UNIVERSITY OF CALIFORNIA SAN DIEGO

Exploring Visual Perception with Transformers and World Model Representation

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

in

Cognitive Science

by

Yifan Xu

2023

University of California San Diego

2023

DEDICATION

To my family and their unwavering support.

EPIGRAPH

All truths are easy to understand once they are discovered;

the point is to discover them.

—Galileo Galilei

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am profoundly grateful for the privilege of pursuing my Ph.D. under the expert guidance of Professor Zhuowen Tu. His unwavering support, encouragement, and mentorship have been pivotal in shaping my academic journey. Professor Tu's comprehensive understanding of AI and cognitive science has consistently inspired me to explore cutting-edge research areas and examine the intricate aspects of intelligent systems and human cognitive processes. His direction has not only sharpened my research skills but also instilled a profound appreciation for these interdisciplinary fields, propelling my passion for studying their complexities. I am particularly indebted to Professor Tu for his steadfast encouragement to venture into unexplored territories and embrace challenges throughout my Ph.D. experience. Under his mentorship, my research interests have broadened to encompass computer vision, natural language understanding and reinforcement learning, all of which are integral components of human intelligence. This diverse exposure has been vital to my development as a young researcher and has empowered me to make positive contributions to the field.

I would like to express my sincere appreciation to my dissertation committee members and the entire Cognitive Science Department at UC San Diego for cultivating an engaging and research-oriented atmosphere. The liberty to explore a myriad of AI domains and the open, cooperative setting have been crucial to both my academic and personal development. I am especially thankful to my dissertation committee members, Professors Virginia de Sa, Benjamin Bergen, Andrea Chiba, and Xiaolong Wang, for their invaluable insights and guidance throughout the process. Moreover, I am grateful to Professor Marcelo Matta for his thoughtful discussions as a member of my advancement committee. In addition, I would like to convey my gratitude to Professors Marta Kutas and Douglas A. Nitz from the UC San Diego Cognitive Science Department. Their commitment and enthusiasm for supporting young researchers, including myself, have consistently been a tremendous source of inspiration.

I want to express my gratitude to my colleagues at UC San Diego, Weijian Xu, Nicklas

Hansen, Xiang Zhang, Huaijing Wang, Tyler Chang, and David Cheung, whose passion for research and collaborative efforts have played an important role in my research achievements. I am also thankful for the valuable input provided by Kownjoon Lee, Zheng Ding, Justin Lazarow, Sainan Liu, Wenlong Huang, Ke Li, and Shijie Wang, whose collaboration have also been invaluable to my academic and personal growth.

Finally, I would like to extend my deepest appreciation to my parents, whose steadfast support, love, and encouragement have been the driving force behind my academic journey. Last, I would like to extend my heartfelt thanks to my wife, whose unrelenting faith in my abilities has been a constant source of motivation and inspiration.

Chapter 2 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 primary investigator and author of this material.

Chapter 3 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 primary investigator and author of this material.

Chapter 4 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 the International Conference on Learning Representations (ICLR) 2021. The dissertation author is the primary investigator and author of this material.

Chapter 5 is based on the material “On the Feasibility of Cross-Task Transfer with Model-Based Reinforcement Learning" by Yifan Xu*, Nicklas Hansen*, Zirui Wang, Yung-Chieh Chan, Hao Su, and Zhuowen Tu, which appears in the International Conference on Learning Representations (ICLR) 2023. The dissertation author is the primary investigator and author of this material.

VITA

2018	B. S. in Computer Science, University of California San Diego
2023	Ph. D. in Cognitive Science, University of California San Diego

PUBLICATIONS

Yifan Xu*, Nicklas Hansen*, Zirui Wang, Yung-Chieh Chan, Hao Su, and Zhuowen Tu. “On the Feasibility of Cross-Task Transfer with Model-Based Reinforcement Learning" In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*, 2023. (* equal contribution)

Weijian Xu*, **Yifan Xu***, Tyler Chang and Zhuowen Tu. “Co-Scale Conv-Attentional Image Transformers" , In *Proceddings 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" In *Proceedings of the Ninth International Conference on Learning Representations (ICLR)*, 2021. (*equal contribution)

ABSTRACT OF THE DISSERTATION

Exploring Visual Perception with Transformers and World Model Representation

by

Yifan Xu

Doctor of Philosophy in Cognitive Science

University of California San Diego, 2023

Professor Zhuowen Tu, Chair

This research explores the development of generalized representations in artificial intelligence by leveraging visual attention and world models. The primate visual system processes vast amounts of sensory data through bidirectional visual pathways, utilizing top-down influence, such as visual attention, from high-level cognitive processes to affect early-stage visual processing. Drawing inspiration from this, attention-based visual systems, with the transformer model as the most prominent example, have significantly advanced computer vision by incorporating top-down information, which enables them to exhibit adaptability and versatility when processing a wide range of complex visual tasks. Concurrently, the concept of world models connects visual perception to higher-level cognitive processes and has led to a renaissance in model-based reinforcement learning. Deepening our understanding of visual attention and world models is an essential step towards achieving general artificial intelligence capable of performing a wide range of visual tasks and following complex instructions.

The thesis begins by focusing on designing transformer-based attention mechanisms in visual representation learning for diverse computer vision tasks. First, The research explores the application of these attention mechanisms to develop a geometry perception framework for line segment detection. Next, It presents a novel transformer-based visual system capable of handling multi-scale and contextual information. Furthermore, the thesis highlights the use of attention mechanisms in modeling spatial relationships among object parts in few-shot classification tasks. Lastly, it explores the potential of model-based reinforcement learning algorithms for efficient task transfer, introducing a framework that leverages learned world models to accelerate learning in new and distinct tasks. Through these works, we contribute to ongoing efforts to develop AI systems that closely resemble the flexibility and versatility of the human brain.

Chapter 1

Introduction

The primate visual system is responsible for processing vast amounts of sensory data, with about 55% of the neocortex dedicated to vision [FVE91]. Early views of visual processing suggested a hierarchical feedforward process in the cortex, involving sequential progression through cortical areas [FVE91, Mar82]. However, this perspective evolved with evidence pointing to the significance of "top-down" processing [DD95, UN96, DL02]. Current consensus indicates that visual pathways in the primate brain operate bidirectionally where high-level cognitive processes can influence early-stage visual processes through feedback connections [GL13]. This bidirectional processing is evident in various forms of visual attention, including spatial and feature attention [AEC13, RP95].

Early computer vision approaches for visual processing, such as David Marr's influential *stages of vision* framework, focused on a hierarchical computational framework [Mar82]. Despite its early promise, Marr's approach faced challenges due to the error-prone nature of its stages under complex scenarios [Lee03]. The deep learning era led to a great success in computer vision with the development of end-to-end system with convolutional neural networks (CNNs) that were inspired by the connectivity patterns in the visual cortex [LBBH98]. CNNs facilitated great advancements in computer vision and related fields [HZRS16, HGDG17, GPAM+14, SHM+16a].

However, these approaches did not account for top-down feedback mechanisms, such as attention [AEC13, RP95], and faced difficulties in performing diverse visual tasks requiring adaptability and versatility to follow complex instructions.

Recently, attention-based visual systems have emerged as an alternative to CNN-based systems with flexibility to integration top-down information, revolutionizing computer vision [DBK$^+$21, CMS$^+$20, ADL$^+$22, RZP$^+$22a]. Attention mechanisms, which were originally developed for natural language processing, have been successfully applied to computer vision tasks, enabling more accurate and efficient processing of complex visual scenes. Transformer models with self-attention [ADL$^+$22, RZP$^+$22a], showcase the flexibility and robustness of attention mechanisms in handling visual data and complex language instructions. This new paradigm in computer vision combines both "bottom-up" and "top-down" information processing while removing the need for explicit intermediate representations and heuristic-guided designs. This shift moves us closer to general artificial intelligence by forming a generalized deep representation that can study diverse cognitive functions in a unified framework, such as visual processing [DBK$^+$21], language understanding [DCLT19a], speech [DXX18], memory [BKPS20], and decision-making [CLR$^+$21].

Moreover, the concept of world models emerges as another keystone in human cognition, connecting visual perception to higher-level cognitive processes [Lan14]. World models enable humans to form internal representations of their environment, which are crucial for tasks such as planning, navigation and decision-making behavior [QRK$^+$05, NRO$^+$15, CT17]. In artificial intelligence, the incorporation of world models has led to a renaissance of model-based reinforcement learning (RL), with agents benefiting from improved learning efficiency in various tasks [HS18b, HLBN19, SAH$^+$20b, KBM$^+$20, YLK$^+$21, HWS22]. The concurrent development of visual attention and world models offers the potential for developing more generalized and efficient AI systems.

In this thesis, we aim to investigate building generalized representation that leverages visual attention and world models. In the following sections, we (1) discuss human visual attention and connect it to the current state of attention research in artificial intelligence, focusing primarily on recent transformer models featuring self-attention. (2) We outline the internal model theory for human perception and relate it to the world models developed within the context of model-based reinforcement learning. (3) Finally, we provide an overview of this thesis.

1.1 Visual Attention

1.1.1 Human Visual Attention and Its Mechanisms

From cognitive science and system neuroscience perspectives, the history of attention research is mainly centered on visual attention given the critical role visual perception plays in human cognition [IK01, TBG05, Lin20]. Here we divide our discussion on visual attention into a section on *spatial attention* and another on *feature attention*. We will also discuss the interplay between "*bottom-up*" and "*top-down*" in both sections.

Visual Spatial Attention One primary role of visual attention is to provide flexible control to select information for future processing. In primate's visual field, only the fovea, a small area of the visual field at the center, offers high resolution. Our visual system resolves such spatial resolution limitations with rapid eye movement (saccades) multiple times each second. The spatial movement of the focal region is taken to indicate the subject's shift in interest, which is known as *overt* spatial attention [AEC13]. While effective, *overt* spatial attention is not the only option for controlling visual attention. Without eye movements, humans can still attend to locations in peripheral vision given appropriate cues. This is called *covert* spatial attention [AEC13]. Some studies show that a potential role for covert spatial attention is to guide overt spatial attention [RRDU87].

Spatial attention can guide feature detection by directing attention to salient regions of the visual field. Previous studies have found that certain image patterns, such as oriented edges, color contrast, intensity, and motion, can capture attention and bias attention to a particular location [IK01, vZD05]. This is usually referred as *non-volitional* attention and is computed in "*bottom-up*" fashion. When a specific task (e.g., reading text in a cluttered background) is assigned, spatial attention would strongly affect the pattern of eye movements to provide enough visual information for that task [HB05]. This is referred to as *volitional* attention due to the extra effort the individual expends based on "top-down" instructions.

Visual Feature Attention In addition to attending to specific spatial locations, primates like humans are also able to attend to cued visual features *globally* in their visual field [RP95]. The visual features can be a specific color, shape, and orientation and have often been examined in various visual search tasks [ZD11]. A closely related topic is object attention where the appearance of a particular object will attract subject's attention [Che12]. In addition, feature attention often appear together with spatial attention to provide additional "top-down" information [HG09]. A recent study shows the selectivity from both feature and spatial attention implement a normalization model in visual processing flow [RH09].

The interaction between spatial and feature attention allows humans to effectively process complex visual scenes by selectively focusing on the most relevant information. The equilibrium between bottom-up and top-down processes facilitates flexible adaptation to various tasks, environments, and objectives. As our comprehension of visual attention expands, researchers can utilize this knowledge to develop advanced artificial intelligence systems that mimic human-like visual processing capabilities as we will discuss in the following sections.

1.1.2 The Rise of Self-Attention and Transformer Models

Visual attention plays a critical role in not only the primate visual system but also in building artificial intelligence systems during the deep learning era [GBC16]. In exploring the concept of visual attention in computer vision, it is essential to recognize its parallel evolution with natural language processing (NLP), an area in which attention mechanisms have emerged earlier and were studied more extensively [SVL14, BCB15, LPM15, WSC$^+$16, VSP$^+$17, DCLT19b, BMR$^+$20]. Attention mechanisms have played a key role in enhancing neural network-based sequence-to-sequence models, such as recurrent neural networks (RNNs [RHW86, HS97]), in NLP applications like machine translation by effectively capturing long-range dependencies and contextual information [SVL14, BCB15, LPM15, WSC$^+$16]. By empowering RNNs to selectively attend to the most relevant parts of the input sequences, attention mechanisms streamline the processing of lengthy texts.

The success of attention mechanisms in NLP has led researchers to focus on developing neural network models that emphasize attention, surpassing their initial role as a complementary component to classic RNNs. One of the most notable examples is the Transformer model [VSP$^+$17], specifically developed to tackle machine translation challenges with the utilization of a unique attention mechanism named as self-attention. Empirical studies in [VSP$^+$17] demonstrated that the self-attention mechanism alone is capable of capturing global dependencies between inputs and outputs, effectively eliminating the need for recurrent layers in the model. This innovative approach significantly streamlined the architecture design and enhanced the models' ability to comprehend text structure and semantic meaning. Consequently, the Transformer has become the *de facto* standard architecture for numerous language tasks [DCLT19b, BMR$^+$20]. The growing popularity of attention mechanisms can be attributed to their capacity to efficiently process complex relationships within input data, a crucial aspect for tasks that demand a deep understanding of context and semantics.

To illustrate how attention mechanisms operate within Transformer models [VSP$^+$17], consider a French-to-English translation task. The Transformer, built on an encoder-decoder architecture, converts input French words into initial embeddings — learned vector representations capturing meaning. Self-attention layers in both the encoder and decoder form a multi-layered structure, with each layer refining the previous layer's output embeddings for decoding the subsequent English word. During translation, the encoder's self-attention layers process initial embeddings, capturing the input French text's structure and meaning. The processed embeddings are passed to the decoder, which predicts English words using relevant information from the French sentence. The self-attention in decoder ensures coherent, contextually appropriate translations. Each self-attention layer involves these steps: (1) converting input embeddings into query, key, and value vectors with learned transformations; (2) calculating similarity between query-key pairs to create attention scores for word relationships; (3) normalizing scores to emphasize relevant relationships; (4) weighting value vectors using normalized scores; and (5) summing weighted vectors to produce output embeddings with improved contextual relationships.

The term "attention" refers to such described mechanism selectively focuses on specific elements of the input data, simulating how human attention prioritizes certain aspects of information when processing it. The term "self-attention" highlights that the attention is applied to the input data itself, enabling the model to capture internal relationships within the input data and dynamically adjust its focus throughout the translation. By leveraging self-attention in both the encoder and decoder, Transformer models can efficiently process and translate complex language sequences. They selectively concentrate on the most relevant information, integrating data from both source and target languages while capturing subtle relationships within and between languages. This approach leads to accurate and contextually rich translations.

The success of attention mechanisms in natural language processing (NLP) has inspired researchers to apply similar techniques to computer vision within deep learning models, given the shared need to process intricate relationships within data efficiently and manage context and semantics effectively [WGGH18, FLT$^+$19, ZGMO19]. Prior to the introduction of attention mechanisms in computer vision, the primary deep learning architecture for visual tasks was convolutional neural networks (CNNs [LBBH98]). CNNs use fixed-size kernels and operate on spatially local regions of the input image, which limits their ability to effectively process images with rich global context details. In contrast, attention mechanisms enable neural networks to selectively focus their computational resources on the most important components of the input data when processing complex visual scenes. By incorporating attention mechanisms into CNNs, researchers can enhance their ability to handle the inherent complexity of numerous computer vision tasks, such as video classification [WGGH18], semantic segmentation [FLT$^+$19], and image generation [ZGMO19], improving long-range interactions between pixels in time and space.

Incorporating attention mechanisms into CNNs was not the end of the story. As Transformers began to dominate in NLP by processing text using only self-attention, researchers started to apply Transformer attention mechanisms to various computer vision tasks, developing models such as ViT (Vision Transformer) [DBK$^+$21], DETR (DEtection TRansformer) [CMS$^+$20], and others [LMGH22, HCX$^+$21, ADL$^+$22, RZP$^+$22a]. These models showcase the adaptability and versatility of attention mechanisms, which have proven effective not only in language processing but also in handling visual data. By leveraging the strengths of attention mechanisms, these models can selectively focus on the most relevant visual information, enabling more accurate and efficient processing of complex visual scenes. As a result, attention mechanisms have become a powerful tool for advancing the state of the art in both NLP and computer vision. By bridging the gap between the two fields, attention mechanisms open up new possibilities for developing more sophisticated and human-like artificial intelligence systems that can effectively process and interpret complex data across multiple domains, as seen in recent studies [ADL$^+$22, RZP$^+$22a].

1.1.3 Relationship between Self-Attention and Visual Attention

Investigating the connections between visual attention and deep learning attention mechanisms is crucial for advancing artificial intelligence systems. Such investigations provide great insights into how human cognitive principles can inspire and strengthen deep learning mechanisms. In this section, we will dive deeper into the connections between various aspects of visual attention and deep learning attention mechanisms, highlighting the importance of understanding these relationships.

Relation to Spatial Attention: Spatial attention in the human visual system can be divided into overt and covert spatial attention. Both forms have parallels to components in deep learning attention mechanisms, particularly self-attention in Transformer models. Overt spatial attention involves moving the eyes to focus on a specific area within the visual field. This form of attention can be likened to the encoder-decoder architecture of Transformer models [CMS$^+$20, ADL$^+$22]. In this architecture, the encoder selectively attends to the most relevant portions of the input data, similar to how our eyes move to focus on specific parts of a visual scene [AEC13]. The processed information from the encoder is then passed to the decoder, which generates the output. This process mirrors how our brain processes visual input to form a coherent understanding of the scene.

Covert spatial attention, on the other hand, allows humans to attend to peripheral locations without moving their eyes [AEC13]. This ability is similar to how self-attention mechanisms in decoder-only Transformer models selectively focus on different parts of the input data without explicit spatial information [DBK$^+$21, RZP$^+$22a]. Self-attention enables the model to weigh the importance of different elements in the input and adjust its focus accordingly, akin to how covert spatial attention can shift our focus without eye movement.

Relation to Feature Attention: Humans can also attend to specific features, such as color, shape, or orientation within their visual field [RP95, ZD11]. Similarly, attention mechanisms in deep learning models can learn to focus on particular features within the input data to improve performance [ADL$^+$22, RZP$^+$22a]. For example, Flamingo [ADL$^+$22], a single visual language Transformer model with self-attention, might selectively attend to objects with specific colors or shapes based on a language query to answer them accurately [WGGH18, FLT$^+$19]. This selective attention enables the model to prioritize the most relevant features in the input data, leading to better performance on complex tasks. Furthermore, the combination of feature attention and spatial attention in deep learning models allows them to capture complex relationships between features in the input data, which is essential for tasks that require a deep understanding of context and semantics [RH09].

Relation to Bottom-up and Top-down Processes: The interplay between bottom-up and top-down processes in visual attention is also reflected in deep learning models. Consider a visual language task where the encoder-decoder model is given an image and a language query, such as "Find the red ball." The model's goal is to identify the relevant object within the image based on the language instruction.

Bottom-up processes in visual attention, driven by salient features in the visual environment [IK01, vZD05], can be compared to how deep learning models, such as Flamingo [ADL$^+$22], extract essential features from the input image. These models learn to focus on particular elements in the image, such as colors or shapes, based on the inherent characteristics of the visual data.

On the other hand, top-down processes in visual attention, guided by task-specific goals and prior knowledge, are analogous to the goal-oriented aspects of language queries sent to the decoder in [ADL$^+$22]. In our example, the language query "Find the red ball" provides a specific goal for the model to follow. By incorporating attention mechanisms, models can learn to prioritize certain features or relationships in the input data, such as the color red and the shape of a ball, based on the task at hand. This mirrors how top-down processes guide our visual attention to focus on relevant aspects of a scene [HB05].

1.2 World Models

1.2.1 Internal Model Theory for Human Perception

Building on the primate visual system, we will explore the significance of internal world models in human cognition. While visual perception is essential, it alone is not sufficient for comprehending and interacting with the environment. Our ability to interact with objects beyond our visual field highlights the existence of an internal memory model of the surroundings, which is available to the motor system and facilitates perception-action coordination [Lan14]. Functional MRI studies in [Lan14] reveal a stable egocentric representation that contributes to our conscious perception of a stable internal world model. World models serve as an important connection between visual perception and higher-level cognitive processes, guiding the perception-action coordination and decision-making process based on perceived spatial and temporal information in response to environmental changes.

Recent research offers compelling evidence supporting the existence of world models within the human brain. Studies demonstrate the brain has an ability to perceive a scene and store an abstract representation of it [QRK+05, CT17] . This capacity to retain abstract information and form predictions about future events is crucial for various human cognitive functions, such as decision-making, planning, navigation behavior. More recent investigation [NRO+15] have shown the role of the primary visual cortex (V1) in representing the difference between consecutive visual inputs in order to adapt to environmental changes. Moreover, the predictive coding theory [RB99] shows that the human brain continuously adapts its internal world model by comparing and correcting the predicted sensory input with the actual input received.

1.2.2 Renaissance of Model-Based Reinforcement Learning

The recent resurgence of interest in model-based reinforcement learning (RL) can be attributed to the advantages provided by incorporating world models into artificial intelligence systems [HS18b, HLBN19, SAH+20b, KBM+20, YLK+21, HWS22]. Reinforcement learning is a type of machine learning where an agent learns to make decisions by interacting with its environment, receiving feedback in the form of rewards [MKS+13, SHM+16b, BBC+19, CHHS20]. In RL, there are two main approaches: model-free and model-based methods [KLM96, ADBB17]. Model-free RL methods directly learn a policy, which is a mapping from states to actions, without explicitly modeling the environment's dynamics [MKS+13, SLA20]. In contrast, model-based RL methods learn an internal model of the environment - world model - which is used to generate predictions about future states and rewards based on the agent's actions [HS18b, SAH$^+$20b]. In the realm of RL, agents can significantly benefit from world models, which enable them to develop a comprehensive representation of past states and make accurate predictions about future states, thus leading to higher learning efficiency in various tasks, such as navigation, control, and decision-making [YLK$^+$21, HWS22].

Historically, model-free RL methods have been prevalent due to their simplicity form. However, as our understanding of world models has grown, researchers have started exploring the potential of model-based RL approaches [HS18b, HLBN19, SAH$^+$20b]. These approaches have been shown to outperform model-free methods in certain scenarios, as they leverage the world model to generate predictions about the consequences of their actions. By simulating potential outcomes and evaluating them based on their value, model-based RL agents can select the most appropriate actions to achieve their goals. This approach is closely aligned with human decision-making processes, where the internal world model guides our actions and decisions [Lan14, NRO$^+$15].

The renaissance of model-based RL is driven by the growing realization that incorporating world models can significantly improve the learning efficiency of RL agents. With the aid of world models, agents can quickly adapt their decision-making strategies (i.e., policy) to new environments or situations without requiring extensive exploration or trial-and-error [YLK$^+$21, HWS22]. This ability to rapidly learn and adapt enables model-based RL agents to excel in complex tasks and dynamically changing environments, demonstrating the immense potential of integrating world models into artificial intelligence systems.

1.2.3 Connection Between Human Internal Model and Model-Based RL

The parallels between human internal world models and model-based reinforcement learning (RL) approaches offer intriguing insights into the development of artificial intelligence systems. Both humans and model-based RL agents rely on internal models to predict future events, evaluate potential outcomes, and guide decision-making processes. In humans, the internal world model is shaped by the integration of visual perception and higher-level cognitive processes, such as attention, memory, and planning [Lan14]. Similarly, in model-based RL, the world model is learned from the agent's interactions with its environment, allowing it to understand the underlying dynamics and predict future states and rewards based on its actions [HS18b].

The connection between human internal models and model-based RL is not just a matter of shared functionality; it also reflects the underlying cognitive processes that drive decision-making and adaptability in both humans and artificial agents. In humans, the brain continually updates its world model based on the difference between predicted sensory input and actual input, allowing for rapid adaptation to changes in the environment [RB99]. Model-based RL agents also update their world model by incorporating new observations, refining their predictions, and adjusting their decision-making strategies accordingly [SAH+20b].

1.3 Overview

This thesis explores visual perception with Transformers and world model representation. The research is organized into five primary chapters, each addressing a different aspect of Transformers and self-attention in visual representation learning and world model in model-based reinforcement learning:

Chapter 2 investigates the application of attention mechanisms for the development of a geometry perception framework. In this chapter, we introduce an end-to-end method for line segment detection in images without the need for additional processing or heuristic-guided intermediate steps. The proposed technique refines line segments through multiple layers of self-attention, enabling the model to gradually learn the optimal location, length, and orientation of line segment candidates.

Chapter 3 presents the development of a novel image transformer architecture capable of handling multi-scale and contextual information. We design a co-scale mechanism that preserves the integrity of individual encoder branches at different scales while facilitating effective communication between representations learned at these scales. Furthermore, we devise a conv-attentional mechanism by incorporating a relative position embedding formulation in the factorized attention module, resulting in an efficient convolution-like implementation.

Chapter 4 demonstrates the utilization of self-attention for modeling spatial relationships among object parts. We propose an end-to-end framework that combines implicit and explicit part-based representations for few-shot classification tasks by seamlessly integrating constellation models with convolution operations. Our approach incorporates a cell feature clustering module to encode potential object parts and models the relationships between these parts using a self-attention mechanism, resembling the spatial configuration design in constellation models.

Chapter 5 explores the potential of model-based reinforcement learning algorithms for efficient task transfer. We introduce Model-Based Cross-Task Transfer (XTRA), a framework that leverages learned internal world models to accelerate the learning of new, distinctly different tasks. By employing offline multi-task pretraining and online cross-task finetuning, XTRA achieves substantial improvements over model-based RL algorithm trained from scratch.

In Chapter 6, we provide a summary of the contributions made by this research and discuss potential future directions in the field of visual representation learning with attention mechanisms and reinforcement learning with world model representations.

Chapter 2

Line Segment Detection Using Transformers without Edges

2.1 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.

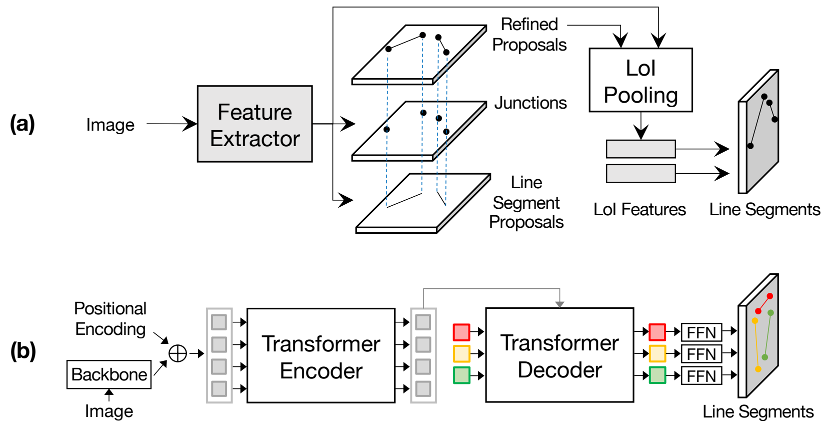


Figure 2.1: **Pipeline comparison** between (a) holistically-attracted wireframe parsing (HAWP) [XWB$^+$20] and (b) our proposed LinE segment TRansformers (LETR). LETR is based on a general-purpose pipeline without heuristics-driven intermediate stages for detecting junctions and generating line segment proposals.

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 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$^+$17, CMS$^+$20] joint end-to-end line segment detection algorithm. We are motivated by the following observations for the Transformer frameworks [VSP$^+$17, CMS$^+$20]: tokenized queries with an integrated encoding and decoding strategy, self-attention mechanism, and bipartite (Hungarian) matching step, capable of addressing the challenges in line segment detection for edge element detection, perceptual grouping, and set prediction; general-purpose pipelines for Transformers that are heuristics free. Our system, named LinE segment TRsformer (LETR), enjoys the modeling power of a general-purpose Transformer architecture while having its own enhanced property for detecting fine-grained geometric structures like line segments. LETR is built on top of a seminal work, DEtection TRansformer (DETR) [CMS$^+$20]. However, as shown in Section 2.4.4 for ablation studies, directly applying the DETR object detector [CMS$^+$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 2.2; 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.

2.2 Related Works

2.2.1 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, vGJMR10] 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.

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 2.1, 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.

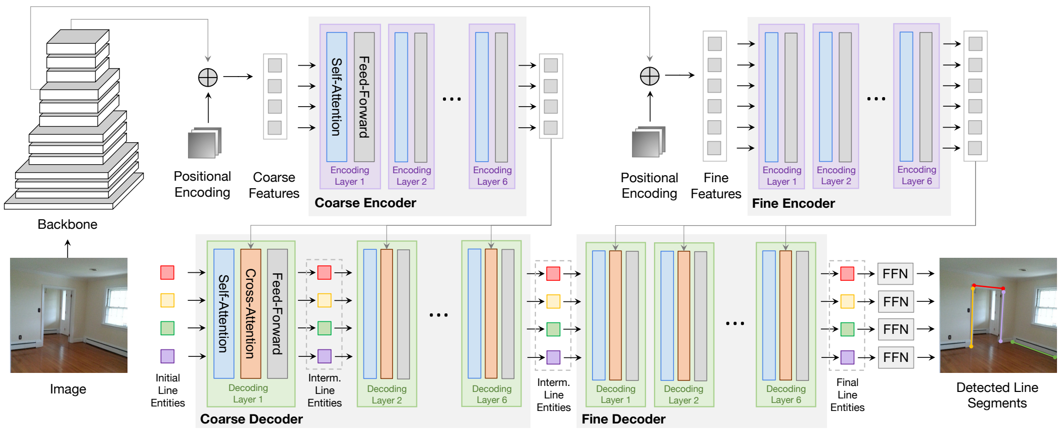


Figure 2.2: **Schematic illustration of our LETR pipeline** An image is fed into a backbone network and generates two feature maps, which are then used by the coarse and the fine encoder respectively. Initial line entities are then first refined by the coarse decoder with the interaction of the coarse 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.

2.2.2 Transformer Architecture

Transformers [VSP$^+$17] have achieved great success in the natural language processing field and become *de facto* standard backbone architecture for many language models [VSP$^+$17, DCLT19b]. It introduces self-attention and cross-attention modules as basic building blocks, modeling dense relations among elements of the input sequence. These attention-based mechanisms also benefit many vision tasks such as video classification [WGGH18], semantic segmentation [FLT$^+$19], image generation [ZGMO19], etc. Recently, end-to-end object detection with Transformers (DETR) [CMS$^+$20] reformulates the object detection pipeline with Transformers by eliminating the need for hand-crafted anchor boxes and non-maximum suppression steps. Instead, [CMS$^+$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.

2.3 Line Segment Detection with Transformers

2.3.1 Motivation

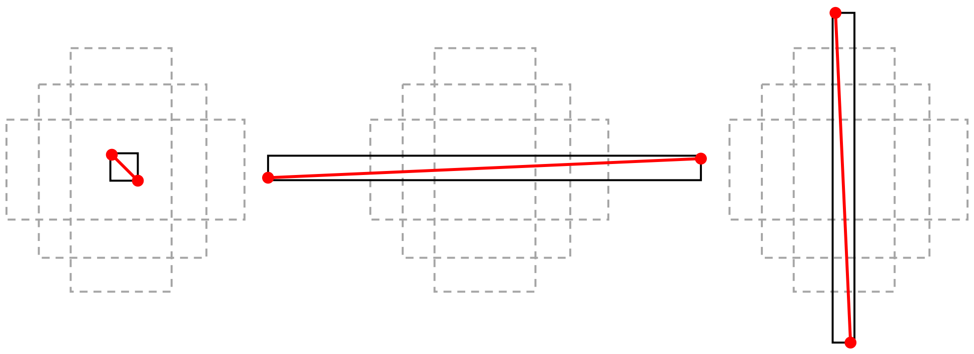


Figure 2.3: **Bounding box representation** Three difficult cases to represent line segments using bounding box diagonals. Red lines, black boxes, and gray dotted boxes refer to as line segments, the corresponding bounding boxes, and anchors respectively.

Despite the exceptional performance achieved by the recent deep learning based approaches [ZQM19, XBW$^+$19, XWB$^+$20] on line segment detection, their pipelines still involve heuristics-driven intermediate representations such as junctions and attraction field maps, raising an interesting question: *Can we directly model all the vectorized line segments with a neural network?* A naive solution could be simply regarding the line segments as *objects* and building a pipeline following the standard object detection approaches [RHGS15]. Since the location of 2-D objects is typically parameterized as a bounding box, the vectorized line segment can be directly read from a diagonal of the bounding box associated with the line segment object. However, the limited choices of anchors make it difficult for standard two-stage object detectors to predict very short line segments or line segments nearly parallel to the axes (see Figure 2.3). The recently appeared DETR [CMS$^+$20] eliminates the anchors and the non-maximum suppression, perfectly meets the need of line segment detection. However, the vanilla DETR still focuses on

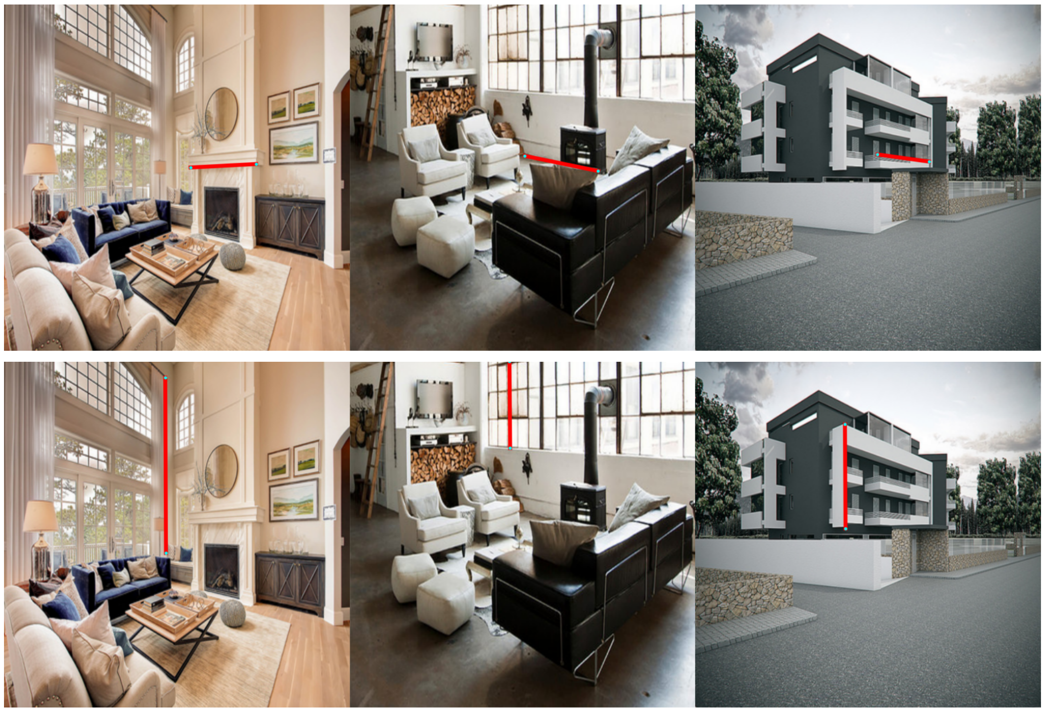


Figure 2.4: **Line entity representation** For each row, we show how a same line entity predicts line segments with same property in three different indoor/outdoor scenes. The top line entity is specialized for horizontal line segments in the middle of the figure, and the bottom one prefers to predict vertical line segments with a various range of lengths.

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.

2.3.2 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 2.1). 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 $\mathbf{x} \in$

$\mathbb{R}^{H\times W\times C}$ 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 $Q \in \mathbb{R}^{m\times d}$, 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:

$$F = \text{Att}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V \tag{2.1}$$

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 2.2). 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.2.1 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.2.1 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 2.4). We discuss line entities together make better predictions through self-attention and cross-attention refinement when encountering heterogeneous line segment structures in Section 2.4.4 and Figure 2.5.

2.3.3 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 2.2). 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.

2.3.4 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):

$$\mathcal{L}_{\text{match}} = \sum_{i=1}^{N} \mathbb{1}_{\{\sigma(i) \leq M\}} \left[\lambda_1 d(\hat{\mathbf{L}}^{(i)}, \mathbf{L}^{(\sigma(i))}) - \lambda_2 p^{(i)}\right] \tag{2.2}$$

$$\sigma^* = \arg\min_{\sigma} \mathcal{L}_{\text{match}} \tag{2.3}$$

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.

2.3.5 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 $\mathcal{L}_{\text{cls}}$:

$$\mathcal{L}_{\text{cls}}^{(i)} = -\mathbb{1}_{\{\sigma^*(i)\leq M\}}\alpha_1(1-p^{(i)})^\gamma \log p^{(i)} \tag{2.4}$$

$$-\mathbb{1}_{\{\sigma^*(i)>M\}}\alpha_2 {p^{(i)}}^\gamma \log(1-p^{(i)}) \tag{2.5}$$

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

$$\mathcal{L}_{\text{dist}}^{(i)} = \mathbb{1}_{\{\sigma^*(i)\leq M\}} d(\hat{\mathbf{L}}^{(i)}, \mathbf{L}^{(\sigma^*(i))}) \tag{2.6}$$

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:

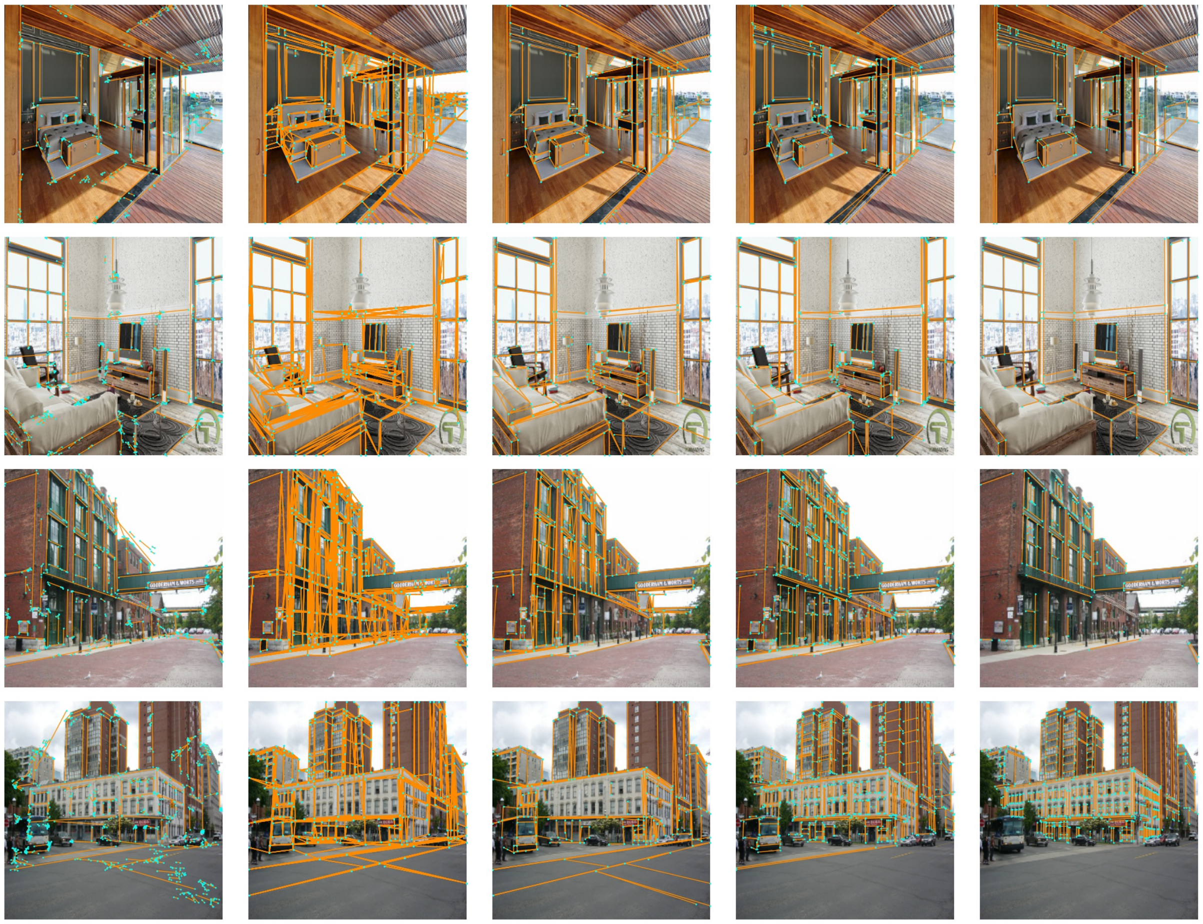
$$\mathcal{L} = \sum_{i=1}^{N} \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}^{(i)} + \lambda_{\text{dist}}\mathcal{L}_{\text{dist}}^{(i)} \tag{2.7}$$

2.4 Experiments

2.4.1 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 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



(a) AFM [XBW+19] (b) LCNN [ZQM19] (c) HAWP [XWB+20] (d) LETR (ours) (e) Ground-Truth

Figure 2.5: **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.

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.

2.4.2 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 2.5. 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 is 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 issue, we also reduce the classification weight for background/no-object instances by a factor of

10.

Table 2.1: **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 [vGJMR10]	/	/	/	/	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

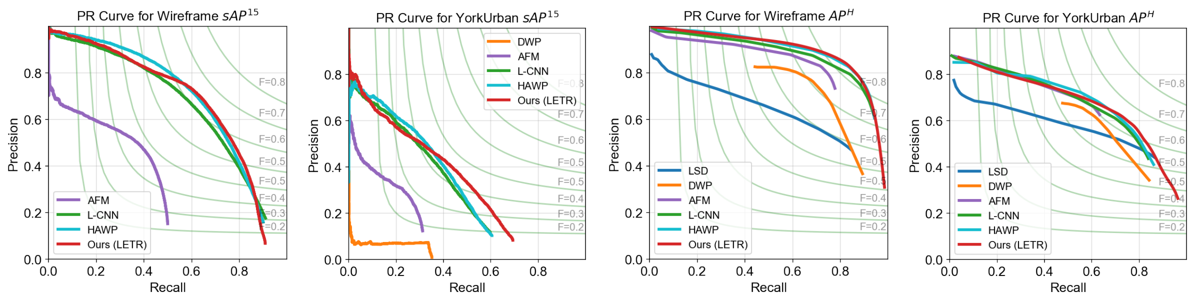


Figure 2.6: **Precision-Recall (PR) curves** PR curves of sAP^{15} and AP^{H} for DWP[HWZ^{+}18], AFM[XBW^{+}19], L-CNN[ZQM19], HAWP[XWB^{+}20] and LETR (ours) on Wireframe and YorkUrban benchmarks.

2.4.3 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.

2.4.4 Results and Comparisons

We summarize quantitative comparison results between LETR and previous line segment detection methods in Table 2.1. 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 2.6 on sAP^{15} and AP^H for both Wireframe [HWZ+18] and YorkUrban benchmarks. In Figure 2.6, 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 2.5. 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.

2.5 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 2.2. We see clear limitations for using bounding box diagonal for both Faster R-CNN and DETR responding to our motivation in Section 2.3.1.

Table 2.2: **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 2.3. 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 2.7 (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 2.7 (c), and using 1000 line entities is optimal for the Wireframe benchmark which contains 74 line segments in average.

Table 2.3: **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

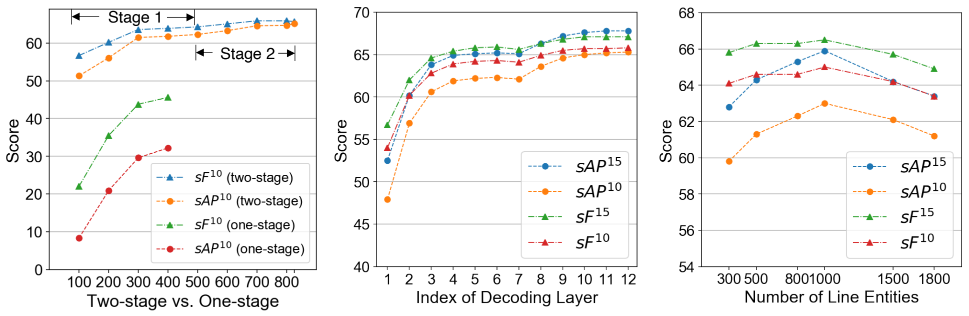


Figure 2.7: Ablation Studies (a) **Multi-stage vs. single-stage training.** We compare results training coarse and fine layers in single stages and multi-stages (b) **Number of decoding layers.** We evaluate the performance of outputs from each decoding layer. The 1-6 layers are coarse decoder layers 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 2.4 below, higher training upsampling resolution improves both methods. LETR obtains additional gains with higher test upsampling resolution.

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

Table 2.4: **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**

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 2.5: **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

2.6 Visualization

We demonstrate LETR's coarse-to-fine decoding process in Figure 2.8. 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

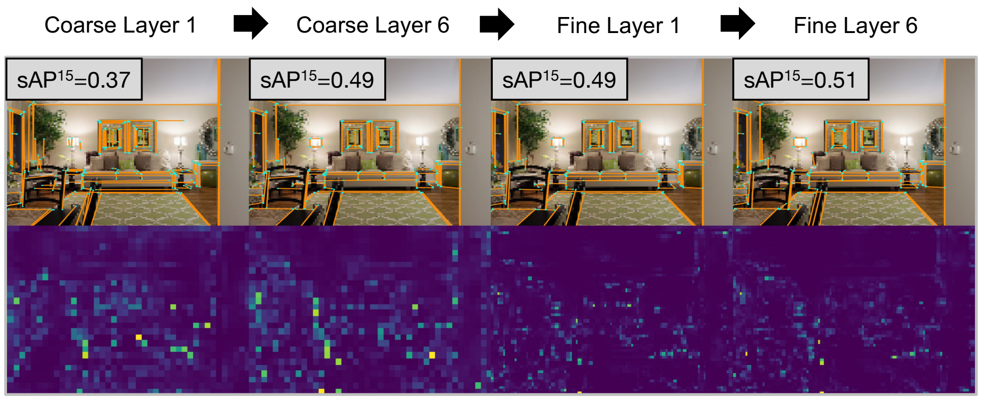


Figure 2.8: **Visualization of LETR coarse-to-fine decoding process** From top to bottom: The 1*st* row shows line segment detection results based on line entities after different layers and the 2*nd* row shows its corresponding overlay of attention heatmaps. From left to right: The 1*st*, 2*nd*, 3*rd*, 4*th* columns are coarse decoder layer 1, coarse decoder layer 6, fine decoder layer 1, fine decoder layer 6, respectively.

2.7(b), where the decoded output after each layer has consistent improvement with the multi-scale encoder-decoder strategy.

2.7 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 3

Co-Scale Conv-Attentional Image Transformers

3.1 Introduction

A notable recent development in artificial intelligence is the creation of attention mechanisms [XBK$^+$15] and Transformers [VSP$^+$17], which have made a profound impact in a range of fields including natural language processing [DCLT19a, RNSS18], document analysis [XLC$^+$20], speech recognition [DXX18], and computer vision [DBK$^+$21, CMS$^+$20]. In the past, state-of-the-art image classifiers have been built primarily on convolutional neural networks (CNNs) [LBBH98, KSH12, SLJ$^+$15, SZ15, HZRS16, XGD$^+$17] that operate on layers of filtering processes. Recent developments [TCD$^+$20, DBK$^+$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" 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

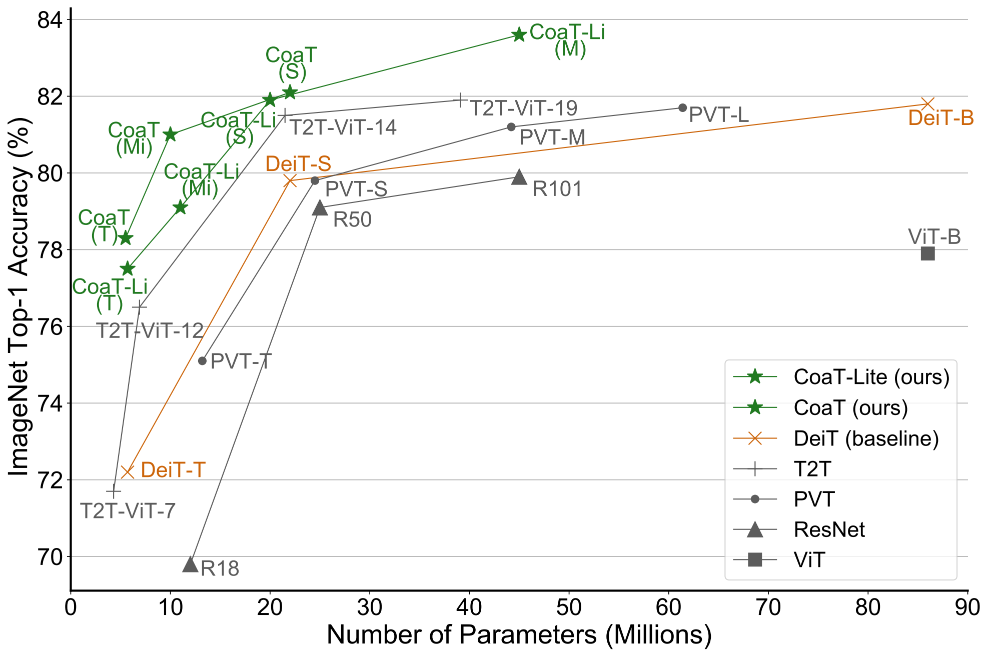


Figure 3.1: **Model Size vs. ImageNet Accuracy** Our CoaT model significantly outperforms other image Transformers. Details are in Table 3.2.

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, DCLT19a, 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+20], 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+20, 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+20], 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+20, WXL+21], as shown in Figure 3.1.

3.2 Related Works

Our work is inspired by the recent efforts [DBK+21, TCD+20] 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+20] attains results comparable to convolution-based classifiers by using an effective training strategy together with model distillation, removing the data requirement in [DBK$^+$21]. Both ViT [DBK$^+$21] and DeiT [TCD$^+$20] 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$^+$20] provides a further enhanced modeling capability by keeping simultaneous fine and coarse scales throughout the convolution layers. In a parallel development [WXL$^+$21] to ours, layers of different scales are in tandem for the image Transformers but [WXL$^+$21] merely carries out a fine-to-coarse strategy. The co-scale mechanism proposed here differs from the existing methods in how the responses are computed and interact with each other: CoaT consists of a series of highly modularized serial and parallel blocks to enable attention with fine-to-coarse, coarse-to-fine, and cross-scale information on tokenized representations. The joint attention mechanism across different scales in our co-scale module provides enhanced modeling power beyond existing vision Transformers [DBK$^+$21, TCD$^+$20, WXL$^+$21].
- **Conv-Attention vs. Attention.** Pure attention-based models [RPV$^+$19, HZXL19, ZJK20, DBK$^+$21, TCD$^+$20] have been introduced to the vision domain. [RPV$^+$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+20] 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 3.4.1 and 3.4.2.

3.3 Revisited 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 :

$$\mathrm{Att}(X) = \mathrm{softmax}\Big(\frac{QK^\top}{\sqrt{C}}\Big)V \tag{3.1}$$

In vision Transformers [DBK+21, TCD+20], 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 3.1 become not affordable for high-resolution images (i.e. $N \gg C$) due to its

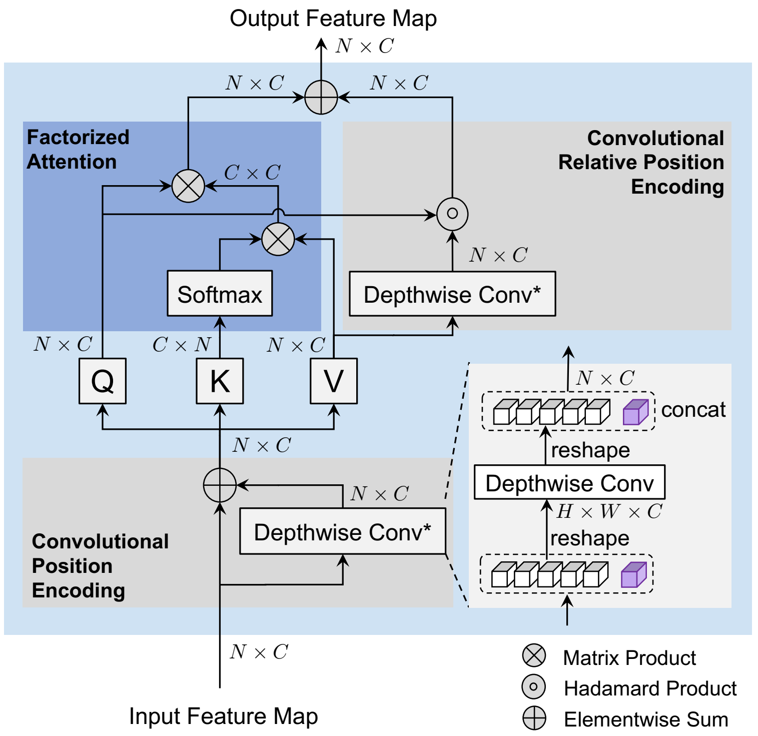
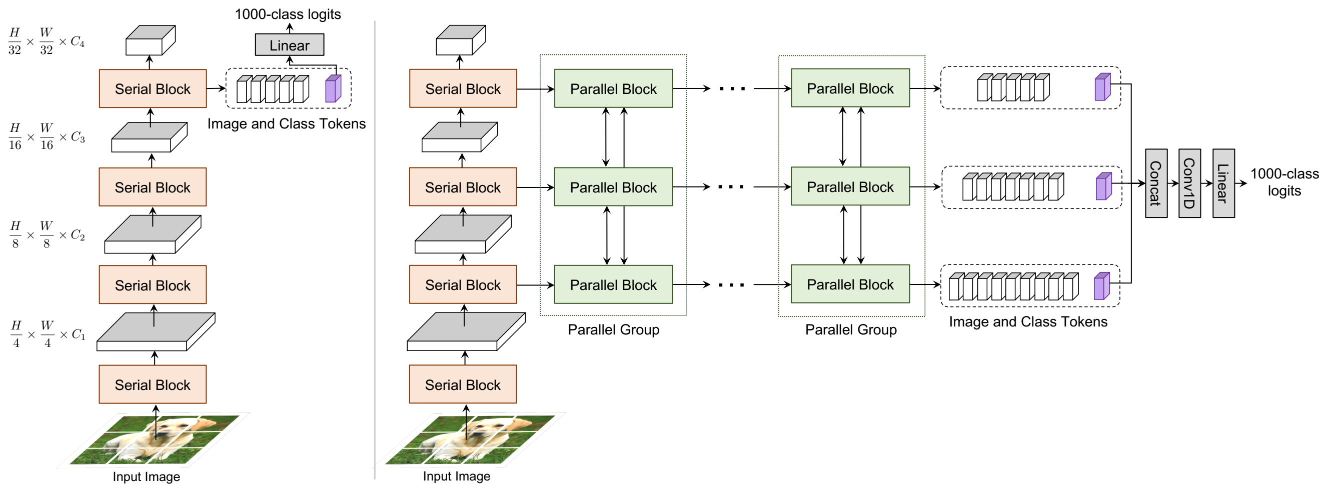


Figure 3.2: **Illustration of the conv-attentional module** We apply a convolutional position encoding to the image tokens from the input. The resulting features are fed into a factorized attention with a convolutional relative position encoding.

$O(N^2)$ space complexity and $O(N^2C)$ time complexity. To reduce the length of the sequence, ViT [DBK+21, TCD+20] 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.



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

Figure 3.3: **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.

3.4 Conv-Attention Module

3.4.1 Factorized Attention Mechanism

In Equation 3.1, 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 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:

$$\text{FactorAtt}(X) = \phi(Q)\left(\psi(K)^\top V\right) \tag{3.2}$$

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:

$$\text{FactorAtt}(X) = \frac{Q}{\sqrt{C}}\left(\text{softmax}(K)^\top V\right) \tag{3.3}$$

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.

3.4.2 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:

$$\text{FactorAtt}(X)_1 = \frac{\mathbf{q}_1}{\sqrt{C}}L = \frac{\mathbf{q}_2}{\sqrt{C}}L = \text{FactorAtt}(X)_2 \tag{3.4}$$

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$^+$21, TCD$^+$20] 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:

$$\text{RelFactorAtt}(X) = \frac{Q}{\sqrt{C}}\Big(\text{softmax}(K)^\top V\Big) + EV \tag{3.5}$$

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

$$E_{ij} = \mathbb{1}(i,j)\mathbf{q}_i \cdot \mathbf{p}_{j-i}, \;\; 1 \le i,j \le N \tag{3.6}$$

in which $\mathbb{1}(i,j) = \mathbb{1}_{\{|j-i|\le(M-1)/2\}}(i,j)$ 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:

$$E_{ij}^{(l)} = \mathbb{1}(i,j)q_i^{(l)}p_{j-i}^{(l)}, \;\; \hat{EV}_i^{(l)} = \textstyle\sum_j E_{ij}^{(l)}v_j^{(l)} \tag{3.7}$$

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

$$\hat{EV}^{(l)} = Q^{(l)} \circ \mathrm{Conv1D}(P^{(l)}, V^{(l)}), \tag{3.8}$$

$$\hat{EV} = Q \circ \mathrm{DepthwiseConv1D}(P, V) \tag{3.9}$$

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. $Q^{\mathrm{img}}, V^{\mathrm{img}} \in \mathbb{R}^{H\times W\times C}$ from Q, V respectively):

$$\hat{EV}^{\mathrm{img}} = Q^{\mathrm{img}} \circ \mathrm{DepthwiseConv2D}(P, V^{\mathrm{img}}) \tag{3.10}$$

$$\hat{EV} = \mathrm{concat}(\hat{EV}^{\mathrm{img}}, \mathbf{0}) \tag{3.11}$$

$$\mathrm{ConvAtt}(X) = \frac{Q}{\sqrt{C}}\left(\mathrm{softmax}(K)^{\top} V\right) + \hat{EV} \tag{3.12}$$

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 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 $\hat{EV}$ with only $O(NC)$ space complexity and $O(NCM^2)$ time complexity, aiming to achieve better

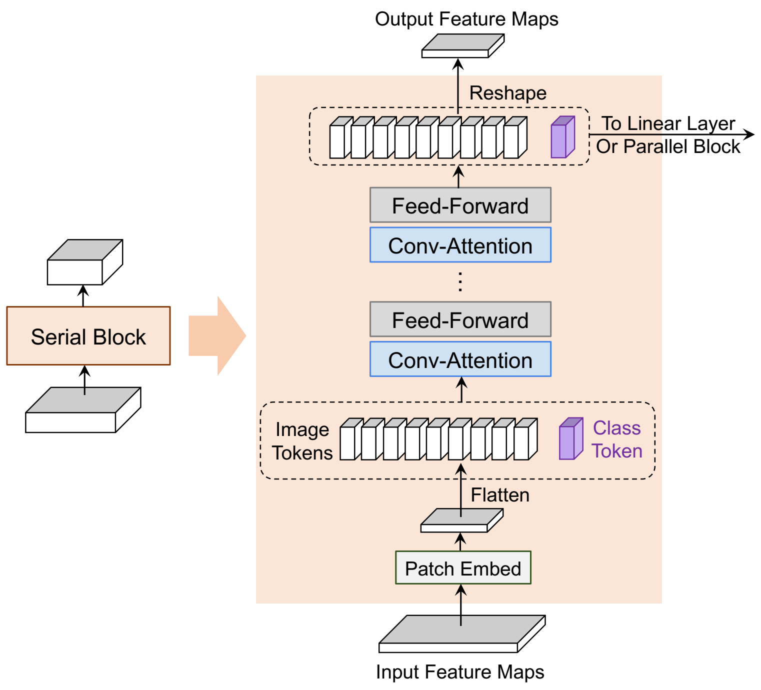


Figure 3.4: **Schematic illustration of the serial block in CoaT** Input feature maps are first down-sampled by a patch embedding layer, and then tokenized features (along with a class token) are processed by multiple conv-attention and feed-forward layers.

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$^+$21, TCD$^+$20], we would like to insert the position relationship into the input image features directly to enrich the effects of relative position encoding. In each conv-attentional module, we insert a depthwise convolution into the input features X and concatenate the resulting position-aware features back to the input features following the standard absolute position encoding scheme (see Figure 3.2 lower part), which resembles the realization of conditional position encoding in CPVT [CZT$^+$21].

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

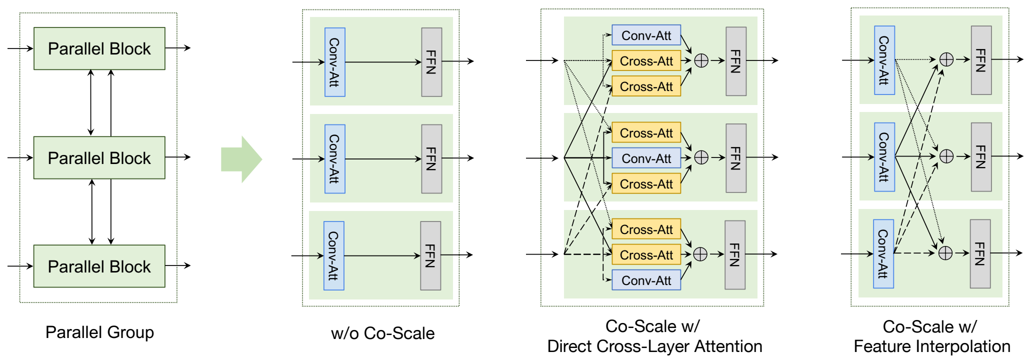


Figure 3.5: **Schematic illustration of the parallel group in CoaT** For "w/o Co-Scale", tokens learned at the individual scales are combined to perform the classification but absent intermediate co-scale interaction for the individual paths of the parallel blocks. We propose two co-scale variants, namely 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.

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 3.2: 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 3.1: **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

3.5 Co-Scale Conv-Attentional Transformers

3.5.1 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 3.4) 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 3.4 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 3.5). 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 3.6.4.

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 3.2) 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.

3.5.2 Model Architecture

CoaT-Lite. CoaT-Lite, Figure 3.3 (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 3.3 (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 3.1. 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 3.2: **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+20]	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+20]	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+20]	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 3.3: **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×	
		$\mathrm{AP^b}$	$\mathrm{AP^m}$	$\mathrm{AP^b}$	$\mathrm{AP^m}$
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 3.4: **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×	
		$\mathrm{AP^b}$	$\mathrm{AP^m}$	$\mathrm{AP^b}$	$\mathrm{AP^m}$
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 3.5: **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**

3.6 Experiments

3.6.1 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+20].

All experimental results for our models in Table 3.2 are reported at 300 epochs, consistent with previous methods [TCD+20]. 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× setting and 36 epochs in 3× 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.

3.6.2 CoaT for ImageNet Classification

Table 3.2 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.

3.6.3 Object Detection and Instance Segmentation

Tables 3.3 and 3.4 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 3.5. 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].

3.6.4 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 3.6. 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 3.6: **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 3.7, 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 3.7: **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 3.8. 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 3.8: **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%**

3.7 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.

Acknowledgments

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 4

Attentional Constellation Nets for Few-Shot Learning

4.1 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 relations for the 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 [OLR18], 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 4.3 and Figure 4.2 (e)).

4.2 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, MYMT17]. 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 competitive 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.

4.3 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, OLR18] usually learn a feature extractor $\phi(\mathbf{x})$ on the dataset $\mathcal{D}^{\text{base}}$ to obtain

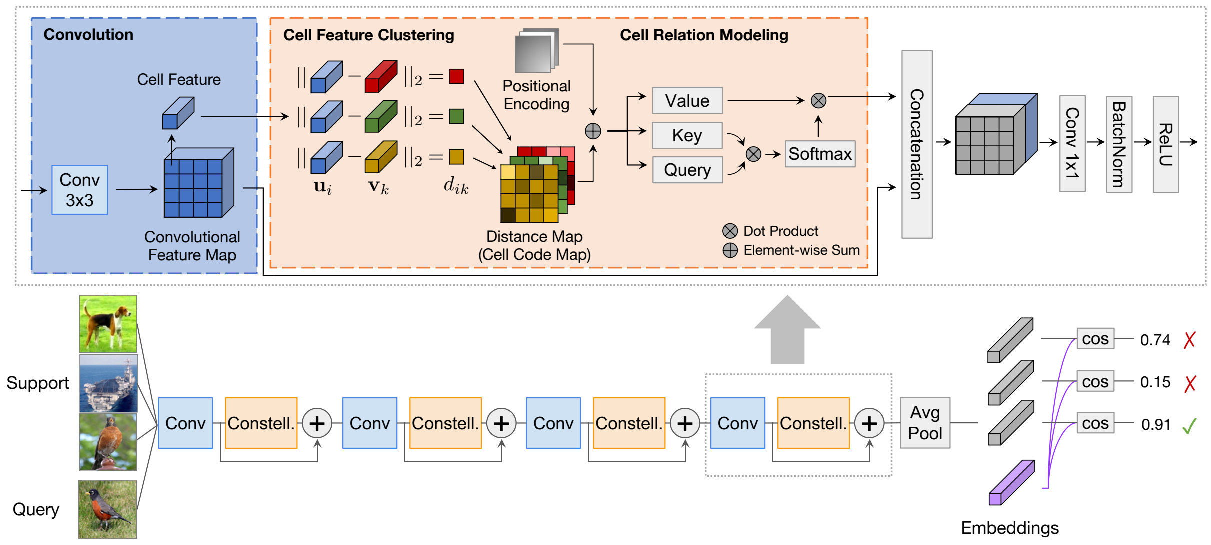


Figure 4.1: **Illustration of our ConstellationNet pipeline** where the bottom part is the network architecture based on Conv-4 backbone, and the top part shows the constellation model. Our proposed ConstellationNet consists of "Constell." modules that perform explicit cell feature clustering with self-attention for joint relation modeling.

generic feature embedding by optimizing the loss $\mathcal{L}(\phi)$:

$$\mathcal{L}(\phi) = \mathbb{E}_{\{(\mathbf{x},y)\}\sim\mathcal{D}_{\text{base}}} \ell\big(\{(\phi(\mathbf{x}),y)\}\big) \tag{4.1}$$

where $\{(\mathbf{x},y)\}$ is a sampled mini-batch of data points and $\ell(\cdot)$ is usually an episodic few-shot loss [VBL$^+$16] or a standard cross-entropy loss [CWL$^+$20].

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 $d(\cdot,\cdot)$:

$$\hat{y}' = \arg\max_k d\big(\phi(\mathbf{x}'), \mathbf{c}_k\big), \qquad \mathbf{c}_k = \frac{1}{N} \sum_{(\mathbf{x},y)\in\mathcal{T}^{\text{supp}},\ y=k} \phi(\mathbf{x}). \tag{4.2}$$

An extended description of the few-shot learning framework can be found from Appendix 4.8.1. 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 [OLR18]) 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.

4.4 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 a *cell feature* as the individual local feature at a position in the feature map (see Figure 4.1). 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 relationships.

4.4.1 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:

$$\min \sum_i \sum_k m_{ik} ||\mathbf{u}_i - \mathbf{v}_k||_2^2 \quad \text{s.t.} \quad m_{ik} \in \{0,1\}, \quad \sum_k m_{ik} = 1 \tag{4.3}$$

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 $\mathbf{v}'_k$:

$$d_{ik} = ||\mathbf{u}_i - \mathbf{v}_k||_2^2, \qquad m_{ik} = \frac{e^{-\beta d_{ik}}}{\sum_j e^{-\beta d_{ij}}}, \qquad \mathbf{v}'_k = \frac{\sum_i m_{ik}\mathbf{u}_i}{\sum_i m_{ik}} \tag{4.4}$$

-0.0mm 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:

$$\mathbf{v}_k \leftarrow (1-\eta)\mathbf{v}_k + \eta\mathbf{v}'_k, \qquad \eta = \frac{\lambda\Delta s_k}{s_k + \Delta s_k} \tag{4.5}$$

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

$$\mathbf{s} \leftarrow \mathbf{s} + \Delta\mathbf{s} \tag{4.6}$$

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 4.8.9.

4.4.2 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 $\mathbf{F}'_\mathrm{I}$:

$$\mathbf{F}_\mathrm{I} = \mathrm{SpatialFlatten}(\mathbf{D}+\mathbf{P}) \in \mathbb{R}^{B\times HW\times K}, \quad \mathbf{F}'_\mathrm{I} = \mathrm{SpatialFlatten}(\mathbf{D}) \in \mathbb{R}^{B\times HW\times K} \tag{4.7}$$

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 $\mathbf{F}_\mathrm{A}$:

$$[\mathbf{F}^q, \mathbf{F}^k, \mathbf{F}^v] = [\mathbf{F}_\mathrm{I}\mathbf{W}^q, \mathbf{F}_\mathrm{I}\mathbf{W}^k, \mathbf{F}'_\mathrm{I}\mathbf{W}^v] \tag{4.8}$$

$$\mathbf{F}_\mathrm{A} = \mathrm{Att}(\mathbf{F}^q, \mathbf{F}^k, \mathbf{F}^v) = \mathrm{softmax}\Big(\frac{\mathbf{F}^q(\mathbf{F}^k)^\top}{\sqrt{K}}\Big)\mathbf{F}^v \tag{4.9}$$

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 relationships of input features and leads to an enhanced output feature representation $\mathbf{F}_\mathrm{A}$. 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:

$$\mathbf{F}_{\text{MHA}} = \text{MultiHeadAtt}(\mathbf{F}^q, \mathbf{F}^k, \mathbf{F}^v) = [\mathbf{F}_1, ..., \mathbf{F}_J]\mathbf{W}, \qquad \mathbf{F}_j = \text{Att}(\mathbf{F}_j^q, \mathbf{F}_j^k, \mathbf{F}_j^v) \tag{4.10}$$

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. $\{\mathbf{F}_j^q, \mathbf{F}_j^k, \mathbf{F}_j^v\} \subset \mathbb{R}^{B \times HW \times K'}$. The output features $\mathbf{F}_j$ of each head are computed following Eq. 4.9. 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.

4.4.3 Integrate Constellation Model with CNNs

Our constellation model has the capability to capture explicit structured features and encodes spatial relations among the cell features. The output features yield informative visual cues which are able to strengthen the convolutional features. Thus, as shown in Figure 4.1, 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 4.3, 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 4.3) 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.

4.4.4 Why clustering and self-attention (clustering map + positional encoding)?

As described in Section 4.1 and 4.2, classical constellation models [FPZ03, FH05] extract parts with their spatial relationships; 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 relationships between the parts together with their spatial configurations.

4.5 Experiment

4.5.1 Datasets

We adopt three standard benchmark datasets that are widely used in few-shot learning, CIFAR-FS dataset [BHTV19], FC100 dataset [OLR18], and *mini*-ImageNet dataset [VBL$^+$16]. Details about dataset settings in few-shot learning are in Appendix 4.8.2.

4.5.2 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 4.8.1. 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 4.8.10. The detailed ablation study is described in Table 4.3.

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.

4.5.3 Results on Standard Benchmarks

Table 4.1 and 4.2 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 4.2, our model outperforms SIB [HMX$^+$20] 1-shot by 0.6% and 5-shot by 5.6%. In Table 4.1, our model outperforms MetaOptNet [LMRS19] by 5.95% in 1-shot and 6.24% in 5-shot. For deep networks with rich features, the constellation module still contributes to the performance, showing its complementary advantage to convolution. Our ResNet-12 model beats [LMRS19] 1-shot result by 2.7% on FC100, 3.4% on CIFAR-FS, and 1.72% on *mini*-ImageNet. The consistent improvement over both shallow and deep networks across all three datasets shows the generality of our method. Our ConstellationNet is orthogonal to the margin loss based methods [LCL$^+$20, LHL$^+$20], and we also do not use extra cross-modal information [XROOP19, LHL$^+$20]. On the contrary, our model enhances the embedding generalization ability by incorporating its own part-based representation. Additionally, to verify the orthogonality of our method, we adapt the negative margin loss following [LCL$^+$20] to our Conv-4 models in Appendix 4.8.8. We observe ConstellationNet with negative margin brings 0.52% improvement to ConstellationNet, and obtains 6.93% gain compared with baseline on *mini*-ImageNet.

4.6 Model Analysis

4.6.1 Architecture alternatives

In Table 4.3, 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 4.1: **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 works. † 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+18]	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 [OLR18]	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 [CWL+20]	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 4.3. 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 4.2 (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 4.2: **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 works.

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 [OLR18]	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 4.3: **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.

4.6.2 Modules Analysis

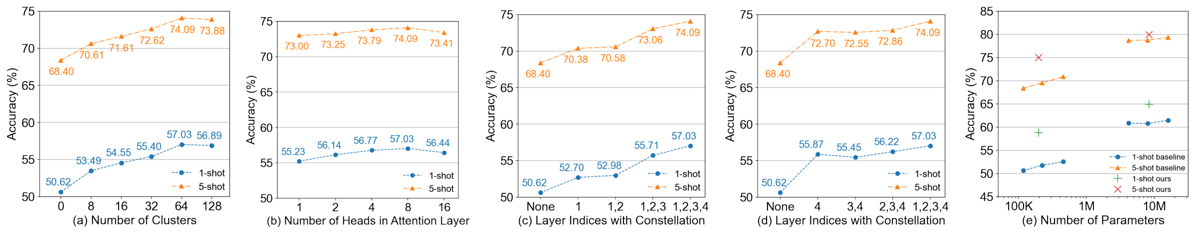


Figure 4.2: **Modules analysis**. (a, b, c, d) We study the effectiveness of changing the number of clusters, the number of heads in attention layer, and the layer indices with constellation based on Conv-4, (e) We demonstrate the performance gain of our ConstellationNet is unmatched by increasing the model complexity of our baselines. All experiments are done on *mini*-ImageNet.

In Figure 4.2 (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 4.2 (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 4.2 (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.

4.6.3 Visualization

Figure 4.3 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

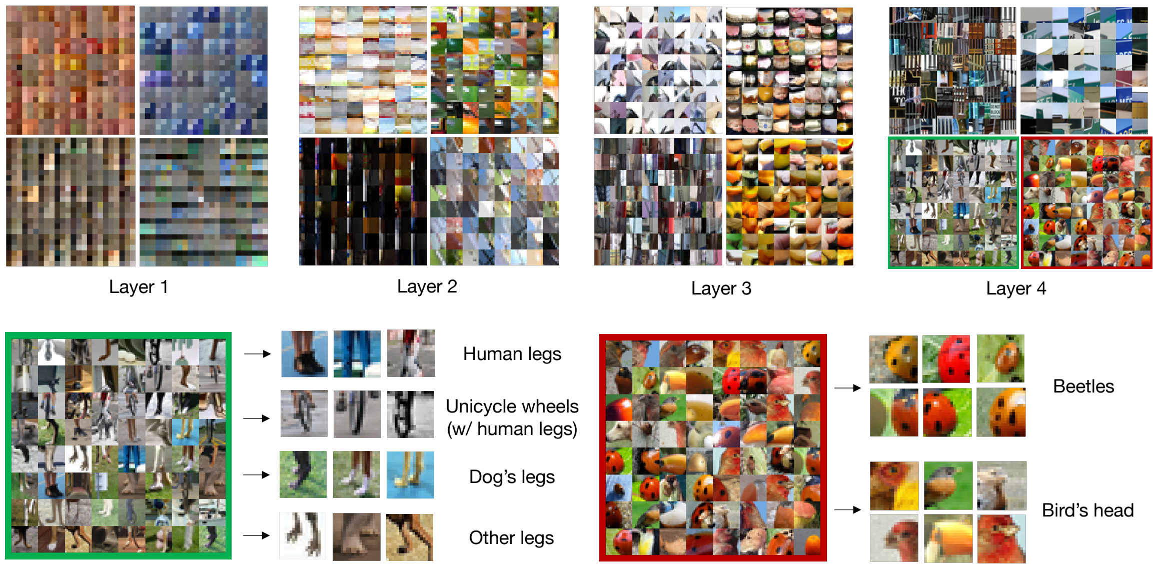


Figure 4.3: **Visualization of cluster centers**. (Upper) We visualize four cluster centers in each layer by showing patches associated with cell features that have the nearest distance to the clustering center. (Lower) Identifying parts from two cluster centers in layer 4: Left one with green box represents various types of legs. Right one with red box mostly shows beetles and bird's head, sharing a dotted structure.

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 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 4.4 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

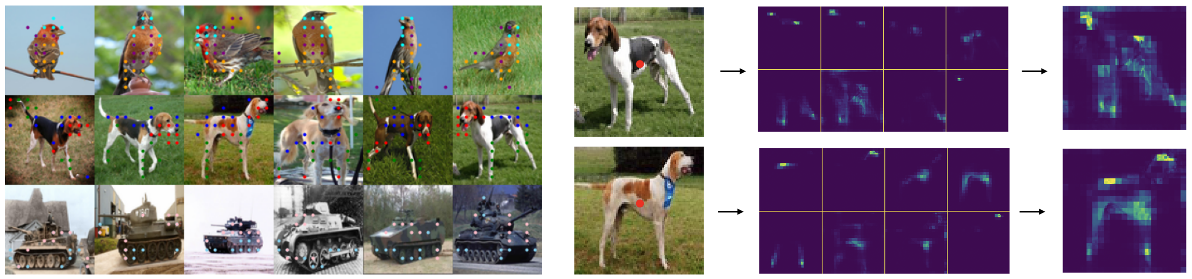


Figure 4.4: **Visualization of the cells assignment and attention maps**. (Left) Each color represents a cluster, and each point, marked as "·", represents a cell assigned to a cluster center. We demonstrate 6 samples for each class (bird, dog and tank). (Right) We visualize attention maps of one query 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.

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 for comparison across 6 samples. As shown in Figure 4.4, 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 4.4 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.

4.7 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.

4.8 Appendix

4.8.1 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:

$$\mathbf{c}_k = \frac{1}{|N|} \sum_{(\mathbf{x},y)\in\mathcal{T}^{\text{supp}}, y=k} \phi(\mathbf{x}) \tag{4.11}$$

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':

$$p_k = p(y = k|\mathbf{x}', \mathcal{T}^{\text{supp}}) = \frac{\exp(d(\phi(\mathbf{x}'), \mathbf{c}_k))}{\sum_{k'} \exp(d(\phi(\mathbf{x}'), \mathbf{c}_{k'}))}, \quad y' = \arg\max_k p_k. \tag{4.12}$$

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:

$$\ell(\mathcal{T}^{\text{supp}}, \mathcal{T}^{\text{query}}) = \mathbb{E}_{(\mathbf{x}',y')\in\mathcal{T}^{\text{query}}} - \log p(y = y'|\mathbf{x}', \mathcal{T}^{\text{supp}}), \tag{4.13}$$

$$\mathcal{L}(\phi) = \mathbb{E}_{\mathcal{T}=(\mathcal{T}^{\text{supp}},\mathcal{T}^{\text{query}})\sim\mathcal{D}^{\text{base}}} \ell(\mathcal{T}^{\text{supp}}, \mathcal{T}^{\text{query}}). \tag{4.14}$$

Another way is the *standard classification scheme* [CWL$^+$20]: 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:

$$\mathcal{L}(\phi) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}^{\text{base}}} - \log \frac{\exp(\mathbf{w}_y \cdot \phi(\mathbf{x}))}{\sum_q \exp(\mathbf{w}_q \cdot \phi(\mathbf{x}))} \tag{4.15}$$

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.

4.8.2 Datasets

The CIFAR-FS dataset [BHTV19] is a few-shot classification benchmark containing 100 classes from CIFAR-100 [KH$^+$09]. The classes are randomly split into 64, 16 and 20 classes as meta-training, meta-validation and meta-testing set respectively. For each class, it contains 600 images of size 32×32. We adopt the split from [LMRS19]. The FC100 dataset [OLR18] is another benchmark based on CIFAR-100 where classes are grouped into 20 superclasses to void the overlap between the splits. The *mini*-ImageNet dataset [VBL$^+$16] is a common benchmark for few-shot classification containing 100 classes from ILSVRC-2012 [DDS$^+$09]. The classes are randomly split into 64, 16 and 20 classes as meta-training, meta-validation and meta-testing set respectively. For each class, it contains 600 images of size 84×84. We follow the commonly-used split in [RL17], [LMRS19] and [CWL$^+$20]. In all experiments, we conduct data augmentation for the meta-training set of all datasets to match [LMRS19]'s implementation.

4.8.3 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 [CWL$^+$20], 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.

4.8.4 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.

4.8.5 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].

4.8.6 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.

4.8.7 Ablation Study on the Number of Clusters

Table 4.4: **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.

4.8.8 Additional Experiments with Negative Margin

Table 4.5 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 4.5: **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**	**Cell Relation**	**Negative**	**Conv-4**	
	Clustering	**Modeling**	**Margin**	**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.

4.8.9 Clarification on Clustering Procedure

In this section, we add more clarification on our cell feature clustering procedure in Sec. 4.4.1: 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.

4.8.10 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.

4.8.11 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.

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.

Chapter 5

On the Feasibility of Cross-Task Transfer with Model-Based Reinforcement Learning

5.1 Introduction

Reinforcement Learning (RL) has achieved great feats across a wide range of areas, most notably game-playing [MKS+13, SHM+16b, BBC+19, CHHS20]. However, traditional RL algorithms often suffer from poor sample-efficiency and require millions (or even billions) of environment interactions to solve tasks – especially when learning from high-dimensional observations such as images. This is in stark contrast to humans that have a remarkable ability to quickly learn new skills despite very limited exposure [DAP+18]. In an effort to reliably benchmark and improve the sample-efficiency of image-based RL across a variety of problems, the Arcade Learning Environment (ALE; [BNVB13]) has become a long-standing challenge for RL. This task suite has given rise to numerous successful and increasingly sample-efficient algorithms [MKS+13, BPK+20, KBM+20, SAH+20a, KYF21, HLNB21, YLK+21], notably most of which are model-based, *i.e.*, they learn a *model* of the environment [HS18a].

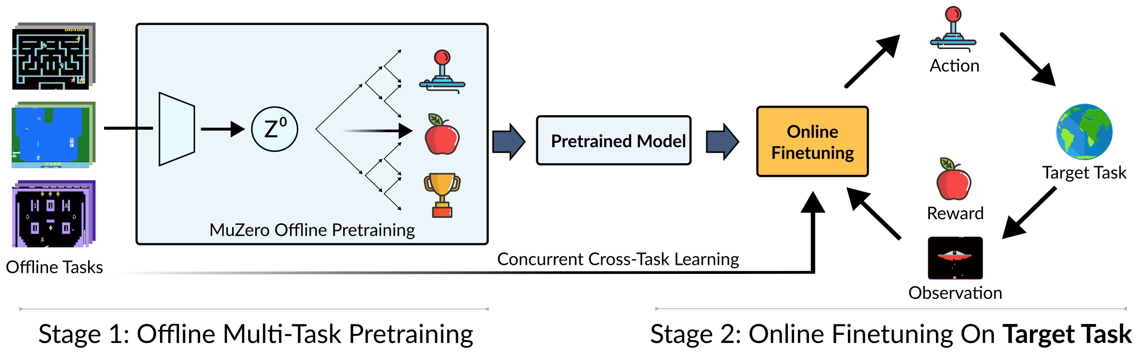

Offline Tasks
Z^0
MuZero Offline Pretraining
Pretrained Model
Online Finetuning
Action
Target Task
Reward
Observation
Concurrent Cross-Task Learning
Stage 1: Offline Multi-Task Pretraining
Stage 2: Online Finetuning On **Target Task**

Figure 5.1: **Illustration of our XTRA pipeline**. Model-Based **Cross**-Task **Tra**nsfer (**XTRA**) is a sample-efficient online RL framework with scalable pretraining and finetuning of learned world models using auxiliary data from offline tasks.

Most recently, EfficientZero [YLK+21], a model-based RL algorithm, has demonstrated impressive sample-efficiency, surpassing human-level performance with as little as 2 hours of real-time game play in select Atari 2600 games from the ALE. This achievement is attributed, in part, to the algorithm concurrently learning an internal *model* of the environment from interaction, and using the learned model to *imagine* (simulate) further interactions for planning and policy improvement, thus reducing reliance on real environment interactions for skill acquisition. However, current RL algorithms, including EfficientZero, are still predominantly assumed to learn both perception, model, and skills *tabula rasa* (from scratch) for each new task. Conversely, humans rely heavily on prior knowledge and visual cues when learning new skills – a study found that human players easily identify visual cues about game mechanics when exposed to a new game, and that human performance is severely degraded if such cues are removed or conflict with prior experiences [DAP+18].

In related areas such as computer vision and natural language processing, large-scale unsupervised/self-supervised/supervised pretraining on large-scale datasets [DCLT19c, BMR+20, LLL+22, RKH+21, CND+22] has emerged as a powerful framework for solving numerous downstream tasks with few samples [ADL+22]. This pretraining paradigm has recently been extended to visuo-motor control in various forms, *e.g.*, by leveraging *frozen* (no finetuning) pretrained representations [XRDM22, PRPG22] or by finetuning in a supervised setting [RZP+22b, LNY+22]. However, the success of finetuning for *online RL* has mostly been limited to same-task initialization of model-free policies from offline datasets [WLRL22, ZZG22], or adapting policies to novel instances of a given task [MRCA17, JSS+20, HJS+21], with prior work citing high-variance objectives and catastrophical forgetting as the main obstacles to finetuning representations with RL [BHDAJ20, XRDM22].

In this work, we explore whether such positive transfer can be induced with current model-based RL algorithms in an *online* RL setting, and across *markedly distinct* tasks. Specifically, we seek to answer the following questions: *when* and *how* can a model-based RL algorithm such as EfficientZero benefit from pretraining on a diverse set of tasks? We base our experiments on the ALE due to cues that are easily identifiable to humans despite great diversity in tasks, and identify two key ingredients – cross-task finetuning and task alignment – for model-based adaptation that improve sample-efficiency substantially compared to models learned tabula rasa. In comparison, we find that a naïve treatment of the finetuning procedure as commonly used in supervised learning [PY10, DGE15, HFW+20, RZP+22b, LNY+22] is found to be unsuccessful or outright *harmful* in an RL context.

Based on our findings, we propose Model-Based **Cross**-Task **Tra**nsfer (**XTRA**), a framework for sample-efficient online RL with scalable pretraining and finetuning of learned world models using extra, auxiliary data from other tasks (see Figure 5.1). Concretely, our framework consists of two stages: *(i) offline multi-task pretraining* of a world model on an offline dataset from *m* diverse tasks, a *(ii) finetuning* stage where the world model is jointly finetuned on a *target task* in addition to *m* offline tasks. By leveraging offline data both in pretraining and finetuning, XTRA overcomes the challenges of catastrophical forgetting. To prevent harmful interference from certain offline tasks, we adaptively re-weight gradient contributions in unsupervised manner based on similarity to target task.

We evaluate our method and a set of strong baselines extensively across 14 Atari 2600 games from the Atari100k benchmark [KBM$^+$20] that require algorithms to be extremely sample-efficient. From Table 5.1, we observe that XTRA improves sample-efficiency substantially across most tasks, improving mean and median performance of EfficientZero by 23% and 25%, respectively.

5.2 Background

Problem setting. We model image-based agent-environment interaction as an episodic Partially Observable Markov Decision Process (POMDP; [KLC98]) defined by the tuple $\mathcal{M} = \langle \mathcal{O}, \mathcal{A}, \mathcal{P}, \rho, r, \gamma \rangle$, where $\mathcal{O}$ is the observation space (pixels), $\mathcal{A}$ is the action space, $\mathcal{P}\colon \mathcal{O} \times \mathcal{A} \mapsto \mathcal{O}$ is a transition function, ρ is the initial state distribution, $r\colon \mathcal{O} \times \mathcal{A} \mapsto \mathbb{R}$ is a scalar reward function, and $\gamma \in [0,1)$ is a discount factor. As is standard practice in ALE [BNVB13], we convert $\mathcal{M}$ to a fully observable Markov Decision Process (MDP; [Bel57]) by approximating state $\mathbf{s}_t \in \mathcal{S}$ at time t as a stack of frames $\mathbf{s}_t \doteq \{o_t, o_{t-1}, o_{t-2}, \dots\}$ where $o \in \mathcal{O}$ [MKS$^+$13], and redefine $\mathcal{P}, \rho, r$ to be functions of $\mathbf{s}$. Our goal is then to find a (neural) policy $\pi_\theta(\mathbf{a}|\mathbf{s})$ parameterized by θ that maximizes discounted return $\mathbb{E}_{\pi_\theta}[\sum_{t=1}^{t} \gamma_t r(\mathbf{s}_t, \mathbf{a}_t)]$ where $\mathbf{a}_t \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, $\mathbf{s}_t \sim \mathcal{P}(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_0 \sim \rho$, and T is the episode horizon. For clarity, we denote all parameterization by θ throughout this work. To obtain a good policy from minimal environment interaction, we learn a *"world model"* from interaction data and use the learned model for action search. Define $\mathcal{M}$ as the *target task* that we aim to solve. Then, we seek to first obtain a good parameter initialization θ that allows us to solve task $\mathcal{M}$ using fewer interactions (samples) than training from scratch, *i.e.*, we wish to improve the *sample-efficiency* of online RL. We do so by first pretraining the model on an *offline* (fixed) dataset that consists of transitions $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$ collected by unknown behavior policies in m environments $\{\tilde{\mathcal{M}}^i \,|\tilde{\mathcal{M}}^i \neq \mathcal{M}, 1 \leq i \leq m\}$, and then *finetune* the model by online interaction on the target task.

EfficientZero [YLK$^+$21] is a model-based RL algorithm based on MuZero [SAH$^+$20a] that learns a discrete-action latent dynamics model from environment interactions, and selects actions by lookahead via Monte Carlo Tree Search (MCTS; [Abr87, Cou06, SHM$^+$16b]) in the latent space of the model. An overview of the three main components of the MuZero algorithm can be break down as a representation (encoder) h_θ, a dynamics (transition) function g_θ, and a prediction head f_θ. Given an observed state $\mathbf{s}_t$, EfficientZero projects the state to a latent representation $\mathbf{z}_t = h_\theta(\mathbf{s}_t)$, and predicts future latent states $\mathbf{z}_{t+1}$ and instantaneous rewards $\hat{r}_t$ using an action-conditional latent dynamics function $\mathbf{z}_{t+1}, \hat{r}_t = g_\theta(\mathbf{z}_t, \mathbf{a}_t)$. For each latent state, a prediction network f_θ estimates a probability distribution $\hat{p}$ over (valid) actions $\mathbf{a} \in \mathcal{A}$, as well as the expected state value $\hat{v}$ of the given state, *i.e.*, $\hat{v}_t, \hat{p}_t = f_\theta(\mathbf{z}_t)$. Intuitively, h_θ and g_θ allow EfficientZero to search for actions entirely in its latent space before executing actions in the real environment, and f_θ predicts quantities that help guide the search towards high-return action sequences. Concretely, $\hat{v}$ provides a return estimate for nodes at the lookahead horizon (as opposed to truncating the cumulative sum of expected rewards) and $\hat{p}$ provides an action distribution prior that helps guide the search. We describe EfficientZero's learning objective between model prediction $(\hat{p}, \hat{v}, \hat{r})$ and quantity targets (π, z, u) in Appendix 5.7.1. EfficientZero improves the sample-efficiency of MuZero by introducing additional auxiliary losses during training. We adopt EfficientZero as our backbone model and learning algorithm, but emphasize that our framework is applicable to most model-based algorithms, including those for continuous action spaces [HLBN19, HWS22].

5.3 Model-Based Cross-Task Transfer

We propose Model-Based **Cross**-Task **Tra**nsfer (**XTRA**), a two-stage framework for offline multi-task pretraining and cross-task transfer of learned world models by finetuning with online RL. Specifically, we first pretrain a world model on offline data from a set of diverse pretraining tasks, and then iteratively finetune the pretrained model on data from a *target* task collected by online interaction. In the following, we introduce each of the two stages – pretraining and finetuning – in detail.

5.3.1 Offline Multi-Task Pretraining

In this stage, we aim to learn a single world model with general perceptive and dynamics priors across a diverse set of offline tasks. We emphasize that the goal of pretraining is not to obtain a truly generalist agent, but rather to learn a good initialization for finetuning to unseen tasks. Learning a single RL agent for a diverse set of tasks is however difficult in practice, which is only exacerbated by extrapolation errors due to the offline RL setting [KZTL20]. To address such a challenge, we propose to pretrain the model following a *student-teacher* training setup in the same spirit to DQN multi-task policy distillation in [RCG+16] and Actor-Mimic in [PBS16], where *teacher* models are trained separately by offline RL for each task, and then distilled into a single multi-task model using a novel instantiation of the MuZero Reanalyze [SHM+21].

For each pretraining task we assume access to a fixed dataset $\{\tilde{\mathcal{D}}^i \mid 1 \leq i \leq m\}$ that consists of trajectories from an unknown (and potentially sub-optimal) behavior policy. Importantly, we do

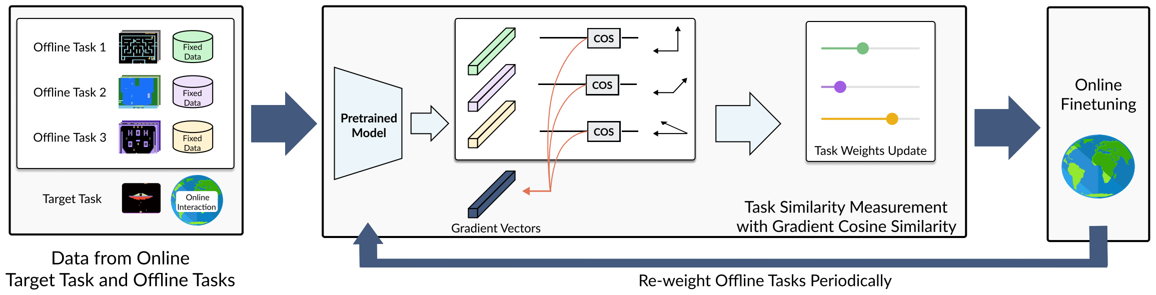


Figure 5.2: Illustration of our **Concurrent Cross-Task Learning** strategy, where we selectively include a subset of the available pretraining tasks while finetuning on a target task.

not make any assumptions about the quality or the source of trajectories in the dataset, *i.e.*, we do not assume datasets to consist of expert trajectories. We first train individual EfficientZero *teacher* models on each dataset for a fixed number of iterations in a single-task (offline) RL setting, resulting in m *teacher* models $\{\tilde{\pi}^i_\psi \mid 1 \leq i \leq m\}$. After training, we store the model predictions, $(\hat{p}, \hat{v})$, from each *teacher* model $\tilde{\pi}^i_\psi$ together with environment reward u as the student's quantity targets (π, z, u) respectively for a given game $\tilde{\mathcal{M}}^i$ (see Appendix 5.7.1 for the definition of each quantity). Next, we learn a *multi-task student* model π_θ by distilling the task-specific teachers into a single model via these quantity targets. Specifically, we optimize the student policy by sampling data uniformly from all pretraining tasks, and use value/policy targets generated by their respective teacher models rather than bootstrapping from student predictions as commonly done in the (single-task) MuZero Reanalyze algorithm. This step can be seen as learning multiple tasks simultaneously with direct supervision by distilling predictions from multiple teachers into a single model. Empirically, we find this to be a key component in scaling up the number of pretraining tasks. Although teacher models may not be optimal depending on the provided offline datasets, we find that they provide stable (due to fixed parameters during distillation) targets of sufficiently good quality. The simpler alternative – training a multi-task model on all m pretraining tasks simultaneously using RL is found to not scale beyond a couple of tasks in practice, as we will demonstrate our experiments in Appendix 5.7.3. After distilling a multi-task student model, we now have a single set of pretrained parameters that can be used for finetuning to a variety of tasks via online interaction, which we introduce in the following section.

5.3.2 Online Finetuning on a Target Task

In this stage, we iteratively interact with a target task (environment) to collect interaction data, and finetune the pretrained model on data from the target task. However, we empirically observe that directly finetuning a pretrained model often leads to poor performance on the target task due to *catastrophical forgetting*. Specifically, the initial sub-optimal data collected from the target task can cause a large perturbation in the original pretrained model parameters, ultimately erasing inductive priors learned during pretraining. To overcome this challenge, we propose a concurrent cross-task learning strategy: we retain offline data from the pretraining stage, and concurrently finetune the model on both data from the target task, as well as data from the pretraining tasks. While this procedure addresses catastrophical forgetting, interference between the target task and certain pretraining tasks can be harmful for the sample-efficiency in online RL. As a solution, gradient contributions from offline tasks are periodically re-weighted in an unsupervised manner based on their similarity to the target task. Figure 5.2 shows the specific concurrent cross-task learning procedure for target task finetuning in our framework.

At each training step t, we jointly optimize the target online task $\mathcal{M}$ and m offline (auxiliary) tasks $\{\tilde{\mathcal{M}}^i \mid \tilde{\mathcal{M}}^i \neq \mathcal{M}, 1 \leq i \leq m\}$ that were used during the *offline multi-task pretraining* stage. Our online finetuning objective is defined as:

$$\mathcal{L}_t^{\text{adapt}}(\theta) = \mathcal{L}_t^{\text{ez}}(\mathcal{M}) + \sum_{i=1}^{m} \eta^i \mathcal{L}_t^{\text{ez}}(\tilde{\mathcal{M}}^i) \tag{5.1}$$

where $\mathcal{L}^{\text{ez}}$ is the ordinary (single-task) EfficientZero objective (see Appendix 5.7.1), and η^i are dynamically (and independently) updated task weights for each of the m pretraining tasks. The target task loss term maintains a constant task weight of 1. During online finetuning, we use distillation targets from teachers obtained from each pretraining game, and use MuZero Reanalyze to compute targets for the target task for which we have no teacher available.

In order to dynamically re-weight task weights η^i throughout the training process, we break down the total number of environment steps (*i.e.*, 100k in our experiments) into even

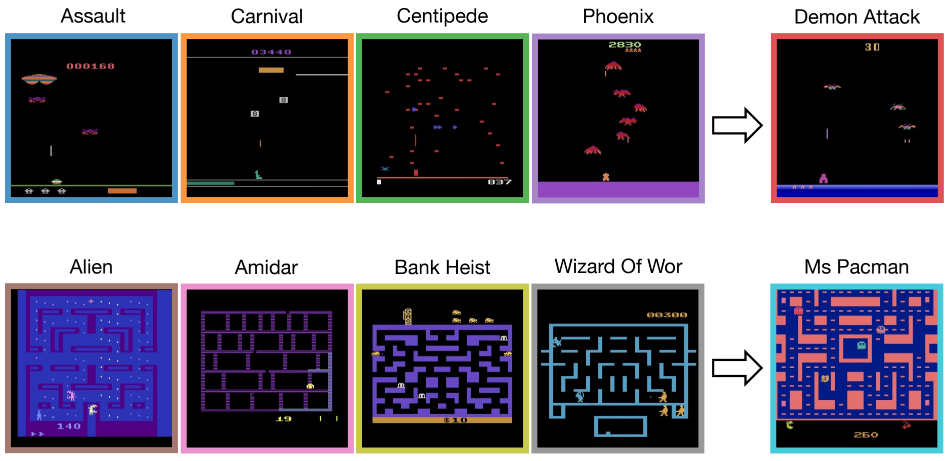
T-step cycles (intervals). Within each cycle, we spend first N-steps to compute an updated η^i corresponding to each offline task $\tilde{\mathcal{M}}^i$. The new η^i will then be fixed during the remaining $T-N$ steps in the current cycle and the first N steps in the next cycle. We dynamically assign the task weights by measuring the "relevance" between each offline task $\tilde{\mathcal{M}}^i$ and the (online) target task $\mathcal{M}$. Inspired by the conflicting gradients measurement for multi-task learning in [YKG$^+$20], we compute the cosine similarity between loss gradients $\tilde{\mathcal{G}}^i_n$ from $\mathcal{L}^{\text{ez}}_n(\tilde{\mathcal{M}}^i)$ and $\mathcal{G}_n$ from $\mathcal{L}^{\text{ez}}_n(\mathcal{M})$ given by

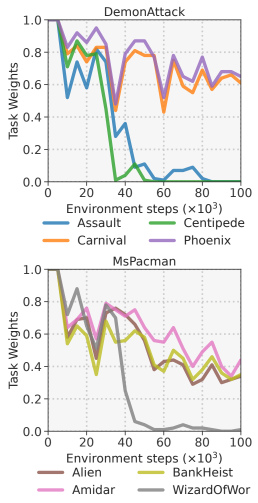
$$\text{Sim}(\tilde{\mathcal{M}}^i, \mathcal{M}) = \frac{\tilde{\mathcal{G}}^i_n \cdot \mathcal{G}_n}{\|\tilde{\mathcal{G}}^i_n\| \|\mathcal{G}_n\|}. \tag{5.2}$$

Within the N-step update, we maintain a task-specific counter s^i and the new task weights η^i can be reset by $\eta^i = \frac{s^i}{N}$ at the beginning of each every T-cycle. The procedure for obtaining s^i is described in Appendix 5.7.2. Concretely, $\text{Sim}(\tilde{\mathcal{M}}^i, \mathcal{M})$ measures the angle between two task gradients $\tilde{\mathcal{G}}^i_n$ and $\mathcal{G}_n$. Intuitively, we aim to (approximately) prevent gradient contributions from the offline tasks from conflicting with the gradient update direction for the target task by regulating offline tasks objectives with task weights η. While re-weighting task weights at every gradient update would result in the least amount of conflicting gradients, it is prohibitively costly to do so in practice. However, we empirically find the cosine similarity of task gradients to be strongly correlated in time, *i.e.*, the cosine similarity does not change much between consecutive gradient steps. By instead updating task weights every N steps, our proposed technique mitigates gradient conflicts at a negligible computational cost in contrast to the compute-intensive gradient modification method proposed in [YKG$^+$20]. Figure 5.3 shows adjustments to task weights during finetuning for each of two distinct sets of pretraining and target tasks.

5.4 Experiments

We evaluate our method and baselines on **14** tasks from the limited-interaction Atari100k benchmark [KBM$^+$20] where only 100k environment steps are permitted. We seek to answer:





(a) Cross-Task Transfer from 4 offline games (left) to 1 target game (right). (b) Task weights.

Figure 5.3: **Visualization of Concurrent Cross-Task Learning**. (*left*) the model adapts to the online target game while concurrently learns 4 offline games. (*right*) the figure shows the task weights of the 4 offline games that are periodically recomputed based on their gradient similarity to the target games (DemonAttack and MsPacman).

- How does our proposed framework compare to alternative pretraining and online RL approaches with *limited* online interaction from the target task?
- How do the individual components of our framework influence its success?
- When can we empirically expect finetuning to be successful?

Experimental setup. We base our architecture and backbone learning algorithm on EfficientZero [YLK+21] and focus our efforts on the pretraining and finetuning aspects of our problem setting. We consider EfficientZero with two different network sizes to better position our results: *(i)* the same network architecture as in the original EfficientZero implementation which we simply refer to as **EfficientZero**, and *(ii)* a larger variant with 4 times more parameters in the representation network (denoted **EfficientZero-L**). We use the EfficientZero-L variant as the default network for our framework through our experiments, unless stated otherwise. However, we find that our EfficientZero baseline generally does not benefit from a larger architecture, and we thus include both variants for a fair comparison. We experiment with cross-task transfer on three subsets of tasks: tasks that share *similar* game mechanics (for which we consider two

Table 5.1: **Atari100k benchmark results (*similar* pretraining tasks)**. Methods are evaluated at 100k environment steps. For each game, XTRA is first pretrained on all 4 other games from the same category. Our main result is highlighted. We also include three ablations that remove *(i)* cross-task optimization in finetuning (only online RL), *(ii)* the pretraining stage (random initialization), and *(iii)* task re-weighting (constant weights of 1). We also include zero-shot performance of our method for target tasks in comparison to behavioral cloning. Mean of 5 seeds and 32 evaluation episodes.

Category	Game	BC (finetuned)	Efficient Zero	Efficient Zero-L	XTRA (Ours)	Ablations (XTRA) w.o. cross-task	w.o. pretraining	w.o. task weights	Zero-Shot BC	XTRA (Ours)
Shooter	Assault	838.4	1027.1	1041.6	**1294.6**	1246.4	1257.5	1164.2	0.0	92.8
	Carnival	1952.4	3022.1	2784.3	**3860.9**	3544.4	2370.0	3071.6	93.75	719.3
	Centipede	1814.1	3322.7	2750.7	5681.4	3833.2	**6322.7**	5484.1	162.2	1206.8
	Demon Attack	825.5	11523.0	4691.0	14140.9	6381.5	9486.8	**51045.9**	73.8	113.6
	Phoenix	427.6	10954.9	3071.0	14579.8	10797.3	9010.6	**22873.9**	0.0	8073.4
	Mean Improvement	0.42	1.00	0.69	1.36	1.02	1.11	**2.06**	0.02	0.29
	Median Improvement	0.55	1.00	0.83	1.28	1.15	0.82	**1.65**	0.01	0.24
Maze	Alien	152.9	695.0	641.5	**954.8**	722.8	703.6	633.6	108.1	294.1
	Amidar	25.5	109.7	84.2	90.2	**121.8**	70.8	69.7	0.0	5.2
	Bank Heist	178.8	246.1	244.5	**304.9**	280.1	225.1	261.4	0.0	7.3
	Ms Pacman	550.0	1281.4	1172.8	**1459.7**	1011.1	1122.6	809.2	147.6	448.9
	Wizard Of Wor	163.8	1033.1	928.8	985.0	**1246.1**	654.4	263.5	100.0	9.4
	Mean Improvement	0.35	1.00	0.90	**1.11**	1.06	0.82	0.70	0.07	0.17
	Median Improvement	0.23	1.00	0.92	**1.14**	1.11	0.88	0.64	0.10	0.05
Overall	Mean Improvement	0.39	1.00	0.79	1.23	1.04	0.96	**1.38**	0.05	0.23
	Median Improvement	0.33	1.00	0.91	**1.25**	1.12	0.85	1.04	0.02	0.16

Shooter and ***Maze*** categories), and tasks that have no discernible properties in common (referred to as ***Diverse***). We measure performance on individual Atari games by absolute scores, and also provide aggregate results as measured by mean and median scores across games, normalized by human performance or EfficientZero performance at 100k environment steps. All of our results are averaged across 5 random seeds (see Appendix 5.7.4 for more details). We provide details on our pretraining dataset in Appendix 5.7.6.

Baselines. We compare our method against **7** prior methods for online RL that represent the state-of-the-art on the Atari100k benchmark (including EfficientZero), a multi-task behavior cloning policy pretrained on the same offline data as our method does for zero-shot performance on the target task and the performance after finetuning on the target task (see Appendix 5.7.14 for details), and a direct comparison to CURL [SLA20], a strong model-free RL baseline, under an offline pretraining + online finetuning setting. We also include a set of ablations that include

EfficientZero with several different model sizes and pretraining/finetuning schemes. The former baselines serve to position our results with respect to the state-of-the-art, and the latter baselines and ablations serve to shed light on the key ingredients for successful multi-task pretraining and finetuning.

5.4.1 Results & Discussion

We introduce our results in the context of each of our three questions, and discuss our main findings.

1. How does our proposed framework compare to alternative pretraining and online RL approaches with limited online interaction from the target task?

Tasks with *similar* game mechanics. We first investigate the feasibility of finetuning models that are pretrained on games with *similar* mechanics. We select 5 shooter games and 5 maze games for this experiment. Results for our method, baselines, and a set of ablations on the Atari100k benchmark are shown in Table 5.1. For completeness, we also provide learning curves in Figure 5.9. We find that pretraining improves sample-efficiency substantially across most tasks, improving mean and median performance of EfficientZero by **23**% and **25**%, respectively, overall. Interestingly, XTRA also had a notable zero-shot ability compared to a multi-game Behavioral Cloning baseline that is trained on the same offline dataset. We also consider three ablations: *(1)* **XTRA without cross-task:** a variant of our method that naively finetunes the pretrained model without any additional offline data from pretraining tasks during finetuning, *(2)* **XTRA without pretraining:** a variant that uses our concurrent cross-task learning (*i.e., leverages offline data during finetuning*) but is initialized with random parameters (no pretraining), and finally *(3)* **XTRA without task weights:** a variant that uses constant weights of 1 for all task loss terms during finetuning. We find that XTRA achieves extremely high performance on 2 games (DemonAttack and Phoenix) without dynamic task weights, improving over EfficientZero by as much as **343**% on DemonAttack. However, its median performance is overall low compared to our

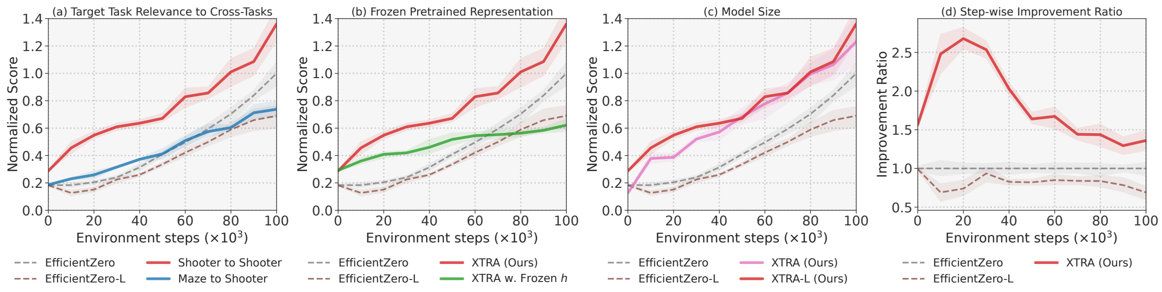


Figure 5.4: ***(a)* Effectiveness of Task Relavance, *(b)* Frozen Representation, *(c)* Model Size, and *(d)* Environment Steps**. We visualize model performance on aggregated scores (5 seeds) from 5 shooter games.

default variant that uses dynamic weights. We conjecture that this is because some (combinations of) games are more susceptible to gradient conflicts than others.

Tasks with *diverse* game mechanics. We now consider a more diverse set of pretraining and target games that have no discernible properties in common. Specifically, we use the following tasks for pretraining: Carnival, Centipede, Phoenix, Pooyan, Riverraid, VideoPinball, WizardOfWor, and YarsRevenge, and evaluate our method on 5 tasks from Atari100k. Results are shown in Table 5.2. We find that XTRA advances the state-of-the-art in a majority of tasks on the Atari100k benchmark, and achieve a mean human-normalized score of **187**% vs. **129**% for the previous SOTA, EfficientZero. We perform the same set of the ablations as we do for tasks with similar game mechanics with XTRA, and the results are shown in Table 5.6 from Appendix 5.7.8. Additionally, we include an ablation that examines the effect of the number of pretrained tasks on later finetuning performance. Details and results for this ablation are shown in Table 5.7 from Appendix 5.7.9.

Model-free comparisons. For both settings (e.g., tasks with similar & diverse game mechanics), we also compare our framework with a strong model-free baseline, CURL [SLA20], where CURL is pretrained on the same pretraining tasks as XTRA is, and later finetuned to each of the target tasks. We find that pretraining does not improve the performance of this model-free baseline as consistently as for our model-based framework, XTRA, under both settings. More details and results on this comparison can be found in Table 5.4 and 5.5 from Appendix 5.7.7.

2. How do the individual components of our framework influence its success?

Table 5.2: **Atari100k benchmark results (diverse pretraining tasks)**. XTRA results use the *same* set of pretrained model parameters obtained by offline pretraining on 8 diverse games. Mean of 5 seeds each with 32 evaluation episodes. Our result is highlighted. All other results are adopted from EfficientZero [YLK+21]. We also report human-normalized mean and median scores.

Game	XTRA (Ours)	EfficientZero	Random	Human	SimPLe	OTRainbow	DrQ	SPR	MuZero	CURL
Assault	**1742.2**	1263.1	222.4	742.0	527.2	351.9	452.4	571.0	500.1	600.6
BattleZone	14631.3	13871.2	2360.0	37187.5	5184.4	4060.6	12954.0	**16651.0**	7687.5	14870.0
Hero	**10631.8**	9315.9	1027.0	30826.4	2656.6	6458.8	3736.3	7019.2	3095.0	6279.3
Krull	**7735.8**	5663.3	1598.0	2665.5	4539.9	3277.9	4018.1	3688.9	4890.8	4229.6
Seaquest	749.5	1100.2	68.4	42054.7	683.3	286.9	301.2	583.1	208.0	384.5
Normed Mean	**1.87**	1.29	0.00	1.00	0.70	0.41	0.62	0.65	0.77	0.75
Normed Median	0.35	0.33	0.00	1.00	0.08	0.18	0.30	**0.41**	0.15	0.36

A deeper look at task relevance. While our previous experiments established that XTRA benefits from pretraining even when games are markedly different, we now seek to better quantify the importance of task relevance. We compare the online finetuning performance of XTRA in two different settings: *(1)* pretraining on 4 *shooter* games and finetuning to 5 new *shooter* games, and *(2)* pretraining on 4 *maze* games and finetuning to the same 5 *shooter* games. Aggregate results across all 5 target tasks are shown in Figure 5.4 *(a)*. Unsurprisingly, we observe that offline pretraining and concurrently learning from other shooter games significantly benefit the online target shooter games through training, with particularly large gains early in training. On the contrary, pretraining on maze games and finetuning to shooter games show similar performance compared to EfficientZero trained from scratch. This result indicates that *(1)* selecting pretraining tasks that are relevant to the target task is key to benefit from pretraining, and *(2)* in the extreme case where there are *no* pretraining tasks relevant to the target task, finetuning with XTRA generally does not harm the online RL performance since it can automatically assign small weights to the pretraining tasks.

Which components transfer in model-based RL? Next, we investigate which model component(s) are important to the success of cross-task transfer. We answer this question by only transferring a subset of the different model components – representation h, dynamics function g, and prediction head f – to the online finetuning stage and simply using a random initialization

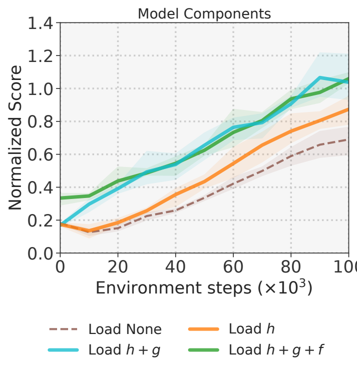


Figure 5.5: **Effectiveness of model components**. The aggregated scores from 5 shooter games by loading parameters of different pretrained model components. Mean of 5 seeds; shaded area indicates 95% CIs.

for the remaining components. Results are shown in Figure 5.5. Interestingly, we find that only transferring the pretrained representation h to the online RL stage only improves slightly over learning from scratch, especially in the early stages of online RL. In comparison, loading both the pretrained representation *and* dynamics function accounts for the majority of the gains in XTRA, whereas loading the prediction heads has no significant impact on sample-efficiency (but matters for zero-shot performance). We conjecture that this is because learning a good dynamics function is relatively more difficult from few samples than learning a *task-specific* visual representation, and that the prediction head accounts for only a small amount of the overall parameters in the model. Finally, we hypothesize that the visual representation learned during pretraining will be susceptible to distribution shifts as it is transferred to an unseen target task. To verify this hypothesis, we consider an additional experiment where we transfer all components to new tasks, but *freeze* the representation h during finetuning, *i.e.*, it remains fixed. Results for this experiment are shown in Figure 5.4 *(b)*. We find that, although this variant of our framework improves over training from scratch in the early stages of training, the frozen representation eventually hinders the model from converging to a good model, which is consistent with observations made in (supervised) imitation learning [PRPG22].

Scaling model size. In this experiment, we investigate whether XTRA benefits from larger model sizes. Since dynamics network g and prediction network f are used in MCTS search, increasing the parameter counts for these two networks would increase inference/training time complexity significantly. However, increasing the size of the representation network h has a relatively small impact on overall inference/training time. We compare the performance of our method and EfficientZero trained from scratch with each of our two model sizes, the original EfficientZero architecture and a larger variant (denoted EfficientZero-**L**); results are shown in Figure 5.4 *(c)*. We find that our default, larger variant of XTRA (denoted XTRA-L in the figure) is slightly better than the smaller model size. In comparison, EfficientZero-L, performs significantly worse than the smaller variant of EfficientZero.

Relative improvement vs. environment steps. Finally, we visualize the average improvement over EfficientZero throughout training in Figure 5.4 *(d)*. Results show that XTRA is particularly useful in the early stages of training, *i.e.*, in an extremely limited data setting. We therefore envision that cross-task pretraining could benefit many real-world applications of RL, where environment interactions are typically constrained due to physical constraints.

3. When can we empirically expect finetuning to be successful?

Based on Table 5.1 and 5.2, we conclude that cross-task transfer with model-based RL is feasible. Further, Figure 5.4 *(a)* shows that our XTRA framework benefits from online finetuning when pretraining tasks are relevant, and both representation and dynamics networks contribute to its success (Figure 5.5).

5.5 Related Work

Pretrained representations are widely used to improve downstream performance in learning tasks with limited data or resources available, and have been adopted across a multitude of areas such as computer vision [GDDM14, DGE15, HFW^{+}20], natural language processing [DCLT19c, BMR^{+}20], and audio [vdOLV18]. By first learning a good representation on a large dataset, the representation can quickly be finetuned with, *e.g.*, supervised learning on a small labelled dataset to solve a given task [PY10]. For example, [HFW$^+$20] show that contrastive pretraining on a large, unlabelled dataset learns good features for ImageNet classification, and [BMR$^+$20] show that a generative model trained on large-scale natural language data can be used to solve unseen tasks given only a few examples. While this is a common paradigm for problems that can be cast as (self-)supervised learning problems, it has seen comparably little adoption in RL literature. This discrepancy is, in part, attributed to optimization challenges in RL [BHDAJ20, HSW21, XRDM22, WLRL22], as well as a lack of large-scale datasets that capture both the visual, temporal, and control-relevant (actions, rewards) properties of RL [HYZ$^+$22]. In this work, we show that – despite these challenges – modern model-based RL algorithms can still benefit substantially from pretraining on multi-task datasets, but require a more careful treatment during finetuning.

Sample-efficient RL. Improving the sample-efficiency of visual RL algorithms is a long-standing problem and has been approached from many – largely orthogonal – perspectives, including representation learning [KGI$^+$19, YZK$^+$19, SLA20, SAG$^+$21], data augmentation [LLS$^+$20, KYF21, HSW21], bootstrapping from demonstrations [ZZP$^+$20] or offline datasets [WLRL22, ZZG22, BAZ$^+$22], using pretrained visual representations for model-free RL [SK21, XRDM22, ZHC$^+$22], and model-based RL [HS18a, FL17, NPD$^+$18, HLF$^+$19, KBM$^+$20, SAH$^+$20a, HLNB21, YLK$^+$21, HWS22, SLJA22, HLS$^+$23]. We choose to focus our efforts on sample-efficiency from the perspective of pretraining in a model-based context, *i.e.*, jointly learning perception *and* dynamics. Several prior works consider these problems independently from each other: [XRDM22] shows that model-free policies can be trained with a frozen pretrained visual backbone, and [SLJA22] shows that learning a world model on top of features from a visual backbone pretrained with video prediction can improve model learning. Our work differs from prior work in that we show it is possible to pretrain *and* finetune both the representation *and* the dynamics using model-based RL.

Finetuning in RL. Gradient-based finetuning is a well-studied technique for adaptation in (predominantly model-free) RL, and has been used to adapt to either changes in visuals or dynamics [MRCA17, YCBI19, DSC$^+$16, JSS$^+$20, HJS$^+$21, BHDAJ20, WLRL22, ZHC$^+$22], or task specification [XF21, WYY$^+$22]. For example, [JSS$^+$20] shows that a model-free policy for robotic manipulation can adapt to changes in lighting and object shape by finetuning via rewards on a mixture of data from the new and old environment, and recover original performance in less than 800 trials. Similarly, [HJS$^+$21] shows that model-free policies can (to some extent) also adapt to small domain shifts by self-supervised finetuning within a single episode. Other works show that pretraining with offline RL on a dataset from a specific task improve sample-efficiency during online finetuning on the same task [ZZG22, WLRL22]. Finally, [LNY$^+$22] shows that offline multi-task RL pretraining via sequence modelling can improve offline finetuning on data from unseen tasks. Our approach is most similar to [JSS$^+$20] in that we finetune via rewards on a mixture of datasets. However, our problem setting is fundamentally different: we investigate whether *multi-task* pretraining can improve online RL on an *unseen* task across *multiple* axes of variation.

5.6 Conclusion

In this paper, we propose Model-Based **Cross**-Task **Tra**nsfer (**XTRA**), a framework for sample-efficient online RL with scalable pretraining and finetuning of learned world models using extra, auxiliary data from other tasks. We find that XTRA improves sample-efficiency substantially across most tasks, improving mean and median performance of EfficientZero by 23% and 25%, respectively, overall. As a feasibility study, we hope that our empirical analysis and findings on cross-task transfer with model-based RL will inspire further research in this direction.

5.7 Appendix

5.7.1 XTRA/EfficientZero Objectives

XTRA uses the same learning objective as EfficientZero during both offline pretraining and online finetuning, except the quantity targets are predicted by the teacher model during distillation and concurrent learning for pretrained games instead of Muzero Reanalysis procedure.

Here, we explain the objectives of EfficientZero [YLK$^+$21] and its predecessor MuZero [SAH$^+$20a]. To warrant the latent dynamics that can mirror the true states of the environment, MuZero is trained to predict three necessary quantities directly relevant for planning: (1) the policy target π obtained from visit count distribution of the MCTS (2) immediate reward u from environment (3) bootstrapped value target z where $z = \sum_{i=0}^{k-1} \gamma^i u^i + \gamma^k v_{t+k}$. On top of MuZero, EfficientZero adds a self-supervised consistency loss term, and predicts sum of environment rewards from next k steps, $\sum_{i=0}^{k-1} \gamma^i u^i$, instead of single-step reward. We refer reader to the original manuscripts for implementation details. We present the learning objective for EfficientZero at time step t with k unroll steps:

$$\mathcal{L}_t^{\text{ez}}(\theta) = \sum_{k=0}^{K} \underbrace{\|\mathcal{L}^r(u_{t+k}, \hat{r}_t^k)\|_2^2}_{\text{reward}} + \lambda_1 \underbrace{\|\mathcal{L}^p(\pi_{t+k}, \hat{p}_t^k)\|_2^2}_{\text{policy}} + \lambda_2 \underbrace{\|\mathcal{L}^v(z_{t+k}, \hat{v}_t^k)\|_2^2}_{\text{value}} + \lambda_3 \underbrace{\|\mathcal{L}^s(s_{t+1}, \hat{s}_{t+1})\|_2^2}_{\text{consistency}} + c||\theta|| \tag{5.3}$$

5.7.2 Task Weights Computation

Within the N-steps update, we maintain a task-specific counter s^i and update the counter by Δs_n^i at each step n as follows:

$$\Delta s_n^i = \begin{cases} 1, & \text{if} \quad \text{Sim}(\tilde{\mathcal{M}}^i, \mathcal{M}) > 0.1 \\ 0, & \text{otherwise} \end{cases}$$

$$s^i = s^i + \Delta s_n^i \tag{5.4}$$

At every N steps, the new task weights η^i are updated by $\eta^i = \frac{s^i}{N}$, and used in subsequent finetuning objectives according to Equation 5.1. In practice, we start task weight updates at 10k steps to ensure enough data from the online target task has been collected for a meaningful similarity measure. All task weights are initialized as 1 for the first 10k steps.

Figure 5.6 shows how task weights are adaptively adjusted by the model during 100k environment steps during online finetuning stage for 10 games reported in Figure 5.9. Figure 5.7 shows the adjustments of task weights for 5 games reported in Figure 5.10.

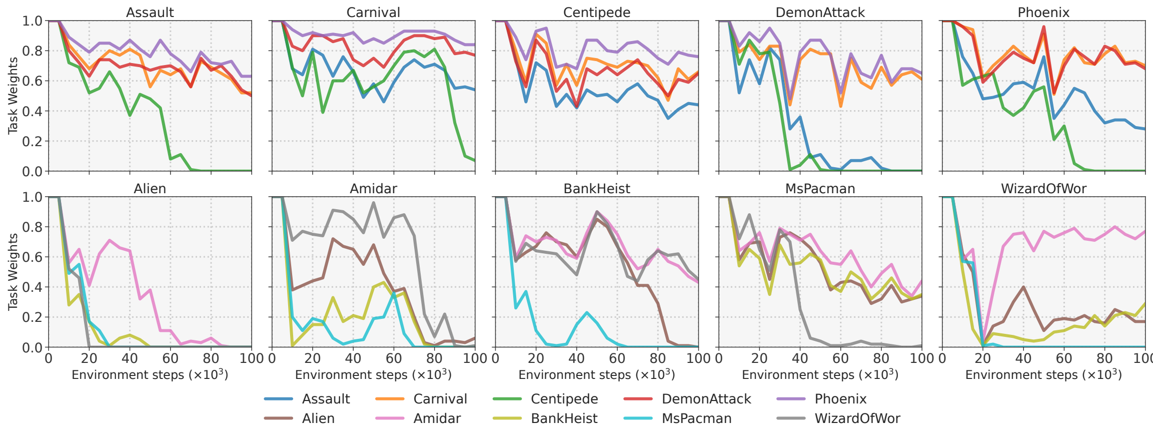


Figure 5.6: **Visualization of periodic task re-weighting with similar pretraining tasks**. We visualize task weights as a function of environment steps for each of our 10 tasks from Table 5.1. First row corresponds to *shooter* games and the bottom row corresponds to *maze* games. We evaluate task weights on all tasks from the same category except for the target task itself.

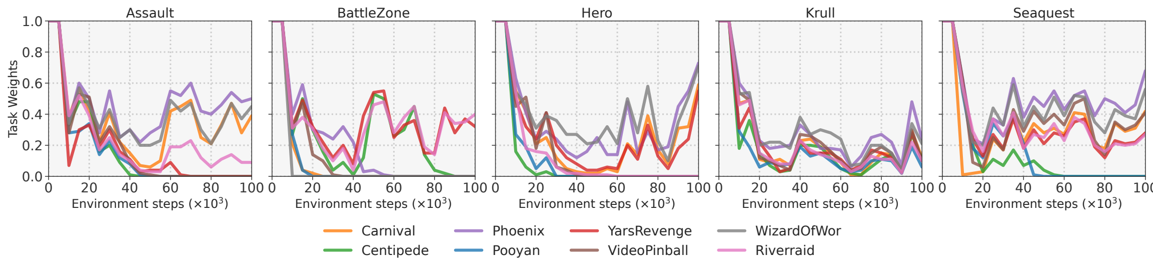


Figure 5.7: **Visualization of periodic task re-weighting with diverse pretraining tasks**. We visualize task weights as a function of environment steps for each of our 5 tasks from Table 5.2. We evaluate task weights on all 8 tasks used during pretraining.

5.7.3 Distillation vs. Multi-Game Offline RL

Our method learns a multi-game world model from offline data via distillation of task-specific world models trained with offline RL. An alternative way to obtain such a multi-game model would be to directly train a single world model on the multi-game dataset with offline RL. However, we find that learning such a model is difficult. A comparison between the two approaches on four different pretraining games is shown in Figure 5.8. We observe that multi-game offline RL (*green*) achieves low scores in all four pretraining games, whereas our wold model obtained by distillation (*orange*) performs comparably to single-task world models (*blue*) in 3 out of 4 tasks.

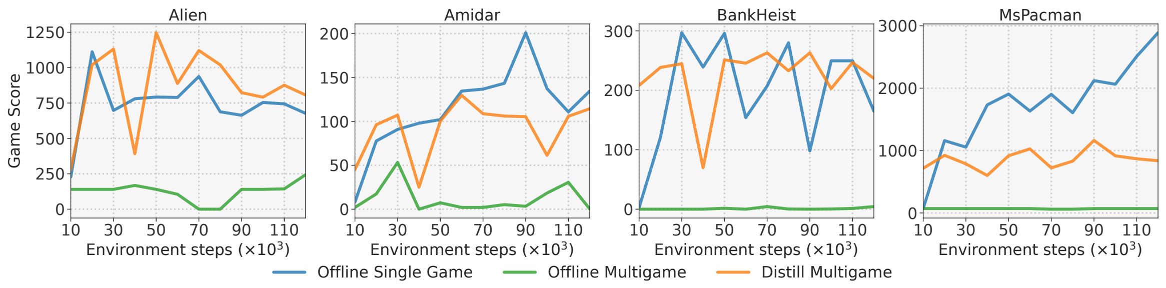


Figure 5.8: **Distillation vs. multi-game offline RL**. Results are shown on four pretraining tasks. Single-game models *(blue)* are trained via offline RL on each individual task, whereas results for multi-game offline RL (*green*) and our proposed distillation (*orange*) are obtained by evaluating a single set of pretraining parameters. Distillation nearly matches single-task performance.

5.7.4 Scores for Individual Seeds

Our results in Table 5.1 are aggregated across 5 seeds. In Table 5.3, we report game scores for each individual seed, as well as the mean, median, and standard deviation of game scores for each game. We also list random and human scores obtained from [BPK$^+$20], and calculate the Human Normalized Score based on the formula: $(\text{score}_{\text{agent}} - \text{score}_{\text{random}})/(\text{score}_{\text{human}} - \text{score}_{\text{random}})$ as in prior work. To the best of our knowledge, there are no human performance results for the Carnival game, and we therefore exclude this game from the aggregate Human

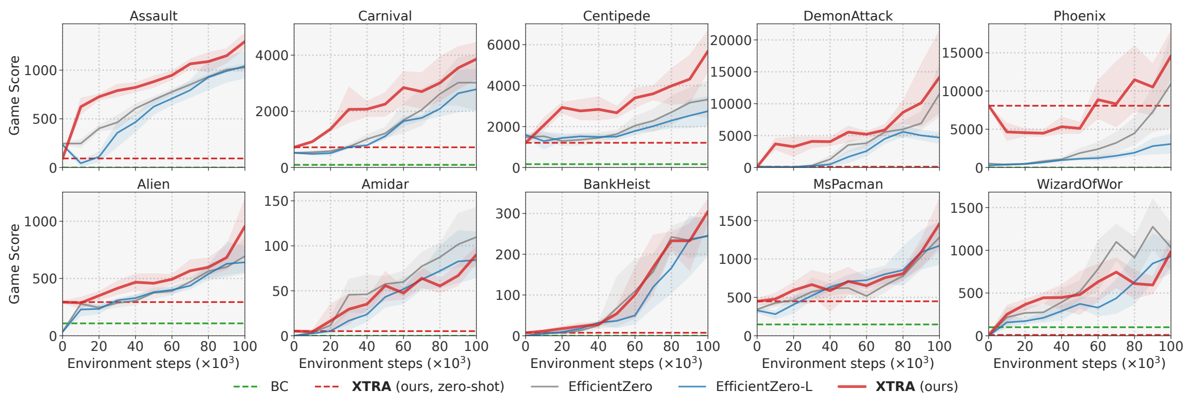


Figure 5.9: **Atari100k benchmark results (*similar* pretraining tasks)**. We report unnormalized scores, aggregated across 5 seeds per game. Shaded area indicates 95% confidence intervals.

Normalized Mean and Median Scores computed in Table 5.3.

Table 5.3: **Scores for individual seeds**. Per-seed game scores for our method, a random behavior baseline, and human performance. We report both unnormalized and normalized scores (Human Normalized Scores), as well as their aggregate results. Random and human scores are obtained from [BPK+20]. We evaluate each random seed on 32 evaluation episodes at 100k steps.

Game	Game Score per Seed					Aggregated Metrics			References		Human
	Seed 0	**Seed 1**	**Seed 2**	**Seed 3**	**Seed 4**	**Mean**	**Median**	**Std**	**Random**	**Human**	**Normed**
Assault	1450.19	1356.53	1236.19	1215.12	1214.72	1294.55	1236.19	105.06	222.40	742.00	2.06
Carnival	3865.31	4867.50	4155.62	2601.88	3814.38	3860.94	3865.31	819.67	-	-	
Centipede	7596.38	6179.25	5380.41	5300.03	3950.88	5681.39	5380.41	1336.58	2090.90	12017.00	0.36
DemonAttack	10470.78	8051.25	27574.06	8117.81	16490.47	14140.88	10470.78	8258.35	152.10	1971.00	7.69
Phoenix	20875.94	10988.44	10521.88	15803.44	14709.06	14579.75	14709.06	4198.81	761.40	7242.60	2.13
Alien	569.69	807.50	814.06	1388.12	1194.38	954.75	814.06	329.77	227.80	7127.70	0.11
Amidar	93.00	104.34	76.47	97.38	79.59	90.16	93.00	11.84	5.80	1719.50	0.05
BankHeist	303.12	316.56	270.62	270.00	364.06	304.88	303.12	38.83	14.20	753.10	0.39
MsPacman	1109.69	1960.00	1865.94	1228.44	1134.38	1459.69	1228.44	417.48	307.30	6951.60	0.17
WizardOfWor	1275.00	687.50	1056.25	990.62	915.62	985.00	990.62	213.62	563.50	4756.50	0.10
Human Normed Mean											1.45
Human Normed Median											0.36

5.7.5 Additional Evaluation curves of XTRA on Atari100k benchmark

For completeness, Figure 5.9 and 5.10 include evaluation curves of XTRA on the games for which we report final performance in Table 5.1 and 5.2.

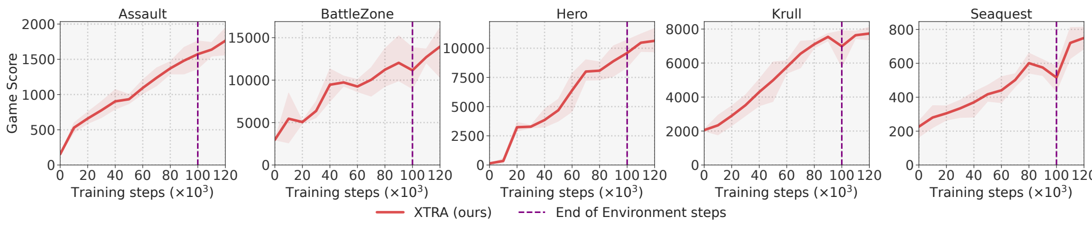


Figure 5.10: **Atari100k benchmark results (*diverse* pretraining tasks)**. We report unnormalized scores, aggregated across 5 seeds per game. Shaded area indicates 95% confidence intervals.

5.7.6 Offline Data Preparation

To train the model in offline multi-task pretraining stage, we use trajectories collected by EfficientZero [YLK+21] on the Atari100k benchmark. For each pretraining game, we assume we can access model checkpoints obtained every 10k steps from 120k training steps (the environment step is capped at 100k), resulting in 12 model checkpoints. For each checkpoint, we evaluate the model performance on the Atari environment following the same procedure from EfficientZero and collect 64 trajectories. This translates to an average of 1M transitions per game, but varies depending on episode length – for example, this only results in 636k transitions for the game of Assault. Since trajectories are collected from model checkpoints both at the early and late training stage within the 120k training steps, the collected data does not necessarily come from an expert agent. Thus, we show that XTRA is effective even when pretraining data is suboptimal, allowing us to learn from very diverse data sources.

5.7.7 Pretraining + Finetuning in Model-free RL

We compare the effectiveness of our framework, XTRA, with a strong model-free baseline, CURL, that we also implement following a similar pretraining and finetuning scheme. This is in contrast to the original formulation of CURL that does not leverage pretraining. We implement our training scheme for CURL with the following setup: *(1)* pretrain a multi-task CURL model on the same pretraining tasks as our framework uses (using offline data generated from training individual CURL agents), and *(2)* directly finetune the pretrained model on the target task with online RL for 100k environment steps.

Table 5.4: **Comparison to the model-free method with diverse pretraining tasks** The model is pretrained with Carnival, Centipede, Phoenix, Pooyan, Riverraid, VideoPinball, WizardOfWor, and YarsRevenge. Results for EfficientZero, CURL, Random, and Human are adopted from EfficientZero [YLK+21]. All other results are based on the average of 5 runs.

Game	**Model-Based**		**Model-Free**			
	Efficient Zero	**XTRA (Ours)**	**CURL**	**CURL$^{\text{ft}}$**	**Random**	**Human**
Assault	1263.1	**1742.2**	600.6	588.6	222.4	742.0
BattleZone	13871.2	14631.3	14870.0	**16450.0**	2360.0	37187.5
Hero	9315.9	**10631.8**	6279.3	6294.5	1027.0	30826.4
Krull	5663.3	**7735.8**	4229.6	3472.8	1598.0	2665.5
Seaquest	**1100.2**	749.5	384.5	385.5	68.4	42054.7
Normed Mean	1.29	**1.87**	0.75	0.60	0.00	1.00
Normed Median	0.33	**0.35**	0.36	0.40	0.00	1.00

5.7.8 XTRA Ablations for Tasks with Diverse Game Mechanics

We perform the same set of ablations for tasks that share *diverse* game mechanics as we do for tasks that share *similar* game mechanics for XTRA. Results are shown in Table 5.6. Details of each ablation can be found in Section 5.4.1.

5.7.9 Effects of Number of Tasks in Pretraining and Cross-Tasks in Finetuning

We perform an additional ablation for tasks that share *diverse* game mechanics – whether changing the number of tasks during pretraining would help or hurt the performance in later cross-task finetuning. By reducing the number of tasks in pretraining, the model is exposed to (1) less diverse game mechanics and (2) less offline training data in pretraining, and fewer cross-tasks in finetuning. In this ablation, we gradually reduce the number of pretrained tasks

Table 5.5: **Comparison to the model-free method with diverse pretraining tasks**. For each target game for finetuning, the model is first pretrained on all other 4 games from the same category. All results are based on the average of 5 runs.

Category	Game	Model-Based		Model-Free	
		Efficient Zero	XTRA (Ours)	CURL	CURL^{ft}
Shooter	Assault	1027.1	**1294.6**	590.2	461.2
	Carnival	3022.1	**3860.9**	591.6	714.8
	Centipede	3322.7	**5681.4**	4137.7	3731.0
	DemonAttack	11523.0	**14140.9**	908.3	638.9
	Phoenix	10954.9	**14579.8**	901.2	1168.4
	Mean Improvement	1.00	**1.36**	0.44	0.39
	Median Improvement	1.00	**1.28**	0.20	0.24
Maze	Alien	695.0	**954.8**	905.2	782.6
	Amidar	109.7	90.2	109.7	**169.9**
	BankHeist	246.1	**304.9**	151.8	86.2
	MsPacman	1281.4	**1459.7**	1421.6	1234.1
	WizardOfWor	1033.1	985	**1262.0**	1244.4
	Mean Improvement	1.00	**1.11**	1.05	1.04
	Median Improvement	1.00	**1.14**	1.11	1.13
Overall	Mean Improvement	1.00	**1.23**	0.74	0.72
	Median Improvement	1.00	**1.25**	0.81	0.71

from 8 (Carnival, Centipede, Phoenix, Pooyan, Riverraid, VideoPinball, WizardOfWor, and YarsRevenge), to 4 (Phoenix, WizardOfWor, VideoPinball, YarsRevenge), and to 2 (Phoenix, VideoPinball). XTRA is reduced to EfficientZero-L when the number of pretrained tasks and cross-tasks during finetuning is set to 0. We find that increasing the number of tasks during pretraining (and later cross-task finetuning) mostly consistently improves XTRA performance.

5.7.10 Architectural Details

We adopt the architecture of EfficientZero [YLK+21]. For EfficientZero-L and XTRA, we increase the number of residual blocks from 1 (default) to 4 (ours) in the representation network, which we find to improve pretraining slightly for XTRA. However, we find that our baseline

Table 5.6: **XTRA ablation with diverse pretraining tasks**. Results for EfficientZero, Random, and Human are adopted from EfficientZero [YLK+21]. All other results are based on the average of 5 runs.

Game			**Ablations (XTRA)**				
	Efficient Zero	**XTRA (Ours)**	**w.o. cross-task**	**w.o. pretraining**	**w.o. task weights**	**Random**	**Human**
Assault	1263.1	**1742.2**	1716.11	1183.58	1605.07	222.4	742.0
BattleZone	13871.2	**14631.3**	12918.8	8718.8	10087.5	2360.0	37187.5
Hero	9315.9	**10631.8**	8275.3	8672.9	7755.4	1027.0	30826.4
Krull	5663.3	**7735.8**	5910.7	6767.3	7104.7	1598.0	2665.5
Seaquest	1100.2	749.5	**811.4**	540.6	493.1	68.4	42054.7
Normed Mean	1.29	**1.87**	1.50	1.43	1.66	0.00	1.00
Normed Median	0.33	**0.35**	0.30	0.26	0.23	0.00	1.00

Table 5.7: **XTRA ablation with different number of tasks in pretraining**. Results for EfficientZero are adopted from EfficientZero [YLK+21]. All other results are based on the average of 5 runs.

Game	**XTRA**	**Ablations (XTRA)**			**EfficientZero**
	8 Games	**4 Games**	**2 Games**	**0 Games**	**0 Games**
Assault	**1742.2**	1676.7	1463.8	1255.9	1263.1
BattleZone	**14631.3**	9581.3	9550.0	10125.0	13871.2
Hero	**10631.8**	9654.9	8506.5	6815.1	9315.9
Krull	**7735.8**	7375.6	7348.9	5590.6	5663.3
Seaquest	749.5	656.4	627.5	770.8	**1100.2**
Normed Mean	**1.87**	1.74	1.65	1.23	1.29
Normed Median	**0.35**	0.29	0.25	0.22	0.33

EfficientZero (without pretraining) performs significantly worse with a larger representation network. Therefore, we use the default EfficientZero as the main point of comparison throughout this work and only include Efficient-L for completeness.

The architecture of the **representation networks** is as follows:

- 1 convolution with stride 2 and 32 output planes, output resolution 48x48. (BN + ReLU)
- 1 residual block with 32 planes.
- 1 residual downsample block with stride 2 and 64 output planes, output resolution 24x24.
- 1 residual block with 64 planes.

- Average pooling with stride 2, output resolution 12x12. (BN + ReLU)
- 1 residual block with 64 planes.
- Average pooling with stride 2, output resolution 6x6. (BN + ReLU)
- 1 residual block with 64 planes.

, where the kernel size is 3×3 for all operations.

The architecture of the **dynamics networks** is as follows:

- Concatenate the input states and input actions into 65 planes.
- 1 convolution with stride 2 and 64 output planes. (BN)
- A residual link: add up the output and the input states. (ReLU)
- 1 residual block with 64 planes.

The architecture of the **reward prediction network** is as follows:

- 1 1x1convolution and 16 output planes. (BN + ReLU)
- Flatten.
- LSTM with 512 hidden size. (BN + ReLU)
- 1 fully connected layers and 32 output dimensions. (BN + ReLU)
- 1 fully connected layers and 601 output dimensions.

The architecture of the **value and policy prediction networks** is as follows:

- 1 residual block with 64 planes.
- 1 1x1convolution and 16 output planes. (BN + ReLU)
- Flatten.
- 1 fully connected layers and 32 output dimensions. (BN + ReLU)
- 1 fully connected layers and D output dimensions.

where $D = 601$ in the value prediction network and $D = |\mathcal{A}|$ in the policy prediction network.

5.7.11 Hyper-parameters

We adopt our hyper-parameters from EfficientZero [YLK$^+$21] with minimal modification. Because XTRA uses data from offline tasks to perform cross-task transfer during online finetuning, we have an additional hyper-parameter for mini-batch size for offline tasks, which is set to 256 (default). We list all hyper-parameters in Table 5.8 for completeness. Lastly, we note that EfficientZero performs an additional 20k gradient steps at 100k environment steps, with a $10\times$ smaller learning rate. We follow this procedure when comparing to previous state-of-the-art methods (Table 5.2), but for simplicity we omit these additional gradient steps in the remainder of our experiments for both XTRA and baselines.

5.7.12 Effect of Mini-Batch Size

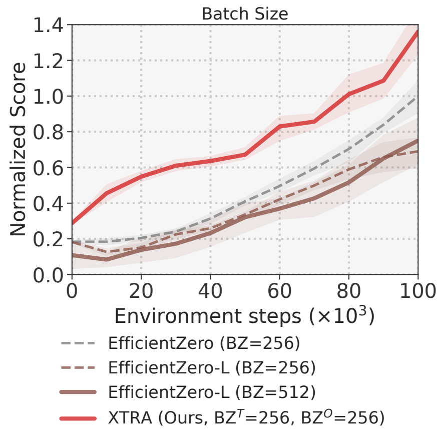


Figure 5.11: **Effect of mini-batch size**(BZ). For XTRA, we denote the batch size of the **T**arget task and **O**ffline tasks as BZ^T and BZ^O, respectively.

During online finetuning (stage 2), we finetune both with data from the target task, and data from the pretraining tasks. We maintain the same batch size (256) for the target task data as the non-pretraining baselines, but add additional data from the pretraining tasks with a 1:1 ratio. Thus, our effective batch size is $2\times$ that of the baselines. To verify that performance improvements stem from our pretraining (stage 1) and not the larger batch size, we compare our method to a variant of our EfficientZero-L baseline that uses a $2\times$ larger batch size (512). Results are shown in Figure 5.11. We do not observe any significant change in performance by doubling the batch size for the baseline. Thus, we conclude that a larger (effective) batch size is not the source of our performance gains, but rather our pretraining and inclusion of pretraining tasks during the online finetuning.

5.7.13 Game Information

In this section, we aim to provide additional context about the games that we consider during both pretraining and finetuning. Table 5.9 lists core properties for each game. The **Similar Task** column marks all games used in Table 5.1 (*similar* games), and the two **Diverse**

Task columns mark all games used in Table 5.2 (*diverse* games) for pretraining and finetuning, respectively. We further categorize games into five categories based on game mechanics: *Maze*, *Shooter*, *Tank*, *Adventure*, and *Ball Tracking*, and also report whether the scene is static or dynamic, as well the (valid) action space for each task. The maximum dimensionality of the action space is 18 for Atari games.

5.7.14 Behavioral Cloning Baseline

We use the representation + prediction network in XTRA for the behavioral cloning (BC) study. The BC (finetune) from Table 5.1 follows an offline pretraining + offline finetuning paradigm. The model is finetuned on offline data for the target task (also generated by the EfficientZero baseline). We find BC (finetune) underperforms the EfficientZero baseline. We also report zero-shot performance of the pretrained BC on their designated target tasks directly.

Acknowledgements

This chapter is based on the material "On the Feasibility of Cross-Task Transfer with Model-Based Reinforcement Learning" by Yifan Xu*, Nicklas Hansen*, Zirui Wang, Yung-Chieh Chan, Hao Su, Zhuowen Tu, which appears in International Conference on Learning Representations (ICLR) 2023. The dissertation author is the co-primary investigator and author of this material.

Table 5.8: **XTRA Hyper-parameters**. We list all relevant hyper-parameters below. Values are adopted from [YLK$^+$21] with minimal modification but included here for completeness.

Parameter	Setting
Observation down-sampling	96×96
Frames stacked	4
Frames skip	4
Reward clipping	True
Terminal on loss of life	True
Max frames per episode	108K
Discount factor	0.997^4
Minibatch size (offline tasks)	256
Minibatch size (target task)	256
Optimizer	SGD
Optimizer: learning rate	0.2
Optimizer: momentum	0.9
Optimizer: weight decay (c)	0.0001
Learning rate schedule	$0.2 \to 0.02$
Max gradient norm	5
Priority exponent (α)	0.6
Priority correction (β)	$0.4 \to 1$
Training steps	100K/120K
Evaluation episodes	32
Min replay size for sampling	2000
Self-play network updating inerval	100
Target network updating interval	200
Unroll steps (l_{unroll})	5
TD steps (k)	5
Policy loss coefficient (λ_1)	1
Value loss coefficient (λ_2)	0.25
Self-supervised consistency loss coefficient (λ_3)	2
LSTM horizontal length (ζ)	5
Dirichlet noise ratio (ξ)	0.3
Number of simulations in MCTS (N_{sim})	50
Reanalyzed policy ratio	1.0

Table 5.9: **Atari Game information**. We consider a variety of games in our experiments. Here, we provide more context to our selection of games. In our *similar* experiments (Table 5.1), we finetune model to each game after pretraining it on the other games within the same category. In *diverse* experiments (Table 5.2), we finetune a single model pretrained on all eight games to each of the target games.

Games	**Similar Task** **(Pretrain & Fine Tune)**	**Diverse Task** **(Pretrain)**	**Diverse Task** **(Fine Tune)**	**Category**	**Scene Continuity**	**Action Space**
Alien	✓			Maze		18
Amidar	✓			Maze	✓	10
Assault	✓		✓	Shooter	✓	7
Bank Heist	✓			Maze		18
Carnival	✓	✓		Shooter	✓	6
Centipede	✓	✓		Shooter	✓	18
DemonAttack	✓			Shooter	✓	6
MsPacman	✓			Maze		9
Phoenix	✓	✓		Shooter		8
WizardOfWor	✓	✓		Maze		10
BattleZone			✓	Tank	✓	18
Hero			✓	Adventure		18
Krull			✓	Adventure		18
Seaquest			✓	Shooter	✓	18
Pooyan		✓		Shooter		6
Riverraid		✓		Shooter	✓	18
VideoPinball		✓		Ball Tracking		9
YarsRevenge		✓		Shooter		18

Chapter 6

Discussion and Future Directions

In conclusion, we have described the state of attention research in computer vision and its connections to visual attention in neuroscience and cognitive science. We study the attention mechanism in visual representation learning in various computer vision tasks, including line segment detection, image recognition, object detection, semantic segmentation, and few-shot learning. Beyond computer vision, attention-based models offer a promising direction towards general artificial intelligence by creating a unified framework for studying cognitive functions such as visual processing [DBK$^+$21], language understanding [DCLT19a], speech [DXX18] memory [BKPS20], and decision making [CLR$^+$21].

Recently, research has shown that the success of attention-based models in computer vision and natural language processing can be extended to the RL decision-making domain [CLR$^+$21, JLL21]. Initial success using a vanilla transformer has been demonstrated in modeling trajectories of future states, actions, and rewards with less compounding prediction errors on Atari, OpenAI Gym, and Key-to-Door tasks [CLR$^+$21]. We expect planning with attention-based models to increase in the next few years, and the study on attention mechanisms discussed in Chapters 2, 3, and 4 will be of great value to this development.

The development of world models in AI has significantly contributed to connecting visual perception with higher-level cognitive processes and enhancing learning efficiency across various tasks [HS18b, HLBN19, SAH$^+$20b, KBM$^+$20, YLK$^+$21, HWS22]. Our work on the Model-Based Cross-Task Transfer (XTRA) framework, introduced in Chapter 5, exemplifies this progress. By leveraging learned internal world models, XTRA accelerates the learning of new and distinctly different tasks, paving the way for studying scalable systems capable of learning generalized world representations. In parallel, advancements in attention-based models have shown promise in developing more generalized and efficient AI systems. A key opportunity lies in the integration of attention mechanisms and world models within a unified framework, creating more versatile, adaptable, and efficient AI systems that can handle complex decision-making and planning tasks, much like the human brain.

Bibliography

[Abr87] Bruce D. Abramson. *The Expected-Outcome Model of Two-Player Games*. PhD thesis, Columbia University, 1987. AAI8827528.

[ADBB17] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[ADL$^+$22] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

[AEC13] Katharina Anton-Erxleben and Marisa Carrasco. Attentional enhancement of spatial resolution: linking behavioural and neurophysiological evidence. *Nature Reviews Neuroscience*, 14(3):188–200, 2013.

[BAZ$^+$22] Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. In *NeurIPS*, 2022.

[BBC$^+$19] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub W. Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *ArXiv*, abs/1912.06680, 2019.

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[Bel57] Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

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

[BHDAJ20] Cristian Bodnar, Karol Hausman, Gabriel Dulac-Arnold, and Rico Jonschkowski. A geometric perspective on self-supervised policy adaptation. *ArXiv*, 2020.

[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 HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2019.

[BKPS20] Mikhail S Burtsev, Yuri Kuratov, Anton Peganov, and Grigory V Sapunov. Memory transformer. *arXiv preprint arXiv:2006.11527*, 2020.

[BMR$^+$20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[BNVB13] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[BPK$^+$20] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *ICML*, 2020.

[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*, pages 679–698, 1986.

[Che12] Zhe Chen. Object-based attention: A tutorial review. *Attention, Perception, & Psychophysics*, 74(5):784–802, 2012.

[CHHS20] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *ICML*, 2020.

[CLD$^+$21] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *ICLR*, 2021.

[CLR$^+$21] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 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.

[CND$^+$22] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[Cou06] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[CT17] Le Chang and Doris Y Tsao. The code for facial identity in the primate brain. *Cell*, 169(6):1013–1028, 2017.

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

[CWL$^+$20] Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. A new meta-baseline for few-shot learning. *arXiv preprint arXiv:2003.04390*, 2020.

[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.

[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.

[DAP$^+$18] Rachit Dubey, Pulkit Agrawal, Deepak Pathak, Thomas L. Griffiths, and Alexei A. Efros. Investigating human priors for playing video games. In *ICML*, 2018.

[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.

[DCLT19a] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[DCLT19b] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[DCLT19c] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

[DD95] Robert Desimone and John Duncan. Neural mechanisms of selective visual attention. *Annual review of neuroscience*, 18(1):193–222, 1995.

[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 *ECCV*, 2008.

[DGE15] Carl Doersch, Abhinav Kumar Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1422–1430, 2015.

[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.

[DL02] Gustavo Deco and Tai Sing Lee. A unified model of spatial and object attention based on inter-cortical biased competition. *Neurocomputing*, 44:775–781, 2002.

[DSC$^+$16] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and P. Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *ArXiv*, abs/1611.02779, 2016.

[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):5–5, 2002.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 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.

[FL17] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793, 2017.

[FLT$^+$19] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, 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.

[FVE91] Daniel J Felleman and David C Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, NY: 1991)*, 1(1):1–47, 1991.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[GDDM14] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014.

[GL13] Charles D Gilbert and Wu Li. Top-down influences on visual processing. *Nature Reviews Neuroscience*, 14(5):350–363, 2013.

[GLY19] Weifeng Ge, Xiangru Lin, and Yizhou Yu. Weakly supervised complementary parts models for fine-grained image classification from the bottom up. In *CVPR*, 2019.

[GPAM$^+$14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, volume 27, 2014.

[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.

[HB05] Mary Hayhoe and Dana Ballard. Eye movements in natural behavior. *Trends in cognitive sciences*, 9(4):188–194, 2005.

[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 *NeurIPS*, pages 4005–4016, 2019.

[HCX$^+$21] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

[HFW$^+$20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[HG09] Benjamin Y Hayden and Jack L Gallant. Combined effects of spatial and feature-based attention on responses of v4 neurons. *Vision research*, 49(10):1182–1187, 2009.

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

[HJS$^+$21] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A. Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. In *ICLR*, 2021.

[HLBN19] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[HLF$^+$19] Danijar Hafner, Timothy P. Lillicrap, Ian S. Fischer, Ruben Villegas, David R Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *ArXiv*, abs/1811.04551, 2019.

[HLNB21] Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *ArXiv*, abs/2010.02193, 2021.

[HLS$^+$23] Nicklas Hansen, Yixin Lin, Hao Su, Xiaolong Wang, Vikash Kumar, and Aravind Rajeswaran. Modem: Accelerating visual model-based reinforcement learning with demonstrations. In *ICLR*, 2023.

[HMX$^+$20] Shell Xu Hu, Pablo G Moreno, Yang Xiao, Xi Shen, Guillaume Obozinski, Neil D Lawrence, and Andreas Damianou. Empirical bayes transductive meta-learning with synthetic gradients. *arXiv preprint arXiv:2004.12696*, 2020.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HS18a] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *NeurIPS*, volume 31, 2018.

[HS18b] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

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

[HSW21] Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. In *NeurIPS*, 2021.

[HWS22] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *ICML*, 2022.

[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.

[HYZ$^+$22] Nicklas Hansen, Zhecheng Yuan, Yanjie Ze, Tongzhou Mu, Aravind Rajeswaran, Hao Su, Huazhe Xu, and Xiaolong Wang. On pre-training for visuo-motor control: Revisiting a learning-from-scratch baseline, 2022.

[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.

[IK01] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3):194–203, 2001.

[JLL21] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *NeurIPS*, volume 34, 2021.

[JSS$^+$20] Ryan C. Julian, Benjamin Swanson, Gaurav S. Sukhatme, Sergey Levine, Chelsea Finn, and Karol Hausman. Efficient adaptation for end-to-end vision-based robotic manipulation. *ArXiv*, abs/2004.10190, 2020.

[KBM$^+$20] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, K. Czechowski, D. Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Ryan Sepassi, G. Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari. *ArXiv*, abs/1903.00374, 2020.

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

[KGI$^+$19] Tejas D. Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *ArXiv*, abs/1906.11883, 2019.

[KH$^+$09] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[KLC98] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

[KLM96] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[KSTH19] Adam Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E Hinton. Stacked capsule autoencoders. In *NeurIPS*, pages 15486–15496, 2019.

[KYF21] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *ArXiv*, abs/2004.13649, 2021.

[KZTL20] Aviral Kumar, Aurick Zhou, G. Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *ArXiv*, abs/2006.04779, 2020.

[Lan14] Michael F Land. Do we have an internal model of the outside world? *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1636):20130045, 2014.

[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.

[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. *arXiv preprint arXiv:2003.12060*, 2020.

[Lee03] Tai Sing Lee. Computations in the early visual cortex. *Journal of Physiology-Paris*, 97(2-3):121–139, 2003.

[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.

[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 *CVPR*, 2020.

[LHS20] Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. Self-supervised label augmentation via input transformations. In *ICML*, 2020.

[Lin20] Grace W Lindsay. Attention in psychology, neuroscience, and machine learning. *Frontiers in computational neuroscience*, page 29, 2020.

[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.

[LLL$^+$22] Chunyuan Li, Haotian Liu, Liunian Harold Li, Pengchuan Zhang, Jyoti Aneja, Jianwei Yang, Ping Jin, Yong Jae Lee, Houdong Hu, Zicheng Liu, et al. Elevater: A benchmark and toolkit for evaluating language-augmented visual models. *arXiv preprint arXiv:2204.08790*, 2022.

[LLP$^+$18] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018.

[LLS$^+$20] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *ArXiv*, abs/2004.14990, 2020.

[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.

[LMGH22] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. *arXiv preprint arXiv:2203.16527*, 2022.

[LMRS19] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, pages 10657–10665, 2019.

[LNY$^+$22] Kuang-Huei Lee, Ofir Nachum, Mengjiao Yang, L. Y. Lee, Daniel Freeman, Winnie Xu, Sergio Guadarrama, Ian S. Fischer, Eric Jang, Henryk Michalewski, and Igor Mordatch. Multi-game decision transformers. *ArXiv*, 2022.

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

[LPM15] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

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

[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., New York, NY*, 2(4.2), 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.

[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.

[MKS$^+$13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[MRCA17] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and P. Abbeel. Meta-learning with temporal convolutions. *ArXiv*, abs/1707.03141, 2017.

[MRCA18] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2018.

[MYMT17] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. *arXiv preprint arXiv:1712.09926*, 2017.

[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.

[NPD$^+$18] Ashvin Nair, Vitchyr H. Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *NeurIPS*, 2018.

[NRO$^+$15] Nora Nortmann, Sascha Rekauzke, Selim Onat, Peter König, and Dirk Jancke. Primary visual cortex represents the difference between past and present. *Cerebral Cortex*, 25(6):1427–1440, 2015.

[OLR18] Boris N Oreshkin, Alexandre Lacoste, and Pau Rodriguez. Tadam: Task dependent adaptive metric for improved few-shot learning. *arXiv preprint arXiv:1805.10123*, 2018.

[PBS16] Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *ICLR (Poster)*, 2016.

[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.

[PRPG22] Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Kumar Gupta. The unsurprising effectiveness of pre-trained vision models for control. In *ICML*, 2022.

[PY10] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359, 2010.

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

[QRK$^+$05] R Quian Quiroga, Leila Reddy, Gabriel Kreiman, Christof Koch, and Itzhak Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–1107, 2005.

[RB99] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

[RCG$^+$16] Andrei A Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *ICLR*, 2016.

[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.

[RH09] John H Reynolds and David J Heeger. The normalization model of attention. *Neuron*, 61(2):168–185, 2009.

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

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[RKH$^+$21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763, 2021.

[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. *OpenAI*, 2018.

[RP95] Andrew F Rossi and Michael A Paradiso. Feature-specific effects of selective visual attention. *Vision research*, 35(5):621–634, 1995.

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

[RRDU87] Giacomo Rizzolatti, Lucia Riggio, Isabella Dascola, and Carlo Umiltá. Reorienting attention across the horizontal and vertical meridians: evidence in favor of a premotor theory of attention. *Neuropsychologia*, 25(1):31–40, 1987.

[RZP$^+$22a] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[RZP$^+$22b] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley D. Edwards, Nicolas Manfred Otto Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *ArXiv*, abs/2205.06175, 2022.

[SAG$^+$21] Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron C. Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *ICLR*, 2021.

[SAH$^+$20a] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, L. Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 2020.

[SAH$^+$20b] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[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.

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

[SFH17] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *NeurIPS*, 2017.

[SHM$^+$16a] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[SHM$^+$16b] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016.

[SHM$^+$21] Julian Schrittwieser, Thomas K Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and offline reinforcement learning by planning with a learned model. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *NeurIPS*, 2021.

[SK21] Rutav Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. *ArXiv*, abs/2107.03380, 2021.

[SLA20] A. Srinivas, Michael Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *ICML*, 2020.

[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.

[SLJA22] Younggyo Seo, Kimin Lee, Stephen James, and P. Abbeel. Reinforcement learning with action-free pre-training from videos. In *ICML*, 2022.

[SR15] Marcel Simon and Erik Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *ICCV*, 2015.

[SSZ17] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 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.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NeurIPS*, 2014.

[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 *CVPR*, 2018.

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

[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.

[TBG05] Benjamin W Tatler, Roland J Baddeley, and Iain D Gilchrist. Visual correlates of fixation selection: Effects of scale and time. *Vision research*, 45(5):643–659, 2005.

[TCD+20] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[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. *arXiv preprint arXiv:2002.04764*, 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 *ICCV*, 2019.

[UN96] Marius Usher and Ernst Niebur. Modeling the temporal dynamics of it neurons in visual search: A mechanism for top-down selective attention. *Journal of cognitive neuroscience*, 8(4):311–327, 1996.

[VBL+16] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NeurIPS*, pages 3630–3638, 2016.

[vdOLV18] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

[vGJMR10] R G von Gioi, J Jakubowicz, J M Morel, and G Randall. LSD: A Fast Line Segment Detector with a False Detection Control. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(4):722–732, 2010.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[vZD05] Wieske van Zoest and Mieke Donk. The effects of salience on saccadic target selection. *Visual Cognition*, 12(2):353–375, 2005.

[WGGH18] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 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.

[WLRL22] Che Wang, Xufang Luo, Keith W. Ross, and Dongsheng Li. Vrl3: A data-driven framework for visual deep reinforcement learning. *ArXiv*, abs/2202.10324, 2022.

[WSC$^+$16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[WSC$^+$20] J Wang, K Sun, T Cheng, B Jiang, C Deng, Y Zhao, D Liu, Y Mu, M Tan, X Wang, et al. Deep high-resolution representation learning for visual recognition. *TPAMI*, 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.

[WYY$^+$22] Homer Walke, Jonathan Yang, Albert Yu, Aviral Kumar, Jedrzej Orbik, Avi Singh, and Sergey Levine. Don't start from scratch: Leveraging prior data to automate robotic reinforcement learning. *ArXiv*, abs/2207.04703, 2022.

[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.

[XF21] Annie Xie and Chelsea Finn. Lifelong robotic reinforcement learning by retaining experiences. *ArXiv*, abs/2109.09180, 2021.

[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.

[XRDM22] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *ArXiv*, abs/2203.06173, 2022.

[XROOP19] Chen Xing, Negar Rostamzadeh, Boris Oreshkin, and Pedro O O Pinheiro. Adaptive cross-modal few-shot learning. In *NeurIPS*, 2019.

[XT15] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *ICCV*, 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 *CVPR*, pages 2788–2797, 2020.

[YCBI19] Lin Yen-Chen, Maria Bauza, and Phillip Isola. Experience-embedded visual foresight. In *Conference on Robot Learning*, 2019.

[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.

[YKG$^+$20] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *NeurIPS*, volume 33, pages 5824–5836, 2020.

[YLK$^+$21] Weirui Ye, Shaohuai Liu, Thanard Kurutach, P. Abbeel, and Yang Gao. Mastering atari games with limited data. In *NeurIPS*, 2021.

[YSM19] Sung Whan Yoon, Jun Seo, and Jaekyun Moon. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. *arXiv preprint arXiv:1905.06549*, 2019.

[YZK$^+$19] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv*, 2019.

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

[ZD11] Huihui Zhou and Robert Desimone. Feature-based attention in the frontal eye field and area v4 during visual search. *Neuron*, 70(6):1205–1217, 2011.

[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 *ICML*, 2019.

[ZHC$^+$22] Yanjie Ze, Nicklas Hansen, Yinbo Chen, Mohit Jain, and Xiaolong Wang. Visual reinforcement learning with self-supervised 3d representations. *arXiv preprint arXiv:2210.07241*, 2022.

[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. *arXiv preprint arXiv:1605.07146*, 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.

[ZQM19] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *ICCV*, pages 962–971, 2019.

[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.

[ZZG22] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *ICML*, 2022.

[ZZK$^+$20] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 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 *ICCV*, pages 1685–1694, 2019.

[ZZP$^+$20] Albert Zhan, Philip Zhao, Lerrel Pinto, P. Abbeel, and Michael Laskin. A framework for efficient robotic manipulation. *ArXiv*, abs/2012.07975, 2020.

[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 *ICCV*, 2017.