Data and Label Efficient Representation Learning

by

Colorado James Reed

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Kurt Keutzer, Chair
Professor Trevor Darrell
Assistant Professor Manuela Girotto

Spring 2023

Data and Label Efficient Representation Learning

Abstract

Data and Label Efficient Representation Learning

by

Colorado James Reed

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Kurt Keutzer, Chair

Recent advances in unsupervised representation learning have led to a host of widely used AI tools, such as ChatGPT and Stable Diffusion. These tools have been the result of applying relatively simple training algorithms to massive models on massive GPU clusters, even larger amounts of unlabeled training data, and by tuning the algorithms on a host of labeled evaluation tasks. In this dissertation, we present methodologies to address removing each of these components when training models for representation learning, i.e. limited compute, limited training data, and limited evaluation data. This dissertation contains four main chapters that focus on data and label-efficient representation learning.

Data efficient representation learning focuses on learning useful representations with less data (labeled or unlabeled), which as discussed throughout this dissertation, can be particularly important for applications with limited data availability. *Label efficient representation learning* focuses on learning useful representations with little or no human annotations for the training data. As will be discussed, this is important for applications where it is often difficult or impossible to obtain accurately labeled data, such as in privacy sensitive fields or for applications with highly ambiguous label definitions.

The four chapters in this dissertation that address these topics include: (1) SelfAugment: Automatic Augmentation Policies for Self-Supervised Learning, which explored how to develop augmentation policies with little/no labeled training data and small amount of unlabeled data for unsupervised learning pipelines. (2) Data Efficient Self-Supervised Representation Learning, which explored how to leverage a form of *hierarchical pretraining* for 80x more data efficient pretraining. (3) Region Similarity Representation Learning, which explored one of the first methods for learning region-level representation by performing contrastive learning at a region (patch-based) level and let to substantial improvements for downstream tasks such as object detection/segmentation when few labeled data were available. (4) Scale-MAE: A Scale-Aware Masked Autoencoder for Multiscale Geospatial Representation Learning, which explored methods for leveraging known scale information for geospatial representation learning.

For my parents, who taught me the importance of research.

Contents

List of Figures

List of Tables

Acknowledgments

Finishing my Ph.D. feels bittersweet: an accomplishment, sure, but also the end of an era in which I've been accompanied by some of the most humble, thoughtful, fun, and well, downright brilliant people. I'm excited to move forward, but I'll miss this time. The least I can do, then, is to capture a few thoughts and sincere gratitudes for the wonderful people around me.

First, Kurt: from your guidance, I've grown as a researcher, thinker, entrepreneur, and perhaps most importantly, you've taught me to embrace the strange combinations of inconsistencies that make me, me. Kurt is a student of Tibetan history and philosophy and Buddhism, who also served as a top executive at a multi-billion dollar tech company, and then became a research professor at one of the top CS institutions in the world all while being a successful investor as well. Phew - what a description! Not to mention, Kurt is likely the only Berkeley professor with a deep investigation of Kundalini yoga practices on his homepage. Taken together, it's a ridiculously unexpected combination, but one that Kurt embraces in such a reassuring and successful way that it has shown me that I, too, can embrace my own particularities. There's no rules to any of this.

While Kurt has shown me that I can travel any path on any mountain, Trevor has shown me that it's also possible to just move the damn mountain. Trevor encouraged me to pursue ambitious research direction, create big initiatives (like the Berkeley Climate Initiative), and form large collaborations and projects. Trevor's continual perspective of *just make it happen and figure out the details when you need to*, has been empowering, and I know it's this mentality that led to him building Berkeley AI Research (BAIR) as we know it, not to mention his long history of ground-breaking research. Thanks for showing me that I can move mountains, Trevor. I hope that I can continue to learn and grow from your mentorship: it's meant more to me than you know.

I came to Berkeley way back in 2013, and I did some "rotations" with a few different research advisers: Pieter Abbeel, Zach Pardos, and Maneesh Agrawala, all of whom where accomodating, kind, and wonderful advisers in different ways. At this time, I was hungry to focus on building tools that people could really use, and to my own fault as a Ph.D. student, I wasn't particularly interested in writing papers. Nevertheless, Maneesh was very supportive and encouraged me to build tools then: in fact, along with Amy Pavel, we built a really fascinating type of textbook-inspired video interface called "video digests" and we even published a paper on it. But then I met a fellow grad student Sean Arietta: the most talented engineer I've ever met, a brilliant mind, and because some people just seem to have it all, he was also frusturatingly charismatic.

Sean and I fed off of each other's hunger to build impactful tools, and through late night coding sessions in the lab, weekend coding sessions over beers, and an unexpected affinity for skeeball brought to us from a zany business partner and close friend named Joey Mucha, our ideas evolved into a blossoming startup. We even had some investor interest, and enchanted by the idea that we could build something impactful (and become insanely rich in the process), we went on leave from our Ph.D.'s. We spent five years running our company, *DotDashPay* which later evolved into *Fraction*, where we had some wonderful highs and devastating lows. Of course, this experience alone could (and should) fill the pages of a novel, but I'll keep it brief here. We had a lot of fun, we wrote a lot of code, we hired, we fired :(, we learned a lot, and I got to spend time with some absolutely wonderful people that even went on to start their own companies (nice job Avi Press!).

But all of that aside, we did not become insanely rich in dollar dollar bills, but we became wealthy in experiences - sort of a consolation prize as far as wealth is concerned, sure, but I wouldn't trade this experience for any other.

Following the start-up life, Kurt welcomed me back to Berkeley to finish my Ph.D. Thanks again for this Kurt. Also - thanks to Roger Grosse and Forrest Iandola for encouraging me to take a leap back into research and finish my Ph.D. Once landing back at Berkeley, several people helped me find my bearings again: Amir Gholami, Sicheng Zhao, Zhen Dong, David Chan, Daniel Seita, Sheng Shen, and Roshan Rao. Then, I had a bunch of fantastic research collaborations with these folks and others including: Sean Metzger, Tete Xiao, Aravind Srinivas, Xiaolong Wang, Bo Li, Shanghang Zhang, Ani Nrusimha, Amir Bar, Alberto Todeschini, Xin Wang, Vadim Kantorov, Roei Herzig, Gal Chechik, Anna Rohrbach, Amir Globerson, Suzanne Petryk, Konstantinos Kallidromitis, Ankur Mahesh, Ritwik Gupta, Medhini Narasimhan, Christopher Funk, Brian Clipp, Manuela Girotto, Salvatore Candido, Matt Uyttendaele, Devin Guillory, Xiangyu Yue, Sayna Ebrahimi, Peter Vajda, Bichen Wu, Marianne Cowherd, and many others. I sincerely enjoyed advising the following folks as well: Akash Gokul, Araav Patel, Kevin Miao, Raymond Mo, Richard Mao, Shashwath Senthil, Shufan Li, and Vivek Vijaykumar. Oh, and of course, there were people that I just plain enjoyed chatting with in classrooms and hallways, including: Giscard Biamby, Lisa Dunlap, Grace Luo, Seth Park, Norman Mu, Sanjay Subramanian, Rudy Corona, and Shizhan Zhu. Not to mention, my qualifying committee were very helpful in bringing me to the point that I can write this dissertation, thanks Kurt Keutzer, Trevor Darrell, Daniel Feldman, and Angjoo Kanazawa.

Another big chapter of my Ph.D. has been creating the Berkeley AI Research Climate Initiative (BCI) with Ritwik Gupta and Medhini Narasimhan. Daniel Feldman and Manuela Girotto were absolutely crucial in getting this idea off of the ground, as was Aditi Krishnapriyan, Trevor Darrell, Kurt Keutzer, Daniel Rothchild, and many others. Thank you!

Of course, none of this would have been remotely possible without my wonderful family, friends, and in-laws. In particular, the amount of trust and love from my wife Jackie was the only reason I was able to do this. I'm so excited for everything that we do, all of our adventures, and all of the magic still to come. My brother Rocky, my sister Sade, and my parents James and Jo Anne have always been and will always be my foundation, my family. I can't express my gratitude in words, but if I had to try, it would be something like "oh yeah, thanks." :P

Chapter 1

Introduction

Solving a problem simply means representing it so as to make the solution transparent.

-Herbert Simon

This dissertation centers around *representation learning*: a branch of machine learning that focuses on learning to extract useful signals, or *representations*, from data. Representation learning builds upon traditional signal processing methodology by training machine learning models to automatically extract useful representations of data. These representations are typically represented computationally as high-dimensional vectors, but the underlying concept is simply to usefully represent the key information within the data – see Fig. 1.1. Practically, representation learning methods learns to extract features that are relevant to a given task, such as classifying images or translating text.

While representation learning has been around for 100+ years [124], recent advances in deep learning have led to a renewed interest in representation learning, as deep neural networks have proven to be very effective at learning useful representations from raw data [8]. Before this period, human-driven feature engineering was the dominant form of practical representation learning, where human knowledge, ingenuity, and experimentation were used to extract useful signals from noisy data. The past decade of research in deep learning has largely focused on replacing and augmenting manual feature engineering with learned representations. This saves time compared to requiring human input and experimentation, but perhaps more importantly, this better aligns with how an autonomous AI agent would need to operate in order to understand the world around it. As eloquently written in [8]:

> *An AI must fundamentally understand the world around us, and we argue that this can only be achieved if it can learn to identify and disentangle the underlying explanatory factors hidden in the observed milieu of low-level sensory data.*

This dissertation focuses on data and label efficient self-supervised representation learning.

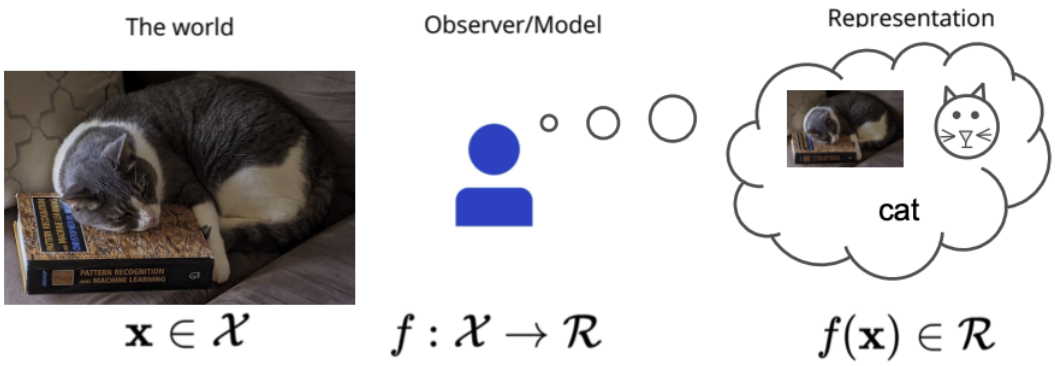

Figure 1.1: Representation learning extracts useful signals from observed data. In this case, we sample a state of the world, $x \in \mathcal{X}$ of a cat (the author's cat) sleeping on a classic machine learning textbook. An observer (e.g. a person, a sensor, or model) is viewed as a function, $f : \mathcal{X} \to \mathcal{R}$, that maps the world to some representation space, such as RGB values, a word, or a simplified representation (such as a vector graphic) of the cat. (Note: this figure was adapted from course materials from the CS159 course material at Caltech [url].)

Data efficient representation learning focuses on learning useful representations with less data, which as discussed throughout this dissertation, can be particularly important for applications with limited data availability, e.g. medical applications or certain scientific applications. *Label efficient representation learning* focuses on learning useful representations with little or no human annotations for the training data. As will be discussed, this is important for applications where it is often difficult or impossible to obtain accurately labeled data, such as in privacy sensitive fields (e.g. medical imaging [145]), applications with highly ambiguous label definitions (e.g. fashion or retail categorization [187]). *Self-supervised learning* is a type of unsupervised learning that creates target objectives without human annotation and has recently led to a dramatic increase in the ability to capture salient feature representations from unlabeled data across many domains [16, 73, 68].

The chapters in this dissertation are comprised of the research articles that define this body of work, where each chapter provides an introduction, motivation, summary of related work, methodology, experiments, and conclusions. The first two chapter focus on data efficient representation learning techniques. Chapter 2 presents the publication "SelfAugment: Automatic Augmentation Policies for Self-Supervised Learning" [137], published in CVPR 2021. This paper presented an AutoML approach to determine optimal augmentations to use for contrastive learning pipelines. Before this work, popular papers such as SimCLR [17] determined which augmentations to use in contrastive learning pipelines by exhaustively evaluating all possible combinations of augmentations using *labeled* evaluation data. This process, as we argued, leaks supervised information into the unsupervised pipeline and is not possible when working with a limited amount of (unlabeled) training data. Therefore, SelfAugment provides a direct method for determining high performing augmentation policies without requiring massive amounts of data, labels, or exhaustive

computational searches.

Chapter 3 presents the publication "Self-Supervised Pretraining Improves Self-Supervised Pretraining" [136], published in WACV 2022. Before this work, self-supervised pretraining methods had focused on pretraining from scratch using massive unlabeled datasets. This work showed that similar (or better) quality representations could be learned by simply finetuning a very small number of parameters in the network using a very small amount of unlabeled data and an appropriate self-supervised finetuning pipeline. Following this work, the unsupervised learning community adopted the term "Foundation Models," [9], which cited this work, adopted a similar unsupervised training/finetuning process as this work.

The next two chapters focus on label-efficient representation learning techniques. Chapter 4 presents the publication "Region Similarity Representation Learning" (ReSim) [168], published in ICCV 2022. Before ReSim, all self-supervised learning methods in computer vision had focused on developing methods for learning scene-level representations (which captured all information in an entire image). ReSim presented a method for learning region-level representation by performing contrastive learning at a region (patch-based) level. The region level representations could then be used for downstream region-level tasks such as object detection or segmentation, and as we showed, leads to several state-of-the-art results. Furthermore, this led to a particularly large increase in performance when there was a limited amount of labeled downstream data.

Chapter 5 presents the article "Scale-MAE: A Scale-Aware Masked Autoencoder for Multiscale Geospatial Representation Learning" [135], currently under review at CVPR 2023. Scale-MAE focused on representation learning for remote sensing imagery, which provides comprehensive views of the Earth, where different sensors collect complementary data at different spatial scales. Before Scale-MAE, large, pretrained models were commonly trained with imagery that is heavily augmented to mimic different conditions and scales, with the resulting models used for various tasks with imagery from a range of spatial scales. Such models overlooked scale-specific information in the data. Scale-MAE presented a pretraining method that explicitly learns relationships between data at different, known scales throughout the pretraining process. Scale-MAE pretrains a network by masking an input image at a known input scale, where the area of the Earth covered by the image determines the scale of a Vision Transformer (ViT) positional encoding, not the image resolution. We found that tasking the network with reconstructing both low/high frequency images led to robust multiscale representations for remote sensing imagery and led to several state-of-the-art results.

At a high level, this dissertation presents complementary and exploratory methods for data and label efficient representation learning. These methods have been adopted by several industry partners and collaborators throughout this work, formed a key part of our entry to DARPA's LWLL competition [33], and provided a foundation for other works to build upon (indeed, even the foundational Foundation Models work [9]!). Since the public release of these publications, several works have been published in the field that directly build upon this work, and each of the subsequent chapters contain a discussion and additional information on the impact that each work has had since publication.

Chapter 2

SelfAugment: Automatic Augmentation Policies for Self-Supervised Learning

Iin this chapter, we present several methods for both label and data efficient representation learning. In particular, we present a method called SelfAugment that allows a user to construct a self-supervised learning pipeline without requiring labeled data for a supervised evaluation in order to determine the settings of the pipeline, e.g. the hyperparameters and data augmentations.

2.1 Introduction

Self-supervised learning, a type of unsupervised learning that creates target objectives without human annotation, has led to a dramatic increase in the ability to capture salient feature representations from unlabeled visual data. So much so, that in an increasing number of cases these representations outperform representations learned from the same data with labels [16, 73, 68]. At the center of these advances is a form of *instance contrastive learning* where a single image is augmented using two separate data augmentations, and then a network is trained to distinguish which augmented images originated from the same image when contrasted with other randomly sampled augmented images, see [16, 68, 21, 167].

As illustrated in Figure 2.1, recent works [16, 21, 156] have used extensive *supervised* evaluations to determine which augmentation policies to use for training. The best policies obtain a *sweet spot*, where the augmentations make it difficult for the contrastive task to determine the corresponding image pairs while retaining salient image features; finding this sweet spot can be the difference between state-of-the-art performance or poor performance for various tasks [156].

However, it is often difficult or impossible to obtain accurately labeled data in privacy sensitive fields (e.g. medical imaging [145]), applications with highly ambiguous label definitions (e.g. fashion or retail categorization [187]), or not practical when one set of representations is used for a diverse set of downstream tasks (e.g. in autonomous driving systems [181]). This leads to the open question: *How can we evaluate self-supervised models, especially to efficiently select augmentation policies, when labeled data is not available?* We address this question via the following contributions:

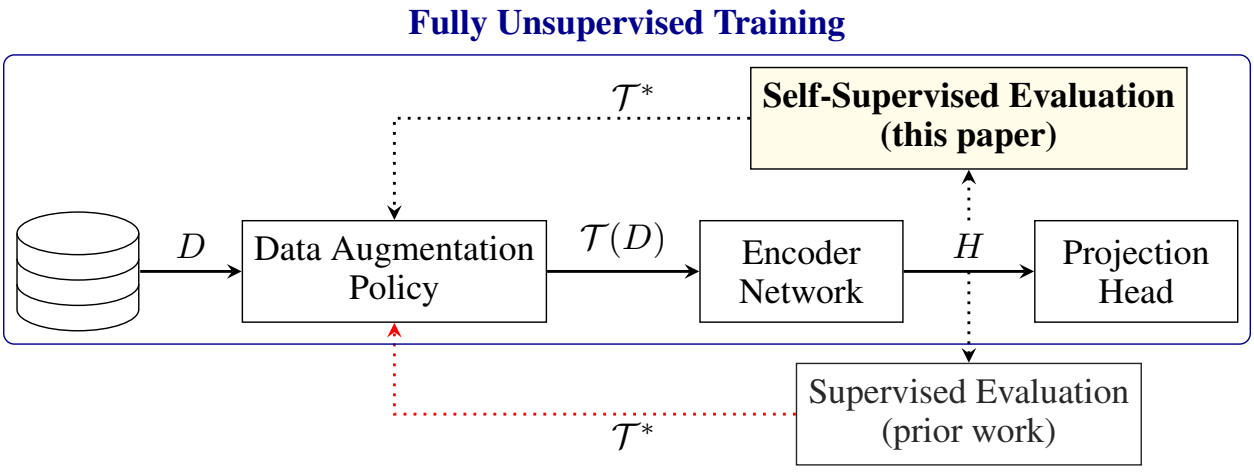

Figure 2.1: The blue box highlights our fully unsupervised training pipeline for instance contrastive representation learning: data D are augmented with policy $\mathcal{T}$, then encoded into representations, H, which are fed into a projection head yielding features that determine the InfoNCE loss (Eq. 2.1). As shown by the red arrow, prior work uses supervised evaluations of the representations to inform the training process, e.g. to update the augmentation policy $\mathcal{T} \rightarrow \mathcal{T}^*$. In this chapter, we show that self-supervised evaluation can be used in lieu of supervised evaluation and show how to use this evaluation for automatic and efficient augmentation selection.

- We show that a linear, image-rotation-prediction evaluation task is highly correlated with the downstream supervised performance (rank correlation $\rho > 0.94$) on six standard recognition datasets (CIFAR-10 [91], SVHN [58], ImageNet [142], PASCAL [44], COCO [101], Places-205 [189]) and tasks (image classification, object detection, and few-shot variants) across hundreds of learned representations, spanning three types of common evaluation techniques: linear separability performance, semi-supervised performance, and transfer learning performance.
- Using self-supervised evaluation, we adapt two automatic data augmentation algorithms for instance contrastive learning. Without using labeled evaluations, these algorithms discover augmentation policies that match or outperform policies obtained using supervised feedback and only use a fraction of the compute.
- We further show that using linear image rotation prediction to evaluate the representations works across network architectures, and that image rotation prediction has a stronger correlation with supervised performance than a jigsaw [121] or color prediction [186] evaluation task.

Based on these contributions and experiments, we conclude that image rotation prediction is a strong, unsupervised evaluation criteria for evaluating and selecting data augmentations for instance contrastive learning.

2.2 Background and Related Work

In this chapter, we study evaluations for self-supervised representations, particularly through the lens of learning data augmentation policies. We discuss these topics next.

Self-supervised representation learning: The general goal of representation learning is to pre-train a network and then either fine-tune it for a particular task or transfer it to a related model, e.g. see [73, 68, 136, 39, 43, 185, 57, 95, 132, 35]. Recently, [16] and [21] demonstrated substantial improvements by using similar forms of instance contrastive learning whereby one image is augmented using two separate data augmentations and then a network is trained to identify this positive pair contrasted with a large set of distractor images. A common loss function for contrastive methods is the InfoNCE loss, where given two images originating from the same image i and K_d distractor images we have:

$$\mathcal{L}_{NCE} = -\mathbb{E}\left[\log \frac{\exp(sim(\mathbf{z_{1,i}}, \mathbf{z_{2,i}})}{\sum_{j=1}^{K_d} \exp(sim(\mathbf{z_{1,i}}, \mathbf{z_{2,j}}))}\right] \tag{2.1}$$

where $\mathbf{z_{1,i}}, \mathbf{z_{2,i}}$ are the two different image representations from image i following the encoder network and projection head as shown in Figure 2.1, and $sim(\cdot,\cdot)$ is a similarity function such as a weighted dot product.

The InfoNCE loss [122, 73] has been shown to maximize a lower bound on the mutual information $I(\mathbf{h_1};\mathbf{h_2})$. The SimCLR framework [16] relies on large batch sizes to contrast the image pairs, while the MoCo framework [21] maintains a large queue of contrasting images. Given the increased adoption, broad application, and strong performance of instance contrastive learning [73, 16, 21], we focus our work on this type of self-supervised learning, specifically using the MoCo algorithm and training procedure for experimentation [21].

Self-supervised model evaluation is typically done via:

- *separability*: the network is frozen and the training data trains a supervised linear model (the justification is that good representations will reveal linear separability in the data) [27, 54, 37, 121, 53, 86]
- *transferability*: the network is either frozen or jointly fine-tuned, with a transfer task model that is fine-tuned using a different, labeled dataset (the justification is that good representations will generalize to a number of downstream tasks) [73, 68, 132, 35]
- *semi-supervised*, in which the network is either frozen or jointly fine-tuned using a fraction of labeled data with either the *separability* or *transferability* tasks mentioned above (the justification is that a small set of labeled data will benefit good representations) [73, 16].

While these evaluations characterize the unsupervised model in several ways, they have limited use for making training decisions because: **(i)** accessing labels as a part of the training process is not possible for unlabeled datasets, and **(ii)** evaluating the model on a different, labeled dataset requires an integrated understanding of the relationship between the transfer task, datasets, and models (see [181, 129, 139]). We seek a label-free, task-agnostic evaluation.

Learning data augmentation policies: Data augmentation has played a fundamental role in visual learning, and indeed, has a large body of research supporting its use [146]. In this work, we use a self-supervised evaluation to automatically learn an augmentation policy for instance contrastive models. To formulate our automatic data augmentation framework, we draw on several, equally competitive recent works in the supervised learning space [30, 74, 98, 31], where [30, 74, 98] use a separate search phase to determine the augmentation policy and [31] use a simplified augmentation space and sample from it via a grid search. For instance contrastive learning, we adapt a search-based automatic augmentation framework, Fast AutoAugment (FAA) [98], and a sampling-based approach, RandAugment [31]. We discuss these two algorithms in greater detail in the next section.

2.3 Self-Supervised Evaluation and Data Augmentation

Our central goals are to **(i)** establish a strong correlation between a self-supervised evaluation task and a supervised evaluation task commonly used to evaluate self-supervised models, and **(ii)** develop a practical algorithm for self-supervised data augmentation selection. The following subsections defines these goals in more detail.

Self-supervised evaluation

With labeled data, augmentation policy selection can directly optimize the supervised task performance [74, 98]. With unlabeled data, we seek an evaluation criteria that is highly correlated with the supervised performance without requiring labels. Inspired by [102], where the authors show that self-supervised tasks can be used to evaluate network architectures, we investigate the following self-supervised tasks to evaluate representations:

- **rotation** [54]: the input image undergoes one of four preset rotations $\{0^o, 90^o, 180^o, 270^o\}$, and the evaluation metric is the 4-way rotation prediction classification accuracy
- **jigsaw** [121]: the four quadrants of the input image are randomly shuffled into one of $4! = 24$ permutations, and the evaluation metric is the 24-way classification accuracy
- **colorization** [186]: the input is a grayscale image, and the evaluation metric is formulated as a pixel-wise classification on pre-defined color classes (313, from [186])

A key point to emphasize is that these self-supervised tasks are used to *evaluate* the representations learned from instance contrastive algorithms, e.g. MoCo. These self-supervised tasks were originally used to learn representations themselves, but in this work, we evaluate the representations using these tasks. In §2.4, we compute the correlation of each of these evaluations with a supervised, top-1 linear evaluation on a frozen backbone trained using a cross entropy loss on the training data.

Self supervised data augmentation policies

We study and adapt two approaches for augmentation policy selection from the supervised domain: a sampling-based strategy, RandAugment [30] and a search-based strategy, Fast AutoAugment (FAA) [98]. Using the notation from [98], let $\mathbb{O}$ represent the set of image transformations operations $\mathcal{O} : \mathcal{X} \to \mathcal{X}$ on input image $\mathcal{X}$. Following [98, 31], we define $\mathbb{O}$ as the set: {`cutout`, `autoContrast`, `equalize`, `rotate`, `solarize`, `color`, `posterize`, `contrast`, `brightness`, `sharpnes`, `shear-x`, `shear-y`, `translate-x`, `translate-y`, `invert`}.

Each transformation $\mathcal{O}$ has two parameters: **(i)** the magnitude λ that determines the strength of the transformation and **(ii)** the probability of applying the transformation, p. Let $\mathcal{S}$ represent the set of augmentation sub-policies, where a sub-policy $\tau \in \mathcal{S}$ is defined as the sequential application of N_τ consecutive transformation $\{\bar{\mathcal{O}}_n^{(\tau)}(x; p_n^{(\tau)}, \lambda_n^{(\tau)}) : n = 1, \ldots, N_\tau\}$ where each operation is applied to an input image sequentially with probability p. Applying sub-policy $\tau(x)$ is then a composition of transformation $\tilde{x}_{(n)} = \bar{\mathcal{O}}_n^{(\tau)}(\tilde{x}_{(n-1)})$ for $n = 1, \ldots, N_\tau$, where the full sub-policy application has shorthand $\tilde{x}_{(N_\tau)} = \tau(x)$ and the first application is $\tilde{x}_{(0)} = x$. The full policy, $\mathcal{T}$, is a collection of $N_\mathcal{T}$ sub-policies, and $\mathcal{T}(D)$ represents the set of images from D obtained by applying $\mathcal{T}$.

SelfRandAugment: RandAugment makes the following simplifying assumptions: **(i)** all transformations share a single, discrete magnitude, $\lambda \in [1, 30]$ **(ii)** all sub-policies apply the same number of transformations, N_τ **(iii)** all transformations are applied with uniform probability, $p = K_T^{-1}$ for the $K_T = |\mathbb{O}|$ transformations. RandAugment selects the best result from a grid search over (N_τ, λ). To adapt this algorithm for instance contrastive learning, we simply evaluate the searched (N_τ, λ) states using a self-supervised evaluation from §2.3 and refer to this as *SelfRandAugment*.

SelfAugment: We adapt the search-based FAA algorithm to the self-supervised setting; we call this adaptation *SelfAugment*. Formally, let $\mathcal{D}$ be a distribution on the data $\mathcal{X}$. For model $\mathcal{M}(\cdot|\theta) : \mathcal{X}$ with parameters θ, define the supervised loss as $\mathcal{L}(\theta|D)$ on model $\mathcal{M}(\cdot|\theta)$ with data $D \sim \mathcal{D}$. For any given pair of D_{train} and D_{valid}, FAA selects augmentation policies that approximately align the density of D_{train} with the density of the augmented $\mathcal{T}(D_{\text{valid}})$. This means that the transformations should help the model bolster meaningful features and become invariant to unimportant features after retraining with the augmented dataset. In practice, FAA splits D_{train} into $D_\mathcal{M}$ and $D_\mathcal{A}$, where $D_\mathcal{M}$ is used to train the model and $D_\mathcal{A}$ is used to determine the policy via:

$$\mathcal{T}^* = \underset{\mathcal{T}}{\operatorname{argmin}}\ \mathcal{L}(\theta_\mathcal{M}|\mathcal{T}(D_\mathcal{A})) \tag{2.2}$$

where $\theta_\mathcal{M}$ is trained using $D_\mathcal{M}$. This approximates minimizing the distance between the density of $D_\mathcal{M}$ and $\mathcal{T}(D_\mathcal{A})$ by using augmentations to improve predictions with shared model parameters, $\theta_\mathcal{M}$; see [98] for derivations. FAA obtains the final policy, $\mathcal{T}^*$, by exploring B candidate policies $\mathcal{B} = \{\mathcal{T}_1, \ldots, \mathcal{T}_B\}$ with a Bayesian optimization method that samples a sequence of sub-policies from $\mathcal{S}$ and adjusts the probabilities $\{p_1, \ldots, p_{N_\mathcal{T}}\}$ and magnitudes $\{\lambda_1, \ldots, \lambda_{N_\mathcal{T}}\}$ to minimize $\mathcal{L}(\theta|\cdot)$ on $\mathcal{T}(D_\mathcal{A})$ (see §2.5 for details). The top P policies from each data split are merged into $\mathcal{T}^*$. The network is then retrained using this policy on all training data, $\mathcal{T}^*(D_{\text{train}})$, to obtain the final network parameters θ^*. SelfAugment has three main differences from Fast AutoAugment that we discuss next.

Select the base policy: A *base augmentation policy* is required to perform the first pass of training to determine $\theta_{\mathcal{M}}$. SelfAugment determines this policy by training a MoCo network [21] for a short period of time on each of the individual transformations in $\mathbb{O}$ as well as `random-resize-crop` (the top performing single transformation in [16]). Each transformation is applied at every iteration, with $p = 1$ and a magnitude parameter λ stochastically set at each iteration to be within the ranges from [98]. Each network is trained until the loss curves separate – we found this to be around 10% of the total training epochs typically used for pre-training. The backbone is then frozen and a linear self-supervised evaluation task, ϕ_{ss}, is trained for each network and evaluated using held out training data. The base policy is then the transformation with the best evaluation.

Search augmentation policies: Given the base augmentations, we split the training data into k-folds. For each fold, we train a MoCo network, θ_{moco}, using the base augmentation, freeze the network, and train a self-supervised evaluation layer, ϕ_{ss}. We use the same Bayesian optimization search strategy as FAA to determine the policies. However, as the loss function $\mathcal{L}(\theta|D)$ cannot use supervised accuracy as in FAA, we explore four variants of a self-supervised loss function. § 2.5 discusses and compares each of these loss functions in more detail and §2.6 compares these loss functions with a supervised variant:

- **Min. eval error**: $\mathcal{T}^{SS} = \text{argmin}_{\mathcal{T}} \mathcal{L}_{\text{ss}}(\theta_{\mathcal{M}}, \phi_{\text{ss}}|\mathcal{T}(D_{\mathcal{A}}))$ where $\mathcal{L}_{\text{ss}}$ is the self-supervised evaluation loss, which yields policies that would directly result in improved performance on the evaluation task if the linear layer was retrained on top of the same base network. Minimizing the self-supervised error encourages augmentation policies that bolster distinguishable image features.

- **Min. InfoNCE**: $\mathcal{T}^{\text{I-min}} = \text{argmin}_{\mathcal{T}} \mathcal{L}_{\text{NCE}}(\theta_{\mathcal{M}}|\mathcal{T}(D_{\mathcal{A}}))$ where $\mathcal{L}_{\text{NCE}}$ is the InfoNCE loss from Eq. 2.1, which yields policies that make it easier to distinguish image pairs in the contrastive feature space. It is worth noting that a trivial way to distinguish image pairs is to use weak augmentations so paired images have high similarity.

- **Max InfoNCE**: $\mathcal{T}^{\text{I-max}} = \text{argmin}_{\mathcal{T}} -\mathcal{L}_{\text{NCE}}(\theta_{\mathcal{M}}|\mathcal{T}(D_{\mathcal{A}}))$ negates the previous loss function, yielding policies that make it difficult to distinguish image pairs in the feature space. Optimizing this loss function encourages a challenging augmentation policy, which may be overly challenging for training a network to learn meaningful representations.

- **Min $\mathcal{L}_{\text{ss}}$ max $\mathcal{L}_{\text{NCE}}$**: $\mathcal{T}^{\text{minmax}} = \text{argmin}_{\mathcal{T}} \mathcal{L}_{\text{ss}} - \mathcal{L}_{\text{NCE}}$ yields policies that simultaneously maximize InfoNCE, encouraging challenging augmentation policies, and minimize $\mathcal{L}_{\text{ss}}$, encouraging distinguishable image features.

Retrain MoCo using the full training dataset and augmentation policy: SelfAugment uses the selected policy from the loss functions and then retrains from scratch on the full dataset $\mathcal{D}_{train}$. It is worth noting that because the augmentation policy learned from SelfAugment is used for an instance contrastive task, and not for the evaluation task, the augmentations that minimize the self-supervised evaluation loss are not necessarily the best augmentations for instance contrastive pre-training. Rather, this method provides the set of augmentations that would lead to high

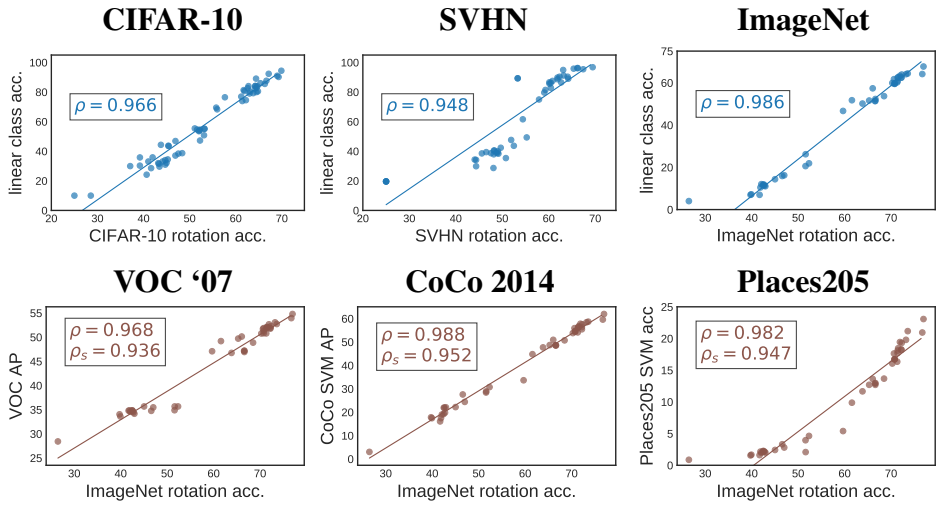

Figure 2.2: Top row: Correlation between supervised linear classification accuracy and linear image rotation prediction accuracy for three datasets. Bottom row: Correlation between rotation prediction and three transfer tasks from the ImageNet pretraining. ρ is the rank correlation between rotation prediction and the supervised task. ρ_s is the rank correlation between ImageNet supervised linear classification and the transfer task performance.

evaluation performance if the linear layer were directly retrained on top of the backbone used during augmentation selection [98]. Hence, incorporating the InfoNCE loss balances the instance contrastive task with the downstream task; we observe strong empirical evidence for this in §2.4 and §2.5.

2.4 Key Experiments

Through the following experiments, we aim to establish that **(i)** a self-supervised evaluation task is highly correlated with the supervised performance of standard visual recognition tasks (image classification, object detection, and few-shot variants) on common datasets (CIFAR-10 [91], SVHN [58], ImageNet [142], PASCAL [44], COCO [101], Places-205 [189]) and **(ii)** SelfAugment provides a competitive approach to augmentation selection, despite being fully unsupervised. All experiments used MoCo training [21] with the standard ResNet-50 backbone [148] on 4 GPUs, using default training parameters from [68], see § 2.5.

Self-supervised evaluation correlation

As evaluating all possible data augmentations for every dataset, training schedule, and downstream task is intractable, we evaluate a diverse sampling of RandAugment, SelfAugment, MoCoV2, and single-transform policies and training schedules. For CIFAR-10 [91], SVHN [58], and Im-

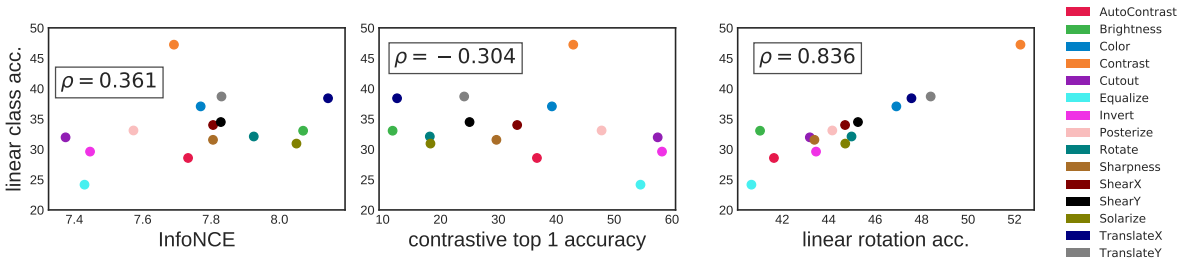

Figure 2.3: For SVHN, we plot the supervised classification accuracy (y-axis) vs the InfoNCE loss function (left), contrastive top-1 accuracy (middle), and self-supervised linear rotation accuracy (right), for a self-supervised model trained using one of each transformation used by SelfAugment. Neither of the left two training metrics are a consistent measure of the quality of the representations, while the rotation prediction accuracy provides a strong linear relationship.

ageNet [142] we use augmentation policies: **(i)** random horizontal flip and random resize crop, **(ii)** RandAugment on top of (i) grid searched over parameters $\lambda = \{4,5,7,9,11\}, N_\tau = \{1,2,3\}$ at $\{100,500\}$ epochs for CIFAR-10/SVHN and $\lambda = \{5,7,9,11,13\}$, $N_\tau = 2$ at $\{20,60,100\}$ epochs for ImageNet, **(iii)** each individual RandAugment transformations at 100 and 10 epochs for CIFAR-10 and ImageNet, respectively, **(iv)** scaling the magnitude λ from its min to max value for each N_τ at 500 epochs for CIFAR-10/SVHN and $\{20,60,100\}$ epochs for ImageNet, **(v)** for CIFAR-10/SVHN we also performed RandAugment using $K_T = \{3,6,9\}$ transformations at each of $\lambda = \{4,7\}$ with $N_\tau = 2$, at 500 epochs, **(vi)** MoCoV2 augmentations at 100, 200 epochs for ImageNet, **(vii)** the five SelfAugment policies at 750 epochs for CIFAR-10/SVHN and 100 epochs for ImageNet. In total, this yields a diverse set of 61 models for CIFAR-10/SVHN and 43 models for ImageNet.

Linear Evaluation: We first compare the rotation, jigsaw, and colorization evaluation tasks on the models obtained by MoCo pre-training with the above augmentation policies (full details in §2.5). We compute the Spearman rank correlation [147] between the top-1 supervised linear classification accuracy and each evaluation task, where a higher correlation indicates a better evaluation task. As shown below, the rotation prediction task has a uniformly higher correlation with the supervised linear classification compared to the jigsaw and colorization tasks:

Evaluation	CIFAR-10 (ρ)	SVHN (ρ)	ImageNet (ρ)
Rotation	**0.966**	**0.948**	**0.986**
Jigsaw	0.919	0.904	0.881
Colorization	0.612	0.806	0.627

The rotation evaluation correlations indicate a very strong relationship with the supervised evaluations, and based on its improvement over the jigsaw and colorization evaluations, we focus our main experiments, ablations, and SelfAugment/SelfRandAugment implementation of the automatic augmentation algorithms on this evaluation task. §2.5 and 2.5 contain further details and analyses between the self-supervised evaluation tasks, where for instance, we also observe that the network activations from the rotation layer are more similar to the activations from the supervised

classification layer compared to the other evaluations. We note that a rotation-based evaluation will not work for rotation invariant images (similarly, a color-based evaluation will not work for black-and-white images), and discuss this direction for future work in § 2.5.

The top row of Figure 2.2 shows scatterplots of the correlation between the supervised and self-supervised evaluations for the rotation evaluation task, where the poorer performing models come from single transformation augmentation policies and early evaluation schedules, and the better performing models come from the RandAugment, SelfAugment, and MoCoV2 augmentation policies. We observe that the rank correlation is maintained for both the poor and strong performing models, indicating that the rotation evaluation can be used across a wide range of model performance.

Transfer Learning: A central goal of representation learning is to learn transferable features. We therefore study the correlation between rotation evaluation and the ImageNet transfer performance for the following datasets/tasks:

- **VOC07** [44] **object detection**: Following the specifics from [68], we transfer the pre-trained models to a Faster R-CNN R50-C4 model and fine-tune all layers. Over three runs, we evaluate the mean results on VOC07 using the challenging COCO metric, $AP_{50:95}$, and report these results in Table 2.1. Further, we report the AP_{50}/AP_{75} in § 2.5. Fine-tuning is performed on the `train2007+2012` set and evaluation is on the `test2007` set.

- **COCO2014** [101] **multi-class image classification** Following [59], we train linear SVMs [10] on the frozen network and evaluate the accuracy over three end-to-end runs, denoted as COCO-C. **instance segmentation**, We use Mask-RCNN [66] with R50-FPN [100] as our base model and add new Batch Normalization layers before the FPN parameters. Unlike the classification, training is performed on `train2017` split with ∼118k images, and testing is performed on `val2017` split. We report the Average Precision on masks (AP^{mk}) for the standard 1x schedule, denoted as COCO-mk.

- **Places205** [189] **low shot scene classification:** Following [59], we train linear SVMs on the frozen network using $k = \{1,4,8,16,32,64\}$ labeled examples and evaluate the accuracy over five runs, with the average across all k used as the evaluation criteria, see § 2.5 for a breakdown.

For VOC07, COCO2014, and Places205, the bottom row of Figure 2.2 shows the ImageNet rotation performance vs the transfer task performance, yielding strong rank correlations of $\rho = \{0.968, 0.988, 0.982\}$, respectively. For comparison, the rank correlation of the supervised linear classification on ImageNet is $\rho_s = \{0.936, 0.952, 0.947\}$. For each transfer task, the rotation correlation is stronger than the supervised correlation. In [21], the authors observe that "linear classification accuracy is not monotonically related to transfer performance in detection," an observation that we find further evidence for across more transfer tasks. Furthermore, we observe that rotation prediction has a stronger transfer correlation than linear classification.

Finding the best individual image transformations Similar to the exhaustive evaluation of single-transform augmentation policies performed in [16], Figure 2.3 shows the performance of single-transform policies for SVHN (additional datasets in § 2.5). The left and middle plots show the supervised classification accuracy compared with the InfoNCE loss and top-1 contrastive accuracy

(how well the instance contrastive model predicts the augmented image pairs), while the right plot shows the rotation prediction accuracy for the image transformations in $\mathbb{O}$ evaluated after 100 training epochs. Using high or low values of InfoNCE or contrastive accuracy to select the best transformations would select a mixture of mediocre transformations, missing the top performing transformation in the middle. By using the rotation prediction, each transformation has a clear linear relationship with the supervised performance, enabling the unsupervised selection of the best transformations.

Selecting augmentation policies across architectures In [86], the authors showed that when training an entire network to classify image rotations, rather than just a linear layer, the rotation prediction performance did not correlate across architectures. We study this same problem but using only a linear evaluation layer. Specifically, for CIFAR-10 we study the rotation and supervised evaluation correlation for ResNet18, ResNet50, Wide-ResNet-50-2, using RandAugment with a grid search over the number of transformations $N_\tau = \{1,2,3\}$ and magnitudes $\lambda = \{4,5,7,9,11\}$ evaluated after 100 epochs. Figure 2.4 shows the results for each architecture: the overall Spearman rank correlation across all architectures is $\rho = 0.921$, between the Wide-ResNet-50-2 and ResNet18 Spearman correlations of 0.924 and 0.914 and less than the ResNet50 correlation of 0.957.

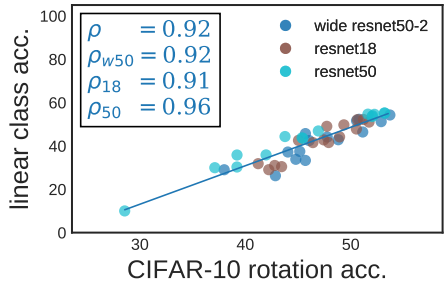

Figure 2.4: The Spearman rank correlation for CIFAR-10 RandAugment grid search for ResNet18, ResNet50, and Wide-ResNet-50-2, as well as the combined rank correlation.

Overall, these strong correlations indicate that a linear rotation prediction evaluation is effective across architectures. In § 2.5, we show that a drop in correlation occurs when using a two-layer MLP for rotation evaluation instead of a linear layer. Combined with the lack of correlation discovered when using a full network in [86], we surmise that using a *linear evaluation layer* to classify rotations is important as it disentangles the learned representations from the ability of the network to learn its own rotation features.

Performance benchmarks

Here, we evaluate the SelfAugment and SelfRandAugment augmentation policies with the goal of establishing comparable results to the state-of-the art policies obtained through supervised feedback.

Setup: We evaluate: **(i)** linear classification performance on top of the frozen network using CIFAR-10, SVHN, and ImageNet, **(ii)** transfer performance of ImageNet models on PASCAL

Table 2.1: Top-1 accuracy of SelfAug on CIFAR-10, SVHN, ImageNet, as well as ImageNet transfer tasks and semi-supervised experiments. An "R" superscript denotes best rotation accuracy. COCO-c denotes multi-label image classification on COCO2014 and COCO0-mk denotes mask-RCCN instance segmentation on COCO2017. Bold results are greater than one standard deviation better across multiple runs, see Appendix 2.5. Without using labels or hand-tuning hyperparameters, SelfAugment results in better performance than MoCoV2 for 2 out of 3 benchmark datasets it was directly trained on and comparable transfer performance.

	Self-Supervised			Transfer				Semi-Supervised	
unsup. feedback	C10	SVHN	IN	VOC	COCO-c	COCO-mk	Places	IN-1%	IN-10%
Base Aug	89.1	89.2	46.7	47.1	33.6	29.9	5.4	16.0	33.6
SelfRandAug	90.3	**96.8**^{R}	64.1	53.1	58.4	33.8	19.8	36.1	53.4
SelfAug (min rot)	91.0	94.9	57.4	50.2	50.9	31.9	13.6	26.4	45.1
SelfAug (min Info)	87.5	86.0	51.7	49.2	44.80	31.2	9.9	20.5	38.6
SelfAug (max Info)	90.1	96.2	63.3	52.6	57.8	33.8	19.4	34.4	52.7
SelfAug (minimax)	**92.6**^{R}	95.8	**64.4**	53.0	58.7	34.4	**21.2**	36.2	54.1
supervised feedback									
MoCoV2 [21]	92.3	96.4	64.2^R	**54.0**	**59.6**	34.5	20.8	**37.9**^{R}	**54.9**^{R}

VOC07 object detection, COCO2014 multi-class image classification, and Places205 few-show scene classification as described in the previous subsection, **(iii)** semi-supervised ImageNet top-1 accuracy with using only 1% or 10% of labels for linear training. All methods were evaluated with the same number of epochs and hyperparameters: 750 pre-train and 150 linear epochs for CIFAR-10/SVHN, 100 pre-train and 50 linear epochs for ImageNet, 24k fine-tuning iterations on VOC07, and the exact training parameters/schedules from [59] for COCO2014 and Places205. For SelfAugment, we used the settings from [98]: $P = 10$ policies, $T = 2$ transformations, and $K = 5$ folds of training. For SelfRandAugment, we used the grid search described in the previous subsection. See § 2.5 for all details.

Linear classification: Table 2.1 compares all versions of SelfAugment to the MoCoV2 augmentation policies [21] that were based on the extensive study of supervised policy evaluations in [16]. For linear classification with CIFAR-10, SVHN, and ImageNet, SelfAugment policies outperform MoCoV2's policy. Where the largest gain occurs in the SVHN dataset. SVHN, consisting of images of house numbers, is the most distinct from ImageNet's diverse, object centric images. Since MoCoV2's policy is the result of extensive, supervised study on ImageNet, it does not transfer as well to a distributionally distinct dataset such as SVHN. These results indicate that SelfAugment is a stronger approach to obtaining quality representations for datasets that are distributionally distinct from ImageNet.

Transfer and semi-supervised: For the VOC07 object detection and COCO2014 image classification transfer tasks, the MoCoV2 policy performed best. For COCO2017 instance segmentation, SelfAugment and MoCoV2 had similar transfer performance (0.1 mask AP difference), while SelfAugment had a stronger transfer performance for few-shot scene classification on the Places205 dataset at each k value (see § 2.5 for all details). Like ImageNet, VOC07 and COCO are natural images containing objects, while Places205 is a scene classification benchmark, consisting of complex scenes and diverse settings. SelfAugment's policies, as with the linear classification,

perform better for the dataset and task that substantially differs from ImageNet.

While the SelfAugment policy outperformed MoCoV2 for the linear classification with 100% of the labels used for training, MoCoV2 performed better when using only 1% and 10% of the labeled data for training the linear classifier. These results indicate that using supervised evaluation to select a policy can lead to strong semi-supervised performance on the same dataset, but as indicated by the Places205 results, this policy may not transfer to other semi-supervised tasks.

Rotation prediction: The mean rank correlation for the supervised linear classification and rotation prediction for all results in this subsection is $\rho = 0.978$, indicating that the strong correlation holds when removing the poorer performing models from the previous subsection. For every linear classification result except ImageNet, the top performing augmentation policy also corresponds to the top performing rotation prediction performance, as indicated with an "R" superscript. For ImageNet, MoCoV2's rotation prediction performance was significantly better than all other policies: $76.7 \pm 0.2\%$ compared to $73.6 \pm 0.2\%$ for SelfAugment's minimax, over three linear evaluations. While for a similar analysis, the linear classification performance for MoCoV2 performed worse: $64.2 \pm 0.1\%$ compared to $64.4 \pm 0.1\%$.

As shown in the previous subsection, however, the rotation prediction has a stronger correlation to transfer performance than the supervised linear classification, and indeed, that is the case here: on the VOC07 and COCO2014 transfer tasks and the ImageNet 1% and 10% semi-supervised evaluations, the MoCoV2 policy significantly outperformed all other policies, while the SelfAugment minimax policy had the best transfer performance for the Places205 task. In other words, *across the transfer and semi-supervised evaluations, rotation prediction was more indicative of performance than supervised linear classification.*

2.5 Additional Ablations

CIFAR-10, SVHN, ImageNet: Table 2.4 lists the training and experimental parameters for CIFAR-10, SVHN, and ImageNet. The training parameters were taken from [68] and adjusted for 4 GPUs, i.e. the learning rate and batch size were scaled by 0.5 since [68] experiments were conducted on 8 GPUs. Consult [68] and [21] for MoCo parameter information. For the linear classifier, we used 50 training iterations where the learning rate was $10\times$ reduced at 30 and 40 epochs for ImageNet and 20 and 30 epochs for CIFAR-10 and SVHN. In [68], the linear layer was trained over 150 iterations with a reduction at 80 and 100 iterations. In early experiments, we found the performance converged much earlier, and to reduce computational resources, we reduced all linear training iterations.

VOC07 Following [68], we transfer the ImageNet ResNet-50 weights to perform object detection using a Faster R-CNN R50-C4, with BN tuned. We fine-tuned all layers end-to-end. The image scale is [480, 800] pixels during training and 800 at inference. Training was on the VOC `trainval07+12` set and evaluation was on the `test2007` set. The R50-C4 backbones is similar to those available in Detectron2[1], where the backbone stops at the conv4 stage, and the box prediction head consists of the conv5 stage followed by a BN layer. Table 2.3 displays the $AP/AP_{50}/AP_{75}$ breakdown for three fine-tunings.

[1] https://github.com/facebookresearch/detectron2

Table 2.2: Transfer results: Places205 top-1 accuracy for frozen ResNet50 encoder after pre-training, with a linear SVM trained for scene classification, with $k = \{1, 4, 8, 16, 32, 64\}$ labeled training images for each class, averaged over five SVM trainings, where the errors indicate the standard deviation (see text for details).

Labeled samples	1	4	8	16	32	64
Base Aug	$1.35 \pm .03$	2.8 ± 0.05	$4.18 \pm .11$	$5.88 \pm .12$	$8.04 \pm .16$	$10.13 \pm .15$
SelfRandAug	$6.17 \pm .05$	$13.01 \pm .15$	$18.36 \pm .12$	$23.13 \pm .11$	$27.18 \pm .11$	$30.89 \pm .09$
SelfAug (min rot)	$3.58 \pm .06$	7.85 ± 0.09	$11.77 \pm .10$	$15.60 \pm .17$	$19.66 \pm .10$	$23.26 \pm .05$
SelfAug (min Info)	$2.46 \pm .04$	5.36 ± 0.07	$8.11 \pm .11$	$11.04 \pm .07$	$14.52 \pm .09$	$17.75 \pm .29$
SelfAug (max Info)	$5.87 \pm .06$	$12.73 \pm .14$	$17.88 \pm .16$	$22.64 \pm .21$	$26.95 \pm .12$	$30.56 \pm .17$
SelfAug (minimax)	$6.73 \pm .08$	$14.51 \pm .20$	$19.89 \pm .18$	$24.72 \pm .10$	$28.77 \pm .13$	$32.35 \pm .09$
MoCoV2	$6.65 \pm .11$	$14.06 \pm .13$	$19.59 \pm .14$	$24.57 \pm .13$	$28.56 \pm .08$	$32.24 \pm .10$

Table 2.3: Transfer Result: For `VOC07 test`, this table reports the average AP50 and COCO-style AP/AP_{75} over three runs of fine-tuning the ResNet50 encoder from ImageNet pre-training, where the errors indicate the standard deviation (see text for details).

Evaluation	AP	AP_{50}	AP_{75}
Base Aug	47.08 ± 0.17	74.80 ± 0.17	50.26 ± 0.24
SelfRandAug	53.06 ± 0.21	80.09 ± 0.14	57.58 ± 0.29
SelfAug (min rot)	50.13 ± 0.20	77.66 ± 0.18	53.94 ± 0.21
SelfAug (min Info)	49.18 ± 0.21	76.31 ± 0.17	53.17 ± 0.18
SelfAug (max Info)	52.66 ± 0.18	79.69 ± 0.15	57.33 ± 0.25
SelfAug (minimax)	52.72 ± 0.22	79.79 ± 0.21	57.62 ± 0.27
MoCoV2	53.94 ± 0.23	80.64 ± 0.18	59.34 ± 0.31

COCO2014/Places205: Following [59], we train Linear SVMs on frozen feature representations. We train a linear SVM per class for (80 for COCO2014, 205 for Places205) for the cost values $C \in 2^{[-19,-4]} \cup 10^{[-7,2]}$ We used 3-fold cross-validation to select the cost parameter per class and then further calculate the mean average precision. The features are first normalized in a (N, 9k) matrix, where N is number of samples in data and 9k is the resized feature dimension, to have norm=1 along each feature dimension. This normalization step is applied on evaluation data too. We use the following hyperparameter setting for training using `LinearSVC` sklearn class: `class_weight` ratio of 2:1 for positive:negative classes, `penalty`=l2, `loss`=squared_hinge, `tol`=1e-4, `dual`=True and `max_iter`=2000. Table 2.2 displays the Places205 results across five linears SVM trainings for all k values.

Table 2.4: Detailed training parameters for CIFAR-10, SVHN, and ImageNet experiments carried out in this chapter.

-MoCo Params	**CIFAR 10/SVHN**	**ImageNet**
Batch Size	512	128
moco-dim	128	128
moco-k	65536	65536
moco-m	0.999	0.99
moco-t	0.2	0.2
num-gpus	4	4
lr	0.4	0.015
schedule	120, 160	60, 80
momentum	0.9	0.9
weight decay	1e-4	1e-4
Classifier Params		
lr	15	30
batch size	256	256
momentum	0.9	0.9
weight decay	0.0	0.0
schedule	20, 30	30, 40
epochs	50	50

Correlation study

In this section, we detail the full experimental setup, investigation, and results from our study of the correlation between rotation prediction performance with a linear network and supervised downstream task performance. We also include additional preliminary and ablation studies.

Correlation study details We study RandAugment, SelfAugment, and MoCoV2 augmentation policies, and then evaluate the performance using self-supervised rotation prediction with a linear layer. For CIFAR-10 and SVHN, we evaluate:

- A *base augmentation* of random left-right horizontal flips with $p = 0.5$ of being applied and random resize and crop transformation with magnitude range $(0.2, 1.0)$, trained for 750 epochs.
- On top of the base augmentation[2], we performed a RandAugment grid search with magnitude $\lambda = \{4, 5, 7, 9, 11\}$ and number of transformations applied $N_\tau = \{1, 2, 3\}$, evaluated at $\{100, 500\}$ epochs. We initially included $N_\tau = 4$ in the grid search, but this always led to a

[2]Using this base augmentation systematically improved all RandAugment results over not using a base augmentation, both in terms of its rotation prediction performance and supervised linear classification performance.

degenerate solution, whereby the training loss would not minimize the objective function and the evaluation would yield a chance result (25% self-supervised rotation prediction and 10% supervised classification for CIFAR-10).

- Following [31], we also experimented with scaling the magnitude parameter λ from $[4, 11]$ linearly throughout the training. We did this for each of the three N_τ values and evaluated the results at 500 epochs.

- As part of the SelfAugment algorithm, we trained an augmentation policy consisting of each of the fifteen individual transformations in RandAugment (transformations listed in §2.3). In addition, we include the random resize crop transformation, as it was shown to be the best performing individual transformation in [16]. The magnitude for each transformation was stochastically selected from its magnitude range defined at each iteration. Each of these 16 transformations were evaluated after a short training cycle of 100 epochs. Each of these transformations were applied on top of a random left-right horizontal flip applied with $p = 0.5$.

- Using the rotation prediction results from each of the individual transformation policies, we selected the top $K_T = \{3, 6, 9\}$ augmentations, and trained RandAugment using only these transformations. We performed this training for $\lambda = \{4, 7\}$ with $N_\tau = 2$, evaluated at 500 epochs.

- We further include the four SelfAugment policies from each of its loss functions, evaluated at 750 epochs.

In total, this yields 61 different models for each of CIFAR-10/SVHN. For ImageNet we evaluate:

- A *base augmentation* of random left-right horizontal flips with $p = 0.5$ of being applied and random resize and crop transformation with magnitude range $(0.2, 1.0)$, trained for 100 epochs.

- On top of the base augmentation[3], we performed a RandAugment grid search with magnitude $\lambda = \{5, 7, 9, 11, 13\}$ and number of transformations applied $N_\tau = 2$, evaluated at $\{20, 60, 100\}$ epochs.

- Following [31], we also experimented with scaling the magnitude parameter λ from $[5, 13]$ linearly throughout the training. We did this for $N_\tau = 2$ and evaluated the results at 100 epochs.

- As part of the SelfAugment algorithm, we trained an augmentation policy consisting of each of the fifteen individual transformations in RandAugment. In addition, we include the random resize crop transformation, as it was shown to be the best performing individual

[3] Using this base augmentation systematically improved all RandAugment results over not using a base augmentation, both in terms of its rotation prediction performance and supervised linear classification performance.

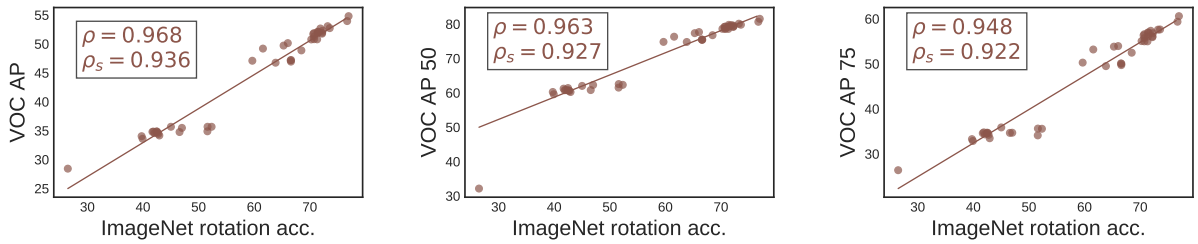

Figure 2.5: The ImageNet rotation prediction correlations with transfer performance to VOC07 for all AP, AP_{50}, and AP_{75} evaluations. The Spearman rank correlation with rotation prediction is indicated with ρ, while the rank correlation with the supervised ImageNet classification performance is indicated with ρ_s.

transformation in [16]. The magnitude for each transformation was stochastically selected from its magnitude range at each iteration. Each of these 16 transformations were evaluated after a short training cycle of 100 epochs. Each of these transformations were applied on top of a random left-right horizontal flip applied with $p = 0.5$.

- We further included the five SelfAugment policies from each of its loss functions, evaluated at 100 epochs.
- To further compare with state-of-the-art models, we compare with the MoCoV2 augmentation policy evaluated at $100, 200$ epochs

In total, this yields 43 different models for ImageNet.

Supplementing the main correlation results shown in the experiments section, Figure 2.5 shows the the ImageNet rotation prediction correlations with transfer performance to VOC07 for all AP, AP_{50}, and AP_{75} evaluations. Figure 2.6 shows the ImageNet rotation prediction correlations with transfer performance to Places205 few label scene classification using $k = \{1, 4, 8, 16, 32, 64\}$ labels per scene class. For both VOC07 and Places205, the Spearman rank correlation with rotation prediction is indicated with ρ, while the rank correlation with the supervised ImageNet classification performance is indicated with ρ_s. Across all evaluation metrics, the rotation correlation is higher than the supervised correlation.

Correlation with an MLP rotation prediction

When following the same evaluation protocol, except using a 2 layer MLP instead of a linear layer for rotation prediction, we find that the CIFAR-10 Spearman rank correlation with the supervised linear classification drops from 0.966 to 0.904 while the rotation prediction performance increases by $3.4 \pm 1.4\%$ across all evaluations. This correlation drop indicates that using a simple linear layer, rather than a more complicated network, is ideal for evaluation of the learned representations. A more complicated network can learn its own representation, which distances the evaluation from the learned representations.

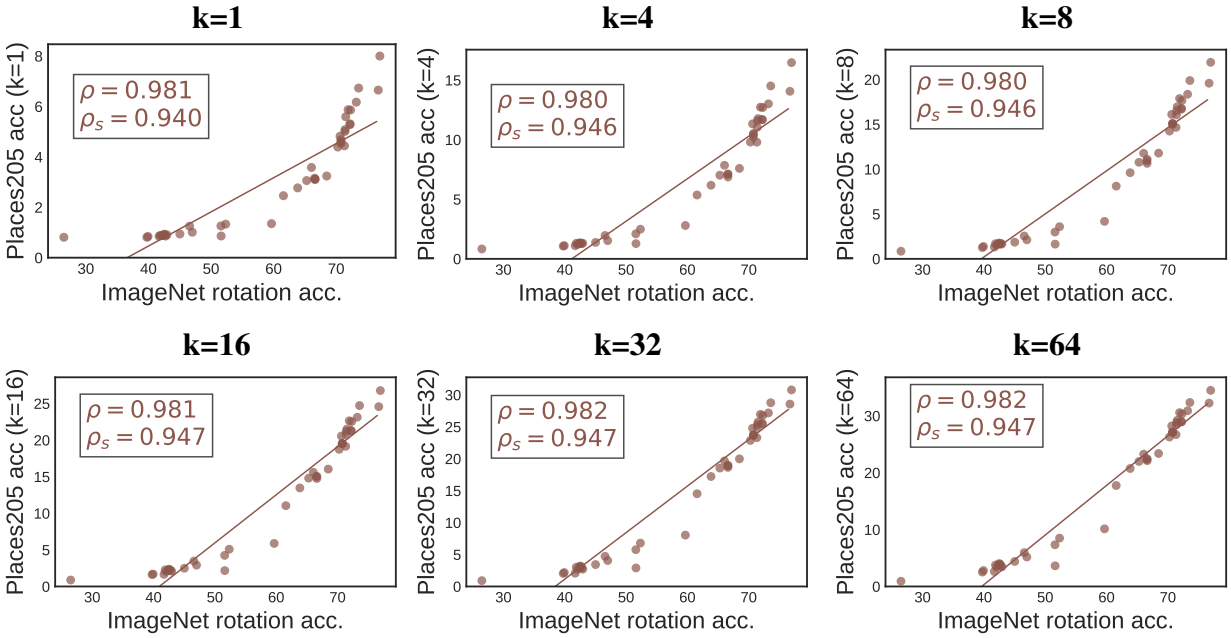

Figure 2.6: The ImageNet rotation prediction correlations with transfer performance to Places205 few label scene classification using $k = \{1, 4, 8, 16, 32, 64\}$ labels per scene class. The Spearman rank correlation with rotation prediction is indicated with ρ, while the rank correlation with the supervised ImageNet classification performance is indicated with ρ_s.

Correlation demonstration: finding and tuning transformation parameters

Figure 2.7 shows the performance of single-transform policies for CIFAR-10. The left and middle plots show the supervised classification accuracy compared with the InfoNCE loss and top-1 contrastive accuracy (how well the instance contrastive model predicts the augmented image pairs), while the right plot shows the rotation prediction accuracy for the image transformations in $\mathbb{O}$ evaluated after 100 training epochs. Using high or low values of InfoNCE or contrastive accuracy to select the best transformations would select a mixture of mediocre transformations, missing the top performing transformation in the middle. By using the rotation prediction, each transformation has a clear linear relationship with the supervised performance, enabling the unsupervised selection of the best transformations.

Next, Figure 2.8 demonstrates that the individual image transformation *parameters* can also be determined through rotation prediction. Specifically, we pre-trained CIFAR-10 using the default MoCoV2 augmentation policy. Then, for each of the fifteen transformation in $\mathbb{O}$, we applied the transformation with probability 1.0 and a magnitude λ randomly selected between 0 and $\{0.25, 0.5, 0.75, 1.0\}$. We evaluate the individual transformation's magnitude parameter using the supervised top-1 linear accuracy (classification) and self-supervised top-1 rotation prediction (rotation) as shown in Figure 2.8. Overall, the supervised linear classification and self-supervised rotation prediction select the the same magnitude parameter for 13 of the 15 transformations and

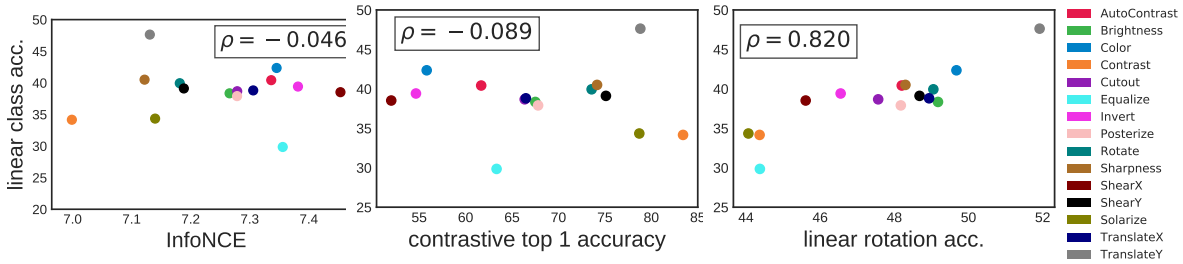

Figure 2.7: Similar to Figure 2.3, for CIFAR-10, we plot the supervised classification accuracy (y-axis) vs the InfoNCE loss function (left), contrastive top-1 accuracy (middle), and self-supervised linear rotation accuracy (right), for a self-supervised model trained using one of each transformation used by SelfAugment. Neither of the left two training metrics are a consistent measure of the quality of the representations, while the rotation prediction accuracy provides a strong linear relationship.

have a strong Spearman rank correlation of 0.929. Taken together, these results indicate that rotation prediction can be used to both select individual transformations as well as the parameters of the transformations for an augmentation policy.

Rotation correlation shows the benefit of Gaussian blur on ImageNet

In [16], the authors conducted a thorough, supervised investigation of a diverse set of image transformations that could be used in an augmentation policy for instance contrastive learning with ImageNet. The `Gaussian blur` augmentation was shown to be one of the most effective image transformations for ImageNet, and indeed, [21] released a follow-up paper showing that the MoCo framework [68] significantly benefits from its use. We find that our rotation-based evaluation similarly indicates that Gaussian blur is an effective image transformation for ImageNet under the same conditions studied in [21]:

Augmentation Policy	Top-1 Supervised Acc.	Rotate Acc.
MoCoV1	60.6	72.1
MoCoV2 no G-Blur	63.6	74.1
MoCoV2	67.7	77.0

Rotation invariant and black-and-white images

We note that certain types of images are not amenable to certain self-supervised evaluations. For instance, rotation evaluation will not work for rotation invariant images (such as images of textures) as the self-supervised evaluation task will not be able to discern the rotations. Similarly, a jigsaw task will not be able to discern images with a interchangeable quadrants (such as centered images of flowers), and a color prediction tasks will not work on black and white images (such as x-ray

images). Therefore, for each image dataset, we recommend using a self-supervised evaluation that does not evaluate an invariant factor of the image dataset, e.g. use rotation prediction if the images are black-and-white. We leave an investigation of the trade-offs between image invariances and self-supervised evaluations to future work.

Modified RV similarity analysis

To obtain a better understanding of why the the rotation evaluation has the best correlation with the supervised evaluation performance, we conduct a similarity analysis of the *activations* from the linear evaluation layers. Specifically, we use a modified RV coefficient (as in [154]) to measure the similarity between the activations from the rotation, jigsaw, and colorization evaluation layers on top of the frozen encoder network. The RV coefficient is a matrix correlation method that compares paired comparisons X and Y with different number of columns, and is defined as:

$$RV(X,Y) = \frac{tr(XX'YY')}{\sqrt{tr[(XX')^2]tr[(YY')^2]}} \tag{2.3}$$

The RV coefficient approaches 1 when datasets are small, even for random and unrelated matricies. To fix this issue, the modified RV coefficient (RV_2) ignores the diagonal elements of XX' and YY', which pushes the numerator to zero when X and Y are random matricies. Hence, the RV_2 similarity metric is less sensitive to dataset size.

$$RV_2(X,Y) = \frac{Vec(\tilde{XX'})'Vec(\tilde{YY'})}{\sqrt{Vec(\tilde{XX'})'Vec(\tilde{XX'}) \times Vec(\tilde{YY'})'Vec(\tilde{YY'})}} \tag{2.4}$$

Where $\tilde{XX'} = XX' - diag(XX')$ and similarly for $\tilde{YY'}$. This metric is invariant to orthogonal transformations and isomorphic scaling, but critically not invariant to arbitrary linear invertible linear transformations between representations (e.g. batch normalization). In [154], the authors show that the RV_2 metric could recover expected similarity patterns in neural networks and that it could be used to suggest hypotheses about intermediate representations in deep neural networks.

To study the similarity between linear layers trained using the rotation prediction, jigsaw, and colorization tasks and linear layers trained using supervised learning, we evaluated the RV_2 coefficient of 32 different linear layers trained on top of frozen encoders using the CIFAR-10 dataset. Each of the 32 encoders used a different augmentation policy during training. We evaluated the activations at the final linear layer across the entire validation set of CIFAR-10. We used the same set of image transformations before feeding each image into the the network (center crop to 28x28, and normalization across each channel by it's mean and std deviation across the dataset) across all self supervised tasks. As demonstrated in Figure 2.9, activations from rotation prediction layers had significantly stronger similarities with activations from supervised layers than other self supervised tasks. This demonstrates that the rotation prediction task uses the learned representation in a significantly more similar way for evaluation compared to the other evaluation tasks, and provides evidence that rotation evaluation performance not only correlates, but so does the activations from the evaluation layer.

Augmentation policy exploration via Bayesian optimization

As in [98], we used policy exploration search to automate the augmentation search. Since there are an infinite number of possible policies, we applied Bayesian optimization to explore augmentation strategies. In line 11 in Algorithm 1, we employed the Expected Improvement (EI) criterion as an acquisition function to explore $\mathcal{B}$ candidate policies efficiently: $EI(\mathcal{T}) = \mathbb{E}[min(\mathcal{L}(\theta,\phi|\mathcal{T}(\mathcal{D}_{\mathcal{A}}) - \mathcal{L}^{\dagger},0)]$. Here $L^{\dagger}$ represents a constant threshold determined by the quantile of observations amongst previously explored policies. As in [98], we used variable kernel density estimation on a graph-structured search space to approximate the criterion. Since this method is already implemented in the tree-structured Parzen estimator algorithm we used `Ray`[4] and `Hyperopt` to implement this in parallel.

In [98], the authors try to align distributions of data using supervised loss, then retrain the network using supervised loss. Since we do not directly retrain with the loss functions used to find augmentations, our method can instead be thought of as finding distributions of the data - via augmentation policies - that minimize (or maximize) alignment as defined by our loss functions.

SelfAugment Loss Functions

In this section we discuss the loss functions used for SelfAugment in greater detail and their resulting augmentation policies which are summarized in Figure 2.10.

- **Min. eval error**:

  $$\mathcal{T}^{SS} = \text{argmin}_{\mathcal{T}}\mathcal{L}_{\text{SS}}(\theta_{\mathcal{M}},\phi_{\text{ss}}|\mathcal{T}(D_{\mathcal{A}}))$$

  where $\mathcal{L}_{\text{SS}}$ is the self-supervised evaluation loss, which yields policies that should result in improved performance of the evaluation if we were to retrain the linear classifier with the selected augmentations. However, since the augmentations are instead used to create a contrastive learning task, it is important that we find a different set of augmentations that take the contrastive task into consideration. Indeed, using this loss function results in weak expected augmentation strengths (see Figure 2.10), resulting in relatively poor downstream performance (see Table 2.1).

- **Min. InfoNCE**:

  $$\mathcal{T}^{\text{I-min}} = \text{argmin}_{\mathcal{T}}\mathcal{L}_{\text{NCE}}(\theta_{\mathcal{M}}|\mathcal{T}(D_{\mathcal{A}}))$$

  where $\mathcal{L}_{\text{NCE}}$ is the InfoNCE loss from Eq. 2.1, which yields policies that make it easier to distinguish image pairs in the contrastive feature space. This loss function should results in small magnitude augmentations, since a trivial way to minimize InfoNCE is to apply no transforms - resulting in trivial minimization of the InfoNCE loss. Indeed, the loss function yields (i) lower expected augmentation strengths and (ii) emphasizes transformations like `Sharpness`, Figure 2.10. We did not expect this loss function to perform well, and was primarily included as a sanity check that minimizing InfoNCE would result in light augmentations.

[4] https://github.com/ray-project/ray

- **Max InfoNCE**:
$$\mathcal{T}^{\text{I-max}} = \text{argmin}_{\mathcal{T}} - \mathcal{L}_{\text{NCE}}(\theta_{\mathcal{M}}|\mathcal{T}(D_{\mathcal{A}}))$$
negates the previous loss function, yielding policies that make it difficult to distinguish image pairs in the feature space. In practice, this results in high magnitude augmentations and emphasizes augmentations like `Invert`. We hypothesized that maximizing InfoNCE could result in strong augmentations that could create a challenging contrastive task. However, the augmentations do not have any regularizing that would ensure the image maintains important features, leading to relatively suboptimal performance (see Table 2.1).

- **Min** $\mathcal{L}_{\text{ss}}$ **max** $\mathcal{L}_{\text{NCE}}$: $\mathcal{T}^{\text{minmax}} = \text{argmin}_{\mathcal{T}}\mathcal{L}_{\text{ss}} - \mathcal{L}_{\text{NCE}}$ yields policies with difficult transformations that maximize InfoNCE, while maintaining salient object features that minimize the self-supervised evaluation. When using a linear rotation evaluation prediction, we found this loss function to have the strongest performance for contrastive learning (§2.4). In practice, we normalized $\mathcal{L}_{\text{rot}}$ and $\mathcal{L}_{\text{NCE}}$ by their expected value for the training data across the K-folds when training with the base augmentation, since $\mathcal{L}_{\text{NCE}}$ was usually higher than $\mathcal{L}_{\text{rot}}$, but had each loss contribute equally for augmentation policy selection. Policies that optimized this loss emphasized transformations like `Equalize, AutoContrast` and `Contrast` more than others. For ImageNet, these transformations proved to be challenging yet useful. Notably, they closely resemble the effects of MoCov2 and SimCLR's Color Jitter [21].

Finally, while **min** $\mathcal{L}_{\text{rot}}$ **max** $\mathcal{L}_{\text{NCE}}$ works well, one could potentially gain performance by weighting the importance of minimizing $\mathcal{L}_{\text{rot}}$ vs maximizing $\mathcal{L}_{\text{NCE}}$. Hence we propose experimenting with different values of λ_{NCE} and λ_{rot} when optimizing the following objective: **min** $\lambda_{\text{rot}}\mathcal{L}_{\text{rot}}$ **max** $\lambda_{\text{NCE}}\mathcal{L}_{\text{NCE}}$.

Using the Original MoCoV2 [21] policy as the base policy

We replicated our SelfAugment results in Table 2.1 for the CIFAR10 dataset, where instead of using a single augmentation as the base policy, we used the full augmentation policy from [21] as the base policy. Results are shown in Table 2.5. This improved performance when we used the augmentation policies learned using min $\mathcal{L}_{\text{rot}}$ as feedback. Minimizing rotation may be the best approach for learning augmentation policies on top of established augmentation policies, because it produces augmentations that would improve performance if we retrained the linear classifier with the selected augmentations. Since the augmentation policy for contrastive learning is already quite strong when we use the MoCov2 augmentations as the base policy, this likely allows us to focus on improving generalization [98], leading to improved performance. Interestingly, the other SelfAugment Loss functions (see §2.5 for more detail) all deteriorated performance. This is likely because trying to change the contrastive performance of such a finely tuned set of augmentations requires more careful tuning. It is possible the minimax approach may result in improved performance with more careful tuning of the weighting of the rotation and contrastive losses.

It is also worth noting that using the best policy using the MoCov2 augmentations as the base augmentation set, then using SelfAugment with min $\mathcal{L}_{\text{rot}}$ as feedback slightly outperformed the

full SelfAugment pipeline with min $\mathcal{L}_{\text{rot}}$ max $\mathcal{L}_{\text{ICL}}$ as feedback. For datasets where researchers are confident that the augmentations from [21] are a strong base set of augmentations, this approach is worth exploring, and can be easily done using our pipeline. However, it is not possible when a good base augmentation policy is unknown. We leave further exploration of this approach as future research, as we chose to focus on the case where no existing augmentation policy is known.

unsup. feedback	C10
SelfAug (min rot)	**92.8**
SelfAug (min Info)	92.2
SelfAug (max Info)	90.5
SelfAug (minimax)	90.9
supervised feedback	
MoCov2[21]	92.3

Table 2.5: Results on CIFAR10 when using SelfAugment but using the augmentation policy from [21] as the base policy. Because the base augmentation policy is already strong, minimizing rotation loss during the augmentation policy search produces the best results. The results with the MoCoV2 augmentations as the base policy and minimization of rotation prediction slightly outperform the best results using SelfAugment and the minimax loss as feedback (92.6, see Table 2.1).

Computational Efficiency

SelfAugment can be broken up into three steps:

1. **Finding the base augmentation**. This entails training 16 instances of MoCoV2 for 10-15% of the total epochs typically used for pre-training, using all of the training data available.

2. **Training on K-Folds using the base augmentation**. For CIFAR-10/SVHN we used the entire training dataset for this step, and evaluated for the same number of epochs as we use to train the final models, using all the training data from each fold. For ImageNet, we used a subset of 50,000 images and trained for 5x longer, to make up for the smaller amount of images. This was done to improve the computational efficiency of step 3.

3. **Finding augmentation policies**. Because we only need to complete forward passes of the network, this is relatively efficient. We use the held out data from each of the K-folds to evaluate the model with different augmentation policies applied.

It is important to note that steps 1 and 2 are *shared* across all augmentation policies found with SelfAug, meaning that they do not need to be repeated to find a new augmentation policy using a different loss function. This could allow for rapid experimentation with new loss functions for SelfAugment in the future.

Meanwhile, SelfRandAug's computational time is completely determined by the user-defined search space over λ and N_τ. However, because it requires training multiple models to completion on the entire dataset, it is more computationally intensive than SelfAug. SelfRandAug's computation time could be reduced by training on a subset of Images in a large training set.

Finally, it is worth noting that evaluating a *single* augmentation's various hyperparameters for MoCoV2 can be extremely computationally expensive. Because pre-training MoCoV2 for 100 epochs then training a linear head for evaluation takes a total of 244 GPU Hours on a Tesla V100 GPU, searching over various augmentations and their strength and probability parameters can easily require thousands of GPU hours. Hence, an additional contribution of this work is providing a fast, automatic method for selecting augmentations for instance contrastive learning.

Table 2.6: Computational time, measured in one hour on one NVIDIA Tesla V100 GPU for SelfAugment and SelfRandAug on ImageNet. SelfAug times are evaluated when using the minimax loss function, which takes the most time because we evaluate both InfoNCE and rotation loss. For SelfRandAug, computational time reflects our grid search over $\lambda = \{5,7,9,11,13\}$, $N_\tau = 2$, plus the time to train a rotation head on top of each representation. This time could be larger or smaller depending on the parameters λ and N_τ a user wants to search over.

Algorithm	Find Base Aug	Train	Find Aug	**Total**
SelfAug	476.6	155.0	97.7	**729.4**
SelfRandAug				**1220.0**

2.6 Discussion

We have shown that a self-supervised rotation evaluation has a strong correlation with supervised evaluation (outperforming jigsaw/colorization tasks) and that this evaluation can be incorporated into an effective loss function for efficient augmentation selection (§2.4). Here, we further reflect on the utility of the self-supervised rotation evaluation task.

If rotation prediction is highly correlated with supervised evaluations, why not directly train on it? As discussed in §2.5, the authors of [86] found that rotation accuracy from training a full network on rotation prediction was only weakly correlated with supervised performance. Furthermore, as discussed in §2.5, using a 2-layer MLP for rotation prediction drops the correlation from $\rho = .97$ to .90 even though the prediction accuracy improves. This indicates that while rotation prediction using a linear combination of the learned representations is an effective *evaluation*, actually learning the representations via rotation prediction is not as strongly correlated.

Shouldn't minimizing rotation error yield the best policies? We evaluate policies only *after* contrastive training, so minimizing rotation error yields policies that improve rotation prediction if we were to retrain the linear classifier. However, as indicated by the poor performance of using the rotation-error as the loss function, it is important to also take the contrastive task into consideration when retraining the entire network. To explore this idea further, we minimize a

supervised classification loss function for SelfAugment. Due to the strong correlation between the rotation task and supervised evaluation, this results in similar performance to minimizing rotation loss with SelfAugment: we observe an accuracy of 90.7 on CIFAR-10 and 94.9 on SVHN using supervised feedback (rotation evaluation yields an accuracy of 91.0 on CIFAR-10 and 93.7 on SVHN). This performance is worse than the loss functions that simultaneously maximize the InfoNCE loss, which encourage difficult augmentation policies (see § 2.5).

Which SelfAugment loss function should be used on a new, unlabeled dataset? When training with a new unlabeled dataset, we recommend starting with the minimax SelfAugment loss function for augmentation policy selection. This recommendation stems from its superior performance across all datasets and tasks. For comparison, the SelfAugment loss function minimizing the InfoNCE produced results that were often worse than the baseline policy, while both maximizing the InfoNCE or minimizing rotation prediction exceeded the baseline, but generally did not surpass MoCoV2's policy. In § 2.5, we show the effective magnitude of each individual transformation for each loss functions. We see that, as expected, the minimax loss function produces policies with intermediate magnitudes across all datasets. Its strong performance indicate that these intermediate magnitudes have found a *sweet spot* where the augmentations strike a balance between creating difficult instance contrastive tasks and retaining salient image features.

It is worth noting that across both SVHN and CIFAR-10, SelfAugment with min InfoNCE loss function results in *worse* results than simply using the baseline augmentation. We believe this is because the selected augmentations make it easier to tell samples apart using weak augmentations that do not actually add meaningful contrastive tasks (see § 2.5). These augmentations minimize InfoNCE but do not appear to be helpful in improving the model's ability to generalize to new data. Accordingly, the model performs poorly when retrained with these augmentations.

Computational efficiency: Importantly, SelfAugment and RandAugment provide an efficient framework for finding augmentations for instance contrastive learning without using labels. Prior work [16] performed grid searches over various augmentation policies, requires training many models to completion. Training a single instance of MoCoV2 on ImageNet using 4 V100 GPUs for 100 epochs takes 52 hours (208 GPU hours). SelfAugment finds useful, and typically improved, augmentation policies in 730 GPU hours, and can learn additional augmentation policies in an additional 98 GPU Hours (see §2.5 for details). This is less than the computational cost of training just four total instances. For comparison, [16] reported at least 49 full training runs.

2.7 Conclusion

In this chapter, we identified the problem that self-supervised representations are evaluated using labeled data, oftentimes using the labels from the "unlabeled" training dataset itself. We established that a self-supervised image rotation task is strongly correlated with the supervised performance for standard computer vision recognition tasks, and as a result, can be used to evaluate learned representations. Using this evaluation, we establish two unsupervised data augmentation policy selection algorithms and show that they can outperform or perform comparably to policies obtained using supervised feedback.

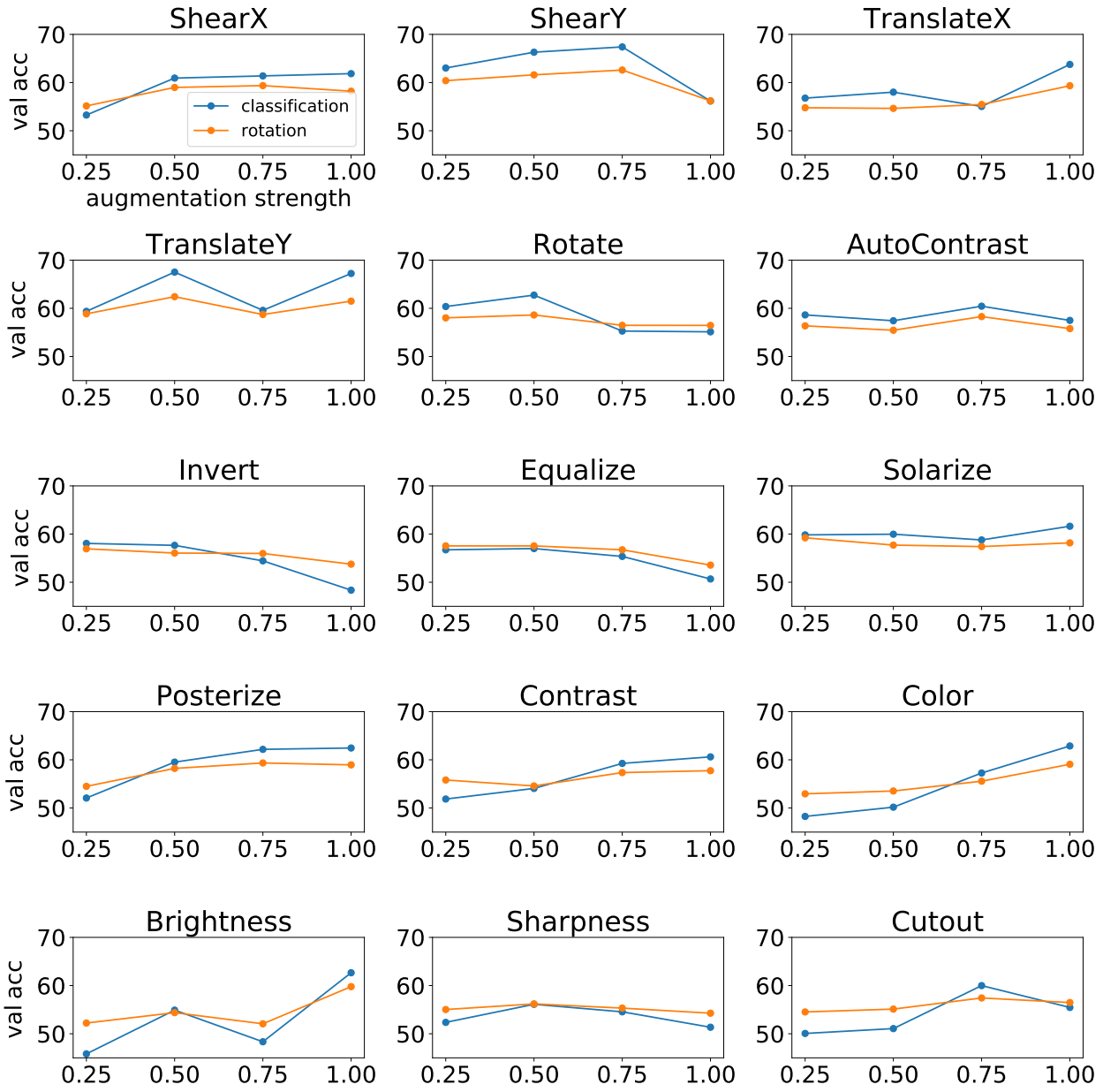

Figure 2.8: Result of magnitude sweeps on CIFAR-10 for the 15 transformations optimized in SelfAugment. For each augmentation, we train MoCoV2 for 250 epochs using the usual MoCoV2 augmentations, and then added the single augmentation on top. We then vary the range of possible augmentation strength λ parameter to be between 0 and the value on the x-axis, and randomly select a value in that range, and apply the augmentation with 100 % probability. Rotation accuracy and classification accuracy have Spearman rank correlation $\rho = .929$, demonstrating how the relationship between rotation accuracy and classification accuracy can be used to select the parameters of individual transformations.

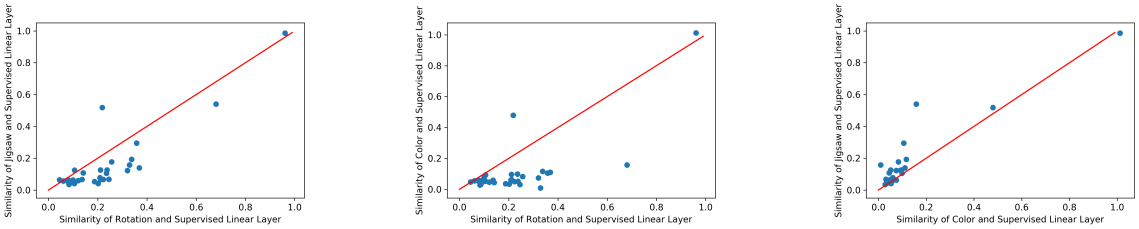

Figure 2.9: Comparisons of RV_2 Similarity [154] between each self supervised task studied and supervised linear layers. Rotation prediction layers have stronger similarity with supervised linear layers than either of the jigsaw and colorization self supervised tasks ($p < 2.3x10^{-5}$, Wilcoxon one-tailed signed rank test). The datapoint with highest similarity was using the strongest settings of SelfRandAugment, which learned poor representations with near chance performance on the supervised downstream task, explaining why similarity was so high across tasks. The strong similarity between rotation prediction and supervised linear layers suggests that they use the underlying representations in a similar way for evaluation, explaining why rotation prediction makes for a strong evaluation metric that correlates more strongly with supervised evaluation than the jigsaw and colorization evaluations.

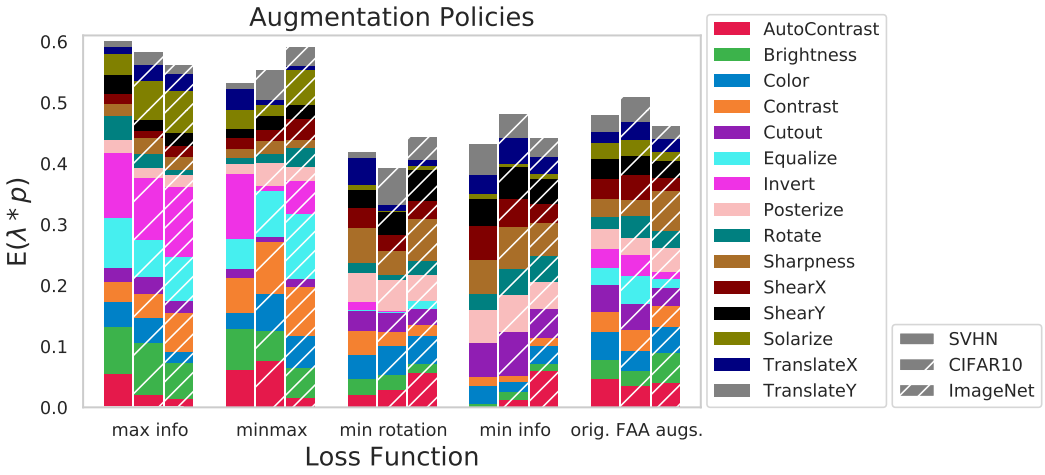

Figure 2.10: Visualization of the augmentation policies found with SelfAugment as well as Fast AutoAugment for supervised learning. This figure shows expected augmentation strength (mean magnitude$\times$normalized probability) of each augmentation, evaluated across all datasets and loss functions. As expected, minimizing InfoNCE results in augmentations with smaller magnitude, and emphasizes augmentations that do not alter the image much (e.g. `Sharpness`). Maximizing InfoNCE results in augmentations with a larger magnitude and emphasizes augmentations (e.g. `Invert, Solarize`) which heavily alter the image. The minimax loss function yields an augmentation policy that strikes a middle ground between the two, with strong augmentations, but reduced emphasis on heavy augmentations like `Invert`. Comparison of the policies that worked best with contrastive learning relative to the original FAA policies reveals that our policies are (i) stronger, and (ii) have more variability in a single augmentation's E(λ*p) relative to policies that were used for supervised learning.

Chapter 3

Data Efficient Self-Supervised Representation Learning

This chapter discussed various methods for data-efficient representation learning. In particular, this chapter focuses on how a model can be trained via self-supervised learning using the least amount of data possible.

3.1 Introduction

Recently, self-supervised pretraining – an unsupervised pretraining method that self-labels data to learn salient feature representations – has outperformed supervised pretraining in an increasing number of computer vision applications [16, 21, 14]. These advances come from *instance contrastive learning*, where a model is trained to identify visually augmented images that originated from the same image from a set [41, 167]. Typically, self-supervised pretraining uses unlabeled *source* data to pretrain a network that will be *transferred* to a supervised training process on a *target* dataset. Self-supervised pretraining is particularly useful when labeling is costly, such as in medical and satellite imaging [145, 25].

However, self-supervised pretraining requires long training time on large datasets, e.g. SimCLR [16] showed improved performance out to 3200 epochs on ImageNet's 1.2 million images [142]. In addition, instance contrastive learning is sensitive to the data augmentation policies and many trials are needed to determine the right settings [137, 170].

The computational intensity and sensitivity of self-superivsed pretraining may lead researchers to seek self-supervised models from model zoos and research repositories. However, models pretrained on domain-specific datasets are not commonly available. In turn, many practitioners do not use a model pretrained on data similar to their target data, but instead, use a pretrained, publicly available model trained on a large, general dataset, such as ImageNet. We refer to this process as **generalist pretraining**. A growing body of research indicates that pretraining on domain-specific datasets, which we refer to as **specialist pretraining**, leads to improved transfer performance [133, 115, 120].

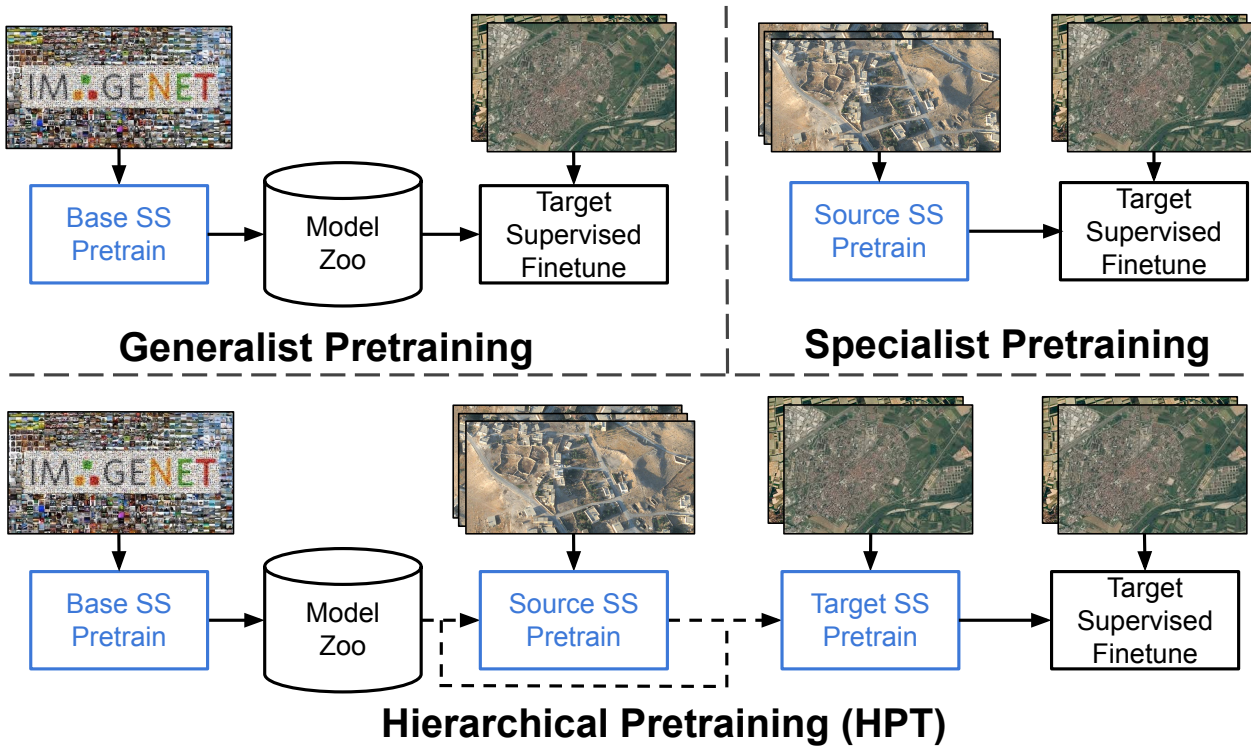

Figure 3.1: *Methods of using self-supervision.* The top row are the two common prior approaches to using self-supervised (SS) pretraining. In *Generalist Pretraining*, a large, general, *base* dataset is used for pretraining, e.g. ImageNet. In *Specialist Pretraining*, a large, specialized *source* dataset is collected and used for pretraining, e.g. aerial images. In this chapter, we explore *Hierarchical Pre-Training* (HPT), which sequentially pretrains on datasets that are similar to the target data, thus providing the improved performance of specialist pretraining while leveraging existing models.

Figure 3.1 formalizes this categorization of self-supervised pretraining methods. Generalist and specialist pretraining are as described above, with one round of self-supervised pretraining on a domain-general and domain-specific dataset, respectively. **Hierarchical Pretraining** refers to models pretrained on datasets that are progressively more similar to the target data. HPT first pretrains on a domain-general dataset (referred to as the *base pretrain*), then optionally pretrains on domain-specific datasets (referred to as the *source pretrain*), before finally pretraining on the target dataset (referred to as the *target pretrain*). In all cases, pretraining is followed by supervised finetuning on the target task.

Specialist pretraining presents the same core challenge that transfer learning helps alleviate: a sensitive training process that requires large datasets and significant computational resources [89]. While transfer learning has been carefully investigated in supervised and semi-supervised settings for computer vision [149], it has not been formally studied for self-supervised pretraining, itself. Furthermore, several recent papers that apply self-supervised learning to domain-specific problems

did not apply transfer learning to the pretraining process itself, which motivated our work [151, 3, 93].

In this chapter, we investigate the HPT framework with a diverse set of pretraining procedures and downstream tasks. We test 16 datasets spanning visual domains, such as medical, aerial, driving, and simulated images. In our empirical study, we observe that HPT shows the following benefits compared to self-supervised pretraining from scratch:

- HPT reduces self-supervised pretraining convergence time up to 80× compared to pretraining from scratch.
- HPT consistently converges to better performing representations than generalist or specialist pretraining for 15 of the 16 studied datasets on image classification, object detection, and semantic segmentation tasks.
- HPT is significantly more resilient to the set of image augmentations and amount of data used during self-supervised pretraining.

The following sections provide the background, methodology, and experiments used to reach these conclusion and the appendix significantly broadens the scope of our analyses. From this experimental effort, our key conclusion is straightforward: *self-supervised pretraining improves self-supervised pretraining*.

3.2 Background and Related Work

Transfer learning studies how a larger, more general, or more specialized *source* dataset can be leveraged to improve performance on *target* downstream datasets/tasks [134, 131, 7, 35, 73, 68, 39, 43, 185, 57, 95, 132]. this chapter focuses on a common type of transfer learning in which model weights trained on source data are used to initialize training on the target task [183]. Model performance generally scales with source dataset size and the similarity between the source and target data [133, 115, 120].

A fundamental challenge for transfer learning is to improve the performance on target data when it is not similar to source data. Many works have tried to increase performance when the target and source datasets are not similar. Recently, [129] proposed first training on the base dataset and then training with subsets of the base dataset to create specialist models, and finally using the target data to select the best specialist model. Similarly, [118] used target data to reweight the importance of base data. Unlike these works, we do not revisit the base data, modify the pretrained architecture, or require expert model selection or reweighting.

Self-supervised pretraining is a form of unsupervised training that captures the intrinsic patterns and properties of the data without using human-provided labels to learn discriminative representations for the downstream tasks [37, 38, 186, 53, 160]. In this work we focus on a type of self-supervised pretraining called *instance contrastive learning* [41, 167, 68], which trains a network by determining which visually augmented images originated from the same image, when contrasted with augmented images originating from different images. Instance contrastive learning

has recently outperformed supervised pretraining on a variety of transfer tasks [68, 18], which has lead to increased adoption in many applications. Specifically, we use the MoCo algorithm [21] due to its popularity, available code base, reproducible results without multi-TPU core systems, and similarity to other self-supervised algorithms [84]. We also explore additional self-supervised methods in the appendix.

Our focus is on self-supervised learning for vision tasks. Progressive self-supervised pretraining on multiple datasets has also been explored for NLP tasks, e.g. see [64, 127] and the citations within. In [64], the authors compare NLP generalist models with models trained on additional source and task-specific data. While our work is similar in spirit to the language work of [64], our work focuses on computer vision, includes a greater variation of pretraining pipelines, and allows for adaptation with fewer parameter updates.

Label-efficient learning includes weak supervision methods [110] that assume access to imperfect but related labels, and semi-supervised methods that assume labels are only available for a subset of available examples [18, 87, 174]. While some of the evaluations of the learned representations are done in a semi-supervised manner, HPT is complementary to these approaches and the representations learned from HPT can be used in conjunction with them.

3.3 Hierarchical pretraining

In this section, we formalize each of the HPT components as depicted in Figure 3.1.

Base pretraining: We use the term *base pretraining* to describe the initial pretraining step where a large, general vision dataset (*base dataset*) is used to pretrain a model from scratch. Practically, few users will need to perform base pretraining, and instead, can use publicly available pretrained models, such as ImageNet models. Because base pretraining, like many prior transfer learning approaches, is domain agnostic, most practitioners will select the highest performing model on a task with a large domain [88].

Source pretraining: Given a base trained model, we select a source dataset that is both larger than the target dataset and more similar to the target dataset than the base dataset. Many existing works have explored techniques to select a model or dataset that is ideal for transfer learning with a target task [139]. Here, we adopt an approach studied by [89, 139] in a supervised context called a *task-aware search strategy*: each potential source dataset is used to perform self-supervised pretraining on top of the base model for a very short amount of pretraining, e.g. $\sim$5k pretraining steps as discussed in Section 4.4. The supervised target data is then used to train a linear evaluator on the frozen source model. The source model is then taken to be the model that produces the highest linear evaluation score on the target data, and is then used for additional target pretraining.

Experimentally, we have found that using a single, similar, and relatively large (e.g. $>$ 30K images) source dataset consistently improves representations for the target task. Furthermore, we view source pretraining as an optional step, and as shown in Section 4.4, HPT still leads to improved results when directly performing self-supervised pretraining on the target dataset following the base pretraining. We further discuss source model selection in the appendix.

Target pretraining: Finally, we perform self-supervised pretraining with the target dataset, initialized with the final source model, or the base model in the case when no source model was used. This is also the stage where layers of the model can be frozen to prevent overfitting to the target data and enable faster convergence speed. Experimentally, we have found that freezing all parameters except the modulation parameters of the batch norm layers leads to consistently strong performance for downstream tasks when the target dataset is relatively small ($<$ 10K images).

Supervised finetune: Given the self-supervised pretrained model on the target dataset, we transfer the final model to the downstream target task, e.g. classification.

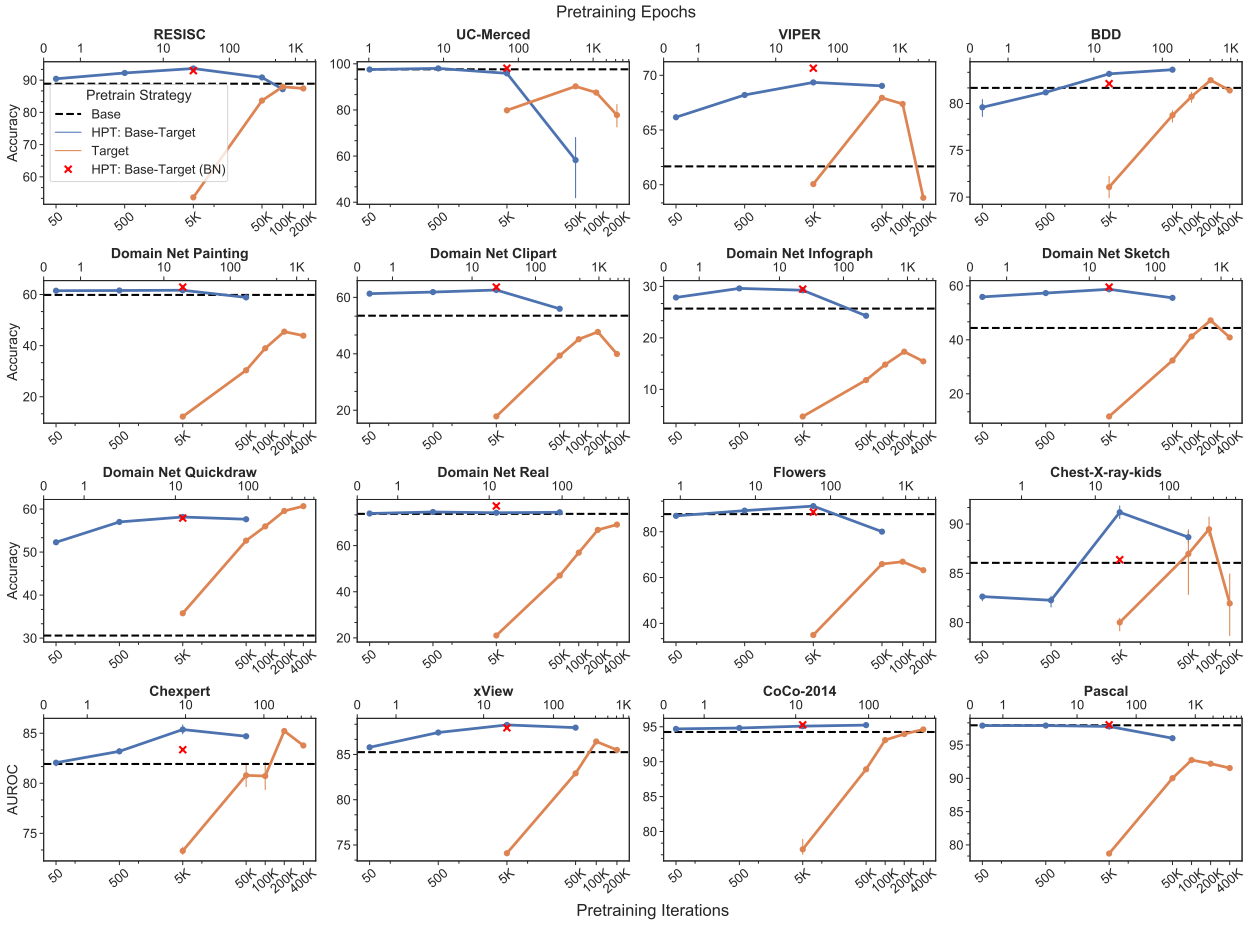

Figure 3.2: *Linear separability evaluation*. For each of the 16 datasets, we train a generalist model for 800 epochs on ImageNet (Base). We either train the whole model from 50-50k iters (HPT Base-Target) or just the batch norm parameters for 5k iters (HPT Base-Target (BN)). We compare HPT to a Specialist model trained from a random initialization (Target). For each, we train a linear layer on top of the final representation. HPT obtains the best results on 15 out of 16 datasets without hyperparameter tuning.

3.4 Experiments

Through the following experiments, we investigate the quality, convergence, and robustness of self-supervised pretraining using the HPT framework.

Datasets

We explored self-supervised pretraining on the following datasets that span several visual domains (see the appendix for all details). Dataset splits are listed with a train/val/test format in square brackets after the dataset description.

Aerial: **xView** [92] is a 36-class object-centric, multi-label aerial imagery dataset. **RESISC** [25] is a 45-class scene classification dataset for remote sensing [18900/6300/6300]. **UC-Merced** [180] is a 21-class aerial imagery dataset [1260/420/420].

Autonomous Driving: **BDD** [184] is a high resolution driving dataset with 10 object detection labels and 6 weather classification labels. We evaluate HPT performance over the object detection task, as well as the weather classification task [60k/10k/10k]. **VIPER** [140] is a 23-class simulated driving dataset for which we perform multi-label each object in the image [13367/2868/4959].

Medical: **Chexpert** [79] is a large, multi-label X-ray dataset, where we determine whether each image has any of 5 conditions [178731/44683/234]. **Chest-X-ray-kids** [83] provides pediatric X-rays used for 4-way pneumonia classification [4186/1046/624].

Natural, Multi-object: **COCO-2014** [101] is an 81-class object detection benchmark. We perform multi-label classification for each object, and we further use the 2017 split to perform object detection and segmentation [82783/20252/20252]. **Pascal** VOC 2007+2012 [47] is a standard 21-class object detection benchmark we use for multi-label classification to predict whether each object is in each image. We also use the object detection labels for an object detection transfer task [13.2k/3.3k/4.9k].

Assorted: **DomainNet** [126] contains six distinct datasets, where each contains the same 345 categories. The domains consist of `real` images similar to ImageNet, `sketch` images of greyscale sketches, `painting` images, `clipart` images, `quickdraw` images of binary black-and-white drawings from internet users, and `infograph` illustrations. We use the original train/test splits with 20% of the training data used for validation. **Oxford Flowers** [119]: we use the standard split to classify 102 fine-grain flower categories [1020/1020/6149].

Evaluations

The features of self-supervised pretrained models are typically evaluated using one of the following criteria:

- **Separability**: Tests if a linear model can differentiate classes in a dataset using learned features. Good representations should be linearly separable [122, 27].
- **Transferability**: Tests the performance of the model when finetuned on new datasets and tasks. Better representations will generalize to more tasks [68].

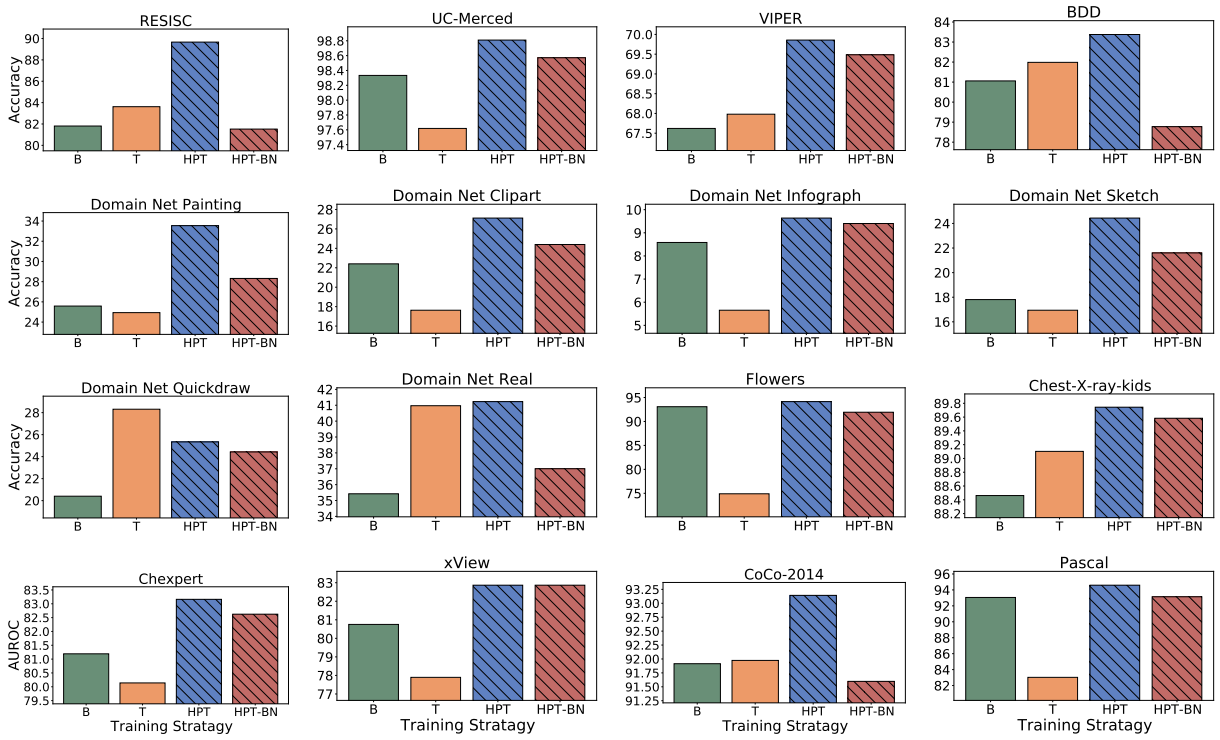

Figure 3.3: *Semi-supervised evaluation*. We compared the best semi-supervised finetuning performance from the (B)ase model, (T)arget pretrained model, HPT pretrained model, and HPT-BN pretrained model using a 1k labeled subset of each dataset. Despite performing 10x-80x less pretraining, HPT consistently outperformed the Base and Target. HPT-BN generally showed improvement over Base model transfer, but did not surpass HPT's performance.

- **Semi-supervised**: Test performance with limited labels. Better representations will suffer less performance degradation [73, 16].

We explored these evaluation methods with each of the above datasets. For all evaluations, unless otherwise noted, we used a single, centered crop of the test data with no test-time augmentations. For classification tasks, we used top-1 accuracy and for multi-label classification tasks we used the Area Under the ROC (AUROC) [11].

In our experiments, we used MoCo-V2 [21] as the self-supervised training algorithm. We selected MoCo-V2 as it has state-of-the-art or comparable performance for many transfer tasks, and because it uses the InfoNCE loss function [122], which is at the core of many recent contrastive pretraining algorithms [105]. Unless otherwise noted, all training is performed with a standard ResNet-50 backbone [148] on 4 GPUs, using default training parameters from [68]. We also explored additional self-supervised pretraining algorithms and hyperparameters in the appendix.

In the following experiments, we compare implementations of the following self-supervised pretraining strategies:

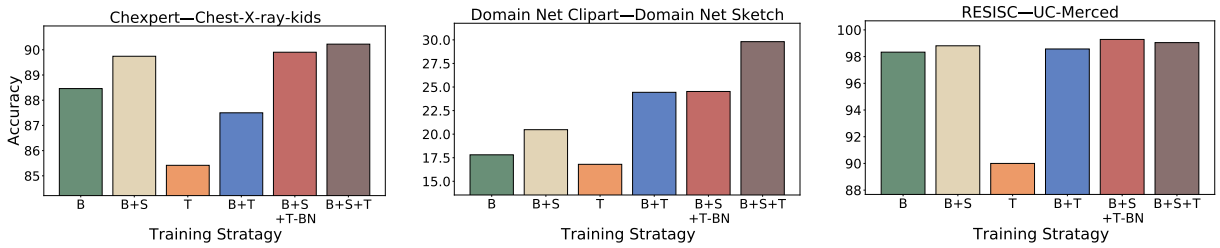

Figure 3.4: *Full finetuning evaluations.* Finetuning performance on target datasets. For these datasets, we evaluated the performance increase on the target dataset by pretraining on sequences of (B)ase (ImageNet), (S)ource (left) dataset, and (T)arget (right) dataset. All HPT variants beat all baselines, with HPT-BN getting slightly better performance on UC Merced and B+S+T having the best performance elsewhere.

- **Base**: transfers the 800-epoch MoCo-V2 ImageNet model from [21] and also updates the batch norm's non-trainable mean and variance parameters using the target dataset (this uniformly led to slightly improved performance for Base transfer).
- **Target**: performs MoCo-V2 on the target dataset from scratch.
- **HPT**: initializes MoCo-V2 pretraining with the 800-epoch MoCo-V2 ImageNet model from [21], then optionally performs pretraining on a source dataset before pretraining on the target dataset. The batch norm variant (**HPT-BN**) only trains the batch norm parameters (γ, β), e.g. a ResNet-50 has 25.6M parameters, where only $\sim$0.2% are BN parameters.

Existing work largely relies on supervised evaluations to tune the pretraining hyperparameters [16], but in practice, it is not possible to use supervised evaluations of unlabeled data to tune the hyperparameters. Therefore, to emphasize the practicality of HPT, we used the default pretraining hyperparameters from [21] with a batch size of 256 (see the appendix for full details).

Pretraining Quality Analysis

Separability analysis: We first analyzed the quality of the learned representations through a linear separability evaluation [16]. We trained the linear model with a batch size of 512 and the highest performing learning rate of $\{0.3, 3, 30\}$. Similar to [89], we used steps rather than epochs to allow for direct computational comparison across datasets. For Target pretraining, we pretrained for {5k, 50k, 100k, 200k, 400k} steps, where we only performed 400k steps if there was an improvement between 100k and 200k steps. For reference, one NVIDIA P100 GPU-Day is 25k steps. We pretrained HPT for much shorter schedules of {50, 500, 5k, 50k} steps, and HPT-BN for 5k steps – we observed little change for HPT-BN after 5k steps.

Key observations: From Figure 3.2, we observe that HPT typically converges by 5k steps of pretraining *regardless of the target dataset size*, and that for 15 out of 16 datasets, HPT and HPT-BN

converged to models that performed as well or better than the Base transfer or Target pretraining at 400k steps (80x longer). The only dataset in which the Target pretraining outperformed HPT was `quickdraw` – a large, binary image dataset of crowd-sourced drawings. We note that `quickdraw` is the only dataset in Target pretraining at 5k steps outperformed directly transferring the Base model, indicating that the direct transfer performance from ImageNet is quite poor due to a large domain gap – an observation further supported by its relatively poor domain adaptation in [126].

HPT improved performance on RESISC, VIPER, BDD, Flowers, xView, and `clipart`, `infograph`, and `sketch`: a diverse range of image domains and types. HPT had similar performance as Base transfer for the datasets that were most similar to ImageNet: `real`, COCO-2014, and Pascal, as well as for UC-Merced, which had 98.2% accuracy for Base transfer and 99.0% accuracy for HPT and HPT-BN. The two medical datasets, Chexpert and Chest-X-ray-kids had comparable performance with HPT and Target pretraining, yet HPT reached equivalent performance in 5k steps compared to 200k and 100k, respectively. Finally, HPT exhibited overfitting characteristics after 5k steps, where the overfitting was more pronounced on the smaller datasets (UC-Merced, Flowers, Chest-X-ray-kids, Pascal), leading us to recommend a very short HPT pretraining schedule, e.g. 5k iterations, regardless of dataset size (see appendix for additional experiments).

Semi-supervised transferability: Next, we conducted a semi-supervised transferability evaluation of the pretrained models. This experiment tested whether the benefit from the additional pretraining is nullified when finetuning all model parameters. Specifically, we selected the top performing models from the linear analysis for each pretraining strategy and fully finetuned the pretrained models using 1000 randomly selected labels without class balance but such that each class occured at least once. We finetune using a combination of two learning rates (0.01, 0.001) and two finetuning schedules (2500 steps, 90 epochs) with a batch size of 512 and report the top result for each dataset and model – see the appendix for all details.

Key observations: Figure 3.3 shows the top finetuning performance for each pretraining strategy. The striped bars show the HPT pretraining variants, and we observe that similar to the linear analysis, HPT has the best performing pretrained models on 15 out of 16 datasets, with `quickdraw` being the exception. One key observation from this experiment is that HPT is beneficial in the semi-supervised settings and that the representational differences from HPT and the Base model are different enough that full model finetuning cannot account for the change. We further note that while HPT-BN outperformed HPT in several linear analyses, HPT-BN never outperformed HPT when finetuning all parameters. This result indicates that some of the benefit from pretraining only the batch norm parameters is redundant with supervised finetuning. We also note that whether Base or Target pretraining performed better depended on the dataset, while HPT had uniformly strong performance.

Sequential pretraining transferability: Here, we explore HPT's performance when pretraining on a source dataset before pretraining on the target dataset and finally transferring to the target task. We examined three diverse target datasets: Chest-X-ray-kids, `sketch`, and UC-Merced. We select the source dataset for each of the target dataset by choosing the source dataset that yielded the highest linear evaluation accuracy on the target dataset after 5k pretraining steps on top of the base model. This selection yielded: ImageNet then Chexpert then Chest-X-ray-kids, ImageNet then `clipart` then `sketch`, and ImageNet then RESISC then UC-Merced.

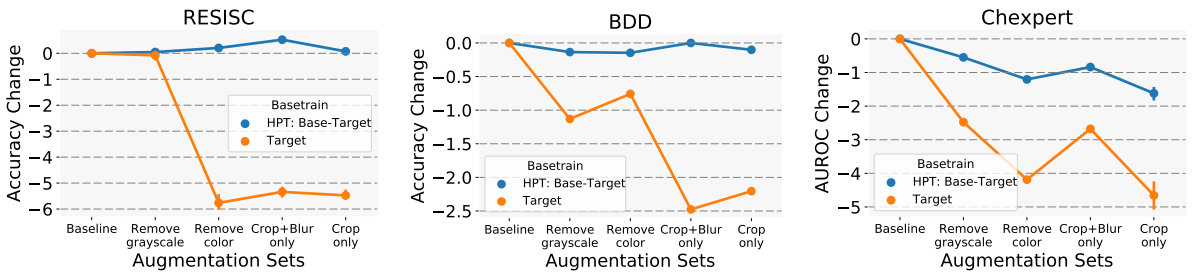

Figure 3.5: *Augmentation robustness*. We compare the accuracy change of sequentially removing data augmentations on linear evaluation performance. HPT performs better with only cropping than any other policy does with any incomplete combination.

Key observations: Figure 3.4 compares finetuning the 1000-label subset of the target data after the following pretraining strategies: directly using the Base model (B), Target pretraining (T), Base then Source pretraining (B+S), Base then Target pretraining (B+T), Base then Source pretraining then Target pretraining (B+S+T), and Base then Source pretraining then Target pretraining on the batch norm parameters (B+S+T-BN). The full HPT pipeline (B+S+T) leads to the top results on all three target datasets. In the appendix, we further show that the impact of an intermediate source model decreases with the size of the target.

Object detection and segmentation transferability: For Pascal and BDD, we transferred HPT pretrained models to a Faster R-CNN R50-C4 model and finetuned the full model; for COCO, we used a Mask-RCNN-C4. Over three runs, we report the median results using the COCO AP metric as well as AP_{50}/AP_{75}. For Pascal, we performed finetuning on the `train2007+2012` set and performed evaluation on the `test2007` set. For BDD we used the provided train/test split, with 10k random images in the train split used for validation. For COCO, we used the 2017 splits and trained with the 1x schedule (see appendix for all details).

Key observations: Tables 3.1-4.5 show the object detection and segmentation results. For Pascal, we tested HPT instantiations of Base-Target, Base-Target (BN), and Base-Source-Target, where COCO-2014 was selected as the source model using the top-linear-analysis selection criteria. For the larger BDD and COCO datasets, we tested Base-Target and Base-Target (BN). Overall, the results are consistent across all datasets for image classification, object detection, and segmentation: HPT: both Base-Target and Base-Target (BN) lead to improvements over directly transferring the Base model to the target task.

The Base-Source-Target Pascal results show an improvement when pretraining all model parameters, but remain consistent when only pretraining the batch norm parameters. This indicates that while the batch norm parameters can improve the pretrained model, sequentially pretraining from the source to the target on these values does not always yield an improved result. While the overall gains are modest, these results indicate that HPT is not directly learning redundant information with either the MoCo pretraining or the finetuning task. Furthermore, it is surprising that only tuning the batch norm parameters on the target dataset leads to an improvement in object detection, and

Table 3.1: Transfer Result: This table reports the median AP, AP_{50}, AP_{75} over three runs of finetuning a Faster-RCNN C4 detector. For Pascal, the Source dataset is COCO-2014. A bold result indicates a $+0.2$ improvement over all other pretraining strategies.

Pretrain	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
`Pascal VOC07`			
Target	48.4	75.9	51.9
Base	57.0	82.5	63.6
HPT: Base-Target	57.1	**82.7**	63.7
HPT: Base-Target (BN)	**57.5**	**82.8**	64.0
HPT: Base-Source-Target	**57.5**	**82.7**	**64.4**
HPT: Base-Source-Target (BN)	**57.6**	**82.9**	64.2
`BDD`			
Target	24.3	46.9	24.0
Base	27.1	48.7	25.4
HPT: Base-Target	**28.1**	**50.0**	**26.3**
HPT: Base-Target (BN)	**28.0**	49.6	**26.3**

Table 3.2: Transfer Result: This table reports the median AP, AP_{50}, AP_{75} over three runs of finetuning a Mask-RCNN-C4 detector on COCO-2017. A bold result indicates at least a 0.2 improvement over all other pretraining strategies.

Pretrain	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
Target	36.0	54.7	38.6	19.3	40.6	49.1
Base	38.0	57.4	41.3	20.7	43.3	51.4
HPT: B-T	**38.4**	**58.0**	41.3	**21.6**	**43.5**	**52.2**
HPT: B-T (BN)	38.2	57.4	40.9	20.6	43.4	**52.2**

we note that pretraining specific subsets of object detector backbone parameters may provide a promising direction for future work.

HPT Robustness

Here, we investigate the robustness of HPT to common factors that impact the effectiveness of self-supervised pretraining such as the augmentation policy [16, 137] and pretraining dataset size [117]. For these robustness experiments, we used the BDD, RESISC, and Chexpert datasets as they provided a diversity in data domain and size.

Augmentation robustness: MoCo-V2 sequentially applies the following image augmentations: `RandomResizedCrop`, `ColorJitter`, `Grayscale`, `GaussianBlur`, and `RandomHorizontalFlip`. We studied the robustness of HPT by systematically removing these augmentations and evaluating the change in the linear evaluation for HPT and Target pretraining.

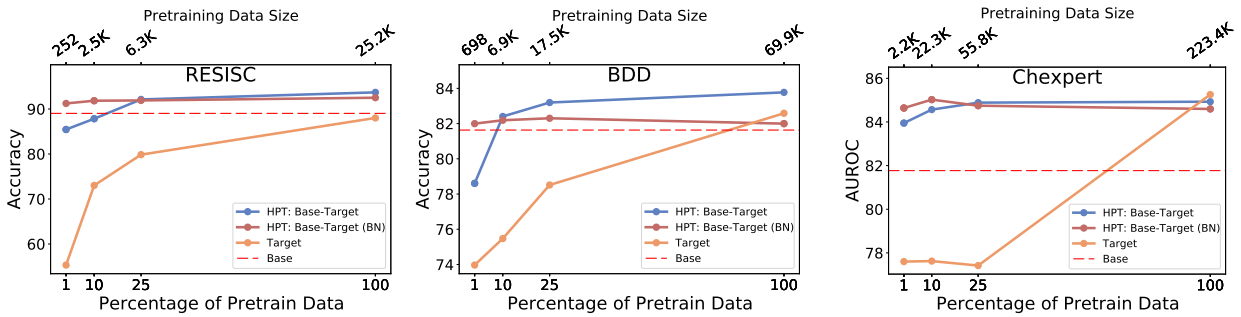

Figure 3.6: *HPT performance as the amount of pretraining data decreases*. The top axis shows the number of images, and the bottom shows the percentage of pretraining data. HPT outperforms Base model transfer or Target pretraining with limited data.

Key observations: Figure 3.5 shows separability results across datasets after sequentially removing augmentations. In all three data domains, HPT maintained strong performance compared to Target pretraining. Unlike BDD and RESISC, the Chexpert performance decreased as the augmentation policy changed. This illustrates that changes to the augmentation policy can still impact performance when using HPT, but that the overall performance is more robust. In turn, as a practitioner explores a new data domain or application, they can either use default augmentations directly or choose a conservative set, e.g. only cropping.

Pretraining data robustness: We pretrained with $\{1\%, 10\%, 25\%, 100\%\}$ of the target dataset. For HPT we used 5k pretraining steps. With 25% or 100% of the data, we used the same number of steps as the top performing result in Figure 3.2, and $1/10$ of the steps at 1% and 10%.

Key observations: Figure 3.6 shows separability results. CheXpert has 3x more training data than BDD, which in turn has 3x more training data than RESISC. While more data always performed better, the accuracy improvements of HPT increased as the amount of pretraining data decreased. HPT-BN had minimal degradation in low data regimes – outperforming other methods with $<$5k samples.

Domain Adaptation Case Study

In this section, we explore the utility of HPT through a realistic case study experiment in which we apply HPT in a domain adaptation context. The training procedure is as follows: we performed HPT to train a model using both source and target datasets on top of the standard MSRA ImageNet model [65]. We used this model to initialize the feature encoder in a Minimax Entropy (MME) Domain Adaptation (DA) method [143]. At the end of each budget level we evaluated accuracy on the entire test set from the target domain. We perform two experiments on DomainNet datasets [126] with 345 classes in 7 budget levels with increasing amount of target labels: (i) from `real` to `clip` and (ii) from `real` to `sketch` and use a `EfficientNet_B2` [150] backbone.

Table 3.3 shows the benefit of adding HPT to the DA experiments. From the results, we observe that HPT consistently outperforms the baseline on both domains by achieving a higher accuracy

Table 3.3: Budget levels and test accuracy in target domain for semi-supervised DA with and without HPT between `real`→`clip` and `real`→`sketch`.

Budget levels in target domains							
# of shots	1	11	16	22	32	46	68
Test accuracy (%) for `real`→`clip`							
MME	49.7	61.1	63.9	66.7	68.0	70.0	71.1
MME+HPT	**57.2**	**64.36**	**66.7**	**68.2**	**69.7**	**71.5**	**72.4**
Test accuracy (%) for `real`→`sketch`							
MME	41.4	51.8	54.9	57.5	59.7	61.4	62.5
MME+HPT	**50.2**	**56.4**	**58.8**	**60.7**	**62.8**	**63.9**	**64.9**

across all the budget levels. On the extreme low data regime (one shot/class), HPT achieves nearly 8% better accuracy in both `clipart` and `sketch` domains in the extreme case of providing one shot per class in the target domain. These results demonstrate HPT's effectiveness in a realistic, end-to-end inference system.

3.5 Discussion

We have shown that HPT achieves faster convergence, improved performance, and increased robustness across domains. Here, we further reflect on the utility of the HPT.

What is novel about HPT? The transfer learning methodology underlying HPT is well established in transfer learning. That is, transfer learning tends to work in a lot of situations, and our work could be perceived as a natural extension of this general observation. However, our work provides the first thorough empirical analysis of transfer learning applied to self-supervised pretraining in computer vision. We hope this analysis encourages practitioners to include an HPT baseline in their investigations – a baseline that is surprisingly absent from current works.

How should I use HPT in practice? We provide our code, documentation, and models to use HPT and reproduce our results (see appendix). For existing codebases, using HPT is usually as simple as starting training from an existing checkpoint. If working with a small dataset (e.g. $<$ 10k images), using HPT-BN is ideal.

Does this work for supervised learning? Yes. In the appendix, we reproduce many of these analyses using supervised ImageNet base models and show that HPT further improves performance across datasets and tasks.

3.6 Conclusion and Implications

Our work provides the first empirical analysis of transfer learning applied to self-supervised pretraining for computer vision tasks. In our experiments, we have observed that HPT resulted in 80x faster convergence, improved accuracy, and increased robustness for the pretraining process. These

results hold across data domains, including aerial, medical, autonomous driving, and simulation. Critically HPT requires fewer data and computational resources than prior methods, enabling wider adoption of self-supervised pretraining for real-world applications. Pragmatically, our results are easy to implement and use: we achieved strong results without optimizing hyperparameters or augmentation policies for each dataset. Taken together, HPT is a simple framework that improves self-supervised pretraining while decreasing resource requirements.

Chapter 4

Region Similarity Representation Learning

This chapter focuses on representation learning for localization-sensitive tasks, such as object detection and semantic segmentation. As I discuss, in several applications, localization-sensitive representation learning leads to improved performance with fewer labeled and unlabeled data.

4.1 Introduction

Recently, self-supervised pre-training has outperformed supervised pre-training for a number of computer vision applications such as image classification and object detection [14, 17, 22]. Much of this recent progress comes from exploiting the *instance discrimination* task [6, 40, 111, 167, 182], in which a network learns image-level features that are invariant to certain image augmentations. Specifically, instance discrimination maximizes the similarity of two *views* of an image, where each view is an augmented version of an image, while minimizing its similarity to views which originate from other images [14, 17, 19].

Chen et al. [17] compared a diverse set of possible augmentations, and found that random cropping and scaling have the largest impact on downstream ImageNet [34] classification performance. Several follow-up works have further explored augmentation policies and confirmed this finding [137, 156, 170]. Through cropping and scaling augmentations and similarity maximization, the network learns to map various scales and crops of an image to the same feature representations. For example, an image crop of the top half of a dog's body and the right half of a dog's body would map to the same representation in the embedding space, which a downstream task could then classify as "dog".

However, instance discrimination uses global image-level feature representations for these views, which is obtained by average pooling the final convolutional feature map. It does not enforce any type of spatial consistency in the convolutional features (see Figure 4.1, "instance discrimination" path). For example, different crops that scale and shift the dog's ear will not necessarily have a corresponding scale and shift in the convolutional feature maps throughout the network – instance discrimination only optimizes the final globally pooled features. This is problematic for downstream tasks such as object detection that leverage the spatial information from the convolutional feature

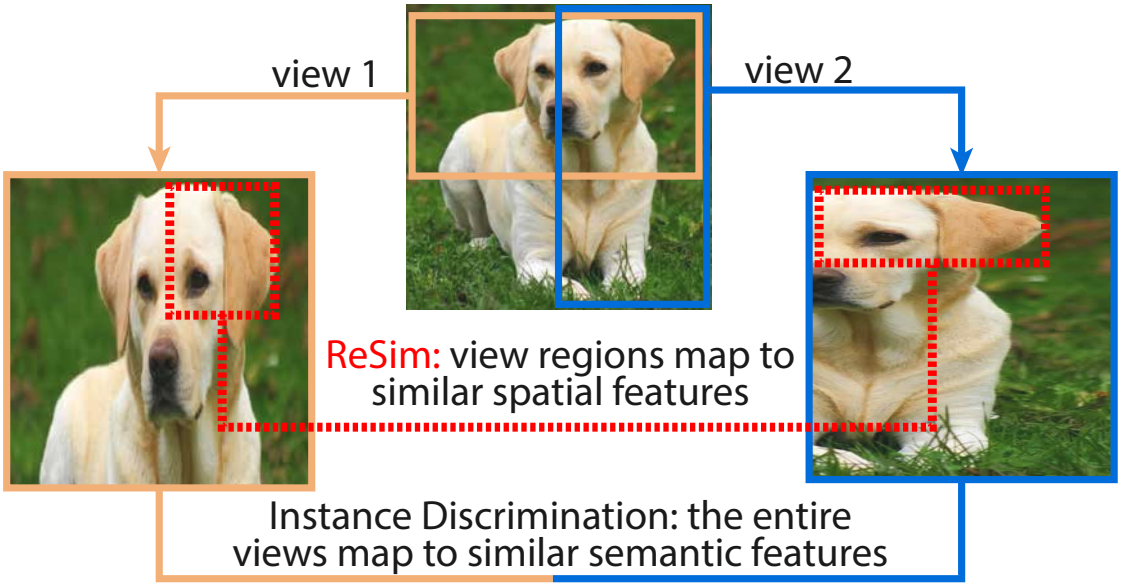

Figure 4.1: Existing instance discrimination-based self-supervised learning frameworks learn representations by augmenting an image into two different views (*e.g.*, cropping/scaling the input image) and then maximizing the similarity between the image features for the entire views. In this work, we present Region Similarity Representation Learning (ReSim), which learns representations by maximizing the similarity of corresponding sub-image *regions* throughout the convolutional layers of a network. In the above example, instance discrimination learns to map both views to the same features, despite the fact that the dog's eyes and ears are in different locations. On the other hand, ReSim learns features which explicitly align these changes with corresponding changes in the convolutional feature maps.

maps for object localization.

To address this issue, we introduce Region Similarity Representation Learning (ReSim): a self-supervised pre-training method which learns spatially consistent features across multiple convolutional layers. Inspired by the Region Proposal Network (RPN) used in Faster-RCNN [138], ReSim operates by sliding a fixed-size window across the overlapping region between two image views, mapping the corresponding regions in each view to their associated regions in the convolutions layers throughout the network, and then maximizing the similarity of these convolution feature regions, along with the global similarity objective. See Figure 4.1 for a high-level difference between ReSim and existing instance discrimination techniques, and see Figure 4.2 for a detailed description of the full ReSim pipeline.

As we show, maximizing the similarity of these convolutional feature map regions leads to representations that improve object localization for downstream detection and instance segmentation tasks. Furthermore, we extend the framework to learn features at various scales by using sliding windows of multiple sizes at different feature maps. We adopt the Feature Pyramid Network (FPN)

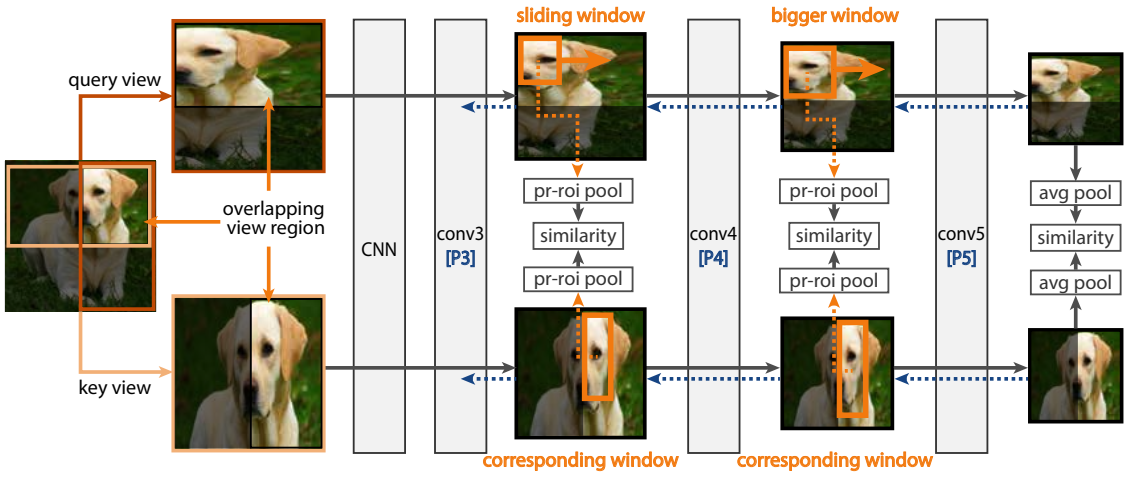

Figure 4.2: ReSim takes two different views of an image as input, *i.e.*, a *query* and *key* view obtained by cropping/scaling an image. The views have an associated overlapping area as highlighted in each of the above views. Both views are encoded using the same network (CNN), *e.g.*, a ResNet-50 [65]. Before the final convolutional layers in the network, ReSim slides a fixed-size window over the overlapping area between the two views, aligns window regions with corresponding regions in the convolutional feature maps using Precise RoI Pooling from [82], and then maximizes the similarity between these features. Earlier layers use smaller sliding windows as the feature maps have higher spatial resolution. Furthermore, similar to Feature Pyramid Networks [100], the feature maps are combined with semantic top-down features from later convolution layers, as indicated by the blue arrows and "[P3]/[P4]/[P5]" layers. The feature maps following the final convolutional layer are used for instance discrimination learning following either [22] or [19].

design from Lin et al. [100], a design which naturally incorporates feature hierarchies and propagates stronger semantic features to earlier convolutional layers through the top-down path. Region-level self-supervised similarity learning trains feature pyramid layers without labeled supervision and leads to further improvement on downstream tasks.

We conduct object detection, instance segmentation, and dense pose estimation experiments on PASCAL VOC [46], COCO [101], and Cityscapes [29] and show that ReSim learns representations which significantly improve the classification and localization performance compared to a MoCo-v2 baseline, *i.e.*, $+2.7$ $\text{AP}^{\text{bb}}_{75}$ on VOC, $+1.1$ $\text{AP}^{\text{bb}}_{75}$ on COCO, and $+1.9$ AP^{mk} on Cityscapes.

4.2 Related work

Self-supervised representation learning. The goal of representation learning is to reveal the intrinsic qualities of data in such a way that they are informative and effective for a desired task [8]. Practically, this often manifests as pre-training a deep network so that it can be finetuned for a particular downstream task, see [35, 39, 43, 57, 68, 73, 95, 132, 185]. Recently, SimCLR [17]

and MoCo [22, 68] demonstrated substantial improvements by using similar forms of instance contrastive learning where a network was trained to identify a pair of views originating from the same image when contrasted with a large set of views from other images. Following SimCLR and MoCo, later works, such as SwAV [14] and BYOL [60], reported substantial improvements for image classification tasks, but as several follow-up works have shown [36, 171, 177], SwAV and BYOL do not tend to lead to improvements on localization-based tasks such as object detection and segmentation. This decrease in performance indicated that strictly optimizing global, image-level representations could decrease performance for tasks which require localization.

In contrast to prior work that leveraged instance discrimination to learn global, image-level representations, we propose a region-based pretext task to learn representations for tasks which require both localization and semantic classification. In earlier work, several authors proposed representation learning at the pixel or region level via color prediction [159, 186], key-point encoding [153, 112], optical flow similarity [42], and cycle consistency or frame prediction in videos [97, 162]. Our work differs in that we build on instance discrimination pretext task and simultaneously learn semantic features from an entire image as well as regional features across multiple scales, shifts, and resolutions.

Object detection and instance segmentation. Object detection and segmentation localize and classify object bounding boxes or pixels within an image – see [103] for a survey and review. Many commonly used object detection and instance segmentation techniques such as Fast-RCNN [56], Faster-RCNN [138], and Mask-RCNN [66] operate through a two-stage process in which they extract region features from convolutional feature maps, regress the location of the region to align with a ground truth bounding box, and then classify the region.

When regressing the location of the regional features, small adjustments to the region location within the convolutional feature map can have a large impact on the classification, *e.g.*, shifting a region may cause it to overlap with different objects in the input pixel space. As a result, the regional features of the convolutional feature maps require spatial sensitivity for regression as well as semantic sensitivity for classification. Existing self-supervised methods have largely focused on learning strong semantic representations by average pooling the final convolutional layer, and they do not explicitly maintain spatial consistency throughout the convolutional layers. In this work, we propose a self-supervised learning method which learns both spatially consistent and semantically sensitive regional features.

Pixel and grid-based contrastive learning. Recently, several related papers emerged: [128] explored pixel-level contrastive learning, whereas our work examines region-level consistency across the convolutional features. Wang et al. [163] proposed contrastive learning using a grid of feature vectors over the feature maps. Rather than maximizing the similarity of the regions, their solution was more akin to clustering, in that they compared all combination of feature vectors from the feature grid and maximized the similarity of the most similar pair. As we show below, our method demonstrates superior performance over these models.

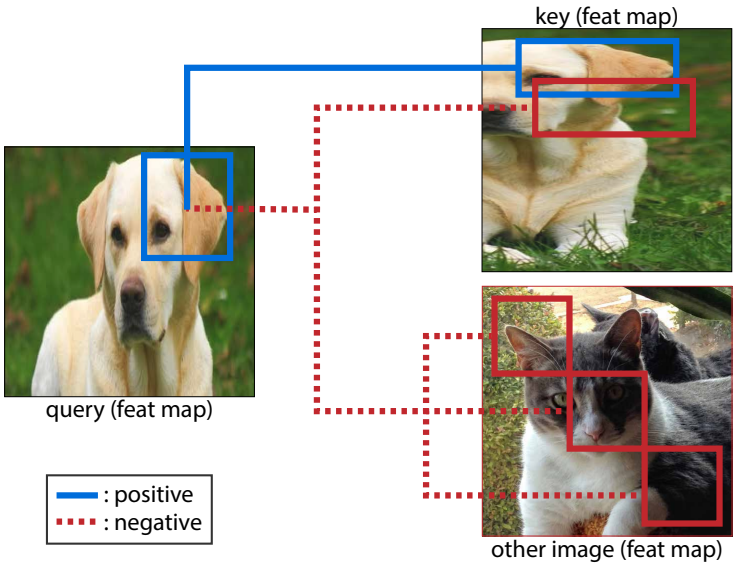

Figure 4.3: At each of the final convolutional layers, ReSim maximizes the similarity of aligned convolutional feature map regions across the query and key views of an image – the *positive* samples. ReSim simultaneously minimizes the similarity with a set of *negative* samples: the non-positive regions from the same key view (potentially overlapping with the positive region) and any other region from other images.

4.3 Region Similarity Representation Learning

This section presents ReSim: a self-supervised pre-training method which learns spatially consistent representations on feature pyramids.

Preliminaries

State-of-the-art self-supervised representation learning frameworks overwhelmingly exploit instance discrimination as their pretext task, see [80]. In instance discrimination, a reference image is augmented by a set of predefined augmentation modules $\mathcal{T} = [\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n]$, yielding two views: query and key. For instance, Figure 4.2 shows the query and key view after cropping and scaling augmentations. The features from the final convolutional layer from the two views are average-pooled into image features that are subsequently enforced to be similar in the embedding space by a learning objective, such as contrastive loss [17] or cosine similarity [19].

In the context of representation learning, an ideal encoder for localization-sensitive tasks should consistently encode the same image region, *e.g.*, the same object components across different views of an image, to the same point in the representational embedding space. This is necessary for tasks such as object detection with Faster-RCNN [138] or related detectors [66, 103], which localize and then classify regions within the convolutional feature maps. When regressing the location of the

regional features, small adjustments to the region within the convolutional feature map can have a large impact on the object classification prediction at the pixel level, *e.g.*, shifting a feature map region by a few pixels may cause the region to overlap with different objects in the input pixel space, changing the ground-truth classification and regression target.

Similarly, different image regions, *e.g.*, different object components across two different views of an image, should map to different points in the embedding space. By aligning similar image regions to the same representations and dissimilar image regions to dissimilar representations, downstream tasks can leverage the representations to localize and classify the underlying image components. As illustrated in Figure 4.2, the key and query views of the dog image correspond to the same image even though the two views are scaled and translated differently.

When learning representations for image classification, existing works map all regions of an image to similar representations [17, 22]. This occurs because the widely adopted set of augmentations from Chen et al. [17] include Random Resized Crop (RRC), which enforces that the query and key views always correspond to different regions in the original image. Therefore, we propose a region-level similarity learning component (Figure 4.2) which enforces that the same regions within the views encode to the same spatially and semantically consistent feature representations.

Framework Overview

The full ReSim framework, shown in Figure 4.2, operates as follows: First, ReSim augments an input image into two different views – a query and key, where each view is passed through a common encoder, *e.g.*, a ResNet-50. On a subset of the final convolutional layers, *e.g.*, `conv3` and `conv4`, ReSim slides a fixed-size window over the query view, but only within the overlapping region between the query and key view – this area is highlighted in Figure 4.2. Following a foundational idea from feature pyramids [100], earlier layers use a smaller window because they have a higher spatial resolution. ReSim then uses Precise RoI Pooling [82] to extract a feature vector from the associated feature map region for both views.

ReSim uses a similarity optimization function to maximize the similarity of the aligned feature vectors from the corresponding view regions. We chose a contrastive optimization function from [68], which also takes dissimilar regions from the same image as well as other images as negative examples, as illustrated in Figure 4.3. Note: we also explored a cosine similarity optimization function from [19] which does not require negative examples – see §4.4.

Finally, following the final convolutional layer, ReSim adopts the global average pool of the final feature map for an instance-level contrastive optimization from [68] or a cosine similarity optimization from [19]. As shown by the blue dotted arrows in Figure 4.2, ReSim uses 1x1 convolutional layers to propagate semantically strong features to the earlier convolutional layers; these are known as *top-down* connections in the original FPN work [100].

Region-level Similarity

We formulate a region as a rectangular sub-area of an image $\mathcal{I}$ described by its top, left, bottom, and right coordinates (t,l,b,r). An encoder f yields the representation of any region of a given image, *i.e.*, $f(\mathcal{I},(t,l,b,r))$. Figure 4.2 shows a CNN encoder taking two image regions as input. Our objective is to perform self-supervised pre-training of the encoder in such a way that the learned representations are tailored for downstream tasks such as object detection and instance segmentation.

Aligning corresponding regions. As shown in Figure 4.2, ReSim performs self-supervised similarity learning on sub-image level regions across two views of an image. Given two augmented views $\mathcal{I}_q$ and $\mathcal{I}_k$ from the same input image $\mathcal{I}$, and a region in the query view denoted by its coordinates (t_q,l_q,b_q,r_q), we need to find its corresponding region in the key view. From the set of widely adopted augmentations listed in the previous subsection, the only spatial transformations which affect the image coordinates are RRC and horizontal flips. We ensure that the query and key views are *either both horizontally flipped or both not flipped*, as otherwise they should have difference ground-truth regression term for downstream tasks (see object regression in [56]). Thus, it leaves us with RRC alone. Denote the operator as $\mathcal{R}$, the problem can be formulated as, given $\mathcal{I}_q = \mathcal{R}_q(\mathcal{I})$, $\mathcal{I}_k = \mathcal{R}_k(\mathcal{I})$ and (t_q,l_q,b_q,r_q), find $(t_k,l_k,b_k,r_k) = \mathcal{R}_{q\to k}(t_q,l_q,b_q,r_q)$. The solution is trivial as RRC is simple linear translations, therefore $\mathcal{R}_{q\to k} = \mathcal{R}_k(\mathcal{R}_q^{-1})$, where $\mathcal{R}^{-1}$ is the inverse translation of the applied spatial transformations.

Generating candidate regions and extracting features. ReSim generates candidate regions for the region-level contrastive learning by sliding a small window across the overlapping region between the query and key view – see Figure 4.2. The overlapping areas between the query and key views are selected as valid areas for the sliding window because they can be mapped between the views.

Rather than cropping regions from query/key images and feeding them into encoders, we encode an entire image and extract region features through Precise RoI Pooling [82] which takes in a rectangular window and a convolutional feature map, and yields features of the input window in the size of $h\times w\times c$, where we set $h=1$ and $w=1$ to create a feature vector from the region, and c is the same as the number of input feature map channels. Given a candidate region on the input views, ReSim performs region-level similarity across multiple convolutional feature maps at the end of the network, e.g., using $C3$ and $C4$ as shown in Figure 4.2.

Region similarity on feature pyramids. Lin et al. [100] proposed Feature Pyramid Networks (FPNs), an object detection architecture which combined the low-resolution, semantically strong features from later convolutional layers with high-resolution, semantically weak features from earlier layers via *top-down connections*. As shown by the blue dotted arrows in Figure 4.2, we add top-down connections to propagate semantically strong features to the earlier convolutional layers used for region similarity learning — we call this variant ReSim-FPN. We follow the proposed methodology from [100]: we use lateral convolutional connections to map the output of the convolutional layers to FPN layers (indicated with $\{P3,P4,P5\}$ in Figure 4.2), and then directly

add the top-down layers after using nearest neighbor sampling to match the spatial resolutions and 1×1 convolutional layers to match the channel dimensions.

Similarity learning objectives. Denote the 4-tuple of a region (t,l,b,r) as u, the set of valid region pairs in query and key from image $\mathcal{I}$ as $\{(u_q^1,u_k^1),(u_q^2,u_k^2),\ldots,(u_q^n,u_k^n)\}$, and the region-level similarity function

$$E_{i,j}^{\{k_+,k_-\}} = \exp\Big(f(\mathcal{I}_q,u_q^i)\cdot f(\mathcal{I}_{\{k_+,k_-\}},u_{\{k_+,k_-\}}^j)/\tau\Big), \tag{4.1}$$

where the positive sample $\mathcal{I}_q=\mathcal{T}_q(I)$, $\mathcal{I}_k=\mathcal{T}_k(I)$, $f(I,u)$ is the Precise RoI-pooled feature of region u from image $\mathcal{I}$, and the negative pairs can be *any* image and region pairs not identical to $\mathcal{I}_q$ and u_q – see Figure 4.3. τ is a temperature parameter to regularize the similarity distribution. We learn the region-level similarity for a query via the following objective:

$$\mathcal{L}_q^{\text{rs}} = \frac{1}{n}\sum_{i=1}^{n}\frac{E_{i,i}^{k_+}}{E_{i,i}^{k_+}+\sum_{j\neq i}E_{i,j}^{k_+}+\sum_{k_-,j}E_{i,j}^{k_-}}, \tag{4.2}$$

Denote the image-level similarity function with global average pooling (GAP) as

$$D^{\{k_+,k_-\}} = \exp\big(f(\mathcal{I}_q,\text{GAP})\cdot f(\mathcal{I}_{k\{+,-\}},\text{GAP})/\tau\big). \tag{4.3}$$

We use the global image-level similarity objective from [68]:

$$\mathcal{L}_q^{\text{is}} = \frac{D^{k_+}}{D^{k_+}+\sum_{k_-}D^{k_-}}. \tag{4.4}$$

We combine the region-level similarity and global image-level similarity objective via a weighting hyperparameter, λ, to yield the final objective:

$$\mathcal{L}_q = \mathcal{L}_q^{\text{rs}} + \lambda\mathcal{L}_q^{\text{is}}. \tag{4.5}$$

Framework Configurations

We use two 3×3 convolutions on $C4/P4$ or $P3$ to project the feature maps to 128 channels. The first convolution is followed by Batch Normalization and ReLU activation. The similarity boxes shown in Figure 4.2 compute the similarity between aligned convolutional feature map regions. For this similarity computation, we use a momentum contrastive similarity with the associated settings from [22] as default, while we also investigated a cosine similarity with settings from [19] – see §4.4.

For the region-based contrastive similarity, we take negative samples to be both the non-positive regions from the same key view and any other region from other images, see Figure 4.3. While MoCo uses a momentum-based queue to maintain a large number of negative image-level samples, we find that such a queue is unnecessary for region-level samples as there are a large number of

pre-train	AP	AP_{50}	AP_{75}
random init.	33.8	60.2	33.1
supervised	44.0	72.8	45.5
MoCo-v2	54.4	80.1	60.0
ReSim-C4	**55.9** (+1.5)	**81.3** (+1.2)	**62.1** (+2.1)

Table 4.1: **Comparisons of ReSim and MoCo-v2 [68] on PASCAL VOC object detection task.** The models are pre-trained on *IN-100*, the weights of which are transferred to a Faster R-CNN R50-C4 subsequently finetuned on VOC `trainval07+12`, and evaluated on `test2007`. AP_k is the average precision at k IoU threshold, and AP is the COCO-style metric which averages scores from `[0.5:0.95:0.05]`. In the brackets are the deltas to the MoCo-v2 baseline.

negative region samples within each batch, *i.e.*, we can generate over 30,000 negatives within a mini-batch of 256 images on $C4$ – see the appendix for more details.

We design two variations of ReSim: 1) ReSim-C4, which applies region-similarity learning on ResNet $C4$ feature map *without* FPN; and 2) ReSim-FPN, which applies region-similarity learning on FPN $P3$ and $P4$ feature maps. When applying the Precise RoI Pooling operator to obtain the features for the convolutional feature map regions, we apply the pooling operator on $C4$/$P4$ with a down-sampling rate of 16, and on feature map $P3$ with a down-sampling rate of 8. By default, we use a sliding window of 48 pixels with a stride of 32 pixels on the input image on $C4$/$P4$, and a sliding window of 32 pixels with a stride of 24 pixels on $P3$. We ablate these settings in § 4.4.

4.4 Experiments

We study the transfer capability of ReSim for localization-dependent downstream tasks. We perform *self-supervised pre-training* on two splits of the ImageNet dataset [34]: 1) 1000-category ImageNet (IN-1K), the standard ImageNet training set containing ∼1.25M images; and 2) 100-category ImageNet (IN-100), a subset of IN-1K split containing ∼125k images, following previous works [155, 170]. The pre-trained model is subsequently finetuned on PASCAL VOC [45] for object detection, COCO [101] for instance segmentation, Cityscapes [29] for instance segmentation, and DensePose [62] for dense pose estimation. For the ablation studies, we use IN-100 for pre-training and report full IN-1K pre-trained results for selected best models.

Training. Our hyperparameters closely follow the adopted self-supervised learning framework MoCo-v2 [22]. We use a ResNet-50 [65] backbone, pre-training for 200 epochs for IN-1K, 500 epochs for IN-100, with a mini-batch size of 256 on 8 GPUs. The initial learning rate is set as 0.03 and follows a cosine decaying schedule [108]. The weight decay is 0.0001, SGD momentum is 0.9, and the augmentations are the same as [22]. We use 1.0 for loss balancing term λ. We use a momentum queue of 65,536 for IN-1K experiments, and of 16,384 for IN-100 experiments. ReSim-FPN takes ∼62 hours to train 200 IN1K-epochs with 8 NVIDIA V100 GPUs, compared to

pre-train	AP	AP_{50}	AP_{75}
SimSiam	54.6	80.6	60.7
+ReSim-C4	55.7 (+1.1)	81.2 (+0.6)	61.9 (+1.2)
MoCo-v2	54.4	80.1	60.0
+ReSim-C4	**55.9** (+1.5)	**81.3** (+1.2)	**62.1** (+2.1)

Table 4.2: **Building ReSim on various self-supervised representation learning frameworks**, *i.e.*, SimSiam [19] and MoCo-v2. All models are obtained by self-supervised pre-training on IN-100, transferring weights to a Faster R-CNN R50-C4 detector subsequently finetuned on VOC `trainval07+12`, and evaluated on `test2007`. We use the prediction and projection heads as in [19] for ReSim on SimSiam. The improvement over SimSiam shows that ReSim is applicable to multiple frameworks.

$\sim$53 hours for MoCo. Note that the design of ReSim does *not* affect the complexity of its transferred downstream model.

IN-100 Study

We first experiment with various ReSim implementation decisions using IN-100, and from these results, we then study the performance of ReSim on IN-1k.

Setup. Throughout these experiments, we adopt the commonly used PASCAL VOC object detection task [47]. We use a Faster R-CNN [138] with R50-C4 [66] backbone. As in [68], we unfreeze all Batch Normalization layers and synchronizing their statistics across GPUs [125] during training. The short-side length of input images is randomly selected from [480, 800] pixels during training and fixed at 800 for inference. Training is performed on `trainval07+12` set ($\sim$16.5k images), and testing is performed on `test2007` set ($\sim$4.9k images), unless otherwise specified. The network is trained on 8 GPUs of mini-batch size 16. Training takes 24,000 iterations with a base learning rate 0.02. The learning rate is decreased by a factor of $1/10$ at the 18,000 and 22,000 iteration. All settings follow [68].

Comparisons with baselines. We first compare the ReSim framework with similarity learning on C4 (ReSim-C4) to the MoCo-v2 baseline. Since ReSim is built on MoCo-v2, the key difference between the two frameworks is the presence of similarity learning across the convolutional feature map regions in ReSim. Table 4.1 shows the VOC transfer results of the two frameworks, as well as the models which are supervise pre-trained on ImageNet and randomly initialized. Albeit competing against a strong reference, ReSim surpasses MoCo-v2 by a large margin of 1.5, 1.2 and 2.1 at AP, AP_{50}, and AP_{75}, respectively. The performance gap at high AP metric (AP_{75}) is larger than it at low AP metric (AP_{50}) which is less localization-dependent.

Self-supervised learning frameworks. We investigated ReSim with an additional self-supervised learning framework. Specifically, we selected the recently proposed Siamese-based framework, SimSiam [19], for its simplicity and effectiveness. We use identical designs of its projection and

pre-train	window	AP	AP_{50}	AP_{75}
MoCo-v2	N/A	54.4	80.1	60.0
ReSim-C4	W16-S16	55.0	80.7	61.3
ReSim-C4	W48-S16	55.8	81.2	61.6
ReSim-C4	W48-S32	55.9	**81.3**	62.1
ReSim-C4	W64-S48	**56.0**	81.0	**62.7**

(a) Full finetuning setting on VOC

pre-train	window	AP	AP_{50}	AP_{75}
MoCo-v2	N/A	47.1	75.4	50.6
ReSim-C4	W16-S16	48.7	76.0	53.2
ReSim-C4	W48-S16	49.5	76.7	54.1
ReSim-C4	W48-S32	**50.0**	**77.2**	**55.0**
ReSim-C4	W64-S48	49.7	**77.2**	54.3

(b) Frozen-backbone setting, finetuning on VOC

Table 4.3: **Comparisons of various sliding window sizes and strides under (a) full-finetuning and (b) frozen-backbone settings.** "Wl-Sk" denotes sliding a window of size $l \times l$ at stride k. ReSim-C4 by default uses W48-S32 setting. Inspired by the linear classification metric for classification, in the frozen-backbone setting the backbone of object detector is frozen from finetuning to directly evaluate the quality of features for object detection. The more significant improvement in the frozen-backbone setting implies ReSim learns features of better quality for localization-dependent task.

prediction heads (see their reference for details) for ReSim as the image-level similarity learning heads presented in SimSiam for consistency. Note that since those heads have several sequentially connected 2048 channel layers rather than a single 2048-128 channel connection used in MoCo-v2, the design significantly increases computational complexity over ReSim on MoCo, *i.e.*, ReSim-FPN based on SimSiam takes 46.7 hours to train on IN-100, compared to 15.5 hrs for ReSim-FPN based on MoCo.

Table 4.2 shows the results. ReSim built upon SimSiam improves the baseline by 1.1, 0.6 and 1.2 at AP, AP_{50}, and AP_{75}, respectively, and is comparable to ReSim built upon MoCo. Due to the extra pre-training time, we choose MoCo as our backbone framework.

Frozen backbone for finetuning on VOC. Linear classification, *i.e.*, training a linear classifier on a frozen backbone is widely adopted to evaluate the representations *for classification* [14, 17, 19, 68, 60, 167]. We design a similar feature evaluation experiment on object detection. After transferring the self-supervise pre-trained weights to a Faster R-CNN object detector, we freeze the backbone and finetune the RPN and object classifier head on labeled object data, as RPN is randomly initialized and object classifier is task-specific. For the Faster R-CNN R50-C4 detector, specifically, all weight layers from $C1$ to $C4$ feature map, along with statistics of Batch Normalization layers are frozen; weights in RPN and C5 (object classification head) are finetuned. According to Goyal et al. [59], this type of evaluation is ideal for representation learning because finetuning an entire network "evaluates not only the quality of the representations but also the initialization and optimization

pre-train	AP	AP_{50}	AP_{75}
random init.	33.8	60.2	33.1
supervised	53.5	81.3	58.8
Jigsaw [59]	48.9 (−4.6)	75.1 (−6.2)	52.9 (−5.9)
Rotation [59]	46.3 (−7.2)	72.5 (−8.8)	49.3 (−9.5)
NPID++ [113]	52.3 (−1.2)	79.1 (−2.2)	56.9 (−1.9)
SimCLR [17]	51.5 (−2.0)	79.4 (−1.9)	55.6 (−3.2)
PIRL [113]	54.0 (+0.5)	80.7 (−0.6)	59.7 (+0.9)
BoWNet [55]	55.8 (**+2.3**)	81.3 (+0.0)	61.1 (**+2.3**)
MoCo [68]	55.9 (**+2.4**)	81.5 (+0.2)	62.6 (**+3.8**)
MoCo-v2 [22]	57.0 (**+3.5**)	82.4 (**+1.1**)	63.6 (**+4.8**)
SwAV [14]	56.1 (**+2.6**)	82.6 (**+1.3**)	62.7 (**+3.9**)
DenseCL [163]	58.7 (**+5.2**)	82.8 (**+1.5**)	65.2 (**+6.4**)
ReSim-C4	58.7 (**+5.2**)	**83.1** (**+1.8**)	**66.3** (**+7.5**)
ReSim-FPN	**59.2** (**+5.7**)	82.9 (**+1.6**)	65.9 (**+7.1**)

Table 4.4: **Comparison with previous works on PASCAL VOC** `trainval07+12`, **evaluated on** `test2007`. Jigsaw and Rotation results are from [113]. SimCLR result is from [163]. Both ReSim-C4 and ReSim-FPN improve MoCo-v2 baseline, and achieve state-of-the-art over competitive concurrent work DenseCL [163]. In the brackets are the deltas to the ImageNet supervised pre-training baseline. **Green**: deltas at least +1.0.

method."

Table 4.3(b) shows the results. ReSim improves the baseline MoCo-v2 (default W48-S32) by 2.9, 1.8, 4.4 at AP, AP_{50}, and AP_{75}, respectively, a much larger margin compared to the full-finetune setting. The improvement is particularly significant at high IoU metric AP_{75}. It indicates ReSim yields features of higher quality for localization-dependent tasks, particularly when limiting the impact of the initialization and optimizations used during finetuning.

Region size and stride. We vary the size and stride of sliding windows in ReSim-C4 to determine the optimal combination. The results are reported in Table 4.3(a,b). "Wl-Sk" denotes sliding a window of size $l \times l$ at stride k. By considering the performance both at frozen-backbone and standard full-finetune settings, we choose W48-S32 setting as the default for ReSim-C4.

RoI Align vs. Precise RoI Pooling. We compare Precise RoI Pooling [82] and RoI Align [66] as the region feature extraction operator. On VOC, ReSim using RoI Align achieves AP/AP_{50}/AP_{75} of 55.7/81.0/62.0 with all parameters finetuned, and 48.9/76.6/53.0 with a frozen backbone. Comparing these results with Table 4.3 indicates that both RoI Align and Precise RoI Pooling work well, but in the frozen-backbone setting, RoI Align performs worse than Precise RoI Pooling. We conjecture this is because Precise RoI Pooling does not heuristically set the number of sampling points, enabling a more robust and simpler optimization.

pretrain	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init	31.0	49.5	33.2	28.5	46.8	30.4
supervised	38.9	59.6	42.7	35.4	56.5	38.1
MoCo-v2	38.9 (+0.0)	59.2 (−0.4)	42.4 (−0.3)	35.4 (+0.0)	56.2 (−0.3)	37.8 (−0.3)
VADeR [128]	39.2 (+0.3)	59.7 (+0.1)	42.7 (+0.0)	35.6 (+0.2)	56.7 (+0.2)	38.2 (+0.1)
DenseCL [163]†	39.4 (+0.5)	59.9 (+0.3)	42.7 (+0.0)	35.6 (+0.2)	56.7 (+0.2)	38.2 (+0.1)
ReSim-C4	39.3 (+0.4)	59.7 (+0.1)	43.1 (+0.4)	35.7 (+0.3)	56.7 (+0.2)	38.1 (+0.0)
ReSim-FPN	39.5 (+0.6)	59.9 (+0.3)	43.3 (+0.6)	35.8 (+0.4)	57.0 (+0.5)	38.4 (+0.3)
ReSim-FPN^T	**39.8** (+0.9)	**60.2** (+0.6)	**43.5** (+0.8)	**36.0** (+0.6)	**57.1** (+0.6)	**38.6** (+0.5)
ReSim-FPN^T (400 ep)	**40.3** (+1.4)	**60.6** (+1.0)	**44.2** (+1.5)	**36.4** (+1.0)	**57.5** (+1.0)	**38.9** (+0.8)

(a) Mask R-CNN R50-FPN, 1× schedule

AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
36.7	56.7	40.0	33.7	53.8	35.9
40.6	61.3	44.4	36.8	58.1	39.5
40.9 (+0.3)	61.5 (+0.2)	44.6 (+0.2)	37.0 (+0.2)	58.4 (+0.3)	39.6 (+0.1)
-	-	-	-	-	-
41.2 (+0.6)	61.9 (+0.6)	45.1 (+0.7)	37.3 (+0.5)	58.9 (+0.8)	40.1 (+0.6)
41.1 (+0.5)	61.5 (+0.2)	44.8 (+0.4)	37.1 (+0.3)	58.6 (+0.5)	39.8 (+0.3)
41.4 (+0.8)	61.8 (+0.5)	45.4 (+1.0)	**37.5** (+0.7)	**59.1** (+1.0)	**40.4** (+0.9)
41.4 (+0.8)	**61.9** (+0.6)	**45.4** (+1.0)	**37.5** (+0.7)	**59.1** (+1.0)	40.3 (+0.8)
41.9 (+1.3)	**62.4** (+1.1)	**45.9** (+1.5)	**37.9** (+1.1)	**59.4** (+1.3)	**40.6** (+1.1)

(b) Mask R-CNN R50-FPN, 2× schedule

Table 4.5: **Results on COCO object detection and instance segmentation tasks.** The models are pre-trained on IN-1K for 200 epochs (except the final entry, which shows extended performance out to 400 epochs), the weights of which are transferred to Mask R-CNN R50-FPN model, subsequently finetuned on `train2017` (1× and 2× schedules) and evaluated on `val2017`. Averaging precision on bounding-boxes (AP^{bb}) and masks (AP^{mk}) are used as benchmark metrics. All ReSim models outperform VADeR [128] and the MoCo-v2 counterpart, surpassing the supervised pre-training under all metrics (we report VADeR results from their paper which did not include a 2x schedule). In the brackets are the gaps to the ImageNet supervised pre-training baseline. **Green**: deltas at least +0.5. †: Pre-trained weights are downloaded from the official releases of the authors and trained on our COCO frameworks, as [163] adopted a different COCO finetuning baseline settings from the common approach used in this paper and other previous works [68].

IN-1K Results

Following the IN-100 experiments, we studied pre-training on the full IN-1K and report the results on PASCAL VOC, Cityscapes, COCO detection and COCO dense pose estimation. We evaluate two models as introduced in §4.3: 1) ReSim-C4, which performs region-level similarity on $C4$ feature map of ResNet; and 2) ReSim-FPN, which performs region-level similarity on the $P3$ and $P4$ feature maps on the top-down path of FPN. Only the backbone ResNet weights are transferred for finetuning downstream tasks. On the other hand, *if the network of a downstream task adopts the FPN design*, then we can transfer the FPN weights from ReSim-FPN to the new network. We term this as ReSim-FPN^T, and report the results on selected downstream tasks which use FPN as its baseline.

PASCAL VOC. We use the Faster R-CNN R50-C4 detector on VOC and adopt the same setup for PASCAL VOC as described in §4.4. We first report the results of finetuning on `trainval07+12` and evaluating on `test2007`. Table 4.4 compares our method with a series of previous state-of-the-arts. Respectively, ReSim-C4 and ReSim-FPN improve over their baseline MoCo-v2 by 1.7/0.7/2.7 and 2.2/0.5/2.3 at AP/AP_{50}/AP_{75}, and claims state-of-the-art results over the competitive concurrent work DenseCL [163].

In Figure 4.4 we show the AP improvements of ReSim over MoCo-v2 at IoU threshold from 0.5 to 0.9. Not only do both ReSim methods outperform MoCo-v2, the larger improvement at high IoU indicates superior localization accuracy, particular for ReSim-FPN. The growing performance gap

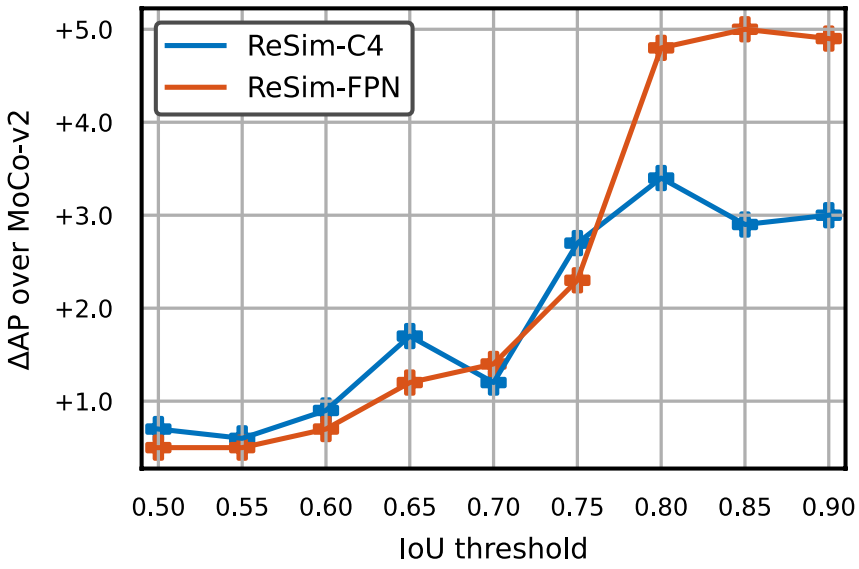

Figure 4.4: **ΔAP of ReSim over MoCo-v2 at various IoU thresholds on PASCAL VOC.** As the IoU threshold increases, both ReSim methods perform considerably better than MoCo-v2, especially at higher IoU threshold, indicating that the ReSim features have better localization capability.

as the IoU threshold increases indicates that ReSim has led to improved localization of the objects.

We also report the results of finetuning on PASCAL VOC `trainval2007` and evaluating on `test2007`. ReSim-C4 achieves 48.9/75.4/53.5 at AP/AP_{50}/AP_{75}, in comparison with 46.6/72.8/50.9 for MoCo-v2.

COCO Object Detection and Instance Segmentation. We use Mask-RCNN [66] with R50-FPN [100] as our base model and add new Batch Normalization layers before the FPN parameters. All Batch Normalization statistics are synchronized across GPUs. The short-side length of input images is randomly selected from [640, 800] pixels during training and fixed at 800 for inference. Training is performed on `train2017` split with ∼118k images, and testing is performed on `val2017` split. Two metrics are used, Average Precision on bounding-boxes (AP^{bb}), and Average Precision on masks (AP^{mk}). We report finetuning results on the standard 1x and 2x schedules, the latter of which trains twice as long as the former.

Table 4.5 shows the results. All variations of ReSim improve over its MoCo-v2 counterpart under both box and mask metrics, outperforming the supervised pre-training baseline. Unlike MoCo-v2, ReSim improves with the 1× schedule. We note that ReSim-FPN outperforms ReSim-C4, and in turn ReSim-FPN^T, in which FPN weights are transferred, outperforms ReSim-FPN, demonstrates the effectiveness of our framework on feature pyramids. Furthermore, the final row in Table 4.5 shows a continued increase in performance after 400 epochs of pre-training, indicating that ReSim continues to its representations in longer pre-training regimes.

COCO DensePose Estimation. We use DensePose R-CNN [62] with R50-FPN as our base model. As in COCO object detection and instance segmentation, we add Batch Normalization layers in

pre-train	COCO dense pose estimation AP^{gpsm}	AP^{gpsm}_{50}	AP^{gpsm}_{75}
supervised	65.8	92.6	77.8
MoCo-v2	66.2(+0.4)	93.1(+0.5)	77.4(−0.4)
ReSim-C4	66.6(+0.8)	92.9(+0.3)	78.5(+0.7)
ReSim-FPN	**67.1**(+1.3)	**93.2**(+0.6)	**80.0**(+2.2)
ReSim-FPN^T	66.8(+1.0)	92.8(+0.2)	79.2(+1.4)

pre-train	Cityscapes Instance Seg. AP^{mk}	AP^{mk}_{50}
supervised	33.0	59.8
MoCo-v2	33.7(+0.7)	61.8(+2.0)
ReSim-C4	35.4(+2.4)	63.1(+3.3)
ReSim-FPN	**35.6**(+2.6)	**63.4**(+3.6)
ReSim-FPN^T	35.4(+2.4)	63.3(+3.5)

Table 4.6: **Results on COCO dense pose estimation and Cityscapes instance segmentation tasks.** The performance deltas with the supervised baseline is shown in the brackets. **Green**: deltas at least +0.5.

FPN, and finetune and sync all such layers during training. We use the DensePose implementation in [165]. The model is trained on `train2014` split (1× schedule) and evaluated on `val2014`. We report results under masked-GPS (AP^{gpsm}). Table 4.6 upper shows the comparisons of ReSim versus MoCo-v2 and supervised pre-training counterparts. Our ReSim-FPN improves over the supervised baseline by 2.2 points under AP^{gpsm}_{75}.

Cityscapes Instance Segmentation. We use Mask R-CNN [66] with R50-FPN as our base model, add Batch Normalization layers before the FPN, and finetune and sync all such layers during training. The model is trained on the standard splits and training settings from [165]. Table 4.6 lower shows the comparisons of ReSim versus MoCo-v2 and supervised pre-training counterparts. All of the ReSim variants substantially improve over the supervised baseline, *e.g.*, by 3.3-3.6 points under AP^{mk}_{50}.

4.5 Conclusion

We presented ReSim, a novel self-supervised representation learning algorithm for localization-based tasks such as object detection and segmentation. ReSim performs both global and region-level contrastive learning to simultaneously learn semantic and spatial feature representations. The spatially consistent feature representations are learned on regions of the convolutional feature maps, while the global representations are learned over the entire image. In order to learn hierarchical representations, we further incorporated feature pyramids into the ReSim pre-training, and showed how these parameters can also be transferred to popular FPN-based frameworks for downstream

tasks. Our method shows significant improvements for various localization-based tasks, such as object detection, instance segmentation, and dense pose estimation.

Chapter 5

Scale-MAE: A Scale-Aware Masked Autoencoder for Multiscale Geospatial Representation Learning

Continuing with the notion of data and label efficiency, this chapter explores a key missing component for data and label efficient representation learning for geospatial imagery. Similar to the prior chapters, this chapter will explore the necessity of label-efficient representation learning, however, in this case we focus on a key area for scientific and industrial applications: geospatial imagery.

5.1 Introduction

Remote sensing data is captured from satellites and planes through a mixture of sensors, processing pipelines, and viewing geometries. Depending on the composition and relative geometry of the sensor to the Earth, each image's Ground Sample Distance (GSD - the physical distance between two adjacent pixels in an image) can vary from 0.3m to 1km, so a 100x100 pixel image could span anywhere from an Olympic-size swimming pool (900 m^2) to almost the entire country of Jamaica (10,000 km^2). The data within each image, and the corresponding objects and points of interest, can therefore vary across wide spatial ranges. Data from these multiscale sensors provide critical and complementary information for various operational and research applications in areas such as atmospheric, hydrologic, agricultural, and environmental monitoring [152, 114].

Few modern computer vision methods have explicitly addressed multiscale remote sensing imagery [90]. Nevertheless, the remote sensing vision community has increasingly used large, pretrained models [50, 28], where such applications finetune a pretrained model for a single source of data at a specific scale [50, 28, 51, 78, 106]. In this chapter we present *Scale-MAE*, a masked reconstruction model that explicitly learns relationships between data at different, known scales throughout the pretraining process. By leveraging this information, *Scale-MAE* produces a pretrained model that performs better across a wide range of GSDs and tasks.

Scale-MAE is a self-supervised pretraining framework based on the Masked Autoencoder

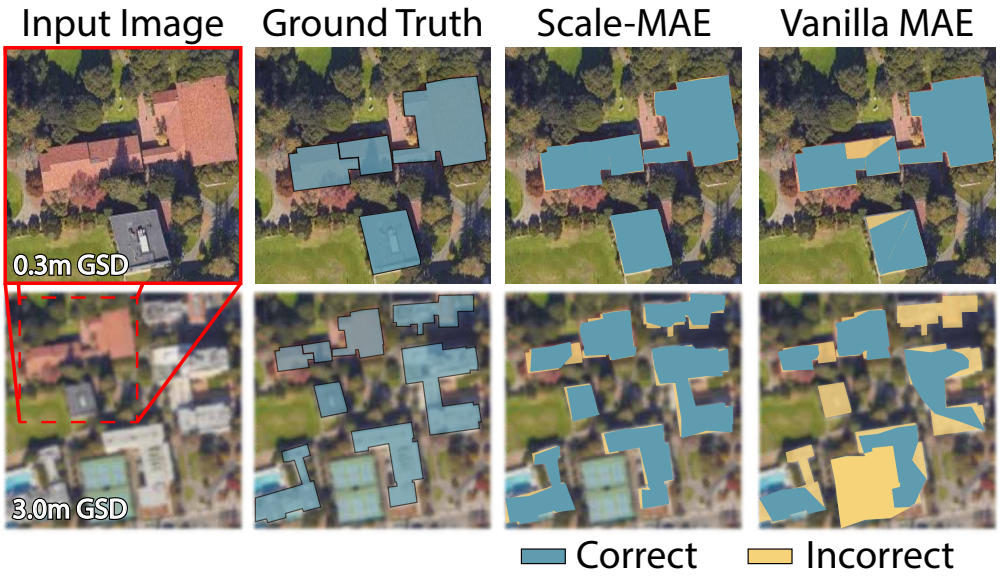

Figure 5.1: ***Scale-MAE* learns better representations for multiscale tasks compared to vanilla MAE.** (Column 1) The top image spans an area at 0.3m GSD and the bottom image shows the same region at a coarser GSD. (Columns 2-4) The following columns show a ground truth building segmentation, *Scale-MAE* segmentation from a finetuned UperNet, and segmentation from an analogously finetuned UperNet from a vanilla MAE, respectively. *Scale-MAE* demonstrates better performance across images at both scales. See the supplementary material for more examples.

(MAE) [67]. A standard MAE resizes/crops an image, masks the majority of the transformed image, and then tasks a Vision Transformer (ViT) based autoencoder with embedding the unmasked components. A decoding ViT then decodes the full image from these learned embeddings, where the decoder is later discarded and the encoder is used to produce representations for an unmasked input image.

Scale-MAE has two key differences that lead to its strong performance across spatial scales (Figure 5.1). First, a standard MAE uses relative positional encodings to inform the ViT of the position of the unmasked components. *Scale-MAE* uses a GSD-based positional encoding derived from the land area covered in the image which informs the ViT of both the position *and scale* of the input image. Second, *Scale-MAE* uses a Laplacian-pyramid decoder to encourage the network to learn multiscale representations. The embeddings are decoded to two images containing low and residual high frequency information, respectively – see Figure 5.2. As we discuss in Section 5.3, this structure allows the ViT decoder to use fewer parameters than MAE while still producing strong representations across multiple scales.

We show that *Scale-MAE* leads to better performing, more robust multiscale representations than both a standard MAE and a recently proposed, state-of-the-art remote sensing MAE called SatMAE [28] across remote sensing datasets with a variety of scale and resolution characteristics. Our contributions can be summarized as follows:

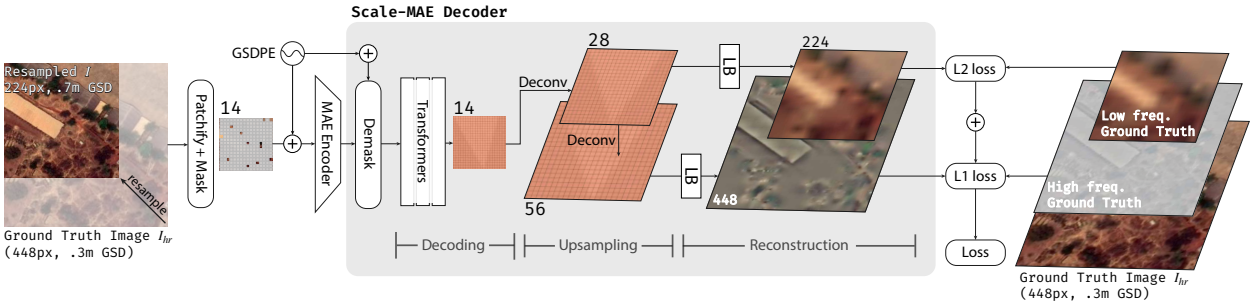

Figure 5.2: ***Scale-MAE* architecture.** Following the Masked Autoencoder framework, an input image is patchified and masked before being passed into an MAE encoder. A Ground Sample Distance Positional Encoding (GSDPE) is added to the encoder input, which scales the positional encodings to the area of ground covered. The ReSim decoders has three stages: (1) Decoding, which uses a smaller number of transformer layers than MAE to decode the encoded values (2) Upsampling, which progressively deconvolves the decoded feature map to a larger size before being passed through the Laplacian Blocks (abbreviated LB, see Section 5.3), (3) Reconstruction, which then reconstructs low and high frequency features at different scales. These outputs are used to compute an aggregate loss with ground truth low and high frequency features, where following super resolution literature [2], an L1 loss is used for high frequency output to better reconstruct edges and an L2 loss is used for low frequency output to better reconstruct average values.

1. We present *Scale-MAE*, a pretraining method that leads to better representations for remote sensing data that are more robust across a range of spatial scales. *Scale-MAE* has two key contributions: a scale-aware positional encoding and a Laplacian pyramid decoder that decodes low and high frequency features at multiple scales.

2. *Scale-MAE* achieves an average of a 5.0% nonparametric kNN classification improvement across eight remote sensing datasets compared to current state-of-the-art.

3. *Scale-MAE* obtains a 0.9 mIoU to 3.8 mIoU improvement on the SpaceNet building segmentation transfer task for a range of evaluation scales (see Figure 5.1).

5.2 Related Work

Representation learning and the Masked Autoencoder. Representation learning aims to extract meaningful, intrinsic features from data for downstream use [8]. In practice, this often entails pretraining a deep network so that a lightweight learning routine can then finetune it for a particular downstream task, see [35, 39, 43, 57, 69, 72, 95, 132, 185]. The Masked Autoencoder (MAE) is a recent state-of-the-art self-supervised representation learning method in computer vision that pretrains a ViT encoder by masking an image, feeding the unmasked portion into a transformer-based encoder, and then tasking the decoder with reconstructing the input image [67]. *Scale-MAE*

builds upon the MAE and learns representations specifically designed to be robust with multiscale remote sensing imagery.

Remote Sensing Representation Learning Neumann *et al.* [116] were one of the first to exhaustively share results on existing representation learning and semi-supervised learning techniques for remote sensing imagery. Gao *et al.* [51] demonstrated the effectiveness of MAE pretraining for remote sensing image classification. Ayush *et al.* [4] leveraged the metadata from remote sensing images via spatially aligned but temporally separated images as positive pairs for contrastive learning and predicted the latitude and longitude as pretext tasks. Gupta *et al.* [63] demonstrated the use of MAEs as a pretraining approach for passive and active remote sensing imagery. Their method introduced flexible "adapters" which could be used interchangeably with an encoder for a set of input imagery modes. Cong *et al.* [28] introduced the SatMAE, which used temporal and spectral metadata in a positional encoding to encode spatio-temporal relationships in data. The temporal data contains the year, month, and hour enabling understanding of long-term change with the year, weather information from the month, and hour information for the time of day. Further Liu *et al.* [106] and Ibañez *et al.* [78] have shown that MAE architectures can be used for band selection in hyperspectral remote sensing images, significantly reducing data redundancy while maintaining high classification accuracy. *Scale-MAE* leverages inherent absolute scale information present in remote sensing data as a way to learn robust, multiscale features that reduce data usage downstream.

Super-resolution Super-resolution has proven effective in improving accuracy within remote sensing images due to the extremely small size of objects within the image [144]. Previous works have aimed to learn continuous implicit representations for images at arbitrary resolutions to aid the super-resolution task. These representations are used to upsample the images either to specific scales [96] or to arbitrary resolutions [173, 23, 75]. Most super-resolution work aims to increase the resolution of the input image, whereas *Scale-MAE* produces both higher *and lower* resolution images. There is some work on super-resolution for satellite imagery, but much of this work is focused on synthetically creating high-resolution datasets for use with models trained specifically for high-resolution data [70, 90]. *Scale-MAE*, however, utilizes super-resolution as a means to obtain multiscale representations during pretraining.

Features at multiple scales. Because images can contain objects of many different pixel resolutions, the vision community has proposed many methods to extract multiscale features. These include spatial pyramids [94, 12, 141, 85] and dense sampling of windows [175, 81, 176]. These approaches have been combined by methods such as [49], in which dense histogram-of-gradient features are computed for each feature pyramid level. Rather than using classical computer vision techniques to extract multiscale features, convolutional neural networks have been used to build deep multiscale features. CNNs with subsampling layers inherently build feature pyramids, a property exploited explicitly by [104, 99, 52]. Recently, this multiscale idea has been extended to vision transformers by [48], who show that this architecture improves various video recognition and image classification tasks. *Scale-MAE* reconstructs a version of Laplacian pyramids as a way to force an encoder to learn multi-scale features.

5.3 Scale-MAE

This section describes the *Scale-MAE* pretraining framework as illustrated in Figure 5.2. *Scale-MAE* is a self-supervised pretraining framework based on the Masked Autoencoder (MAE) [67]. Our method has two key differences: first, a standard MAE uses relative positional encodings to inform the ViT of the position of the unmasked components, where an image at resolution r will have the same positional encodings regardless of the image content. *Scale-MAE*, however, uses a Ground Sample Distance (GSD) based positional encoding that scales in proportion to the area of land in an image, regardless of the resolution of the image. Second, to encourage the network to learn multiscale representations, *Scale-MAE* uses a Laplacian-pyramid decoder, where the embeddings are decoded to a lower resolution image that captures the lower frequency information and a higher resolution image that captures the high-frequency information. We formalize *Scale-MAE* in the following subsections by first specifying the necessary MAE background, describing the GSD-based positional encoding, and then explaining the Laplacian-pyramid decoder.

Setup Let $I \in \mathbb{R}^{H \times W \times C}$ represent an input image of height H, width W, and C channels. The MAE *patchifies* I into a sequence S of independent patches of height and width P pixels, where each of the N_p patches, $s \in S$ has dimension $s \in \mathbb{R}^{P^2C}$. A fraction, m, of the patches are then removed and the remaining patches are then passed through a projection function (e.g., a linear layer) to project the patches S into D dimensions, $f_E : \mathbb{R}^{P^2C} \to \mathbb{R}^D$, to obtain embedded patches $S_E = f_E(S)$. An $\mathbb{R}^2$ positional encoding vector, is then added to the embedded patches with

$$v_x(pos, 2i) = \sin \frac{pos}{10000^{\frac{2i}{D}}} \tag{5.1}$$

$$v_y(pos, 2i+1) = \cos \frac{pos}{10000^{\frac{2i}{D}}} \tag{5.2}$$

where pos is the position of the patch along the given axis and i is the feature index (visualized in Figure 5.3), exactly as introduced in [158]. These positional encodings are then concatenated and added to the embedded patches, which are then fed into a ViT encoder. After the encoder, the removed m patches are then placed back into their original location in the sequence of patches where a *learned mask token* represents the masked patches that were not encoded. Another positional encoding vector is added to all patches and a sequence of transformer blocks decodes these patches to form the original input image, which is used as the learning target.

Input Unlike MAE, *Scale-MAE* performs a super resolution reconstruction, where the input image I is downsampled from a higher resolution image I_{hr} at the ground truth GSD. Instead of targeting the input image, *Scale-MAE* targets high frequency and low frequency components of I_{hr}, which is common in Laplacian pyramid super resolution models [178], where the high frequency component is at the same resolution as the ground truth image I_{hr} and the low frequency component is at the same resolution as the input image I, as shown in Figure 5.2. Following many works in super resolution[178], the low frequency target image is obtained by interpolating I_{hr} to a much lower resolution, r_{low} and then interpolating to the same resolution as the input image I. The high frequency target image is obtained by downsampling I_{hr} to another lower

resolution $r_{\text{high-low}}$, and then upsampling to the same resolution as the ground truth image I_{hr} and subtracting this image $I_{\text{hf}} = I_{\text{hr}} - I_{\text{high-low}}$. The supplementary material provide more information on the upsampling/downsampling methodology. The key components for *Scale-MAE* are described next.

GSD Positional Encoding Remote sensing images inherently have the concept of GSD, a metric which defines the absolute scale for the image. GSD is critical to understanding, conceptually, the kinds of features that will be available in an image. An image with finer GSD (lower number) will have higher frequency details than an image with coarser GSD (high number). Models are generally unaware of absolute scale when learning over a set of data. Specifically, even if they implicitly learn that all images in a dataset share a varying resolution from input-space augmentations, then these models do not explicitly condition on the GSDs encountered in unseen data.

We extend the positional encoding from Equation (5.2) to include GSD by scaling the positional encoding relative to the land area covered in an image as depicted in Figure 5.3 and mathematically:

$$v_{gsd,x}(pos, 2i) = \sin \frac{G}{g} \frac{pos}{10000^{\frac{2i}{D}}} \tag{5.3}$$

$$v_{gsd,y}(pos, 2i+1) = \cos \frac{G}{g} \frac{pos}{10000^{\frac{2i}{D}}} \tag{5.4}$$

where g is the GSD of the image and G is a reference GSD, nominally set to 1m. Intuitively, an object imaged at a finer resolution has more pixels representing it. When imaging the same object at a coarser resolution, those pixels must map to fewer pixels. In Equation (5.4), we interpolate the positional encoding by a factor of $\frac{G}{g}$ to account for the ordering of the coarser set of pixels. This simple idea underpins the GSD Positional Encoding, visualized in Figure 5.3.

Scale-MAE decoder The standard MAE learns representations by tasking a network with reconstructing an image after masking out most of its pixels. While the standard MAE decoder reconstructs the input image at the same scale as its input, the objective of *Scale-MAE* is to learn multiscale representations. We draw on works from progressive super-resolution such as [164], that learn a high resolution, high frequency image and a lower resolution low frequency image, that when combined together, yield the input image at a higher resolution.

The standard MAE decoder reconstructs to the input image scale by using a series of transformer layers. The *Scale-MAE* decoder, on the other hand, decodes to multiple scales by replacing the majority of the MAE decoder's transformer layers with a progressive Laplacian decoder architecture. This architecture consists of three stages: decoding, upsampling, and reconstruction, which are shown in Figure 5.2 and detailed below.

Decoding follows the standard MAE decoder where following the encoder, the removed m patches are then placed back into their original location in the sequence of patches where a *learned mask token* represents the masked patches that were not encoded, a positional encoding is added, and then a series of transformer layers decode all patches. In contrast to the standard MAE decoder, the *Scale-MAE* decoder uses fewer transformer layers (e.g. 3 layers instead of 8), which more than offsets the parameter complexity as quantified in Section 5.5. The output of these layers is then fed into the upsampling stage.

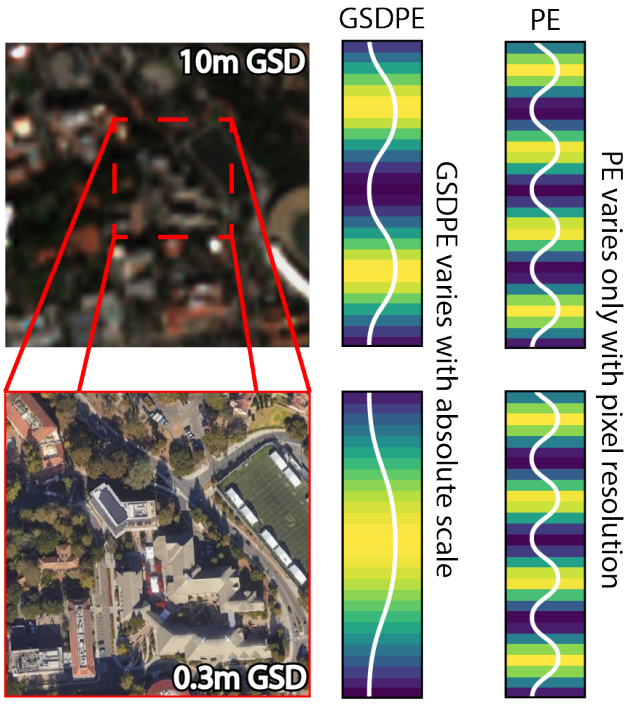

Figure 5.3: **The difference between GSDPE and a standard Positional Encoding (PE).** (Left) Input images at the same pixel resolution but different GSDs are shown. The image on the top is a subset of the image on the bottom. (Center) This overlap in location, albeit at a different resolution, is reflected in the GSDPE. The finer image with smaller spatial extent is represented by a corresponding subsection of the overall sine wave on the bottom. (Right) A standard positional encoding is strictly dependent on the image resolution and uses the same embedding for both. The colors behind the sine waves show the intensity and quantization of the encoding.

Upsampling The latent feature maps from the decoding stage are progressively upsampled to 2x and 4x resolution using deconvolution blocks, where the first deconvolution is 2x2 with stride 2 that outputs a feature map at 2x the input resolution (28 in Figure 5.2), followed by a LayerNorm and GELU, and then another 2x2 deconvolution layer that outputs a feature maps at 2x the previous resolution (56 in Figure 5.2). See the supplementary material for a full architectural diagram.

Reconstruction After having been upsampled, the lower resolution and higher resolution feature maps are passed into *Laplacian Blocks* (LBs in Figure 5.2) that reconstruct high and low resolution images for the high and low frequency reconstruction, respectively. Architecturally, the Laplacian Blocks consist of a sequence of three sub-blocks: a Laplacian Feature Mapping Block, a Laplacian Upsample Block, and a Laplacian Pyramid Reconstruction Block. The Feature Mapping Block is used to project features within a particular layer of the Laplacian Pyramid back to the RGB space. The Laplacian Upsample Block represents a learnable upsample function that maps latent features from one layer of the Laplacian Pyramid to a higher level. Finally, the Laplacian Pyramid Reconstruction Block is used to reconstruct information at the different frequencies in RGB space.

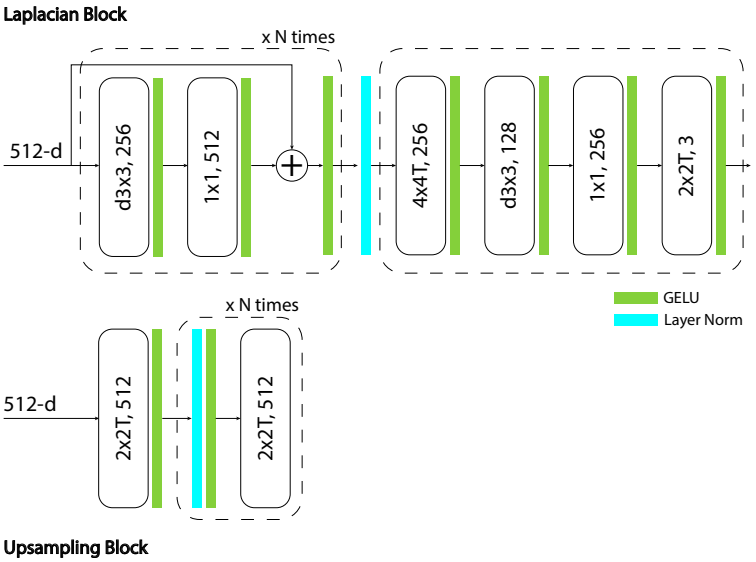

Figure 5.4: (top) The Laplacian Block (LB) is a fully convolutional architecture consists of a chain of Feature Mapping Block followed by one final Reconstruction Block. (bottom) The UpSampling Block (UB) consists of a series of transpose convolution layers separated by LayerNorm and GELU activation.

Following super resolution literature [2], an L1 loss is used for high frequency output to better reconstruct edges and an L2 loss is used for low frequency output to better reconstruct average values.

Figure 5.4 illustrates the architecture of Laplacian and Upsampling block architectures described below.

Laplacian Block Details

Laplacian Blocks are used to reconstruct the target at a specific resolution and frequency. A Laplacian Block consists of a chain of Feature Mapping Block, which distills information at a specific frequency, followed by one final Reconstruction Block, which generates the final output. A Feature Mapping Block consists of a 3x3 depth-wise convolution layer with GELU activation, followed by 1x1 convolution. A Reconstruction Block consists of a 4x4 transpose convolution layer followed by a 3x3 depth-wise convolution layer, a 1x1 convolution layer, and a 2x2 transpose convolution layer. In our experiments, we have two Feature Mapping Blocks per Laplacian Block.

Upsampling Blocks are used to upsample the feature map to a higher resolution. It consists of a series of 2x2 transpose convolution layers with LayerNorm and GELU activation between them. The number of such transposed convolution layers are a function of the output and input resolution. This is a progressive process in which we repetitively upsample the feature map by a factor of 2 until we reach the desired target resolution. Figure 5.4 illustrates the architecture of these two blocks.

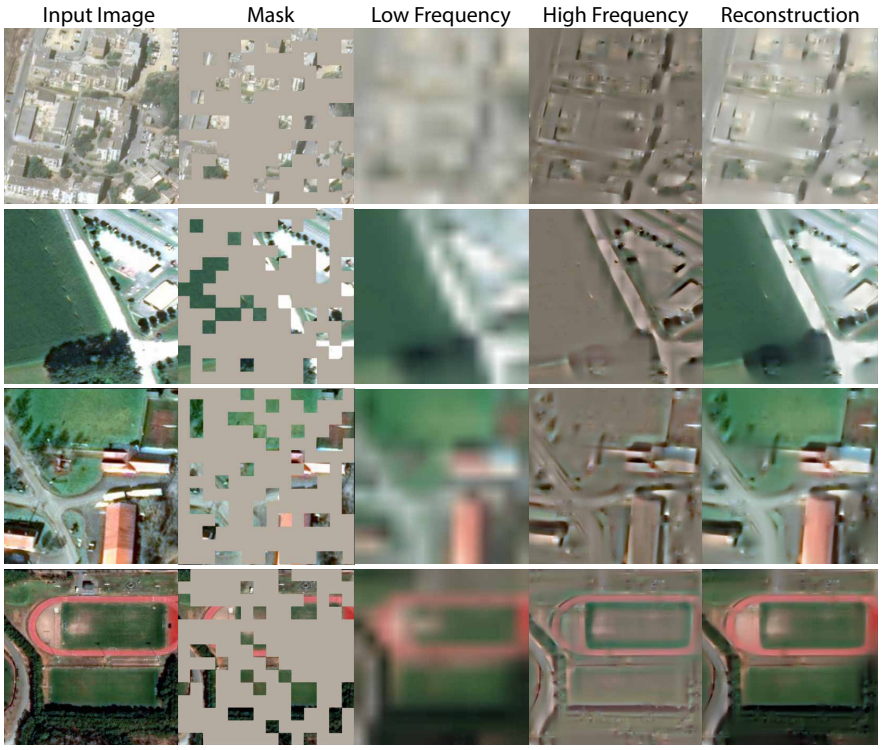

Figure 5.5: ***Scale-MAE* reconstruction.** Examples from Functional Map of the World are shown. From left to right, an input image at 224x224 resolution is shown. Its corresponding mask is visualized as well. Columns 3 and 4 show the low and high frequency produced by the *Scale-MAE* decoder. The last column is the reconstruction obtained from summing the low and high frequency features together.

5.4 Experiments

We investigate the quality of representations learned from *Scale-MAE* pretraining through a set of experiments that explore their robustness to scale as well as their transfer performance to additional tasks. First, we present our main experiments in Section 5.4 and compare with SatMAE [28], a current state-of-the-art MAE for remote sensing imagery, as well as several other approaches detailed throughout. The exact implementation of *Scale-MAE* for the main experiments was determined through a set of ablation experiments presented in Section 5.4.

Experimental Datasets

In our experiments, we used a total of ten datasets (Table 5.1) for the tasks of land-use/land-cover classification and semantic segmentation. There are a large amount of remote sensing datasets in existence. Many remote sensing datasets fundamentally capture the same data with minor changes in location or distribution. We selected datasets with key, representative *properties*. These properties

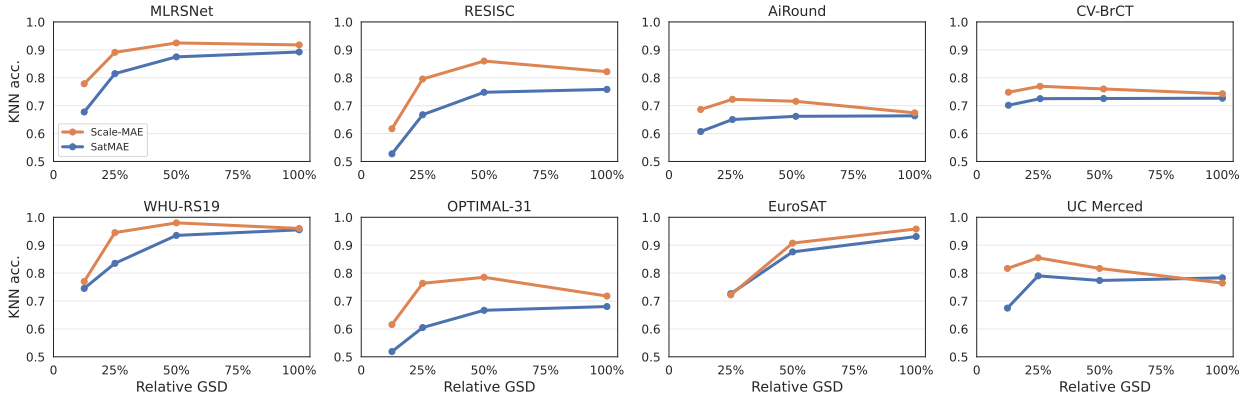

Figure 5.6: **Learning better representations at all scales.** *Scale-MAE* (orange) features perform better than state-of-the-art. We evaluate kNN accuracy on eight datasets with a large variance in GSD. *Scale-MAE* consistently produces better results at coarser resolutions. In addition to using evaluation datasets at different GSDs, to further test the multiscale representations, we create multiple test sets for each dataset in which we downsampled the full resolution validation set to coarser GSDs at fixed percentages: $X_{val}^{G\%}, G \in \{12.5, 25, 50, 100\}$, where Eurosat does not include the 12.5% because the images are at a resolution of 64px, our patch size is 16px, and an input image of 8px is too small.

include (1) a diversity in the amount of kinds of classes/objects represented, (2) a large spectrum of ground sample distances from (ideally) known sensor configurations, and (3) pansharpened, othrorectified, and quality controlled imagery and labels. We capture these properties in Table 5.1.

Diversity in classes For both pretraining and downstream evaluations, it is a desirable property to include as much geographic and class diversity as possible. In order to capture a wide amount of classes in remote sensing, it is necessary to include multiple localities and environments. This property serves as a proxy for the amount of unique "features" available in the dataset.

Evaluation Details

As discussed in the main experimental section, we investigated the quality of representations learned from *Scale-MAE* pretraining through a set of experiments that explore their robustness to scale as well as their transfer performance to additional tasks. We provide more information and details on these evaluations here. In order to compare with SatMAE [28], for our main experiments, we pretrained *Scale-MAE* with a ViT-Large model using the Functional Map of the World (FMoW) RGB training set, which consists of 363.6k images of varying image resolution and GSD. The initial higher resolution image I_{hr} is taken as a random 448px^2 crop of the input image, and the input image I is then a downsampled 224px^2 from I_{hr}. The low frequency groundtruth is obtained by downscaling I_{hr} to 14px^2 and then upscaling to 224px^2, while the high frequency groundtruth is obtained by downscaling I_{hr} to 56px^2 and then upscaling to 448px^2 and subtracting this image from

Dataset	Res (px)	GSD (m)	Num Images	Num Classes	Task
AiRound [109]	500	0.3 - 4800	11,753	11	C
CV-BrCT [109]	500	0.3 - 4800	24,000	9	C
EuroSAT [71]	64	10	27,000	10	C
MLRSNet [130]	256	0.1 - 10	109,161	46	C
Optimal-31 [161]	256	0.5 - 8	1,860	31	C
RESISC-45 [24]	256	0.2 - 30	31,500	45	C
UC Merced [179]	256	0.3	2,100	21	C
WHU-RS19 [32]	256	0.5	950	19	C
fMoW [26]	Various	0.3	1,047,691	62	C
SpaceNet v1 [157]	Various	0.5	6,940	2	SS

Table 5.1: Statistics of all datasets used in our experiments. Task types are classification (C) and semantic segmentation (SS).

I_{hr}. This is a common method for band pass filtering used in several super resolution works, where a high to low to high resolution interpolation is used to obtain only low frequency results, and then high frequency results are obtained by subtracting the low frequency image.

As further discussed in the main experimental section, we evaluate the quality of representations from *Scale-MAE* by freezing the encoder and performing a nonparametric k-nearest-neighbor (kNN) classification with eight different remote sensing imagery classification datasets with different GSDs, none of which were encountered during pretraining. The kNN classifier operates by encoding all train and validation instances, where each embedded instance in the validation set computes the cosine distance with each embedded instance in the training set, where the instance is classified correctly if the majority of its k-nearest-neighbors are in the same class as the validation instance. The justification for a kNN classifier evaluation is that a strong pretrained network will output semantically grouped representation for unseen data of the same class. This evaluation for the quality of representations occurs in other notable works [13, 20, 166].

Representation Quality

In order to compare with SatMAE [28], for our main experiments, we pretrain *Scale-MAE* with a ViT-Large model using the Functional Map of the World (FMoW) [26] RGB training set, which consists of 363.6k images of varying image resolution and GSD. The initial higher resolution image I_{hr} is taken as a random 448px^2 crop of the input image, and the input image I is then a downsampled 224px^2 from I_{hr}. The low frequency groundtruth is obtained by downscaling I_{hr} to 14px^2 and then

Method	Backbone	Model	mIoU
Sup. (Scratch)	ResNet50	PSANet	75.6
GASSL [4]	ResNet50	PSANet	78.5
Sup. (Scratch)	ViT-Large	PSANet	74.7
SatMAE[28]	ViT-Large	PSANet	78.1
Sup. (Scratch)	ViT-Large	UperNet	71.6
Vanilla MAE	ViT-Large	UperNet	77.9
SatMAE	ViT-Large	UperNet	78.0
Scale-MAE	ViT-Large	UperNet	**78.9**

Table 5.2: Semantic segmentation results on SpaceNet v1. *Scale-MAE* outperforms other methods across backbone and segmentation architectures, where Sup. (Scratch) indicates a supervised model trained from scratch (a randomly initialized network).

upscaling to 224px^2, while the high frequency groundtruth is obtained by downscaling I_{hr} to 56px^2 and then upscaling to 448px^2 and subtracting this image from I_{hr}.

Figure 5.5 shows examples of the masked input, low resolution/frequency, high resolution/frequency, and combined reconstruction of FMoW images during training. The low resolution/frequency images capture color gradients and landscapes, while the residual high resolution/frequency images capture object edges, roads, and building outlines.

We evaluate the quality of representations from *Scale-MAE* by freezing the encoder and performing a nonparametric k-nearest-neighbor (kNN) classification with eight different remote sensing imagery classification datasets with different GSDs, none of which were encountered during pretraining. The kNN classifier operates by encoding all train and validation instances, where each embedded instance in the validation set computes the cosine distance with each embedded instance in the training set, where the instance is classified correctly if the majority of its k-nearest-neighbors are in the same class as the validation instance.

The reasoning behind the kNN classifier evaluation is that a strong pretrained network will output semantically grouped representation for unseen data of the same class. This evaluation for the quality of representations occurs in other notable works [13, 20, 166]. In addition to using evaluation datasets at different GSDs, to further test the multiscale representations, we create multiple test sets for each dataset. Since we cannot synthesize data at a finer GSD than the provided ground truth, we only downsample the full resolution validation set to coarser GSDs at fixed percentages: $X_{val}^{G\%}, G \in \{12.5, 25, 50, 100\}$.

Our analysis uses eight different land-use classification datasets: RESISC-45 [24], the UC Merced Land Use Dataset [179], AiRound and CV-BrCT [109], MLRSNet [130], EuroSAT [71], Optimal-31 [161], WHU-RS19 [32], SpaceNet v1 and v2 [157], and Functional Map of the World [26]. The datasets used span a wide range of GSDs, e.g., MLRSNet consists of data captured from

	Average Accuracy (%)	
Dataset	***Scale-MAE***	**SatMAE**
AiRound	**62.7** (+4.9)	57.8
CV-BrCT	**69.6** (+3.4)	66.2
EuroSAT	**86.2** (+1.8)	84.4
MLRSNet	**80.8** (+5.8)	75.0
OPTIMAL-31	**64.1** (+8.4)	55.7
RESISC	**69.1** (+8.1)	61.0
UC Merced	**74.0** (+4.2)	69.8
WHU-RS19	**81.5** (+3.0)	78.5

Table 5.3: *Scale-MAE* performs better, across all GSDs (as in Figure 5.6), for all datasets we experimented with compared to SatMAE. The average improvement across all dataset for *Scale-MAE* compared to SatMAE is 5.0%.

aerial platforms with 0.1m GSD, while RESISC45 has imagery from medium-resolution satellites at >30m GSD. In some cases, the datasets present imagery at mixed GSDs which are not specified, in which case we assume an approximate constant GSD: see the supplementary material for all details. Furthermore, we provide an expanded set of experiments with linear probing and finetuning in the supplementary material.

We run kNN classification with $k = 20$. Figure 5.6 shows that *Scale-MAE* outperforms SatMAE across GSD scales in the different evaluation datasets and across relative GSD scales within individual datasets. At the lowest fixed GSD, UC Merced has a GSD of 0.3m, where evaluating at scales [12.5%,100%] provides an artificial GSD range of [0.3m, 2.4m], we see that *Scale-MAE* has the largest performance gap at the 2.4m GSD, with similar performance at 0.3m.

Across the other datasets and range of GSDs, *Scale-MAE* outperforms SatMAE, where the performance gap tends to grow as the GSD varies from the original GSD, indicating that *Scale-MAE* learns representations that are more robust to changes in scale for remote sensing imagery. We outperform SatMAE by an average of 5.0% across all resolutions and datasets (see Table 5.3). UC Merced at 100% of the true GSD is the only evaluation where SatMAE outperforms *Scale-MAE*. The supplementary material contains an extensive table demonstrating kNN classification results with varying k.

Semantic segmentation transfer We use the SpaceNet v1 building segmentation dataset [157] to evaluate semantic segmentation results on contrastive and MAE-based pretraining methods. Prior methods relied on the PSANet [188] segmentation architecture, while *Scale-MAE* uses the UperNet[169] segmentation architecture which is more common for ViT backbones. For even comparison, we test the current state-of-the-art SatMAE method with UperNet as well. Results are detailed in Table 5.2.

Scale-MAE outperforms SatMAE by 0.9 mIoU and a vanilla MAE (with the same pretraining settings) by 1.0 mIoU. *Scale-MAE* outperforms all other prior work, including GASSL [4], which

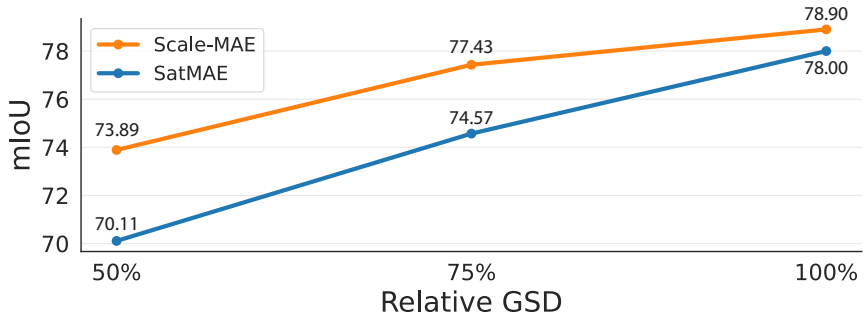

Figure 5.7: SpaceNet v1 evaluation across downscaled images for both *Scale-MAE* and SatMAE. *Scale-MAE* maintains higher semantic segmentation performance over SatMAE, even with images of coarser GSD.

SatMAE did not outperform on the mean Intersection over Union (mIoU) metric for semantic segmentation. Particularly, *Scale-MAE* increases the gap in performance as the resolution of input imagery becomes coarser, highlighting the absolute scale-invariance introduced by our method.

In Figure 5.7, we compare SpaceNet v1 evaluations across downscaled images (at 50%, 75%, and 100% of the original image size) for *Scale-MAE* and SatMAE. Similar to the classification results, *Scale-MAE* maintains higher semantic segmentation performance over SatMAE, even with images at a coarser GSD. In fact, the performance gap grows at coarser GSDs, where at the input GSD, *Scale-MAE* is 0.9 mIoU higher, at 75% GSD *Scale-MAE* is 2.9 mIoU higher, and at 50% *Scale-MAE* is 3.8 mIoU higher.

Linear Classification Results

We perform Linear classification on the RESISC-45 and FMoW-RGB datasets. We fine-tune for 50 epochs using the same hyperparameter settings as SatMAE [28]: a base learning rate of 5×10^{-3}, a weight decay of 5×10^{-3}. We do not use temporal data for classification. For RESISC-45, we fine-tune for 100 epochs with a base learning rate of 4×10^{-3}, a weight decay of 5×10^{-3}, and a global batch size of 256 across 2 GPUs. The learning rate on the backbone is multiplied by a factor of 0.1. We use RandomResizedCrop for augmentation. We train on 224x224 images and evaluate 256x256 images because we found evaluating at a higher scale improves the performance of all models. We report both the performance of end-to-end fine-tuning and linear probing with a frozen backbone. The linear probing setup was the same as finetuning except the learning rate was 0.1. The results are shown in Table 5.4 and Table 5.5.

Visualization of SpaceNet Segmentation

Figure 5.8 shows an additional set of segmentation examples comparing *Scale-MAE* and vanilla MAE pre-trained on FMoW and finetuned on SpaceNet v1. The left, center, right columns are

Model	Frozen/Finetune
Scale-MAE	**89.6/95.7**
SatMAE [28]	88.3/94.8
MAE [67]	88.9/93.3

Table 5.4: **Transfer classification results on RESISC-45.** Frozen indicates a linear probe and finetune is a full end-to-end finetuning of the entire model.

Model	Backbone	Top-1/Top-5
Scale-MAE	ViT-Large	**77.9/94.3**
SatMAE † [28]	ViT-Large	72.4/91.9
MAE [67]	ViT-Large	68.4/90.3
SatMAE ∗ [28]	ViT-Large	77.8/-
GASSL [5]	ResNet-50	71.55/-
MoCo-V2 [69]	ResNet-50	64.34/-

Table 5.5: **Linear probe and finetune classification results on FMoW-RGB.** †: We reproduce the SatMAE by taking the publicly available codebase and pretrain on FMoW dataset for 800 epochs. The results differ from their reported results, but are evaluated consistently with ours. * Reports the results from the SatMAE paper [28].

Method	GSDPE	KNN 50%	KNN 100%
Vanilla MAE		72.8	77.8
Vanilla MAE	✓	75.4	78.5
Scale-MAE		75.3	79.6
Scale-MAE	✓	**78.1**	**80.7**

Table 5.6: Ablation results indicating the importance of GSDPE as determined by a KNN classification on RESISC-45 at a relative GSD of 50% and 100% of its native GSD. Using the GSDPE leads to better performance for both *Scale-MAE* and the Vanilla MAE.

ground truth labels, *Scale-MAE* and vanilla MAE respectively. The top row shows a 0.3m GSD image and the bottom row shows a 3.0m GSD image. As shown in the figure, *Scale-MAE* performs better at both higher and lower GSDs.

Ablations

We ablate the key components of the *Scale-MAE* pretraining framework. For these experiments, we use a lightweight pretraining setting, where we pretrain for 300 epochs on FMoW (rather than

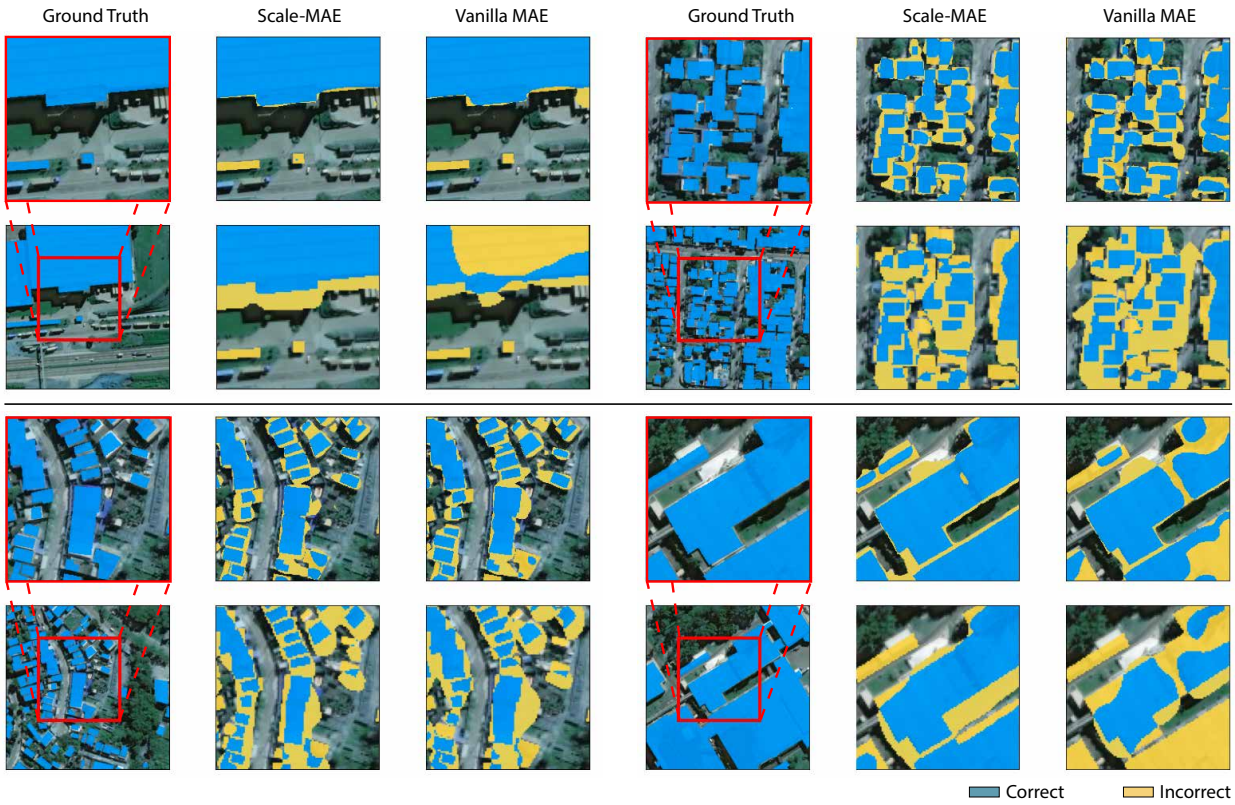

Figure 5.8: Visualization of Segmentation Results on SpaceNet. The left, center, right columns are ground truth labels, *Scale-MAE* and vanilla MAE, respectively. The top row shows a 0.3m GSD image and the bottom row shows a 3.0m GSD image. As shown in the figure, *Scale-MAE* performs better at both higher and lower GSDs.

800) and use a ViT-Base encoder (rather than ViT-Large), and evaluate using a kNN evaluation on RESISC-45 at 100% and 50% of its native GSD. The key contributions that we ablate are as follows: the GSD positional encoder in Table 5.6, in which we find that the GSD postional encoder benefits both *Scale-MAE* and Vanilla MAE across resolutions. In Table 5.7, we see that the number of transformer layers can be reduced from 8 to 3 compared to a Vanilla MAE, which results in a performance improvement. The standard masking rate of 75% still appears optimal for *Scale-MAE* according to the results in Table 5.8.

In Table 5.9 we ablate the necessity of the low and high resolution reconstructions. Specifically, we test reconstructing the low resolution image only, the high resolution image, and a combined image (rather than independent low/high reconstructions). In this case, when the high resolution component is reconstructed, we do not use the low-resolution residual, but rather, directly reconstruct the high resolution result. The "Combined" entry combines the low and high resolution results instead of treating them as separate learning objectives. The separate low/high resolution reconstructions obtain the best performance and robustness to changes in scale.

Decoding Layers	KNN 50%	KNN 100%
1	76.0	78.4
2	77.9	80.4
3	**78.1**	**80.7**
4	77.5	80.0
8	77.7	78.9

Table 5.7: Ablation results indicating that fewer transformer layers in the decoding stage tend to work better for *Scale-MAE* as determined by a KNN classification on RESISC-45 at a relative GSD of 50% and 100% of its native GSD.

Mask Rate	KNN 50%	KNN 100%
70%	77.3	79.3
75%	**78.1**	**80.7**
80%	**78.1**	79.9

Table 5.8: Ablation results indicating that a 75% mask rate is optimal as determined by a KNN classification on RESISC-45 at a relative GSD of 50% and 100% of its native GSD.

Low Res	High Res	Combined	KNN 50%	KNN 100%
		✓	77.6	80.2
✓			72.9	74.3
	✓		77.2	80.3
✓	✓		**78.1**	**80.7**

Table 5.9: These ablation results indicate that reconstructing both the low resolution and high resolution components lead to robust performance. Note: when the high resolution component is reconstructed, the low-resolution residual is not used—the high resolution result is directly reconstructed. The "Combined" entry merges the low and high resolution results instead of treating them as separate losses. The evaluations are a kNN classification (k=20) on RESISC-45 at relative GSDs 50% and 100% of its native GSD.

5.5 Discussion

In this section, we share observations about *Scale-MAE*, sketch our vision for future work, and discuss high-level questions about *Scale-MAE*.

Computational complexity. *Scale-MAE* requires a much smaller decoder than vanilla MAE—instead of a decoder depth of eight, *Scale-MAE* works well with a depth of three. In fact, with

322.9M vs 329.5M parameters using ViT-Large, *Scale-MAE* is smaller than vanilla MAE. However, GPU memory usage for equal batch sizes are higher for *Scale-MAE* since we reconstruct a higher resolution image in the *Scale-MAE* Decoder.

Multi-spectrality and modality. Electro-optical (EO) satellites, such as the ones comprising the datasets mentioned in this work, capture light at different wavelengths. Each wavelength has a different sensor, and each sensor can have a different resolution. *Scale-MAE* requires input tensors to be stacked to pass through the model. This means that we are unable to use *Scale-MAE* when the input image's bands are all of different GSDs. Additionally, synthetic aperture radar (SAR) imagery is another form of remote sensing where resolution varies across a single band. Extending *Scale-MAE* to work with different resolution bands and modalities is reserved for future work.

Can the *Scale-MAE* methodology be applied to other backbones? Methods such as ConvNeXt[107] provide competitive performance compared to Transformers. The core components of our work can be integrated, with additional work, into different architectures. The Laplacian Decoder in *Scale-MAE* can be engineered to ingest convolutional feature maps. Existing work on scale-aware CNNs can be extended to work with the Laplacian Decoder.

Evaluating on more remote sensing datasets. The field of remote sensing has had a renaissance in the last five years with the amount of available datasets. These can be generic, like Functional Map of the World, to highly specific, such as identifying illegal airstrips in Brazil[1, 15] or identifying illegal fishing vessels[123]. In fact, there are so many small, specific remote sensing datasets that entire review papers are written to enumerate them[172]. We chose to focus datasets with *properties* of remote sensing that are relevant to multi-scale representation learning.

Chapter 6

Conclusion

In the past few years, representation learning research has dramatically shifted from supervised approaches to (unsupervised) contrastive approaches to (unsupervised) reconstructive approaches. In retrospect, it's easy to think "well, of course, there's way more unlabeled data so this transition is necessary," or "reconstructive approaches have fewer inductive biases and scale; all of that work on contrastive approaches was a waste of time." But of course, the reality is that these realizations are hard-earned through careful experimentation and refinement of ideas through a broad community. This dissertation summarizes my contribution to this progress, and I look forward to continuing this collective journey.

Scale has been one of the latent factors for progressing this line of research, where the research community has cyclically revisited existing ideas with new architectures, datasets, and hardware that better enable scaling. While one thrust of research has examined scale, another complementary thrust of research (including this dissertation) has asked: how can we do more with less? This dissertation provided a number of approaches to improve the label and data efficiency in modern representation learning pipelines, and the methods and directions presented in this dissertation are reflective of the macroscopic research directions taken by the community as a whole.

Chapter 2, "SelfAugment: Automatic Augmentation Policies for Self-Supervised Learning" [137] presented an AutoML approach to determine optimal augmentations to use for contrastive learning pipelines. Several notable works have cited and built upon this paper [76, 61], and the community as a whole recognized that handcrafted augmentation policies were inherently brittle [170]. Since this work, reconstructive representation learning methods such as the Masked AutoEncoder (MAE) have found that image augmentations are, in fact, unnecessary to obtain strong performing visual representations.

Chapter 3 presented "Self-Supervised Pretraining Improves Self-Supervised Pretraining" [136], which showed that high-performing representations could be learned by simply finetuning a very small number of parameters in the network using a very small amount of unlabeled data and an appropriate self-supervised finetuning pipeline. Following this work, the unsupervised learning community adopted the term "Foundation Models," [9], which cited this work, and adopted a similar unsupervised training/finetuning process as proposed this work. Of course, the nature of research is cyclical, and my work built upon similar proposals from others. The resulting branding of the term

"Foundation Models", though not novel, appears here to stay, with thousands of papers adopting this term and direction in the past two years alone [9].

Chapter 4 presented "Region Similarity Representation Learning" (ReSim) [168], a method for learning region-level representation by performing contrastive learning at a region (patch-based) level. The region level representations could then be used for downstream region-level tasks such as object detection or segmentation, and as we showed, led to several state-of-the-art results, particularly when there was a limited amount of labeled downstream data. Many works have built upon ReSim since its publication, and in fact, this area has garnered so much attention that there is an entire survey devoted to this exact problem [77]. Current research directions have explored applying the same concepts presented in ReSim to transformer architectures and MAEs [**TODO**].

Chapter 5 presented "Scale-MAE: A Scale-Aware Masked Autoencoder for Multiscale Geospatial Representation Learning," [135], my most recent publication before this dissertation. Remote sensing imagery has accelerated the rate of scientific discovery in a broad set of disciplines. With increasingly precise methods to extract environmental indicators using computer vision methods, automated understanding of remotely sensed sources has become a mainstay in scientific literature. Remote sensing payloads are diverse and capture data at a wide range of resolutions, a feature heavily utilized by scientists. Current computer vision methods for remote sensing necessitate the training of a new model per input resolution. Not only is the training process expensive, but the overhead of curating a dataset at multiples scales makes this a daunting task. *Scale-MAE* is a pretraining framework that directly introduces scale invariance into encoders that are used for a diverse set of downstream tasks. Our insights into scale-inclusive positional encodings and progressive multi-frequency feature extraction result in models that perform significantly better than state-of-the-art pretraining methods across (1) multiple scales and (2) many benchmarks. Our goal was to take the extremely diverse and rich source of information present in remote sensing imagery and make it simple to use with minimal training iterations required. With the introduction of *Scale-MAE*, I hope to have further accelerated the rate at which scientific disciplines create impact in this space.

At a high level, this dissertation presents complementary and exploratory methods for data and label efficient representation learning. These methods have been adopted by several industry partners and collaborators throughout this work, formed a key part of our entry to DARPA's LWLL competition [33], and provided a foundation for other works to build upon (indeed, even the foundational Foundation Models work [9]!). I look forward to continuing this journey in representation learning and further contributing and drawing from the wonderful progress in our collective field.

Bibliography

[1] Manuela Andreoni et al. “The Illegal Airstrips Bringing Toxic Mining to Brazil’s Indigenous Land.” In: *The New York Times* (Aug. 2, 2022).

[2] Saeed Anwar, Salman Khan, and Nick Barnes. “A deep journey into super-resolution: A survey.” In: *ACM Computing Surveys (CSUR)* 53.3 (2020), pp. 1–34.

[3] Kumar Ayush et al. “Geography-Aware Self-Supervised Learning.” In: *arXiv preprint arXiv:2011.09980* (2020).

[4] Kumar Ayush et al. “Geography-Aware Self-Supervised Learning.” In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 10161–10170. ISBN: 978-1-66542-812-5. DOI: 10.1109/ICCV48922.2021.01002.

[5] Kumar Ayush et al. “Geography-aware self-supervised learning.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10181–10190.

[6] Philip Bachman, R Devon Hjelm, and William Buchwalter. “Learning Representations by Maximizing Mutual Information Across Views.” In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.

[7] Yoshua Bengio. “Deep learning of representations for unsupervised and transfer learning.” In: *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings. 2012, pp. 17–36.

[8] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation learning: A review and new perspectives.” In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

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

[10] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. “A training algorithm for optimal margin classifiers.” In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 144–152.

[11] Andrew P Bradley. “The use of the area under the ROC curve in the evaluation of machine learning algorithms.” In: *Pattern recognition* 30.7 (1997), pp. 1145–1159.

[12] P. Burt and E. Adelson. "The Laplacian Pyramid as a Compact Image Code." In: *IEEE Transactions on Communications* 31.4 (Apr. 1983), pp. 532–540. ISSN: 1558-0857. DOI: 10.1109/TCOM.1983.1095851.

[13] Mathilde Caron et al. "Emerging Properties in Self-Supervised Vision Transformers." In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 9630–9640. ISBN: 978-1-66542-812-5. DOI: 10.1109/ICCV48922.2021.00951.

[14] Mathilde Caron et al. "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments." In: *Thirty-fourth Conference on Neural Information Processing Systems (NeurIPS)*. 2020.

[15] Fen Chen et al. "Fast Automatic Airport Detection in Remote Sensing Images Using Convolutional Neural Networks." In: *Remote Sensing* 10.3 (Mar. 2018), p. 443. ISSN: 2072-4292. DOI: 10.3390/rs10030443.

[16] Ting Chen et al. "A Simple Framework for Contrastive Learning of Visual Representations." In: *arXiv:2002.05709 [cs, stat]* (Mar. 2020). arXiv: 2002.05709. URL: http://arxiv.org/abs/2002.05709 (visited on 03/31/2020).

[17] Ting Chen et al. "A simple framework for contrastive learning of visual representations." In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[18] Ting Chen et al. "Big self-supervised models are strong semi-supervised learners." In: *arXiv preprint arXiv:2006.10029* (2020).

[19] Xinlei Chen and Kaiming He. "Exploring Simple Siamese Representation Learning." In: *arXiv preprint arXiv:2011.10566* (2020).

[20] Xinlei Chen and Kaiming He. "Exploring Simple Siamese Representation Learning." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15750–15758.

[21] Xinlei Chen et al. "Improved Baselines with Momentum Contrastive Learning." In: (Mar. 2020). arXiv: 2003.04297. URL: http://arxiv.org/abs/2003.04297 (visited on 04/14/2020).

[22] Xinlei Chen et al. "Improved baselines with momentum contrastive learning." In: *arXiv preprint arXiv:2003.04297* (2020).

[23] Yinbo Chen, Sifei Liu, and Xiaolong Wang. "Learning continuous image representation with local implicit image function." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 8628–8638.

[24] Gong Cheng, Junwei Han, and Xiaoqiang Lu. "Remote Sensing Image Scene Classification: Benchmark and State of the Art." In: *Proceedings of the IEEE* 105.10 (Oct. 2017), pp. 1865–1883. ISSN: 1558-2256. DOI: 10.1109/JPROC.2017.2675998.

[25] Gong Cheng, Junwei Han, and Xiaoqiang Lu. "Remote sensing image scene classification: Benchmark and state of the art." In: *Proceedings of the IEEE* 105.10 (2017), pp. 1865–1883.

[26] Gordon Christie et al. "Functional Map of the World." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, June 2018, pp. 6172–6180. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00646.

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

[28] Yezhen Cong et al. *SatMAE: Pre-training Transformers for Temporal and Multi-Spectral Satellite Imagery*. Oct. 2022. DOI: 10.48550/arXiv.2207.08051. arXiv: 2207.08051 [cs].

[29] Marius Cordts et al. "The cityscapes dataset for semantic urban scene understanding." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[30] Ekin D. Cubuk et al. "AutoAugment: Learning Augmentation Policies from Data." In: *arXiv:1805.09501 [cs, stat]* (Apr. 2019). arXiv: 1805.09501, pp. 113–123. URL: http://arxiv.org/abs/1805.09501 (visited on 04/14/2020).

[31] Ekin D. Cubuk et al. "RandAugment: Practical automated data augmentation with a reduced search space." In: *arXiv:1909.13719 [cs]* (Nov. 2019). arXiv: 1909.13719. URL: http://arxiv.org/abs/1909.13719 (visited on 04/14/2020).

[32] Dengxin Dai and Wen Yang. "Satellite Image Classification via Two-Layer Sparse Coding With Biased Image Representation." In: *IEEE Geoscience and Remote Sensing Letters* 8.1 (Jan. 2011), pp. 173–176. ISSN: 1558-0571. DOI: 10.1109/LGRS.2010.2055033.

[33] *DARPA Learning with Less Labeling Program*. URL: https://www.darpa.mil/program/learning-with-less-labeling.

[34] Jia Deng et al. "Imagenet: A large-scale hierarchical image database." In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[35] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *arXiv preprint arXiv:1810.04805* (2018).

[36] Jian Ding et al. "Unsupervised Pretraining for Object Detection by Patch Reidentification." In: *arXiv preprint arXiv:2103.04814* (2021).

[37] Carl Doersch, Abhinav Gupta, and Alexei A Efros. "Unsupervised visual representation learning by context prediction." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1422–1430.

[38] Carl Doersch and Andrew Zisserman. "Multi-task self-supervised visual learning." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2051–2060.

[39] Jeff Donahue et al. "Decaf: A deep convolutional activation feature for generic visual recognition." In: *International conference on machine learning*. PMLR. 2014, pp. 647–655.

[40] Alexey Dosovitskiy et al. “Discriminative unsupervised feature learning with convolutional neural networks.” In: *Advances in neural information processing systems*. 2014, pp. 766–774.

[41] Alexey Dosovitskiy et al. “Discriminative unsupervised feature learning with exemplar convolutional neural networks.” In: *IEEE transactions on pattern analysis and machine intelligence* 38.9 (2015), pp. 1734–1747.

[42] Alexey Dosovitskiy et al. “Flownet: Learning optical flow with convolutional networks.” In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2758–2766.

[43] Dumitru Erhan et al. “Why does unsupervised pre-training help deep learning?” In: *Journal of Machine Learning Research* 11.Feb (2010), pp. 625–660.

[44] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[45] Mark Everingham et al. “The pascal visual object classes (voc) challenge.” In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[46] Mark Everingham et al. “The PASCAL visual object classes challenge 2007 (VOC2007) results.” In: (2007).

[47] Mark Everingham et al. “The pascal visual object classes challenge: A retrospective.” In: *International journal of computer vision* 111.1 (2015), pp. 98–136.

[48] Haoqi Fan et al. “Multiscale vision transformers.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6824–6835.

[49] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. “A Discriminatively Trained, Multiscale, Deformable Part Model.” In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. June 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587597.

[50] Anthony Fuller, Koreen Millard, and James R. Green. “SatViT: Pretraining Transformers for Earth Observation.” In: *IEEE Geoscience and Remote Sensing Letters* 19 (2022), pp. 1–5. ISSN: 1558-0571. DOI: 10.1109/LGRS.2022.3201489.

[51] Yuan Gao, Xiaojuan Sun, and Chao Liu. “A General Self-Supervised Framework for Remote Sensing Image Classification.” In: *Remote Sensing* 14.19 (2022), p. 4824.

[52] Golnaz Ghiasi and Charless C. Fowlkes. “Laplacian Pyramid Reconstruction and Refinement for Semantic Segmentation.” In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 519–534. ISBN: 978-3-319-46487-9. DOI: 10.1007/978-3-319-46487-9_32.

[53] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. *Unsupervised Representation Learning by Predicting Image Rotations*. 2018. arXiv: 1803.07728 [cs.CV].

[54] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. “Unsupervised Representation Learning by Predicting Image Rotations.” In: *CoRR* abs/1803.07728 (2018). arXiv: 1803.07728. URL: http://arxiv.org/abs/1803.07728.

[55] Spyros Gidaris et al. “Learning representations by predicting bags of visual words.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6928–6938.

[56] Ross Girshick. “Fast r-cnn.” In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[57] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT press, 2016.

[58] Ian J Goodfellow et al. “Multi-digit number recognition from street view imagery using deep convolutional neural networks.” In: *arXiv preprint arXiv:1312.6082* (2013).

[59] Priya Goyal et al. “Scaling and benchmarking self-supervised visual representation learning.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6391–6400.

[60] Jean-Bastien Grill et al. “Bootstrap your own latent: A new approach to self-supervised learning.” In: *arXiv preprint arXiv:2006.07733* (2020).

[61] Devin Guillory et al. “Predicting with confidence on unseen distributions.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1134–1144.

[62] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. “Densepose: Dense human pose estimation in the wild.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7297–7306.

[63] Ritwik Gupta et al. *Accelerating Ukraine Intelligence Analysis with Computer Vision on Synthetic Aperture Radar Imagery*. http://bair.berkeley.edu/blog/2022/03/21/ukraine-sar-maers/. Mar. 2022.

[64] Suchin Gururangan et al. “Don’t Stop Pretraining: Adapt Language Models to Domains and Tasks.” In: *arXiv preprint arXiv:2004.10964* (2020).

[65] Kaiming He et al. “Deep residual learning for image recognition.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[66] Kaiming He et al. “Mask r-cnn.” In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[67] Kaiming He et al. *Masked Autoencoders Are Scalable Vision Learners*. Dec. 2021. DOI: 10.48550/arXiv.2111.06377. arXiv: 2111.06377 [cs].

[68] Kaiming He et al. “Momentum contrast for unsupervised visual representation learning.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2020.

[69] Kaiming He et al. “Momentum contrast for unsupervised visual representation learning.” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9729–9738.

[70] Yutong He et al. “Spatial-Temporal Super-Resolution of Satellite Imagery via Conditional Pixel Synthesis.” In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 27903–27915.

[71] Patrick Helber et al. "Introducing Eurosat: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification." In: *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*. July 2018, pp. 204–207. DOI: 10.1109/IGARSS.2018.8519248.

[72] Olivier Henaff. "Data-efficient image recognition with contrastive predictive coding." In: *International conference on machine learning*. PMLR. 2020, pp. 4182–4192.

[73] Olivier J Hénaff et al. "Data-efficient image recognition with contrastive predictive coding." In: *arXiv preprint arXiv:1905.09272* (2019).

[74] Daniel Ho et al. "Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules." In: *arXiv:1905.05393 [cs, stat]* (May 2019). arXiv: 1905.05393. URL: http://arxiv.org/abs/1905.05393 (visited on 04/14/2020).

[75] X Hu et al. "A magnification-arbitrary network for super-resolution." In: *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA*. 2019, pp. 15–20.

[76] Tianyu Hua et al. "On feature decorrelation in self-supervised learning." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9598–9608.

[77] Gabriel Huang et al. "A survey of self-supervised and few-shot object detection." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[78] Damian Ibañez et al. "Masked Auto-Encoding Spectral-Spatial Transformer for Hyperspectral Image Classification." In: *IEEE Transactions on Geoscience and Remote Sensing* (2022).

[79] Jeremy Irvin et al. "Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 590–597.

[80] Ashish Jaiswal et al. "A survey on contrastive self-supervised learning." In: *Technologies* 9.1 (2021), p. 2.

[81] Pengpeng Ji, Shengye Yan, and Qingshan Liu. "Region-Based Spatial Sampling for Image Classification." In: *2013 Seventh International Conference on Image and Graphics*. July 2013, pp. 874–879. DOI: 10.1109/ICIG.2013.177.

[82] Borui Jiang et al. "Acquisition of localization confidence for accurate object detection." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 784–799.

[83] Daniel Kermany, Kang Zhang, and Michael Goldbaum. "Large dataset of labeled optical coherence tomography (oct) and chest x-ray images." In: *Mendeley Data, v3 http://dx. doi. org/10.17632/rscbjbr9sj* 3 (2018).

[84] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. "Contrastive representation learning: A framework and review." In: *IEEE Access* (2020).

[85] Jan J. Koenderink. "The Structure of Images." In: *Biological Cybernetics* 50.5 (Aug. 1984), pp. 363–370. ISSN: 1432-0770. DOI: 10.1007/BF00336961.

[86] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. "Revisiting self-supervised visual representation learning." In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2019, pp. 1920–1929.

[87] Alexander Kolesnikov et al. *Big Transfer (BiT): General Visual Representation Learning*. 2020. arXiv: 1912.11370 [cs.CV].

[88] Alexander Kolesnikov et al. "Big transfer (bit): General visual representation learning." In: *arXiv preprint arXiv:1912.11370* 6.2 (2019), p. 8.

[89] Simon Kornblith, Jonathon Shlens, and Quoc V Le. "Do better imagenet models transfer better?" In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2661–2671.

[90] Pawel Kowaleczko et al. "MuS2: A Benchmark for Sentinel-2 Multi-Image Super-Resolution." In: *arXiv preprint arXiv:2210.02745* (2022).

[91] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-10 (Canadian Institute for Advanced Research)." In: (). URL: http://www.cs.toronto.edu/~kriz/cifar.html.

[92] Darius Lam et al. "xview: Objects in context in overhead imagery." In: *arXiv preprint arXiv:1802.07856* (2018).

[93] Nick Lamm et al. "Vehicle Trajectory Prediction by Transfer Learning of Semi-Supervised Models." In: *arXiv preprint arXiv:2007.06781* (2020).

[94] S. Lazebnik, C. Schmid, and J. Ponce. "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories." In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. June 2006, pp. 2169–2178. DOI: 10.1109/CVPR.2006.68.

[95] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." In: *nature* 521.7553 (2015), pp. 436–444.

[96] Christian Ledig et al. "Photo-realistic single image super-resolution using a generative adversarial network." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4681–4690.

[97] Xueting Li et al. "Joint-task self-supervised learning for temporal correspondence." In: *arXiv preprint arXiv:1909.11895* (2019).

[98] Sungbin Lim et al. "Fast AutoAugment." In: *arXiv:1905.00397 [cs, stat]* (May 2019). arXiv: 1905.00397, pp. 6662–6672. URL: http://arxiv.org/abs/1905.00397 (visited on 04/14/2020).

[99] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2117–2125.

[100] Tsung-Yi Lin et al. "Feature pyramid networks for object detection." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[101] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context." In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[102] Chenxi Liu et al. "Are Labels Necessary for Neural Architecture Search?" In: *arXiv preprint arXiv:2003.12056* (2020).

[103] Li Liu et al. "Deep learning for generic object detection: A survey." In: *International journal of computer vision* 128.2 (2020), pp. 261–318.

[104] Wei Liu et al. "SSD: Single Shot MultiBox Detector." In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 21–37. ISBN: 978-3-319-46448-0. DOI: 10.1007/978-3-319-46448-0_2.

[105] Xiao Liu et al. "Self-supervised learning: Generative or contrastive." In: *arXiv preprint arXiv:2006.08218* 1.2 (2020).

[106] Yufei Liu et al. "A Band Selection Method with Masked Convolutional Autoencoder for Hyperspectral Image." In: *IEEE Geoscience and Remote Sensing Letters* (2022).

[107] Zhuang Liu et al. "A ConvNet for the 2020s." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 11976–11986.

[108] Ilya Loshchilov and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts." In: *arXiv preprint arXiv:1608.03983* (2016).

[109] Gabriel Machado et al. "AiRound and CV-BrCT: Novel Multiview Datasets for Scene Classification." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14 (2021), pp. 488–503. ISSN: 2151-1535. DOI: 10.1109/JSTARS.2020.3033424.

[110] Dhruv Mahajan et al. "Exploring the limits of weakly supervised pretraining." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 181–196.

[111] Tomasz Malisiewicz and Alexei A Efros. "Recognition by association via learning per-exemplar distances." In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8.

[112] Matthias Minderer et al. "Unsupervised learning of object structure and dynamics from videos." In: *arXiv preprint arXiv:1906.07889* (2019).

[113] Ishan Misra and Laurens van der Maaten. "Self-supervised learning of pretext-invariant representations." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6707–6717.

[114] Malachy Moran et al. *Snowpack Estimation in Key Mountainous Water Basins from Openly-Available, Multimodal Data Sources*. Aug. 2022. DOI: 10.48550/arXiv.2208.04246. arXiv: 2208.04246 [cs].

[115] Maxim Neumann et al. "In-domain representation learning for remote sensing." In: *arXiv preprint arXiv:1911.06721* (2019).

[116] Maxim Neumann et al. “In-domain representation learning for remote sensing.” In: *ArXiv* abs/1911.06721 (2019).

[117] Alejandro Newell and Jia Deng. “How useful is self-supervised pretraining for visual tasks?” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 7345–7354.

[118] Jiquan Ngiam et al. “Domain adaptive transfer learning with specialist models.” In: *arXiv preprint arXiv:1811.07056* (2018).

[119] Maria-Elena Nilsback and Andrew Zisserman. “Automated flower classification over a large number of classes.” In: *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. IEEE. 2008, pp. 722–729.

[120] Shuteng Niu et al. “Distant Domain Transfer Learning for Medical Imaging.” In: *arXiv preprint arXiv:2012.06346* (2020).

[121] Mehdi Noroozi and Paolo Favaro. “Unsupervised learning of visual representations by solving jigsaw puzzles.” In: *European Conference on Computer Vision*. Springer. 2016, pp. 69–84.

[122] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation learning with contrastive predictive coding.” In: *arXiv preprint arXiv:1807.03748* (2018).

[123] Fernando Paolo et al. “xView3-SAR: Detecting Dark Fishing Activity Using Synthetic Aperture Radar Imagery.” In: *Thirty-Sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. Oct. 2022.

[124] Karl Pearson. “LIII. On lines and planes of closest fit to systems of points in space.” In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11 (1901), pp. 559–572.

[125] Chao Peng et al. “Megdet: A large mini-batch object detector.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6181–6189.

[126] Xingchao Peng et al. “Moment matching for multi-source domain adaptation.” In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 1406–1415.

[127] Jonas Pfeiffer et al. “AdapterHub: A Framework for Adapting Transformers.” In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020, pp. 46–54.

[128] Pedro O Pinheiro et al. “Unsupervised Learning of Dense Visual Representations.” In: *arXiv preprint arXiv:2011.05499* (2020).

[129] Joan Puigcerver et al. “Scalable transfer learning with expert models.” In: *arXiv preprint arXiv:2009.13239* (2020).

[130] Xiaoman Qi et al. “MLRSNet: A Multi-Label High Spatial Resolution Remote Sensing Dataset for Semantic Scene Understanding.” In: *ISPRS Journal of Photogrammetry and Remote Sensing* 169 (Nov. 2020), pp. 337–350. DOI: 10.1016/j.isprsjprs.2020.09.020.

[131] Ariadna Quattoni, Michael Collins, and Trevor Darrell. "Transfer learning for image classification with sparse prototype representations." In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8.

[132] Alec Radford et al. "Improving language understanding by generative pre-training." In: *Preprint* (2018).

[133] Maithra Raghu et al. "Transfusion: Understanding transfer learning for medical imaging." In: *arXiv preprint arXiv:1902.07208* (2019).

[134] Rajat Raina et al. "Self-taught learning: transfer learning from unlabeled data." In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 759–766.

[135] Colorado J Reed et al. "Scale-MAE: A Scale-Aware Masked Autoencoder for Multiscale Geospatial Representation Learning." In: *arXiv preprint arXiv:2212.14532* (2022).

[136] Colorado J Reed et al. "Self-Supervised Pretraining Improves Self-Supervised Pretraining." In: *arXiv preprint arXiv:2103.12718* (2021).

[137] Colorado J Reed et al. "SelfAugment: Automatic Augmentation Policies for Self-Supervised Learning." In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2021. arXiv: 2009.07724 [cs.CV].

[138] Shaoqing Ren et al. "Faster R-CNN: towards real-time object detection with region proposal networks." In: *IEEE transactions on pattern analysis and machine intelligence* 39.6 (2016), pp. 1137–1149.

[139] Cedric Renggli et al. "Which Model to Transfer? Finding the Needle in the Growing Haystack." In: *arXiv preprint arXiv:2010.06402* (2020).

[140] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. "Playing for benchmarks." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2213–2222.

[141] A. Rosenfeld and M. Thurston. "Edge and Curve Detection for Visual Scene Analysis." In: *IEEE Transactions on Computers* C-20.5 (May 1971), pp. 562–569. ISSN: 1557-9956. DOI: 10.1109/T-C.1971.223290.

[142] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge." In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[143] Kuniaki Saito et al. "Semi-supervised domain adaptation via minimax entropy." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 8050–8058.

[144] Jacob Shermeyer and Adam Van Etten. "The effects of super-resolution on object detection performance in satellite imagery." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[145] H. Shin et al. "Autoencoder in Time-Series Analysis for Unsupervised Tissues Characterisation in a Large Unlabelled Medical Image Dataset." In: *2011 10th International Conference on Machine Learning and Applications and Workshops*. Vol. 1. 2011, pp. 259–264.

[146] Connor Shorten and Taghi M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning." en. In: *Journal of Big Data* 6.1 (July 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0 (visited on 04/08/2020).

[147] C Spearman. "THE PROOF AND MEASUREMENT OF ASSOCIATION BETWEEN TWO THINGS." In: *The American Journal of Psychology* (1904), p. 72.

[148] Christian Szegedy et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." In: *Thirty-first AAAI conference on artificial intelligence*. 2017.

[149] Chuanqi Tan et al. "A survey on deep transfer learning." In: *International conference on artificial neural networks*. Springer. 2018, pp. 270–279.

[150] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.

[151] Chao Tao et al. "Remote Sensing Image Scene Classification With Self-Supervised Paradigm Under Limited Labeled Samples." In: *IEEE Geoscience and Remote Sensing Letters* (2020).

[152] Beth Tellman, Jonathan A. Sullivan, and Colin S. Doyle. "Global Flood Observation with Multiple Satellites." In: *Global Drought and Flood*. American Geophysical Union (AGU), 2021. Chap. 5, pp. 99–121. ISBN: 978-1-119-42733-9. DOI: 10.1002/9781119427339.ch5.

[153] James Thewlis, Hakan Bilen, and Andrea Vedaldi. "Unsupervised learning of object frames by dense equivariant image labelling." In: *arXiv preprint arXiv:1706.02932* (2017).

[154] Jessica A. F. Thompson, Yoshua Bengio, and Marc Schönwiesner. "The effect of task and training on intermediate representations in convolutional neural networks revealed with modified RV similarity analysis." In: *CoRR* abs/1912.02260 (2019). arXiv: 1912.02260. URL: http://arxiv.org/abs/1912.02260.

[155] Yonglong Tian, Dilip Krishnan, and Phillip Isola. "Contrastive multiview coding." In: *arXiv preprint arXiv:1906.05849* (2019).

[156] Yonglong Tian et al. "What Makes for Good Views for Contrastive Learning?" In: *arXiv preprint arXiv:2005.10243* (2020).

[157] Adam Van Etten, Dave Lindenbaum, and Todd M. Bacastow. *SpaceNet: A Remote Sensing Dataset and Challenge Series*. July 2019. DOI: 10.48550/arXiv.1807.01232. arXiv: 1807.01232 [cs].

[158] Ashish Vaswani et al. "Attention Is All You Need." In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[159] Carl Vondrick et al. "Tracking emerges by colorizing videos." In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 391–408.

[160] Hanchen Wang et al. "Pre-Training by Completing Point Clouds." In: *arXiv preprint arXiv:2010.01089* (2020).

[161] Qi Wang et al. "Scene Classification With Recurrent Attention of VHR Remote Sensing Images." In: *IEEE Transactions on Geoscience and Remote Sensing* 57.2 (Feb. 2019), pp. 1155–1167. ISSN: 1558-0644. DOI: 10.1109/TGRS.2018.2864987.

[162] Xiaolong Wang, Allan Jabri, and Alexei A Efros. "Learning correspondence from the cycle-consistency of time." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2566–2576.

[163] Xinlong Wang et al. "Dense Contrastive Learning for Self-Supervised Visual Pre-Training." In: *arXiv preprint arXiv:2011.09157* (2020).

[164] Zhongrui Wang et al. "Fully memristive neural networks for pattern classification with unsupervised learning." In: *Nature Electronics* 1.2 (2018), pp. 137–145.

[165] Yuxin Wu et al. *Detectron2*. github.com/facebookresearch/detectron2. 2019.

[166] Zhirong Wu et al. "Unsupervised Feature Learning via Non-Parametric Instance Discrimination." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3733–3742.

[167] Zhirong Wu et al. "Unsupervised feature learning via non-parametric instance discrimination." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3733–3742.

[168] Tete Xiao et al. "Region Similarity Representation Learning." In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 10519–10528. ISBN: 978-1-66542-812-5. DOI: 10.1109/ICCV48922.2021.01037.

[169] Tete Xiao et al. "Unified Perceptual Parsing for Scene Understanding." In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Vol. 11209. Cham: Springer International Publishing, 2018, pp. 432–448. ISBN: 978-3-030-01227-4 978-3-030-01228-1. DOI: 10.1007/978-3-030-01228-1_26.

[170] Tete Xiao et al. "What should not be contrastive in contrastive learning." In: *International Conference on Learning Representations*. 2021.

[171] Enze Xie et al. "Detco: Unsupervised contrastive learning for object detection." In: *arXiv preprint arXiv:2102.04803* (2021).

[172] Zhitong Xiong et al. *EarthNets: Empowering AI in Earth Observation*. Oct. 2022. DOI: 10.48550/arXiv.2210.04936. arXiv: 2210.04936 [cs].

[173] Xingqian Xu, Zhangyang Wang, and Humphrey Shi. "Ultrasr: Spatial encoding is a missing key for implicit image function-based arbitrary-scale super-resolution." In: *arXiv preprint arXiv:2103.12716* (2021).

[174] I Zeki Yalniz et al. "Billion-scale semi-supervised learning for image classification." In: *arXiv preprint arXiv:1905.00546* (2019).

[175] Shengye Yan et al. "Beyond Spatial Pyramids: A New Feature Extraction Framework with Dense Spatial Sampling for Image Classification." In: *European Conference on Computer Vision 2012*. Aug. 2012, pp. 473–487.

[176] Shengye Yan et al. "Image Classification With Densely Sampled Image Windows and Generalized Adaptive Multiple Kernel Learning." In: *IEEE Transactions on Cybernetics* 45.3 (Mar. 2015), pp. 381–390. ISSN: 2168-2275. DOI: 10.1109/TCYB.2014.2326596.

[177] Ceyuan Yang et al. "Instance localization for self-supervised detection pretraining." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3987–3996.

[178] Wenming Yang et al. "Deep learning for single image super-resolution: A brief review." In: *IEEE Transactions on Multimedia* 21.12 (2019), pp. 3106–3121.

[179] Yi Yang and Shawn Newsam. "Bag-of-Visual-Words and Spatial Extensions for Land-Use Classification." In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '10. New York, NY, USA: Association for Computing Machinery, Nov. 2010, pp. 270–279. ISBN: 978-1-4503-0428-3. DOI: 10.1145/1869790.1869829.

[180] Yi Yang and Shawn Newsam. "Bag-of-visual-words and spatial extensions for land-use classification." In: *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*. 2010, pp. 270–279.

[181] Zhengyuan Yang et al. "End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions." In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE. 2018, pp. 2289–2294.

[182] Mang Ye et al. "Unsupervised embedding learning via invariant and spreading instance feature." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6210–6219.

[183] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: *arXiv preprint arXiv:1411.1792* (2014).

[184] Fisher Yu et al. "Bdd100k: A diverse driving dataset for heterogeneous multitask learning." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2636–2645.

[185] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks." In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[186] Richard Zhang, Phillip Isola, and Alexei A Efros. "Colorful image colorization." In: *European conference on computer vision*. Springer. 2016, pp. 649–666.

[187] X Zhang et al. "The iMaterialist Challenge 2017 Dataset." In: *FGVC Workshop at CVPR*. Vol. 2. 2017, p. 3.

[188] Hengshuang Zhao et al. "PSANet: Point-wise Spatial Attention Network for Scene Parsing." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 267–283.

[189] Bolei Zhou et al. "Learning deep features for scene recognition using places database." In: *Advances in neural information processing systems*. 2014, pp. 487–495.