, and structural consistency.

Weighted Hausdorff Distance

To combat the problem described above, we introduce geometry-aware loss in training to take into account both global and local geometry of the learned skeletons. Following our previous notations, let Y_+ and $\hat{Y}_+$ be the set of skeleton pixels for the ground-truth and prediction respectively. We adopt a Hausdorff distance (HD) to capture the geometric relation between these two sets. For two point sets $\hat{Y}_+$ and Y_+, the Hausdorff distance is computed as:

$$D_{\mathrm{H}} = \max\left(\max_{b \in \hat{Y}_+} \min_{a \in Y_+} d(b,a), \max_{a \in Y_+} \min_{b \in \hat{Y}_+} d(a,b)\right). \tag{2.2}$$

where $d(a,b)$ is the Euclidean distance between point a and b. To increase the robustness of the Hausdorff distance measure against the outliers due to the max operation, a variant of the Hausdorff distance, the average Hausdorff distance (AHD) is adopted:

$$D_{\mathrm{AH}} = \frac{1}{\left|\hat{Y}_+\right|} \sum_{b \in \hat{Y}_+} \min_{a \in Y_+} d(b,a) + \frac{1}{|Y_+|} \sum_{a \in Y_+} \min_{b \in \hat{Y}_+} d(a,b). \tag{2.3}$$

Adding geometric constraints such as Equation 2.2 and 2.3 to a problem that exhibits a strong shape prior seems to be natural step to take. However, making geometry-aware loss in an end-to-end learning framework has been under-explored, due primarily to the difficulty in making the loss differentiable through back-propagation. In this work, we combat this issue by adopting a weighted Hausdorff distance (WHD) that was originally proposed in [RGCD19] for object detection/localization:

$$D_{\mathrm{WH}} = \frac{1}{|\tilde{Y}_+| + \varepsilon} \sum_{b \in \Lambda} p_b \min_{a \in Y_+} d(b,a) + \frac{1}{|Y_+|} \sum_{a \in Y_+} \min_{b \in \Lambda} \frac{d(a,b) + \varepsilon}{(p_b)^{\alpha} + \varepsilon / d_{max}}, \tag{2.4}$$

where $|\tilde{Y}_+| = \sum_{b \in \Lambda} p_b$ is an estimate of the number of positive skeletal points in prediction. ε is a small positive number (*e.g.* 10^{-6}) to avoid zero numerator and denominator and d_{max} is the length of diagonal of the skeleton map. When p_b takes one of the two extreme values $\in \{0, 1\}$, the weighted Hausdorff distance reduces to the average Hausdorff distance [RGCD19]. Note the difference between our method and that in [RGCD19] where the main focus of [RGCD19] is to use a WHD to better localize the object whereas our motivation is to employ WHD as a geometry-aware loss for end-to-end learning of image-to-image prediction.

The weighted Hausdorff distance enjoys the benefit of capturing a shape constraint beyond pixel-wise prediction, encouraging both local and global shape match between the predicted and the ground-truth skeletons. This is a much needed property in the current end-to-end skeleton learning frameworks but remains largely absent in the previous literature.

Patch-based Point Loss

Including the WHD in the objective function reduces the blurry artifacts and makes the predictions better localized. However, directly training on WHD is unstable and disconnected skeleton segments are still present. To address this issue, we add an additional patch-based point loss (PPL) term. PPL aims to minimize the difference between the number of points in P above a specified threshold λ_T and the number of points in Y. To prevent the predicted skeleton from becoming too thick and to enforce local geometric regularities, we apply the proposed point loss in a patch-wise manner.

We divide the image into patches in a grid like manner with the patch size $M \times M$. Each patch coordinate set $\Lambda_{i,j}$ can be represented as a strict subset of Λ such that $\Lambda = \bigcup \Lambda_{i,j}$. $\tilde{p_b}$ represents the probability at the position b if it greater than λ_T, else it is 0. Thus, the patch-based point loss term $\mathcal{L}_{\text{PPL}}$ is formulated as:

$$\mathcal{L}_{\text{PPL}} = \sum_{\Lambda_{i,j}} \left| \sum_{b \in \Lambda_{i,j}} \tilde{p_b} - \left| \Lambda_{i,j} \cap Y_+ \right| \right|. \tag{2.5}$$

Geometrically Weighted Cross-Entropy Loss

Based on the weighted Hausdorff distance in Equation 2.4, we further adapt the weighted cross-entropy in Equation 2.1 to incorporate geometric awareness. For the predicted probabilities corresponding to negative ground-truth points, we scale each pixel-wise cross-entropy term by multiplying with a geometric distance between current point and its nearest positive point in the ground-truth:

$$\mathcal{L}_{\text{GWC}} = -\beta \sum_{a \in Y_+} \log p_a - (1-\beta) \sum_{b \in Y_-} \min_{a \in Y_+} d(a,b)^{\gamma} \log(1-p_b). \tag{2.6}$$

where γ is a hyper-parameter to adjust the effect of distance. The second term in $\mathcal{L}_{\text{GWC}}$ resembles the first term in D_{WH}, which brings similar benefits such as removing unwanted blurs and background noise. In practice, this geometrically weighted cross-entropy loss works significantly well in enhancing the overall performance.

Combining the weighted Hausdorff distance, patch-based point loss and the geometrically weighted cross-entropy loss, the final objective function $\mathcal{L}$ is represented as:

$$\mathcal{L} = \lambda_1 D_{\text{WH}} + \lambda_2 \mathcal{L}_{\text{PPL}} + \lambda_3 \mathcal{L}_{\text{GWC}}. \tag{2.7}$$

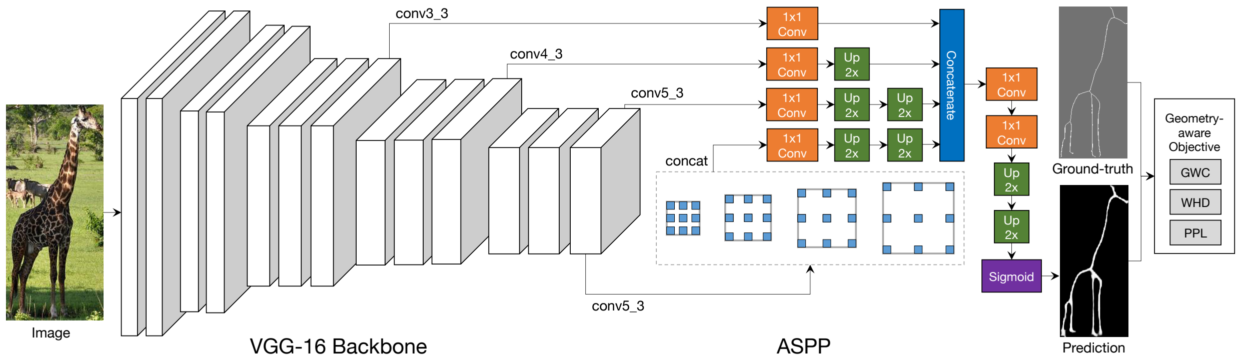


Figure 2.2: Schematic illustration of the network architecture of our GeoSkeletonNet framework.

2.3.3 Network Architecture

Figure 2.2 displays the neural network architecture for our model. We follow the network design from [WXT+19]: The VGG-16 [SZ15] network is used as the feature backbone for fair comparison with other approaches. On top of the last convolutional layer (conv5_3) of VGG network, the atrous spatial pyramid pooling (ASPP) [CPK+17] is applied to enlarge the receptive field. Then, to construct a multi-scale intermediate feature map, we fuse the ASPP output and VGG side outputs (conv3_3, conv4_3, conv5_3) after 1x1 convolutions and bilinear up-sampling kernels. We convert the intermediate feature map to a single channel probability map with original image size as prediction.

2.4 Experiments

2.4.1 Datasets

We evaluate our method on five major datasets for skeleton detection: SK-LARGE [SZJ+17], SK-SMALL [SZJ+16], SYM-PASCAL [KCJ+17], SYMMAX300 [TK12] and WH-SYMMAX [SBHZ16]. Images in SK-LARGE and SK-SMALL are selected from MS COCO dataset [KFF15] with single object, while the ones in SYM-PASCAL may have multiple objects.

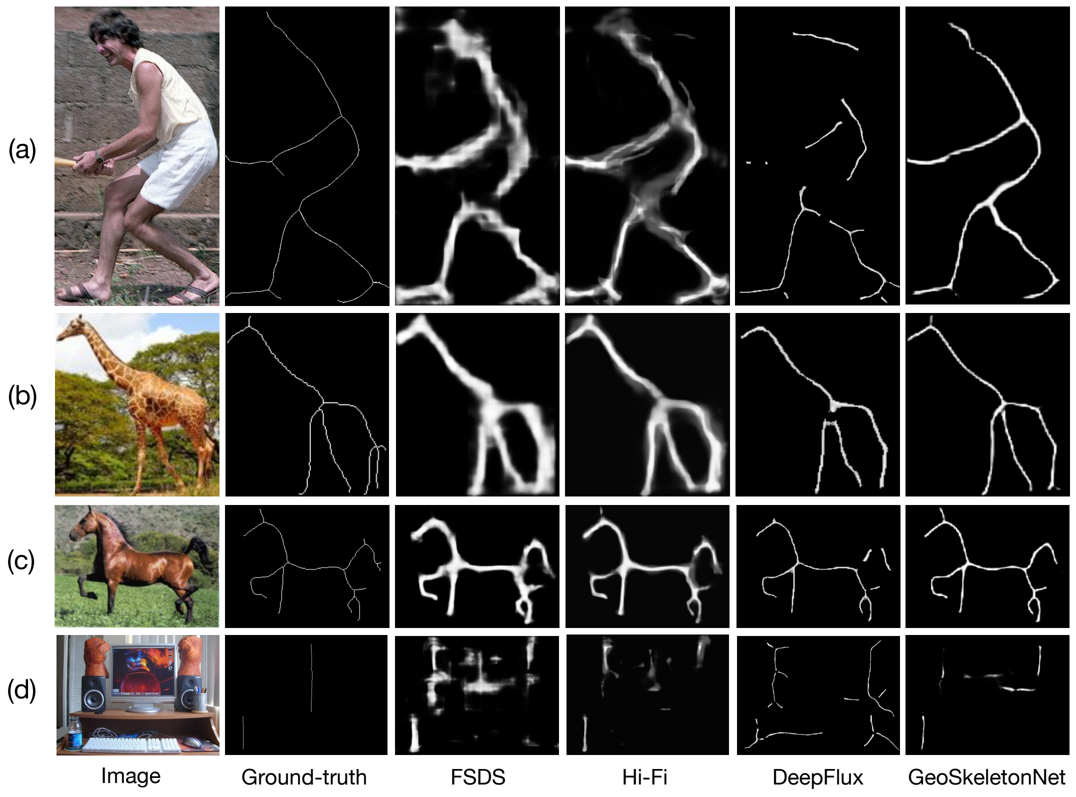


Figure 2.3: Qualitative comparison with the existing methods. We show four examples from four benchmark datasets including (a), (b), (c) and (d) that are from SK-LARGE, SK-SMALL, WH-SYMMAX and SYM-PASCAL respectively. The skeleton maps are the predictions by the competing method and ours, before the non-maximum suppression operation (NMS).

SYMMAX300 and WH-SYMMAX are adapted from the BSDS dataset [MFTM01] and the Weizmann Horse dataset [BU02] respectively.

2.4.2 Evaluation Protocol

PR Curve and F-measure ODS. After obtaining the predicted probability map P from the network, we apply the standard non-maximum suppression (NMS) to P and threshold it by $\delta \in \{0.01, ..., 0.99\}$ to create the actual skeleton map $\hat{Y}_\delta$. We evaluate the performance of the model by the precision-recall (PR) curve of $\hat{Y}_\delta$ in the dataset over all thresholds. In addition, the optimal F-measure on the PR curve, named as F-measure at optimal dataset scale (ODS), is used as evaluation metric as well.

2.4.3 Implementation Details

Training Settings. We build our network and pipeline in PyTorch and run the experiments on NVIDIA TITAN X GPUs. In the experiments, we use the Adam optimizer with the learning rate of 0.0001 and momentum coefficients $\beta_1 = 0.9, \beta_2 = 0.999$. Batch size is set as 1 due to the GPU memory limitation, but we track the *average gradients* every 10 batches and update the weights once, which indicates an equivalent batch size of 10. In the loss terms, we set $\alpha = 4$, $\varepsilon = 10^{-6}$, as recommended by [RGCD19], and $\gamma = 0.5$ due to the ablation study. In $\mathcal{L}_{\text{PPL}}$, the patch size $M = 32$ and the threshold $\lambda_T = 0.95$. Besides, during training, we optimize the model on $\mathcal{L}_{\text{GWC}}$ (i.e. $\lambda_1 = \lambda_2 = 0, \lambda_3 = 1$) for 30 epochs and then fine-tune the model on $D_{\text{WH}} + \mathcal{L}_{\text{PPL}}$ (i.e. $\lambda_3 = 0, \lambda_1 = \lambda_2 = 0.01$) for 5 epochs. The 2-stage procedure leads to a more stable training process.

Resolution Normalization. In most of the skeleton datasets, the images have quite scattered resolutions, which bring a long series of various scales and increase the difficulty in model training. Therefore, in the training stage before data augmentation, we resize the image and ground-truth from size $H \times W$ to $\sqrt{\frac{KH}{W}} \times \sqrt{\frac{KW}{H}}$, which keeps the original aspect ratio and normalize the number of pixels to a fixed value K. We also apply a standard thinning algorithm [ZS84] on the resized ground-truth to avoid unnecessary thickness. In test stage, we still feed the normalized image into network to obtain the prediction, then resize the prediction map back to $H \times W$ for evaluation. We use $K = 180,000$ for the SYM-PASCAL dataset and $K = 60,000$ for the other datasets.

Data Augmentation. We employ the standard data augmentation following [SZJ+16] in training stage: The original image is resized to 3 scales (0.8x, 1.0x, 1.2x), rotated with 4 angles (0°, 90°, 180°, 270°) and then flipped to 3 directions (none, left-to-right, up-to-down).

Table 2.1: Test F-measure ODS comparison on all skeleton datasets. The best numbers are in bold.

Methods	**SK-LARGE**	**SK-SMALL**	**WH-SYMMAX**	**SYM-PASCAL**	**SYMMAX300**
MIL [TK12]	0.353	0.392	0.365	0.174	0.362
HED [XT15]	0.497	0.541	0.732	0.369	0.427
RCF [LCH+17]	0.626	0.613	0.751	0.392	-
FSDS [SZJ+16]	0.633	0.623	0.769	0.418	0.467
LMSDS [SZJ+17]	0.649	0.621	0.779	-	-
SRN [KCJ+17]	0.678	0.632	0.780	0.443	0.446
LSN [LKQY18]	0.668	0.633	0.797	0.425	0.480
Hi-Fi [ZSG+18]	0.724	0.681	0.805	0.454	-
DeepFlux [WXT+19]	0.732	0.695	0.840	0.502	0.491
GeoSkeletonNet	**0.757**	**0.727**	**0.849**	**0.520**	**0.501**

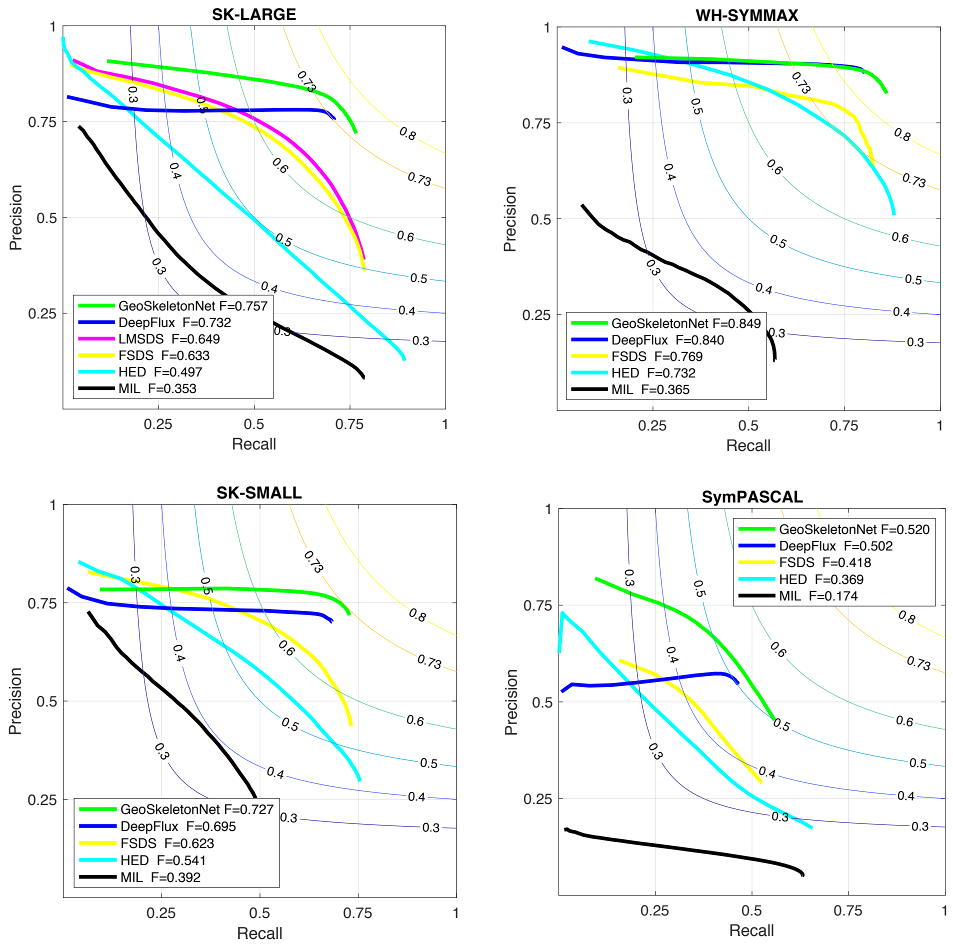


Figure 2.4: Precision-recall curves on four skeleton datasets.

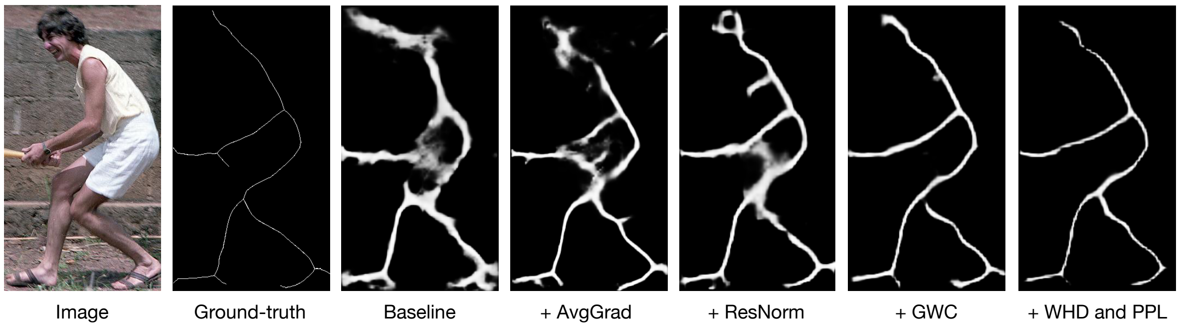


Figure 2.5: Qualitative comparison on the role of modules.

2.4.4 Comparison with the State-of-the-art

As indicated in Table 2.1, our method outperforms the current state-of-the-art by a decent margin on all datasets in terms of F-measure ODS. Compared to the recently proposed DeepFlux [WXT+19], our GeoSkeletonNet improves by 2.5%, 3.2% and 1.8% on the SK-LARGE, SK-SMALL and SYM-PASCAL datasets respectively, under a similar network design. The PR curve shown in Figure 2.4 also indicates a clear performance boost.

In Figure 2.3, we provide a qualitative comparison between our method and the current approaches. In Figure 2.3 (a), (b) and (c), our method significantly reduces the blurry effect in the previous HED-based methods [SZJ+16, ZSG+18]. Meanwhile, our method avoids the occasional dis-continuities in DeepFlux with acceptable sacrifice of accurate localization (*e.g.* in (a), our predicted skeleton has a thicker junction near the knee while maintaining the whole leg complete). Figure 2.3 (d) reveals a failure case: Our method is not able to detect the skeleton of the screen, but still tries to reduce false positive predicted points as possible. In contrast, both FSDS and DeepFlux generate a noisy background.

2.4.5 Ablation Study

Table 2.2: Quantitative comparison on the role of modules.

Baseline	AvgGrad	ResNorm	GWC	WHD+PPL	F-measure ODS
✓					0.712
✓	✓				0.724
✓	✓	✓			0.741
✓	✓	✓	✓		0.753
✓	✓	✓		✓	0.746
✓	✓	✓	✓	✓	0.757

Role of Modules. We conduct an ablation analysis to understand the role of modules in performance contribution. Table 2.2 shows the F-measure ODS on SK-LARGE dataset under multiple module settings of the model. Average gradients (AvgGrad), resolution normalization (ResNorm) and geometrically weighted cross-entropy loss (GWC) greatly contribute to the final performance. Especially, the ResNorm brings the substantial 1.8% improvement, which reflects the difficulty of training on images with various scales.

In addition, Figure 2.5 compares the qualitative results of different module settings. Visually, AvgGrad and ResNorm slightly refine the results from baseline. On the contrary, GWC mitigates the blurry effect by a large margin, which shows the benefits of geometric awareness. Weighted Hausdorff distance (WHD) and patch-based point loss (PPL) further makes the predicted skeleton thinner and provides a better localization.

It is noteworthy that we do not separate the WHD and PPL in the performance analysis on the role of modules. The reason is that directly training on WHD is unstable: WHD is able to provide better localization in the predicted skeleton, but it sometimes generates unexpected disconnection or thickness after certain epochs and leads to a performance drop. Thus, we employ the PPL to stablize the WHD training and always evaluate the performance when both loss terms are turned on. Besides, we also do not include an inference speed comparison since AvgGrad, GWC and WHD+PPL do not affect the time of inference. Only ResNorm will bring a bit overhead, but it is neglectable.

Influence of Distance Hyperparameter γ. We further analyze the influence of the distance hyperparameter γ in geometrically weighted cross-entropy loss (GWC). When $\gamma \to 0$, the GWC reduces to the weighted cross-entropy loss (WC).

Table 2.3: Influence of distance hyper-parameter γ.

Distance Hyper-parameter	$\gamma = 0.125$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 1.0$
F-measure ODS	0.746	0.751	0.753	0.745

Table 2.3 shows that $\gamma = 0.5$ provides the best F-measure ODS in the GWC ablation setting (Baseline + AvgGrad + ResNorm + GWC). Thus, we set $\gamma = 0.5$ in all other experiments.

2.5 Conclusion

In this work, we have developed an end-to-end skeleton detection method that employs geometric awareness. Specifically, we devise a geometry-aware objective function to compute the global similarity between the predicted skeleton map and the ground truth in an end-to-end learning framework. A weighted Hausdorff distance (WHD) is adopted. In addition, we propose a patch-based point loss (PPL) to mitigate the instability in optimizing WHD and capture local features. Furthermore, we adapt the weighted cross-entropy into a geometric form, which significantly boosts the performance of skeleton detection. Evaluation on five standard skeleton detection benchmarks demonstrates the advantages of our proposed method, consistently outperforming the current state-of-the-art methods. In the future, we would like to explore the possibilities of the geometry-aware objective in wider fields, such as semantic segmentation and object detection.

Acknowledgements

This chapter is based on the material “Geometry-Aware End-to-End Skeleton Detection” by Weijian Xu, Gaurav Parmar, and Zhuowen Tu, which appears in British Machine Vision Conference (BMVC) 2019. The dissertation author is the primary investigator and author of this material.

Chapter 3

Constellation Nets for Few-Shot Learning

3.1 Introduction

Tremendous progress has been made in both the development and the applications of the deep convolutional neural networks (CNNs) [KSH12, SZ15, SLJ$^+$15, HZRS16, XGD$^+$17]. Visualization of the internal CNN structure trained on e.g. ImageNet [DDS$^+$09] has revealed the increasing level of semantic relevance for the learned convolution kernels/filters to the semantics of the object classes, displaying bar/edge like patterns in the early layers, object parts in the middle layers, and face/object like patterns in the higher layers [ZF14]. In general, we consider the learned convolution kernels being somewhat implicit about the underlying objects since they represent projections/mappings for the input but without the explicit knowledge about the parts in terms of their numbers, distributions, and spatial configurations.

On the other hand, there has been a rich history about explicit object representations starting from deformable templates [YHC92], pictorial structure [FH05], constellation models [WWP00, FPZ03, STFW05, FFFP06], and grammar-based model [ZM07]. These part-based models [WWP00, FH05, FPZ03, STFW05, ZM07] share three common properties in the algorithm design: (1) *unsupervised learning*, (2) *explicit clustering* to obtain the *parts*, and (3) modeling to characterize the *spatial configuration of the parts*. Compared to the CNN architectures, these methods are expressive with explicit part-based representation. They have pointed to a promising direction for object recognition, albeit a lack of strong practice performance on the modern datasets. Another line of object recognition system with the part concept but trained discriminatively includes the discriminative trained part-based model (DPM) [FGMR09] and the spatial pyramid matching method (SPM) [LSP06]. In the context of deep learning, efforts exist to bring the explicit part representation into deep hierarchical structures [STT12].

The implicit and explicit feature representations could share mutual benefits, especially in *few-shot learning* where training data is scarce: CNNs may face difficulty in learning a generalized representation due to lack of sufficient training data, whereas clustering and dictionary learning provide a direct means for data abstraction. In general, end-to-end learning of both the implicit and explicit part-based representations is a viable and valuable means in machine learning. We view convolutional features as an implicit part-based representation since they are learned through back-propagation via filtering processes. On the other hand, an explicit representation can be attained by introducing feature clustering that captures the data abstraction/distribution under a mixture model.

In this work, we develop an end-to-end framework to combine the implicit and explicit part-based representations for the few-shot classification task by seamlessly integrating constellation models with convolution operations. In addition to keeping a standard CNN architecture, we also employ a cell feature clustering module to encode the potential object parts. This procedure is similar to the clustering/codebook learning for appearance in the constellation model [WWP00]. The cell feature clustering process generates a dense distance map. We further model the relation between cells using a self-attention mechanism, resembling the spatial configuration design in the constellation model [WWP00]. Thus, we name our method constellation networks (ConstellationNet). We demonstrate the effectiveness of our approach on standard few-shot benchmarks, including FC100 [ORLL18], CIFAR-FS [BHTV19] and *mini*-ImageNet [VBL$^+$16] by showing a significant improvement over the existing methods. An ablation study also demonstrates the effectiveness of ConstellationNet is not achieved by simply increasing the model complexity using e.g. more convolution channels or deeper and wider convolution layers (WRN-28-10 [ZK16]) (see ablation study in Table 3.3 and Figure 3.2 (e)).

3.2 Related Work

Few-Shot Learning. Recently, few-shot learning attracts much attention in the deep learning community [SSZ17, LMRS19]. Current few-shot learning is typically formulated as a *meta-learning* problem [FAL17], in which an effective feature embedding is learned for generalization across novel tasks. We broadly divide the existing few-shot learning approaches into three categories: (1) *Gradient-based methods* optimize feature embedding with gradient descent during meta-test stage [FAL17, BHTV19, LMRS19]. (2) *Metric-based methods* learn a fixed optimal embedding with a distance-based prediction rule [VBL$^+$16, SSZ17]. (3) *Model-based methods* obtains a conditional feature embedding via a weight predictor [MRCA18, MYMT18]. Here we adopt ProtoNet [SSZ17], a popular metric-based framework, in our approach and boost the generalization ability of the feature embeddings with explicit structured representations from the constellation model. Recently, [TWH19] proposes a compositional regularization to the image with its attribute annotations, which is different from out unsupervised part-discovery strategy.

Part-Based Constellation/Discriminative Models. The constellation model family [WWP00, FH05, FPZ03, STFW05, FFFP06, ZM07] is mostly generative/expressive that shares two commonalities in the representation: (1) clustering/codebook learning in the appearance and (2) modeling of the spatial configurations. The key difference among these approaches lies in how the spatial configuration is modeled: Gaussian distributions [WWP00]; pictorial structure [FH05]; joint shape model [FPZ03] ; hierarchical graphical model [STFW05]; grammar-based [ZM07]. These constellation models represent a promising direction for object recognition but are not practical compared with deep learning based approaches. There are also discriminative models: The discriminatively trained part-based model (DPM) [FGMR09] is a typical method in this vein where object parts (as HOG features [DT05]) and their configurations (a star model) are learned jointly in a discriminative way. The spatial pyramid matching method (SPM) [LSP06] has no explicit parts but instead builds on top of different levels of grids with codebook learned on top of the SIFT features [Low04]. DPM and SPM are of practical significance for object detection and recognition. In our approach, we implement the constellation model with cell feature clustering and attention-based cell relation modeling to demonstrate the appearance learning and spatial configuration respectively.

Parts models are extensively studied in fine-grained image classifications and object detection to provide spatial guidance for filtering uninformative object proposals [SR15, PHZ17, ZZW$^+$17, GLY19, QLL19]. Related to our work, Neural Activation Constellations (NAC) [SR15] introduces the constellation model to perform unsupervised part model discovery with convolutional networks. Our work is different from NAC in three aspects: (1) The algorithmic mechanisms behind [SR15] and ours are different. [SR15] implements a traditional Gaussian-based constellation module to model the spatial configuration and part selection on top of a fixed pre-trained CNN. However, in our ConstellationNet, our part representation and spatial configuration are modeled by cell feature clustering and self-attention based cell relation module, which is general-purpose, modularized and recursive. (2) In [SR15] , the constellation module is optimized in an EM-like algorithm, which is separate from the CNN optimization. Our constellation modules are seamlessly integrated into the current CNNs and jointly optimized with them. (3) Our ConstellationNet uses the dense cell features from the CNN feature maps, which considers all positions from the images as potential parts and models their relation. However, (Simon et al. 2015) extracts sparse part representations (i.e. it uses at most one part proposal per channel and selects even less parts later), which may not fully utilize the rich information from the CNN feature maps.

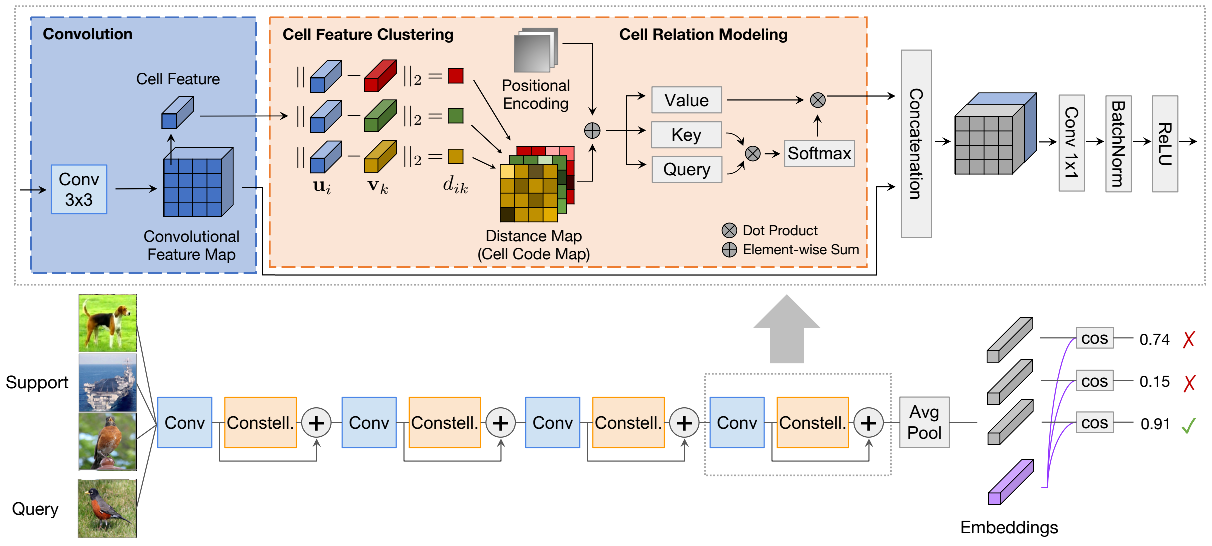


Figure 3.1: **Illustration of our ConstellationNet pipeline** where the bottom part is the network architecture based on Conv-4 backbone, and the top part shows the constellation model. Our proposed ConstellationNet consists of "Constell." modules that perform explicit cell feature clustering with self-attention for joint relation modeling.

3.3 Few-Shot Learning

In a standard classification problem, we aim to learn a model trained on the dataset $\mathcal{D}^{\text{base}}$ that can generalize its classification ability to unseen test set $\mathcal{D}^{\text{novel}}$ belonging to same categories. In few-shot classification problem, we encourage $\mathcal{D}^{\text{base}}$ and $\mathcal{D}^{\text{novel}}$ to be formed from different categories to emphasize model's generalization ability on novel categories, where we denote training categories as $\mathcal{C}_{\text{base}}$, test categories as $\mathcal{C}_{\text{novel}}$, and $\mathcal{C}_{\text{base}} \cap \mathcal{C}_{\text{novel}} = \varnothing$ to ensure the fairness.

In the training stage (a.k.a. *meta-train* stage), metric-based few-shot learning approaches [SSZ17, VBL$^+$16, ORLL18] usually learn a feature extractor $\phi(\mathbf{x})$ on the dataset $\mathcal{D}^{\text{base}}$ to obtain generic feature embedding by optimizing the loss $\mathcal{L}(\phi)$:

$$\mathcal{L}(\phi) = \mathbb{E}_{\{(\mathbf{x},y)\}\sim\mathcal{D}_{\text{base}}} \ell\Big(\{(\phi(\mathbf{x}),y)\}\Big) \tag{3.1}$$

where $\{(\mathbf{x},y)\}$ is a sampled mini-batch of data points and $\ell(\cdot)$ is usually an episodic few-shot loss

[VBL+16] or a standard cross-entropy loss [CLX+21].

In the inference stage (a.k.a. *meta-test* stage), a typical few-shot benchmark evaluates the model on K-way, N-shot classification tasks $\mathcal{T}$ drawn from $\mathcal{D}^{\text{novel}}$, where each task has a support set and a query set, i.e. $\mathcal{T} = (\mathcal{T}^{\text{supp}}, \mathcal{T}^{\text{query}})$. The support set $\mathcal{T}^{\text{supp}}$ contains K classes and each class has N images (e.g. $K = 5, N \in \{1,5\}$). Following [SSZ17], the prediction $\hat{y}'$ of a query image $\mathbf{x}' \in \mathcal{T}^{\text{query}}$ is given by the label of nearest prototype $\mathbf{c}_k$ from $\mathcal{T}^{\text{supp}}$ under a cosine similarity $d(\cdot,\cdot)$:

$$\hat{y}' = \arg\max_k d\Big(\phi(\mathbf{x}'), \mathbf{c}_k\Big), \qquad \mathbf{c}_k = \frac{1}{N} \sum_{(\mathbf{x},y)\in\mathcal{T}^{\text{supp}},\ y=k} \phi(\mathbf{x}). \tag{3.2}$$

An extended description of the few-shot learning framework can be found from Appendix A.1.1. The generalization ability of the feature extractor $\phi(\mathbf{x})$ is improved in terms of training scheme (e.g. episodic learning [VBL+16]), network design (e.g. task condition [ORLL18]) or objective function (e.g. learnable distance [SYZ+18]). In our method, we propose a novel network design by inserting constellation models into CNNs and strengthen the intermediate features.

3.4 Constellation Model

The concept of constellation has been introduced to the few-shot learning scenario in early years [FFFP06], in which the appearance and the shape are independently learned in a mixture model. In our work, we revisit the constellation model in an end-to-end learning framework: First, we define the *cell feature* as the individual local feature at a position in the feature map (see Figure 3.1). We then employ *cell feature clustering* to model the underlying distribution of input cell features, implying a part discovery procedure. We further obtain the distance map of the cell features from clustering and then perform *cell relation modeling* to build spatial relationship.

3.4.1 Cell Feature Clustering

In convolutional neural networks (CNNs), the convolutional filters are learned to detect the discriminative patterns from low-level to high-level through back-propagation [ZF14]. In fact, the backward signal in the back-propagation is not necessarily needed to obtain a pattern detector. With the feature map in the forward step of the CNN, we are able to cluster the individual features at each location of the feature map (a.k.a. *cell features*) into multiple centers and employ the cluster centers as filters [CN12, KDDD15]. Assume we obtain a convolutional feature map $\mathbf{U}$ with batch size B, spatial size $H \times W$ and channels C. We disensemble the feature map $\mathbf{U} \in \mathbb{R}^{B \times H \times W \times C}$ into a *cell features set* $\mathcal{U} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n\}$ where $n = BHW$ and $\mathbf{u}_i \in \mathbb{R}^C$ is a cell feature. Naively, we can conduct a k-means algorithm on input cell features $\mathcal{U}$ to solve the clustering objective:

$$\min \sum_i \sum_k m_{ik} ||\mathbf{u}_i - \mathbf{v}_k||_2^2 \quad \text{s.t.} \quad m_{ik} \in \{0,1\}, \quad \sum_k m_{ik} = 1 \tag{3.3}$$

where $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_K\}$ is a set of cluster centers and m_{ik} indicates if the input cell feature $\mathbf{u}_i$ is assigned to cluster center $\mathbf{v}_k$. The clustering-based filters $\mathcal{V}$ can model the underlying cell feature distributions and capture the most frequent features, which can be explicitly interpreted as meaningful part patterns/part types. The hard assignment map $\mathbf{m}_i = (m_{i1}, m_{i2}, ..., m_{iK})$ of input cell feature $\mathbf{u}_i$ onto the cluster centers can be used as a part-based representation, providing alternative information to the next layer in the CNN.

However, there are two issues remaining unsolved in the naive design: Firstly, CNNs are typically optimized in a stochastic gradient descent (SGD) manner. Thus, in each forward step, only a mini-batch of images are proceeded to provide cell features, which implies that the cluster centers cannot extract the global feature distribution across the whole dataset. Secondly, the hard assignment map has limited information due to its discrete representation. Therefore, inspired by [Scu10], we design a mini-batch soft k-means algorithm to cluster the cell features approximately:

- **Initialization.** Randomly initialize global cluster centers $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_K\}$ and a counter $\mathbf{s} = (s_1, s_2, ..., s_K) = \mathbf{0}$.

- **Cluster Assignment.** In forward step, given input cell features $\mathcal{U} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n\}$, we compute the distance vector $\mathbf{d}_i = (d_{i1}, d_{i2}, ...d_{iK})$ between input cell feature $\mathbf{u}_i$ and all cluster centers $\mathcal{V}$. We then compute the soft assignment $m_{ik} \in \mathbb{R}$ and generate the current mini-batch centers $\mathbf{v}'_k$:

$$d_{ik} = ||\mathbf{u}_i - \mathbf{v}_k||_2^2, \qquad m_{ik} = \frac{e^{-\beta d_{ik}}}{\sum_j e^{-\beta d_{ij}}}, \qquad \mathbf{v}'_k = \frac{\sum_i m_{ik}\mathbf{u}_i}{\sum_i m_{ik}} \tag{3.4}$$

  where $\beta > 0$ is an inverse temperature.

- **Centroid Movement.** We formulate a count update $\Delta\mathbf{s} = \sum_i \mathbf{m}_i$ by summing all assignment maps $\mathbf{m}_i = (m_{i1}, m_{i2}, ...m_{iK})$. The current mini-batch centers $\mathbf{v}'_k$ are then updated to the global centers $\mathbf{v}_k$ with a momentum coefficient η:

$$\mathbf{v}_k \leftarrow (1-\eta)\mathbf{v}_k + \eta\mathbf{v}'_k, \qquad \eta = \frac{\lambda \Delta s_k}{s_k + \Delta s_k} \tag{3.5}$$

- **Counter Update.** Counter $\mathbf{s}$ is updated and distance vectors $\{\mathbf{d}_i\}$ are reshaped and returned:

$$\mathbf{s} \leftarrow \mathbf{s} + \Delta\mathbf{s} \tag{3.6}$$

With gradually updating global cluster centers, the above algorithm is able to address the issue of limited data in a mini-batch. In addition, we reshape the distance vectors $\{\mathbf{d}_i\}$ of all input cell features to a distance map $\mathbf{D} \in \mathbb{R}^{B \times H \times W \times K}$. Each distance vector $\mathbf{d}_i$ can be seen as a *learned cell code* in codebook (dictionary) learning, which encodes a soft assignment of the visual word (i.e. cell feature) onto the codewords (i.e. cluster centers) and implies a part representation. The distance map $\mathbf{D}$ then can be viewed as a cell code map that represents a spatial distribution of identified parts, which is passed to following layers. Empirically, it is observed that when $\mathbf{u}_i$ and $\mathbf{v}_k$ are L_2 normalized, the training procedure is more stable and the Euclidean distance d_{ik} is equivalent to a cosine similarity up to an affine transformation. Details of the cell feature clustering can be found in Appendix A.1.9.

3.4.2 Cell Relation and Spatial Configuration Modeling

Before the deep learning era, traditional constellation models [FFFP06] decompose visual information into appearance and shape representation. The appearance of different parts in the image is treated independently while the shape of parts is assumed to have spatial connections. In our constellation model, we establish the spatial relationship among the individual part-based representations at a different location from the distance map as well. Specifically, we apply the self-attention mechanism [VSP+17] to build the spatial relationship and enhance the representation instead of using probabilistic graphical models in prior work [FFFP06].

In cell relation modeling, we add a *positional encoding* $\mathbf{P} \in \mathbb{R}^{B\times H\times W\times C}$ following [CMS+20] for spatial locations to the distance map $\mathbf{D}$ and obtain the input feature map $\mathbf{F}_\mathrm{I}$ for query and key layers. For value layer, we directly flatten the distance map $\mathbf{D}$ to another input feature map $\mathbf{F}'_\mathrm{I}$:

$$\mathbf{F}_\mathrm{I} = \mathrm{SpatialFlatten}(\mathbf{D}+\mathbf{P}) \in \mathbb{R}^{B\times HW\times K}, \quad \mathbf{F}'_\mathrm{I} = \mathrm{SpatialFlatten}(\mathbf{D}) \in \mathbb{R}^{B\times HW\times K} \tag{3.7}$$

The input feature maps $\mathbf{F}_\mathrm{I}, \mathbf{F}'_\mathrm{I}$ are transformed into query, key and value $\{\mathbf{F}^q, \mathbf{F}^k, \mathbf{F}^v\} \subset \mathbb{R}^{B\times HW\times K}$ by three linear layers $\{\mathbf{W}^q, \mathbf{W}^k, \mathbf{W}^v\} \subset \mathbb{R}^{K\times K}$ and further computes the output feature $\mathbf{F}_\mathrm{A}$:

$$[\mathbf{F}^q, \mathbf{F}^k, \mathbf{F}^v] = [\mathbf{F}_\mathrm{I}\mathbf{W}^q, \mathbf{F}_\mathrm{I}\mathbf{W}^k, \mathbf{F}'_\mathrm{I}\mathbf{W}^v] \tag{3.8}$$

$$\mathbf{F}_\mathrm{A} = \mathrm{Att}(\mathbf{F}^q, \mathbf{F}^k, \mathbf{F}^v) = \mathrm{softmax}\left(\frac{\mathbf{F}^q(\mathbf{F}^k)^\top}{\sqrt{K}}\right)\mathbf{F}^v \tag{3.9}$$

The softmax of dot product between query and key matrix $\mathbf{F}^q(\mathbf{F}^k)^\top \in \mathbb{R}^{B\times HW\times HW}$ calculates the similarity scores in the embedding space among features across the spatial dimension. This encodes the spatial relationship of input features and leads to an enhanced output feature representation $\mathbf{F}_\text{A}$. Besides, $\sqrt{K}$ in the denominator is to stabilize the gradient. In practice, we adopt a multi-head attention to model the feature relation in the embedding subspaces:

$$\mathbf{F}_\text{MHA} = \text{MultiHeadAtt}(\mathbf{F}^q, \mathbf{F}^k, \mathbf{F}^v) = [\mathbf{F}_1, ..., \mathbf{F}_J]\mathbf{W}, \qquad \mathbf{F}_j = \text{Att}(\mathbf{F}^q_j, \mathbf{F}^k_j, \mathbf{F}^v_j) \tag{3.10}$$

In a J-head attention, the aforementioned similarity scores in the $K' = \frac{K}{J}$ dimensional embedding subspace are calculated using the query, key and value from j-th head, i.e. $\{\mathbf{F}^q_j,\ \mathbf{F}^k_j,\ \mathbf{F}^v_j\} \subset \mathbb{R}^{B\times HW\times K'}$. The output features $\mathbf{F}_j$ of each head are computed following Eq. 3.9. All the output features $\{\mathbf{F}_1, ..., \mathbf{F}_J\}$ are concatenated back into K dimension embedding and further processed with a linear layer $\mathbf{W} \in \mathbb{R}^{K\times K}$ to generate multi-head output features $\mathbf{F}_\text{MHA}$. Such multi-head attention settings could provide more diverse feature relation without introducing extra parameters.

3.4.3 Integrate Constellation Model with CNNs

Our constellation model has the capability to capture explicit structured features and encodes spatial relation among the cell features. The output features yield informative visual cues which are able to strengthen the convolutional features. Thus, as shown in Figure 3.1, we place the constellation model after the convolution operation to extract its unique explicit features and concatenate them with the original convolutional feature map. A following 1×1 convolutional layer is used on the concatenated features to restore the channels of convolutional feature map. In Table 3.3, we provide evidence that merging features from constellation model to the CNN backbone can significantly improve the representation ability. In contrast, increasing channels in CNNs alone to double the parameters (second row in Table 3.3) can only improve the performance marginally. Optionally, we found it is useful to adopt auxiliary loss when training the constellation model in deeper networks (e.g. ResNet-12). On top of each constellation model, we conduct a standard classification to acquire additional regularization.

3.4.4 Why Clustering and Self-attention?

As described in Section 3.1 and 3.2, classical constellation models [FPZ03, FH05] extract parts with their spatial relationship; they are expressive but do not produce competitive results on modern image benchmarks. CNN models [KSH12, HZRS16] attain remarkable results on large-scale image benchmarks [DDS+09] but they are limited when training data is scarce. We take the inspiration from the traditional constellation models, but with a realization that overcomes their previous modeling limitations.

The main contribution of our work is a constellation module/block that performs **cell-wise clustering**, followed by **self-attention** on the **clustering distance map + positional encoding**. This separates our work from previous attempts, e.g. non-local block work [WGGH18] in which long-range non-linear averaging is performed on the convolution features (no clustering, nor positional encoding for the spatial configuration). The main properties of our constellation block include: (1) **Cell based dense representation** as opposed to the sparse part representation in [WWP00] to make the cells recursively modeled in the self-attention unit in a modularized and general-purpose way. (2) **Clustering** to generate the cell code after clustering (codebook learning) that attains abstraction and is not dependent on the CNN feature dimensions. (3) **Positional encoding** (as in [CMS+20]) for cells to encode the spatial locations. (4) **Tokenized representation** as expressive parts (code/clustering distance map + positional encoding) for the cells. (5) **Self-attention** to jointly model the cell code and positional encoding to capture the relationship between the parts together with their spatial configurations.

3.5 Experiment

3.5.1 Datasets

We adopt three standard benchmark datasets that are widely used in few-shot learning, CIFAR-FS dataset [BHTV19], FC100 dataset [ORLL18], and *mini*-ImageNet dataset [VBL$^+$16]. Details about dataset settings in few-shot learning are in Appendix A.1.2.

3.5.2 Network with Multi-Branch

We build ConstellationNet on two ProtoNet variants, namely Conv-4 and ResNet-12, which are commonly used in few-shot learning. Details of networks and the optimization are in Appendix.

We develop a new technique, *Multi-Branch*, to optimize standard classification loss and prototypical loss simultaneously. We find the two training schemes, standard classification scheme and prototypical scheme, can be a companion rather than a conflict. Details of these two schemes can be found from Appendix A.1.1. Different from standard network backbone used in prior works, our embedding $\phi(\mathbf{x})$ is separated into two branches after a shared stem (Y-shape). Details of our multi-branch design are elaborated in A.1.10. The detailed ablation study is described in Table 3.3.

Feature Augmentation. During the meta-testing stage, we discover that concatenating features before average pooling to the final output can improve classification accuracy. The advantage of this technique is that no additional training and model parameters are introduced.

3.5.3 Results on Standard Benchmarks

Table 3.1 and 3.2 summarize the results of the few-shot classification tasks on CIFAR-FS, FC100, and *mini*-ImageNet, respectively. Our method shows a notable improvement over several strong baselines in various settings. ConstellationNet significantly improves the performance on shallow networks (Conv-4). In Table 3.2, our model outperforms SIB [HMX$^+$20] 1-shot by 0.6% and 5-shot by 5.6%. In Table 3.1, our model outperforms MetaOptNet [LMRS19] by 5.95% in 1-shot and 6.24% in 5-shot. For deep networks with rich features, the constellation module still contributes to the performance, showing its complementary advantage to convolution. Our ResNet-12 model beats [LMRS19] 1-shot result by 2.7% on FC100, 3.4% on CIFAR-FS, and 1.72% on *mini*-ImageNet. The consistent improvement over both shallow and deep networks across all three datasets shows the generality of our method. Our ConstellationNet is orthogonal to the margin loss based methods [LCL$^+$20, LHL$^+$20], and we also do not use extra cross-modal information [XROOP19, LHL$^+$20]. On the contrary, our model enhances the embedding generalization ability by incorporating its own part-based representation. Additionally, to verify the orthogonality of our method, we adapt the negative margin loss following [LCL$^+$20] to our Conv-4 models in Appendix A.1.8. We observe ConstellationNet with negative margin brings 0.52% improvement to ConstellationNet, and obtains 6.93% gain compared with baseline on *mini*-ImageNet.

3.6 Model Analysis

3.6.1 Architecture Alternatives

In Table 3.3, we first study the role of each module in ConstellationNet, where the number of parameters is controlled approximately equivalent to the baseline's size. Our constellation model brings 6.41% and 2.59% improvements over baseline on 1-shot Conv-4 and ResNet-12 results. Combined with our multi-branch training procedure, the model further improves additional 1.34% and 1.26% on 1-shot Conv-4 and ResNet-12, respectively. Finally, feature augmentation from penultimate layer to final output embedding brings additional 0.45% and 0.27% improvements on two variants.

Table 3.1: **Comparison to prior work on *mini*-ImageNet.** Average 5-way classification accuracies (%) on *mini*-ImageNet meta-test split are reported with 95% confidence intervals. Results of prior works are adopted from [LMRS19] and original papers. † used extra cross-modal information.

Model	**Backbone**	***mini*-ImageNet 5-way**	
		1-shot	**5-shot**
Meta-Learning LSTM [RL17]	Conv-4	43.44 ± 0.77	60.60 ± 0.71
Matching Networks [VBL$^+$16]	Conv-4	43.56 ± 0.84	55.31 ± 0.73
Prototypical Networks [SSZ17]	Conv-4	49.42 ± 0.78	68.20 ± 0.66
Transductive Prop Nets [LLP$^+$19]	Conv-4	55.51 ± 0.86	69.86 ± 0.65
MetaOptNet [LMRS19]	Conv-4	52.87 ± 0.57	68.76 ± 0.48
Negative Margin [LCL$^+$20]	Conv-4	52.84 ± 0.76	70.41 ± 0.66
ConstellationNet (ours)	Conv-4	$\mathbf{58.82 \pm 0.23}$	$\mathbf{75.00 \pm 0.18}$
SNAIL [MRCA18]	ResNet-12	55.71 ± 0.99	68.88 ± 0.92
TADAM [ORLL18]	ResNet-12	58.50 ± 0.30	76.70 ± 0.30
TapNet [YSM19]	ResNet-12	61.65 ± 0.15	76.36 ± 0.10
Variational FSL [ZZN$^+$19]	ResNet-12	61.23 ± 0.26	77.69 ± 0.17
MetaOptNet [LMRS19]	ResNet-12	62.64 ± 0.61	78.63 ± 0.46
CAN [HCB$^+$19]	ResNet-12	63.85 ± 0.48	79.44 ± 0.34
SLA-AG [LHS20]	ResNet-12	62.93 ± 0.63	79.63 ± 0.47
Meta-Baseline [CLX$^+$21]	ResNet-12	63.17 ± 0.23	79.26 ± 0.17
AM3 [XROOP19] †	ResNet-12	65.21 ± 0.30	75.20 ± 0.27
ProtoNets + TRAML [LHL$^+$20]	ResNet-12	60.31 ± 0.48	77.94 ± 0.57
AM3 + TRAML [LHL$^+$20] †	ResNet-12	$\mathbf{67.10 \pm 0.52}$	79.54 ± 0.60
Negative Margin [LCL$^+$20]	ResNet-12	63.85 ± 0.81	$\mathbf{81.57 \pm 0.56}$
ConstellationNet (ours)	ResNet-12	64.89 ± 0.23	79.95 ± 0.17

We also test the baseline model with extra channels in the Table 3.3. The new model only shows slight improvements over original baseline, and is outperformed by our ConstellationNet with a large margin. We also obtain WRN-28-10 baseline results to validate our improvement. While making ResNet baselines deeper and wider, our ConstellationNet still outperforms this strong baseline. In Figure 3.2 (e), we further study whether the performance gap between ConstellationNet and baseline can be reduced by simply altering the baseline's model complexity using e.g. more convolution channels. Although the trend of baseline accuracy increases when increasing the model parameter number gradually, the performance gap is still significant. This validates our concept that modeling hierarchical part structures can greatly benefit features learned from convolution operation, and obtain a more robust feature representation. In addition,

Table 3.2: **Comparison to prior work on FC100 and CIFAR-FS.** Average 5-way classification accuracies (%) on CIFAR-FS and FC100 meta-test split are reported with 95% confidence intervals. Results of prior works are adopted from [LMRS19] and original papers.

Model	Backbone	CIFAR-FS 5-way		FC100 5-way	
		1-shot	5-shot	1-shot	5-shot
MAML [FAL17]	Conv-4	58.9 ± 1.9	71.5 ± 1.0	-	-
Prototypical Networks [SSZ17]	Conv-4	55.5 ± 0.7	72.0 ± 0.6	-	-
Relation Networks [SYZ$^+$18]	Conv-4	55.0 ± 1.0	69.3 ± 0.8	-	-
R2D2 [BHTV19]	Conv-4	65.3 ± 0.2	79.4 ± 0.1	-	-
SIB [HMX$^+$20]	Conv-4	68.7 ± 0.6	77.1 ± 0.4	-	-
ConstellationNet (ours)	Conv-4	**69.3 ± 0.3**	**82.7** ± **0.2**	-	-
Prototypical Networks [SSZ17]	ResNet-12	72.2 ± 0.7	83.5 ± 0.5	37.5 ± 0.6	52.5 ± 0.6
TADAM [ORLL18]	ResNet-12	-	-	40.1 ± 0.4	56.1 ± 0.4
MetaOptNet-RR [LMRS19]	ResNet-12	72.6 ± 0.7	84.3 ± 0.5	40.5 ± 0.6	55.3 ± 0.6
MetaOptNet-SVM [LMRS19]	ResNet-12	72.0 ± 0.7	84.2 ± 0.5	41.1 ± 0.6	55.5 ± 0.6
ConstellationNet (ours)	ResNet-12	**75.4 ± 0.2**	**86.8** ± **0.2**	**43.8** ± **0.2**	**59.7** ± **0.2**

Table 3.3: **Effectiveness of modules.** Average classification accuracies (%) on *mini*-ImageNet meta-test split. We compare our ConstellationNet with alternative architectures including the baseline and the modified baseline with extra channels based on Conv-4 and ResNet-12. We also include a baseline with WideResNet-28-10 [ZK16] backbone for comparison.

Baseline	Cell Feature Clustering	Cell Relation Modeling	Multi Branch	Feature Augment	Extra Channels	1x1 Convolution	#Params	Conv-4		ResNet-12	
							Conv-4/Res-12	1-shot	5-shot	1-shot	5-shot
✓							117K/8.0M	50.62 ± 0.23	68.40 ± 0.19	60.77 ± 0.22	78.76 ± 0.17
✓					✓		222K/16M	51.76 ± 0.22	69.54 ± 0.18	61.45 ± 0.22	79.33 ± 0.16
✓	✓						146K/8.3M	53.34 ± 0.23	70.61 ±0.19	62.24 ± 0.23	79.55 ± 0.16
✓		✓					184K/9.7M	55.92 ± 0.23	73.02 ± 0.18	62.75 ± 0.23	79.21 ± 0.17
✓		✓				✓	192K/8.4M	55.46 ± 0.23	72.52 ± 0.18	61.54 ± 0.24	76.51 ± 0.18
✓	✓	✓					200K/8.4M	57.03 ± 0.23	74.09 ± 0.18	63.36 ± 0.23	79.72 ± 0.17
✓	✓	✓	✓				200K/8.4M	58.37 ± 0.23	74.52 ± 0.18	64.62 ± 0.23	79.60 ± 0.17
✓	✓	✓	✓	✓			200K/8.4M	**58.82** ± **0.23**	**75.00** ± **0.18**	**64.89** ± **0.23**	**79.95** ± **0.17**
							WRN			WideResNet-28-10	
✓					✓		36.5M			61.54 ± 0.25	79.41 ± 0.23

applying self-attention on the distance map (6-th row: 57.03% on Conv-4, 1-shot) achieves better performance than directly applying it to the original cell features (i.e. convolutional feature map) (4-th row: 55.92% on Conv-4, 1-shot). We also tried to replace the cell feature clustering module with a 1x1 convolution layer (output dimension is equal to the number of clusters) (5-th row: 55.46% on Conv-4, 1-shot). It is worse than our results (6-th row) as well. We observe that the 1x1 convolution layer is less expressive than the cell feature clustering module, making it difficult to extract enough context information during cell relation modeling.

3.6.2 Modules Analysis

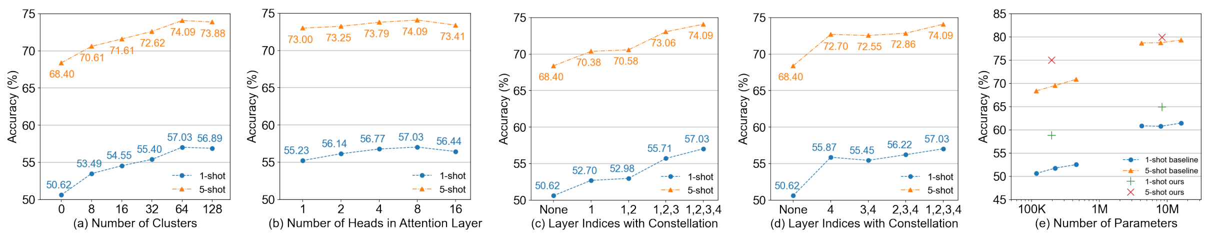


Figure 3.2: **Modules analysis.** (a, b, c, d) We study the effectiveness of changing the number of clusters, the number of heads in attention layer, and the layer indices with constellation based on Conv-4, (e) We demonstrate the performance gain of our ConstellationNet is unmatched by increasing the model complexity of our baselines. All experiments are done on *mini*-ImageNet.

In Figure 3.2 (a), we vary the number of clusters adapted in all layers to observe the performance change. We found that increasing the number of clusters improves the accuracy in general, and set clusters to 64 is optimal in terms of both model size and classification performance. Figure 3.2 (b) shows the number of attention heads does not effect performance as much as the number of cluster, and 8-head attention obtains 1.80% performance gain on the 1-shot setting compared to 1-head attention. In Figure 3.2 (c, d), we also study the effectiveness of clustering algorithm applied to different layers. The results show both early features and high-level features benefit from introducing clusters algorithm into the original CNN architecture.

3.6.3 Visualization

Figure 3.3 demonstrates the visualization of cluster centers in each layer of Conv-4 model on *mini*-ImageNet. In the upper part of the figure, each image shows patches corresponding to the nearest cell features to a cluster center (i.e. with lowest Euclidean distance). It is observed that clusters in early layers (e.g. layer 1,2) represent simple low-level patterns while the clusters

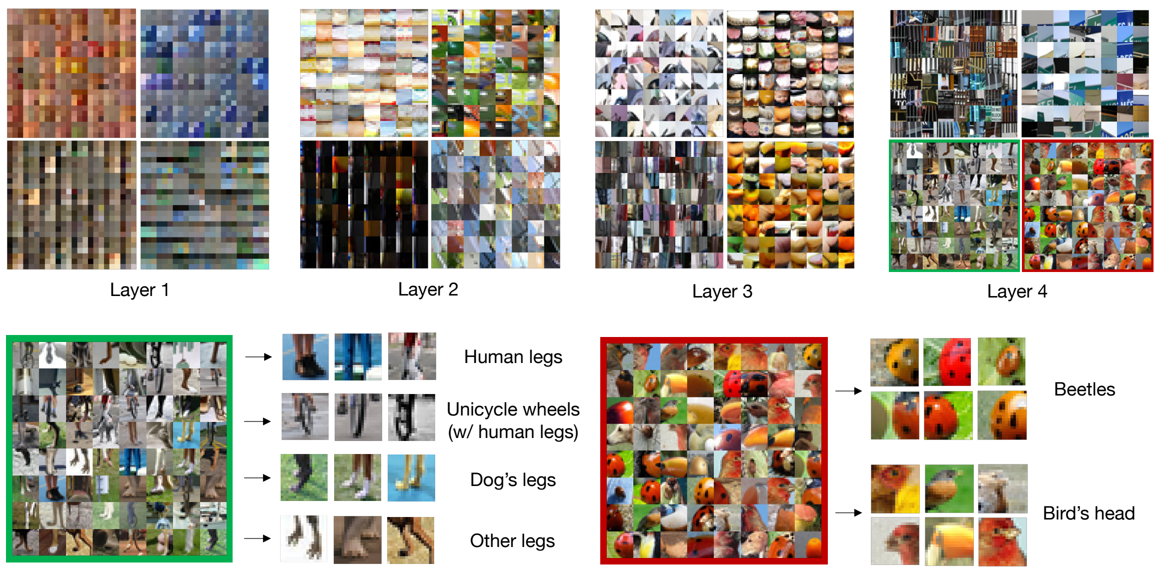


Figure 3.3: **Visualization of cluster centers.** (Upper) We visualize four cluster centers in each layer by showing patches associated with cell features that have the nearest distance to the clustering center. (Lower) Identifying parts from two cluster centers in layer 4: Left one with green box represents various types of legs. Right one with red box mostly shows beetles and bird's head, sharing a dotted structure.

in high layers (e.g. layer 3,4) indicate more complex structures and parts. In the lower part of the figure, we choose two cluster centers from layer 4 for further interpretation: The left one with green box could possibly represent *legs* since it consists of various types of legs from human, dog and other animals. The right one with the red box shows most nearest cell features to this cluster center are parts with bird's head or beetles, which share a dotted structure (i.e. black dots on beetles / eyes on bird's head).

The left side of Figure 3.4 shows the visualization of cell features that are assigned to different clusters. For each image, we extract the assignment maps corresponding to three cluster centers generated in the last constellation module of Conv-4 and find multiple cell features with the highest assignments within each assignment map. The locations of cell features are projected back in the original image space, marked by three different colors of "·" in the raw image to show three different feature clusters. For a given class of images, the same cluster centers are selected

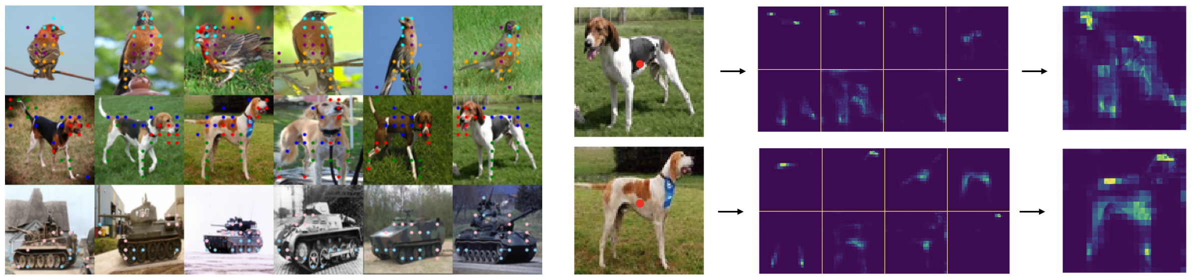


Figure 3.4: **Visualization of the cells assignment and attention maps.** (Left) Each color represents a cluster, and each point, marked as "·", represents a cell assigned to a cluster center. We demonstrate 6 samples for each class (bird, dog and tank). (Right) We visualize attention maps of one query feature (at the location of red point in left part) with all key features. The middle part shows the attention maps corresponding to 8 heads in the multi-head attention. The right part shows an overlapped map of all attention maps.

for comparison across 6 samples. As shown in Figure 3.4, we observe part information of each class is explicitly discovered. For the bird category, we can see different parts in each image, including head (cyan "·"), body (purple "·") and tail (yellow "·"). For the dog category, we see parts including heads (red "·"), legs (green "·") and body (blue "·"). For the tank category, we see parts like track (light blue "·") and turret (pink "·").

The right side of Figure 3.4 visualizes the attention maps in the cell relation model. We use the last constellation module in the ResNet-12 model for visualization since it captures high-level features that better represent parts. We choose one query feature at the center of the object and show its attention map to all key features. The middle part of the figure shows the attention maps corresponding to 8 heads in the multi-head attention. It is observed that some parts are identified such as head (second map in first row), legs (first two map in second row), buttock (first map in first row) and body (second map in the second row). A merged attention map by overlaying all 8 attention maps is presented at right part of the figure. It indicates that all the attention heads together can extract the features of the whole object, which would be useful for final classification.

3.7 Conclusion

In this work, we present ConstellationNet by introducing an explicit feature clustering procedure with relation learning via self-attention. We implement a mini-batch soft k-means algorithm to capture the cell feature distribution. With integrated implicit (standard CNN modules) and explicit (cell feature clustering + cell relation modeling) representations, our proposed ConstellationNet achieves significant improvement over the competing methods on few-shot classification benchmarks.

Acknowledgements

This chapter is based on the material "Attentional Constellation Nets for Few-Shot Learning" by Weijian Xu*, Yifan Xu*, Huaijin Wang*, and Zhuowen Tu, which appears in International Conference on Learning Representations (ICLR) 2021. The dissertation author is the co-primary investigator and author of this material.

Part II

Multi-Scale Structures in Transformers

Chapter 4

Co-Scale Conv-Attentional Transformers

4.1 Introduction

A notable recent development in artificial intelligence is the creation of attention mechanisms [XBK$^+$15] and Transformers [VSP$^+$17], which have made a profound impact in a range of fields including natural language processing [DCLT19, RNSS18], document analysis [XLC$^+$20], speech recognition [DXX18], and computer vision [DBK$^+$21, CMS$^+$20]. In the past, state-of-the-art image classifiers have been built primarily on convolutional neural networks (CNNs) [LBBH98, KSH12, SLJ$^+$15, SZ15, HZRS16, XGD$^+$17] that operate on layers of filtering processes. Recent developments [TCD$^+$21, DBK$^+$21] however begin to show encouraging results for Transformer-based image classifiers.

In essence, both the convolution [LBBH98] and attention [XBK$^+$15] operations address the fundamental representation problem for structured data (e.g. images and text) by modeling the local contents, as well as the contexts. The receptive fields in CNNs are gradually expanded through a series of convolution operations. The attention mechanism [XBK$^+$15, VSP$^+$17] is, however, different from the convolution operations: (1) the receptive field at each location or token in self-attention [VSP$^+$17] readily covers the entire input space since each token is "matched"

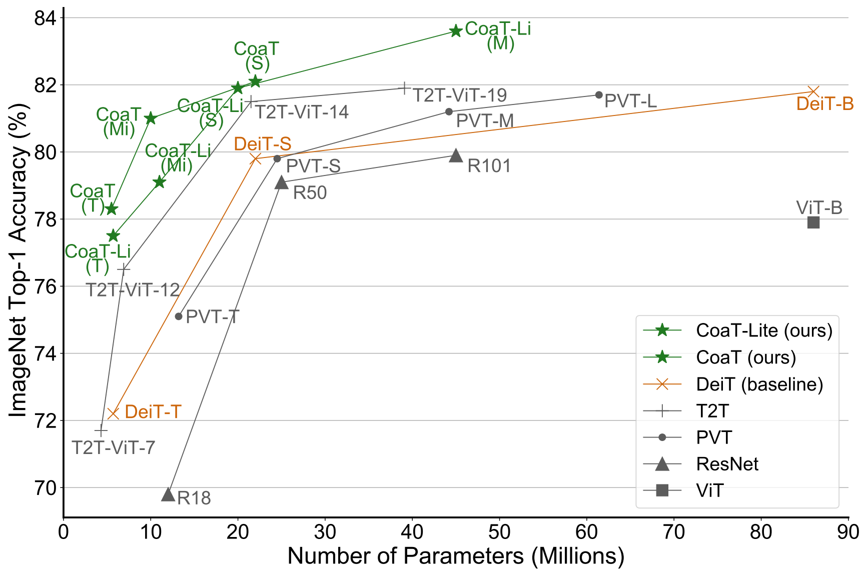


Figure 4.1: **Model Size vs. ImageNet Accuracy.** Our CoaT model significantly outperforms other image Transformers. Details are in Table 4.2.

with all tokens including itself; (2) the self-attention operation for each pair of tokens computes a dot product between the "query" (the token in consideration) and the "key" (the token being matched with) to weight the "value" (of the token being matched with).

Moreover, although the convolution and the self-attention operations both perform a weighted sum, their weights are computed differently: in CNNs, the weights are learned during training but fixed during testing; in the self-attention mechanism, the weights are dynamically computed based on the similarity or affinity between every pair of tokens. As a consequence, the self-similarity operation in the self-attention mechanism provides modeling means that are potentially more adaptive and general than convolution operations. In addition, the introduction of position encodings and embeddings [VSP+17] provides Transformers with additional flexibility to model spatial configurations beyond fixed input structures.

Of course, the advantages of the attention mechanism are not given for free, since the self-attention operation computes an affinity/similarity that is more computationally demanding than linear filtering in convolution. The early development of Transformers has mainly focused on natural language processing tasks [VSP$^+$17, DCLT19, RNSS18] since text is “shorter” than an image, and text is easier to tokenize. In computer vision, self-attention has been adopted to provide added modeling capability for various applications [WGGH18, XLCT18, ZJK20]. With the underlying framework increasingly developed [DBK$^+$21, TCD$^+$21], Transformers start to bear fruit in computer vision [CMS$^+$20, DBK$^+$21] by demonstrating their enriched modeling capabilities.

In the seminal DEtection TRansformer (DETR) [CMS$^+$20] algorithm, Transformers are adopted to perform object detection and panoptic segmentation, but DETR still uses CNN backbones to extract the basic image features. Efforts have recently been made to build image classifiers from scratch, all based on Transformers [DBK$^+$21, TCD$^+$21, WXL$^+$21]. While Transformer-based image classifiers have reported encouraging results, performance and design gaps to the well-developed CNN models still exist. For example, in [DBK$^+$21, TCD$^+$21], an input image is divided into a single grid of fixed patch size. In this work, we develop Co-scale conv-attentional image Transformers (CoaT) by introducing two mechanisms of practical significance to Transformer-based image classifiers. The contributions of our work are summarized as follows:

- We introduce a co-scale mechanism to image Transformers by maintaining encoder branches at separate scales while engaging attention across scales. Two types of building blocks are developed, namely a serial and a parallel block, realizing **fine-to-coarse**, **coarse-to-fine**, and **cross-scale** image modeling.
- We design a conv-attention module to realize **relative position embeddings** with convolutions in the **factorized attention** module that achieves significantly enhanced computation efficiency when compared with vanilla self-attention layers in Transformers.

Our resulting Co-scale conv-attentional image Transformers (CoaT) learn effective representations under a modularized architecture. On the ImageNet benchmark, CoaT achieves state-of-the-art classification results when compared with the competitive convolutional neural networks (e.g.

EfficientNet [TL19]), while outperforming the competing Transformer-based image classifiers [DBK$^+$21, TCD$^+$21, WXL$^+$21], as shown in Figure 4.1.

4.2 Related Works

Our work is inspired by the recent efforts [DBK$^+$21, TCD$^+$21] to realize Transformer-based image classifiers. ViT [DBK$^+$21] demonstrates the feasibility of building Transformer-based image classifiers from scratch, but its performance on ImageNet [RDS$^+$15] is not achieved without including additional training data; DeiT [TCD$^+$21] attains results comparable to convolution-based classifiers by using an effective training strategy together with model distillation, removing the data requirement in [DBK$^+$21]. Both ViT [DBK$^+$21] and DeiT [TCD$^+$21] are however based on a single image grid of fixed patch size.

The development of our co-scale conv-attentional Transformers (CoaT) is motivated by two observations: (1) multi-scale modeling typically brings enhanced capability to representation learning [HZRS16, RFB15, WSC$^+$20]; (2) the intrinsic connection between relative position encoding and convolution makes it possible to carry out efficient self-attention using conv-like operations. As a consequence, the superior performance of the CoaT models shown in the experiments comes from two of our new designs in Transformers: (1) a co-scale mechanism that allows cross-scale interaction; (2) a conv-attention module to realize an efficient self-attention operation. Next, we highlight the differences of the two proposed modules with the standard operations and concepts.

- **Co-Scale vs. Multi-Scale**. Multi-scale approaches have a long history in computer vision [Wit84, Low04]. Convolutional neural networks [LBBH98, KSH12, HZRS16] naturally implement a fine-to-coarse strategy. U-Net [RFB15] enforces an extra coarse-to-fine route in addition to the standard fine-to-coarse path; HRNet [WSC$^+$20] provides a further enhanced modeling capability by keeping simultaneous fine and coarse scales throughout the convolution layers. In a parallel development [WXL+21] to ours, layers of different scales are in tandem for the image Transformers but [WXL+21] merely carries out a fine-to-coarse strategy. The co-scale mechanism proposed here differs from the existing methods in how the responses are computed and interact with each other: CoaT consists of a series of highly modularized serial and parallel blocks to enable attention with fine-to-coarse, coarse-to-fine, and cross-scale information on tokenized representations. The joint attention mechanism across different scales in our co-scale module provides enhanced modeling power beyond existing vision Transformers [DBK+21, TCD+21, WXL+21].

- **Conv-Attention vs. Attention.** Pure attention-based models [RPV+19, HZXL19, ZJK20, DBK+21, TCD+21] have been introduced to the vision domain. [RPV+19, HZXL19, ZJK20] replace convolutions in ResNet-like architectures with self-attention modules for better local and non-local relation modeling. In contrast, [DBK+21, TCD+21] directly adapt the Transformer [VSP+17] for image recognition. Recently, there have been works [Bel21, CZT+21] enhancing the attention mechanism by introducing convolution. LambdaNets [Bel21] introduce an efficient self-attention alternative for global context modeling and employ convolutions to realize the relative position embeddings in local context modeling. CPVT [CZT+21] designs 2-D depthwise convolutions as the conditional positional encoding after self-attention. In our conv-attention, we: (1) adopt an efficient factorized attention following [Bel21]; (2) extend it to be a combination of depthwise convolutional relative position encoding and convolutional position encoding, related to CPVT [CZT+21]. Detailed discussion of our network design and its relation with LambdaNets [Bel21] and CPVT [CZT+21] can be found in Section 4.4.1 and 4.4.2.

4.3 Revisit Scaled Dot-Product Attention

Transformers take as input a sequence of vector representations (i.e. tokens) $\mathbf{x}_1, ..., \mathbf{x}_N$, or equivalently $X \in \mathbb{R}^{N \times C}$. The self-attention mechanism in Transformers projects each $\mathbf{x}_i$ into corresponding query, key, and value vectors, using learned linear transformations W^Q, W^K, and $W^V \in \mathbb{R}^{C \times C}$. Thus, the projection of the whole sequence generates representations $Q, K, V \in \mathbb{R}^{N \times C}$. The *scaled dot-product attention* from original Transformers [VSP+17] is formulated as :

$$\mathrm{Att}(X) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{C}}\right)V \tag{4.1}$$

In vision Transformers [DBK+21, TCD+21], the input sequence of vectors is formulated by the concatenation of a class token `CLS` and the flattened feature vectors $\mathbf{x}_1, ..., \mathbf{x}_{HW}$ as image tokens from the feature maps $F \in \mathbb{R}^{H \times W \times C}$, for a total length of $N = HW + 1$. The softmax logits in Equation 4.1 become not affordable for high-resolution images (i.e. $N \gg C$) due to its $O(N^2)$ space complexity and $O(N^2C)$ time complexity. To reduce the length of the sequence, ViT [DBK+21, TCD+21] tokenizes the image by patches instead of pixels. However, the coarse splitting (e.g. 16×16 patches) limits the ability to model details within each patch. To address this dilemma, we propose a *co-scale* mechanism that provides enhanced multi-scale image representation with the help of an efficient *conv-attentional* module that lowers the computation complexity for high-resolution images.

4.4 Conv-Attention Module

4.4.1 Factorized Attention Mechanism

In Equation 4.1, the materialization of the softmax logits and attention maps leads to the $O(N^2)$ space complexity and $O(N^2C)$ time complexity. Inspired by recent works [CLD+21, SZZ+21, Bel21] on linearization of self-attention, we approximate the softmax attention map

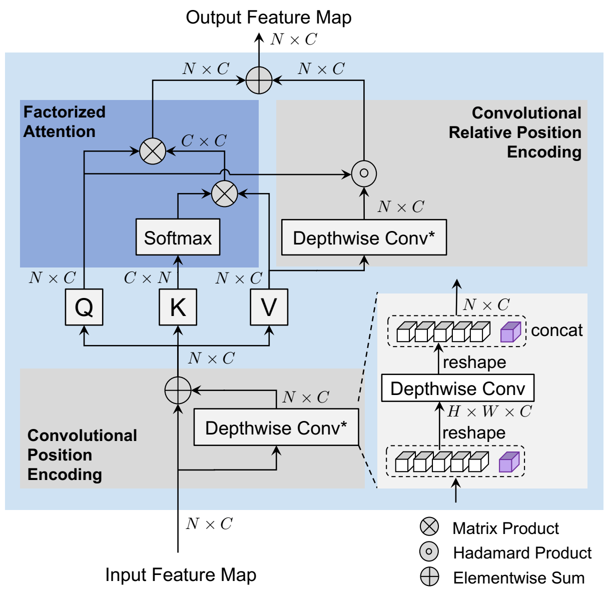


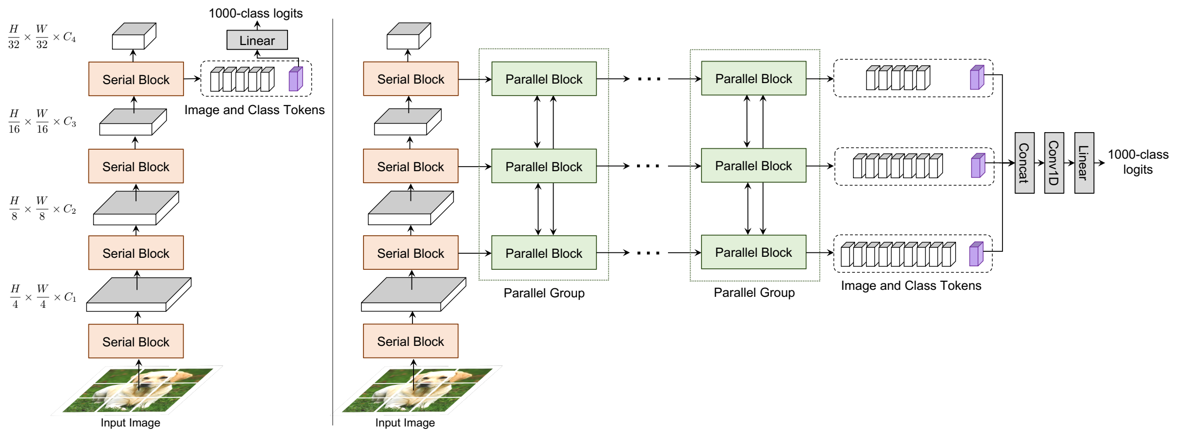
Figure 4.2: **Illustration of the conv-attentional module.** We apply a convolutional position encoding to the image tokens from the input. The resulting features are fed into a factorized attention with a convolutional relative position encoding.

by factorizing it using two functions $\phi(\cdot), \psi(\cdot) : \mathbb{R}^{N\times C} \to \mathbb{R}^{N\times C'}$ and compute the second matrix multiplication (keys and values) together:

$$\text{FactorAtt}(X) = \phi(Q)\left(\psi(K)^\top V\right) \tag{4.2}$$

The factorization leads to a $O(NC' + NC + CC')$ space complexity (including output of $\phi(\cdot), \psi(\cdot)$ and intermediate steps in the matrix product) and $O(NCC')$ time complexity, where both are linear functions of the sequence length N. Performer [CLD+21] uses random projections in ϕ and ψ for a provable approximation, but with the cost of relatively large C'. Efficient-Attention [SZZ+21] applies the softmax function for both ϕ and ψ, which is efficient but causes a significant performance drop on the vision tasks in our experiments. Here, we develop our factorized attention mechanism following LambdaNets [Bel21] with ϕ as the identity function and ψ as the softmax:

$$\text{FactorAtt}(X) = \frac{Q}{\sqrt{C}}\left(\text{softmax}(K)^\top V\right) \tag{4.3}$$



$\frac{H}{32} \times \frac{W}{32} \times C_4$
$\frac{H}{16} \times \frac{W}{16} \times C_3$
$\frac{H}{8} \times \frac{W}{8} \times C_2$
$\frac{H}{4} \times \frac{W}{4} \times C_1$
1000-class logits
Linear
Serial Block
Image and Class Tokens
Input Image
Parallel Block
Parallel Group
Concat
Conv1D

Figure 4.3: **CoaT model architecture.** (Left) The overall network architecture of **CoaT-Lite**. CoaT-Lite consists of serial blocks only, where image features are down-sampled and processed in a sequential order. (Right) The overall network architecture of **CoaT**. CoaT consists of serial blocks and parallel blocks. Both blocks enable the co-scale mechanism.

where softmax$(\cdot)$ is applied across the tokens in the sequence in an element-wise manner and the projected channels $C' = C$. In LambdaNets [Bel21], the scaling factor $1/\sqrt{C}$ is implicitly included in the weight initialization, while our factorized attention applies the scaling factor explicitly. This factorized attention takes $O(NC + C^2)$ space complexity and $O(NC^2)$ time complexity. It is noteworthy that the proposed factorized attention following [Bel21] is not a direct approximation of the scaled dot-product attention, but it can still be regarded as a generalized attention mechanism modeling the feature interactions using query, key and value vectors.

4.4.2 Convolution as Position Encoding

Our factorized attention module mitigates the computational burden from the original scaled dot-product attention. However, because we compute $L = \text{softmax}(K)^\top V \in \mathbb{R}^{C \times C}$ first, L can be seen as a global data-dependent linear transformation for every feature vector in the query map Q. This indicates that if we have two query vectors $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{R}^C$ from Q and $\mathbf{q}_1 = \mathbf{q}_2$, then their corresponding self-attention outputs will be the same:

$$\text{FactorAtt}(X)_1 = \frac{\mathbf{q}_1}{\sqrt{C}} L = \frac{\mathbf{q}_2}{\sqrt{C}} L = \text{FactorAtt}(X)_2 \tag{4.4}$$

Without the position encoding, the Transformer is only composed of linear layers and self-attention modules. Thus, the output of a token is dependent on the corresponding input without awareness of any difference in its locally nearby features. This property is unfavorable for vision tasks such as semantic segmentation (e.g. the same blue patches in the sky and the sea are segmented as the same category).

Convolutional Relative Position Encoding. To enable vision tasks, ViT and DeiT [DBK+21, TCD+21] insert absolute position embeddings into the input, which may have limitations in modeling relations between local tokens. Instead, following [SUV18], we can integrate a relative position encoding $P = \{\mathbf{p}_i \in \mathbb{R}^C, i = -\frac{M-1}{2}, ..., \frac{M-1}{2}\}$ with window size M to obtain the relative attention map $EV \in \mathbb{R}^{N \times C}$; in attention formulation, if tokens are regarded as a 1-D sequence:

$$\text{RelFactorAtt}(X) = \frac{Q}{\sqrt{C}} \Big(\text{softmax}(K)^\top V \Big) + EV \tag{4.5}$$

where the encoding matrix $E \in \mathbb{R}^{N \times N}$ has elements:

$$E_{ij} = \mathbb{1}(i,j)\mathbf{q}_i \cdot \mathbf{p}_{j-i}, \;\; 1 \leq i, j \leq N \tag{4.6}$$

in which $\mathbb{1}(i,j) = \mathbb{1}_{\{|j-i| \leq (M-1)/2\}}(i,j)$ is an indicator function. Each element E_{ij} represents the relation from query $\mathbf{q}_i$ to the value $\mathbf{v}_j$ within window M, and $(EV)_i$ aggregates all related value vectors with respect to query $\mathbf{q}_i$. Unfortunately, the EV term still requires $O(N^2)$ space complexity and $O(N^2C)$ time complexity. In CoaT, we propose to simplify the EV term to $\hat{EV}$ by considering each channel in the query, position encoding and value vectors as *internal heads*.

Thus, for each internal head l, we have:

$$E_{ij}^{(l)} = \mathbb{1}(i,j) q_i^{(l)} p_{j-i}^{(l)}, \ \ \hat{EV}_i^{(l)} = \textstyle\sum_j E_{ij}^{(l)} v_j^{(l)} \tag{4.7}$$

In practice, we can use a 1-D depthwise convolution to compute $\hat{EV}$:

$$\hat{EV}^{(l)} = Q^{(l)} \circ \text{Conv1D}(P^{(l)}, V^{(l)}), \tag{4.8}$$

$$\hat{EV} = Q \circ \text{DepthwiseConv1D}(P, V) \tag{4.9}$$

where $\circ$ is the Hadamard product. It is noteworthy that in vision Transformers, we have two types of tokens, the class (`CLS`) token and image tokens. Thus, we use a 2-D depthwise convolution (with window size $M \times M$ and kernel weights P) and apply it only to the reshaped image tokens (i.e. $Q^{\text{img}}, V^{\text{img}} \in \mathbb{R}^{H \times W \times C}$ from Q, V respectively):

$$\hat{EV}^{\text{img}} = Q^{\text{img}} \circ \text{DepthwiseConv2D}(P, V^{\text{img}}) \tag{4.10}$$

$$\hat{EV} = \text{concat}(\hat{EV}^{\text{img}}, \mathbf{0}) \tag{4.11}$$

$$\text{ConvAtt}(X) = \frac{Q}{\sqrt{C}} \left(\text{softmax}(K)^\top V \right) + \hat{EV} \tag{4.12}$$

Based on our derivation, the depthwise convolution can be seen as a special case of relative position encoding.

Convolutional Relative Position Encoding vs Other Relative Position Encodings. The commonly referenced relative position encoding [SUV18] works in standard scaled dot-product attention settings since the encoding matrix E is combined with the softmax logits in the attention maps, which are not materialized in our factorized attention. Related to our work, the main results of the original LambdaNets [Bel21] use a 3-D convolution to compute EV directly and reduce the channels of queries and keys to C_K where $C_K < C$, but it costs $O(NCC_K)$ space complexity and $O(NCC_K M^2)$ time complexity, which leads to relatively heavy computation when channel

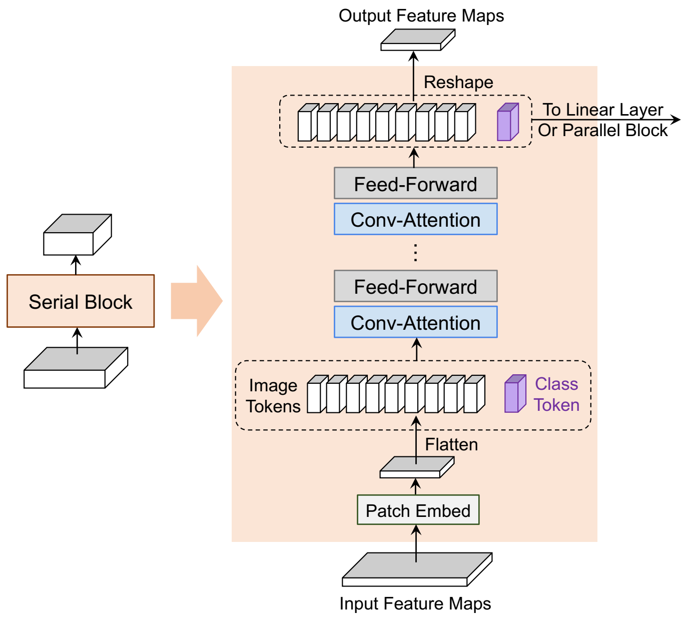


Figure 4.4: **Schematic illustration of the serial block in CoaT.** Input feature maps are first down-sampled by a patch embedding layer, and then tokenized features (along with a class token) are processed by multiple conv-attention and feed-forward layers.

sizes C_K, C are large. A recent update in LambdaNets [Bel21] provides an efficient variant with depth-wise convolution under resource constrained scenarios. Our factorized attention computes $E\hat{V}$ with only $O(NC)$ space complexity and $O(NCM^2)$ time complexity, aiming to achieve better efficiency.

Convolutional Position Encoding. We then extend the idea of convolutional relative position encoding to a general convolutional position encoding case. Convolutional relative position encoding models local position-based relationships between queries and values. Similar to the absolute position encoding used in most image Transformers [DBK+21, TCD+21], we would like to insert the position relationship into the input image features directly to enrich the effects of relative position encoding. In each conv-attentional module, we insert a depthwise convolution into the input features X and concatenate the resulting position-aware features back to the input features following the standard absolute position encoding scheme (see Figure 4.2 lower part), which resembles the realization of conditional position encoding in CPVT [CZT+21].

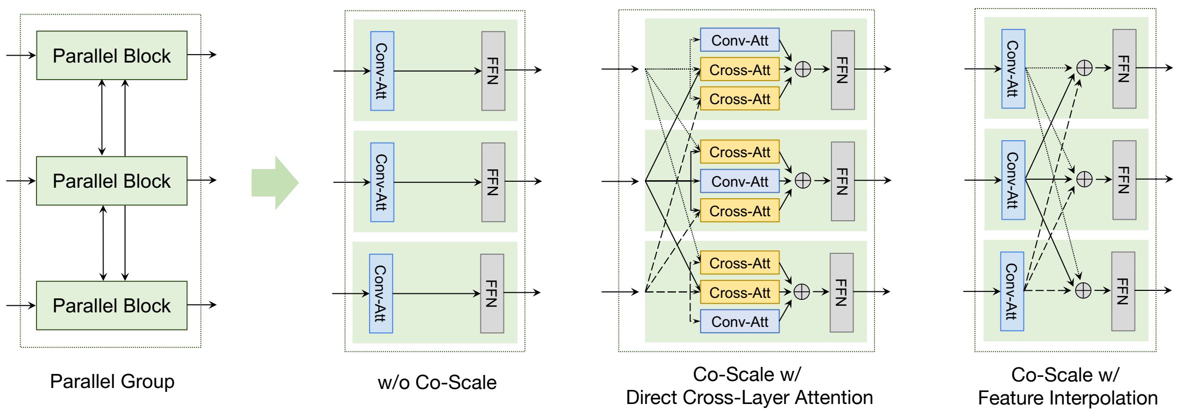


Figure 4.5: **Schematic illustration of the parallel group in CoaT.** For “w/o Co-Scale”, tokens learned at the individual scales are combined to perform the classification but absent intermediate co-scale interaction for the individual paths of the parallel blocks. We propose two co-scale variants, namely direct cross-layer attention and attention with feature interpolation. Co-scale with feature interpolation is adopted in the final CoaT-Lite and CoaT models reported on the ImageNet benchmark.

CoaT and CoaT-Lite share the convolutional position encoding weights and convolutional relative position encoding weights for the serial and parallel modules within the same scale. We set convolution kernel size to 3 for the convolutional position encoding. We set convolution kernel size to 3, 5 and 7 for image features from different attention heads for convolutional relative position encoding.

The work of CPVT [CZT$^+$21] explores the use of convolution as conditional position encodings by inserting it after the feed-forward network under a single scale ($\frac{H}{16} \times \frac{W}{16}$). Our work focuses on applying convolution as relative position encoding and a general position encoding with the factorized attention.

Conv-Attentional Mechanism The final conv-attentional module is shown in Figure 4.2: We apply the first convolutional position encoding on the image tokens from the input. Then, we feed it into $\mathrm{ConvAtt}(\cdot)$ including factorized attention and the convolutional relative position encoding. The resulting map is used for the subsequent feed-forward networks.

Table 4.1: **Architecture details of CoaT-Lite and CoaT models.** C_i represents the hidden dimension of the attention layers in block i; H_i represents the number of attention heads in the attention layers in block i; R_i represents the expansion ratio for the feed-forward hidden layer dimension between attention layers in block i. Multipliers indicate the number of conv-attentional modules in block i.

Blocks	Output	CoaT-Lite				CoaT		
		Tiny	Mini	Small	Medium	Tiny	Mini	Small
Serial Block (S_1)	56×56	$\begin{bmatrix} C_1 = 64 \\ H_1 = 8 \\ R_1 = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 64 \\ H_1 = 8 \\ R_1 = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 64 \\ H_1 = 8 \\ R_1 = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} C_1 = 128 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} C_1 = 152 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 152 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 152 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 2$
Serial Block (S_2)	28×28	$\begin{bmatrix} C_2 = 128 \\ H_2 = 8 \\ R_2 = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C_2 = 128 \\ H_2 = 8 \\ R_2 = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C_2 = 128 \\ H_2 = 8 \\ R_2 = 8 \end{bmatrix} \times 4$	$\begin{bmatrix} C_1 = 256 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 6$	$\begin{bmatrix} C_2 = 152 \\ H_2 = 8 \\ R_2 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_2 = 216 \\ H_2 = 8 \\ R_2 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 320 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 2$
Serial Block (S_3)	14×14	$\begin{bmatrix} C_3 = 256 \\ H_3 = 8 \\ R_3 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_3 = 320 \\ H_3 = 8 \\ R_3 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_3 = 320 \\ H_3 = 8 \\ R_3 = 4 \end{bmatrix} \times 6$	$\begin{bmatrix} C_1 = 320 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 10$	$\begin{bmatrix} C_3 = 152 \\ H_3 = 8 \\ R_3 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_3 = 216 \\ H_3 = 8 \\ R_3 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 320 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 2$
Serial Block (S_4)	7×7	$\begin{bmatrix} C_4 = 320 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_4 = 512 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_4 = 512 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} C_1 = 512 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 8$	$\begin{bmatrix} C_4 = 152 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_4 = 216 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C_1 = 320 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 2$
Parallel Group	$\begin{bmatrix} 28 \times 28 \\ 14 \times 14 \\ 7 \times 7 \end{bmatrix}$					$\begin{bmatrix} C_4 = 152 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 6$	$\begin{bmatrix} C_4 = 216 \\ H_4 = 8 \\ R_4 = 4 \end{bmatrix} \times 6$	$\begin{bmatrix} C_1 = 320 \\ H_1 = 8 \\ R_1 = 4 \end{bmatrix} \times 6$
#Params		5.7M	11M	20M	45M	5.5M	10M	22M

4.5 Co-Scale Conv-Attentional Transformers

4.5.1 Co-Scale Mechanism

The proposed co-scale mechanism is designed to introduce fine-to-coarse, coarse-to-fine and cross-scale information into image Transformers. Here, we describe two types of building blocks in the CoaT architecture, namely serial and parallel blocks, in order to model multiple scales and enable the co-scale mechanism.

CoaT Serial Block. A serial block (shown in Figure 4.4) models image representations in a reduced resolution. In a typical serial block, we first down-sample input feature maps by a certain ratio using a patch embedding layer, and flatten the reduced feature maps into a sequence of image tokens. We then concatenate image tokens with an additional `CLS` token, a specialized vector to perform image classification, and apply multiple conv-attentional modules as described in Section 4.4 to learn internal relationships among image tokens and the `CLS` token. Finally, we separate the `CLS` token from the image tokens and reshape the image tokens to 2-D feature maps for the next serial block.

CoaT Parallel Block. We realize a co-scale mechanism between parallel blocks in each parallel group (shown in Figure 4.5). In a typical parallel group, we have sequences of input features (image tokens and `CLS` token) from serial blocks with different scales. To enable fine-to-coarse, coarse-to-fine, and cross-scale interaction in the parallel group, we develop two strategies: (1) direct cross-layer attention; (2) attention with feature interpolation. In this work, we adopt attention with feature interpolation for better empirical performance. The effectiveness of both strategies is shown in Section 4.6.4.

Direct cross-layer attention. In direct cross-layer attention, we form query, key, and value vectors from input features for each scale. For attention within the same layer, we use the conv-attention (Figure 4.2) with the query, key and value vectors from current scale. For attention across different layers, we down-sample or up-sample the key and value vectors to match the resolution of other scales, which enables fine-to-coarse and coarse-to-fine interaction. We then perform cross-attention, which extends the conv-attention with queries from the current scale with keys and values from another scale. Finally, we sum the outputs of conv-attention and cross-attention together and apply a shared feed-forward layer. With direct cross-layer attention, the cross-scale information is fused in a cross-attention fashion.

Attention with feature interpolation. Instead of performing cross-layer attention directly, we also present attention with feature interpolation. First, the input image features from different scales are processed by independent conv-attention modules. Then, we down-sample or up-sample image features from each scale to match the dimensions of other scales using bilinear interpolation, or keep the same for its own scale. The features belonging to the same scale are summed in the parallel group, and they are further passed into a shared feed-forward layer. In this way, the conv-attentional module in the next step can learn cross-scale information based on the feature interpolation in the current step.

4.5.2 Model Architecture

CoaT-Lite. CoaT-Lite, Figure 4.3 (Left), processes input images with a series of serial blocks following a fine-to-coarse pyramid structure. Given an input image $I \in \mathbb{R}^{H \times W \times C}$, each serial block down-samples the image features into lower resolution, resulting in a sequence of four resolutions:$F_1 \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times C_1}$, $F_2 \in \mathbb{R}^{\frac{H}{8} \times \frac{W}{8} \times C_2}$, $F_3 \in \mathbb{R}^{\frac{H}{16} \times \frac{W}{16} \times C_3}$, $F_4 \in \mathbb{R}^{\frac{H}{32} \times \frac{W}{32} \times C_4}$. In CoaT-Lite, we obtain the `CLS` token in the last serial block, and perform image classification via a linear projection layer based on the `CLS` token.

CoaT. Our CoaT model, shown in Figure 4.3 (Right), consists of both serial and parallel blocks. Once we obtain multi-scale feature maps $\{F_1, F_2, F_3, F_4\}$ from the serial blocks, we pass F_2, F_3, F_4 and corresponding `CLS` tokens into the parallel group with three separate parallel blocks. To perform classification with CoaT, we aggregate the `CLS` tokens from all three scales.

Model Variants. In this work, we explore CoaT and CoaT-Lite with several different model sizes, namely Tiny, Mini, Small and Medium. Architecture details are shown in Table 4.1. For example, tiny models represent those with a 5M parameter budget constraint. Specifically, these tiny models have four serial blocks, each with two conv-attentional modules. In CoaT-Lite Tiny architectures, the hidden dimensions of the attention layers increase in later blocks. CoaT Tiny sets the hidden dimensions of the attention layers in the parallel group to be equal, and performs the co-scale mechanism within the six parallel groups. Mini, small and medium models follow the same architecture design but with increased embedding dimensions and increased numbers of conv-attentional modules within blocks.

Table 4.2: **CoaT performance on ImageNet-1K validation set.** Our CoaT models consistently outperform other methods while being parameter efficient. ConvNets and ViTNets with similar model size are grouped together for comparison. "#GFLOPs" and Top-1 Acc are measured at input image size. "*" results are adopted from [WXL+21].

Arch.	Model	#Params	Input	#GFLOPs	Top-1 Acc.
ConvNets	EfficientNet-B0 [TL19]	5.3M	224^2	0.4	77.1%
	ShuffleNet [ZZLS18]	5.4M	224^2	0.5	73.7%
ViTNets	DeiT-Tiny [TCD+21]	5.7M	224^2	1.3	72.2%
	CPVT-Tiny [CZT+21]	5.7M	224^2	-	73.4%
	CoaT-Lite Tiny (Ours)	5.7M	224^2	1.6	77.5%
	CoaT Tiny (Ours)	5.5M	224^2	4.4	**78.3%**
ConvNets	EfficientNet-B2[TL19]	9M	260^2	1.0	80.1%
	ResNet-18* [HZRS16]	12M	224^2	1.8	69.8%
ViTNets	PVT-Tiny [WXL+21]	13M	224^2	1.9	75.1%
	CoaT-Lite Mini (Ours)	11M	224^2	2.0	79.1%
	CoaT Mini (Ours)	10M	224^2	6.8	**81.0%**
ConvNets	EfficientNet-B4 [TL19]	19M	380^2	4.2	**82.9%**
	ResNet-50* [HZRS16]	25M	224^2	4.1	78.5%
	ResNeXt50-32x4d* [XGD+17]	25M	224^2	4.3	79.5%
ViTNets	DeiT-Small [TCD+21]	22M	224^2	4.6	79.8%
	PVT-Small [WXL+21]	24M	224^2	3.8	79.8%
	CPVT-Small [CZT+21]	22M	224^2	-	80.5%
	T2T-ViT$_t$-14 [YCW+21]	22M	224^2	6.1	81.7%
	Swin-T [LLC+21]	29M	224^2	4.5	81.3%
	CoaT-Lite Small (Ours)	20M	224^2	4.0	81.9%
	CoaT Small (Ours)	22M	224^2	12.6	82.1%
ConvNets	EfficientNet-B6 [TL19]	43M	528^2	19	84.0%
	ResNet-101* [HZRS16]	45M	224^2	7.9	79.8%
	ResNeXt101-64x4d* [XGD+17]	84M	224^2	15.6	81.5%
ViTNets	PVT-Large [WXL+21]	61M	224^2	9.8	81.7%
	T2T-ViT$_t$-24 [YCW+21]	64M	224^2	15	82.6%
	DeiT-Base [TCD+21]	86M	224^2	17.6	81.8%
	CPVT-Base [CZT+21]	86M	224^2	-	82.3%
	Swin-B [LLC+21]	88M	224^2	15.4	83.5%
	Swin-B [LLC+21]	88M	384^2	47	**84.5%**
	CoaT-Lite Medium (Ours)	45M	224^2	9.8	83.6%
	CoaT-Lite Medium (Ours)	45M	384^2	28.7	**84.5%**

Table 4.3: **Object detection and instance segmentation results based on Mask R-CNN on COCO val2017**. Experiments are performed under the MMDetection framework [CWP+19]. "*" results are adopted from Detectron2.

Backbone	#Params (M)	w/ FPN 1×		w/ FPN 3×	
		AP^b	AP^m	AP^b	AP^m
ResNet-18*	31.3	34.2	31.3	36.3	33.2
PVT-Tiny [WXL+21]	32.9	36.7	35.1	39.8	37.4
CoaT-Lite Mini (Ours)	30.7	41.4	38.0	42.9	38.9
CoaT Mini (Ours)	30.2	**45.1**	**40.6**	**46.5**	**41.8**
ResNet-50*	44.3	38.6	35.2	41.0	37.2
PVT-Small [WXL+21]	44.1	40.4	37.8	43.0	39.9
Swin-T [LLC+21]	47.8	43.7	39.8	46.0	41.6
CoaT-Lite Small (Ours)	39.5	45.2	40.7	45.7	41.1
CoaT Small (Ours)	41.6	**46.5**	**41.8**	**49.0**	**43.7**

Table 4.4: **Object detection and instance segmentation results based on Cascade Mask R-CNN on COCO val2017.** Experiments are performed using the MMDetection framework [CWP+19].

Backbone	#Params (M)	w/ FPN 1×		w/ FPN 3×	
		AP^b	AP^m	AP^b	AP^m
Swin-T [LLC+21]	85.6	48.1	41.7	50.4	43.7
CoaT-Lite Small (Ours)	77.3	49.1	42.5	48.9	42.6
CoaT Small (Ours)	79.4	**50.4**	**43.5**	**52.2**	**45.1**

Table 4.5: **Object detection results based on Deformable DETR on COCO val2017.** DD ResNet-50 represents the baseline result using the official checkpoint. ResNet-50 and our CoaT-Lite as DD backbones are directly comparable due to similar model size.

Backbone	Deformable DETR (Multi-Scale)					
	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
DD ResNet-50 [ZSL+21]	44.5	63.7	48.7	26.8	47.6	59.6
DD **CoaT-Lite Small** (Ours)	47.0	66.5	51.2	28.8	50.3	63.3
DD **CoaT Small** (Ours)	**48.4**	**68.5**	**52.4**	**30.2**	**51.8**	**63.8**

4.6 Experiments

4.6.1 Experiment Details

Image Classification. We perform image classification on the standard ILSVRC-2012 ImageNet dataset [RDS+15]. The standard ImageNet benchmark contains 1.3 million images in the training set and 50K images in the validation set, covering 1000 object classes. Image cropping sizes are set to 224×224. For fair comparison, we perform data augmentation such as MixUp [ZCDLP18], CutMix [YHO$^+$19], random erasing [ZZK$^+$20], repeated augmentation [HBNH$^+$20], and label smoothing [SVI$^+$16], following identical procedures in DeiT [TCD$^+$21].

All experimental results for our models in Table 4.2 are reported at 300 epochs, consistent with previous methods [TCD$^+$21]. All models are trained with the AdamW [LH19] optimizer under the NVIDIA Automatic Mixed Precision (AMP) framework. The learning rate is scaled as $5 \times 10^{-4} \times \frac{\text{global batch size}}{512}$.

Object Detection and Instance Segmentation. We conduct object detection and instance segmentation experiments on the Common Objects in Context (COCO2017) dataset [LMB$^+$14]. The COCO2017 benchmark contains 118K training images and 5K validation images. We evaluate the generalization ability of CoaT in object detection and instance segmentation with the Mask R-CNN [HGDG17] and Cascade Mask R-CNN [CV19]. We use the MMDetection [CWP$^+$19] framework and follow the settings from Swin Transformers [LLC$^+$21]. In addition, we perform object detection based on Deformable DETR [ZSL$^+$21] following its data processing settings.

For Mask R-CNN optimization, we train the model with the ImageNet-pretrained backbone on 8 GPUs via AdamW optimizer with learning rate 0.0001. The training period contains 12 epochs in $1\times$ setting and 36 epochs in $3\times$ setting. For Cascade R-CNN experiments, we use three detection heads, with the same optimization and training period as Mask R-CNN. For Deformable DETR optimization, we train the model with the pretrained backbone for 50 epochs, using an AdamW optimizer with initial learning rate 2×10^{-4}, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. We reduce the learning rate by a factor of 10 at epoch 40.

4.6.2 CoaT for ImageNet Classification

Table 4.2 shows top-1 accuracy results for our models on the ImageNet validation set comparing with state-of-the-art methods. We separate model architectures into two categories: convolutional networks (ConvNets), and Transformers (ViTNets). Under different parameter budget constraints, CoaT and CoaT-Lite show strong results compared to other ConvNet and ViTNet methods.

4.6.3 Object Detection and Instance Segmentation

Tables 4.3 and 4.4 demonstrate CoaT object detection and instance segmentation results under the Mask R-CNN and Cascade Mask R-CNN frameworks on the COCO val2017 dataset. Our CoaT and CoaT-Lite models show clear performance advantages over the ResNet, PVT [WXL+21] and Swin [LLC+21] backbones under both the $1\times$ setting and the $3\times$ setting. In particular, our CoaT models bring a large performance gain, demonstrating that our co-scale mechanism is essential to improve the performance of Transformer-based architectures for downstream tasks.

We additionally perform object detection with the Deformable DETR (DD) framework in Table 4.5. We compare our models with the standard ResNet-50 backbone on the COCO dataset [LMB+14]. Our CoaT backbone achieves 3.9% improvement on average precision (AP) over the results of Deformable DETR with ResNet-50 [ZSL+21].

4.6.4 Ablation Study

Effectiveness of Position Encodings. We study the effectiveness of the combination of the convolutional relative position encoding (CRPE) and convolutional position encoding (CPE) in our conv-attention module in Table 4.6. Our CoaT-Lite without any position encoding results in poor performance, indicating that position encoding is essential for vision Transformers. We observe great improvement for CoaT-Lite variants with either CRPE or CPE, and the combination of CRPE and CPE leads to the best performance (77.5% top-1 accuracy), making both position encoding schemes complementary rather than conflicting.

Table 4.6: **Effectiveness of position encodings.** All experiments are performed with the CoaT-Lite Tiny architecture. Performance is evaluated on the ImageNet-1K validation set.

Model	CPE	CRPE	Top-1 Acc.
CoaT-Lite Tiny	✗	✗	68.8%
	✗	✓	75.0%
	✓	✗	75.9%
	✓	✓	77.5%

Effectiveness of Co-Scale. In Table 4.7, we present performance results for two co-scale variants in CoaT, direct cross-layer attention and attention with feature interpolation. We also report CoaT without co-scale as a baseline. Comparing to CoaT without a co-scale mechanism, CoaT with feature interpolation shows performance improvements on both image classification and object detection (Mask R-CNN w/ FPN 1$\times$). Attention with feature interpolation offers a clear advantage over direct cross-layer attention due to less computational complexity and higher accuracy.

Table 4.7: **Effectiveness of co-scale.** All experiments are performed with the CoaT Tiny architecture. Performance is evaluated on the ImageNet-1K validation set and the COCO val2017 dataset.

Model	#Params	Input	#GFLOPs	Top-1 Acc. @input	AP^b	AP^m
CoaT w/o Co-Scale	5.5M	224^2	4.4	77.8%	41.6	37.9
CoaT w/ Co-Scale						
- Direct Cross-Layer Attention	5.5M	224^2	4.8	77.0%	42.1	38.3
- Attention w/ Feature Interp.	5.5M	224^2	4.4	78.3%	42.5	38.6

Computational Cost. We report FLOPs, FPS, latency, and GPU memory usage in Table 4.8. In summary, CoaT models attain higher accuracy than similar-sized Swin Transformers, but CoaT models in general do have larger latency/FLOPs. The current parallel groups in CoaT are more computationally demanding, which can be mitigated by reducing high-resolution parallel blocks and re-using their feature maps in the co-scale mechanism in future work. The latency overhead in CoaT is possibly because operations (e.g. layers, position encodings, upsamples/downsamples) are not running in parallel.

Table 4.8: **ImageNet-1K validation set results** compared with the concurrent work Swin Transformer[LLC+21]. Computational metrics are measured on a single V100 GPU.

Model	#Params	Input	GFLOPs	FPS	Latency	Mem	Top-1 Acc.	Top-5 Acc.
Swin-T [LLC+21]	28M	224^2	4.5	755	16ms	222M	81.2%	95.5%
CoaT-Lite Small (Ours)	20M	224^2	4.0	634	32ms	224M	81.9%	95.6%
CoaT Small (Ours)	22M	224^2	12.6	111	60ms	371M	**82.1%**	**96.1%**
Swin-S [LLC+21]	50M	224^2	8.7	437	29ms	372M	83.2%	96.2%
Swin-B [LLC+21]	88M	224^2	15.4	278	30ms	579M	83.5%	96.5%
CoaT-Lite Medium (Ours)	45M	224^2	9.8	319	52ms	429M	**83.6%**	**96.7%**
Swin-B [LLC+21]	88M	384^2	47.1	85	33ms	1250M	**84.5%**	97.0%
CoaT-Lite Medium (Ours)	45M	384^2	28.7	97	56ms	937M	**84.5%**	**97.1%**

4.7 Conclusion

In this work, we present a Transformer based image classifier, Co-scale conv-attentional image Transformer (CoaT), in which cross-scale attention and efficient conv-attention operations have been developed. CoaT models attain strong classification results on ImageNet, and their applicability to downstream computer vision tasks has been demonstrated for object detection and instance segmentation.

Acknowledgements

This chapter is based on the material "Co-Scale Conv-Attentional Image Transformers" by Weijian Xu*, Yifan Xu*, Tyler Chang, and Zhuowen Tu, which appears in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) 2021. The dissertation author is the co-primary investigator and author of this material.

Chapter 5

Line Segment Detection Using Transformers without Edges

5.1 Introduction

Line segment detection is an important mid-level visual process [Mar82] useful for solving various downstream computer vision tasks, including segmentation, 3D reconstruction, image matching and registration, depth estimation, scene understanding, object detection, image editing, and shape analysis. Despite its practical and scientific importance, line segment detection remains an unsolved problem in computer vision.

Although dense pixel-wise edge detection has achieved an impressive performance [XT15], reliably extracting line segments of semantic and perceptual significance remains a further challenge. In natural scenes, line segments of interest often have heterogeneous structures within the cluttered background that are locally ambiguous or partially occluded. Morphological operators [SB97] operated on detected edges [Can86] often give sub-optimal results. Mid-level representations such as Gestalt laws [EG02] and contextual information [Tu08] can play an important role in the perceptual grouping, but they are often hard to be

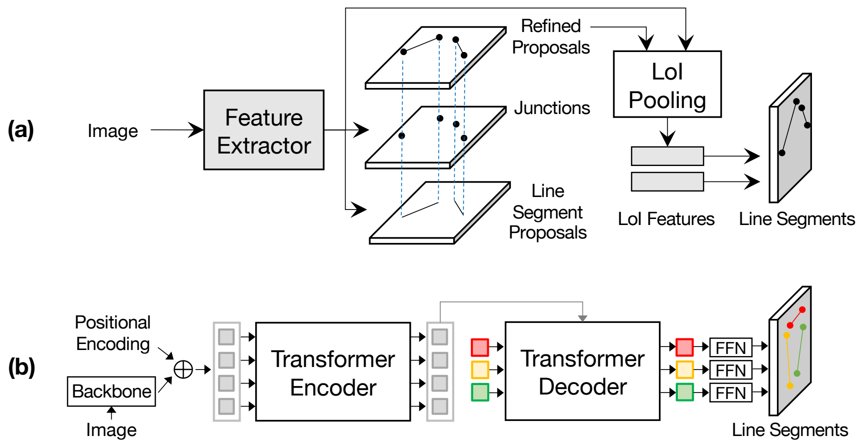


Figure 5.1: **Pipeline comparison** between: (a) holistically-attracted wireframe parsing (HAWP) [XWB$^+$20] and (b) our proposed LinE segment TRansformers (LETR). LETR is based on a general-purpose pipeline without heuristics-driven intermediate stages for detecting junctions and generating line segment proposals.

seamlessly integrated into an end-to-end line segment detection pipeline. Deep learning techniques [KSH12, LSD15, HZRS16, XT15] have provided greatly enhanced feature representation power, and algorithms such as [ZQM19, XBW$^+$19, XWB$^+$20] become increasingly feasible in real-world applications. However, systems like [ZQM19, XBW$^+$19, XWB$^+$20] still consist of heuristics-guided modules [SB97] such as edge/junction/region detection, line grouping, and post-processing, limiting the scope of their performance enhancement and further development.

In this work, we skip the traditional edge/junction/region detection + proposals + perceptual grouping pipeline by designing a Transformer-based [VSP$^+$17, CMS$^+$20] joint end-to-end line segment detection algorithm. We are motivated by the following observations for the Transformer frameworks [VSP$^+$17, CMS$^+$20]: tokenized queries with an integrated encoding and decoding strategy, self-attention mechanism, and bipartite (Hungarian) matching step, capable of addressing the challenges in line segment detection for edge element detection, perceptual grouping, and set prediction; general-purpose pipelines for Transformers that are heuristics free. Our system, named LinE segment TRsformer (LETR), enjoys the modeling power of a general-purpose Transformer architecture while having its own enhanced property for detecting fine-grained geometric structures like line segments. LETR is built on top of a seminal work, DEtection TRansformer (DETR) [CMS+20]. However, as shown in Section 5.4.4 for ablation studies, directly applying the DETR object detector [CMS+20] for line segment detection does not yield satisfactory results since line segments are elongated geometric structures that are not feasible for the bounding box representations.

Our contributions are summarized as follows.

- We cast the line segment detection problem in a joint end-to-end fashion without explicit edge/junction/region detection and heuristics-guided perceptual grouping processes, which is in distinction to the existing literature in this domain. We achieve state-of-the-art results on the Wireframe [HWZ+18] and YorkUrban benchmarks [DEE08].
- We perform line segment detection using Transformers, based specifically on DETR [CMS+20], to realize tokenized entity modeling, perceptual grouping, and joint detection via an integrated encoder-decoder, a self-attention mechanism, and joint query inference within Transformers.
- We introduce two new algorithmic aspects to DETR [CMS+20]: first, a multi-scale encoder/decoder strategy as shown in Figure 5.2; second, a direct endpoint distance loss term in training, allowing geometric structures like line segments to be directly learned and detected — something not feasible in the standard DETR bounding box representations.

5.2 Related Works

5.2.1 Line Segment Detection

Traditional Approaches. Line detection has a long history in computer vision. Early pioneering works rely on low-level cues from pre-defined features (e.g. image gradients). Typically, line (segment) detection performs edge detection [Can86, MFM04, DTB06, DZ13, XT15], followed by a perceptual grouping [GVZ95, SB97, EG02] process. Classic *perceptual grouping* frameworks [BHR86, BWR89, NCSG11, LYLL15, VGJMR08] aggregate the low-level cues to form line segments in a bottom-up fashion: an image is partitioned into line-support regions by grouping similar pixel-wise features. Line segments are then approximated from line-support regions and filtered by a validation step to remove false positives. Another popular series of line segment detection approaches are based on *Hough transform* [DH72, GVZ95, MGK00, FS03] by gathering votes in the parameter space: the pixel-wise edge map of an image is converted into a parameter space representation, in which each point corresponds to a unique parameterized line. The points in the parameter space that accumulate sufficient votes from the candidate edge pixels are identified as line predictions. However, due to the limitations in the modeling/inference processes, these traditional approaches often produce sub-optimal results.

Deep Learning Based Approaches. The recent surge of deep learning based approaches has achieved much-improved performance on the line segment detection problem [HWZ$^+$18, XBW$^+$19, ZQM19, ZLB$^+$19, XWB$^+$20] with the use of learnable features to capture extensive context information.

One typical family of methods is *junction-based pipelines*: Deep Wireframe Parser (DWP) [HWZ$^+$18] creates two parallel branches to predict the junction heatmap and the line heatmap, followed by a merging procedure. Motivated by [RHGS15], L-CNN [ZQM19] simplifies [HWZ$^+$18] into a unified network. First, a junction proposal module produces the junction heatmap and then converts detected junctions into line proposals. Second, a line verification module classifies proposals and removes unwanted false-positive lines. Methods like [ZQM19] are end-to-end, but they are at the instance-level (for detecting the individual line segments). Our LETR, like DETR [CMS$^+$20], has a general-purpose architecture that is trained in a holistically end-to-end fashion. PPGNet [ZLB$^+$19] proposes to create a point-set graph with junctions as vertices and model line segments as edges. However, the aforementioned approaches are heavily dependent on high-quality junction detection, which is error-prone to various imaging conditions and complex scenarios.

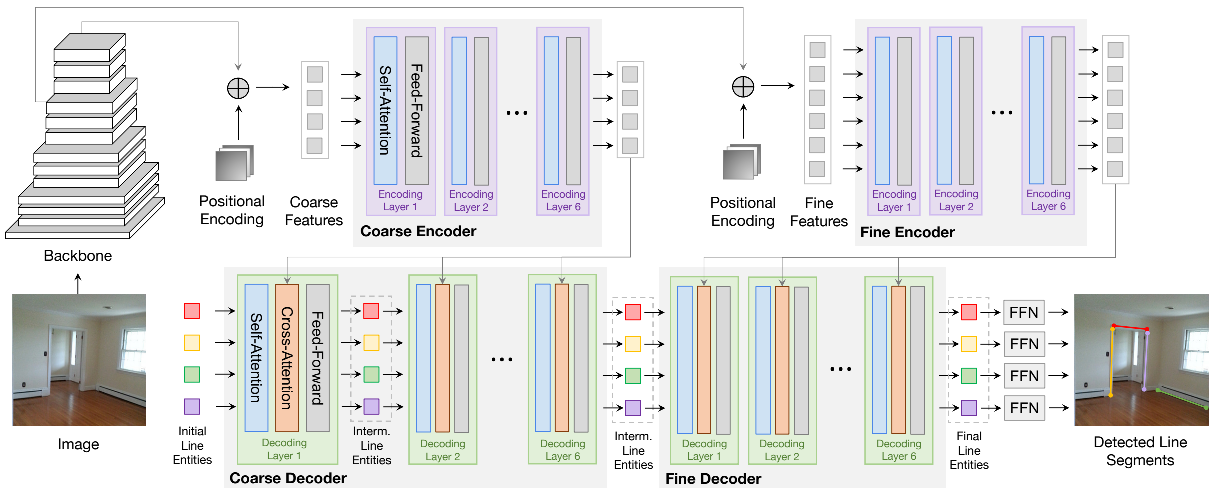


Figure 5.2: **Schematic illustration of our LETR pipeline**: An image is fed into a backbone network and generates two feature maps, which are then used by the coarse and the fine encoder respectively. Initial line entities are then first refined by the coarse decoder with the interaction of the coarse encoder output, and then the intermediate line entities from the coarse decoder are further refined by the fine decoder attending to the fine encoder. Finally, line segments are detected by feed-forward networks (FFNs) on top of line entities.

Another line of approaches employs *dense prediction* to obtain a surrogate representation map and applies a post-process procedure to extract line segments: AFM [XBW$^+$19] proposes an attraction field map as an intermediate representation that contains 2-D projection vectors pointing to associated lines. A squeeze module then recovers vectorized line segments from the attraction field map. Despite a relatively simpler design, [XBW$^+$19] demonstrates its inferior performance compared with junction-based approaches. Recently, HAWP [XWB$^+$20] builds a hybrid model of AFM [XBW$^+$19], and L-CNN [ZQM19] by computing line segment proposals from the attraction field map and then refining proposals with junctions before further line verification.

In contrast, as shown in Figure 5.1, our approach differs from previous methods by removing heuristics-driven intermediate stages for detecting edge/junction/region proposals and surrogate prediction maps. Our approach is able to directly predict vectorized line segments while keeping competitive performances under a general-purpose framework.

5.2.2 Transformer Architecture

Transformers [VSP$^+$17] have achieved great success in the natural language processing field and become *de facto* standard backbone architecture for many language models [VSP$^+$17, DCLT19]. It introduces self-attention and cross-attention modules as basic building blocks, modeling dense relations among elements of the input sequence. These attention-based mechanisms also benefit many vision tasks such as video classification [WGGH18], semantic segmentation [FLT$^+$19], image generation [ZGMO19], etc. Recently, end-to-end object detection with Transformers (DETR) [CMS$^+$20] reformulates the object detection pipeline with Transformers by eliminating the need for hand-crafted anchor boxes and non-maximum suppression steps. Instead, [CMS$^+$20] proposes to feed a set of object queries into the encoder-decoder architecture with interactions from the image feature sequence and generate a final set of predictions. A bipartite matching objective is then optimized to force unique assignments between predictions and targets.

We introduce two new aspects to DETR [CMS$^+$20] when realizing our LETR: 1) multi-scale encoder and decoder; 2) direct distance loss for the line segments.

5.3 Line Segment Detection with Transformers

5.3.1 Motivation

Despite the exceptional performance achieved by the recent deep learning based approaches [ZQM19, XBW$^+$19, XWB$^+$20] on line segment detection, their pipelines still involve heuristics-driven intermediate representations such as junctions and attraction field maps, raising an interesting question: *Can we directly model all the vectorized line segments with a neural network?* A naive solution could be simply regarding the line segments as *objects* and building a pipeline following the standard object detection approaches [RHGS15]. Since the location

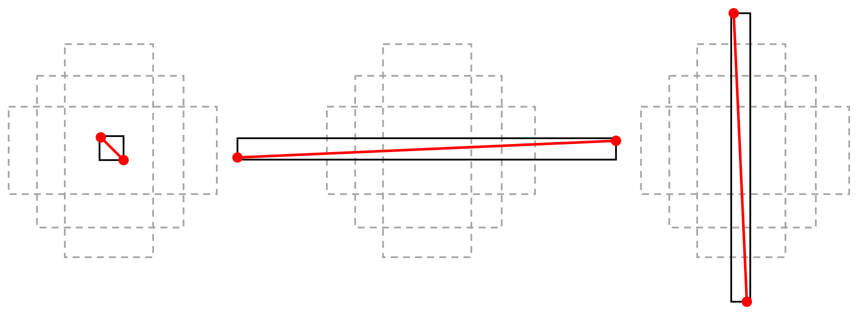


Figure 5.3: **Bounding box representation.** Three difficult cases to represent line segments using bounding box diagonals. Red lines, black boxes, and gray dotted boxes refer to as line segments, the corresponding bounding boxes, and anchors respectively.

of 2-D objects is typically parameterized as a bounding box, the vectorized line segment can be directly read from a diagonal of the bounding box associated with the line segment object. However, the limited choices of anchors make it difficult for standard two-stage object detectors to predict very short line segments or line segments nearly parallel to the axes (see Figure 5.3). The recently appeared DETR [CMS$^+$20] eliminates the anchors and the non-maximum suppression, perfectly meets the need of line segment detection. However, the vanilla DETR still focuses on bounding box representation with a GIoU loss. We further convert the box predictor in DETR into a vectorized line segment predictor by adapting the losses and enhancing the use of multi-scale features in our designed model.

5.3.2 Overview

In a line segment detection task, a detector aims to predict a set of line segments from given images. Performing line segment detection with Transformers removes the need of explicit edge/junction/region detection [ZQM19, XWB$^+$20] (see Figure 5.1). Our LETR is built purely based on the Transformer encoder-decoder structure. The proposed line segment detection process consists of four stages:

(1) *Image Feature Extraction*: Given an image input, we obtain the image feature map

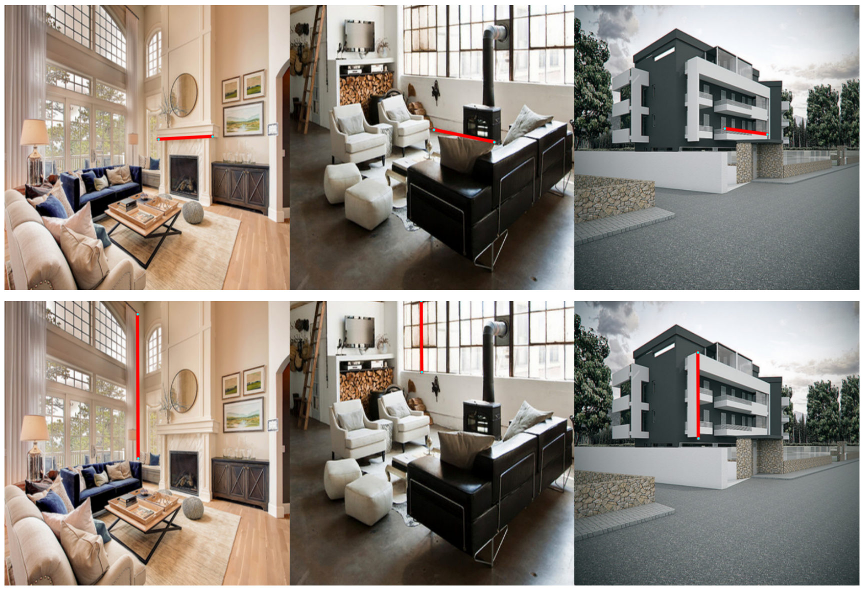


Figure 5.4: **Line entity representation.** For each row, we show how a same line entity predicts line segments with same property in three different indoor/outdoor scenes. The top line entity is specialized for horizontal line segments in the middle of the figure, and the bottom one prefers to predict vertical line segments with a various range of lengths.

$\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ from a CNN backbone with reduced dimension. The image feature is concatenated with positional embeddings to obtain spatial relations. (2) *Image Feature Encoding:* The flattened feature map $\mathbf{x} \in \mathbb{R}^{HW \times C}$ is then encoded to $\mathbf{x}' \in \mathbb{R}^{HW \times C}$ by a multi-head self-attention module and a feed forward network module following the standard Transformer encoding architecture. (3) *Line Segment Detection:* In the Transformer decoder networks, N learnable line entities $\mathbf{l} \in \mathbb{R}^{N \times C}$ interact with the encoder output via the cross-attention module. (4) *Line Segment Prediction:* Line entities make line segment predictions with two prediction heads built on top of the Transformer decoder. The line coordinates are predicted by a multi-layer perceptron (MLP), and the prediction confidences are scored by a linear layer.

Self-Attention and Cross-Attention. We first visit the scaled dot-product attention popularized by Transformer architectures [VSP+17]. The basic scaled dot-product attention consists of a set of m queries $Q \in \mathbb{R}^{m \times d}$ and a set of n key-value pairs notated as a key matrix $K \in \mathbb{R}^{n \times d}$ and a value matrix $V \in \mathbb{R}^{n \times d}$. Here we set Q, K, V to have same feature dimension d.

The attention operation F is defined as:

$$F = \text{Att}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V \tag{5.1}$$

In our encoder-decoder Transformer architecture, we adopt two attention modules based on the multi-head attention, namely the self-attention module (SA) and cross-attention (CA) module (see Figure 5.2). The SA module takes in a set of input embeddings notated as $\mathbf{x} = [x_1, ..., x_i] \in \mathbb{R}^{i \times d}$, and outputs a weighted summation $\mathbf{x}' = [x'_1, ..., x'_i] \in \mathbb{R}^{i \times d}$ of input embeddings within $\mathbf{x}$ following Eq.5.1 where $F = \text{Att}(Q = \mathbf{x}, K = \mathbf{x}, V = \mathbf{x})$. The CA module takes in two sets of input embeddings notated as $\mathbf{x} = [x_1, ..., x_i] \in \mathbb{R}^{i \times d}$, $\mathbf{z} = [x_1, ..., x_j] \in \mathbb{R}^{j \times d}$ following Eq.5.1 where $F = \text{Att}(Q = \mathbf{z}, K = \mathbf{x}, V = \mathbf{x})$.

Transformer Encoder in LETR is stacked with multiple encoder layers. Each encoder layer takes in image features $\mathbf{x} \in \mathbb{R}^{HW \times c}$ from its predecessor encoder layer and processes it with a SA module to learn the pairwise relation. The output features from SA module are passed into a point-wise fully-connected layer (FC) with activation and dropout layer followed by another point-wise fully-connected (FC) layer. Layer norm is applied between SA module and first FC layer and after second FC layer. Residual connection is added before the first FC layer and after the second FC layer to facilitate optimization of deep layers.

Transformer Decoder in LETR is stacked with multiple decoder layers. Each decoder layer takes in a set of image features $\mathbf{x}' \in \mathbb{R}^{HW \times C}$ from the last encoder layer and a set of line entities $\mathbf{l} \in \mathbb{R}^{N \times C}$ from its predecessor decoder layer. The line entities are first processed with a SA module, each line entity $l \in \mathbb{R}^{C}$ in $\mathbf{l}$ attends to different regions of image feature embeddings $\mathbf{x}'$ via the CA module. FC layers and other modules are added into the pipeline similar to the Encoder setting above.

Line Entity Interpretation. The *line entities* are analogous with the *object queries* in DETR [CMS+20]. We found each line entity has its own preferred existing region, length, and orientation of potential line segment after the training process (shown in Figure 5.4). We discuss line entities together make better predictions through self-attention and cross-attention refinement when encountering heterogeneous line segment structures in Section 5.4.4 and Figure 5.5.

5.3.3 Coarse-to-Fine Strategy

Different from object detection, line segment detection requires the detector to consider the local fine-grained details of line segments with the global indoor/outdoor structures together. In our LETR architecture, we propose a coarse-to-fine strategy to predict line segments in a refinement process. The process allows line entities to make precise predictions with the interaction of multi-scale encoded features while having an awareness of the holistic architecture with the communication to other line entities. During the coarse decoding stage, our line entities attend to potential line segment regions, often unevenly distributed, with a low resolution. During the fine decoding stage, our line entities produce detailed line segment predictions with a high resolution (see Figure 5.2). After each decoding layer at both coarse and fine decoding stage, we require line entities to make predictions through two shared prediction heads to make more precise predictions gradually.

Coarse Decoding. During the coarse decoding stage, we pass image features and line entities into an encoder-decoder Transformer architecture. The encoder receives coarse features from the output of Conv5 (C5) from ResNet with $\frac{1}{32}$ original resolution. Then, line entity embeddings attend to coarse features from the output of the encoder in the cross-attention module at each layer. The coarse decoding stage is necessary for success at fine decoding stage and its high efficiency with less memory and computation cost.

Fine Decoding. The fine decoder inherits line entities from the coarse decoder and high-resolution features from the fine encoder. The features to the fine encoder come from the output of Conv4 (C4) from ResNet with $\frac{1}{16}$ original resolution. The line entity embeddings decode feature information in the same manner as the coarse decoding stage.

5.3.4 Line Segment Prediction

In the previous decoding procedure, our multi-scale decoders progressively refine N initial line entities to produce same amount final line entities. In the prediction stage. Each final entity l will be fed into a feed-forward network (FFN), which consists of a classifier module to predict the confidence p of being a line segment, and a regression module to predict the coordinates of two end points $\hat{\mathbf{p}}_1 = (\hat{x}_1, \hat{y}_1)$, $\hat{\mathbf{p}}_2 = (\hat{x}_2, \hat{y}_2)$ that parameterizes the associated line segment $\hat{\mathbf{L}} = (\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2)$.

Bipartite Matching. Generally, there are many more line entities provided than actual line segments in the image. Thus, during the *training* stage, we conduct a set-based bipartite matching between line segment predictions and ground-truth targets to determine whether the prediction is associated with an existing line segment or not: Assume there are N line segment predictions $\{(p^{(i)}, \hat{\mathbf{L}}^{(i)}); i = 1, ..., N\}$ and M targets $\{\mathbf{L}^{(j)}; j = 1, ..., M\}$, we optimize a bipartite matching objective on a permutation function $\sigma(\cdot) : \mathbb{Z}_+ \to \mathbb{Z}_+$ which maps prediction indices $\{1, ..., N\}$ to potential target indices $\{1, ..., N\}$ (including $\{1, ..., M\}$ for ground-truth targets and $\{M+1, ..., N\}$ for unmatched predictions):

$$\mathcal{L}_{\text{match}} = \sum_{i=1}^{N} \mathbb{1}_{\{\sigma(i) \le M\}} \left[\lambda_1 d(\hat{\mathbf{L}}^{(i)}, \mathbf{L}^{(\sigma(i))}) - \lambda_2 p^{(i)} \right] \tag{5.2}$$

$$\sigma^* = \arg\min_{\sigma} \mathcal{L}_{\text{match}} \tag{5.3}$$

where $d(\cdot, \cdot)$ represents L_1 distance between coordinates and $\mathbb{1}_{\{\cdot\}}$ is an indicator function. $\mathcal{L}_{\text{match}}$ takes both distance and confidence into account with balancing coefficients λ_1, λ_2. The optimal permutation σ^* is computed using a Hungarian algorithm, mapping M positive prediction indices to target indices $\{1, ..., M\}$. During the *inference* stage, we filter the N line segment predictions by setting a fixed threshold on the confidence $p^{(i)}$ if needed due to no ground-truth provided.

5.3.5 Line Segment Losses

We compute line segment losses based on the optimal permutation σ^* from the bipartite matching procedure, in which $\{i;\sigma^*(i) \leq M\}$ represents indices of positive predictions.

Classification Loss. Based on a binary cross-entropy loss, we observe that hard examples are less optimized after learning rate decay and decide to apply adaptive coefficients inspired by focal loss [LGG+17] to the classification loss term $\mathcal{L}_{\text{cls}}$:

$$\mathcal{L}_{\text{cls}}^{(i)} = -\mathbb{1}_{\{\sigma^*(i)\leq M\}}\alpha_1(1-p^{(i)})^{\gamma}\log p^{(i)} \tag{5.4}$$

$$-\mathbb{1}_{\{\sigma^*(i)>M\}}\alpha_2 {p^{(i)}}^{\gamma}\log(1-p^{(i)}) \tag{5.5}$$

Distance Loss. We compute a simple L_1-based distance loss for line segment endpoint regression:

$$\mathcal{L}_{\text{dist}}^{(i)} = \mathbb{1}_{\{\sigma^*(i)\leq M\}} d(\hat{\mathbf{L}}^{(i)}, \mathbf{L}^{(\sigma^*(i))}) \tag{5.6}$$

where $d(\cdot,\cdot)$ represents the sum of L_1 distances between prediction and target coordinates. The distance loss is only applied to the positive predictions. Note that we remove the GIoU loss from [CMS+20] since GIoU is mainly designed for the similarity between bounding boxes instead of line segments. Thus, the final loss $\mathcal{L}$ of our model is formulated as:

$$\mathcal{L} = \sum_{i=1}^{N} \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}^{(i)} + \lambda_{\text{dist}}\mathcal{L}_{\text{dist}}^{(i)} \tag{5.7}$$

5.4 Experiments

5.4.1 Datasets

We train and evaluate our model on the ShanghaiTech *Wireframe* dataset [HWZ+18], which consists of 5000 training images and 462 testing images. We also evaluate our model on

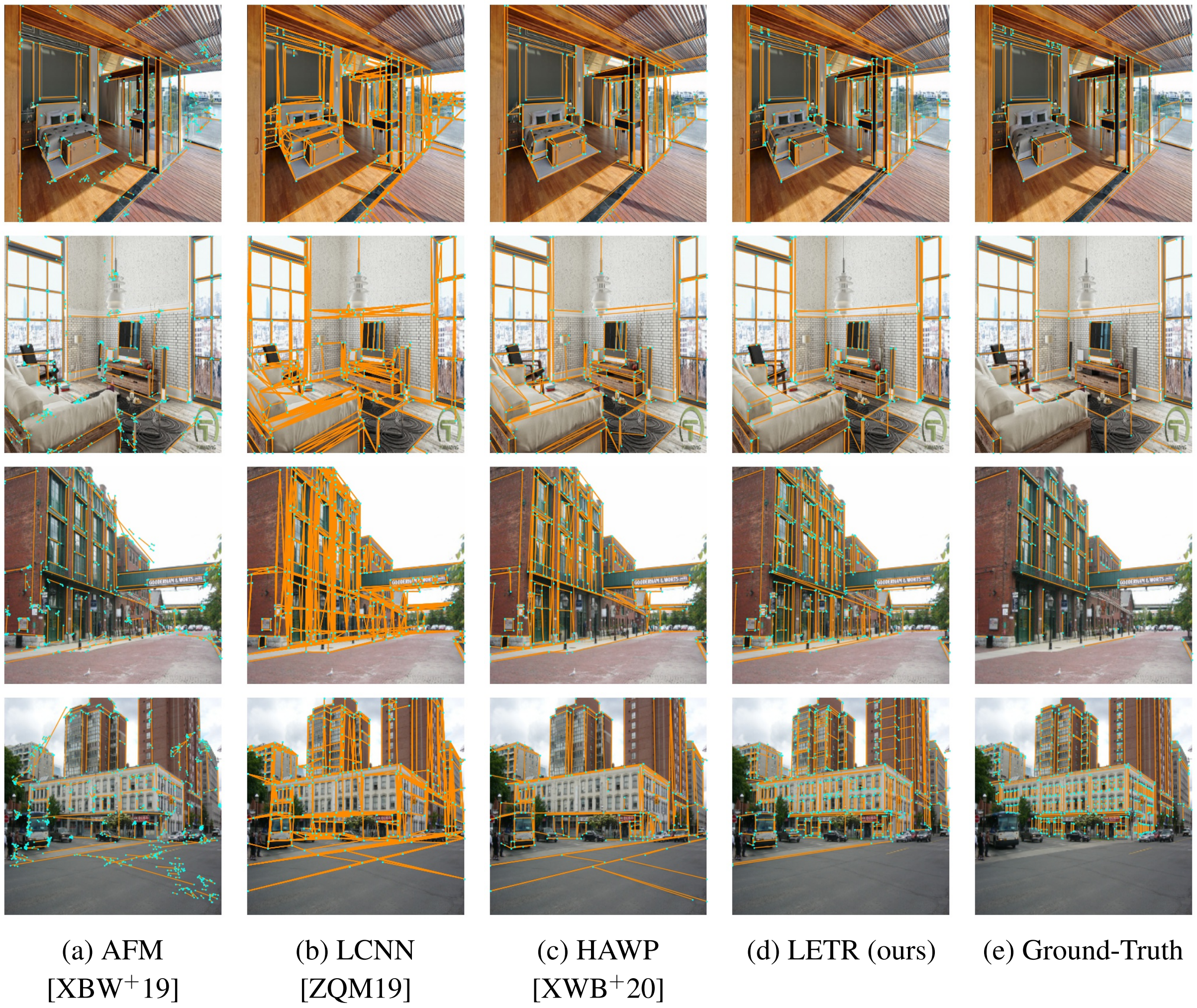


Figure 5.5: **Qualitative evaluation of line detection methods.** From left to right: the columns are the results from AFM [XBW$^+$19], LCNN [ZQM19], HAWP [XWB$^+$20], LETR (ours) and the ground-truth. From top to bottom: the top two rows are the results from the Wireframe test set, and the bottom two rows are the results from the YorkUrban test set.

the *YorkUrban* dataset [DEE08] with 102 testing images from both indoor scenes and outdoor scenes.

Through all experiments, we conduct data augmentations for the training set, including random horizontal/vertical flip, random resize, random crop, and image color jittering. At the training stage, we resize the image to ensure the shortest size is at least 480 and at most 800 pixels while the longest size is at most 1333. At the evaluation stage, we resize the image with the shortest side at least 1100 pixels.

5.4.2 Implementation

Networks. We adopt both ResNet-50 and ResNet-101 as our feature backbone. For an input image $X \in \mathbb{R}^{H_0 \times W_0 \times 3}$, the coarse encoder takes in the feature map from the Conv5 (C5) layer of ResNet backbone with resolution $x \in \mathbb{R}^{H \times W \times C}$ where $H = \frac{H_0}{32}, W = \frac{W_0}{32}, C = 2048$. The fine encoder takes in a higher resolution feature map ($H = \frac{H_0}{16}, W = \frac{W_0}{16}, C = 1024$) from the Conv4 (C4) layer of ResNet. Feature maps are reduced to 256 channels by a 1x1 convolution and are fed into the Transformer along with the *sine/cosine* positional encoding. Our coarse-to-fine strategy consists of two independent encoder-decoder structures processing multi-scale image features. Each encoder-decoder structure is constructed with 6 encoder and 6 decoder layers with 256 channels and 8 attention heads.

Optimization. We train our model using 4 Titan RTX GPUs through all our experiments. Model weights from DETR [CMS$^+$20] with ResNet-50 and ResNet-101 backbone are loaded as pre-training, and we discuss the effectiveness of pre-training in Section 5.5. We first train the coarse encoder-decoder for 500 epochs until optimal. Then, we freeze the weights in the coarse Transformer and train the fine Transformer initialized by coarse Transformer weights for 325 epochs (including a 25-epoch focal-loss fine-tuning). We adopt deep supervision [LXG$^+$15, XT15] for all decoder layers following DETR [CMS$^+$20]. FFN prediction head weights are shared through all decoder layers. We use AdamW as the model optimizer and set weight decay as 10^{-4}. The initial learning rate is set to 10^{-4} and reduced by a factor of 10 every 200 epochs for the coarse decoding stage and every 120 epochs for the fine prediction stage. We use 1000 line entities in all reported benchmarks unless specified elsewhere. To mitigate the class imbalance, we also reduce the classification weight for background/no-object instances by a factor of 10.

Table 5.1: **Comparison to prior work on Wireframe and YorkUrban benchmarks.** Our proposed LETR reaches state-of-the-art performance except sAP^{10} and sAP^{15} slightly worse than HAWP [XWB+20] in Wireframe. FPS Results for LETRs are tested on a single Tesla V100. Results for other prior works are adopted from HAWP paper.

Method	*Wireframe Dataset*						*YorkUrban Dataset*						FPS
	sAP^{10}	sAP^{15}	sF^{10}	sF^{15}	AP^H	F^H	sAP^{10}	sAP^{15}	sF^{10}	sF^{15}	AP^H	F^H	
LSD [VGJMR08]	/	/	/	/	55.2	62.5	/	/	/	/	50.9	60.1	**49.6**
DWP [HWZ+18]	5.1	5.9	/	/	67.8	72.2	2.1	2.6	/	/	51.0	61.6	2.24
AFM [XBW+19]	24.4	27.5	/	/	69.2	77.2	9.4	11.1	/	/	48.2	63.3	13.5
L-CNN [ZQM19]	62.9	64.9	61.3	62.4	82.8	81.3	26.4	27.5	36.9	37.8	59.6	65.3	15.6
HAWP [XWB+20]	66.5	68.2	64.9	65.9	86.1	83.1	28.5	29.7	39.7	40.5	61.2	66.3	29.5
LETR (ours)	65.2	67.7	**65.8**	**67.1**	**86.3**	**83.3**	**29.4**	**31.7**	**40.1**	**41.8**	**62.7**	**66.9**	5.04

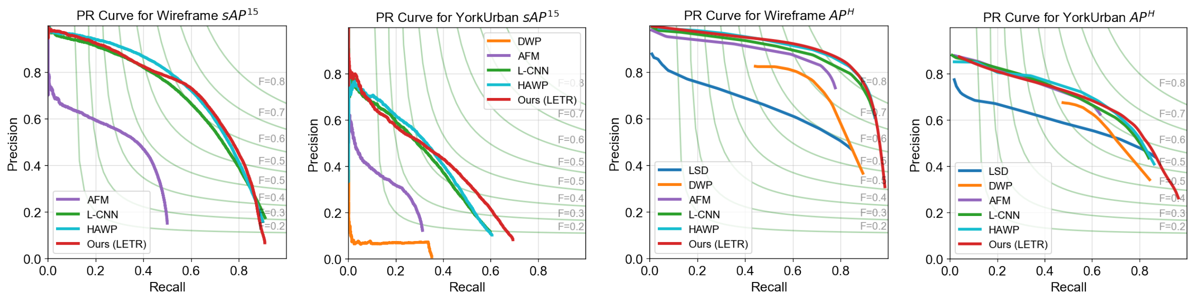


Figure 5.6: **Precision-recall (PR) curves.** PR curves of sAP^{15} and AP^H for DWP[HWZ+18], AFM[XBW+19], L-CNN[ZQM19], HAWP[XWB+20] and LETR (ours) on Wireframe and YorkUrban benchmarks.

5.4.3 Evaluation Metric

We evaluate our results based on two heatmap-based metrics, AP^H and F^H, which are widely used in previous LSD task[ZQM19, HWZ+18], and Structural Average Precision (sAP) which is proposed in L-CNN [ZQM19]. On top of that, we evaluate the result with a new metric, Structural F-score (sF), for a more comprehensive comparison.

Heatmap-based metrics, AP^H, F^H: Prediction and ground truth lines are first converted to heatmaps by rasterizing the lines, and we generate the precision-recall curve comparing each pixel along with their confidence. Then we can use the curve to calculate F^H and AP^H.

Structural-based metrics, sAP[ZQM19], sF: Given a set of ground truth line and a set of predicted lines, for each ground-truth line $\mathbf{L}$, we define a predicted line $\hat{\mathbf{L}}$ to be a match of $\mathbf{L}$ if their L_2 distance is smaller than the pre-defined threshold $\vartheta \in \{10, 15\}$. Over the set of lines matched to $\mathbf{L}$, we select the line with the highest confidence as a true positive and treat the rest as candidates for false positives. If the set of matching lines is empty, we would regard this ground-truth line as false negative. Each predicted line would be matched to at most one ground truth line, and if a line isn't matched to any ground-truth line, then it is considered as a false positive. The matching is recomputed at each confidence level to produce the precision-recall curve, and we consider sAP as the area under this curve. Considering F^H as the complementary F-score measurement for AP^H, we evaluate the F-score measurement for sAP, denoted as sF, to be the best balanced performance measurement.

5.4.4 Results and Comparisons

We summarize quantitative comparison results between LETR and previous line segment detection methods in Table 5.1. We report results for LETR with ResNet-101 backbone for Wireframe dataset and results with ResNet-50 backbone for York dataset. Our LETR achieves new state-of-the-art for all evaluation metrics on YorkUrban Dataset [DEE08]. In terms of heatmap-based evaluation metrics, our LETR is consistently better than other models for both benchmarks and outperforms HAWP [XWB+20] by 1.5 for AP^H on YorkUrban Dataset. We show PR curve comparison in Figure 5.6 on sAP^{15} and AP^H for both Wireframe [HWZ+18] and YorkUrban benchmarks. In Figure 5.6, we notice the current limitation of LETR comes from lower precision prediction when we include fewer predictions compare to HAWP. When we include all sets of predictions, LETR predicts slightly better than HAWP and other leading methods, which matches our hypothesis that holistic prediction fashion can guide line entities to refine low confident predictions (usually due to local ambiguity and occlusion) with high confident predictions.

We also show both Wireframe and YorkUrban line segment detection qualitative results from LETR and other competing methods in Figure 5.5. The top two rows are indoor scene detection results from the Wireframe dataset, while the bottom two rows are outdoor scene detection results from the YorkUrban dataset.

5.5 Ablation Study

Compare with Object Detection Baselines. We compare LETR results with two object detection baseline where the line segments are treated as 2-D objects within this context in Table 5.2. We see clear limitations when using bounding box diagonal for both Faster R-CNN and DETR responding to our motivation in Section 5.3.1.

Table 5.2: **Comparison with object detection baselines** on Wireframe [HWZ+18].

Method	sAP^{10}	sAP^{15}	sF^{10}	sF^{15}
Faster R-CNN	38.4	40.7	51.5	53.0
Vanilla DETR	53.8	57.2	57.2	59.0
LETR (ours)	65.2	67.7	65.8	67.1

Effectiveness of Multi-Stage Training. We compare the effectiveness of different modules in LETR in Table 5.3. During the coarse decoding stage, LETR reaches 62.3 and 65.2 for sAP^{10} and sAP^{15} with encoding features from the C5 layer of ResNet backbone, and 63.8 and 66.5 with the one from C4 of ResNet backbone. The fine decoder reaches 64.7 and 67.4 for sAP^{10} and sAP^{15} by improving the coarse prediction with fine-grained details from high-resolution features. We then adjust the data imbalance problem with focal loss to reach 65.2 and 67.7 for sAP^{10} and sAP^{15}.

As shown in Figure 5.7 (a), we found it is necessary to train the fine decoding stage after the coarse decoding stage converges. Training both stages together as a one-stage model results a significant worse performance after 400 epochs.

Effect of Number of Queries. We found a large number of line entities is essential to the line segment detection task by experimenting on a wide range of the number of line entities (See Figure 5.7 (c)), and using 1000 line entities is optimal for the Wireframe benchmark which contains 74 line segments in average.

Table 5.3: **Effectiveness of modules.** Ablation study of the architecture design and learning aspects in the proposed LETR on Wireframe dataset. (C) indicates the indexed feature used for coarse decoder; (F) indicates the indexed feature used for fine decoder.

Coarse Decoding	Fine Decoding	Focal Loss	Feature Index	sAP^{10}	sAP^{15}
✓			C5(C)	62.3	65.2
✓			C4(C)	63.8	66.5
✓	✓		C5(C), C4(F)	64.7	67.4
✓	✓	✓	C5(C), C4(F)	65.2	67.7

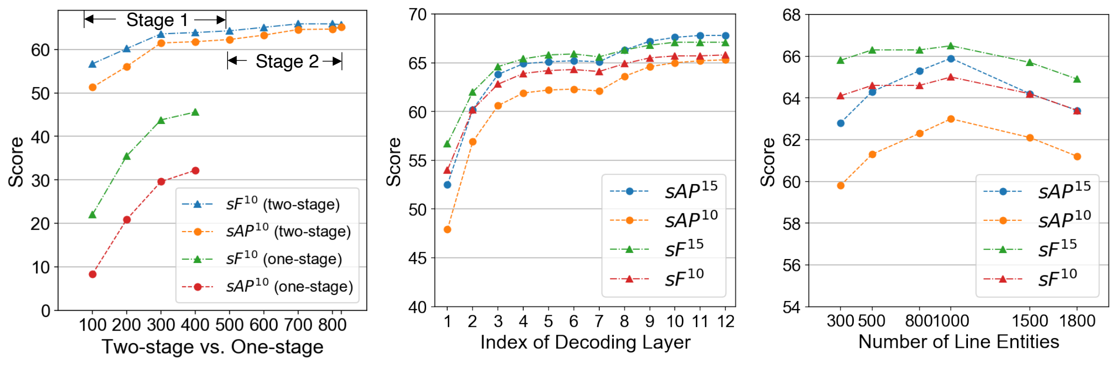


Figure 5.7: (a) **Multi-stage vs. single-stage training.** We compare results training coarse and fine layers in single stages and multi-stages (b) **Number of decoding layers.** We evaluate the performance of outputs from each decoding layer. The 1-6 layers are coarse decoder layers and 7-12 layers fine decoder layers. (c) **Number of line entities.** We test LETR (coarse decoding stage only) with different numbers of line entities on Wireframe.

Effect of Image Upsampling. All algorithms see the same input image resolution (640×480 typically). However, some algorithms try more precise predictions by upsampling images. To understand the impact of upsampling, we train and test HAWP and LETR under multiple upsampling scales. In Table 5.4 below, higher training upsampling resolution improves both methods. LETR obtains additional gains with higher test upsampling resolution.

Table 5.4: **Effectiveness of upsampling** with Wireframe dataset. LETR uses ResNet-101 backbone. * Our LETR-512 resizes original image with the shortest size in a range between 288 and 512 † Our LETR-800 resizes original image with the shortest size in a range between 480 and 800.

	Train Size	Test Size	sAP^{10}	sAP^{15}	sF^{10}	sF^{15}
HAWP	512	512	65.7	67.4	64.7	65.8
HAWP	832	832	**67.7**	**69.1**	65.5	66.4
HAWP	832	1088	65.7	67.1	64.3	65.1
LETR	512*	512	61.1	64.1	63.1	64.8
LETR	800†	800	64.3	67.0	65.5	66.9
LETR	800†	1100	65.2	67.7	**65.8**	**67.1**

Effectiveness of Pretraining. We found model pretraining is essential for LETR to obtain state-of-the-art results. With DETR pretrained weights for COCO object detection [LMB$^+$14], our coarse-stage-only model converges at 500 epochs. With CNN backbone pretrained weights for ImageNet classification, our coarse-stage-only model converges to a lower score at 900 epochs. Without pretraining, LETR is difficult to train due to the limited amount of data in the Wireframe benchmark.

Table 5.5: **Effectiveness of pretraining.** We train LETR (coarse decoding stage only) with two variants. ImageNet represents LETR with ImageNet pretrained ResNet backbone. COCO represents LETR with COCO pretrained DETR weights.

Method	Epochs	sAP^{10}	sAP^{15}	sF^{10}	sF^{15}
ImageNet	900	58.4	62.0	62.4	64.6
COCO	500	62.3	65.2	64.3	65.9

5.6 Visualization

We demonstrate LETR's coarse-to-fine decoding process in Figure 5.8. The first two columns are results from the coarse decoder receiving decoded features from the C5 ResNet layer. While the global structure of the scene is well-captured efficiently, the low-resolution features prevent it from making predictions precisely. The last two columns are results from the fine decoder receiving decoded features from the C4 ResNet layer and line entities from the coarse decoder. The overlay of attention heatmaps depicts more detailed relations in the image space, which is the key to the detector performance. This finding is also shown in Figure 5.7(b), where the decoded output after each layer has consistent improvement with the multi-scale encoder-decoder strategy.

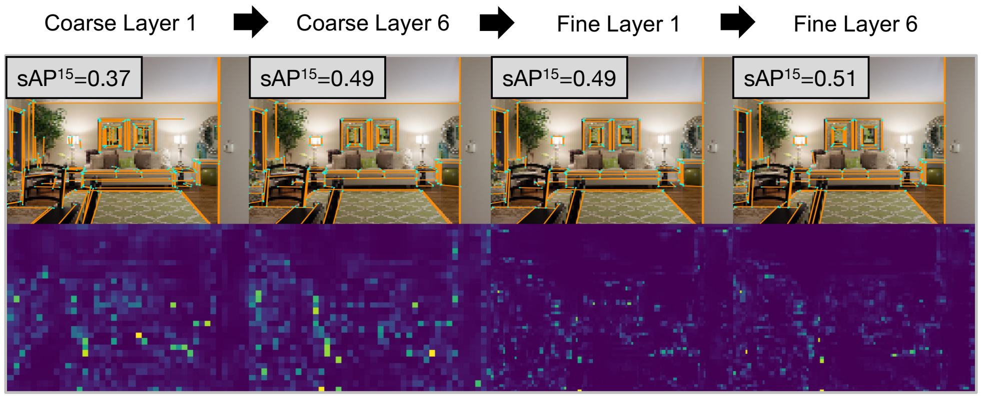


Figure 5.8: **Visualization of LETR coarse-to-fine decoding process.** From top to bottom: The 1st row shows line segment detection results based on line entities after different layers and the 2nd row shows its corresponding overlay of attention heatmaps. From left to right: The 1st, 2nd, 3rd, 4th columns are coarse decoder layer 1, coarse decoder layer 6, fine decoder layer 1, fine decoder layer 6, respectively.

5.7 Conclusion

In this work, we presented LETR, a line segment detector based on a multi-scale encoder/decoder Transformer structure. By casting the line segment detection problem in a holistically end-to-end fashion, we perform set prediction without explicit edge/junction/region detection and heuristics-guided perceptual grouping processes. A direct endpoint distance loss allows geometric structures beyond bounding box representations to be modeled and predicted.

Acknowledgements

This chapter is based on the material “Line Segment Detection Using Transformers without Edges” by Yifan Xu*, Weijian Xu*, David Cheung, and Zhuowen Tu, which appears in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2021. The dissertation author is the co-primary investigator and author of this material.

Chapter 6

Conclusion

This dissertation explores a wide range of visual structures in convolutional neural networks and Transformers in computer vision.

In convolutional neural networks, we discuss the geometric structure in object skeleton detection and the part structure in few-shot learning. In object skeleton detection, we introduce a geometry-aware objective function that provides global and local geometric constraints, often ignored in prior approaches. In few-shot image recognition, we devise an end-to-end framework with constellation modules that implements an explicit part representation. We observe that the explicit part representation and implicit feature representation from CNN filters can share mutual benefits in few-shot recognition.

In Transformers, we concentrate on multi-scale structures in a generic vision Transformer model and a specific line segment detection method. First, we present a co-scale mechanism in vision Transformer that allows the representations of different scales to communicate effectively. This co-scale mechanism collaborating with conv-attention obtains superb performance on image classification and many downstream tasks. Second, we introduce a multi-scale Transformer that progressively refines line segments from coarse to fine. This Transformer can detect accurate line segments without heavy heuristic designs in prior works.

Though there have been extensive studies that utilize certain visual structures to improve representation in deep learning, it is still difficult to identify proper visual structures and embed them in a *principled* way. We list several open questions as follows:

Large-Scale Visual Pretraining. Our visual structures are usually designed for a fixed and constrained data scenario. However, with the visual datasets of rapidly increased size, we may need to reduce several structures in the current neural network design, or add some different structures. As an early attempt, ViT [DBK$^+$21] shows that a large vanilla Transformer is enough to encode rich visual representation without inductive biases from commonly considered visual structures. However, it remains unclear which visual structures should still be essential when the model is trained on very large-scale datasets.

Multi-Task Learning. Currently, visual structures for deep representation typically target a single task. However, an ultimate goal in computer vision and machine learning is to build a unified model for as many tasks as possible. In this context, we have two future directions in the design of visual structures: (1) we devise visual structures that meet the common need of all tasks; (2) we collect visual structures for every task and combine them. These future directions can also benefit self-supervised representation learning, which aims for feature transferability to numerous downstream tasks.

Adversarial Robustness. Neural networks are known to be vulnerable to adversarial attacks [SZS$^+$13]. Much progress has been made to improve the adversarial robustness of deep neural networks, but there is still no complete solution. Thus, it would be intriguing to explore if the issues of adversarial examples can be mitigated by certain visual structures. Such visual structures may not necessarily bring a theoretical guarantee on adversarial robustness, but the resulting models with these visual structures are nevertheless valuable for many areas such as face recognition.

Appendix A

Constellation Nets for Few-Shot Learning

A.1 Appendix

A.1.1 Few-Shot Learning Framework

In this section, we introduce background concepts of meta-learning and elaborate the few-shot learning framework used in our ConstellationNet.

Meta-Learning in Few-Shot Classification. Current few-shot learning is typically formulated as a *meta-learning* task [FAL17], in which an dataset $\mathcal{D}^{\text{base}}$ is used to provide commonsense knowledge and a dataset $\mathcal{D}^{\text{novel}}$ for the few-shot classification. $\mathcal{D}^{\text{base}}$ has the classes $\mathcal{C}_{\text{base}}$ which are disjoint from the $\mathcal{C}_{\text{novel}}$ in $\mathcal{D}^{\text{novel}}$ to ensure fairness. There are two stages, *meta-training* and *meta-test*, in the meta-learning framework: In *meta-training* stage, we attempt to train a model to learn generic features from $\mathcal{D}^{\text{base}}$. In *meta-test* stage, we adapt the model on the limited training split from $\mathcal{D}^{\text{novel}}$ and evaluate the performance of the model on the test split.

ProtoNet-Based Framework. In our ConstellationNet, we adopt ProtoNet [SSZ17] as the base few-shot learning framework. In ProtoNet, the dataset $\mathcal{D}^{\text{novel}}$ is represented by a series of K-way N-shot tasks $\{\mathcal{T}\}$ where each task consists of a *support* set and a *query* set, i.e. $\mathcal{T} = (\mathcal{T}^{\text{supp}}, \mathcal{T}^{\text{query}})$. The support set $\mathcal{T}^{\text{supp}}$ contains K classes and each class has N examples from the training split of $\mathcal{D}^{\text{novel}}$, which are used to adapt the model in *meta-test* stage. The query set $\mathcal{T}^{\text{query}}$ from the test split of $\mathcal{D}^{\text{novel}}$ is then used to evaluate the model.

The ProtoNet attempts to learn a generic feature extractor $\phi(\mathbf{x})$ on image $\mathbf{x}$, and represent a class k by the prototype $\mathbf{c}_k$, which is the average feature of examples from support set $\mathcal{T}^{\text{supp}}$ with this class:

$$\mathbf{c}_k = \frac{1}{|N|} \sum_{(\mathbf{x},y)\in\mathcal{T}^{\text{supp}}, y=k} \phi(\mathbf{x}) \tag{A.1}$$

During the *meta-test* stage, we use the prototypes to compute the probability p_k of a query example $\mathbf{x}' \in \mathcal{T}^{\text{query}}$ on class k and predict its label y':

$$p_k = p(y=k|\mathbf{x}', \mathcal{T}^{\text{supp}}) = \frac{\exp(d(\phi(\mathbf{x}'), \mathbf{c}_k))}{\sum_{k'} \exp(d(\phi(\mathbf{x}'), \mathbf{c}_{k'}))}, \quad y' = \arg\max_k p_k. \tag{A.2}$$

where $d(\cdot,\cdot)$ is a cosine similarity function (different from the Euclidean distance in [SSZ17]).

During the *meta-training* stage, there are two different training schemes: The *prototypical scheme* from ProtoNet uses an *episodic learning* strategy that also formulates the dataset $\mathcal{D}^{\text{base}}$ as a series of tasks $\{\mathcal{T}\}$. The negative log-likelihood loss $\mathcal{L}(\phi)$ is optimized:

$$\ell(\mathcal{T}^{\text{supp}}, \mathcal{T}^{\text{query}}) = \mathbb{E}_{(\mathbf{x}',y')\in\mathcal{T}^{\text{query}}} - \log p(y=y'|\mathbf{x}', \mathcal{T}^{\text{supp}}), \tag{A.3}$$

$$\mathcal{L}(\phi) = \mathbb{E}_{\mathcal{T}=(\mathcal{T}^{\text{supp}},\mathcal{T}^{\text{query}})\sim\mathcal{D}^{\text{base}}} \ell(\mathcal{T}^{\text{supp}}, \mathcal{T}^{\text{query}}). \tag{A.4}$$

Another way is the *standard classification scheme* [CLX⁺21]: It simply uses $\mathcal{D}^{\text{base}}$ as a standard classification dataset $\{(\mathbf{x},y)\}$ consisting of Q classes in total. Thus, a cross-entropy loss $\mathcal{L}(\phi)$ is optimized:

$$\mathcal{L}(\phi) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}^{\text{base}}} - \log \frac{\exp(\mathbf{w}_y \cdot \phi(\mathbf{x}))}{\sum_q \exp(\mathbf{w}_q \cdot \phi(\mathbf{x}))} \tag{A.5}$$

where $\mathbf{w}_q$ is the linear weight for class q. In our ConstellationNet, we use the standard classification scheme at default. For the experiment with multi-branch network, we use the prototypical scheme and standard classification scheme for separate branches.

A.1.2 Datasets

The CIFAR-FS dataset [BHTV19] is a few-shot classification benchmark containing 100 classes from CIFAR-100 [KH09]. The classes are randomly split into 64, 16 and 20 classes as meta-training, meta-validation and meta-testing set respectively. For each class, it contains 600 images of size 32×32. We adopt the split from [LMRS19]. The FC100 dataset [ORLL18] is another benchmark based on CIFAR-100 where classes are grouped into 20 superclasses to void the overlap between the splits. The *mini*-ImageNet dataset [VBL$^+$16] is a common benchmark for few-shot classification containing 100 classes from ILSVRC-2012 [DDS$^+$09]. The classes are randomly split into 64, 16 and 20 classes as meta-training, meta-validation and meta-testing set respectively. For each class, it contains 600 images of size 84×84. We follow the commonly-used split in [RL17], [LMRS19] and [CLX$^+$21]. In all experiments, we conduct data augmentation for the meta-training set of all datasets to match [LMRS19]'s implementation.

A.1.3 Network Backbone

Conv-4. Following [LMRS19], we adopt the same network with 4 convolutional blocks. Each of the 4 blocks has a 3×3 convolutional layer, a batch normalization layer, a ReLU activation and a 2×2 max-pooling layer sequentially. The numbers of filters are 64 for all 4 convolutional layers.

ResNet-12. Following [CLX$^+$21], we construct the residual block with 3 consecutive convolutional blocks followed by an addition average pooling layer where each convolutional block has a 3×3 convolutional layer, a batch normalization layer, a leaky ReLU activation, and max-pooling layers. The ResNet-12 network has 4 residual blocks with each filter size set to 64, 128, 256, 512, respectively.

WRN-28-10. WideResNet expands the residual blocks by increasing the convolutional channels and layers [ZK16]. WRN-28-10 uses 28 convolutional layers with a widening factor of 10.

A.1.4 Constellation Module Configuration

To achieve the best performance with constellation modules, we do not always fully enable them after all the convolutional layers. For Conv-4, we use constellation modules after all four convolutional layers, but the cell relation modeling module is disabled in first two constellation modules due to the high memory consumption. For ResNet-12, we enable the constellation modules after the convolutional layer 1,7,8,9 and disable the relation modeling module in the first constellation module. We use the deep supervision in ResNet-12 to stablize the training of constellation modules.

A.1.5 Self-Attention Settings

We follow the common practice in [VSP^{+}17] to set the attention layer with residual connections, dropout and layer normalization. The sine positional encoding follows settings in [CMS^{+}20].

A.1.6 Training Details

Optimization Settings. We follow implementation in [LMRS19], and use SGD optimizer with initial learning rate of 1, and set momentum to 0.9 and weight decay rate to 5×10^{-4}. The learning rate reduces to 0.06, 0.012, and 0.0024 at epoch 20, 40 and 50. The inverse temperature β is set to 100.0 in the cluster assignment step, and λ is set to 1.0 in the centroid movement step.

A.1.7 Ablation Study on the Number of Clusters

Table A.1: **Ablation study on the number of clusters for random and similar classes.** We investigate how similarities of images in the training dataset affect the optimal number of clusters. The first group of experiments use training dataset with 30 similar classes while the second group use 30 random classes from FC100 dataset, all of which performed on ResNet-12 with Constellation module.

# Clusters	Similar Classes		Random Classes	
	1-shot	5-shot	1-shot	5-shot
8	38.9 ± 0.2	$\mathbf{52.8 \pm 0.2}$	40.9 ± 0.2	54.5 ± 0.2
16	$\mathbf{39.1 \pm 0.2}$	51.8 ± 0.2	40.9 ± 0.2	$\mathbf{54.9 \pm 0.2}$
32	38.7 ± 0.2	52.3 ± 0.2	40.9 ± 0.2	54.7 ± 0.2
64	38.8 ± 0.2	52.3 ± 0.2	$\mathbf{41.2 \pm 0.2}$	$\mathbf{54.9 \pm 0.2}$
128	38.8 ± 0.2	52.1 ± 0.2	40.8 ± 0.2	54.7 ± 0.2

Table 4 studies the number of clusters needed for random and similar classes. The result shows the optimal number of clusters are less affected by the number of clusters but more affected by the similarity between classes. Less number of clusters are needed for dataset with classes of high similarity, which aligns with our intuition, limited number of patterns exist in this dataset so that small number of clusters are enough to represent its part-based information.

FC100 training dataset consists of 60 classes that are grouped evenly into 12 superclasses. In the random classes group, the training dataset includes 6 randomly selected super-classes (i.e., 30 classes) and models are trained with 8, 16, 32, 64 and 128 number of clusters. The highest accuracy occurs at 16 clusters (1-shot: 39.12% in ResNet-12). In the similar classes group, 30 classes are randomly sampled from the original training dataset and we repeat the same experiments as above. The highest accuracy occurs at 64 clusters (1-shot: 41.22% in ResNet-12), which is much more than the 16 clusters used for images from similar classes.

A.1.8 Additional Experiments with Negative Margin

Table A.2 studies the use of negative margin loss [LCL$^+$20] on our Conv-4 models. In the negative margin loss, we use the inner-product similarity, the temperature coefficient $\beta = 1.0$ and

Table A.2: **Additional experiments with the use of negative margin.** Average classification accuracies (%) on *mini*-ImageNet meta-test split. We compare our ConstellationNet and baseline with and without the negative margin loss based on Conv-4.

Baseline	**Cell Feature Clustering**	**Cell Relation Modeling**	**Negative Margin**	**Conv-4**	
				1-shot	**5-shot**
✓				50.62 ± 0.23	68.40 ± 0.19
✓			✓	51.42 ± 0.23	68.84 ± 0.19
✓	✓	✓		57.03 ± 0.23	74.09 ± 0.18
✓	✓	✓	✓	57.55 ± 0.23	74.49 ± 0.18

the negative margin $m = -0.5$, which attains the best performance improvement on our models. Besides, we do not have the fine-tune step during meta-test. Our baseline with the negative margin loss obtains 0.80% improvement on 1-shot and 0.44% improvement on 5-shot compared with the baseline. Similarly, our ConstellationNet with the negative margin loss achieves 0.52% improvement on 1-shot and 0.40% improvement on 5-shot. The consistent improvement of negative margin loss on the baseline and our ConstellationNet indicates that our constellation module is orthogonal to the negative margin loss, and both modules can boost the performance on few-shot classification.

A.1.9 Clarification on Clustering Procedure

In this section, we add more clarification on our cell feature clustering procedure in Sec. 3.4.1: During the training stage, the global cluster centers $\mathcal{V} = \{\mathbf{v}_k\}$ are updated by the computed clustering centers $\{\mathbf{v}'_k\}$ in current mini-batch. Each update to a cluster center $\mathbf{v}_k$ is weighted by a momentum coefficient η determined by the value of an associated counter s_k, since we would like to avoid large adjustment from the current mini-batch in order to stabilize the global cluster centers. Besides, the mini-batches of examples are randomly drawn from the dataset following [Scu10], without specialized design to optimize clustering learning. During the evaluation stage, we fix the global cluster centers $\mathcal{V}$ in the forward step of our model, avoiding the potential information leak or transduction from the test mini-batches.

A.1.10 Multi-Branch Details

Our embedding $\phi(\mathbf{x})$ is separated into two branches after a shared stem (Y-shape), which is defined as $\phi(\mathbf{x}) = \{\phi^{\text{cls}}(\mathbf{x}), \phi^{\text{proto}}(\mathbf{x})\}$ and $\phi^{\text{cls}}(\mathbf{x}) = g^{\text{cls}}(f^{\text{stem}}(\mathbf{x}))$, $\phi^{\text{proto}}(\mathbf{x}) = g^{\text{proto}}(f^{\text{stem}}(\mathbf{x}))$. Two branches $\phi^{\text{cls}}(\mathbf{x}), \phi^{\text{proto}}(\mathbf{x})$ are trained by standard classification and prototypical schemes separately in a multi-task learning fashion. During the testing time, $\phi^{\text{cls}}(\mathbf{x})$ and $\phi^{\text{proto}}(\mathbf{x})$ are concatenated together to compute distance between support prototypes and query images.

For our ConstellationNet, we split the network into two branches after the second convolutional blocks (Conv-4) or the second residual blocks (ResNet-12). We keep the shared stem identical to the network backbone and reduce the channels of two separate branches to match the parameter size of the model without multi-branch.

A.1.11 Connection with Capsule Networks

A notable development to learning the explicit structured representation in an end-to-end framework is the capsule networks (CapsNets) [SFH17]. The line of works on CapsNets [SFH17, HSF18, KSTH19, TSGS20] intends to parse a visual scene in an interpretable and hierarchical way. [SFH17] represents parts and objects in vector-based capsules with a dynamic routing mechanism. [TSGS20] uses a stacked autoencoder architecture to model the hierarchical relation among parts, objects and scenes. Here our ConstellationNet maintains part modeling by enabling the joint learning of the convolution and constellation modules to simultaneously attain implicit and explicit representations.

Bibliography

[Bel21] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. In *ICLR*, 2021.

[BHR86] J. Brian Burns, Allen R. Hanson, and Edward M. Riseman. Extracting straight lines. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(4):425–455, 1986.

[BHTV19] Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.

[Blu73] Harry Blum. Biological shape and visual science (part i). *Journal of theoretical Biology*, 38(2):205–287, 1973.

[BU02] Eran Borenstein and Shimon Ullman. Class-specific, top-down segmentation. In *European conference on computer vision*, pages 109–122. Springer, 2002.

[BWR89] Michael Boldt, Richard Weiss, and Edward Riseman. Token-based extraction of straight lines. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1581–1594, 1989.

[Can86] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[CLD$^+$21] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

[CLX$^+$21] Yinbo Chen, Zhuang Liu, Huijuan Xu, Trevor Darrell, and Xiaolong Wang. Meta-baseline: exploring simple meta-learning for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9062–9071, 2021.

[CMS$^+$20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[CN12] Adam Coates and Andrew Y Ng. Learning feature representations with k-means. In *Neural networks: Tricks of the trade*, pages 561–580. Springer, 2012.

[CPK$^+$17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[CTH$^+$18] Jingchun Cheng, Yi-Hsuan Tsai, Wei-Chih Hung, Shengjin Wang, and Ming-Hsuan Yang. Fast and accurate online video object segmentation via tracking parts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7415–7424, 2018.

[CV19] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *TPAMI*, 2019.

[CWP$^+$19] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[CWRL19] Yuanqiang Cai, Weiqiang Wang, Haiqing Ren, and Ke Lu. Spn: short path network for scene text detection. *Neural Computing and Applications*, Feb 2019.

[CZT$^+$21] Xiangxiang Chu, Bo Zhang, Zhi Tian, Xiaolin Wei, and Huaxia Xia. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021.

[DBK$^+$21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[DDS$^+$09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[DEE08] Patrick Denis, James H Elder, and Francisco J Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *European conference on computer vision*, 2008.

[DH72] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[DJ94] M-P Dubuisson and Anil K Jain. A modified hausdorff distance for object matching. In *Proceedings of 12th international conference on pattern recognition*, volume 1, pages 566–568. IEEE, 1994.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[DTB06] Piotr Dollár, Zhuowen Tu, and Serge Belongie. Supervised learning of edges and object boundaries. In *CVPR*, 2006.

[DXX18] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *Proc. IEEE Int. Conf. Acoust. Sph. Sig Process.*, 2018.

[DZ13] Piotr Dollár and C Lawrence Zitnick. Structured forests for fast edge detection. In *CVPR*, 2013.

[EG02] James H Elder and Richard M Goldberg. Ecological statistics of gestalt laws for the perceptual organization of contours. *Journal of Vision*, 2(4):5, 2002.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[FFFP06] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[FGMR09] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009.

[FH05] Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International journal of computer vision*, 61(1):55–79, 2005.

[FLT$^+$19] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

[FPZ03] Robert Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003.

[FS03] Yasutaka Furukawa and Yoshihisa Shinagawa. Accurate and robust line segment extraction by analyzing distribution around peaks in hough space. *Computer Vision and Image Understanding*, 92(1):1–25, 2003.

[FSG17] 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*, pages 605–613, 2017.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[Gir15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[GLY19] Weifeng Ge, Xiangru Lin, and Yizhou Yu. Weakly supervised complementary parts models for fine-grained image classification from the bottom up. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3034–3043, 2019.

[GVZ95] Nicolas Guil, Julio Villalba, and Emilio L Zapata. A fast hough transform for segment detection. *IEEE Transactions on Image Processing*, 4(11):1541–1548, 1995.

[GWJ$^+$14] Yue Gao, Meng Wang, Rongrong Ji, Xindong Wu, and Qionghai Dai. 3-d object retrieval with hausdorff distance learning. *IEEE Transactions on industrial electronics*, 61(4):2088–2098, 2014.

[HBNH$^+$20] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *CVPR*, 2020.

[HCB$^+$19] Ruibing Hou, Hong Chang, MA Bingpeng, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. In *Advances in Neural Information Processing Systems*, pages 4005–4016, 2019.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[HJL$^+$19] Wei-Chih Hung, Varun Jampani, Sifei Liu, Pavlo Molchanov, Ming-Hsuan Yang, and Jan Kautz. Scops: Self-supervised co-part segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 869–878, 2019.

[HMX$^+$20] Shell Xu Hu, Pablo Garcia Moreno, Yang Xiao, Xi Shen, Guillaume Obozinski, Neil Lawrence, and Andreas Damianou. Empirical bayes transductive meta-learning with synthetic gradients. In *International Conference on Learning Representations*, 2020.

[HSF18] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *ICLR*, 2018.

[HWZ$^+$18] Kun Huang, Yifan Wang, Zihan Zhou, Tianjiao Ding, Shenghua Gao, and Yi Ma. Learning to parse wireframes in images of man-made environments. In *CVPR*, pages 626–635, 2018.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[HZXL19] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, 2019.

[JCLY17] Koteswar Rao Jerripothula, Jianfei Cai, Jiangbo Lu, and Junsong Yuan. Object co-skeletonization with co-segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3881–3889. IEEE, 2017.

[JH01] Jeong-Hun Jang and Ki-Sang Hong. A pseudo-distance map for the segmentation-free skeletonization of gray-scale images. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 18–23. IEEE, 2001.

[JKF01] Oliver Jesorsky, Klaus J Kirchberg, and Robert W Frischholz. Robust face detection using the hausdorff distance. In *International conference on audio-and video-based biometric person authentication*, pages 90–95. Springer, 2001.

[KCJ$^+$17] Wei Ke, Jie Chen, Jianbin Jiao, Guoying Zhao, and Qixiang Ye. Srn: side-output residual network for object symmetry detection in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1068–1076, 2017.

[KDDD15] Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015.

[KFF15] 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*, pages 3128–3137, 2015.

[KH09] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[KSJ+20] Jogendra Nath Kundu, Siddharth Seth, Varun Jampani, Mugalodi Rakesh, R Venkatesh Babu, and Anirban Chakraborty. Self-supervised 3d human pose estimation via part guided novel image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6152–6162, 2020.

[KSTH19] Adam Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E Hinton. Stacked capsule autoencoders. In *Advances in Neural Information Processing Systems*, pages 15486–15496, 2019.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LCH+17] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3000–3009, 2017.

[LCL+20] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification. In *European Conference on Computer Vision*, pages 438–455. Springer, 2020.

[LDG+17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[LGG+17] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2999–3007, 2017.

[LGY98] Tyng-Luh Liu, Davi Geiger, and Alan L Yuille. Segmenting by seeking the symmetry axis. In *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No. 98EX170)*, volume 2, pages 994–998. IEEE, 1998.

[LH19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[LHL+20] Aoxue Li, Weiran Huang, Xu Lan, Jiashi Feng, Zhenguo Li, and Liwei Wang. Boosting few-shot learning with adaptive margin loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12576–12584, 2020.

[LHS20] Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. Self-supervised label augmentation via input transformations. *37 th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020*, 2020.

[Lin98a] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. *International journal of computer vision*, 30(2):117–156, 1998.

[Lin98b] Tony Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116, 1998.

[LKQY18] Chang Liu, Wei Ke, Fei Qin, and Qixiang Ye. Linear span network for object skeleton detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 133–148, 2018.

[LLC$^+$21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[LLP$^+$19] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sungju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. In *International Conference on Learning Representations*, 2019.

[LMB$^+$14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[LMRS19] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LSD13] Alex Levinshtein, Cristian Sminchisescu, and Sven Dickinson. Multiscale symmetric part detection and grouping. *International journal of computer vision*, 104(2):117–134, 2013.

[LSD15] 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*, pages 3431–3440, 2015.

[LSP06] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[LWZ$^+$21] Ke Li, Shijie Wang, Xiang Zhang, Yifan Xu, Weijian Xu, and Zhuowen Tu. Pose recognition with cascade transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1944–1953, 2021.

[LXG$^+$15] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Proc. Int. Conf. Artificial Intell. & Stat.*, 2015.

[LYLL15] Xiaohu Lu, Jian Yao, Kai Li, and Li Li. Cannylines: A parameter-free line segment detector. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 507–511. IEEE, 2015.

[Mar82] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., USA, 1982.

[MFM04] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on pattern analysis and machine intelligence*, 26(5):530–549, 2004.

[MFTM01] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423. IEEE, 2001.

[MGK00] Jiri Matas, Charles Galambos, and Josef Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer vision and image understanding*, 78(1):119–137, 2000.

[MRCA18] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018.

[MYMT18] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. In *International Conference on Machine Learning*, pages 3664–3673. PMLR, 2018.

[NCSG11] Marcos Nieto, Carlos Cuevas, Luis Salgado, and Narciso García. Line segment detection using weighted mean shift procedures on a 2d slice sampling strategy. *Pattern Analysis and Applications*, 14(2):149–163, 2011.

[ORLL18] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *Advances in neural information processing systems*, 31, 2018.

[PHZ17] Yuxin Peng, Xiangteng He, and Junjie Zhao. Object-part attention model for fine-grained image classification. *IEEE Transactions on Image Processing*, 27(3):1487–1500, 2017.

[QLL19] Lei Qi, Xiaoqiang Lu, and Xuelong Li. Exploiting spatial relation for fine-grained image classification. *Pattern Recognition*, 91:47–55, 2019.

[RDS$^+$15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C.

Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[RGCD19] Javier Ribera, David Guera, Yuhao Chen, and Edward J Delp. Locating objects without bounding boxes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6479–6489, 2019.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[RL17] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[RNSS18] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[RPV$^+$19] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. In *Proc. Advances in Neural Inf. Process. Syst.*, 2019.

[SB97] Stephen M Smith and J Michael Brady. Susan—a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997.

[SBHZ16] Wei Shen, Xiang Bai, Zihao Hu, and Zhijiang Zhang. Multiple instance subspace learning via partial random projection tree for local reflection symmetry in natural images. *Pattern Recognition*, 52:306–316, 2016.

[Scu10] D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178, 2010.

[SFC$^+$11] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *CVPR 2011*, pages 1297–1304. Ieee, 2011.

[SFH17] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.

[SLJ$^+$15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[SP08] Kaleem Siddiqi and Stephen Pizer. *Medial representations: mathematics, algorithms and applications*, volume 37. Springer Science & Business Media, 2008.

[SR15] Marcel Simon and Erik Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1143–1151, 2015.

[SSDZ99] Kaleem Siddiqi, Ali Shokoufandeh, Sven J Dickinson, and Steven W Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35(1):13–32, 1999.

[SSZ17] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[STFW05] Erik B Sudderth, Antonio Torralba, William T Freeman, and Alan S Willsky. Learning hierarchical models of scenes, objects, and parts. In *ICCV*, volume 2, 2005.

[STT12] Ruslan Salakhutdinov, Joshua B Tenenbaum, and Antonio Torralba. Learning with hierarchical-deep models. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1958–1971, 2012.

[SUV18] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *NAACL-HLT*, 2018.

[SVI$^+$16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[SYZ$^+$18] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[SZJ$^+$16] Wei Shen, Kai Zhao, Yuan Jiang, Yan Wang, Zhijiang Zhang, and Xiang Bai. Object skeleton extraction in natural images by fusing scale-associated deep side outputs. In *CVPR*, 2016.

[SZJ$^+$17] Wei Shen, Kai Zhao, Yuan Jiang, Yan Wang, Xiang Bai, and Alan Yuille. Deepskeleton: Learning multi-task scale-associated deep side outputs for object skeleton extraction in natural images. *IEEE Transactions on Image Processing*, 26(11):5298–5311, 2017.

[SZS$^+$13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[SZZ$^+$21] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *WACV*, pages 3531–3539, 2021.

[TCD$^+$21] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[TK12] Stavros Tsogkas and Iasonas Kokkinos. Learning-based symmetry detection in natural images. In *European Conference on Computer Vision*, pages 41–54. Springer, 2012.

[TL19] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[TSGS20] Yao-Hung Hubert Tsai, Nitish Srivastava, Hanlin Goh, and Ruslan Salakhutdinov. Capsules with inverted dot-product attention routing. In *International Conference on Learning Representations*, 2020.

[Tu08] Zhuowen Tu. Auto-context and its application to high-level vision tasks. In *CVPR*, 2008.

[TWH19] Pavel Tokmakov, Yu-Xiong Wang, and Martial Hebert. Learning compositional representations for few-shot recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6372–6381, 2019.

[VBL$^+$16] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[VGJMR08] Rafael Grompone Von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, 32(4):722–732, 2008.

[Vis22] Visual system. Visual system — Wikipedia, the free encyclopedia, 2022. [Online; accessed 13-February-2022].

[VSP$^+$17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[WGGH18] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[Wit84] Andrew Witkin. Scale-space filtering: A new approach to multi-scale description. In *Proc. IEEE Int. Conf. Acoust. Sph. Sig Process.*, volume 9, pages 150–153, 1984.

[WL18] Tie-Qiang Wang and Cheng-Lin Liu. Fully convolutional network based skeletonization for handwritten chinese characters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[WRKS16] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[WSC$^+$20] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020.

[WWP00] Markus Weber, Max Welling, and Pietro Perona. Unsupervised learning of models for recognition. In *ECCV*, 2000.

[WXL$^+$21] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021.

[WXT$^+$19] Yukang Wang, Yongchao Xu, Stavros Tsogkas, Xiang Bai, Sven Dickinson, and Kaleem Siddiqi. Deepflux for skeletons in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[WXW$^+$21] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8741–8750, 2021.

[WZX17] Qingfu Wan, Wei Zhang, and Xiangyang Xue. Deepskeleton: Skeleton map for 3d human pose regression. *arXiv preprint arXiv:1711.10796*, 2017.

[XBK$^+$15] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[XBW$^+$19] Nan Xue, Song Bai, Fudong Wang, Gui-Song Xia, Tianfu Wu, and Liangpei Zhang. Learning attraction field representation for robust line segment detection. In *CVPR*, 2019.

[XGD+17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[XLC+20] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout for document image understanding. In *SIGKDD*, 2020.

[XLCT18] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *CVPR*, 2018.

[XROOP19] Chen Xing, Negar Rostamzadeh, Boris Oreshkin, and Pedro O O Pinheiro. Adaptive cross-modal few-shot learning. *Advances in Neural Information Processing Systems*, 32:4847–4857, 2019.

[XT15] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.

[XWB+20] Nan Xue, Tianfu Wu, Song Bai, Fudong Wang, Gui-Song Xia, Liangpei Zhang, and Philip HS Torr. Holistically-attracted wireframe parsing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2788–2797, 2020.

[XXCT21] Yifan Xu, Weijian Xu, David Cheung, and Zhuowen Tu. Line segment detection using transformers without edges. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4257–4266, 2021.

[YB04] Zeyun Yu and Chandrajit Bajaj. A segmentation-free approach for skeletonization of gray-scale images via anisotropic vector diffusion. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE, 2004.

[YCW+21] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, 2021.

[YHC92] Alan L Yuille, Peter W Hallinan, and David S Cohen. Feature extraction from faces using deformable templates. *International journal of computer vision*, 8(2):99–111, 1992.

[YHO+19] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

[YRW+21] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12498–12507, 2021.

[YSM19] Sung Whan Yoon, Jun Seo, and Jaekyun Moon. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *International Conference on Machine Learning*, pages 7115–7123. PMLR, 2019.

[ZCDLP18] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[ZF14] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[ZGMO19] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363. PMLR, 2019.

[ZJK20] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, 2020.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016.

[ZLB$^+$19] Ziheng Zhang, Zhengxin Li, Ning Bi, Jia Zheng, Jinlei Wang, Kun Huang, Weixin Luo, Yanyu Xu, and Shenghua Gao. Ppgnet: Learning point-pair graph for line segment detection. In *CVPR*, 2019.

[ZM07] Song-Chun Zhu and David Mumford. *A stochastic grammar of images*. Now Publishers Inc, 2007.

[ZQHM19] Yichao Zhou, Haozhi Qi, Jingwei Huang, and Yi Ma. Neurvps: Neural vanishing point scanning via conic convolution. *Advances in Neural Information Processing Systems*, 32, 2019.

[ZQM19] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 962–971, 2019.

[ZS84] TY Zhang and Ching Y Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.

[ZSG$^+$18] Kai Zhao, Wei Shen, Shanghua Gao, Dandan Li, and Ming-Ming Cheng. Hi-fi: hierarchical feature integration for skeleton detection. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 1191–1197. AAAI Press, 2018.

[ZSL$^+$21] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021.

[ZY96] Song Chun Zhu and Alan L Yuille. Forms: a flexible object recognition and modelling system. *International journal of computer vision*, 20(3):187–212, 1996.

[ZZK$^+$20] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proc. AAAI Conf. Artificial Intell.*, pages 13001–13008, 2020.

[ZZLS18] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.

[ZZN$^+$19] Jian Zhang, Chenglong Zhao, Bingbing Ni, Minghao Xu, and Xiaokang Yang. Variational few-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1685–1694, 2019.

[ZZW$^+$17] Yousong Zhu, Chaoyang Zhao, Jinqiao Wang, Xu Zhao, Yi Wu, and Hanqing Lu. Couplenet: Coupling global structure with local parts for object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 4126–4134, 2017.