COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

Université de Rennes

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Computer Science /Informatique*

Par

Shashanka Venkataramanan

Metric learning for instance and category-level visual representation

Thèse présentée et soutenue à Rennes, le 1st July 2024
Unité de recherche : INRIA Rennes

Composition of the Jury :

President:	Prof. Dr. Nicolas Courty	Professor, Université Bretagne Sud
Reviewers:	Prof. Dr. Andrew Zisserman	Professor, University of Oxford
Reviewers:	Prof. Dr. Jiri Matas	Professor, Czech Technical University, Prague
	Dr. Piotr Bojanowski	Principal Research Scientist, Meta AI (FAIR)
Examiners :	Dr. Diane Larlus	Principal Research Scientist, Naver Labs Europe
	Dr. Andrei Bursuc	Senior Research Scientist, valeo.ai

Thesis supervised by:

Thesis Director :	Dr. Laurent Amsaleg	Senior Researcher, HDR, CNRS
Thesis Co-directors :	Dr. Yannis Avrithis	HDR, Principal Investigator, IARAI
	Dr. Ewa Kijak	HDR, Associate Professor, University of Rennes 1

To the most influential women in my life:
K. Saraswati, Padmalatha, Jayashree, and Sritanaya

endaro mahanubhavulu
andariki vandanamulu

(To all the great souls who have
guided me in my path,
please accept my humble salutations)

Acknowledgement

Throughout my PhD journey, I've been truly grateful by the support and encouragement from so many wonderful people. It's incredibly difficult to fully express the deep gratitude I feel towards all those who have made this journey so rewarding and exciting. The next few hundred words can never truly capture all the appreciation I have for everyone who contributed to this incredible experience. I am deeply grateful for their remarkable contributions and humbly acknowledge their role in this thesis.

First and foremost, to my exceptional supervisor Yannis Avrithis, whose unbridled support has been an invaluable gift throughout my PhD journey. It is a rare and extraordinary privilege to have had the opportunity to work with him, and I consider myself truly fortunate. The countless hours he dedicated to reviewing and refining our papers, coupled with his insightful feedback and unwavering belief in my abilities, have fostered an environment of growth and innovation. I am immensely grateful for his trust, wisdom, and the personal investment he has made in my success. I almost do not want to graduate, because I know I will miss him. His advice on research and life will stay in my heart forever.

To my supervisors Laurent Amsaleg and Ewa Kijak, thank you for helping me integrate into a new environment in France. Your guidance and support have shaped the course of my doctoral pursuit.

I extend my sincere gratitude to all three of you for assembling an exceptional jury for my defense. Words cannot adequately convey the immense pride and honor I feel to present and defend my work in the presence of researchers whom I deeply admire. I am immensely grateful to each member of the jury for generously devoting their valuable time to meticulously reviewing the manuscript and for their presence during the defense.

I also owe an anonymous acknowledgement to the entire scientific community of computer vision and machine learning for the incredible progress over the past two decades. Such advancements would have been impossible for any individual researcher or team working alone in this ever-evolving field. I am also deeply grateful to the anonymous reviewers and area chairs whose valuable feedback has helped me become a better researcher today.

During my PhD, I had been fortunate to intern at Qualcomm Research with Amir Ghodrati and Amirhossein Habibian. I thank them for their support and efforts in making my time with the team both comfortable and fruitful. Their welcoming and inclusive approach created an environment where I felt truly at ease, allowing me to contribute to the best of my abilities. I would also like to highlight the delightful moments spent engaging in lively discussions about Italian cuisines, property investments, literature, and "stuff like that" with Davide Abati, which added a touch of joy to the overall experience. I am also grateful to Haitam Ben Yahia for his patience with my constant whining about limited computational resources, even when I occupied half of Qualcomm's GPUs :). I hold dear the memories of this internship experience: It was Lekker!!!

I am grateful to Yuki Asano, my collaborator, for the countless inspiring discussions and for motivating me to find the "wow" factor in our research problems. His insightful comments and collaborative spirit have been invaluable in shaping the direction and quality of our research. I am truly fortunate to have had the opportunity to work alongside such an exceptional individual. To many more collaborations ahead :) To Joao Carreira, with whom I had the pleasure of collaborating, your one or two-liner insights were like a jolt of lightning to my brain, leaving me pondering for weeks on end. In our brief time together, I gained more knowledge than I ever thought possible. I'm also eternally grateful for your unwavering commitment to the project, even when it meant sacrificing precious time from your busy schedule. And let's be real, that controversial yet catchy paper title? Pure genius. I'm pretty sure it's the reason we won that Best Paper Honorable Mention award at ICLR, haha. To Bill Psomas, for the enthusiasm and joy you brought to our project. Thank you for the many interesting conversations about life, Greek food, workouts and for recommending some amazing Greek songs which I listen to on a loop :p. It was a pleasure and I learned a lot from working with you all!

To my friend Mohammadreza Salehi, thanks for your unwavering support and motivation during difficult times and inspiring me to explore life beyond just research. Your wisdom and compassion were instrumental in helping me regain balance and perspective. To my labmates Deniz Engin and Ali Yesilkanat, who have been my companions throughout this research journey. Together, we have weathered the ups and downs, sharing not only the bitter-sweet experiences of research but also the challenges and joys of life. This collective experience has been invaluable, shaping us into better researchers and creating lasting memories that I will cherish. To Konstantinos Tertikas, Hanwei Zhang and Karnik Ram, thanks for being such wonderful and supportive friends. Lastly, to Kassem Kallas,

Guillaume Le Noe-Bienvenu, Suresh Kirthi Kumaraswamy, Ankit Sharma, Kevin Duarte, Jogendra Kumar, Avishrey Chauhan, and Umang Jain – thank you for being a crucial part of my home away from home, in this beautiful corner of the world.

Finally, I would like to extend my deepest appreciation to my parents Padmalatha and Venkataramanan, and my sister Jayashree, for supporting me through thick and thin. To my mom, who has been my rock during the challenging times, patiently listening to my rants about unfair reviews of R2 and always finding ways to calm me down. To my dad, who would meticulously comb through every review and comment on my papers, wielding a red pen like a surgeon's scalpel. Your relentless pursuit of perfection may have driven me to the brink of insanity, but hey, at least now I can handle any critique thrown my way. Thanks for making me bulletproof, old man! I feel like we have all graduated together and earned this doctoral degree as a collective accomplishment. I also thank my in-laws Srinivas and Syamasri Tatipamala for all their encouragement and support. Perhaps the most extraordinary part of this PhD journey has been to meet Sritanaya, who has been my pillar of support, my food partner, advisor, fan, therapist, and biggest source of happiness.

Table of Contents

Chapter 1

Introduction

1.1 Visual Representations

Designing effective representations is a central component of many Artificial Intelligence (AI) systems, and it has evolved significantly over the past decades. The performance of an AI system is heavily dependent on the quality of the input data representation. Just as humans find arithmetic more intuitive when working with numerals rather than binary or Roman numerals [Dehaene, 2011], the input representation plays a crucial role in the performance of machine learning (ML) systems. For example, in a simple ML system tasked with identifying the risk of prostate cancer, the system does not interact directly with the patient, but rather inputs a set of variables, such as clinical and demographic conditions, gathered by a specialist [Stamey, 1989]. This set of variables constitutes the patient representation seen by the ML algorithm, which then learns how these different variables interact to make predictions.

Beyond the textual and numerical representations commonly used, modern AI systems have expanded to encompass a wide range of data modalities. *Computer vision* is one such modality, which deals with the understanding of visual data such as images and videos. Vision algorithms leverage representations to extract meaningful information from visual inputs, enabling AI systems to perceive, interpret, and make decisions based on visual cues, much like humans do. In the field of computer vision, the ability to understand the semantic content of an image is crucial for a wide range of tasks, such as classification, retrieval, detection, segmentation etc. Furthermore, computer vision systems can leverage multimodal data, where images or videos are coupled with textual information, to tackle challenges such as image/video captioning and visual question answering. These tasks often involve two key components: a mechanism to extract information from the image, and a secondary mechanism to perform the specific task based on the extracted information.

What are representations? Representations refer to the mathematical or computational forms used to encode and process data. In computer vision, the features extracted from raw image data are referred to as visual representations, or descriptors [Lowe, 2004; Csurka, 2004; Dalal, 2005]. These representations can take various forms, such as specific structures and patterns like edges, points, or objects, or they can be derived from a feature extractor function (or encoder) that maps raw pixel values to a vector of fixed dimension, which serves as the image representation [Goodfellow, 2016].

Representation learning Representation learning is the process of automatically discovering the representations needed for feature detection or classification from raw data [Bengio, 2012]. The goal of representation learning is to find a transformation of the input data that makes it easier to extract useful information when building classifiers or other predictors. This is in contrast to traditional machine learning approaches, where the representation (e.g., the set of features) is manually engineered based on domain knowledge.

Importance of representation learning The goal is to improve the visual perception capabilities of AI systems, with the aspiration of matching or even surpassing the natural ability of humans to perceive and understand the visual world [Krizhevsky, 2009; Simonyan, 2015]. This motivation stems from the observation that most living beings can effortlessly see and comprehend the visual world without requiring tremendous amounts of energy, suggesting that visual perception could be realized by computer systems as well [Marr, 2010].

Designing effective image featurization is a challenging task because it is not a well-defined problem. It is not clear what information exactly needs to be extracted from an image, nor how to extract it effectively [Bengio, 2012]. For example, in the context of an application that detects dogs in images, the presence of a muzzle might be a useful feature. However, precisely describing a muzzle in an image is a complex task due to the infinite variations in lighting conditions, shadows, occlusions, and other factors that can affect the visual representation of the muzzle [Dalal, 2005].

Representation Learning: pre deep-learning era The idea behind artificial neural networks dates back to mid 1900s, *e.g.* [McCulloch, 1944; Rosenblatt, 1958] and the research in this field has undergone periods of advancement and setback until early 2000s [Goodfellow, 2016]. Some important works during this period include the perceptron model of [Rosenblatt, 1958], its extension to multi-layer perceptrons by [Ivakhnenko,

1971], using backpropagation [Rumelhart, 1986; Werbos, 1974] for training of neural networks with multiple hidden units, and the introduction of convolutional neural networks (CNN) [Fukushima, 1980; LeCun, 1989] to improve the generalization of networks by exploiting certain data biases such as local structures in images, and invariance to translation. We encourage the readers to see [Goodfellow, 2016] for a more detailed discussion.

During the early 2000s, researchers extensively studied the problem of finding effective image representations through handcrafted approaches [Csurka, 2004; Lowe, 2004; Perronnin, 2007]. In these methods, the representations were designed manually by humans based on visual cues such as edges, corners, and other low-level image statistics like pixel gradients. The first generation of methods tackling the challenge of visual representation learning included algorithms like SIFT [Lowe, 2004], HOG [Dalal, 2005] and SURF [Bay, 2006], which were obtained through manual feature engineering. These handcrafted feature extraction pipelines produced representations with desirable properties, such as invariance to scale, illumination, or rotation.

These handcrafted feature extraction techniques were widely used in the pre-deep learning era, as they provided a systematic way to engineer useful visual representations without requiring extensive labeled training data or computational resources.

Representation Learning: deep-learning era With the advancements in deep learning, the paradigm has shifted towards learning representations from "data", rather than manually designing them [Bengio, 2012]. Deep neural networks have the remarkable ability to build a hierarchical representation of their input, starting from low-level features like edges and corners, and progressively learning higher-level concepts, such as object parts and semantic information. This ability to learn task-specific representations directly from data has been a significant advantage of deep learning approaches. Unlike the handcrafted features, which were often general-purpose and required manual tuning for specific tasks, deep neural networks can learn representations that are tailored to the problem at hand [Sharif Razavian, 2014].

The ability to learn and extract representations is directly baked into the deep neural network architectures used in deep learning. For example, a multi-layer perceptron (MLP) maps input to output by composing successive parametrized simple functions. The output of each function (or layer) can be thought of as a new representation of the input, which is then fed to the following layer. This way, the AI system can discover for itself intermediate representations that are useful for solving the specific task, as illustrated Figure 1.1. A

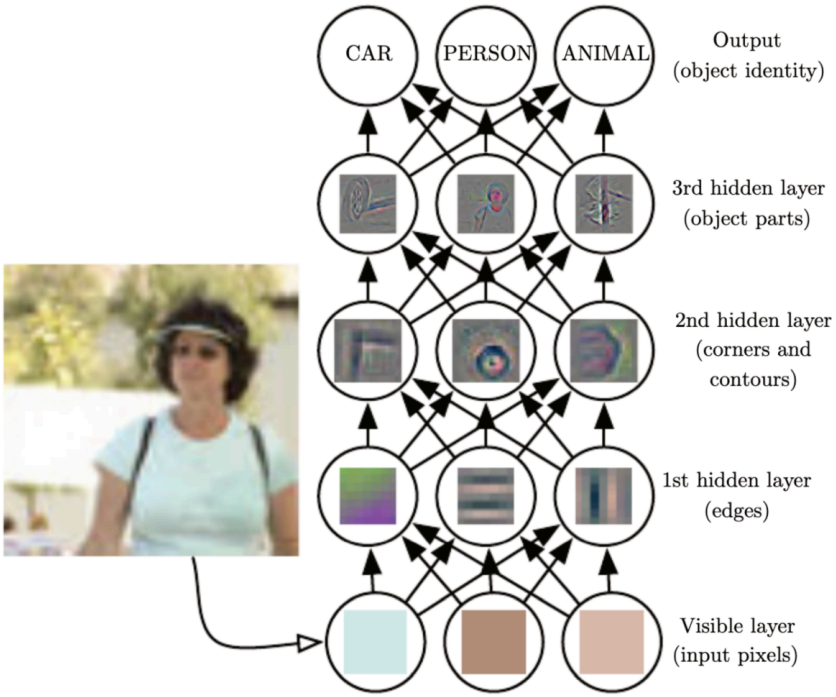


Figure 1.1 – *Illustration on the ability of deep neural networks to learn hierarchical representations of images.* Given input images composed of pixels, low-level representations (e.g., edges) are encoded first, which are combined to form higher-level representations (e.g., object parts) that can be used to solve the task at hand, e.g., image classification. Figure adapted from [Goodfellow, 2016]

well-known example is the `XOR` classification problem, which cannot be solved by a linear model. Yet, a simple deep network, namely an MLP, can solve this problem since it has the ability to learn how to transform the data into an intermediate hidden representation that is linearly separable.

One of the seminal breakthroughs in representation learning was the introduction of AlexNet [Krizhevsky, 2012], which demonstrated the power of learned visual representations, outperforming traditional, hand-crafted features on image classification tasks on ImageNet [Russakovsky, 2015]. The network's ability to automatically extract hierarchical visual features, from low-level edges to high-level object representations, paved the way for a new era of representation learning. Building upon the success of AlexNet, researchers continued to explore deeper and more sophisticated CNN architectures such as VGGNet [Simonyan, 2015]. ResNet [He, 2016b], addressed the challenge of training very deep networks by introducing residual connections, which allowed for the efficient

Figure 1.2 – *Transfer of Representations* is a powerful technique, where the knowledge gained by a model during pretraining on a specific task *e.g.* image classification can be leveraged to improve performance on related tasks *e.g.* object detection, semantic and instance segmentation. This is particularly useful when the target task has limited data available for training, as the pretrained model can provide a strong starting point for fine-tuning. Figure adapted from [Lin, 2014]

training of networks with over 100 layers . These deeper networks were able to learn more complex and expressive visual representations, leading to significant performance improvements on a wide range of computer vision tasks. The field of representation learning further expanded beyond computer vision, with the emergence of transformer-based models, such as BERT [Devlin, 2019], in natural language processing (NLP) . Transformers introduced a novel attention mechanism that enabled the model to dynamically focus on the most relevant parts of the input sequence when generating representations. This allowed for the capture of long-range dependencies and contextual information, leading to state-of-the-art performance on various NLP tasks. The success of transformers in NLP has inspired the development of similar architectures for other domains, such as computer vision. Vision Transformers (ViT) [Dosovitskiy, 2021] have demonstrated the ability to learn powerful visual representations by treating images as sequences of patches and applying the transformer's attention mechanism . These models have shown competitive performance compared to traditional CNN-based approaches, highlighting the versatility of the transformer architecture in representation learning.

Another advantage of representations learning is that, it can be effectively transferred across different tasks, allowing for the transfer of knowledge acquired from solving one task to other related tasks [Bommasani, 2021]. For *e.g.*, in Figure 1.2, the features from a model pretrained for image classification can be transferred to tasks such as object detection [Ren, 2015], semantic segmentation and instance segmentation [He, 2017]. This property of transferability is particularly valuable, as it enables the use of a single representation to solve a wide range of computer vision problems, rather than requiring the development of specialized feature extraction pipelines for each task.

1.2 Data Augmentation

What is data augmentation? Data augmentation artificially increases the size and diversity of training data by creating modified versions of existing data without deviating from the original data distribution. As illustrated in Figure 1.3, applying various transformations to the existing data, such as flipping, rotating, cropping, or adding noise to images, or replacing words with synonyms or paraphrasing sentences in text data, data augmentation generates new, slightly different versions of the original training data. These augmented samples can then be used to train the machine learning model, effectively increasing the size and diversity of the training dataset without the need for additional data collection or labeling efforts [Yang, 2022].

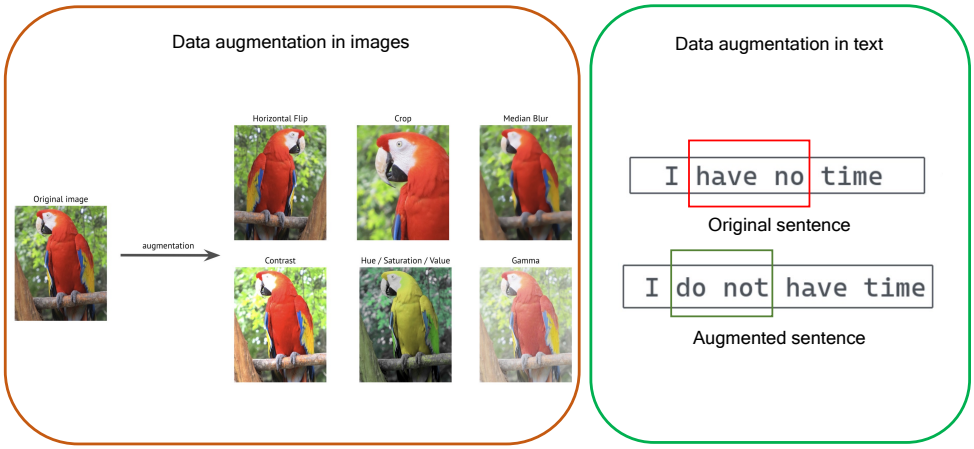


Figure 1.3 – *Common data augmentation techniques.* For images, techniques such as rotation and flipping are shown, while for text, replacing words with synonyms can help increase the diversity of training data. These simple yet effective augmentation strategies can improve model performance by providing a range of variations in the input.

Importance of data augmentation Data augmentation is crucial for several reasons:

1. Deep learning models require large amounts of diverse data to learn effectively and generalize well. However, collecting and annotating data can be time-consuming, expensive, or sometimes impossible. Data augmentation artificially expands the size of the dataset by creating new, modified versions of the existing data.

2. Overfitting occurs when a model learns the training data too well, including its noise and irrelevant patterns, leading to poor performance on new, unseen data.

Data augmentation introduces variations in the training data, forcing the model to learn more robust features and generalize better, thereby reducing overfitting.

3. Real-world data can be noisy, distorted, or incomplete. By augmenting the training data with various transformations (e.g., rotations, flips, noise addition), the model learns to handle such variations, making it more robust and accurate when deployed in real-world scenarios.

4. In many applications, the available data is skewed towards certain classes, leading to biased models. Data augmentation can help balance the class distributions by selectively augmenting the underrepresented classes, improving the model's ability to learn from minority classes.

Transformations in Human Visual Perception Data augmentation techniques have an interesting parallel with human visual perception. While state-of-the-art deep learning models often rely on static images for training, humans perceive the world as a continuous stream of visual information, constantly observing transformations of objects and scenes in real-time [Cinel, 2019].

Continuous perspective transformations, such as those experienced when an object moves or rotates, are critical for perceiving rigid motion and depth in the environment [Gibson, 1957b; Gibson, 1957a]. These transformations allow the visual system to detect and interpret changes in the spatial configuration of objects, facilitating the perception of motion and the three-dimensional structure of the world [Gibson, 1957a]. Large continuous perspective transformations have been empirically shown to be necessary for accurate shape perception, highlighting the importance of dynamic visual information in understanding object properties and spatial relationships [Bingham, 2008]. This continuous stream of visual data enables the brain to develop invariance to various transformations, making it possible to recognize objects despite changes in viewpoint, lighting, or distance. Thus, the human visual system's ability to process continuous transformations is fundamental to how we interact with and understand our surroundings.

By drawing inspiration from the cognitive mechanisms that underlie the ability of humans to learn and generalize, we can develop more sophisticated data augmentation techniques that better mimic and leverage the power of continuous transformation, paving the way for more robust and adaptable AI systems.

1.2.1 Challenges of Data Augmentation

While data augmentation techniques have proven invaluable in enhancing the generalization capabilities of deep learning models, they are not without challenges.

Confined to the image manifold Traditional data augmentation techniques, by their very nature, are confined to the manifold of the original image or sample. While they can introduce variations in orientation, scale, and other geometric properties, they cannot venture beyond the intrinsic characteristics and features present within that specific sample. This limitation restricts the model's exposure to novel combinations of features and patterns that may exist outside the boundaries of the individual samples.

Lack of semantic coherence Another challenge is the potential lack of semantic coherence in the augmented samples. While geometric transformations can introduce variations in appearance, they may inadvertently distort or alter the semantic meaning of the data in unintended ways. This can lead to augmented samples that are visually plausible but semantically inconsistent or unrealistic, potentially introducing noise and confounding factors during the training process.

Neglecting naturalistic variations in videos Most existing data augmentation methods rely on handcrafted transformations like flipping, cropping, rotation, etc., which have limitations in capturing the rich semantic variations present in real-world scenarios, especially for video data. While these geometric and color transformations can artificially increase the diversity of training samples, they often fail to account for more complex and naturalistic variations that occur in videos, such as pose deformations, viewpoint changes, occlusions, and background clutter. For instance, the motion and temporal dynamics present in videos can provide valuable cues about object movements, interactions, and context, which are essential for tasks like action recognition, object tracking, and video understanding.

Challenges in Augmenting Labels In image classification tasks, data augmentation techniques can leverage the inherent label invariance of the transformations applied to the images. For example, rotating, flipping, or cropping an image of a dog does not change the underlying label of the image, which remains "dog." However, in instance retrieval and metric learning tasks, where the goal is to learn discriminative representations for

Figure 1.4 – *Natural augmentation in videos.* differ from hand-crafted augmentation in images. Here, we observe *Perspective distortion*, where objects appear smaller or larger depending on their distance from the camera. We also see *object occlusion*, where the person on the bike is occluded by the traffic pole. Natural augmentations in videos occur organically and can enhance representations learned by encoders.

individual instances or to optimize a distance metric, the label augmentation process becomes more complex.

In instance retrieval tasks, the objective is to retrieve specific instances of objects or scenes from a database, given a query image. Here, the labels are not categorical classes but rather unique identifiers for each instance. Augmenting these instance-level labels is challenging because geometric transformations or other augmentations may alter the visual appearance of the instance in a way that makes it distinct from the original instance, effectively creating a new instance with a different label.

Metric learning aims to learn an embedding space where semantically similar instances are mapped closer together, while dissimilar instances are pushed apart. In this context, the labels represent the pairwise or triplet relationships between instances, indicating whether they are similar or dissimilar. Augmenting these pairwise or triplet labels is non-trivial, as transformations applied to one instance in a pair or triplet may alter the semantic relationship between the instances, rendering the original label invalid.

1.3 Outline and Contributions

To address the limitations and the challenges associated with data augmentations, we propose different approaches that aim to enhance the performance and robustness of image encoders. After a detailed overview of the different methods for data augmentation in image classification, metric learning and self-supervised learning in chapter 2, we present the different contributions conducted during this PhD program to these different settings.

1.3.1 AlignMixup: a natural way of interpolation

Interpolation based data augmention like mixup has shown to improve robustness and model calibration [Verma, 2019]. However, as shown in [Kim, 2020a] the mixed input images tend to look unnatural and the randomness in selecting the patches and mixing their labels may cause the classifier to learn uninformative features. This limitation suggests that exploring interpolation in the feature space, rather than the input space, could be an interesting direction to pursue.

[Bengio, 2013] show that traversing along the manifold of representations obtained from deeper layers of the network more likely results in finding realistic examples. They hypothesize that deeper representations learned by neural networks tend to disentangle the underlying factors of variation better. These disentangled representations can be exploited to produce faster-mixing Markov chains, meaning that deeper representations indeed enable better mixing and generate more realistic interpolations between data points.

Motivated by this observation, we propose to interpolate images in the feature space rather than in the image space. In chapter 3, we show that the idea of deformation is a natural way of interpolating images, where one image may deform into another, in a continuous way. To achieve this, we investigate geometric alignment for mixup, based on explicit semantic correspondences in the feature space. In particular, we align the feature tensors of two images, resulting in soft correspondences. We set a new state-of-the-art on image classification, robustness to adversarial attacks, calibration, weaklysupervised localization and out-of-distribution detection against more sophisticated mixup operations on several networks and datasets.

1.3.2 Extending mixup to metric learning

We discussed AlignMixup, a technique that interpolates between aligned features and improves the performance on image classification tasks. However, these mixup methods do not generalize to different tasks such as instance retrieval and metric learning. Building upon this, we now explore the idea of systematically applying mixup in the domain of deep metric learning.

There exists a striking similarity between using pairwise similarity in metric learning and using pairs of examples in mixup for classification tasks. This observation led us to explore the possibility of interpolating between pairs in metric learning using mixup, similar to how it works in classification. In chapter 4, we introduce mixup in the context of metric learning. However, directly interpolating pairs of embeddings presents a unique challenge. Unlike classification, loss functions in metric learning are not additive over examples, making it non-trivial to directly interpolate target labels using traditional mixup.

To address this challenge, we first develop a generalized formulation that encompasses existing metric learning loss functions and modify it to accommodate mixup. This contributes a principled way of interpolating labels, such that the interpolation factor affects the relative weighting of positives and negatives. Since interpolating between all possible pairs can be computationally expensive, we leverage an efficient linear interpolation strategy, making it significantly faster than complex non-linear interpolation methods. By introducing mixup in metric learning and developing a generalized formulation with efficient interpolation, we aim to improve the performance of deep metric learning tasks, building upon the success of mixup in image classification and the insights gained from AlignMixup in chapter 3.

1.3.3 Interpolation beyond mini-batch, beyond pairs and beyond examples

The previous chapter explored the effectiveness of mixup in deep metric learning, where increasing the number of loss terms by interpolating between all pairs of embeddings improved performance without significant computational overhead. This motivates us to explore the potential of extending mixup further in classification tasks by generating more interpolated examples during training. The original motivation of mixup [Zhang, 2018a], aims to augment the training data by generating new examples through interpolation.

However, it is limited to interpolation between pairs of examples in the input space, as the convex combination of three or more examples did not bring further gains.

In chapter 5, we revisit the initial motivation of mixup and increase the number of augmented examples through interpolation in the embedding space. Instead of interpolating in the input space, we generate an arbitrarily large number of interpolated examples beyond the mini-batch size by interpolating the entire mini-batch in the embedding space. Geometrically, this translates to interpolating between all points, essentially sampling points on the convex hull of the mini-batch. By increasing the number of loss terms per mini-batch by orders of magnitude at little additional cost, made possible by interpolating in the embedding space, we empirically show significant improvements over state-of-the-art mixup methods on four different benchmarks, despite the interpolation being only linear.

1.3.4 Learning strong image encoders from videos

Building upon our exploration of interpolation-based data augmentation for improving representation learning in image classification and deep metric learning, in chapter 6 we venture beyond this approach to investigate the possibility of discovering natural augmentations inherent in real-world data in a self-supervised setting. This shift aligns with the core theme of the thesis, which focuses on combining diverse learning objectives and modalities to enhance representation learning.

The inherent richness of video data presents a unique opportunity to explore natural augmentations. Unlike synthetic augmentations, videos naturally encompass diverse variations in pose, deformation, viewpoint, perspective, occlusion, and background clutter, offering a wealth of rich augmentations for learning robust representations. This eliminates the need for artificially generated augmentations, such as mixup-based interpolations, allowing the model to learn from the intrinsic complexities present within the video data itself. By leveraging the natural variations present in videos, we can potentially learn more robust and generalizable representations without relying on synthetic augmentations. This approach aligns with the core theme of the thesis, which aims to combine diverse learning objectives and modalities to enhance representation learning.

Finally, to wrap up, we summarize in chapter 7 our contributions and give a (subjective) overview of the current challenges in representation learning.

1.3.5 Publications

chapter 3 to chapter 6 each contain a paper which has been peer-reviewed and accepted for publication in a conference. The papers have been left unmodified from their published forms, with the exception of formatting changes. The publications included in this thesis are:

1. chapter 3 is based on the paper "AlignMixup: Improving Representations By Interpolating Aligned Features", Shashanka Venkataramanan, Ewa Kijak, Laurent Amsaleg, and Yannis Avrithis. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022 [Venkataramanan, 2021].

   The code is available at `https://github.com/shashankvkt/AlignMixup_CVPR22`.

2. chapter 4 is based on the paper "It Takes Two to Tango: Mixup for Deep Metric Learning", Shashanka Venkataramanan, Bill Psomas, Ewa Kijak, Laurent Amsaleg, Konstantinos Karantzalos, and Yannis Avrithis. International Conference on Learning Representations (ICLR), 2022 [Venkataramanan, 2022].

   The code is available at `https://github.com/billpsomas/metrix`.

3. chapter 5 is based on the paper "Embedding Space Interpolation Beyond Mini-Batch, Beyond Pairs and Beyond Examples", Shashanka Venkataramanan, Ewa Kijak, Laurent Amsaleg, and Yannis Avrithis. Advances in Neural Information Processing Systems (NeurIPS), 2023 [Venkataramanan, 2023].

4. chapter 6 is based on the paper "Is ImageNet worth 1 video? Learning strong image encoders from 1 long unlabelled video", Shashanka Venkataramanan, Mamshad Nayeem Rizve, João Carreira, Yuki M. Asano, Yannis Avrithis. International Conference on Learning Representations (ICLR), 2024 (ICLR'24 Best paper Honorable mention) [Venkataramanan, 2024b].

   The project page can be found at `https://shashankvkt.github.io/dora`.

Publications not included

1. "Skip-attention: Improving vision transformers by paying less attention", Shashanka Venkataramanan, Amir Ghodrati, Yuki M. Asano, Fatih Porikli, Amirhossein Habibian, International Conference on Learning Representations (ICLR), 2024 [Venkataramanan, 2024a]

Chapter 2

Background

Image-based representation learning has emerged as a pivotal area in the field of computer vision, playing a crucial role in enhancing the performance of various tasks such as *object recognition*, *image classification*, and *semantic segmentation*. The essence of image representation learning lies in the ability to automatically discover meaningful and hierarchical features from raw pixel data, enabling models to capture intricate patterns and structures within images. However, with the increasing complexity of models, the challenge of overfitting has become more pronounced. Furthermore, in the in real-world datasets, the distribution of images during inference often differs from the training set, leading to instances where models make incorrect predictions with high confidence. An interesting direction to solve this challenge has been to develop advanced data augmentation techniques, recognizing the role of expanding training data to improve model generalization.

In this chapter, we explore various methods used in image recognition to tackle overfitting and improve model calibration through data augmentation. We also look into deep metric learning and how data augmentation can help models learn better representations. By incorporating data augmentation techniques in deep metric learning, we aim to equip models with the ability to generalize effectively, thereby enhancing their overall performance and adaptability in diverse scenarios.

2.1 What is data augmentation?

For a given dataset $\{x_i, y_i\}$, where x_i is the training samples and y_i their corresponding labels, the goal of data augmentation is to employ transformation T_j to the original samples x_i. This results in additional training samples $x'_i := T(x_i)$ without modifying the corresponding labels. These transformations are commonly known as label-preserving transformations. Their purpose is to ensure that the modified samples produced by the transformations can still be semantically described by the original label y_i.

Interpolation-based data augmentation, such as mixup, combine two training samples

to create a new, synthetic sample. This approach differs from the single-image transformations. For a given dataset $\mathcal{X}, \mathcal{Y}$, where $\mathcal{X}$ is the examples and $\mathcal{Y}$ the corresponding labels, we sample two training examples $(x_i, x_j) \in \mathcal{X}$ and their labels $(y_i, y_j) \in \mathcal{Y}$. The two samples along with their target labels are then linearly interpolated as $x' = \lambda x_i + (1 - \lambda)x_j$ and $y' = \lambda y_i + (1 - \lambda)y_j$. Here, where $\lambda \sim \text{Beta}(\alpha, \alpha)$is the interpolation factor.

Data augmentation provides an alternative approach to mitigate overfitting in models and improving model calibration by increasing the amount of training data. We first discuss about single-image based data augmentation and then explore interpolation based augmentation.

2.1.1 Image space augmentation techniques

Geometric transformations: Figure 2.1 shows some widely used geometric transformations, which include affine transformations, such as rotation, shearing, translation, scaling or resizing (without zooming or cropping), mirroring, and reflection or flipping. Within affine transformations, rotations, reflections, and translations constitute a subset known as Euclidean transformations [Ryan, 1986]. Several works have demonstrated their high effectiveness in numerous computer vision tasks [Xu, 2016b; Wong, 2016].

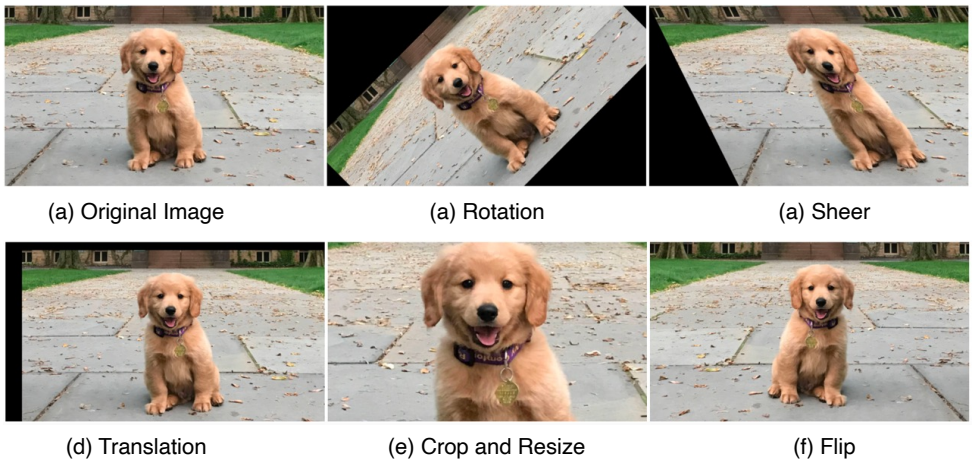


Figure 2.1 – Affine transformations applied to an image. The transformations showcase different manipulations of the original image, demonstrating their effects on its appearance and spatial orientation.

However, they have a few drawbacks. Firstly, these simple transformations are useful only if the current data distribution is similar to the actual data distribution. Secondly, translation and rotation encounter a padding effect wherein portions of the images may

be displaced beyond the boundary and consequently lost. Thus, some interpolation based methods are often used to fill in these areas post-operation.

Photometric transformations Photometric effects shown in Figure 2.2, result from camera artifacts and shooting conditions, such as motion blur, optical noise, and distortions, as well as color artifacts and image data corruption. Unlike geometric transformations, photometric methods alter the pixel content of images while preserving their spatial structure. Specifically, they involve changing visual properties like color, brightness, sharpness, contrast, and saturation levels. This type of data augmentation is crucial for enhancing the robustness of deep learning models to accommodate variations in images caused by environmental conditions like illumination changes, weather fluctuations, and different times of the day. It also addresses artifacts introduced by imaging devices, noise, and camera settings. Commonly, these transformations include color jittering, color space conversion, and image enhancement and distortion techniques.

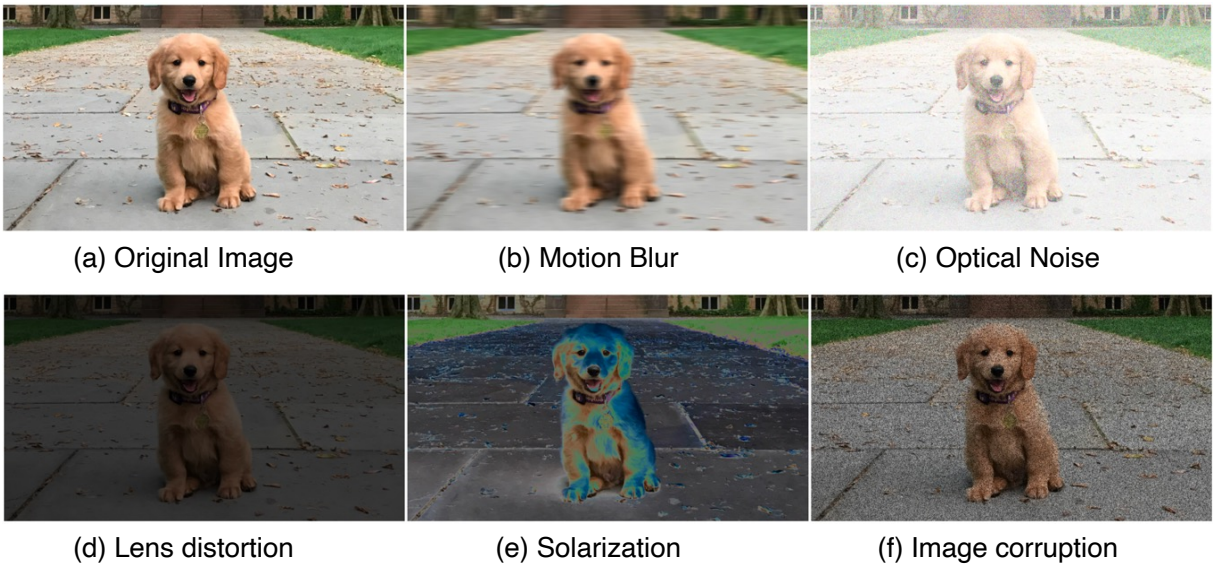


Figure 2.2 – Comparison of different Photometric augmentations. Each subplot illustrates different photometric augmentations applied to the original image such as motion blur, optical noise, lens distortion, solarization, and image corruption.

Image Erasing Image augmentation methods using image erasing shown in Figure 2.3, involve removing specific parts of an image. The basic idea is to replace the pixel values in these removed areas with constant or random values. [DeVries, 2017b], introduce a

simple regularization technique called Cutout, where square regions are randomly masked out during the training. Hide-and-Seek (HaS) [Singh, 2018] randomly hides patches in a training image. This encourages the network to explore other relevant content while the most important content is temporarily hidden. Random erasing [Zhong, 2020] randomly selects a rectangle region in an image and replaces its pixels with random values. GridMask [Chen, 2020a] analyzes the need for dropping information based on deleting regions in input images. Unlike Cutout and HaS, GridMask deletes spatially uniformly distributed squares, allowing control over density and size. FenceMask [Li, 2020b] balances object occlusion and information retention based on simulating an object occlusion strategy.

Figure 2.3 – Comparison of Image erasing Techniques. Each subplot showcases a different method for altering the image, demonstrating various approaches to image erasing: Original Image, Cutout, Random Masking, and Random erasing.

2.1.2 Augmentations in the feature space

[Bengio, 2013], show that lower-level features extracted from deep layers can be used to reconstruct higher-level representations of input images. Essentially, the objective of augmentation strategies inspired by these studies is to transform the data in the feature space. This transformation encourages the resulting deep learning model to learn representations that remain consistent despite transformations in the input space *i.e.* they preserve the equivariance property in deep networks. One significant drawback of these approaches is that feature variability arises from perturbations that lack domain knowledge. Consequently, these perturbations may overlook crucial domain attributes essential for certain tasks *e.g.* domain adaptation, fine-grained classification etc.. Various techniques have been employed to address these challenges, broadly categorized into feature transformation and elimination, which we discuss here.

Feature transformations The goal of feature transformations is to boost the diversity of extracted features, preventing the model from learning specific patterns in the input data. [Shen, 2016] suggest applying random affine operations, like translation, scaling, and rotation, to learned feature maps to increase their variability. [Li, 2016] propose injecting adaptive Gaussian noises into certain layers of deep neural networks to encourage feature diversity and prevent overfitting. A more recent technique, Shakedrop [Yamada, 2019], extends this concept to different multi-branch CNN architectures. KRAM [Jia, 2020] utilizes k-nearest neighbors technique to exploit feature relationships among instances in a multi-dimensional classification setting, transforming the original feature representations to enhance the performance of multi-dimensional classification models. Linear Delta [Kumar, 2019] is another feature transformation, which involves computing the difference between feature vectors from two training samples and then transforming a third feature vector by adding this difference to it.

Feature elimination Feature elimination is a regularization technique that involves eliminating features in the intermediate layers of the network. Among this family of approaches, Dropout and its derivatives [Hinton, 2012; Srivastava, 2014; Bouthillier, 2015; Goodfellow, 2013] have been particularly popular and have been used in state-of-the-art CNN backbones. The aim of these methods is to remove irrelevant features while keeping the useful ones. [Choe, 2019] propose an attention mechanism to identify features that are not important in a specific context. Some recent techniques, like DropBlock [Ghiasi, 2018] Batch DropBlock [Dai, 2019] and SD-Unet [Guo, 2019a] suggest dropping entire groups of units forming a connected region, unlike individual elements as in Dropout [Srivastava, 2014]. DropBlock [Ghiasi, 2018] is an early method introducing a well-structured dropout scheme compatible with CNN architecture, implementing a feature-level Cutout. Similar to cutout, it zeroes out elements in randomly chosen regions of feature maps. SD-Unet [Guo, 2019a] employs a region dropout scheme designed for U-Net architectures [Ronneberger, 2015] to tackle overfitting in medical image segmentation tasks. [Dai, 2019] propose a Person Re-identification (ReID) framework with two network branches. One branch is a regular CNN learning global image features, and the other uses an attention mechanism through a Batch DropBlock (BDB) sub-model for region-wise feature dropping. This concept involves repeatedly dropping units from the same region for each batch of images corresponding to the same human body parts, focusing attention on these parts.

2.1.3 Interpolation based data augmentation

Despite the advantages of data augmentation techniques, its effectiveness is constrained. By operating on one image at a time and limiting to label-preserving transformations, it has limited chances of exploring beyond the image manifold. Hence, it is of little help in combating memorization of training data and sensitivity to adversarial examples.

One approach that has gained significant attention is mixup, which augments training data by interpolating between pairs of examples, thereby encouraging the model to learn more robust representations. In this section, we delve into various mixup techniques and aim to understand their underlying principles and how they contribute to enhancing representation learning.

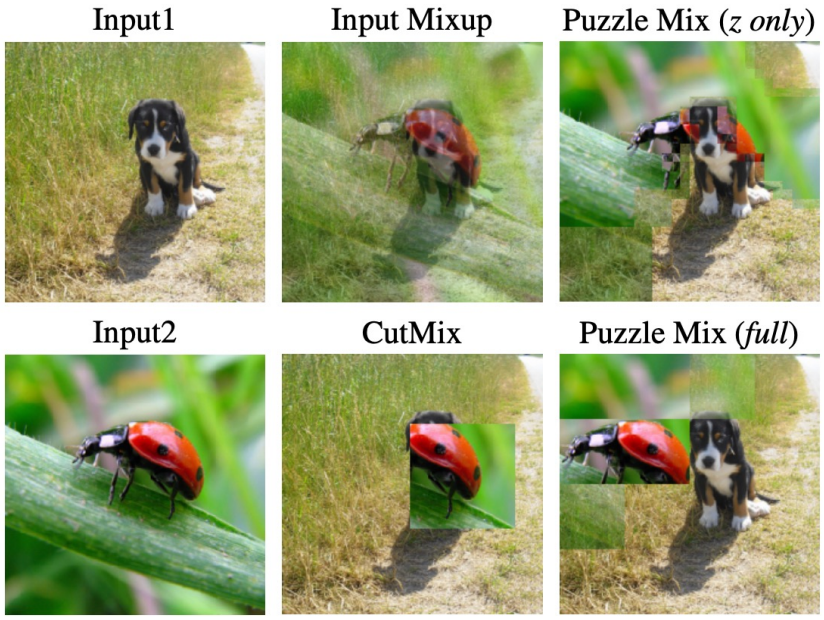


Figure 2.4 – A visual comparison of different mixup methods. The images were taken from [Kim, 2020a].

Input Mixup [Zhang, 2018a] is a data augmentation technique, which generates new training samples by linearly interpolating between pairs of examples and their corresponding labels. Given two samples (x_i, y_i) and $(\ x_j, y_j)$, where x_i and x_j are input samples and y_i and y_j are their respective one-hot encoded labels, the mixup operation can be defined

as follows:

$$\tilde{x} := \lambda x_i + (1 - \lambda) x_j \tag{2.1}$$

$$\tilde{y} := \lambda y_i + (1 - \lambda) y_j \tag{2.2}$$

Here, λ is drawn from a Beta distribution $\text{Beta}(\alpha, \alpha)$ where α is a hyperparameter that controls the strength of interpolation. Mixup encourages the model to learn more linear decision boundaries by interpolating features from different samples, thus enhancing its generalization capability.

Manifold Mixup [Verma, 2019] extends input mixup to operate not only in the input space but also in the latent space. Instead of linearly interpolating between input samples, Manifold Mixup performs mixup in the feature space, aiming to encourage the model to learn smooth and invariant representations. Given two feature maps f_i and f_j corresponding to images x_i and x_j extracted from intermediate layers of a model, the interpolation can be defined as:

$$\tilde{f} := \lambda f_i + (1 - \lambda) f_j \tag{2.3}$$

Manifold Mixup promotes the exploration of the manifold structure of the data, leading to improved robustness against adversarial attacks and out-of-distribution samples.

Cutmix [Yun, 2019] integrates the mixup concept with spatial regularization by replacing rectangular regions of one image with the corresponding regions from another image during training. This process encourages the model to attend to multiple parts of different images simultaneously, promoting robust feature learning. The interpolation operation of Cutmix is defined as:

$$\tilde{f} := M \odot x_i + (1 - M) \odot x_j \tag{2.4}$$

$$\tilde{y} := \lambda y_i + (1 - \lambda) y_j \tag{2.5}$$

Here, $M \in \{0, 1\}^{H \times W}$ is a binary mask drawn from a Bernoulli distribution controlling the areas to be replaced, and λ is sampled from a Beta distribution as in mixup. CutMix enforces the model to learn from the features of the two images simultaneously, thereby improving its robustness to occlusions and variations in object placement.

PuzzleMix [Kim, 2020a] is a data augmentation method that leverages saliency information and local statistics to enhance mixup. It introduces an interpolation technique that conditions the sample mixing on region saliency and local statistics. This results in an interesting optimization challenge that switches between the multi-label objective for determining the optimal mixing mask and the saliency-weighted optimal transport objective. The interpolation strategy for PuzzleMix is given as

$$\tilde{f} := \Pi_0^T \odot x_i + \Pi_1^T \odot x_j \tag{2.6}$$

$$\tilde{y} := \lambda y_i + (1 - \lambda) y_j \tag{2.7}$$

Here, Π_0 and Π_1 represent the $n \times n$ transport plan of the corresponding data with n dimensions.

Figure 2.4 shows the comparisons of different mixup methods, which perform interpolation in the input space. Each of these methods contributes uniquely to enhancing representation learning by introducing various forms of regularization, encouraging feature diversity, promoting robustness, and guiding the model to learn more nuanced and informative representations.

The success of mixup has shown to be effective in the context of image classification tasks. These methods have shown to improve model generalization by creating synthetic training examples that encourage the network to learn more robust decision boundaries. However, the application of these augmentation approaches has been primarily focused on image classification and its downstream tasks such as object detection, semantic segmentation etc., leaving their potential in other domains relatively under-explored. One such area is instance-level retrieval tasks, where the goal is to learn an embedding space that can accurately retrieve semantically similar items. In the next section, we will examine how the core principles of mixup can be adapted and integrated into deep metric learning frameworks. We hypothesize that by applying mixup-based data augmentation directly on the embeddings, the model explores new regions of the embedding space beyond the training data, leading to more robust and discriminative representations.

2.2 Deep Metric Learning

Metric learning involves learning an embedding space where semantically similar items are clustered together, while dissimilar items are pushed apart. The two most studied

problems in metric learning are *loss functions* [**musgrave**] and *hard negative mining* [**Wu et al.**; **2017; Robinson et al.**; **2021**].

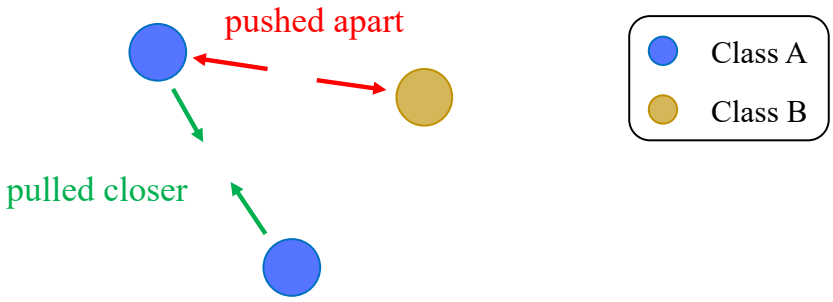


Figure 2.5 – Deep metric learning aims to learn a function, which pulls the embedding of similar classes together and pushes embeddings of dissimilar classes apart.

2.2.1 Metric learning loss functions

The loss functions in deep metric learning can be broadly categorized as pair-based losses and proxy-based losses. *Pair-based* losses [Hadsell, 2006; Wang, 2014; Oh Song, 2016; Wang, 2019b] aim to model relationships between pairs of examples (anchor-positives or anchor-negatives) to learn more discriminative embeddings with the gradients formulated as weighted pair-wise similarities. *Proxy-based* losses define one or more learnable *proxies* per class and only use proxies as anchors [Kim, 2020c] or positives/negatives [Movshovitz-Attias, 2017; Qian, 2019; Teh, 2020]. We illustrate some commonly used pair-based and proxy-based loss functions in deep metric learning in Figure 2.6, and discuss some popular loss functions.

Contrastive The contrastive loss [Hadsell, 2006] encourages positive examples to be pulled towards the anchor (a) and negative examples to be pushed away by a margin $m \in \mathbb{R}$. This loss is *additive* over positives (p) and negatives (n), defined as

$$\text{cont}(a;\theta) := \sum_{p \in P(a)} -s(a,p) + \sum_{n \in N(a)} [s(a,n) - m]_+. \tag{2.8}$$

Multi-Similarity The multi-similarity loss [Wang, 2019b] introduces *relative weighting* to encourage positives (negatives) that are farthest from (closest to) the anchor to be pulled towards (pushed away from) the anchor by a higher weight. This loss is *not* additive

over positives and negatives:

$$\mathrm{MS}(a;\theta) := \frac{1}{\beta}\log\left(1+\sum_{p\in P(a)} e^{-\beta(s(a,p)-m)}\right) + \frac{1}{\gamma}\log\left(1+\sum_{n\in N(a)} e^{\gamma(s(a,n)-m)}\right). \tag{2.9}$$

Here, $\beta, \gamma \in \mathbb{R}$ are scaling factors for positives, negatives respectively.

Proxy Anchor The proxy anchor loss [Kim, 2020c] defines a learnable *proxy* in $\mathbb{R}^d$ for each class and only uses proxies as anchors. For a given anchor (proxy) $a \in \mathbb{R}^d$, the loss has the same form as (4.2), although similarity s is evaluated on $\mathbb{R}^d \times \mathcal{X}$.

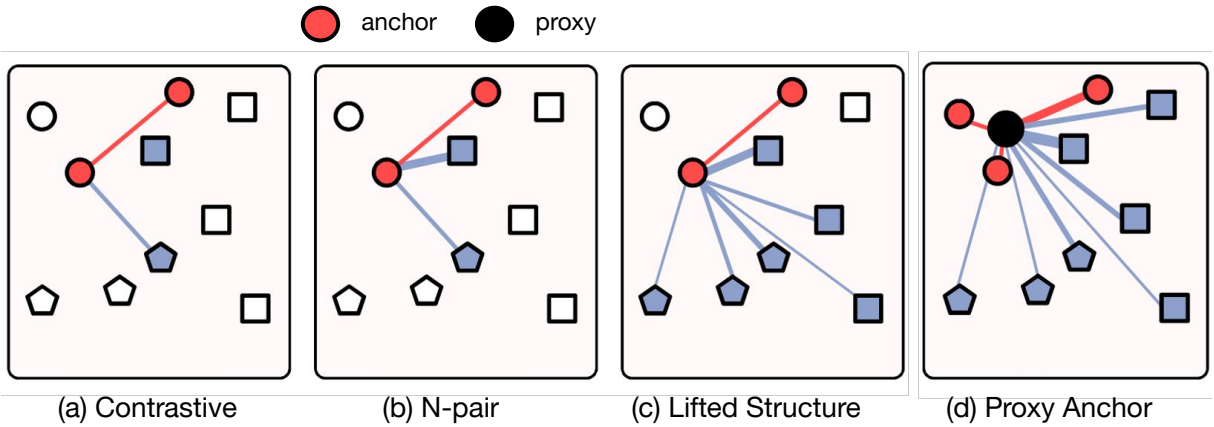


Figure 2.6 – A visual comparison of different pair-based losses (contrastive, N-pair and Lifted Structure) and Proxy based loss (Porxy Anchor) in deep metric learning. The thickness of the lines denotes the strength of the similarity between the examples. The images were adapted from [Kim, 2020c]

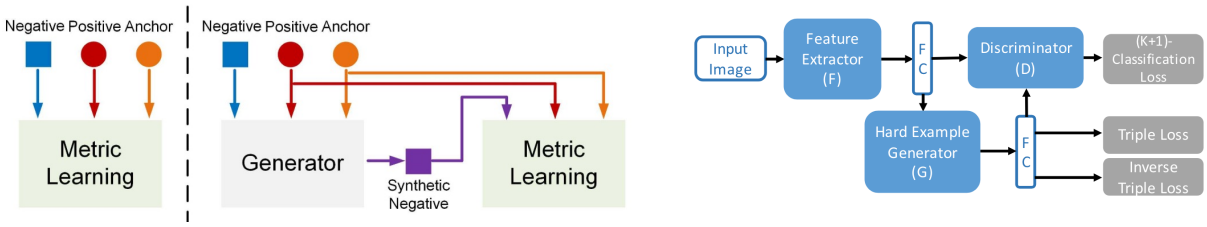
2.2.2 Hard negative mining

In metric learning, achieving well-separated clusters in the embedding space is challenging, as the network can easily overfit to the "easy" negative samples that are already far away from the anchor. Hard negative mining addresses this issue by identifying the most informative negative examples *i.e.* negatives that are closest to the anchor in the embedding space and thus the most difficult to distinguish. By focusing the training on these hard negatives, the network is encouraged to learn more discriminative features that can better separate similar-looking but semantically distinct items.

Selection of hard negatives [Iscen, 2018], focuses on mining hard negatives from a large set of negative examples. They define hard negatives as samples that are similar to the positive examples in the Euclidean feature space, but are not considered as such when using a manifold distance metric based on the nearest neighbor graph. [Chuang, 2020], present a debiased version of the contrastive loss function, which is designed to mitigate the effect of "false negatives" *i.e.* samples that are incorrectly labeled as negative examples. By addressing this bias, the authors aim to better approximate the "true" distribution of negative examples, leading to more robust and accurate contrastive learning models. [Wu, 2020], introduce a variational extension to the InfoNCE loss [Oord, 2018]. They propose a modified strategy for negative sampling, such as restricting the negative samples to a region around the query. In a concurrent work [Ho, 2020], propose to generate more hard positive and negative pairs on-the-fly, leveraging adversarial examples. This approach aims to create more informative training samples, leading to improved performance of contrastive learning models.

Relation between Hard negative mining and Loss functions Many popular metric learning loss functions, such as triplet loss [Vasudeva, 2021] and lifted structured loss [Oh Song, 2016], explicitly incorporate hard negative mining strategies. The triplet loss, for instance, aims to pull an anchor sample closer to a positive sample from the same class, while pushing it away from a negative sample from a different class by a specified margin. By focusing on the "hardest" negative samples - those that are closest to the anchor in the embedding space - the triplet loss encourages the model to learn more discriminative features. Similarly, the lifted structured loss extends the triplet loss by considering all negative pairs in a mini-batch, rather than just a single negative per anchor. This allows the model to leverage the global structure of the embedding space and focus on the "hardest" negatives across the entire batch.

The N-pair loss [Sohn, 2016] takes this a step further by directly optimizing the cosine similarity between the anchor and all negative samples in a batch in a probabilistic manner. Other loss functions, such as the multi-similarity loss [Wang, 2019b], go beyond just considering hard negatives and also weight the contribution of each negative sample based on its similarity to the anchor. This helps the model focus on the most informative negatives during training.



(a) Deep Adversarial Metric Learning (DAML)

(b) Hard Triplet Generation

Figure 2.7 – A visual comparison of different methods that generate synthetic samples using generative models for metric learning. The images were adapted from [Zhao, 2018] and [Zhao, 2018].

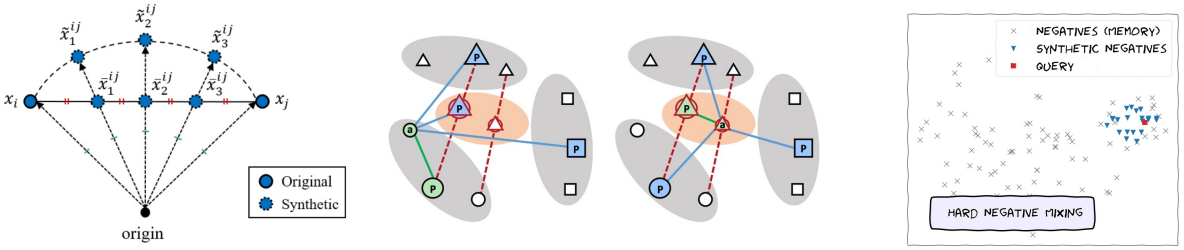
Generation of hard negatives Several works have explored the use of generative models to create synthetic samples as a way for mining hard negatives. We illustrate a few of these works in Figure 2.7. DAML [Duan, 2018] and HTG [Zhao, 2018] use GANs to generate synthetic hard negative samples. The key idea is to train a generator network to produce samples that are challenging negatives, i.e., they are close to the positive samples in the feature space. HDML [Zheng, 2019] takes a different approach by using an autoencoder to generate label-preserving synthetic samples. The main idea is the ability to control the "hardness" of the generated negative samples. The autoencoder is trained to produce negatives that are challenging, but still preserve the original label information. This allows the model to learn from a mix of easy and hard negative examples, leading to improved generalization.

While the above methods can provide a performance boost by training with synthetic samples, they come with some drawbacks. Specifically, these approaches require additional generative networks alongside the main metric learning framework. This can result in a larger model size, slower training time, and more complex optimization challenges.

2.2.3 Interpolation for pairwise loss functions

Unlike classification, classes (and distributions) at training and inference are different in metric learning. Thus, one might expect interpolation-based data augmentation like mixup to be even more important in metric learning than in classification. However, this task is challenging because unlike classification, the loss functions used in metric learning are not additive over examples, so the idea of interpolating target labels is not straightforward. We discuss recent works in deep metric learning that generate embeddings

using mixup.



(a) Embedding Expansion (b) Proxy Synthesis (c) MoCHi

Figure 2.8 – Different methods that perform mixup in the embedding space to generate more synthetic negatives for metric learning. The images were adapted from [Ko, 2020], [Kalantidis, 2020] and [Gu, 2021].

Embedding expansion [Ko, 2020] propose to generate synthetic samples by interpolating between embeddings withing the same class in a deterministic way, *i.e.* they interpolate between positive pairs only. This approach, called "embedding expansion", allows the model to learn a more robust distance metric by exposing it to a wider range of hard negative samples during training.

Proxy Synthesis [Gu, 2021] Instead of generating individual hard negative samples, proxy synthesis proposes to create new "proxy" classes by interpolating between existing class embeddings. These synthetic proxy classes are then used as additional training targets, forcing the model to learn more discriminative features to distinguish between the real and synthetic classes.

MoCHi [Kalantidis, 2020] The key idea is to create new hard negatives by linearly interpolating between existing hard negative and easy negative pairs. This hard negative mixing strategy encourages the model to learn a more discriminative representation by focusing on the challenging regions of the feature space.

A summary of these works is shown in Figure 2.8. While the interpolation-based data augmentation methods offer several advantages over GAN-based approaches, they also have some limitations. These techniques may not be as flexible in generating diverse and complex hard negative samples, as they are constrained by the distribution of the existing data. More importantly, they do not interpolate between positive and negative pairs,

limiting the labels to either positive or negative only. Furthermore, they risk synthesizing false negatives when the interpolation factor is close to 0 or 1.

While deep metric learning has shown great promise in learning effective embedding space, it has traditionally relied on the availability of labeled datasets. This requirement can be a significant limitation, as obtaining high-quality labeled data can be a time-consuming and expensive process, especially for complex domains. In the next section, we study how metric learning loss functions can be applied in an unsupervised/self-supervised setting. Additionally, we will discuss the impact of different data augmentation techniques on representation learning.

2.3 Self-supervised learning

Self-supervised learning (SSL) has emerged as a powerful alternative, offering a way to learn meaningful representations from unlabeled data. The main idea of SSL is to define pretext tasks, where the model is trained to solve a surrogate problem using the inherent structure and patterns present in the data itself, without the need for manual annotations. The connection between deep metric learning and self-supervised learning can be found in the use of the InfoNCE (Noise-Contrastive Estimation) loss function. The idea is to learn representations from unlabeled data by contrasting a data sample (anchor) with its augmented version (positive) against other samples in the batch (negatives). By maximizing the similarity between the anchor and its positive, while minimizing the similarity to the negatives, the model learns to capture the underlying structure and semantics of the data.

2.3.1 Contrastive Representation Learning

Data augmentation has been widely employed in both supervised and unsupervised representation learning, but it has not been systematically leveraged to define contrastive prediction tasks. Traditionally, many existing approaches have focused on defining contrastive prediction tasks by modifying the neural network architecture.

Global-to-local view prediction [Hjelm, 2018] and [Bachman, 2019] have achieved global-to-local view prediction by constraining the receptive field of the network architecture. This means that the model is trained to predict the local features of an image given its global representation.

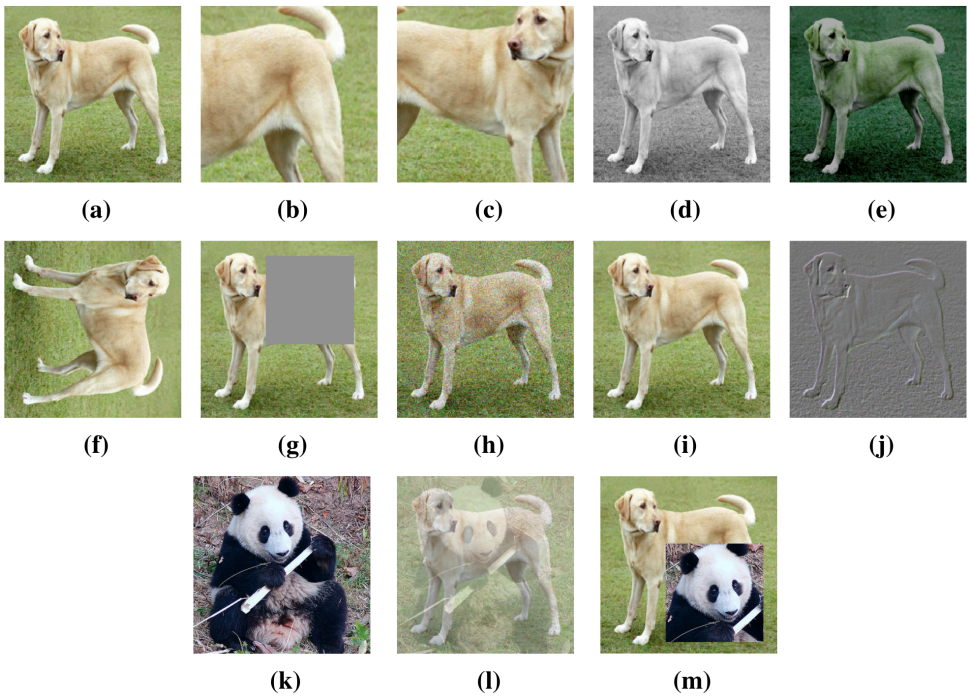


Figure 2.9 – Illustrations of the various data augmentation used in self-supervised learning. Given an example image in (a), the label-preserving augmentations from (b) to (j) are: (b) Crop + resize, (c) Crop + resize + flip, (d) Color distortion (channel drop), (e) Color distortion (jitter), (f) Rotation, (g) Cutout, (h) Gaussian noise, (i) Gaussian blur and (j) Sobel filtering. Provided another image in (k), semantic transformations of (a) and (k) are (l) MixUp and (m) CutMix. Each augmentation operation has one or more parameters determining the output image. The images in (a) to (j) are taken from [Chen, 2020b], and the image in (k) is from ImageNet [Russakovsky, 2015].

Neighboring view prediction In contrast, [Oord, 2018] and [Henaff, 2020] have approached the problem differently, using a fixed image splitting procedure and a context aggregation network to achieve neighboring view prediction. Here, the model is trained to predict the features of a neighboring image patch given the surrounding context.

Simplifying contrastive prediction tasks Architectural complexity can be avoided by employing a simpler data augmentation technique – simple random cropping (with resizing) of target images. This simple design choice of using random cropping as the data augmentation technique conveniently decouples the predictive task from other components, such as the neural network architecture. This allows for greater flexibility in defining and exploring a broader range of contrastive prediction tasks.

The random crop strategy shown in Figure 2.9 was introduced in SimCLR [Chen,

2020b] as part of its data augmentation approach for contrastive learning. In this method, two different augmentations of the same image are used as the positive pairs for the contrastive loss function. The authors show that combining random crop with color distortion as the augmentation for positive pairs helps the model learn robust visual representations, as it has to relate the local details in one crop to the broader context in the other.

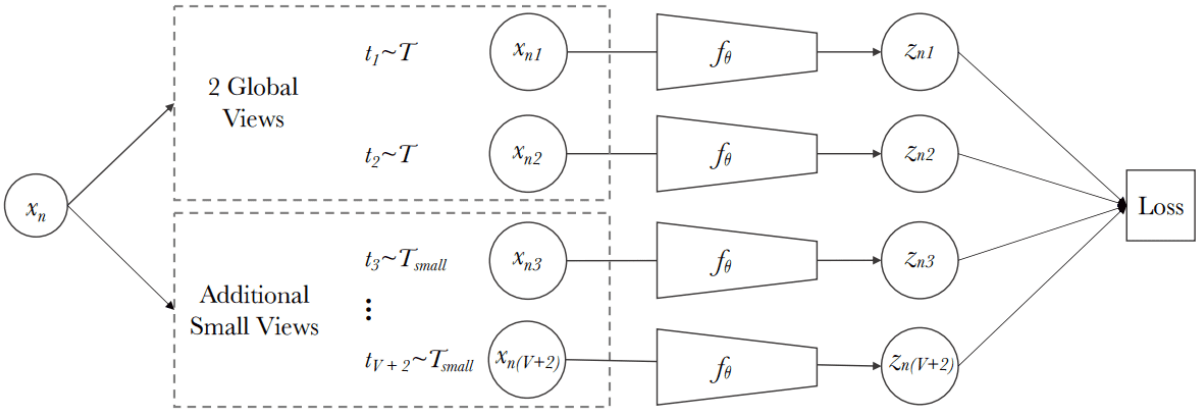


Figure 2.10 – The multi-crop strategy for data augmentation in self-supervised learning. The image x_n is transformed into $V + 2$ views: two global views and V small resolution zoomed views. This strategy has been widely adopted in DINO, IBoT etc. Image source [Caron, 2020].

Building on the random crop strategy from SimCLR, SwAV [Caron, 2020] proposed a "multi-crop" data augmentation technique shown in Figure 2.10. Instead of using only two full-resolution views (e.g. 224×224 crops), the SwAV approach takes a mix of global and local views of the same image. Specifically, the multi-crop strategy involves generating two global views (e.g. 224×224 crops) and 6 local views (e.g. 96×96 crops) from each input image. These multiple views of different resolutions are then fed into the SwAV model during training. Mapping small parts of a scene to more global views helps the model learn to relate local details to broader context, leading to more robust visual representations.

2.3.2 Non-contrastive and Masked Image Modelling

Non-contrastive representation learning like BYOL, DINO and SimSiam do not require negative samples and thereby avoid the need for positive and negative pairs. The core idea behind non-contrastive learning is to align positive pairs alone, without the need for negative samples. This is achieved through the use of asymmetric architectural designs,

such as a student-teacher framework in DINO, which encourages the student network to match the teacher's global features, even though the student only sees local patches.

DINO is a form of self-distillation, where there is a student and a teacher network with the same ViT architecture. The teacher network is a momentum encoder, meaning its weights are an exponentially weighted average of the student's weights. This helps stabilize the training process. It also uses a multi-crop training strategy, consisting of small "local" crops and large "global" crops. These crops are then passed through the student network, while only the global crops are passed through the teacher network. This encourages the student network to learn to interpolate context from the small local crops, aligning its representations with the teacher's global understanding of the image. Additionally, random data augmentations such as color jittering, Gaussian blur, and solarization are applied to make the network more robust.

DINO has some interesting properties. Firstly, it learns representations that are highly robust to various types of image transformations, such as rotation, scaling, and occlusion. This property enhances the model's ability to generalize to a wide range of tasks. Secondly, the self-attention maps of the `CLS` token in the last layer of the ViT show that the model has implicitly learns class-specific features, leading to unsupervised object discovery. This property does not emerge as clearly with supervised ViTs or CNNs.

Masked Image Modeling (MIM) approaches differ from DINO and SimCLR in their core objective - instead of learning representations through contrastive learning or clustering, MIM models learn by predicting the content of masked image patches. Another key distinction is the masking strategy. While contrastive learning approaches typically use data augmentation techniques like cropping, color jittering, flipping etc., MIM models rely on explicitly masking out a portion of the input image. The masking strategy can have a significant impact on the learned representations, with recent works exploring techniques like block-wise masking and attention-guided masking. These masking approaches aim to create a more challenging pretext task that encourages the model to learn richer visual features for dense prediction tasks.

One of the pioneering works in this direction is the Masked Autoencoder (MAE) [He, 2022]. MAE takes a partially masked image as input and trains an encoder-decoder model to reconstruct the original, unmasked image. The encoder maps the partial input to a latent representation, while the decoder tries to predict the missing pixel values. This reconstruction-based pretext task encourages the model to learn rich visual representa-

tions that can capture the underlying structure of the image. In contrast to MAE, other MIM methods have explored alternative reconstruction targets beyond raw pixel values. For example, BEiT [Bao, 2021] and iBOT [Zhou, 2022a] use a discrete visual tokenizer to predict the corresponding tokens for the masked patches, rather than directly predicting pixel values. SimMIM [Xie, 2022] and MaskFeat [Wei, 2022] employ a simpler linear prediction head to regress the RGB values of the masked patches. These approaches demonstrate that the specific choice of reconstruction target is not crucial, as long as the model is forced to learn meaningful representations to solve the pretext task.

2.4 Positioning the contributions

We position the contributions of this thesis (presented in chapter 3 to chapter 6) with respect to the related works discussed in this chapter. We tackle four main research questions as follows:

1. We have observed in subsection 2.1.3 that mixup has many interesting properties as compared to standard data augmentations *e.g.* it flattens class representations, reduces overly confident incorrect predictions etc. However, mixup images are overlays and tend to be unnatural as shown in Figure 2.4. The randomness in patch selection and label mixing may mislead the classifier to learn uninformative features. Hence, we ask the question *what is a good interpolation of images*?

   > We are motivated by the idea of deformation as a natural way of interpolating images, where one image may deform into another, in a continuous way. To achieve this, we introduce a novel mixup operation called *AlignMixup* in chapter 3, advocating interpolation of local structure in the feature space. Thus, instead of interpolating in image space, we investigate geometric alignment for mixup, based on explicit semantic correspondences in the feature space.

2. The task of extending the mixup technique to a different domain, such as metric learning, is challenging. This is because the loss functions used in metric learning are not additive over examples. As a result, the idea of interpolating target labels is not straightforward. Some previous works, as discussed in subsection 2.2.3 have extended this idea to deep metric learning or risk generating false negatives. However, they

do not perform label interpolation. Thus, we ask the question *what is a proper way to define and interpolate labels for metric learning*?

> We first define a generic way of representing and interpolating labels, which allows straightforward extension of any kind of mixup to deep metric learning. We develop our method on a generic formulation that encapsulates different loss functions. In chapter 4, we introduce *Metrix*, where we systematically evaluate mixup under different settings.

3. The application of mixup to deep metric learning has shown that one can interpolate between all positive and negative pairs along with their target labels, without constraining the interpolation factor. This has shown to increase the number of interpolated examples during training, going beyond the original mixup approach. Motivated by this observation and re-visiting the original idea of mixup, we ask *how can mixup be extended to generate a larger number of interpolated examples, beyond the original approach*, in order to enhance performance in classification tasks?

> In chapter 5, we introduce *MultiMix*, which generates an arbitrarily large number of interpolated examples beyond the mini-batch size in the embedding space. Specifically, we sample on the entire convex hull of the mini-batch embeddings, which acts as an implicit regularizer by enforcing linearity and smoothness constraints on the model's predictions in the embedding space.

4. Finally, we discuss moving beyond interpolation-based data augmentation techniques, such as mixup, to explore the potential of discovering natural augmentations inherent in real-world data, particularly in the context of self-supervised learning. Moving beyond the need for interpolation based augmentations like mixup or handcrafted augmentations like random-crop, color jitter etc., we ask the question *how can the richness and diversity of video data be leveraged to discover natural augmentations that can enhance representation learning*?

First, we introduce in chapter 6 a dataset of long (1-3 hour) open-source first-person videos recorded for virtual "walking tours". These videos offer advantages: high diversity of objects, 4K resolution, and contain few or no shot cuts. We then introduce *DoRA*, a new self-supervised image-pretraining method, aimed at learning from video frames. DoRA leads to emergent attention maps from the `CLS` token of distinct heads in a ViT to detect and track multiple objects within a given frame across temporal sequences. By tracking objects, we leverage the natural augmentations in videos to learn robust representations.

Chapter 3

Interpolating Aligned Features

Deep neural networks excel at making accurate predictions on the training data, but often provide incorrect and yet overly confident predictions when evaluated on slightly different test examples. This includes distribution shifts, outliers, and adversarial examples. *Mixup* addresses limitation by interpolating between pairs of examples (input/feature) and their target labels, effectively augmenting the training manifold. This improves model calibration, ensuring confidence scores accurately reflect prediction likelihood.

Several mixup methods propose non-linear interpolation in the input space *e.g.* masking square regions [DeVries, 2017b], cutting a rectangular region from one image and pasting it onto another [Yun, 2019], using *saliency* to locate objects from different images and fit them in one [Uddin, 2021; Qin, 2020; Kim, 2020a; Kim, 2021b] as well as several variants using arbitrary regions [Takahashi, 2018; Summers, 2019; Harris, 2020].

Manifold mixup [Verma, 2019] extends this concept from input space to feature space. By operating in the embedding space, the interpolation process can be decoupled from the specific input modalities. This allows for a more general approach applicable to various data types and potentially reduces computational overhead. However, images generated through non-linear interpolation in the input space or linear interpolation in feature space are overlays and tend to be unnatural. Furthermore, the randomness in the patch selection and thereby label mixing may mislead the classifier to learn uninformative features.

The seed of this chapter is to ask "*what is a good interpolation of images*?". Our motivation stems from the idea of *deformation* as a natural way of interpolating images, allowing for a continuous transformation from one image to another. [Bengio, 2013], show that traversing along the manifold of representations obtained from deeper layers of the network more likely results in finding realistic examples. Thus, instead of exploring interpolation in the input space, we study geometric alignment for mixup, relying on explicit semantic correspondences within the feature space. Specifically, we align the feature tensors of two images, resulting in soft correspondences. By choosing to keep the coordinates of one set or the other, we define an asymmetric operation. What we obtain is one object continuously morphing, rather than two objects in one image.

This effort was published in the proceedings of IEEE/CVF Computer Vision and Pattern Recognition (CVPR), 2022 and is the base of this chapter.

3.1 Introduction

Data augmentation [Krizhevsky, 2012; Paulin, 2014; Cubuk, 2019] is a powerful regularization method that increases the amount and diversity of data, be it labeled or unlabeled [Dosovitskiy, 2013]. It improves the generalization performance and helps learning invariance [Simard, 1998] at almost no cost, because the same example can be transformed in different ways over epochs. However, by operating on one image at a time and limiting to label-preserving transformations, it has limited chances of exploring beyond the image manifold. Hence, it is of little help in combating memorization of training data [Zhang, 2017] and sensitivity to adversarial examples [Szegedy, 2014].

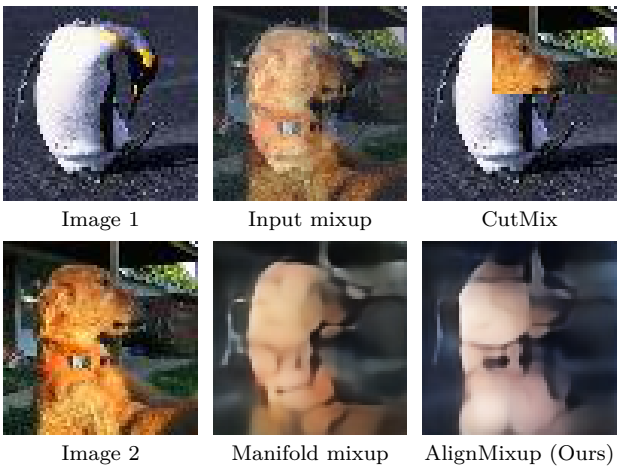


Figure 3.1 – Different mixup methods. AlignMixup retains the pose of image 2 and the texture of image 1. This different from overlay (Input [Zhang, 2018a] and Manifold mixup [Verma, 2019]) or combination of two objects (CutMix [Yun, 2019]). Manifold mixup and AlignMixup visualized by a decoder (subsection 3.3.3) that is not used at training.

Mixup operates on two or more examples at a time, *interpolating* between them in the input space [Zhang, 2018a] or feature space [Verma, 2019], while also interpolating between target labels for image classification. This flattens class representations [Verma, 2019], reduces overly confident incorrect predictions, and smoothens decision boundaries far away from training data. However, input mixup images are overlays and tend to be unnatural [Yun, 2019]. Interestingly, recent mixup methods focus of combining two [Yun,

2019; Kim, 2020a] or more [Kim, 2021b] objects from different images into one in the input space, making efficient use of training pixels. However, randomness in the patch selection and thereby label mixing may mislead the classifier to learn uninformative features [Uddin, 2021], which raises the question: *what is a good interpolation of images?*

[Bengio, 2013], show that traversing along the manifold of representations obtained from deeper layers of the network more likely results in finding realistic examples. This is because the interpolated points smoothly traverse the underlying manifold of the data, capturing salient characteristics of the two images. Furthermore, [Berthelot, 2018] show the ability of autoencoders to capture semantic correspondences obtained by decoding mixed latent codes. This is because the autoencoder may disentangle the underlying factors of variation. Efforts have followed on mixing latent representations of autoencoders to generate realistic images for data augmentation. However, these approaches are more expensive, requiring three networks (encoder, decoder, classifier) [Berthelot, 2018] and more complex, often also requiring an adversarial discriminator [Beckham, 2019; Liu, 2018b]. More importantly, they perform poorly compared to standard input mixup on large datasets [Liu, 2018b], due to the low quality of generated images.

In this work, we are motivated by the idea of *deformation* as a natural way of interpolating images, where one image may deform into another, in a continuous way. Contrary to previous efforts, we do not interpolate directly in the input space, we do not limit to vectors as latent codes and we do not decode. We rather investigate geometric *alignment* for mixup, based on explicit semantic correspondences in the feature space. In particular, we explicitly align the feature tensors of two images, resulting in soft correspondences. The tensors can be seen as sets of features with coordinates. Hence, each feature in one set can be interpolated with few features in the other.

By choosing to keep the coordinates of one set or the other, we define an *asymmetric* operation. What we obtain is one object continuously morphing, rather than two objects in one image. Interestingly, observing this asymmetric morphing reveals that we retain the *geometry* or *pose* of the image where we keep the coordinates and the *appearance* or *texture* of the other. Figure 3.1 illustrates that our method, *AlignMixup*, retains the *pose* of image 2 and the *texture* of image 1, which is different from existing mixup methods. Note that, as in manifold mixup, we *do not* decode, hence we are not concerned about the quality of generated images.

We make the following contributions:

1. We introduce a novel mixup operation, called *AlignMixup*, advocating interpolation

of local structure in the feature space (subsection 3.3.2). Feature tensors are ideal for alignment, giving rise to semantic correspondences and being of low resolution. Alignment is efficient by using *Sinkhorn distance* [Cuturi, 2013].

2. We also show that a *vanilla autoencoder* can further improve representation learning under mixup training, without the classifier seeing decoded clean or mixed images (section 3.4).
3. We set a new state-of-the-art on *image classification*, *robustness to adversarial attacks*, *calibration*, *weakly-supervised localization* and *out-of-distribution detection* against more sophisticated mixup operations on several networks and datasets (section 3.4).

3.2 Related Work

Mixup [Zhang, 2018a], concurrently with similar methods [Inoue, 2018; Tokozume, 2018], introduce *mixup*, augmenting data by linear interpolation between two examples. While [Zhang, 2018a] apply mixup on intermediate representations, it is [Verma, 2019] who make this work, introducing *manifold mixup*. Without alignment, the result is an overlay of either images [Zhang, 2018a] or features [Verma, 2019]. [Guo, 2019b] eliminate "manifold intrusion"—mixed data conflicting with true data. Unlike manifold mixup, AlignMixup interpolates feature tensors from deeper layers after aligning them.

Nonlinear mixing over random image regions is an alternative, *e.g.* from masking square regions [DeVries, 2017b] to cutting a rectangular region from one image and pasting it onto another [Yun, 2019], as well as several variants using arbitrary regions [Takahashi, 2018; Summers, 2019; Harris, 2020]. Instead of choosing regions at random, *saliency* can be used to locate objects from different images and fit them in one [Uddin, 2021; Qin, 2020; Kim, 2020a; Kim, 2021b]. Exploiting the knowledge of a teacher network to mix images based on saliency has been proposed in [Dabouei, 2021b]. Instead of combining more than one objects in an image, AlignMixup attempts to deform one object into another.

Another alternative is Automix [Zhu, 2020a], which employs a U-Net rather than an autoencoder, mixing at several layers. It is limited to small datasets and provides little improvement over manifold mixup [Verma, 2019]. StyleMix and StyleCutMix [Hong, 2021] interpolate content and style between two images, using AdaIN [Huang, 2017], a style transfer autoencoder network. By contrast, AlignMixup aligns feature tensors and interpolates matching features directly, without using any additional network.

Alignment Local correspondences from intra-class alignment of feature tensors have been used in *image registration* [Choy, 2016; Long, 2014], *optical flow* [Weinzaepfel, 2013], *semantic alignment* [Rocco, 2018; Han, 2017] and *image retrieval* [Siméoni, 2019]. Here, we mostly use *inter-class* alignment. In *few-shot learning*, local correspondences between query and support images are important in finding attention maps, used *e.g.* by CrossTransformers [Doersch, 2020] and DeepEMD [Zhang, 2020]. The *earth mover's distance* (EMD) [Rubner, 2000], or *Wasserstein metric*, is an instance of *optimal transport* [Villani, 2008], addressed by linear programming. To accelerate, [Cuturi, 2013] computes optimal matching by *Sinkhorn distance* with *entropic regularization*. This distance is widely applied between distributions in generative models [Genevay, 2018; Patrini, 2020].

EMD has been used for mixup in the input space, for instance *point mixup* for 3D point clouds [Chen, 2020c] and OptTransMix for images [Zhu, 2020a], which is the closest to our work. However, aligning coordinates only applies to images with clean background. We rather *align tensors in the feature space*, which is generic. We do so using the Sinkhorn distance, which is orders of magnitude faster than EMD [Cuturi, 2013].

3.3 AlignMixup

3.3.1 Preliminaries

Problem formulation Let (x, y) be an image $x \in \mathcal{X}$ with its one-hot encoded class label $y \in Y$, where $\mathcal{X}$ is the input image space, $Y = [0, 1]^k$ and k is the number of classes. The *encoder network* consists of two stages. The first is $F : \mathcal{X} \to \mathbb{R}^{c \times w \times h}$, which maps x to feature tensor $\mathbf{A} := F(x)$, where c is the number of channels and $w \times h$ is the spatial resolution. The second is $f : \mathbb{R}^{c \times w \times h} \to \mathbb{R}^d$, which maps tensor $\mathbf{A}$ to embedding $e := f(\mathbf{A})$. Finally, the *classifier* $g : \mathbb{R}^d \to \mathbb{R}^k$ maps the embedding e to the vector $p := g(e)$ of probabilities over classes.

Mixup We follow [Verma, 2019] in mixing the representations from different layers of the network, focusing on the deepest layers at or near the embedding. We are given two labeled images $(x, y), (x', y') \in \mathcal{X} \times Y$. We draw an *interpolation factor* $\lambda \in [0, 1]$ from $\text{Beta}(\alpha, \alpha)$ [Zhang, 2018a] and then we interpolate labels y, y' linearly by the *standard* mixup operator

$$\text{mix}_\lambda(y, y') := \lambda y + (1 - \lambda) y' \tag{3.1}$$

and inputs x, x' by the generic formula

$$\mathrm{Mix}_\lambda^{f_1,f_2}(x, x') := f_2(\mathrm{Mix}_\lambda(f_1(x), f_1(x')), \tag{3.2}$$

where Mix_λ is a mixup operator to be defined. This generic formula allows interpolation of the input, feature or embedding by decomposing the network mapping $f \circ F$ as $f_2 \circ f_1$ according to

$$\text{input } (x): \quad f_1 := \mathrm{id}, f_2 := f \circ F \tag{3.3}$$

$$\text{feature } (\mathbf{A}): \quad f_1 := F, f_2 := f \tag{3.4}$$

$$\text{embedding } (e): \quad f_1 := f \circ F, f_2 := \mathrm{id}, \tag{3.5}$$

where id is the identity mapping. For (3.3) and (3.5), we define Mix_λ in (3.2) as standard mixup mix_λ (3.1), like *input* [Zhang, 2018a] and *manifold mixup* [Verma, 2019], respectively; while for (3.4), we define Mix_λ as discussed in subsection 3.3.2.

By default, we train the encoder network and the classifier by using a classification loss L_c on the output of the classifier g for mixed examples along with the corresponding mixed labels:

$$L_c(g(\mathrm{Mix}_\lambda^{f_1,f_2}(x, x')), \mathrm{mix}_\lambda(y, y')), \tag{3.6}$$

where $L_c(p, y) := -\sum_{i=1}^k y_i \log p_i$ is the standard cross-entropy loss. More options using an autoencoder architecture are investigated in section 3.4.

3.3.2 Interpolation of aligned feature tensors

Alignment Alignment refers to finding a geometric correspondence between image elements before interpolation. The feature tensor is ideal for this purpose, because its spatial resolution is low, reducing the optimization cost, and allows for semantic correspondence, because features close to the classifier—the second encoder f are small. Importantly, we are not attempting to combine two or more objects into one image [Kim, 2020a], but put two objects in correspondence and then interpolate into one. We make no assumptions on the structure of input images in terms of objects and we use no ground truth correspondences.

Our feature tensor alignment is based on *optimal transport* theory [Villani, 2008] and *Sinkhorn distance* (SD) [Cuturi, 2013] in particular. Let $\mathbf{A} := F(x), \mathbf{A}' := F(x')$ be the $c \times w \times h$ feature tensors of images $x, x' \in \mathcal{X}$. We reshape them to $c \times r$ matrices A, A' by

flattening the spatial dimensions, where $r := hw$. Then, every column $a_j, a'_j \in \mathbb{R}^c$ of A, A' for $j = 1, \ldots, r$ is a feature vector representing corresponding to a spatial position in the original image x, x'. Let M be the $r \times r$ *cost matrix* with its elements being the pairwise distances of these vectors:

$$m_{ij} := \left\| a_i - a'_j \right\|^2 \tag{3.7}$$

for $i, j \in \{1, \ldots, r\}$. We are looking for a *transport plan*, that is, a $r \times r$ matrix $P \in U_r$, where

$$U_r := \{P \in \mathbb{R}_+^{r \times r} : P\mathbf{1} = P^\top \mathbf{1} = \mathbf{1}/r\} \tag{3.8}$$

and $\mathbf{1}$ is an all-ones vector in $\mathbb{R}^r$. That is, P is non-negative with row-wise and column-wise sum $1/r$, representing a joint probability over spatial positions of $\mathbf{A}, \mathbf{A}'$ with uniform marginals. It is chosen to minimize the expected pairwise distance of their features, as expressed by the linear cost function $\langle P, M \rangle$, under an entropic regularizer:

$$P^* = \arg\min_{P \in U_r} \langle P, M \rangle - \epsilon H(P), \tag{3.9}$$

where $H(P) := -\sum_{ij} p_{ij} \log p_{ij}$ is the entropy of P, $\langle \cdot, \cdot \rangle$ is Frobenius inner product and ϵ is a regularization coefficient. The optimal solution P^* is unique and can be found by forming the $r \times r$ *similarity* matrix $e^{-M/\epsilon}$ and then applying the Sinkhorn-Knopp algorithm [Knight, 2008], *i.e.*, iteratively normalizing rows and columns. A small ϵ leads to sparser P, which improves one-to-one matching but makes the optimization harder [Alvarez-Melis, 2018], while a large ϵ leads to denser P, causing more correspondences and poor matching.

Interpolation The *assignment matrix* $R := rP^*$ is a doubly stochastic $r \times r$ matrix whose element r_{ij} expresses the probability that column a_i of A corresponds to column a'_j of A'. Thus, we align A and A' as follows:

$$\widetilde{A} := A'R^\top \tag{3.10}$$

$$\widetilde{A}' := AR. \tag{3.11}$$

Here, column $\widetilde{a}_i$ of $c \times r$ matrix $\widetilde{A}$ is a convex combination of columns of A' that corresponds to the same column a_i of A. We reshape $\widetilde{A}$ back to $c \times w \times h$ tensor $\widetilde{\mathbf{A}}$ by expanding spatial dimensions and we say that $\widetilde{\mathbf{A}}$ represents $\mathbf{A}$ *aligned to* $\mathbf{A}'$. We then interpolate

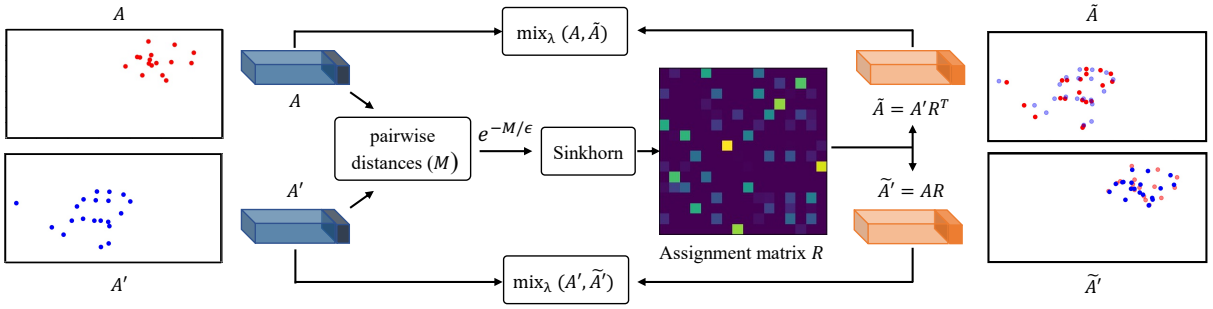


Figure 3.2 – *Feature tensor alignment and interpolation.* Cost matrix M contains pairwise distances of feature vectors in tensors $\mathbf{A}, \mathbf{A}'$. Assignment matrix R is obtained by Sinkhorn-Knopp [Knight, 2008] on similarity matrix $e^{-M/\epsilon}$. $\mathbf{A}$ is aligned to $\mathbf{A}'$ according to R, giving rise to $\widetilde{\mathbf{A}}$. We then interpolate between $\mathbf{A}, \widetilde{\mathbf{A}}$. Symmetrically, we can align $\mathbf{A}'$ to $\mathbf{A}$ and interpolate between $\mathbf{A}', \widetilde{\mathbf{A}}'$. $\mathbf{A}, \mathbf{A}'$ on the left (toy example of 16 points in 2D) shown semi-transparent on the right for reference.

between $\widetilde{\mathbf{A}}$ and the original feature tensor $\mathbf{A}$:

$$\text{mix}_\lambda(\mathbf{A}, \widetilde{\mathbf{A}}). \tag{3.12}$$

As shown in Figure 3.2 (toy example, top right), $\widetilde{\mathbf{A}}$ is geometrically close to $\mathbf{A}$. The correspondence with $\mathbf{A}'$ and the geometric proximity to $\mathbf{A}$ makes $\widetilde{\mathbf{A}}$ appropriate for interpolation with $\mathbf{A}$. Symmetrically, we can also *align* $\mathbf{A}'$ *to* $\mathbf{A}$ and interpolate between $\widetilde{\mathbf{A}}'$ and $\mathbf{A}'$:

$$\text{mix}_\lambda(\mathbf{A}', \widetilde{\mathbf{A}}'). \tag{3.13}$$

When mixing feature tensors with alignment (3.4), we define Mix_λ in (3.2) as the mapping of $(\mathbf{A}, \mathbf{A}')$ to either (3.12) or (3.13), chosen at random.

3.3.3 Visualization and discussion

Decoder We use a decoder to study images generated without or with feature alignment. Specifically, we use $f \circ F$ as an encoder and a *decoder* $D : \mathbb{R}^d \to \mathcal{X}$ maps the embedding e back to the image space, reconstructing image $\hat{x} := D(e)$. The autoencoder is trained using only clean images (without mixup) using *reconstruction* loss L_r between x and $\hat{x}$, where $L_r(x, x') := \|x - x'\|^2$ is the squared Euclidean distance. We use generated images only for visualization purposes below, but we also use the decoder optionally

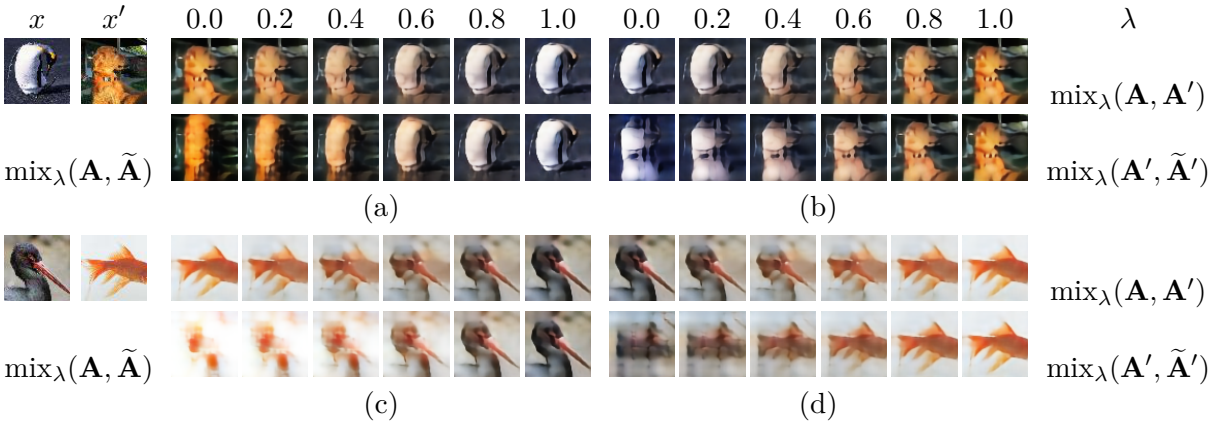


Figure 3.3 – *Visualizing alignment.* For different $\lambda \in [0, 1]$, we interpolate feature tensors $\mathbf{A}, \mathbf{A}'$ without alignment (top) or aligned feature tensors (bottom) of two images x, x' and then we generate a new image by decoding the resulting embedding through the decoder D. (a), (c) We align $\mathbf{A}$ to $\mathbf{A}'$ and mix with (3.12). (b), (d) We align $\mathbf{A}'$ to $\mathbf{A}$ and mix with (3.13). Only meant for illustration: No decoded images are seen by the classifier at training.

during AlignMixup training in section 3.4.

Discussion For different $\lambda \in [0, 1]$, we interpolate the feature tensors $\mathbf{A}, \mathbf{A}'$ of x, x' without or with alignment, using (3.12) or (3.13), and we generate a new image by decoding the resulting embedding through the decoder D.

In Figure 3.3, we visualize such generated images. Interestingly, by aligning $\mathbf{A}$ to $\mathbf{A}'$ and mixing using (3.12) with $\lambda = 0$, the generated image retains the pose of x and the texture of x'. In Figure 3.3(a) in particular, when x is 'penguin' and x' is 'dog', the generated image retains the pose of the penguin, while the texture of the dog aligns to the body of the penguin. Similarly, in Figure 3.3(c), the texture from the goldfish is aligned to that of the stork, while the pose of the stork is retained. Vice versa, as shown in Figure 3.3(b,d), by aligning $\mathbf{A}'$ to $\mathbf{A}$ and mixing using (3.13) with $\lambda = 0$, the generated image retains the pose of x' and the texture of x. By contrast, the image generated from unaligned features appears to be an overlay.

Randomly sampling several values of $\lambda \in [0, 1]$ during training generates an abundance of samples, capturing texture from one image and the pose from another. This allows the model to explore beyond the image manifold, thereby improving its generalization and enhancing its performance across multiple benchmarks, as discussed in section 3.4.

3.4 Experiments

3.4.1 Implementation details

Architecture

We use a residual network as the stage 1 encoder F. The output $\mathbf{A}$ is a $c \times 4 \times 4$ tensor. This is followed by a fully-connected layer as stage 2 encoder f with output embedding $e \in \mathbb{R}^d$ and another fully-connected layer as classifier g.

Autoencoder

In Figure 3.3, we have used a decoder to visualize the effect of feature tensor alignment. In our experiments, we also use a decoder optionally during training of AlignMixup, to investigate its effect on representation learning under mixup. This results in a vanilla autoencoder architecture, which we denote as AlignMixup/AE. We use a residual generator [Gulrajani, 2018] as the decoder D. The encoder and decoder have the same architecture.

Training

By default, we train AlignMixup using only the classification loss L_c (3.6) on mixed examples. For a given mini-batch during training, we mix either x, $\mathbf{A}$ (using either (3.12) or (3.13) for alignment) or e. We choose between the four cases uniformly at random. For AlignMixup/AE, we either use the reconstruction loss L_r on clean examples, training the encoder and decoder, or the classification loss L_c (3.6) on mixed examples, training the encoder and classifier. This gives rise to a fifth case and we choose uniformly at random.

3.4.2 Algorithm

AlignMixup and AlignMixup/AE are summarized in algorithm 1. By default (AlignMixup), for each mini-batch, we uniformly draw at random one among four choices (line 2) over mixup on input (x), embeddings (e), or feature tensors ($\mathbf{A}$, using either (3.12) or (3.13) for mixing). For AlignMixup/AE, there is a fifth choice where we only use reconstruction loss on clean examples (line 7).

For mixup, use only classification loss (3.6) (line 27). Following [Verma, 2019], we form, for each example (x, y) in the mini-batch, a paired example (x', y') from the same

mini-batch regardless of class labels, by randomly permuting the indices (lines 1,10). Inputs x, x' are mixed by (3.2),(3.3) (line 12) and embeddings e, e' by (3.2),(3.5) (line 14). Feature tensors $\mathbf{A}$ and $\mathbf{A}'$ are first aligned and then mixed by (3.2),(3.12) ($\mathbf{A}$ aligns to $\mathbf{A}'$) or (3.2),(3.13) ($\mathbf{A}'$ aligns to $\mathbf{A}$) (lines 17,26).

In computing loss derivatives, we backpropagate through embeddings e, e' or feature tensors $\mathbf{A}, \mathbf{A}'$ but not through the transport plan P^* (line 23). Hence, although the Sinkhorn-Knopp algorithm [Knight, 2008] is differentiable, its iterations take place only in the forward pass. Importantly, AlignMixup is easy to implement and does not require sophisticated optimization like [Kim, 2020a; Kim, 2021b].

Hyperparameters

CIFAR-10/CIFAR-100 We train AlignMixup using SGD for 2000 epochs with an initial learning rate of 0.1, decayed by a factor 0.1 every 500 epochs. We set the momentum as 0.9 with a weight decay of 0.0001 and use a batch size of 128. The interpolation factor is drawn from Beta(α, α) where $\alpha = 2.0$. Using these settings, we reproduce the results of SOTA mixup methods for image classification, robustness to FGSM and PGD attacks, calibration and out-of-distribution detection. For alignment, we apply the Sinkhorn-Knopp algorithm [Knight, 2008] for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

TinyImagenet We follow the training protocol of Kim *et al.* [Kim, 2020a], training R-18 as stage-1 encoder F using SGD for 1200 epochs. We set the initial learning rate to 0.1 and decay it by 0.1 at 600 and 900 epochs. We set the momentum as 0.9 with a weight decay of 0.0001 and use a batch size of 128 on 2 GPUs. The interpolation factor is drawn from Beta(α, α) where $\alpha = 2.0$. For alignment, we apply the Sinkhorn-Knopp algorithm [Knight, 2008] for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

ImageNet We follow the training protocol of Kim *et al.* [Kim, 2020a], where training R-50 as F using SGD for 300 epochs. The initial learning rate of the classifier and the remaining layers is set to 0.1 and 0.01, respectively. We decay the learning rate by 0.1 at 100 and 200 epochs. We set the momentum as 0.9 with a weight decay of 0.0001 and use a batch size of 100 on 4 GPUs. The interpolation factor is drawn from Beta(α, α) where $\alpha = 2.0$. For alignment, we apply the Sinkhorn-Knopp algorithm [Knight, 2008] for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

Algorithm 1: AlignMixup/AE (parts involved in the AE variant indicated in blue)

Input: encoders F, f; decoder D; classifier g
Input: mini-batch $B := \{(x_i, y_i)\}_{i=1}^{b}$
Output: loss values $L := \{\ell_i\}_{i=1}^{b}$

$\pi \sim \text{unif}(S_b)$ ▷ random permutation of $\{1, \dots, b\}$
$mode \sim \text{unif}\{\textit{clean}, \textit{input}, \textit{embed}, \textit{feat}, \textit{feat}'\}$ ▷ mixup?
for $i \in \{1, \dots, b\}$ **do**
  $(x, y) \leftarrow (x_i, y_i)$ ▷ current example
  if $mode = \textit{clean}$ **then** ▷ no mixup
    $\hat{x} \leftarrow D(f(F(x)))$ ▷ encode/decode
    $\ell_i \leftarrow L_r(x, \hat{x})$ ▷ reconstruction loss
  else ▷ mixup
    $\lambda \sim \text{Beta}(\alpha, \alpha)$ ▷ interpolation factor
    $(x', y') \leftarrow (x_{\pi(i)}, y_{\pi(i)})$ ▷ paired example
    if $mode = \textit{input}$ **then** ▷ as in [Zhang, 2018a]
      $e \leftarrow f(F(\text{mix}_\lambda(x, x')))$ ▷ (3.2),(3.3)
    else if $mode = \textit{embed}$ **then** ▷ as in [Verma, 2019]
      $e \leftarrow \text{mix}_\lambda(f(F((x)), f(F(x')))$ ▷ (3.2),(3.5)
    else ▷ $mode \in \{\textit{feat}, \textit{feat}'\}$
      if $mode = \textit{feat}'$ **then** ▷ choose (3.13) over (3.12)
        SWAP (x, x'), SWAP (y, y')
      $\mathbf{A} \leftarrow F(x)$, $\mathbf{A}' \leftarrow F(x')$ ▷ feature tensors
      $A \leftarrow \text{RESHAPE}_{c \times r}(\mathbf{A})$ ▷ to matrix
      $A' \leftarrow \text{RESHAPE}_{c \times r}(\mathbf{A}')$
      $M \leftarrow \text{DIST}(A, A')$ ▷ pairwise distances (3.7)
      $P^* \leftarrow \text{SINKHORN}(\exp(-M/\epsilon))$ ▷ tran. plan (3.9)
      $R \leftarrow \text{DETACH}(rP^*)$ ▷ assignments
      $\widetilde{A} \leftarrow A' R^\top$ ▷ alignment (3.10)
      $\widetilde{\mathbf{A}} \leftarrow \text{RESHAPE}_{c \times w \times h}(\widetilde{A})$ ▷ to tensor
      $e \leftarrow f(\text{mix}_\lambda(\mathbf{A}, \widetilde{\mathbf{A}}))$ ▷ (3.2),(3.12)
    $\ell_i \leftarrow L_c(g(e), \text{mix}_\lambda(y, y'))$ ▷ classification loss (3.6)

Dataset	Cifar-10		Cifar-100		TI
Network	R-18	W16-8	R-18	W16-8	R-18
Baseline	5.19	5.11	23.24	20.63	43.40
Input [Zhang, 2018a]	4.03	3.98	20.21	19.88	43.48
CutMix [Yun, 2019]	3.27	3.54	19.37	19.71	43.11
Manifold [Verma, 2019]	2.95	3.56	19.80	19.23	40.76
PuzzleMix [Kim, 2020a]	2.93	**2.99**	20.01	19.25	36.52
Co-Mixup [Kim, 2021b]	2.89	3.04	19.81	19.57	35.85
SaliencyMix [Uddin, 2021]	2.99	3.53	19.69	19.59	33.81
StyleMix [Hong, 2021]	3.76	3.89	20.04	20.45	36.13
StyleCutMix [Hong, 2021]	3.06	3.12	19.34	19.28	33.49
AlignMixup (ours)	2.95	3.09	18.08	18.67	**31.81**
AlignMixup/AE (ours)	**2.83**	3.15	**17.82**	**18.09**	32.73
Gain	**+0.06**	**-0.10**	**+1.52**	**+1.14**	**+1.68**

Table 3.1 – *Image classification* top-1 error (%) on CIFAR-10/100 and TI (TinyImagenet). R: PreActResnet, W: WRN.

We also train R-50 on ImageNet for 100 epochs, following the training protocol described in Kim *et al.* [Kim, 2021b].

CUB200-2011 For weakly-supervised object localization (WSOL), we use VGG-GAP and R-50 pretrained on ImageNet as F. The training strategy for WSOL is the same as image classification and the network is trained *without bounding box information*. In R-50, following [Yun, 2019], we modify the last residual block (`layer 4`) to have stride 2 instead of 1, resulting in a feature map of spatial resolution 14×14. The modified architecture of VGG-GAP is the same as described in [Zhou, 2016]. The classifier is modified to have 200 classes instead of 1000.

For fair comparisons with [Yun, 2019], during training, we resize the input image to 256×256 and randomly crop the resized image to 224×224. During testing, we directly resize to 224×224. We train the network for 600 epochs using SGD. For R-50, the initial learning rate of the classifier and the remaining layers is set to 0.01 and 0.001, respectively. For VGG, the initial learning rate of the classifier and the remaining layers is set to 0.001 and 0.0001, respectively. We decay the learning rate by 0.1 every 150 epochs. The momentum is set to 0.9 with weight decay of 0.0001 and batch size of 16.

3.4.3 Image classification and robustness

We use PreActResnet18 [He, 2016b] (R-18) and WRN16-8 [Zagoruyko, 2016b] as the backbone architecture on CIFAR-10 and CIFAR-100 datasets [Krizhevsky, 2009]. Using our experimental settings, we reproduce the state-of-the-art (SOTA) mixup methods: Baseline network (without mixup), Input mixup [Zhang, 2018a], Manifold mixup [Verma, 2019], CutMix [Yun, 2019], PuzzleMix [Kim, 2020a], Co-Mixup [Kim, 2021b], SaliencyMix [Uddin, 2021], StyleMix [Hong, 2021] and StyleCutMix [Hong, 2021] using official code provided by the authors. We do not compare AlignMixup with AutoMix [Zhu, 2020a] and Re-Mix [Cao, 2021], since its experimental settings are different from ours and there is no available code.

In addition, we use R-18 as the backbone network on TinyImagenet [Yao, 2015] (TI) and reproduce SaliencyMix [Uddin, 2021], StyleMix [Hong, 2021] and StyleCutMix [Hong, 2021] following the experimental settings of [Kim, 2021b], and Resnet-50 (R-50) on ImageNet [Russakovsky, 2015], following the training protocol of [Kim, 2020a]. Using top-1 error (%) as evaluation metric, we show the effectiveness of AlignMixup on image classification and robustness to FGSM [Goodfellow, 2015] and PGD [Madry, 2018] attacks.

Image classification As shown in Table 3.1, AlignMixup and AlignMixup/AE is on par or outperforms the SOTA methods by achieving the lowest top-1 error, especially on large datasets. On CIFAR-10, AlignMixup and AlignMixup/AE is on par with Co-Mixup and Puzzlemix with R-18 and WRN16-8. On CIFAR-100, AlignMixup outperforms StyleCutMix and Manifold mixup by 1.52% and 1.14% with R-18 and WRN16-8, respectively. On TI, AlignMixup outperforms Co-Mixup by 3.12% using R-18. From Table 3.2, AlignMixup/AE outperforms PuzzleMix by 2.41% on ImageNet. Importantly, while the overall improvement by SOTA methods on ImageNet over Baseline is around 2%, AlignMixup improves SOTA by another 2.5%.

Computational complexity Table 3.2 shows the computational analysis of AlignMixup training as compared with baseline and SOTA mixup methods on ImageNet, in terms of number of parameters and msec/batch on a NVIDIA RTX 2080 TI GPU. AlignMixup has nearly the same computational overhead as Manifold mixup while achieving 3.16% increase of accuracy. While SOTA methods like Co-Mixup and PuzzleMix are computationally more expensive than AlignMixup by 1.8× and 2.3× respectively, they are outperformed by AlignMixup by 1% on average. AlignMixup/AE brings a further 1.49%

METHOD	PARAM.	MSEC/BATCH	TOP-1 ERROR
Baseline	25M	418	23.68
Input† [Zhang, 2018a]	25M	436	22.58
CutMix† [Yun, 2019]	25M	427	21.40
Manifold† [Verma, 2019]	25M	441	22.50
PuzzleMix† [Kim, 2020a]	25M	846	21.24
Co-Mixup* [Kim, 2021b]	25M	1022	–
SaliencyMix* [Uddin, 2021]	25M	462	21.26
StyleMix* [Hong, 2021]	25M	828	-
StyleCutMix* [Hong, 2021]	25M	912	-
AlignMixup (ours)	28M	450	20.32
AlignMixup/AE (ours)	35M	688	**18.83**
Gain			**+2.41**

Table 3.2 – *Image classification* top-1 error (%) and *computational analysis* on ImageNet using Resnet-50. $*$: reported by authors; †: reported by PuzzleMix.

gain in accuracy over AlignMixup. It is important to note that 40% increase in number of parameters of AlignMixup/AE is due to the residual decoder, which is only used in one out of five cases on clean images without mixup. Computational complexity during inference is the same for all methods.

Challenges From Table 3.1, we observe that AlignMixup achieves SoTA top-1 error on CIFAR-10 and CIFAR-100. These results are computed using 2000 epochs following [Verma, 2019], which also achieves its best performance at 2000 epochs. While baseline mixup methods [Zhang, 2018a; Yun, 2019; Kim, 2020a; Kim, 2021b; Uddin, 2021; Hong, 2021] perform best at 300 epochs, they do not benefit from long training time. Unlike these methods, which perform mixup in the image space, Manifold mixup [Verma, 2019] and AlignMixup performs mixup in the feature space. We hypothesize that this takes longer training time until the network learns some meaningful representations. It is even more challenging in our case, since we mix features at deeper layers comparing with Manifold mixup. Empirically, when trained for 2000 epochs instead of 300 epochs, the top-1 error drops from 21.64 $\rightarrow$ 19.80 for Manifold mixup and from 21.38 $\rightarrow$ 18.08 for AlignMixup.

Robustness to FGSM and PGD attacks Following the evaluation protocol of [Kim, 2020a], we use 8/255 l_∞ ϵ-ball for FGSM and 4/255 l_∞ ϵ-ball with step size 2/255 for PGD. We reproduce the results of competitors for FGSM and PGD on CIFAR-10 and CIFAR-

Attack	FGSM					PGD			
Dataset	Cifar-10		Cifar-100		TI	Cifar-10		Cifar-100	
Network	R-18	W16-8	R-18	W16-8	R-18	R-18	W16-8	R-18	W16-8
Baseline	89.41	88.02	87.12	72.81	91.85	99.99	99.94	99.97	99.99
Input [Zhang, 2018a]	78.42	79.21	81.30	67.33	88.68	99.77	99.43	99.96	99.37
CutMix [Yun, 2019]	77.72	78.33	86.96	60.16	88.68	99.82	98.10	98.67	97.98
Manifold [Verma, 2019]	77.63	76.11	80.29	56.45	89.25	97.22	98.49	99.66	98.43
PuzzleMix [Kim, 2020a]	41.11	50.73	78.70	57.77	83.91	97.73	97.00	96.42	95.28
Co-Mixup [Kim, 2021b]	40.19	48.93	77.61	56.59	–	97.59	96.19	95.35	94.23
SaliencyMix [Uddin, 2021]	57.43	68.10	77.79	58.10	81.16	97.51	97.04	95.68	93.76
StyleMix [Hong, 2021]	79.54	71.05	80.54	67.94	84.93	98.23	97.46	98.39	98.24
StyleCutMix [Hong, 2021]	38.79	46.12	77.49	56.83	80.59	97.87	96.70	93.88	93.78
AlignMixup (ours)	38.33	53.41	77.29	**55.05**	**77.34**	**96.36**	96.73	**93.18**	**92.16**
AlignMixup/AE (ours)	**32.13**	**44.86**	**76.40**	55.44	78.98	97.16	**95.32**	93.69	**92.23**
Gain	**+6.66**	**+1.26**	**+1.09**	**+1.40**	**+3.25**	**+0.86**	**+0.87**	**+0.70**	**+1.40**

Table 3.3 – *Robustness to FGSM & PGD attacks.* Top-1 error (%): lower is better. Blue: second best. Gain: reduction of error. TI: TinyImagenet. R: PreActResnet, W: WRN.

100; results of baseline, Input, Manifold, Cutmix and Puzzlemix on TI for FGSM are as reported in [Kim, 2020a] and reproduced for SaliencyMix, StyleMix and StyleCutMix.

As shown in Table 3.3, AlignMixup is more robust comparing to SOTA methods. While AlignMixup is on par with PuzzleMix and Co-Mixup on CIFAR-10 image classification, it outperforms Co-Mixup and PuzzleMix by 8.06% and 8.98% in terms of robustness to FGSM attacks. There is also significant gain of robustness to FGSM on Tiny-ImageNet and to the stronger PGD on CIFAR-100.

3.4.4 Overconfidence

Deep neural networks tend to be overconfident about incorrect predictions far away from the training data and mixup helps combat this problem. Two standard benchmarks to evaluate this improvement are their ability to detect *out-of-distribution* data and their *calibration*, *i.e.*, the discrepancy between accuracy and confidence.

Out-of-distribution detection *In-distribution* (ID) refers to a test example drawn from the same distribution which the network is trained on, while a sample drawn from any other distribution is *out-of-distribution* (OOD) [Hendrycks, 2017]. At inference, given a mixture of ID and OOD examples, the network assigns probabilities to the known classes by softmax. An example is then classified as OOD if the maximum class probability is below a certain threshold, else ID. A well-calibrated network should be able to assign a higher probability to ID than OOD examples, making it easier to distinguish the two

Task	Out-Of-Distribution Detection											
Dataset	LSUN (crop)				iSUN				TI (crop)			
Metric	Det Acc	AuROC	AuPR (ID)	AuPR (OOD)	Det Acc	AuROC	AuPR (ID)	AuPR (OOD)	Det Acc	AuROC	AuPR (ID)	AuPR (OOD)
Baseline	54.0	47.1	54.5	45.6	66.5	72.3	74.5	69.2	61.2	64.8	67.8	60.6
Input [Zhang, 2018a]	57.5	59.3	61.4	55.2	59.6	63.0	60.2	63.4	58.7	62.8	63.0	62.1
Cutmix [Yun, 2019]	63.8	63.1	61.9	63.4	67.0	76.3	81.0	77.7	70.4	84.3	87.1	80.6
Manifold [Verma, 2019]	58.9	60.3	57.8	59.5	64.7	73.1	80.7	76.0	67.4	69.9	69.3	70.5
PuzzleMix [Kim, 2020a]	64.3	69.1	80.6	73.7	73.9	77.2	79.3	71.1	71.8	76.2	78.2	81.9
Co-Mixup [Kim, 2021b]	70.4	75.6	82.3	70.3	68.6	80.1	82.5	75.4	71.5	84.8	86.1	80.5
SaliencyMix [Uddin, 2021]	68.5	79.7	82.2	64.4	65.6	76.9	78.3	79.8	73.3	83.7	87.0	82.0
StyleMix [Hong, 2021]	62.3	64.2	70.9	63.9	61.6	68.4	67.6	60.3	67.8	73.9	71.5	78.4
StyleCutMix [Hong, 2021]	70.8	78.6	83.7	74.9	70.6	82.4	83.7	76.5	75.3	82.6	82.9	78.4
AlignMixup (ours)	76.1	80.7	85.9	75.8	73.4	**85.1**	84.3	80.2	79.4	85.0	88.4	85.0
AlignMixup/AE (ours)	**76.9**	**83.5**	**86.7**	**79.4**	**75.6**	84.1	**85.9**	**81.7**	**79.7**	**88.0**	**89.7**	**85.7**
Gain	+6.1	+3.8	+3.0	+4.5	+1.7	+2.7	+2.2	+1.9	+4.4	+3.2	+2.6	+3.8

Table 3.4 – *Out-of-distribution detection* using PreActResnet18. Det Acc (detection accuracy), AuROC, AuPR (ID) and AuPR (OOD): higher is better; Blue: second best. Gain: increase in performance. TI: TinyImagenet.

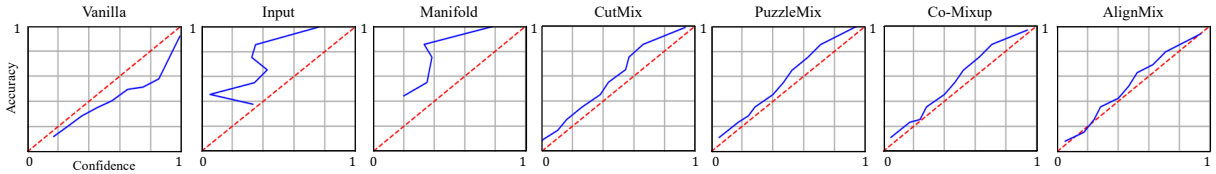


Figure 3.4 – *Calibration* plots on CIFAR-100 using PreActResnet18: near diagonal is better. Baseline is clearly overconfident while Input and Manifold mixup are clearly underconfident. AlignMixup has the best calibrated predictions.

distributions.

We compare AlignMixup with SOTA methods trained using R-18 on CIFAR-100 as discussed in subsection 3.4.3. At inference, ID examples are test images from CIFAR-100, while OOD examples are test images from LSUN (crop) [Yu, 2015], iSUN [Xiao, 2010] and Tiny-ImageNet (crop); where crop denotes that the OOD examples are center-cropped to 32×32 to match the resolution of ID images [Yun, 2019]. We also use test images from CIFAR-100 with Uniform and Gaussian noise as OOD samples. Uniform is drawn from $\mathcal{U}(0, 1)$ and Gaussian from $\mathcal{N}(\mu, \sigma)$ with $\mu = \sigma = 0.5$. Following [Hendrycks, 2017], we measure *detection accuracy* (Det Acc) using a threshold of 0.5, *area under ROC curve* (AuROC) and *area under precision-recall curve* (AuPR).

As shown in Table 3.4 and Table 3.5, AlignMixup outperforms SOTA methods under all metrics by a large margin, indicating that it is better in reducing over-confident predictions. We further observe that Input mixup is inferior to Baseline, which is consistent with the findings of [Yun, 2019].

Dataset	LSUN (resize)				TI (resize)			
Metric	Det Acc	Au ROC	AuPR (ID)	AuPR (OOD)	Det Acc	Au ROC	AuPR (ID)	AuPR (OOD)
Baseline	67.6	73.3	76.6	68.9	65.1	70.6	73.1	67.1
Input [Zhang, 2018a]	61.5	66.5	66.4	65.8	59.6	63.8	63.0	63.4
Cutmix [Yun, 2019]	71.3	77.4	79.1	75.5	69.1	79.4	79.8	73.3
Manifold [Verma, 2019]	67.8	78.9	76.3	71.3	62.5	77.8	76.8	72.2
PuzzleMix [Kim, 2020a]	74.9	79.9	84.0	77.5	73.9	77.3	80.6	71.9
Co-Mixup [Kim, 2021b]	73.8	82.6	86.8	76.9	68.1	78.9	82.5	74.2
SaliencyMix [Uddin, 2021]	75.8	79.7	82.2	84.4	75.3	81.2	83.8	79.5
StyleMix [Hong, 2021]	73.0	74.6	72.4	73.4	72.9	79.5	78.2	74.6
StyleCutMix [Hong, 2021]	74.3	83.1	86.9	78.9	73.8	80.9	83.1	76.3
AlignMixup (ours)	76.1	84.3	87.1	**85.8**	74.7	82.6	86.1	80.9
AlignMixup/AE (ours)	**77.0**	**85.8**	**87.9**	83.7	**76.2**	**84.8**	**87.2**	**82.3**
Gain	**+2.1**	**+2.7**	**+1.0**	**+1.4**	**+0.9**	**+3.6**	**+3.4**	**+2.8**
Noise	Uniform				Gaussian			
Baseline	58.3	75.3	75.0	69.0	60.8	64.3	62.9	63.9
Input [Zhang, 2018a]	50.0	67.9	71.8	71.7	60.2	65.0	63.1	64.1
Cutmix [Yun, 2019]	74.8	80.0	84.9	72.4	75.7	79.0	84.0	70.9
Manifold [Verma, 2019]	69.8	75.9	83.2	71.9	70.8	78.8	81.3	71.6
PuzzleMix [Kim, 2020a]	78.6	85.2	86.0	74.4	78.5	85.1	85.9	74.3
Co-Mixup [Kim, 2021b]	80.4	87.6	87.4	75.2	81.6	78.6	89.5	74.2
SaliencyMix [Uddin, 2021]	83.1	87.4	89.1	76.6	82.4	85.4	81.1	**81.3**
StyleMix [Hong, 2021]	75.3	71.8	77.8	65.5	78.0	75.2	84.3	71.0
StyleCutMix [Hong, 2021]	84.5	83.2	88.6	78.3	84.8	81.9	83.3	73.9
AlignMixup (ours)	86.9	89.1	93.6	77.7	**86.7**	**87.9**	91.8	77.4
AlignMixup/AE (ours)	**88.0**	**90.6**	**94.0**	**80.8**	86.0	87.2	**91.9**	75.6
Gain	**+3.5**	**+3.0**	**+4.9**	**+2.5**	**+1.9**	**+2.8**	**+2.4**	**-3.9**

Table 3.5 – *OOD detection* using PreActResnet18. Det Acc (detection accuracy), AuROC, AuPR (ID) and AuPR (OOD): higher is better. Blue: second best. Gain: increase in performance. TI: TinyImagenet.

Metric	ECE	OE
Baseline	10.25	1.11
Input [Zhang, 2018a]	18.50	1.42
CutMix [Yun, 2019]	7.60	1.05
Manifold [Verma, 2019]	18.41	0.79
PuzzleMix [Kim, 2020a]	8.22	0.61
Co-Mixup [Kim, 2021b]	5.83	0.55
SaliencyMix [Uddin, 2021]	5.89	0.59
StyleMix [Hong, 2021]	11.43	1.31
StyleCutMix [Hong, 2021]	9.30	0.87
AlignMixup (ours)	5.78	**0.41**
AlignMixup/AE (ours)	**5.06**	0.48
Gain	**+0.77**	**+0.14**

Table 3.6 – *Calibration* using PreActResnet18 on CIFAR-100. ECE : expected calibration error; OE: overconfidence error. Lower is better. Blue: second best. Gain: reduction of error.

Calibration According to [DeGroot, 1983], calibration measures the discrepancy between the accuracy and confidence level of a network's predictions. A poorly calibrated network may make incorrect predictions with high confidence.

As shown in Figure 3.4, while SOTA methods are under-confident compared to Baseline, AlignMixup results in the best calibration among all competitors. We quantitatively evaluate the calibration of AlignMixup against SOTA methods in terms of *expected calibration error* (ECE) [Guo, 2017] and *overconfidence error* (OE) [Thulasidasan, 2019] using R-18 on CIFAR-100. As shown in Table 3.6, AlignMixup outperforms SOTA methods by achieveing lower ECE and OE, indicating that it is better calibrated.

3.4.5 Weakly-supervised object localization (WSOL)

WSOL aims to localize an object of interest using only class labels *without bounding boxes* at training. WSOL works by extracting visually discriminative cues to guide the classifier to focus on salient regions in the image.

We train AlignMixup using the same procedure as for image classification. At inference, following [Yun, 2019], we compute a saliency map using CAM [Zhou, 2016], binarize it using a threshold of 0.15 and take the bounding box of the mask. We use VGG-GAP [Simonyan, 2015] and Resnet-50 [He, 2016b] as pretrained on Imagenet [Russakovsky, 2015]

Metric	Top-1 loc.		MaxBoxAcc-v2	
Network	VGG-GAP	ResNet-50	VGG-GAP	ResNet-50
ACoL [Zhang, 2018b]	45.9	–	57.4	–
ADL [Choe, 2019]	52.4	–	61.3	58.4
Baseline CAM [Zhou, 2016]	37.1	49.4	59.0	59.7
Input [Zhang, 2018a]	41.7	49.3	57.1	60.6
CutMix [Yun, 2019]	52.5	54.8	62.6	64.8
AlignMixup (ours)	**53.7**	**57.8**	**64.5**	**65.9**
Gain	**+1.2**	**+3.0**	**+1.9**	**+1.1**

Table 3.7 – *Weakly-supervised object localization* on CUB200-2011. Top-1 loc.: Top-1 localization accuracy (%), MaxBoxAcc-v2: Maximal box accuracy [Choe, 2020]. Higher is better. Blue: second best. Gain: increase of accuracy.

and we fine-tune them on CUB200-2011 [Wah, 2011b]. We follow the evaluation protocol by [Choe, 2020] and use top-1 localization accuracy with IoU threshold of 0.5 and Maximal Box Accuracy (MaxBoxAcc-v2) to compare AlignMixup with baseline CAM (without mixup), Input mixup [Zhang, 2018a], CutOut [DeVries, 2017b] and CutMix [Yun, 2019].

According to Table 3.7, AlignMixup outperforms Input mixup, CutOut and CutMix by 11.98%, 8.88% and 1.18% respectively using VGG-GAP and by 8.5%, 5.02% and 3% respectively using Resnet-50 in terms of top-1 localization accuracy. Furthermore, AlignMixup outperforms CutMix by 1.9% and 1.1% using VGG-GAP and Resnet-50 respectively in terms of MaxBoxAcc-v2. It also outperforms dedicated WSOL methods ACoL [Zhang, 2018b] and ADL [Choe, 2019], which focus on learning spatially dispersed representations.

Qualitative localization results shown in Figure 3.5 indicate that AlignMixup encodes semantically discriminative representations, resulting in better localization performance.

3.4.6 Ablation study

All ablations are performed on CIFAR-100 using R-18 as stage 1 encoder F with feature tensor $\mathbf{A}$ being $512 \times 4 \times 4$ and embedding $e \in \mathbb{R}^{512}$. We study the effect of mixing different layers (x, $\mathbf{A}$ or e), aligning $\mathbf{A}$ or not before mixing, as well as the autoencoder architecture. The latter includes a *vanilla autoencoder* (AlignMixup/AE), a *variational autoencoder* [Kingma, 2013] (AlignMixup/VAE) and *no decoder* (AlignMixup). We report top-1 accuracy (%). All results are in Table 3.8.

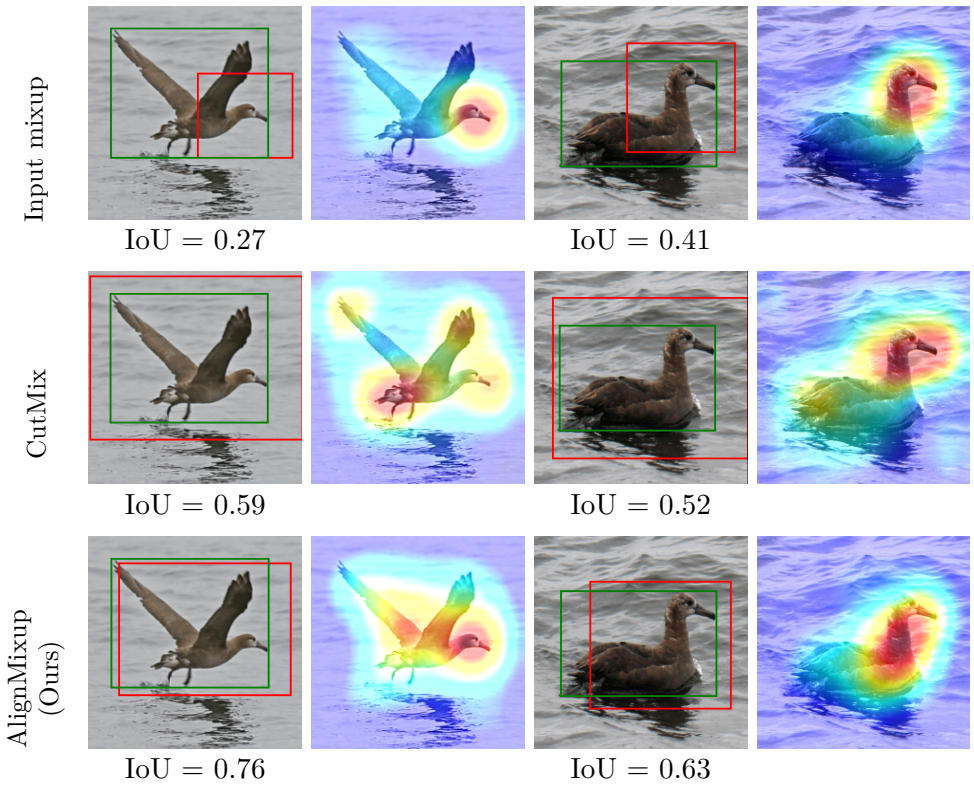


Figure 3.5 – *Localization examples* using ResNet-50 on CUB200-2011. Red boxes: predicted; green: ground truth.

Layers In general, we may mix any layer in $\{x, \mathbf{A}, e\}$ in a given iteration. We ablate the effect of allowing only a particular subset of layers. In general, $e \in \mathbb{R}^{512}$ is a vector. Here, we also consider the case where e is a $128 \times 2 \times 2$ tensor, denoted as $\mathbf{E}$ and obtained from $\mathbf{A}$ by a convolutional layer of kernel size 2×2 and stride 2. In AlignMixup/AE architecture, among different choices of unaligned layer sets, mixing from $\{x, e\}$ results in the highest classification accuracy. Furthermore, AlignMixup/AE outperforms baseline and the best performing competitor StyleCutMix for all choices of layers, even when features are unaligned.

Tensor alignment We ablate the effect of aligning feature tensor $\mathbf{A}$ or not before mixing it, by using standard mixup (3.2) or (3.12), (3.13), respectively. In AlignMixup, we observe that aligning $\mathbf{A}$ before mixing improves classification accuracy significantly. It is important to note that when e is a vector, we do not align it. However, when it is a tensor $\mathbf{E}$, aligning it improves significantly. Overall, AlignMixup/AE works the best when

Method/Arch	Layers	Unaligned	Aligned
Baseline		76.76	–
Manifold [Verma, 2019]		80.20	-
StyleCutMix [Hong, 2021]		80.66	-
AlignMixup	$\{x, e\}$	80.81	–
	$\{\mathbf{A}\}$	79.07	80.28
	$\{e\}$	78.71	-
	$\{x, \mathbf{A}\}$	80.34	81.61
	$\{x, \mathbf{A}, \mathbf{E}\}$	80.46	81.36
	$\{x, \mathbf{A}, e\}$	80.33	**81.92**
AlignMixup/AE	$\{x, e\}$	81.92	-
	$\{\mathbf{A}\}$	79.39	81.04
	$\{e\}$	79.49	-
	$\{x, \mathbf{A}\}$	81.78	81.85
	$\{x, \mathbf{E}\}$	80.80	81.54
	$\{x, \mathbf{A}, e\}$	81.61	**82.18**
AlignMix/AE	$\{x, \mathbf{A}_{2\times2}, e\}$	81.47	81.20
	$\{x, \mathbf{A}_{4\times4}, e\}$	81.61	82.18
	$\{x, \mathbf{A}_{8\times8}, e\}$	80.49	**82.20**
AlignMixup/VAE	$\{x, (\mu, \sigma)\}$	81.81	–
	$\{x, \mathbf{A}\}$	81.35	81.85
	$\{x, (\mathbf{M}, \mathbf{\Sigma})\}$	80.45	81.10
	$\{x, \mathbf{A}, (\mu, \sigma)\}$	81.00	**81.89**

Table 3.8 – *Ablations* using R-18 on CIFAR-100. Top-1 classification accuracy (%): higher is better. Arch: autoencoder architecture. AE: vanilla; VAE: variational [Kingma, 2013]. Layer $x, \mathbf{A}, e$: (3.3), (3.4), (3.5).

$x, \mathbf{A}, e$ are mixed, with $\mathbf{A}$ being aligned. This setting outperforms StyleCutMix by 1.52%. Mixing $\mathbf{A}$ only helps when it is aligned; otherwise, it is preferable to just mix e.

Alignment resolution We ablate the effect of aligning $\mathbf{A}$ at different spatial resolutions. The default is 4×4, denoted as $\mathbf{A}_{4\times4}$. Here, we investigate 2×2 ($\mathbf{A}_{2\times2}$), obtained by average pooling, and 8×8 ($\mathbf{A}_{8\times8}$), by removing downsampling from the last convolutional layer. The accuracy of 8×8 is only slightly better than 4×4 by 0.02%, while being computationally more expensive. Thus, we choose 4×4 as the default. By contrast, aligning at 2×2 is worse than not aligning at all. This may be due to soft correspondences causing loss of information by averaging.

Autoencoder architecture We investigate two more autoencoder architectures, AlignMixup/AE and AlignMixup/VAE. The latter has two vectors $\mu, \sigma \in \mathbb{R}^{512}$ instead of e,

Iterations (i)	0	10	20	50	100	200	500	1000
AlignMixup	80.98	80.96	81.11	81.32	81.92	81.88	81.04	81.08

Table 3.9 – *Ablation* of the number of iterations in Sinkhorn-Knopp algorithm using R-18 on CIFAR-100. Top-1 classification accuracy(%): higher is better.

representing mean and standard deviation, respectively. We also investigate $128 \times 2 \times 2$ tensors, denoted as $\mathbf{M}, \mathbf{\Sigma}$ where the two variables are mixed simultaneously. As for AlignMixup/AE, we investigate different combinations of layers with or without alignment. Both AlignMixup and AlignMixup/VAE are inferior to AlignMixup/AE. However, their best setting still outperforms Baseline and StyleCutmix. All three architecture work best when $x, \mathbf{A}, e$ are mixed. Alignment improves consistently on all three architectures.

Iterations in Sinkhorn-Knopp The default number of iterations for the Sinkhorn-Knopp algorithm in solving (3.9) is $i = 100$. Here, we investigate more choices, as shown in Table 3.9. The case of $i = 0$ is similar to cross-attention. In this case, we only normalize either the rows or columns in (3.8) once, such that $P\mathbf{1} = \mathbf{1}/r$ (when $\mathbf{A}$ aligned to $\mathbf{A}'$) or $P^\top \mathbf{1} = \mathbf{1}/r$ (when $\mathbf{A}'$ aligned to $\mathbf{A}$). We observe that while AlignMixup outperforms the best baseline–StyleCutMix (80.66)–in all cases, it performs best for $i = 100$ iterations.

3.5 Discussions

We have shown that mixup of a combination of input and latent representations is a simple and very effective pairwise data augmentation method. The gain is most prominent on large datasets and in combating overconfidence in predictions, as indicated by out-of-distribution detection. Interpolation of feature tensors boosts performance significantly, but only if they are aligned.

Our work is a compromise between a "good" hand-crafted interpolation in the image space and a fully learned one in the latent space. A challenge is to make progress in the latter direction without compromising speed and simplicity, which would affect wide applicability.

Chapter 4

Mixup for Deep Metric Learning

Deep metric learning involves learning a discriminative representation such that embeddings of similar classes are encouraged to be close, while embeddings of dissimilar classes are pushed far apart. Traditionally, metric learning loss functions rely on pairwise loss functions, focusing on pulling together positive pairs and pushing apart negative ones. The idea of using pairwise loss functions is not limited to metric learning, and has been widely explored in self-supervised learning [Chen, 2020b] and supervised learning [Khosla, 2020] for image classification. For *e.g.* SupCon [Khosla, 2020], effectively uses multiple positive and negative pairs in its loss function for image classification in a supervised setting. This approach significantly improves performance compared to self-supervised methods that rely on a single positive pair.

Another method that has shown to improve performance on image classification is *Mixup*, which interpolates between pairs of examples and its corresponding labels. There exists a striking similarity between using pairwise similarity in metric learning and using pairs of examples in mixup. This led us to explore the possibility of interpolating between pairs in metric learning using mixup, similar to how it works in classification.

In this chapter, we introduce mixup in metric learning. However, directly interpolating the pairs of embeddings presents a unique challenge. Unlike classification, loss functions in metric learning are not additive over examples. This makes it non-trivial to directly interpolate target labels using traditional mixup. To address this challenge, we first develop a generalized formulation that encompasses existing metric learning loss functions and modify it to accommodate for mixup. This contributes a principled way of interpolating labels, such that the interpolation factor affects the relative weighting of positives and negatives. Since interpolating between all possible pairs ($n(n-1)/2$) can be computationally expensive, we leverages an efficient linear interpolation strategy, making it significantly faster than complex non-linear interpolation methods.

We presented this work in the Tenth International Conference on Learning Representations (ICLR), 2022.

4.1 Introduction

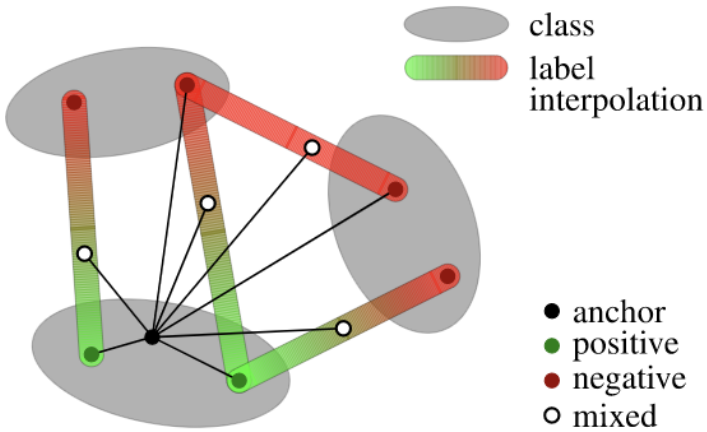


Figure 4.1 – *Metrix* (= *Metric Mix*) allows an anchor to interact with positive (same class), negative (different class) and interpolated examples, which also have interpolated labels.

Classification is one of the most studied tasks in machine learning and deep learning. It is a common source of pre-trained models for *transfer learning* to other tasks [Donahue, 2014; Kolesnikov, 2020]. It has been studied under different *supervision settings* [Caron, 2018; Sohn, 2020], *knowledge transfer* [Hinton, 2015] and *data augmentation* [Cubuk, 2018], including the recent research on *mixup* [Zhang, 2018a; Verma, 2019], where embeddings and labels are interpolated.

Deep metric learning is about learning from pairwise interactions such that inference relies on instance embeddings, *e.g.* for *nearest neighbor classification* [Oh Song, 2016], *instance-level retrieval* [Gordo, 2016], *few-shot learning* [Vinyals, 2016], *face recognition* [Schroff, 2015] and *semantic textual similarity* [Reimers, 2019].

Following [Xing, 2003b], it is most often fully supervised by one class label per example, like classification. The two most studied problems are *loss functions* [Musgrave, 2020] and *hard example mining* [Wu, 2017; Robinson, 2021]. Tuple-based losses with example weighting [Wang, 2019b] can play the role of both.

Unlike classification, classes (and distributions) at training and inference are different in metric learning. Thus, one might expect interpolation-based data augmentation like mixup to be even more important in metric learning than in classification. Yet, recent attempts are mostly limited to special cases of embedding interpolation and have trouble with label interpolation [Ko, 2020]. This raises the question: *what is a proper way to define*

and interpolate labels for metric learning?

In this work, we observe that metric learning is not different from classification, where examples are replaced by pairs of examples and class labels by "positive" or "negative", according to whether class labels of individual examples are the same or not. The positive or negative label of an example, or a pair, is determined in relation to a given example which is called an *anchor*. Then, as shown in Figure 5.1, a straightforward way is to use a *binary* (two class) label per pair and interpolate it linearly as in standard mixup. We call our method *Metric Mix*, or *Metrix* for short.

To show that mixing examples improves representation learning, we quantitatively measure the properties of the test distributions using *alignment* and *uniformity* [Wang, 2020]. *Alignment* measures the clustering quality and *uniformity* measures its distribution over the embedding space; a well clustered and uniformly spread distribution indicates higher representation quality. We also introduce a new metric, *utilization*, to measure the extent to which a test example, seen as a query, lies near any of the training examples, clean or mixed. By quantitatively measuring these three metrics, we show that interpolation-based data augmentation like mixup is very important in metric learning, given the difference between distributions at training and inference.

In summary, we make the following contributions:

1. We define a generic way of representing and interpolating labels, which allows straightforward extension of any kind of mixup to deep metric learning for a large class of loss functions. We develop our method on a generic formulation that encapsulates these functions (section 4.3).
2. We define the "positivity" of a mixed example and we study precisely how it increases as a function of the interpolation factor, both in theory and empirically (subsection 4.3.6).
3. We systematically evaluate mixup for deep metric learning under different settings, including mixup at different representation levels (input/manifold), mixup of different pairs of examples (anchors/positives/negatives), loss functions and hard example mining (subsection 4.4.3).
4. We introduce a new evaluation metric, *utilization*, validating that a representation more appropriate for test classes is implicitly learned during exploration of the embedding space in the presence of mixup (subsection 4.4.4).
5. We improve the state of the art on four common metric learning benchmarks (sub-

section 4.4.3).

4.2 Related Work

Metric learning Metric learning aims to learn a metric such that *positive* pairs of examples are nearby and *negative* ones are far away. In *deep metric learning*, we learn an explicit non-linear mapping from raw input to a low-dimensional *embedding space* [Oh Song, 2016], where the Euclidean distance has the desired properties. Although learning can be unsupervised [Hadsell, 2006], deep metric learning has mostly followed the supervised approach, where positive and negative pairs are defined as having the same or different class label, respectively [Xing, 2003b].

Loss functions can be distinguished into pair-based and proxy-based [Musgrave, 2020]. *Pair-based* losses use pairs of examples [Wu, 2017; Hadsell, 2006], which can be defined over triplets [Wang, 2014; Schroff, 2015; Weinberger, 2009; Hermans, 2017], quadruples [Chen, 2017] or tuples [Sohn, 2016; Oh Song, 2016; Wang, 2019b]. *Proxy-based* losses use one or more proxies per class, which are learnable parameters in the embedding space [Movshovitz-Attias, 2017; Qian, 2019; Kim, 2020c; Teh, 2020; Zhu, 2020b]. Pair-based losses capture data-to-data relations, but they are sensitive to noisy labels and outliers. They often involve terms where given constraints are satisfied, which produce zero gradients and do not contribute to training. This necessitates *mining* of hard examples that violate the constraints, like semi-hard [Schroff, 2015] and distance weighted [Wu, 2017]. By contrast, proxy-based losses use data-to-proxy relations, assuming proxies can capture the global structure of the embedding space. They involve less computations that are more likely to produce nonzero gradient, hence have less or no dependence on mining and converge faster.

Mixup *Input mixup* [Zhang, 2018a] linearly interpolates between two or more examples in the input space for data augmentation. Numerous variants take advantage of the structure of the input space to interpolate non-linearly, *e.g.* for images [Yun, 2019; Kim, 2020a; Kim, 2021b; Hendrycks, 2019b; DeVries, 2017b; Qin, 2020; Uddin, 2021]. *Manifold mixup* [Verma, 2019] interpolates intermediate representations instead, where the structure is learned. This can be applied to or assisted by decoding back to the input space [Berthelot, 2018; Liu, 2018a; Beckham, 2019; Zhu, 2020a; Venkataramanan, 2021]. In both cases, corresponding labels are linearly interpolated too. Most studies are limited

to cross-entropy loss for classification. Pairwise loss functions have been under-studied, as discussed below.

Interpolation for pairwise loss functions As discussed above, interpolating target labels is not straightforward in pairwise loss functions. In *deep metric learning*, *embedding expansion* [Ko, 2020], HDML [Zheng, 2019] and *symmetrical synthesis* [Gu, 2020] interpolate pairs of embeddings in a deterministic way within the same class, applying to pair-based losses, while *proxy synthesis* [Gu, 2021] interpolates between classes, applying to proxy-based losses. None performs label interpolation, which means that [Gu, 2021] risks synthesizing false negatives when the interpolation factor λ is close to 0 or 1.

In *contrastive representation learning*, MoCHi [Kalantidis, 2020] interpolates anchor with negative embeddings but not labels and chooses $\lambda \in [0, 0.5]$ to avoid false negatives. This resembles thresholding of λ at 0.5 in OptTransMix [Zhu, 2020a]. Finally, *i-mix* [Lee, 2021] and MixCo [Kim, 2020b] interpolate pairs of anchor embeddings as well as their (virtual) class labels linearly. There is only one positive, while all negatives are clean, so it cannot take advantage of interpolation for relative weighting of positives/negatives per anchor [Wang, 2019b].

By contrast, Metrix is developed for deep metric learning and applies to a large class of both pair-based and proxy-based losses. It can interpolate inputs, intermediate features or embeddings of anchors, (multiple) positives or negatives *and* the corresponding two-class (positive/negative) labels per anchor, such that relative weighting of positives/negatives depends on interpolation.

4.3 Mixup for metric learning

4.3.1 Preliminaries

Problem formulation We are given a training set $X \subset \mathcal{X}$, where $\mathcal{X}$ is the input space. For each *anchor* $a \in X$, we are also given a set $P(a) \subset X$ of *positives* and a set $N(a) \subset X$ of *negatives*. The positives are typically examples that belong to the same class as the anchor, while negatives belong to a different class. The objective is to train the parameters θ of a model $f : X \to \mathbb{R}^d$ that maps input examples to a d-dimensional *embedding*, such that positives are close to the anchor and negatives are far away in the embedding space. Given two examples $x, x' \in \mathcal{X}$, we denote by $s(x, x')$ the *similarity* between x, x' in the

embedding space, typically a decreasing function of Euclidean distance. It is common to ℓ_2-normalize embeddings and define $s(x, x') := \langle f(x), f(x') \rangle$, which is the *cosine similarity*. To simplify notation, we drop the dependence of f, s on θ.

Pair-based losses [Hadsell, 2006; Wang, 2014; Oh Song, 2016; Wang, 2019b] use both anchors and positives/negatives in X, as discussed above. *Proxy-based* losses define one or more learnable *proxies* $\in \mathbb{R}^d$ per class, and only use proxies as anchors [Kim, 2020c] or as positives/negatives [Movshovitz-Attias, 2017; Qian, 2019; Teh, 2020]. To accommodate for uniform exposition, we extend the definition of similarity as $s(v, x) := \langle v, f(x) \rangle$ for $v \in \mathbb{R}^d, x \in \mathcal{X}$ (proxy anchors) and $s(x, v) := \langle f(x), v \rangle$ for $x \in \mathcal{X}, v \in \mathbb{R}^d$ (proxy positives/negatives). Finally, to accommodate for mixed embeddings in subsection 4.3.5, we define $s(v, v') := \langle v, v' \rangle$ for $v, v' \in \mathbb{R}^d$. Thus, we define $s : (\mathcal{X} \cup \mathbb{R}^d)^2 \to \mathbb{R}$ over pairs of either inputs in $\mathcal{X}$ or embeddings in $\mathbb{R}^d$. We discuss a few representative loss functions below, before deriving a generic form.

Contrastive The contrastive loss [Hadsell, 2006] encourages positive examples to be pulled towards the anchor and negative examples to be pushed away by a margin $m \in \mathbb{R}$. This loss is *additive* over positives and negatives, defined as:

$$\ell_{\text{cont}}(a; \theta) := \sum_{p \in P(a)} -s(a, p) + \sum_{n \in N(a)} [s(a, n) - m]_+. \tag{4.1}$$

Multi-Similarity The multi-similarity loss [Wang, 2019b] introduces *relative weighting* to encourage positives (negatives) that are farthest from (closest to) the anchor to be pulled towards (pushed away from) the anchor by a higher weight. This loss is *not* additive over positives and negatives:

$$\ell_{\text{MS}}(a; \theta) := \frac{1}{\beta} \log \left(1 + \sum_{p \in P(a)} e^{-\beta(s(a,p)-m)} \right) + \frac{1}{\gamma} \log \left(1 + \sum_{n \in N(a)} e^{\gamma(s(a,n)-m)} \right). \tag{4.2}$$

Here, $\beta, \gamma \in \mathbb{R}$ are scaling factors for positives, negatives respectively.

Proxy Anchor The proxy anchor loss [Kim, 2020c] defines a learnable *proxy* in $\mathbb{R}^d$ for each class and only uses proxies as anchors. For a given anchor (proxy) $a \in \mathbb{R}^d$, the loss has the same form as (4.2), although similarity s is evaluated on $\mathbb{R}^d \times \mathcal{X}$.

4.3.2 Generic loss formulation

We observe that both additive (4.1) and non-additive (4.2) loss functions involve a sum over positives $P(a)$ and a sum over negatives $N(a)$. They also involve a decreasing function of similarity $s(a,p)$ for each positive $p \in P(a)$ and an increasing function of similarity $s(a,n)$ for each negative $n \in N(a)$. Let us denote by ρ^+, ρ^- this function for positives, negatives respectively. Then, non-additive functions differ from additive by the use of a nonlinear function σ^+, σ^- on positive and negative terms respectively, as well as possibly another nonlinear function τ on their sum:

$$\ell(a;\theta) := \tau\left(\sigma^+\left(\sum_{p\in P(a)} \rho^+(s(a,p))\right) + \sigma^-\left(\sum_{n\in N(a)} \rho^-(s(a,n))\right)\right). \tag{4.3}$$

With the appropriate choice for $\tau, \sigma^+, \sigma^-, \rho^+, \rho^-$, this definition encompasses contrastive (4.1), multi-similarity (4.2) or proxy-anchor as well as many pair-based or proxy-based loss functions, as shown in Table 4.1. It does not encompass the *triplet loss* [Wang, 2014], which operates on pairs of positives and negatives, forming triplets with the anchor. The triplet loss is the most challenging in terms of mining because there is a very large number of pairs and only few contribute to the loss. We only use function τ to accommodate for *lifted structure* [Oh Song, 2016; Hermans, 2017], where $\tau(x) := [x]_+$ is reminiscent of the triplet loss. We observe that multi-similarity [Wang, 2019b] differs from *binomial deviance* [Yi, 2014] only in the weights of the positive and negative terms. Proxy anchor [Kim, 2020c] is a proxy version of multi-similarity [Wang, 2019b] on anchors and ProxyNCA [Movshovitz-Attias, 2017] is a proxy version of NCA [Goldberger, 2005] on positives/negatives.

This generic formulation highlights the components of the loss functions that are additive over positives/negatives and paves the way towards incorporating mixup.

4.3.3 Improving representations using mixup

To improve the learned representations, we follow [Zhang, 2018a; Verma, 2019] in mixing inputs and features from intermediate network layers, respectively. Both are developed for classification.

Input mixup [Zhang, 2018a] augments data by linear interpolation between a pair of input examples. Given two examples $x, x' \in \mathcal{X}$ we draw $\lambda \sim \text{Beta}(\alpha,\alpha)$ as *interpolation factor* and mix x with x' using the standard mixup operation $\text{mix}_\lambda(x,x') := \lambda x + (1-\lambda)x'$.

Loss	Anchor	Pos/Neg	$\tau(x)$	$\sigma^+(x)$	$\sigma^-(x)$	$\rho^+(x)$	$\rho^-(x)$
Contrastive [Hadsell, 2006]	X	X	x	x	x	$-x$	$[x-m]_+$
Lifted structure [Hermans, 2017]	X	X	$[x]_+$	$\log(x)$	$\log(x)$	e^{-x}	e^{x-m}
Binomial deviance [Yi, 2014]	X	X	x	$\log(1+x)$	$\log(1+x)$	$e^{-\beta(x-m)}$	$e^{\gamma(x-m)}$
Multi-similarity [Wang, 2019b]	X	X	x	$\frac{1}{\beta}\log(1+x)$	$\frac{1}{\gamma}\log(1+x)$	$e^{-\beta(x-m)}$	$e^{\gamma(x-m)}$
Proxy anchor [Kim, 2020c]	proxy	X	x	$\frac{1}{\beta}\log(1+x)$	$\frac{1}{\gamma}\log(1+x)$	$e^{-\beta(x-m)}$	$e^{\gamma(x-m)}$
NCA [Goldberger, 2005]	X	X	x	$-\log(x)$	$\log(x)$	e^{x}	e^{x}
ProxyNCA [Movshovitz-Attias, 2017]	X	proxy	x	$-\log(x)$	$\log(x)$	e^{x}	e^{x}
ProxyNCA++ [Teh, 2020]	X	proxy	x	$-\log(x)$	$\log(x)$	$e^{x/T}$	$e^{x/T}$

Table 4.1 – Loss functions. Anchor/positive/negative: X: embedding of input example from training set X by f; proxy: learnable parameter in $\mathbb{R}^d$; T : temperature. All loss functions are encompassed by (4.3) using the appropriate definition of functions $\tau, \sigma^+, \sigma^-, \rho^+, \rho^-$ as given here.

Manifold mixup [Verma, 2019] linearly interpolates between intermediate representations (features) of the network instead. Referring to 2D images, we define $g_m : \mathcal{X} \to \mathbb{R}^{c\times w\times h}$ as the mapping from the input to intermediate layer m of the network and $f_m : \mathbb{R}^{c\times w\times h} \to \mathbb{R}^d$ as the mapping from intermediate layer m to the embedding, where c is the number of channels (feature dimensions) and $w \times h$ is the spatial resolution. Thus, our model f can be expressed as the composition $f = f_m \circ g_m$.

For manifold mixup, we follow [Venkataramanan, 2021] and mix either features of intermediate layer m or the final embeddings. Thus, we define three *mixup types* in total:

$$f_\lambda(x, x') := \begin{cases} f(\mathrm{mix}_\lambda(x, x')), & \text{input mixup} \\ f_m(\mathrm{mix}_\lambda(g_m(x), g_m(x'))), & \text{feature mixup} \\ \mathrm{mix}_\lambda(f(x), f(x')), & \text{embedding mixup.} \end{cases} \tag{4.4}$$

Function $f_\lambda : \mathcal{X}^2 \to \mathbb{R}^d$ performs both mixup and embedding. We explore different mixup types in subsection 4.4.5.

4.3.4 Label representation

Classification In supervised classification, each example $x \in X$ is assigned an one-hot encoded label $y \in \{0, 1\}^C$, where C is the number of classes. Label vectors are also linearly interpolated: given two labeled examples $(x, y), (x', y')$, the interpolated label is $\mathrm{mix}_\lambda(y, y')$. The loss (cross-entropy) is a continuous function of the label vector. We extend this idea to metric learning.

Metric learning Positives $P(a)$ and negatives $N(a)$ of anchor a are defined as having the same or different class label as the anchor, respectively. To every example in $P(a) \cup N(a)$, we assign a binary (two-class) label $y \in \{0, 1\}$, such that $y = 1$ for positives and $y = 0$ for negatives:

$$U^+(a) := \{(p, 1) : p \in P(a)\} \tag{4.5}$$

$$U^-(a) := \{(n, 0) : n \in N(a)\} \tag{4.6}$$

Thus, we represent both positives and negatives by $U(a) := U^+(a) \cup U^-(a)$. We now rewrite the generic loss function (4.3) as:

$$\ell(a; \theta) := \tau \left(\sigma^+ \left(\sum_{(x,y) \in U(a)} y \rho^+(s(a, x)) \right) + \sigma^- \left(\sum_{(x,y) \in U(a)} (1 - y) \rho^-(s(a, x)) \right) \right). \tag{4.7}$$

Here, every labeled example (x, y) in $U(a)$ appears in both positive and negative terms. However, because label y is binary, only one of the two contributions is nonzero. Now, in the presence of mixup, we can linearly interpolate labels exactly as in classification.

4.3.5 Mixed loss function

Mixup For every anchor a, we are given a set $M(a)$ of pairs of examples to mix. This is a subset of $(S(a) \cup U(a)) \times U(a)$ where $S(a) := (a, 1)$. That is, we allow mixing between positive-negative, positive-positive and negative-negative pairs, where the anchor itself is also seen as positive. We define the possible choices of *mixing pairs* $M(a)$ in subsection 4.4.1. Let $V(a)$ be the set of corresponding *labeled mixed embeddings*

$$V(a) := \{(f_\lambda(x, x'), \mathrm{mix}_\lambda(y, y')) : ((x, y), (x', y')) \in M(a), \lambda \sim \mathrm{Beta}(\alpha, \alpha)\}, \tag{4.8}$$

where f_λ is defined by (4.4). With these definitions in place, the generic loss function $\widetilde{\ell}$ over mixed examples takes exactly the same form as (4.7), with only $U(a)$ replaced by $V(a)$:

$$\widetilde{\ell}(a; \theta) := \tau \left(\sigma^+ \left(\sum_{(v,y) \in V(a)} y \rho^+(s(a, v)) \right) + \sigma^- \left(\sum_{(v,y) \in V(a)} (1 - y) \rho^-(s(a, v)) \right) \right), \tag{4.9}$$

where similarity s is evaluated on $\mathcal{X} \times \mathbb{R}^d$ for pair-based losses and on $\mathbb{R}^d \times \mathbb{R}^d$ for proxy anchor. Now, every labeled embedding (v, y) in $V(a)$ appears in both positive and negative terms and *both* contributions are nonzero for positive-negative pairs, because after interpolation, $y \in [0, 1]$.

Error function Parameters θ are learned by minimizing the error function, which is a linear combination of the *clean loss* (4.3) and the *mixed loss* (4.9), averaged over all anchors

$$E(X; \theta) := \frac{1}{|X|} \sum_{a \in X} \ell(a; \theta) + w\widetilde{\ell}(a; \theta), \tag{4.10}$$

where $w \geq 0$ is the *mixing strength*. At least for manifold mixup, this combination comes at little additional cost, since clean embeddings are readily available.

Interpretation To better understand the two contributions of a labeled embedding (v, y) in $V(a)$ to the positive and negative terms of (4.9), consider the case of positive-negative mixing pairs, $M(a) \subset U^+(a) \times U^-(a)$. Then, for $((x, y), (x', y')) \in M(a)$, the mixed label is $\mathrm{mix}_\lambda(y, y') = \mathrm{mix}_\lambda(1, 0) = \lambda$ and (4.9) becomes

$$\widetilde{\ell}(a; \theta) = \tau \left(\sigma^+ \left(\sum_{(v,\lambda) \in V(a)} \lambda \rho^+(s(a, v)) \right) + \sigma^- \left(\sum_{(v,\lambda) \in V(a)} (1 - \lambda) \rho^-(s(a, v)) \right) \right). \tag{4.11}$$

Thus, the mixed embedding v is both positive (with weight λ) and negative (with weight $1 - \lambda$). Whereas for positive-positive mixing, that is, for $M(a) \subset U^+(a)^2$, the mixed label is 1 and the negative term vanishes. Similarly, for negative-negative mixing, that is, for $M(a) \subset U^-(a)^2$, the mixed label is 0 and the positive term vanishes.

In the particular case of contrastive (4.1) loss, positive-negative mixing (4.11) becomes

$$\widetilde{\ell}_{\mathrm{cont}}(a; \theta) := \sum_{(v,\lambda) \in V(a)} -\lambda s(a, v) + \sum_{(v,\lambda) \in V(a)} (1 - \lambda)[s(a, v) - m]_+. \tag{4.12}$$

Similarly, for multi-similarity (4.2),

$$\begin{aligned} \widetilde{\ell}_{\mathrm{MS}}(a; \theta) := & \frac{1}{\beta} \log \left(1 + \sum_{(v,\lambda) \in V(a)} \lambda e^{-\beta(s(a,v)-m)} \right) + \\ & \frac{1}{\gamma} \log \left(1 + \sum_{(v,\lambda) \in V(a)} (1 - \lambda) e^{\gamma(s(a,v)-m)} \right). \end{aligned} \tag{4.13}$$

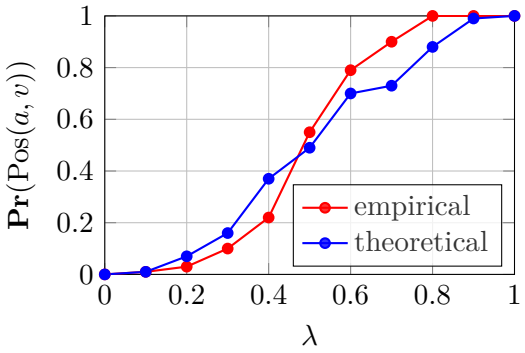


Figure 4.2 – *"Positivity" of mixed embeddings* vs. λ. We measure $\mathbf{Pr}(\mathrm{Pos}(a,v))$ empirically as $\mathbf{Pr}(\partial\widetilde{\ell}_{\mathrm{MS}}(a;\theta)/\partial s(a,v) \leq 0)$ and theoretically by (4.14), where F_λ is again measured from data. We use embedding mixup on MS (4.2) on CUB200 at epoch 0, based on the setup of subsection 4.4.1.

4.3.6 Analysis: Mixed embeddings and positivity

Let $\mathrm{Pos}(a,v)$ be the event that a mixed embedding v behaves as "positive" for anchor a, *i.e.*, minimizing the loss $\widetilde{\ell}(a;\theta)$ will increase the similarity $s(a,v)$. Under positive-negative mixing, *i.e.*, $M(a) \subset U^+(a) \times U^-(a)$, we then estimate the probability of $\mathrm{Pos}(a,v)$ as a function of λ in the case of multi-similarity (4.2) with a single mixed embedding v:

$$\mathbf{Pr}(\mathrm{Pos}(a,v)) = F_\lambda\left(\frac{1}{\beta+\gamma}\ln\left(\frac{\lambda}{1-\lambda}\right)+m\right), \tag{4.14}$$

where F_λ is the CDF of similarities $s(a,v)$ between anchors a and mixed embeddings v with interpolation factor λ. In Figure 4.2, we measure the probability of $\mathrm{Pos}(a,v)$ as a function of λ in two ways, both purely empirically and theoretically by (4.14). Both measurements are increasing functions of λ of sigmoidal shape, where a mixed embedding is mostly positive for λ close to 1 and mostly negative for λ close to 0.

Positivity Under positive-negative mixing, (4.11) shows that a mixed embedding v with interpolation factor λ behaves as both positive and negative to different extents, depending on λ: mostly positive for λ close to 1, mostly negative for λ close to 0. The net effect depends on the derivative of the loss with respect to the similarity $\partial\widetilde{\ell}(a;\theta)/\partial s(a,v)$: if the derivative is negative, then v behaves as positive and vice versa. This is clear from the chain rule

$$\frac{\partial\widetilde{\ell}(a;\theta)}{\partial v} = \frac{\partial\widetilde{\ell}(a;\theta)}{\partial s(a,v)}\cdot\frac{\partial s(a,v)}{\partial v}, \tag{4.15}$$

because $\partial s(a,v)/\partial v$ is a vector pointing in a direction that makes a, v more similar and the loss is being minimized. Let $\text{Pos}(a,v)$ be the event that v behaves as "positive", *i.e.*, $\partial\widetilde{\ell}(a;\theta)/\partial s(a,v) \leq 0$ and minimizing the loss will increase the similarity $s(a,v)$.

Multi-similarity We estimate the probability of $\text{Pos}(a,v)$ as a function of λ in the case of multi-similarity with a single embedding v obtained by mixing a positive with a negative:

$$\widetilde{\ell}_{\text{MS}}(a;\theta) = \frac{1}{\beta}\log\left(1+\lambda e^{-\beta(s(a,v)-m)}\right) + \frac{1}{\gamma}\log\left(1+(1-\lambda)e^{\gamma(s(a,v)-m)}\right). \tag{4.16}$$

In this case, $\text{Pos}(a,v)$ occurs if and only if

$$\frac{\partial\widetilde{\ell}_{\text{MS}}(a;\theta)}{\partial s(a,v)} = \frac{-\lambda e^{-\beta(s(a,v)-m)}}{(1+\lambda e^{-\beta(s(a,v)-m)})} + \frac{(1-\lambda)e^{\gamma(s(a,v)-m)}}{(1+(1-\lambda)e^{\gamma(s(a,v)-m)})} \leq 0. \tag{4.17}$$

By letting $t := s(a,v) - m$, this condition is equivalent to

$$\frac{(1-\lambda)e^{\gamma t}}{(1+(1-\lambda)e^{\gamma t})} \leq \frac{\lambda e^{-\beta t}}{(1+\lambda e^{-\beta t})} \tag{4.18}$$

$$(1-\lambda)e^{\gamma t}(1+\lambda e^{-\beta t}) \leq \lambda e^{-\beta t}(1+(1-\lambda)e^{\gamma t}) \tag{4.19}$$

$$(1-\lambda)e^{\gamma t} + \lambda(1-\lambda)e^{(\gamma-\beta)t} \leq \lambda e^{-\beta t} + \lambda(1-\lambda)e^{(\gamma-\beta)t} \tag{4.20}$$

$$e^{(\beta+\gamma)t} \leq \frac{\lambda}{1-\lambda} \tag{4.21}$$

$$(\beta+\gamma)(s(a,v)-m) \leq \ln\left(\frac{\lambda}{1-\lambda}\right) \tag{4.22}$$

$$s(a,v) \leq \frac{1}{\beta+\gamma}\ln\left(\frac{\lambda}{1-\lambda}\right) + m. \tag{4.23}$$

Finally, the probability of $\text{Pos}(a,v)$ as a function of λ is

$$\mathbf{Pr}(\text{Pos}(a,v)) = F_\lambda\left(\frac{1}{\beta+\gamma}\ln\left(\frac{\lambda}{1-\lambda}\right) + m\right), \tag{4.24}$$

where F_λ is the CDF of similarities $s(a,v)$ between anchors a and mixed embeddings v with interpolation factor λ.

In Figure 4.2, we measure the probability of $\text{Pos}(a,v)$ as a function of λ in two ways. First, we measure the derivative $\partial\widetilde{\ell}_{\text{MS}}(a;\theta)/\partial s(a,v)$ for anchors a and mixed embeddings v

Dataset	CUB200 [Wah, 2011a]	Cars196 [Krause, 2013]	SOP [Oh Song, 2016]	In-Shop [Liu, 2016b]
Objects	birds	cars	household furniture	clothes
# classes	200	196	22,634	7,982
# training images	5,894	8,092	60,026	26,356
# testing images	5,894	8,093	60,027	26,356
# training classes	100	98	11,318	3991
# testing classes	100	98	11,318	3991
sampling	random	random	balanced	balanced
samples per class	–	–	5	5
classes per batch	65†	70†	20	20
learning rate	1×10^{-4}	1×10^{-4}	3×10^{-5}	1×10^{-4}

Table 4.2 – *Statistics and settings* for the four datasets we use in our experiments. †: average.

over the entire dataset and we report the empirical probability of this derivative being non-positive versus λ. Second, we measure $\mathbf{Pr}(\text{Pos}(a, v))$ theoretically using (4.24), where the CDF of similarities $s(a, v)$ is again measured empirically for a and v over the dataset, as a function of λ. Despite the simplifying assumption of a single positive and a single negative in deriving (4.24), we observe that the two measurements agree in general. They are both increasing functions of λ of sigmoidal shape, they roughly yield $\mathbf{Pr}(\text{Pos}(a, v)) \geq 0.5$ for $\lambda \geq 0.5$ and they confirm that a mixed embedding is mostly positive for λ close to 1 and mostly negative for λ close to 0.

4.4 Experiments

4.4.1 Setup

Datasets We experiment on Caltech-UCSD Birds (CUB200) [Wah, 2011a], Stanford Cars (Cars196) [Krause, 2013], Stanford Online Products (SOP) [Oh Song, 2016] and In-Shop Clothing retrieval (In-Shop) [Liu, 2016b] image datasets.

Network, features and embeddings We use Resnet-50 [He, 2016b] (R-50) pretrained on ImageNet [Russakovsky, 2015] as a backbone network. We obtain the intermediate representation (*feature*), a $7 \times 7 \times 2048$ tensor, from the last convolutional layer. Following [Kim, 2020c], we combine adaptive average pooling with max pooling, followed by a fully-connected layer to obtain the *embedding* of $d = 512$ dimensions.

Loss functions We reproduce *contrastive* (Cont) [Hadsell, 2006], *multi-similarity* (MS) [Wang, 2019b], *proxy anchor* (PA) [Kim, 2020c] and *ProxyNCA++* [Teh, 2020] and we evaluate them under different mixup types. For MS (4.2), following [Musgrave, 2020], we use $\beta = 18$, $\gamma = 75$ and $m = 0.77$. For PA, we use $\beta = \gamma = 32$ and $m = 0.1$, as reported by the authors.

Methods We compare our method, *Metrix*, with *proxy synthesis* (PS) [Gu, 2021], *i-mix* [Lee, 2021] and MoCHi [Kalantidis, 2020]. For PS, we adapt the official code [1] to PA on all datasets, and use it with PA only, because it is designed for proxy-based losses. PS has been shown superior to [Ko, 2020; Gu, 2020], although in different networks. MoCHi and *i-mix* are meant for contrastive representation learning. We evaluate using Recall@K [Oh Song, 2016]: For each test example taken as a query, we find its K-nearest neighbors in the test set excluding itself in the embedding space. We assign a score of 1 if an example of the same class is contained in the neighbors and 0 otherwise. Recall@K is the average of this score over the test set.

Datasets and sampling Dataset statistics are summarized in Table 4.2. Since the number of classes is large compared to the batch size in SOP and In-Shop, batches would rarely contain a positive pair when sampled uniformly at random. Hence, we use *balanced sampling* [Zhai, 2018], *i.e.*, a fixed number of classes and examples per class, as shown in Table 4.2. For fair comparison with baseline methods, images are randomly flipped and cropped to 224×224 at training. At inference, we resize to 256×256 and then center-crop to 224×224.

Training We train R-50 using AdamW [Loshchilov, 2019b] optimizer for 100 epochs with a batch size 100. The initial learning rate per dataset is shown in Table 4.2. The learning rate is decayed by 0.1 for Cont and by 0.5 for MS and PA on CUB200 and Cars196. For SOP and In-Shop, we decay the learning rate by 0.25 for all losses. The weight decay is set to 0.0001.

4.4.2 Mixup settings

In mixup for classification, given a batch of n examples, it is standard to form n pairs of examples by pairing the batch with a *random permutation* of itself, resulting in n mixed

1. https://github.com/navervision/proxy-synthesis

examples, either for input or manifold mixup. In metric learning, it is common to obtain n embeddings and then use all $\frac{1}{2}n(n-1)$ pairs of embeddings in computing the loss. We thus treat mixup types differently.

Input mixup Mixing all pairs would be computationally expensive in this case, because we would compute $\frac{1}{2}n(n-1)$ embeddings. A random permutation would not produce as many hard examples as can be found in all pairs. Thus, for each anchor (each example in the batch), we use the k *hardest negative* examples and mix them with positives or with the anchor. We use $k = 3$ by default.

Manifold mixup Originally, manifold mixup [Verma, 2019] focuses on the *first* few layers of the network. Mixing all pairs would then be even more expensive than input mixup, because intermediate features (tensors) are even larger than input examples. Hence, we focus on the *last* few layers instead, where features and embeddings are compact, and we mix all pairs. We use feature mixup by default and call it *Metrix/feature* or just *Metrix*, while input and embedding mixup are called *Metrix/input* and *Metrix/embed*, respectively. All options are studied in subsection 4.4.5.

Mixing pairs Whatever the mixup type, we use clean examples as anchors and we define a set $M(a)$ of pairs of examples to mix for each anchor a, with their labels (positive or negative). By default, we mix positive-negative or anchor-negative pairs, according to $M(a) := U^+(a) \times U^-(a)$ and $M(a) := S(a) \times U^-(a)$, respectively, where $U^-(a)$ is replaced by hard negatives only for input mixup. The two options are combined by choosing uniformly at random in each iteration. More options are studied in subsection 4.4.5.

Hyper-parameters For any given mixup type or set of mixup pairs, the interpolation factor λ is drawn from $\text{Beta}(\alpha, \alpha)$ with $\alpha = 2$. We empirically set the mixup strength (4.10) to $w = 0.4$ for positive-negative pairs and anchor-negative pairs.

4.4.3 Results

Improving the state of the art As shown in Table 4.3, Metrix consistently improves the performance of all baseline losses (Cont, MS, PA, ProxyNCA++) across all datasets. Surprisingly, MS outperforms PA and ProxyNCA++ under mixup on all datasets but

SOP, where the three losses are on par. This is despite the fact that baseline PA outperforms MS on CUB200 and Cars-196, while ProxyNCA++ outperforms MS on SOP and In-Shop. Both contrastive and MS are significantly improved by mixup. By contrast, improvements on PA and ProxyNCA++ are marginal, which may be due to the already strong performance of PA, or further improvement is possible by employing different mixup methods that take advantage of the image structure.

In terms of Recall@1, our MS+Metrix is best overall, improving by 3.6% (67.8 → 71.4) on CUB200, 1.8% (87.8 → 89.6) on Cars196, 4.1% (76.9 → 81.0) on SOP and 2.1% (90.1 → 92.2) on In-Shop. The same solution sets new state of the art, outperforming the previously best PA by 1.7% (69.7 → 71.4) on CUB200, MS by 1.8% (87.8 → 89.6) on Cars196, ProxyNCA++ by 0.3% (80.7 → 81.0) on SOP and SoftTriple by 1.2% (91.0 → 92.2) on In-Shop. Importantly, while the previous state of the art comes from a different loss per dataset, MS+Metrix is almost consistently best across all datasets.

Alternative mixing methods In Table 4.4, we compare Metrix/input with i-Mix [Lee, 2021] and Metrix/embed with MoCHi [Kalantidis, 2020] using contrastive loss, and with PS [Gu, 2021] using PA. MoCHi and PS mix embeddings only, while labels are always negative. For i-Mix, we mix anchor-negative pairs ($S(a) \times U^-(a)$). For MoCHi, the anchor is clean and we mix negative-negative ($U^-(a)^2$) and anchor-negative ($S(a) \times U^-(a)$) pairs, where $U^-(a)$ is replaced by $k = 100$ hardest negatives and $\lambda \in (0, 0.5)$ for anchor-negative. PS mixes embeddings of different classes and treats them as new classes. For clean anchors, this corresponds to positive-negative ($U^+(a) \times U^-(a)$) and negative-negative ($U^-(a)^2$) pairs, but PS also supports mixed anchors.

In terms of Recall@1, Metrix/input outperforms i-Mix with anchor-negative pairs by 0.5% (65.8 → 66.3) on CUB200, 0.9% (82.0 → 82.9) on Cars196, 0.6% (75.2 → 75.8) and 0.6% (87.1 → 87.7) on In-Shop. Metrix/embed outperforms MoCHI with anchor-negative pairs by 1.2% (65.2 → 66.4) on CUB200, 1.4% (82.5 → 83.9) on Cars196, 0.9% (75.8 → 76.7) and 1.2% (87.2 → 88.4) on In-Shop. The gain over MoCHi with negative-negative pairs is significantly higher. Metrix/embed also outperforms PS by 0.4% (70.0 → 70.4) on CUB200, 1% (87.9 → 88.9) on Cars196, 1% (79.6 → 80.6) on SOP and 1.3% (90.3 → 91.6) on In-Shop.

Computational complexity On CUB200 dataset, using a batch size of 100 on an NVIDIA RTX 2080 Ti GPU, the average training time in ms/batch is 586 for MS and

	CUB200			Cars196			SOP			In-Shop		
Method	1	2	4	1	2	4	1	10	100	1	10	20
Triplet [Weinberger, 2009]	63.5	75.6	84.4	77.3	85.4	90.8	70.5	85.6	94.3	85.3	96.6	97.8
LiftedStructure [Oh Song, 2016]	65.9	75.8	84.5	81.4	88.3	92.4	76.1	88.6	95.2	88.6	97.6	98.4
ProxyNCA [Movshovitz-Attias, 2017]	65.2	75.6	83.8	81.2	87.9	92.6	73.2	87.0	94.4	86.2	95.9	97.0
Margin [Wu, 2017]	65.0	76.2	84.6	82.1	88.7	92.7	74.8	87.8	94.8	88.6	97.0	97.8
SoftTriple [Qian, 2019]	67.3	77.7	86.2	86.5	91.9	95.3	79.8	91.2	96.3	**91.0**	97.6	98.3
D&C [Sanakoyeu, 2019]*	65.9	76.6	84.4	84.6	90.7	94.1	75.9	88.4	94.9	85.7	95.5	96.9
EPSHN [Xuan, 2020]*	64.9	75.3	83.5	82.7	89.3	93.0	78.3	90.7	96.3	87.8	95.7	96.8
ProxyNCA++ [Teh, 2020]*	69.0	79.8	**87.3**	86.5	92.5	95.7	**80.7**	**92.0**	**96.7**	90.4	**98.1**	**98.8**
Cont [Hadsell, 2006]	64.7	75.9	84.6	81.6	88.2	92.7	74.9	87.0	93.9	86.4	94.7	96.2
+Metrix/input	66.3	77.1	85.2	82.9	89.3	93.7	75.8	87.8	94.6	87.7	95.9	96.5
	+1.6	+1.2	+0.6	+1.3	+1.1	+1.0	+0.9	+0.8	+0.7	+1.3	+1.2	+0.3
+Metrix	67.4	77.9	85.7	85.1	91.1	94.6	77.5	89.1	95.5	89.1	95.7	97.1
	+2.7	+2.0	+1.1	+3.5	+2.9	+1.9	+2.6	+2.1	+1.5	+2.7	+1.0	+0.9
+Metrix/embed	66.4	77.6	85.4	83.9	90.3	94.1	76.7	88.6	95.2	88.4	95.4	96.8
	+1.7	+1.7	+0.8	+2.3	+2.1	+1.4	+1.8	+1.6	+1.3	+2.0	+0.7	+0.6
MS [Wang, 2019b]	67.8	77.8	85.6	**87.8**	92.7	95.3	76.9	89.8	95.9	90.1	97.6	98.4
+Metrix/input	69.0	79.1	86.0	89.0	93.4	96.0	77.9	90.6	95.9	91.8	98.0	98.9
	+1.2	+1.3	+0.4	+1.2	+0.7	+0.7	+1.0	+0.8	+0.0	+1.7	+0.4	+0.5
+Metrix	**71.4**	80.6	86.8	**89.6**	**94.2**	96.0	81.0	92.0	**97.2**	**92.2**	**98.5**	98.6
	+3.6	+2.8	+1.2	+1.8	+1.5	+0.7	+4.1	+2.2	+1.3	+2.1	+0.9	+0.2
+Metrix/embed	70.2	80.4	86.7	88.8	92.9	95.6	78.5	91.3	96.7	91.9	98.3	98.7
	+2.4	+2.6	+1.1	+1.0	+0.2	+0.3	+1.6	+1.5	+0.8	+1.8	+0.7	+0.3
PA [Kim, 2020c]*	**69.7**	**80.0**	87.0	87.7	**92.9**	**95.8**	–	–	–	–	–	–
PA [Kim, 2020c]	69.5	79.3	87.0	87.6	92.3	95.5	79.1	90.8	96.2	90.0	97.4	98.2
+Metrix/input	70.5	81.2	87.8	88.2	93.2	96.2	79.8	91.4	96.5	90.9	98.1	98.4
	+0.8	+1.2	+0.8	+0.5	+0.3	+0.4	+0.7	+0.6	+0.3	+0.9	+0.7	+0.2
+Metrix	71.0	**81.8**	**88.2**	89.1	93.6	**96.7**	**81.3**	91.7	96.9	91.9	98.2	**98.8**
	+1.3	+1.8	+1.2	+1.4	+0.7	+0.9	+2.2	+0.9	+0.7	+1.9	+0.8	+0.6
+Metrix/embed	70.4	81.1	87.9	88.9	93.3	96.4	80.6	91.7	96.6	91.6	98.3	98.3
	+0.7	+1.1	+0.9	+1.2	+0.4	+0.6	+1.5	+0.9	+0.4	+1.6	+0.9	+0.1
ProxyNCA++ [Teh, 2020]*	69.0	79.8	87.3	86.5	92.5	95.7	**80.7**	**92.0**	**96.7**	90.4	**98.1**	**98.8**
ProxyNCA++ [Teh, 2020]	69.1	79.5	**87.7**	86.6	92.1	95.4	80.4	91.7	**96.7**	90.2	97.6	98.4
+Metrix/input	69.7	79.9	88.3	87.5	92.9	96.0	80.9	92.2	96.9	91.4	98.1	98.8
	+0.6	+0.1	+0.6	+0.9	+0.4	+0.3	+0.2	+0.2	+0.2	+1.0	+0.0	+0.0
+Metrix	70.4	80.6	88.7	88.5	93.4	96.5	**81.3**	**92.7**	97.1	91.9	98.1	98.8
	+1.3	+0.8	+1.0	+1.9	+0.9	+0.8	+0.6	+0.7	+0.4	+1.5	+0.0	+0.0
+Metrix/ embed	70.2	80.2	88.2	88.1	93.0	96.2	81.1	92.4	97.0	91.6	98.1	98.8
	+1.1	+0.4	+0.5	+1.5	+0.5	+0.5	+0.4	+0.4	+0.3	+1.2	+0.0	+0.0
Gain over SOTA	+1.7	+1.8	+0.5	+1.8	+1.3	+0.9	+0.6	+0.0	+0.5	+1.2	+0.4	+0.0

Table 4.3 – *Improving the SOTA with our Metrix* (Metrix/feature) using Resnet-50 with embedding size $d = 512$. R@K (%): Recall@K; higher is better. *: reported by authors. **Bold black**: best baseline (previous SOTA, one per column). **Red**: Our new SOTA. Gain over SOTA is over best baseline. MS: Multi-Similarity, PA: Proxy Anchor.

		CUB200			Cars196			SOP			In-Shop		
Method	Mixing Pairs	R@1	R@2	R@4	R@1	R@2	R@4	R@1	R@10	R@100	R@1	R@10	R@20
Cont [Hadsell, 2006]	–	64.7	75.9	84.6	81.6	88.2	92.7	74.9	87.0	93.9	86.4	94.7	96.3
+ *i*-Mix [Lee, 2021]	anc-neg	65.8	76.2	84.9	82.0	88.5	93.2	75.2	87.3	94.2	87.1	95.4	96.1
+ Metrix/input	pos-neg / anc-neg	66.3	77.1	85.2	82.9	89.3	93.7	75.8	87.8	94.6	87.7	95.9	96.5
+MoCHi [Kalantidis, 2020]	neg-neg	63.1	74.3	83.8	76.3	84.0	89.3	68.9	83.1	91.8	81.8	91.9	93.9
+MoCHi [Kalantidis, 2020]	anc-neg	65.2	75.8	84.2	82.5	88.0	92.9	75.8	87.1	94.8	87.2	92.8	94.9
+Metrix/embed	pos-neg / anc-neg	**66.4**	**77.6**	**85.4**	**83.9**	**90.3**	**94.1**	**76.7**	**88.6**	**95.2**	**88.4**	**95.4**	**96.9**
PA [Kim, 2020c]	–	69.7	80.0	87.0	87.6	92.3	95.5	79.1	90.8	96.2	90.0	97.4	98.2
+PS [Gu, 2021]	pos-neg / neg-neg	70.0	79.8	87.2	87.9	92.8	95.6	79.6	90.9	96.4	90.3	97.4	98.0
+Metrix/embed	pos-neg / anc-neg	**70.4**	**81.1**	**87.9**	**88.9**	**93.3**	**96.4**	**80.6**	**91.7**	**96.6**	**91.6**	**98.3**	**98.3**

Table 4.4 – *Comparison of our Metrix/embed with other mixing methods* using R-50 with embedding size $d = 512$. R@K (%): Recall@K; higher is better. PA: Proxy Anchor, PS: Proxy Synthesis.

817 for MS+Metrix. The 39% increase in complexity is reasonable for 3.6% increase in R@1. Furthermore, the average training time in ms/batch is 483 for baseline PA, 965 for PA+Metrix and 1563 for PS [Gu, 2021]. While the computation cost of PS is higher than Metrix by 62%, Metrix outperform PS by 0.4% and 1.3% in terms of R@1 and R@2 respectively (Table 4.4). At inference, the computational cost is equal for all methods.

Qualitative results of retrieval Figure 4.3 shows qualitative results of retrieval on CUB200 using Contrastive loss, with and without mixup. This dataset has large intra-class variations such as pose variation and background clutter. Baseline Contrastive loss may fail to retrieve the correct images due to these challenges. The ranking is improved in the presence of mixup.

Visualization of embedding space We visualize CUB200 test examples for 10, 15 and 20 classes in the embedding space using Contrastive loss, with and without mixup in Figure 4.4. We observe that in the presence of mixup, the embeddings are more tightly clustered and more uniformly spread, despite the variations in pose and background in the test set. This finding validates our quantitative analysis of alignment and uniformity in subsection 4.4.4.

4.4.4 How does mixup improve representations?

We analyze how Metrix improves representation learning, given the difference between distributions at training and inference. As discussed in section 4.1, since the classes at inference are unseen at training, one might expect interpolation-based data augmentation

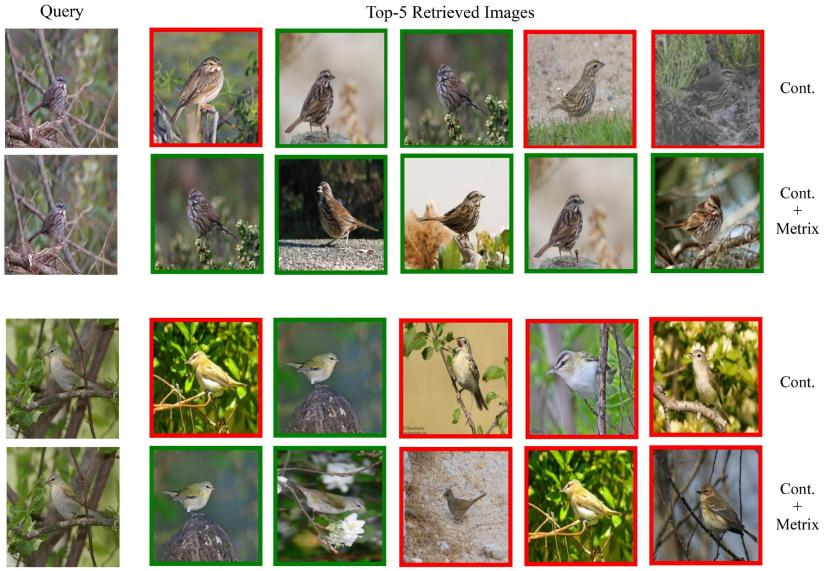


Figure 4.3 – *Retrieval results* on CUB200 using Contrastive loss, with and without mixup. For each query, the top-5 retrieved images are shown. Images highlighted in green (red) are correctly (incorrectly) retrieved images.

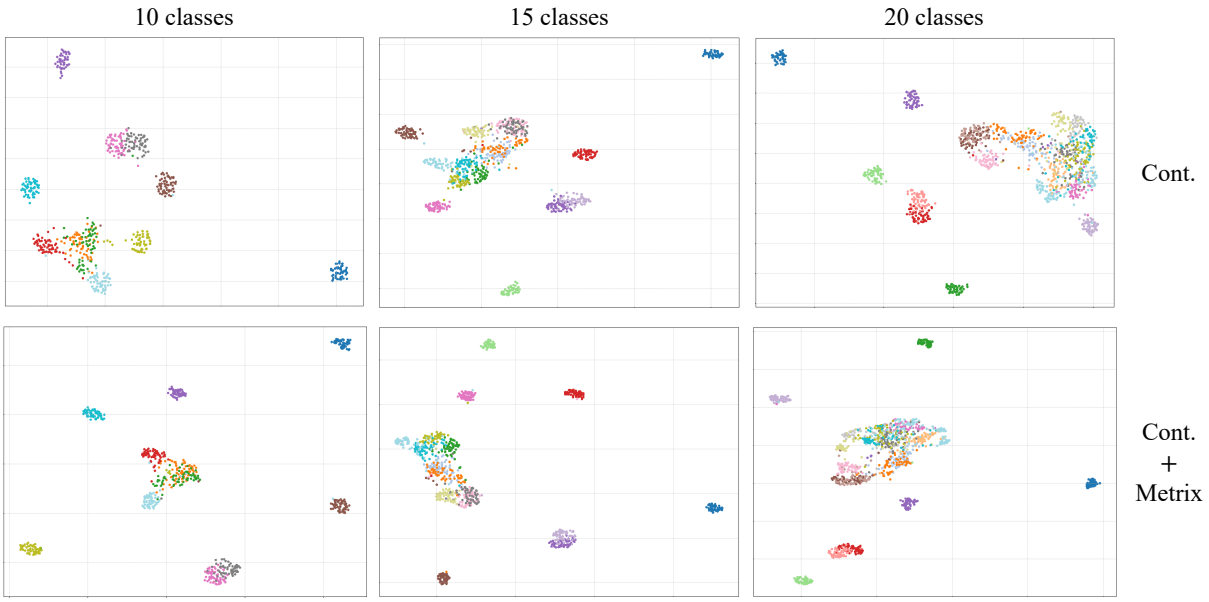


Figure 4.4 – *Embedding space visualization* of CUB200 test examples of a given number of classes using Contrastive loss, with and without mixup.

like mixup to be even more important than in classification. This is so because, by mixing examples during training, we are exploring areas of the embedding space beyond the training classes. We hope that this exploration would possibly lead the model to implicitly learn a representation more appropriate for the test classes, if the distribution of the test classes lies near these areas.

Alignment and Uniformity In terms of quantitative measures of properties of the training and test distributions, we follow [Wang, 2020]. This work introduces two measures – *alignment* and *uniformity* (the lower the better) to be used both as loss functions (on the training set) and as evaluation metrics (on the test set). *Alignment* measures the expected pairwise distance between positive examples in the embedding space. A small value of alignment indicates that the positive examples are clustered together. *Uniformity* measures the (log of the) expected pairwise similarity between all examples regardless of class, using a Gaussian kernel as similarity. A small value of uniformity indicates that the distribution is more uniform over the embedding space, which is particularly relevant to our problem. Meant for contrastive learning, [Wang, 2020] use the same training and test classes, while in our case they are different.

By training with contrastive loss on CUB200 and then measuring on the test set, we achieve an alignment (lower the better) of 0.28 for contrastive loss, 0.28 for i-Mix [Lee, 2021] and 0.19 for Metrix/input. MoCHi [Kalantidis, 2020] and Metrix/embed achieve an alignment of 0.19 and 0.17, respectively. We also obtain a uniformity (lower the better) of -2.71 for contrastive loss, -2.13 for i-Mix and -3.13 for Metrix/input. The uniformity of MoCHi and Metrix/embed is -3.18 and -3.25, respectively. This indicates that Metrix helps obtain a test distribution that is more uniform over the embedding space, where classes are better clustered and better separated.

Utilization The measures proposed by [Wang, 2020] are limited to a single distribution or dataset, either the training set (as loss functions) or the test set (as evaluation metrics). It is more interesting to measure the extent to which a test example, seen as a query, lies near any of the training examples, clean or mixed. For this, we introduce the measure of *utilization* $u(Q, X)$ of the training set X by the test set Q as

$$u(Q, X) = \frac{1}{|Q|} \sum_{q \in Q} \min_{x \in X} \|f(q) - f(x)\|^2 \tag{4.25}$$

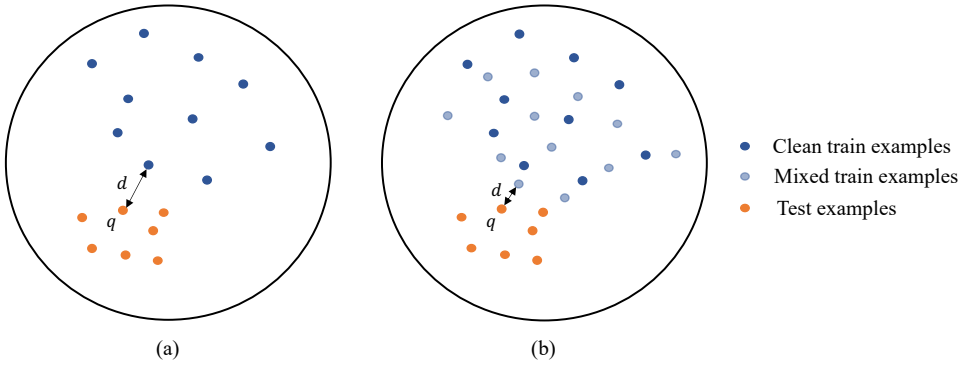


Figure 4.5 – Exploring the embedding space when using (a) only clean examples (b) clean and mixed examples. Given a query q, the distance d to its nearest training embedding (clean or mixed) is smaller with mixup (b) than without (a).

Utilization measures the average, over the test set Q, of the minimum distance of a query q to a training example $x \in X$ in the embedding space of the trained model f (lower is better). A low value of utilization indicates that there are examples in the training set that are similar to test examples. When using mixup, we measure utilization as $u(Q, \hat{X})$, where $\hat{X}$ is the augmented training set including clean and mixed examples over a number of epochs and f remains fixed. Because $X \subset \hat{X}$, we expect $u(Q, \hat{X}) < u(Q, X)$, that is, the embedding space is better explored in the presence of mixup.

By using contrastive loss on CUB200, utilization drops from 0.41 to 0.32 when using Metrix. This indicates that test samples are indeed closer to mixed examples than clean in the embedding space. This validates our hypothesis that a representation more appropriate for test classes is implicitly learned during exploration of the embedding space in the presence of mixup.

We provide an illustration of this exploration in Figure 4.5, where we visualize the embedding space using (a) only clean train examples and (b) clean and mixed train examples. In case (a), the model is trained using only clean examples, exploring a smaller area of the embedding space. In case (b), it is trained using both mixed and clean examples, exploring a larger area. It is clear that the distance between a query and its nearest training example (clean or mixup) is smaller in the presence of mixup. Utilization is the average of this distance over the test set. This shows that the model implicitly learns a representation closer the test example in the presence of mixup during training and it partially explains why mixup leads to better performance.

4.4.5 Ablations

We perform ablations on Cars196 using R-50 with $d = 512$, applying mixup on contrastive loss.

Hard negatives We study the effect of the number k of hard negatives using different mixup types. The set of mixing pairs is chosen from (positive-negative, anchor-negative) uniformly at random per iteration. We choose $k = 3$ for input mixup. For feature/embedding mixup, we mix all pairs in a batch by default, but also study $k \in \{20, 40\}$. As shown in Table 4.5, $k = 3$ for input and all pairs for feature/embedding mixup works best. Still, using few hard negatives for feature/embedding mixup is on par or outperforms input mixup. All choices significantly outperform the baseline.

Mixing pairs We study the effect of mixing pairs $M(a)$, in particular, $U^+(a)^2$ (positive-positive), $U^+(a) \times U^-(a)$ (positive-negative) and $S(a) \times U^-(a)$ (anchor-negative), again using different mixup types. As shown in Table 4.5, when using a single set of mixing pairs during training, positive-negative and anchor-negative consistently outperform the baseline, while positive-positive is actually outperformed by the baseline. This may be due to the lack of negatives in the mixed loss (4.9), despite the presence of negatives in the clean loss (4.3). Hence, we only use positive-negative and anchor-negative by default, combined by choosing uniformly at random in each iteration.

Mixup types We study the effect of mixup type (input, feature, embedding), when used alone. The set of mixing pairs is chosen from (positive-negative, anchor-negative) uniformly at random per iteration. As shown in both "hard negatives" and "mixing pairs" parts of Table 4.5, our default feature mixup works best, followed by embedding and input mixup.

Mixup type combinations We study the effect of using more than one mixup type (input, feature, embedding), chosen uniformly at random per iteration. The set of mixing pairs is also chosen from (positive-negative, anchor-negative) uniformly at random per iteration. As shown in Table 4.5, mixing inputs, features and embeddings works best. Although this solution outperforms feature mixup alone by 0.2% Recall@1 (85.1 $\rightarrow$ 85.3), it is computationally expensive because of using input mixup. The next best efficient

Study	Hard Negatives k	Mixing Pairs	Mixup Type	R@1	R@2	R@4	R@8
baseline				81.6	88.2	92.7	95.8
hard negatives	1	pos-neg / anc-neg	input	82.0	89.1	93.1	96.1
	2	pos-neg / anc-neg	input	82.5	89.2	93.4	96.2
	3	pos-neg / anc-neg	input	82.9	89.3	93.7	95.5
	20	pos-neg / anc-neg	feature	83.5	90.1	94.0	96.5
	40	pos-neg / anc-neg	feature	84.0	90.4	94.2	96.8
	all	pos-neg / anc-neg	feature	**85.1**	**91.1**	**94.6**	**97.0**
	20	pos-neg / anc-neg	embed	82.7	89.2	93.4	96.1
	40	pos-neg / anc-neg	embed	83.0	90.0	93.8	96.4
	all	pos-neg / anc-neg	embed	83.4	89.9	94.1	96.4
mixing pairs	–	pos-pos	input	81.0	88.2	92.6	95.6
	3	pos-neg	input	82.4	89.1	93.3	95.6
	3	anc-neg	input	81.8	89.0	93.6	95.4
	–	pos-pos	feature	81.1	88.3	92.9	95.8
	all	pos-neg	feature	84.0	90.2	94.2	96.6
	all	anc-neg	feature	83.7	90.1	94.4	96.7
	–	pos-pos	embed	78.3	85.7	90.8	94.4
	all	pos-neg	embed	83.1	90.0	93.9	96.6
	all	anc-neg	embed	82.7	89.5	93.5	96.3
mixup type combinations	{1, all}	pos-neg / anc-neg	{input, feature}	83.7	94.2	95.9	96.7
	{3, all}	pos-neg / anc-neg	{input, embed}	83.0	90.9	94.1	96.4
	{all, all}	pos-neg / anc-neg	{feature, embed}	84.7	90.6	94.4	96.9
	{1, all, all}	pos-neg / anc-neg	{input, feature, embed}	**85.3**	**94.9**	**96.2**	**97.1**

Table 4.5 – *Ablation study of our Metrix* using contrastive loss and R-50 with embedding size $d = 512$ on Cars196. R@K (%): Recall@K; higher is better.

choice is mixing features and embeddings, which however is worse than mixing features alone (84.7 *vs*. 85.1). This is why we chose feature mixup by default.

Mixup strength w We study the effect of the mixup strength w in the combination of the clean and mixed loss (4.10) for different mixup types. As shown in Figure 4.6, mixup consistently improves the baseline and the effect of w is small, especially for input and embedding mixup. Feature mixup works best and is slightly more sensitive.

Ablation on CUB200 We perform additional ablations on CUB200 using R-50 with $d = 128$ by applying contrastive loss. All results are shown in Table 4.6. One may draw the same conclusions as from Table 4.5 on Cars196 with $d = 512$, which confirms that our choice of hard negatives and mixup pairs is generalizable across different datasets and embedding sizes.

In particular, following the settings of subsection 4.4.5, we observe in Table 4.6 that using $k = 3$ hard negatives for input mixup and all pairs for feature/embedding mixup

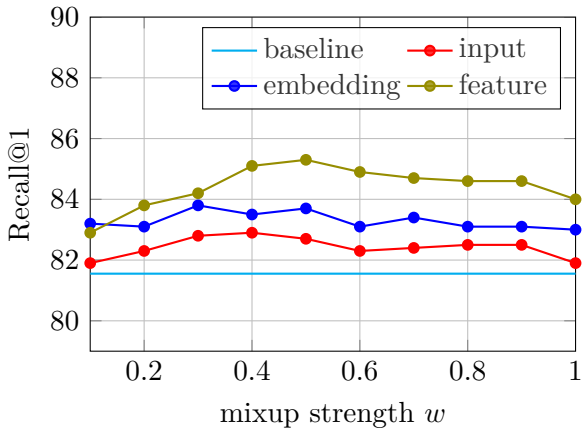


Figure 4.6 – *Effect of mixup strength* for different mixup types using contrastive loss and R-50 with embedding size $d = 512$ on Cars196. Recall@K (%): higher is better.

achieves the best performance in terms of Recall@1. Similarly, using a single set of mixing pairs, positive-negative and anchor-negative consistently outperform the baseline, whereas positive-positive is inferior than the baseline. Furthermore, combining positive-negative and anchor-negative pairs by choosing uniformly at random in each iteration achieves the best overall performance.

We also study the effect of using more than one mixup type (input, feature,embedding), chosen uniformly at random per iteration. The set of mixing pairs is also chosen from (positive-negative, anchor-negative) uniformly at random per iteration in this study. From Table 4.6, we observe that although mixing input, features and embedding works best with an improvement of 0.8% over feature mixup alone (64.5 → 65.3), it is computationally expensive due to using input mixup. The next best choice is mixing features and embeddings, which is worse than using feature mixup alone (64.2 *vs.* 64.5). This confirms our choice of using feature mixup as default.

4.5 Conclusion

Based on the argument that metric learning is binary classification of pairs of examples into "positive" and "negative", we have introduced a direct extension of mixup from classification to metric learning. Our formulation is generic, applying to a large class of loss functions that separate positives from negatives per anchor and involve component functions that are additive over examples. Those are exactly loss functions that require less mining. We contribute a principled way of interpolating labels, such that the inter-

Study	Hard Negatives k	Mixing Pairs	Mixup Type	R@1	R@2	R@4	R@8
baseline				61.6	73.7	83.6	90.1
hard negatives	1	pos-neg / anc-neg	input	62.4	73.9	83.0	89.7
	2	pos-neg / anc-neg	input	62.7	74.2	**83.6**	90.0
	3	pos-neg / anc-neg	input	63.1	74.5	83.5	90.3
	20	pos-neg / anc-neg	feature	63.9	75.0	83.9	89.9
	40	pos-neg / anc-neg	feature	63.5	75.2	83.5	89.8
	all	pos-neg / anc-neg	feature	**64.5**	**75.4**	84.3	**90.6**
	20	pos-neg / anc-neg	embed	63.1	74.3	83.1	90.0
	40	pos-neg / anc-neg	embed	63.5	74.7	83.6	90.1
	all	pos-neg / anc-neg	embed	64.0	75.1	84.8	90.9
mixing pairs	–	pos-pos	input	58.7	70.7	80.1	87.1
	3	pos-neg	input	62.9	75.1	83.4	90.6
	3	anc-neg	input	62.8	74.7	83.6	90.1
	–	pos-pos	feature	61.0	73.1	82.5	89.7
	all	pos-neg	feature	63.9	75.0	83.9	89.9
	all	anc-neg	feature	63.8	74.8	83.6	90.2
	–	pos-pos	embed	59.7	72.2	82.7	89.5
	all	pos-neg	embed	63.8	75.1	83.3	90.5
	all	anc-neg	embed	63.5	75.0	83.9	90.5
mixup type combinations	{1, all}	pos-neg / anc-neg	{input, feature}	63.9	75.1	**84.9**	90.5
	{3, all}	pos-neg / anc-neg	{input, embed}	63.4	74.9	84.5	90.1
	{all, all}	pos-neg / anc-neg	{feature, embed}	64.2	75.2	84.1	90.7
	{1, all, all}	pos-neg / anc-neg	{input, feature, embed}	**65.3**	**76.2**	84.4	**91.2**

Table 4.6 – *Ablation study of our Metrix* using contrastive loss and R-50 with embedding size $d = 128$ on CUB200. R@K (%): Recall@K; higher is better.

polation factor affects the relative weighting of positives and negatives. Other than that, our approach is completely agnostic with respect to the mixup method, opening the way to using more advanced mixup methods for metric learning.

We consistently outperform baselines using a number of loss functions on a number of benchmarks and we improve the state of the art using a single loss function on all benchmarks, while previous state of the art was not consistent in this respect. Surprisingly, this loss function, multi-similarity [Wang, 2019b], is not the state of the art without mixup. Since metric learning can be seen to generalize to unseen classes and distributions, our work may have applications to other such problems, including transfer learning, few-shot learning and continual learning.

Chapter 5

Interpolating Beyond Mini-Batch, Beyond Pairs and Beyond Examples

---

We observe in chapter 4, the effectiveness of mixup in deep metric learning, where increasing the number of loss terms by interpolating between all pairs of embeddings improves performance without incurring significant computational overhead. This motivates us to explore the potential of extending mixup further in classification by increasing the number of interpolated examples generated during training.

This approach aligns with the original motivation of mixup [Zhang, 2018a], which sought to augment the training data by generating new examples through interpolation. Augmenting the training data provides a better approximation of the risk integral, potentially leading to improved generalization performance. However, the original mixup paper [Zhang, 2018a] shows that the convex combination of three or more examples in the *input space* does not bring further gain and limits the interpolation between pairs of examples.

In this chapter, we revisit the initial motivation of mixup and increase the number of augmented examples through interpolation in the *embedding space*. Here, we generate an arbitrarily large number of interpolated examples beyond the mini-batch size, and interpolate the entire mini-batch in the embedding space. Geometrically, this translates to interpolating between all points, essentially sampling points on the convex hull of the mini-batch. On sequence data we further propose to increase the number of loss terms. We densely interpolate features and target labels at each spatial location and also apply the loss densely. To mitigate the lack of dense labels, we inherit labels from examples and weight interpolation factors by attention as a measure of confidence.

Overall, we increase the number of loss terms per mini-batch by orders of magnitude at little additional cost. This is only possible because of interpolating in the *embedding space*. We empirically show that our solutions yield significant improvement over state-of-the-art mixup methods on four different benchmarks, despite interpolation being only linear. This effort was presented in the Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS), 2023.

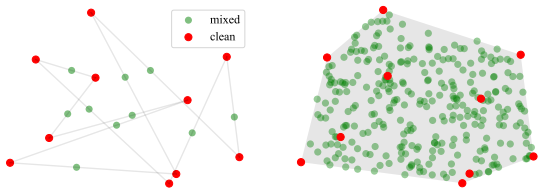
5.1 Introduction

Mixup [Zhang, 2018a] is a data augmentation method that interpolates between pairs of training examples, thus regularizing a neural network to favor linear behavior in-between examples. Besides improving generalization, it has important properties such as reducing overconfident predictions and increasing the robustness to adversarial examples. Several follow-up works have studied interpolation in the *latent* or *embedding* space, which is equivalent to interpolating along a manifold in the input space [Verma, 2019], and a number of nonlinear and attention-based interpolation mechanisms [Yun, 2019; Kim, 2020a; Kim, 2021b; Uddin, 2021; Chen, 2022]. However, little progress has been made in the augmentation process itself, *i.e.*, the number n of generated examples and the number m of examples being interpolated.

Mixup was originally motivated as a way to go beyond *empirical risk minimization* (ERM) [Vapnik, 1999] through a vicinal distribution expressed as an expectation over an interpolation factor λ, which is equivalent to the set of linear segments between all pairs of training inputs and targets. In practice however, in every training iteration, a single scalar λ is drawn and the number of interpolated pairs is limited to the size b of the mini-batch ($n = b$), as illustrated in Figure 5.1(a). This is because if interpolation takes place in the input space, it would be expensive to increase the number of pairs per iteration. To our knowledge, these limitations exist in all mixup methods.

In this work, we argue that a data augmentation process should increase the data seen by the model, or at least by its last few layers, as much as possible. In this sense, we follow *manifold mixup* [Verma, 2019] and generalize it in a number of ways to introduce *MultiMix*.

First, we increase the number n of generated examples beyond the mini-batch size b, by orders of magnitude ($n \gg b$). This is possible by interpolating at the deepest layer, *i.e.*, just before the classifier, which happens to be the most effective choice. To our knowledge, we are the



(a) mixup (b) MultiMix (ours)

Figure 5.1 – Data augmentation in a mini-batch B of $b = 10$ points in two dimensions. (a) *mixup*: sampling $n = b$ points on linear segments between b pairs of points using the same interpolation factor λ. (b) *MultiMix*: sampling $n = 300$ points in the convex hull of B.

Method	Space	Terms	Mixed	Fact	Distr
Mixup [Zhang, 2018a]	input	b	2	1	Beta
Manifold mixup [Verma, 2019]	embedding	b	2	1	Beta
ζ-Mixup [Abhishek, 2022]	input	b	25	1	RandPerm
SuperMix [Dabouei, 2021a]	input	b	3	1	Dirichlet
MultiMix (ours)	embedding	n	b	n	Dirichlet
Dense MultiMix (ours)	embedding	nr	b	nr	Dirichlet

Table 5.1 – *Interpolation method properties.* Space: Space where interpolation takes place; Terms: number of loss terms per mini-batch; Mixed: maximum number m of examples being interpolated; Fact: number of interpolation factors λ per mini-batch; Distr: distribution used to sample interpolation factors; RandPerm: random permutations of a fixed discrete probability distribution. b: mini-batch size; n: number of generated examples per mini-batch; r: spatial resolution.

first to investigate $n > b$.

Second, we increase the number m of examples being interpolated from $m = 2$ (pairs) to $m = b$ (a single *tuple* containing the entire mini-batch). Effectively, instead of linear segments between pairs of examples in the mini-batch, we sample on their entire *convex hull* as illustrated in Figure 5.1(b). This idea has been investigated in the input space: the original mixup method [Zhang, 2018a] found it non-effective, while [Dabouei, 2021a] found it effective only up to $m = 3$ examples and [Abhishek, 2022] went up to $m = 25$ but with very sparse interpolation factors. To our knowledge, we are the first to investigate $m > 2$ in the embedding space and to show that it is effective up to $m = b$.

Third, instead of using a single scalar value of λ per mini-batch, we draw a different vector $\lambda \in \mathbb{R}^m$ for each interpolated example. A single λ works for standard mixup because the main source of randomness is the choice of pairs (or small tuples) out of b examples. In our case, because we use a single tuple of size $m = b$, the only source of randomness being λ.

We also argue that, what matters more than the number of (interpolated) examples is the total number of *loss terms* per mini-batch. A common way to increase the number of loss terms per example is by *dense* operations when working on sequence data, *e.g.* patches in images or voxels in video. This is common in dense tasks like segmentation [Noh, 2015] and less common in classification [Kim, 2021a]. We are the first to investigate this idea in mixup, introducing *Dense MultiMix*.

In particular, this is an extension of MultiMix where we work with feature tensors of spatial resolution r and densely interpolate features and targets at each spatial location,

generating r interpolated features per example and $nr > n$ per mini-batch. We also apply the loss densely. This increases the number of loss terms further by a factor r, typically one or two orders of magnitude, compared with MultiMix. Of course, for this to work, we also need a target label per feature, which we inherit from the corresponding example. This is a *weak* form of supervision [Zhou, 2018]. To carefully select the most representative features per object, we use an *attention map* representing our confidence in the target label per spatial location. The interpolation vectors λ are then weighted by attention.

Table 5.1 summarizes the properties of our solutions against existing interpolation methods. Overall, we make the following contributions:

1. We generate an *arbitrary large number of interpolated examples* beyond the mini-batch size, each by interpolating the entire mini-batch in the embedding space, with one interpolation vector per example. (subsection 5.3.2).
2. We extend to attention-weighted *dense* interpolation in the embedding space, further increasing the number of loss terms per example (subsection 5.3.3).
3. We improve over state-of-the-art (SoTA) mixup methods on *image classification*, *robustness to adversarial attacks*, *object detection* and *out-of-distribution detection*. Our solutions have little or no additional cost while interpolation is only linear (section 5.4).
4. Analysis of the embedding space shows that our solutions yield classes that are *tightly clustered* and *uniformly spread* over the embedding space (section 5.4).

5.2 Related Work

Metric learning Metric learning aims to learn a metric such that *positive* pairs of examples are nearby and *negative* ones are far away. In *deep metric learning*, we learn an explicit non-linear mapping from raw input to a low-dimensional *embedding space* [Oh Song, 2016], where the Euclidean distance has the desired properties. Although learning can be unsupervised [Hadsell, 2006], deep metric learning has mostly followed the supervised approach, where positive and negative pairs are defined as having the same or different class label, respectively [Xing, 2003b].

Loss functions can be distinguished into pair-based and proxy-based [Musgrave, 2020]. *Pair-based* losses use pairs of examples [Wu, 2017; Hadsell, 2006], which can be defined over triplets [Wang, 2014; Schroff, 2015; Weinberger, 2009; Hermans, 2017], quadruples [Chen, 2017] or tuples [Sohn, 2016; Oh Song, 2016; Wang, 2019b]. *Proxy-based* losses

use one or more proxies per class, which are learnable parameters in the embedding space [Movshovitz-Attias, 2017; Qian, 2019; Kim, 2020c; Teh, 2020; Zhu, 2020b]. Pair-based losses capture data-to-data relations, but they are sensitive to noisy labels and outliers. They often involve terms where given constraints are satisfied, which produce zero gradients and do not contribute to training. This necessitates *mining* of hard examples that violate the constraints, like semi-hard [Schroff, 2015] and distance weighted [Wu, 2017]. By contrast, proxy-based losses use data-to-proxy relations, assuming proxies can capture the global structure of the embedding space. They involve less computations that are more likely to produce nonzero gradient, hence have less or no dependence on mining and converge faster.

Mixup *Input mixup* [Zhang, 2018a] linearly interpolates between two or more examples in the input space for data augmentation. Numerous variants take advantage of the structure of the input space to interpolate non-linearly, *e.g.* for images [Yun, 2019; Kim, 2020a; Kim, 2021b; Hendrycks, 2019b; DeVries, 2017b; Qin, 2020; Uddin, 2021]. *Manifold mixup* [Verma, 2019] interpolates intermediate representations instead, where the structure is learned. This can be applied to or assisted by decoding back to the input space [Berthelot, 2018; Liu, 2018a; Beckham, 2019; Zhu, 2020a; Venkataramanan, 2021]. In both cases, corresponding labels are linearly interpolated too. Most studies are limited to cross-entropy loss for classification. Pairwise loss functions have been under-studied, as discussed below.

Interpolation for pairwise loss functions As discussed above, interpolating target labels is not straightforward in pairwise loss functions. In *deep metric learning*, *embedding expansion* [Ko, 2020], HDML [Zheng, 2019] and *symmetrical synthesis* [Gu, 2020] interpolate pairs of embeddings in a deterministic way within the same class, applying to pair-based losses, while *proxy synthesis* [Gu, 2021] interpolates between classes, applying to proxy-based losses. None performs label interpolation, which means that [Gu, 2021] risks synthesizing false negatives when the interpolation factor λ is close to 0 or 1.

In *contrastive representation learning*, MoCHi [Kalantidis, 2020] interpolates anchor with negative embeddings but not labels and chooses $\lambda \in [0, 0.5]$ to avoid false negatives. This resembles thresholding of λ at 0.5 in OptTransMix [Zhu, 2020a]. Finally, *i-mix* [Lee, 2021] and MixCo [Kim, 2020b] interpolate pairs of anchor embeddings as well as their (virtual) class labels linearly. There is only one positive, while all negatives are clean, so

it cannot take advantage of interpolation for relative weighting of positives/negatives per anchor [Wang, 2019b].

By contrast, Metrix is developed for deep metric learning and applies to a large class of both pair-based and proxy-based losses. It can interpolate inputs, intermediate features or embeddings of anchors, (multiple) positives or negatives *and* the corresponding two-class (positive/negative) labels per anchor, such that relative weighting of positives/negatives depends on interpolation.

5.3 Method

5.3.1 Preliminaries and background

Problem formulation Let $x \in \mathcal{X}$ be an input example and $y \in \mathcal{Y}$ its one-hot encoded target, where $\mathcal{X} = \mathbb{R}^D$ is the input space, $\mathcal{Y} = \{0, 1\}^c$ and c is the total number of classes. Let $f_\theta : \mathcal{X} \to \mathbb{R}^d$ be an encoder that maps the input x to an embedding $z = f_\theta(x)$, where d is the dimension of the embedding. A classifier $g_W : \mathbb{R}^d \to \Delta^{c-1}$ maps z to a vector $p = g_W(z)$ of predicted probabilities over classes, where $\Delta^n \subset \mathbb{R}^{n+1}$ is the unit n-simplex, *i.e.*, $p \geq 0$ and $\mathbf{1}_c^\top p = 1$, and $\mathbf{1}_c \in \mathbb{R}^c$ is an all-ones vector. The overall network mapping is $f := g_W \circ f_\theta$. Parameters (θ, W) are learned by optimizing over mini-batches.

Given a mini-batch of b examples, let $X = (x_1, \ldots, x_b) \in \mathbb{R}^{D \times b}$ be the inputs, $Y = (y_1, \ldots, y_b) \in \mathbb{R}^{c \times b}$ the targets and $P = (p_1, \ldots, p_b) \in \mathbb{R}^{c \times b}$ the predicted probabilities of the mini-batch, where $P = f(X) := (f(x_1), \ldots, f(x_b))$. The objective is to minimize the cross-entropy

$$H(Y, P) := -\mathbf{1}_c^\top (Y \odot \log(P)) \mathbf{1}_b / b \tag{5.1}$$

of predicted probabilities P relative to targets Y averaged over the mini-batch, where $\odot$ is the Hadamard (element-wise) product. In summary, the mini-batch loss is

$$L(X, Y; \theta, W) := H(Y, g_W(f_\theta(X))). \tag{5.2}$$

The total number of loss terms per mini-batch is b.

Mixup Mixup methods commonly interpolate pairs of inputs or embeddings and the corresponding targets at the mini-batch level while training. Given a mini-batch of b

examples with inputs X and targets Y, let $Z = (z_1, \ldots, z_b) \in \mathbb{R}^{d \times b}$ be the embeddings of the mini-batch, where $Z = f_\theta(X)$. *Manifold mixup* [Verma, 2019] interpolates the embeddings and targets by forming a convex combination of the pairs with interpolation factor $\lambda \in [0, 1]$:

$$\widetilde{Z} = Z(\lambda I + (1 - \lambda)\Pi) \tag{5.3}$$

$$\widetilde{Y} = Y(\lambda I + (1 - \lambda)\Pi), \tag{5.4}$$

where $\lambda \sim \text{Beta}(\alpha, \alpha)$, I is the identity matrix and $\Pi \in \mathbb{R}^{b \times b}$ is a permutation matrix. *Input mixup* [Zhang, 2018a] interpolates inputs rather than embeddings:

$$\widetilde{X} = X(\lambda I + (1 - \lambda)\Pi). \tag{5.5}$$

Whatever the interpolation method and the space where it is performed, the interpolated data, *e.g.* $\widetilde{X}$ [Zhang, 2018a] or $\widetilde{Z}$ [Verma, 2019], replaces the original mini-batch data and gives rise to predicted probabilities $\widetilde{P} = (p_1, \ldots, p_b) \in \mathbb{R}^{c \times b}$ over classes, *e.g.* $\widetilde{P} = f(\widetilde{X})$ [Zhang, 2018a] or $\widetilde{P} = g_W(\widetilde{Z})$ [Verma, 2019]. Then, the average cross-entropy $H(\widetilde{Y}, \widetilde{P})$ (5.1) between the predicted probabilities $\widetilde{P}$ and interpolated targets $\widetilde{Y}$ is minimized. The number of generated examples per mini-batch is $n = b$, same as the original mini-batch size, and each is obtained by interpolating $m = 2$ examples. The total number of loss terms per mini-batch is again b.

5.3.2 MultiMix

Interpolation The number of generated examples per mini-batch is now $n \gg b$ and the number of examples being interpolated is $m = b$. Given a mini-batch of b examples with embeddings Z and targets Y, we draw interpolation vectors $\lambda_k \sim \text{Dir}(\alpha)$ for $k = 1, \ldots, n$, where $\text{Dir}(\alpha)$ is the symmetric Dirichlet distribution and $\lambda_k \in \Delta^{m-1}$, that is, $\lambda_k \geq 0$ and $\mathbf{1}_m^\top \lambda_k = 1$. We then interpolate embeddings and targets by taking n convex combinations over all m examples:

$$\widetilde{Z} = Z\Lambda \tag{5.6}$$

$$\widetilde{Y} = Y\Lambda, \tag{5.7}$$

where $\Lambda = (\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^{b \times n}$. We thus generalize manifold mixup [Verma, 2019]:

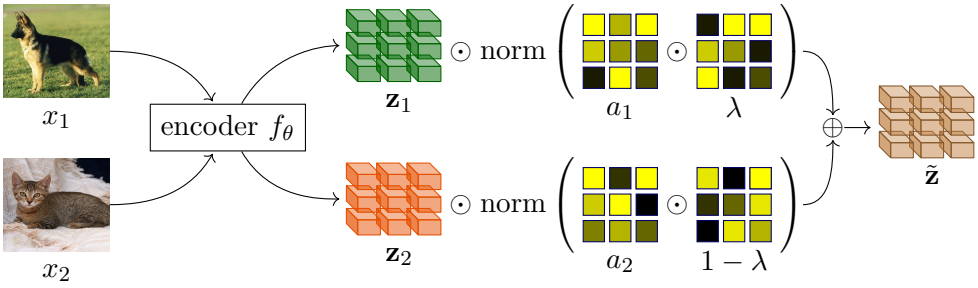


Figure 5.2 – *Dense MultiMix* (subsection 5.3.3) for the special case $m = 2$ (two examples), $n = 1$ (one interpolated embedding), $r = 9$ (spatial resolution 3×3). The embeddings $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^{d \times 9}$ of input images x_1, x_2 are extracted by encoder f_θ. Attention maps $a_1, a_2 \in \mathbb{R}^9$ are extracted (5.9), multiplied element-wise with interpolation vectors $\lambda, (1-\lambda) \in \mathbb{R}^9$ (5.10) and ℓ_1-normalized per spatial position (5.11). The resulting weights are used to form the interpolated embedding $\tilde{\mathbf{z}} \in \mathbb{R}^{d \times 9}$ as a convex combination of $\mathbf{z}_1, \mathbf{z}_2$ per spatial position (5.12). Targets are interpolated similarly (5.13).

1. from b to an arbitrary number $n \gg b$ of generated examples: interpolated embeddings $\widetilde{Z} \in \mathbb{R}^{d \times n}$ (5.6) *vs.* $\mathbb{R}^{d \times b}$ in (5.3), targets $\widetilde{Y} \in \mathbb{R}^{c \times n}$ (5.7) *vs.* $\mathbb{R}^{c \times b}$ in (5.4);
2. from pairs ($m = 2$) to a tuple of length $m = b$, containing the entire mini-batch: m-term convex combination (5.6),(5.7) *vs.* 2-term in (5.3),(5.4), Dirichlet *vs.* Beta distribution;
3. from fixed λ across the mini-batch to a different λ_k for each generated example.

Loss Again, we replace the original mini-batch embeddings Z by the interpolated embeddings $\widetilde{Z}$ and minimize the average cross-entropy $H(\widetilde{Y}, \widetilde{P})$ (5.1) between the predicted probabilities $\widetilde{P} = g_W(\widetilde{Z})$ and the interpolated targets $\widetilde{Y}$ (5.7). Compared with (5.2), the mini-batch loss becomes

$$L_M(X, Y; \theta, W) := H(Y\Lambda, g_W(f_\theta(X)\Lambda)). \tag{5.8}$$

The total number of loss terms per mini-batch is now $n \gg b$.

5.3.3 Dense MultiMix

We now extend to the case where the embeddings are structured, *e.g.* in tensors. This happens *e.g.* with token *vs.* sentence embeddings in NLP and patch *vs.* image embeddings in vision. It works by removing spatial pooling and applying the loss function densely over

all tokens/patches. The idea is illustrated in Figure 5.2. For the sake of exposition, our formulation uses sets of matrices grouped either by example or by spatial position. In practice, all operations are on tensors.

Preliminaries The encoder is now $f_\theta : \mathcal{X} \to \mathbb{R}^{d \times r}$, mapping the input x to an embedding $\mathbf{z} = f_\theta(x) \in \mathbb{R}^{d \times r}$, where d is the number of channels and r is its spatial resolution—if there are more than one spatial dimensions, these are flattened.

Given a mini-batch of b examples, we have again inputs $X = (x_1, \ldots, x_b) \in \mathbb{R}^{D \times b}$ and targets $Y = (y_1, \ldots, y_b) \in \mathbb{R}^{c \times b}$. Each embedding $\mathbf{z}_i = f_\theta(x_i) = (z_i^1, \ldots, z_i^r) \in \mathbb{R}^{d \times r}$ for $i = 1, \ldots, b$ consists of features $z_i^j \in \mathbb{R}^d$ for spatial position $j = 1, \ldots, r$. We group features by position in matrices $Z^1, \ldots, Z^r$, where $Z^j = (z_1^j, \ldots, z_b^j) \in \mathbb{R}^{d \times b}$ for $j = 1, \ldots, r$.

Attention In the absence of dense targets, each spatial location inherits the target of the corresponding input example. This is weak supervision, because the target object is not visible everywhere. To select the most reliable locations, we define a level of confidence according to an attention map. Given an embedding $\mathbf{z} \in \mathbb{R}^{d \times r}$ with target $y \in \mathcal{Y}$ and a vector $u \in \mathbb{R}^d$, the *attention map*

$$a = h(\mathbf{z}^\top u) \in \mathbb{R}^r \tag{5.9}$$

measures the similarity of features of $\mathbf{z}$ to u, where h is a non-linearity, *e.g.* softmax or ReLU followed by ℓ_1 normalization. There are different ways to define vector u. For example, $u = \mathbf{z}\mathbf{1}_r/r$ by global average pooling (GAP) of $\mathbf{z}$, or $u = Wy$ assuming a linear classifier with $W \in \mathbb{R}^{d \times c}$, similar to class activation mapping (CAM) [Zhou, 2016]. In case of no attention, $a = \mathbf{1}_r/r$ is uniform.

Given a mini-batch, let $a_i = (a_i^1, \ldots, a_i^r) \in \mathbb{R}^r$ be the attention map of embedding $\mathbf{z}_i$ (5.9) for $i = 1, \ldots, b$. We group attention by position in vectors $a^1, \ldots, a^r$, where $a^j = (a_1^j, \ldots, a_b^j) \in \mathbb{R}^b$ for $j = 1, \ldots, r$. Figure 5.6 in the supplementary shows the attention obtained by (5.9). We observe high confidence on the entire or part of the object. Where confidence is low, we assume the object is not visible and thus the corresponding interpolation factor should be low.

Interpolation There are again $n \gg b$ generated examples per mini-batch, with $m = b$ examples being densely interpolated. For each spatial position $j = 1, \ldots, r$, we draw interpolation vectors $\lambda_k^j \sim \mathrm{Dir}(\alpha)$ for $k = 1, \ldots, n$ and define $\Lambda^j = (\lambda_1^j, \ldots, \lambda_n^j) \in \mathbb{R}^{m \times n}$.

Since input examples are assumed to contribute according to the attention vector $a^j \in \mathbb{R}^m$, we scale the rows of Λ^j accordingly and normalize its columns back to Δ^{m-1} to define convex combinations:

$$M^j = \operatorname{diag}(a^j)\Lambda^j \tag{5.10}$$

$$\hat{M}^j = M^j \operatorname{diag}(\mathbf{1}_m^\top M^j)^{-1} \tag{5.11}$$

We then interpolate embeddings and targets by taking n convex combinations over m examples:

$$\widetilde{Z}^j = Z^j \hat{M}^j \tag{5.12}$$

$$\widetilde{Y}^j = Y \hat{M}^j. \tag{5.13}$$

This is similar to (5.6),(5.7), but there is a different interpolated embedding matrix $\widetilde{Z}^j \in \mathbb{R}^{d\times n}$ as well as target matrix $\widetilde{Y}^j \in \mathbb{R}^{c\times n}$ per position, even though the original target matrix Y is one. The total number of interpolated features and targets per mini-batch is now nr.

Classifier The classifier is now $g_W : \mathbb{R}^{d\times r} \to \mathbb{R}^{c\times r}$, maintaining the same spatial resolution as the embedding and generating one vector of predicted probabilities per spatial position. This is done by removing average pooling or any down-sampling operation. The interpolated embeddings $\widetilde{Z}^1, \ldots, \widetilde{Z}^r$ (5.12) are grouped by example into $\widetilde{\mathbf{z}}_1, \ldots, \widetilde{\mathbf{z}}_n \in \mathbb{R}^{d\times r}$, mapped by g_W to predicted probabilities $\widetilde{\mathbf{p}}_1, \ldots, \widetilde{\mathbf{p}}_n \in \mathbb{R}^{c\times r}$ and grouped again by position into $\widetilde{P}^1, \ldots, \widetilde{P}^r \in \mathbb{R}^{c\times n}$.

In the simple case where the original classifier is linear, *i.e.* $W \in \mathbb{R}^{d\times c}$, it is seen as 1×1 convolution and applied densely to each column (feature) of $\widetilde{Z}^j$ for $j = 1, \ldots, r$.

Loss Finally, we learn parameters θ, W by minimizing the *weighted cross-entropy* $H(\widetilde{Y}^j, \widetilde{P}^j; s)$ of $\widetilde{P}^j$ relative to the interpolated targets $\widetilde{Y}^j$ again densely at each position j, where

$$H(Y, P; s) := -\mathbf{1}_c^\top (Y \odot \log(P)) s / (\mathbf{1}_n^\top s) \tag{5.14}$$

generalizes (5.1) and the weight vector is defined as $s = \mathbf{1}_m^\top M^j \in \mathbb{R}^n$. This is exactly the vector used to normalize the columns of M^j in (5.11). The motivation is that the columns of M^j are the original interpolation vectors weighted by attention: A small ℓ_1 norm indi-

cates that for the given position j, we are sampling from examples of low attention, hence the loss is to be discounted. The total number of loss terms per mini-batch is now nr.

5.4 Experiments

5.4.1 Setup

We use a mini-batch of size $b = 128$ examples in all experiments. Following manifold mixup [Verma, 2019], for every mini-batch, we apply MultiMix with probability 0.5 or input mixup otherwise. For MultiMix, the default settings are given in subsection 4.4.5. We use PreActResnet-18 (R-18) [He, 2016b] and WRN16-8 [Zagoruyko, 2016b] as encoder on CIFAR-10 and CIFAR-100 datasets [Krizhevsky, 2009]; R-18 on TinyImagenet [Yao, 2015] (TI); and Resnet-50 (R-50) and ViT-S/16 [Dosovitskiy, 2021] on ImageNet [Russakovsky, 2015]. To better understand the effect of mixup in ViT, we evaluate MultiMix and Dense MultiMix on ImageNet without using strong augmentations like Auto-Augment [Cubuk, 2018], Rand-Augment [Cubuk, 2020], random erasing [Zhong, 2020] and CutMix [Yun, 2019]. We reproduce TransMix [Chen, 2022] and TokenMix [Liu, 2022a] using these settings.

Settings and hyperparameters We train MultiMix and Dense MultiMix with mixed examples only. We use a mini-batch of size $b = 128$ examples in all experiments. Following Manifold Mixup [Verma, 2019], for every mini-batch, we apply MultiMix with probability 0.5 or input mixup otherwise. For input mixup, we interpolate the standard $m = b$ pairs (5.5). For MultiMix, we use the entire network as the encoder f_θ by default, except for the last fully-connected layer, which we use as classifier g_W. We use $n = 1000$ tuples and draw a different $\alpha \sim U[0.5, 2.0]$ for each example from the Dirichlet distribution by default. For multi-GPU experiments, all training hyperparameters including m and n are per GPU.

For Dense MultiMix, the spatial resolution is $r = 4 \times 4 = 16$ on CIFAR-10/100 and $r = 7 \times 7 = 49$ on Imagenet by default. We obtain the attention map by (5.9) using GAP for vector u and ReLU followed by ℓ_1 normalization as non-linearity h by default. To predict class probabilities and compute the loss densely, we use the classifier g_W as 1×1 convolution by default; when interpolating at earlier layers, we follow the process described in subsection 5.3.3.

CIFAR-10/100 training Following the experimental settings of AlignMixup [Venkataramanan, 2021], we train MultiMix and its variants using SGD for 2000 epochs using the same random seed as AlignMixup. We set the initial learning rate to 0.1 and decay it by a factor of 0.1 every 500 epochs. The momentum is set to 0.9 and the weight decay to 0.0001. We use a batch size $b = 128$ and train on a single NVIDIA RTX 2080 TI GPU for 10 hours.

TinyImageNet training Following the experimental settings of PuzzleMix [Kim, 2020a], we train MultiMix and its variants using SGD for 1200 epochs, using the same random seed as AlignMixup. We set the initial learning rate to 0.1 and decay it by a factor of 0.1 after 600 and 900 epochs. The momentum is set to 0.9 and the weight decay to 0.0001. We train on two NVIDIA RTX 2080 TI GPUs for 18 hours.

ImageNet training Following the experimental settings of PuzzleMix [Kim, 2020a], we train MultiMix and its variants using the same random seed as AlignMixup. We train R-50 using SGD with momentum 0.9 and weight decay 0.0001 and ViT-S/16 using AdamW with default parameters. The initial learning rate is set to 0.1 and 0.01, respectively. We decay the learning rate by 0.1 at 100 and 200 epochs. We train on 32 NVIDIA V100 GPUs for 20 hours.

Tasks and metrics We use top-1 accuracy (%, higher is better) and top-1 error (%, lower is better) as evaluation metrics on *image classification* and *robustness to adversarial attacks* (subsection 5.4.2). Additional datasets and metrics are reported separately for *transfer learning to object detection* (subsection 5.4.3) and *out-of-distribution detection* (subsection 5.4.4)

5.4.2 Results: Image classification and robustness

Image classification In Table 5.2 we observe that MultiMix and Dense MultiMix already outperform SoTA on all datasets except CIFAR-10 with R-18, where they are on par with Co-Mixup. Dense MultiMix improves over vanilla MultiMix and its effect is complementary on all datasets. On TI for example, Dense MultiMix improves over MultiMix by 1.33% and SoTA by 1.59%.

In Table 5.3 we observe that on ImageNet with R-50, vanilla MultiMix outperforms all methods except AlignMixup. Dense MultiMix outperforms all SoTA with both R-50

Dataset	Cifar-10		Cifar-100		TI
Network	R-18	W16-8	R-18	W16-8	R-18
Baseline†	95.41±0.02	94.93±0.06	76.69±0.26	78.80±0.55	56.49±0.21
Input mixup [Zhang, 2018a]†	95.98±0.10	96.18±0.06	79.39±0.40	80.16±0.1	56.60±0.16
CutMix [Yun, 2019]†	96.79±0.04	96.48±0.04	80.56±0.09	80.25±0.41	56.87±0.39
Manifold mixup [Verma, 2019]†	97.00±0.05	96.44±0.02	80.00±0.34	80.77±0.26	59.31±0.49
PuzzleMix [Kim, 2020a]†	97.04±0.04	97.00±0.03	79.98±0.05	80.78±0.23	63.52±0.42
AugMix⋆ [Hendrycks, 2019b]	96.67±0.05	–	80.10±0.03	–	–
Co-Mixup [Kim, 2021b]†	**97.10**±0.03	96.44±0.08	80.28±0.13	80.39±0.34	64.12±0.43
SaliencyMix [Uddin, 2021]†	96.94±0.05	96.27±0.05	80.36±0.56	80.29±0.05	66.14±0.51
StyleMix [Hong, 2021]†	96.25±0.04	96.27±0.04	80.01±0.79	79.77±0.17	63.88±0.27
StyleCutMix [Hong, 2021]†	96.94±0.05	96.95±0.04	80.67±0.07	80.79±0.04	66.55±0.13
SuperMix [Dabouei, 2021a]‡	96.03±0.05	96.13±0.05	79.07±0.26	79.42±0.05	64.43±0.39
AlignMixup [Venkataramanan, 2021]†	97.06±0.04	96.91±0.01	81.71±0.07	81.24±0.02	66.85±0.07
ζ-Mixup [Abhishek, 2022]⋆	96.26±0.04	96.35±0.04	80.46±0.26	79.73±0.15	63.18±0.14
MultiMix (ours)	97.07±0.03	97.06±0.02	81.82±0.04	81.44±0.03	67.11±0.04
Dense MultiMix (ours)	97.09±0.02	**97.09**±0.02	**81.93**±0.04	**81.77**±0.03	**68.44**±0.05
Gain	**-0.01**	**+0.09**	**+0.22**	**+0.53**	**+1.59**

Table 5.2 – *Image classification* on CIFAR-10/100 and TI (TinyImagenet). Top-1 accuracy (%): higher is better. R: PreActResnet, W: WRN. ⋆: reproduced, †: reported by AlignMixup, ‡: reproduced with same teacher and student model. **Bold black**: best; Blue: second best; underline: best baseline. Gain: improvement over best baseline.

and ViT-S/16, bringing an overall gain of 3% over the baseline with R-50 and 2.2% with ViT-S/16. The gain over AlignMixup with R-50 is small, but it is impressive that it comes with only linear interpolation. To better isolate the effect of each method, we reproduce TransMix [Chen, 2022] and TokenMix [Liu, 2022a] with ViT-S/16 using their official code with our settings, *i.e.*, without strong regularizers like CutMix, Auto-Augment, Random-Augment *etc.* MultiMix is on par, while Dense MultiMix outperforms them by 1%.

Training speed Table 5.3 also shows the training speed as measured on NVIDIA V-100 GPU including forward and backward pass. The vanilla MultiMix has nearly the same speed with the baseline, bringing an accuracy gain of 2.49% with R-50. Dense MultiMix is slightly slower, increasing the gain to 3.10%. The inference speed is the same for all methods.

Robustness to adversarial attacks We follow the experimental settings of AlignMixup [Venkataramanan, 2021] and use 8/255 l_∞ ϵ-ball for FGSM [Goodfellow, 2015] and 4/255 l_∞ ϵ-ball with step size 2/255 for PGD [Madry, 2018] attack. In Table 5.4 we observe that MultiMix is already more robust than SoTA on all datasets and settings. Dense MultiMix also increases the robustness and is complementary.

Network	Resnet-50		ViT-S/16	
Method	Speed	Acc	Speed	Acc
Baseline†	1.17	76.32	1.01	73.9
Input mixup [Zhang, 2018a]†	1.14	77.42	0.99	74.1
CutMix [Yun, 2019]†	1.16	78.60	0.99	74.2
Manifold mixup [Verma, 2019]†	1.15	77.50	0.97	74.2
PuzzleMix [Kim, 2020a]†	0.84	78.76	0.73	74.7
AugMix [Hendrycks, 2019b]⋆	1.12	77.70	–	–
Co-Mixup [Kim, 2021b]†	0.62	–	0.57	74.9
SaliencyMix [Uddin, 2021]†	1.14	78.74	0.96	74.8
StyleMix [Hong, 2021]†	0.99	75.94	0.85	74.8
StyleCutMix [Hong, 2021]†	0.76	77.29	0.71	74.9
SuperMix [Dabouei, 2021a]‡	0.92	77.60	–	–
TransMix [Chen, 2022]⋆	–	–	1.01	75.1
TokenMix [Liu, 2022a]⋆	–	–	0.87	75.3
AlignMixup [Venkataramanan, 2021]†	1.03	79.32	–	–
MultiMix (ours)	1.16	78.81	0.98	75.2
Dense MultiMix (ours)	0.95	**79.42**	0.88	**76.1**
Gain		**+0.1**		**+1.2**

Table 5.3 – *Image classification and training speed* on ImageNet. Top-1 accuracy (%): higher is better. Speed: images/sec ($\times 10^3$): higher is better. †: reported by AlignMixup; ⋆: reproduced; ‡: reproduced with same teacher and student model. **Bold black**: best; Blue: second best; underline: best baseline. Gain: improvement over best baseline.

The overall gain is more impressive than in classification according to Table 5.2. For example, against the strong PGD attack on CIFAR-10 with W16-8, the SoTA Co-Mixup improves the baseline by 3.8% while Dense MultiMix improves it by 7.3%, which is double. MultiMix and Dense MultiMix outperform Co-Mixup and PuzzleMix by 3-6% in robustness on CIFAR-10, even though they are on-par on classification. There is also a significant gain over SoTA AlignMixup by 1-3% in robustness to FGSM on TinyImageNet and to the stronger PGD.

5.4.3 Results: Transfer learning to object detection

We evaluate the effect of mixup on the generalization ability of a pre-trained network to object detection as a downstream task. Following the settings of CutMix [Yun, 2019], we pre-train R-50 on ImageNet with mixup methods and use it as the backbone for SSD [Liu, 2016a] with fine-tuning on Pascal VOC07+12 [Everingham, 2010] and Faster-RCNN [Ren, 2015] with fine-tuning on MS-COCO [Lin, 2014].

Attack	FGSM					PGD				
Dataset	Cifar-10		Cifar-100		TI	Cifar-10		Cifar-100		
Network	R-18	W16-8	R-18	W16-8	R-18	R-18	W16-8	R-18	W16-8
Baseline†	88.8±0.11	88.3±0.33	87.2±0.10	72.6±0.22	91.9±0.06	99.9±0.0	99.9±0.01	99.9±0.01	99.9±0.01
Input mixup [Zhang, 2018a]†	79.1±0.07	79.1±0.12	81.4±0.23	67.3±0.06	88.7±0.08	99.7±0.02	99.4±0.01	99.9±0.01	99.3±0.02
CutMix [Yun, 2019]†	77.3±0.06	78.3±0.05	86.9±0.06	60.2±0.04	88.6±0.03	99.8±0.03	98.1±0.02	98.6±0.01	97.9±0.01
Manifold mixup [Verma, 2019]†	76.9±0.14	76.0±0.04	80.2±0.06	56.3±0.10	89.3±0.06	97.2±0.01	98.4±0.03	99.6±0.01	98.4±0.03
PuzzleMix [Kim, 2020a]†	57.4±0.22	60.7±0.02	78.8±0.09	57.8±0.03	83.8±0.05	97.7±0.01	97.0±0.01	96.4±0.02	95.2±0.03
AugMix [Hendrycks, 2019b]⋆	58.2±0.02	–	79.1±0.04	–	–	98.2±0.01	–	96.3±0.02	–
Co-Mixup [Kim, 2021b]†	60.1±0.05	58.8±0.10	77.5±0.02	56.5±0.04	–	97.5±0.02	96.1±0.03	95.3±0.03	94.2±0.01
SaliencyMix [Uddin, 2021]†	57.4±0.08	68.0±0.05	77.8±0.10	58.1±0.06	81.1±0.06	97.4±0.03	97.0±0.04	95.6±0.03	93.7±0.05
StyleMix [Hong, 2021]†	80.0±0.23	71.2±0.21	80.6±0.15	68.2±0.17	85.1±0.16	98.1±0.09	97.5±0.07	98.3±0.09	98.3±0.09
StyleCutMix [Hong, 2021]†	57.7±0.04	56.0±0.07	77.4±0.05	56.8±0.03	80.5±0.04	97.8±0.04	96.7±0.02	91.8±0.01	93.7±0.01
SuperMix [Dabouei, 2021a]‡	60.0±0.11	58.2±0.12	78.8±0.13	58.3±0.19	81.1±0.12	97.6±0.02	97.2±0.09	91.4±0.03	92.7±0.01
AlignMixup [Venkataramanan, 2021]†	54.8±0.03	56.0±0.05	74.1±0.04	55.0±0.03	78.8±0.03	95.3±0.04	96.7±0.03	90.4±0.01	92.1±0.03
ζ-Mixup [Abhishek, 2022]⋆	72.8±0.23	67.3±0.24	75.3±0.21	68.0±0.21	84.7±0.18	98.0±0.06	98.6±0.03	97.4±0.10	96.1±0.10
MultiMix (ours)	**54.1**±0.09	55.3±0.04	73.8±0.04	54.5±0.01	77.5±0.01	94.2±0.04	94.8±0.01	90.0±0.01	91.6±0.01
Dense MultiMix (ours)	**54.1**±0.01	**53.3**±0.03	**73.5**±0.03	**52.9**±0.04	**75.5**±0.04	**92.9**±0.04	**92.6**±0.01	**88.6**±0.03	**90.8**±0.01
Gain	**+0.7**	**+2.7**	**+0.6**	**+2.1**	**+3.3**	**+2.4**	**+3.5**	**+1.4**	**+1.3**

Table 5.4 – *Robustness to FGSM & PGD attacks.* Top-1 error (%): lower is better. ⋆: reproduced, †: reported by AlignMixup. ‡: reproduced, same teacher and student model. **Bold black**: best; Blue: second best; underline: best baseline. Gain: reduction of error over best baseline. TI: TinyImagenet. R: PreActResnet, W: WRN.

Dataset	VOC07+12		MS-COCO	
Detector	SSD	Speed	FR-CNN	Speed
Baseline†	76.7	9.7	33.27	23.6
Input mixup†	76.6	9.5	34.18	22.9
CutMix†	77.6	9.4	35.16	23.2
AlignMixup†	78.4	8.9	35.84	20.4
MultiMix (ours)	77.9	9.6	35.93	23.2
Dense MultiMix (ours)	**79.2**	8.8	**36.19**	19.8
Gain	**+0.8**		**+0.35**	

Table 5.5 – *Transfer learning* to object detection. Mean average precision (mAP, %): higher is better. †: reported by AlignMixup. **Bold black**: best; Blue: second best; underline: best baseline. Gain: increase in mAP. Speed: images/sec: higher is better.

Metric	ECE	OE
Baseline	10.25	1.11
Input Mixup [Zhang, 2018a]	18.50	1.42
Manifold Mixup [Verma, 2019]	18.41	0.79
CutMix [Yun, 2019]	7.60	1.05
PuzzleMix [Kim, 2020a]	8.22	0.61
Co-Mixup [Kim, 2021b]	5.83	0.55
AlignMixup [Venkataramanan, 2021]	5.78	0.41
MultiMix (ours)	5.63	0.39
Dense MultiMix (ours)	**5.28**	**0.27**

Table 5.6 – *Model calibration* using R-18 on CIFAR-100. ECE: expected calibration error; OE: overconfidence error. Lower is better.

In Table 5.5, we observe that, while MultiMix is slightly worse than AlignMixup on Pascal VOC07+12, Dense MultiMix brings improvements over the SoTA on both datasets and is still complementary. This is consistent with classification results. Dense MultiMix brings a gain of 0.8 mAP on Pascal VOC07+12 and 0.35 mAP on MS-COCO.

5.4.4 Reducing overconfidence

Model calibration A standard way to evaluate over-confident predictions is to measure *model calibration*. We assess model calibration using MultiMix and Dense MultiMix on CIFAR-100. We report *mean calibration error* (mCE) and *overconfidence error* (OE)

Task	Out-Of-Distribution Detection											
Dataset	LSUN (crop)				iSUN				TI (crop)			
Metric	Det Acc	AuROC	AuPR (ID)	AuPR (OOD)	Det Acc	AuROC	AuPR (ID)	AuPR (OOD)	Det Acc	AuROC	AuPR (ID)	AuPR (OOD)
Baseline†	54.0	47.1	54.5	45.6	66.5	72.3	74.5	69.2	61.2	64.8	67.8	60.6
Input mixup [Zhang, 2018a]†	57.5	59.3	61.4	55.2	59.6	63.0	60.2	63.4	58.7	62.8	63.0	62.1
Cutmix [Yun, 2019]†	63.8	63.1	61.9	63.4	67.0	76.3	81.0	77.7	70.4	84.3	87.1	80.6
Manifold mixup [Verma, 2019]†	58.9	60.3	57.8	59.5	64.7	73.1	80.7	76.0	67.4	69.9	69.3	70.5
PuzzleMix [Kim, 2020a]†	64.3	69.1	80.6	73.7	73.9	77.2	79.3	71.1	71.8	76.2	78.2	81.9
AugMix [Hendrycks, 2019b]⋆	62.9	73.2	80.8	72.6	68.2	78.7	81.1	74.1	71.4	83.9	84.6	78.6
Co-Mixup [Kim, 2021b]†	70.4	75.6	82.3	70.3	68.6	80.1	82.5	75.4	71.5	84.8	86.1	80.5
SaliencyMix [Uddin, 2021]†	68.5	79.7	82.2	64.4	65.6	76.9	78.3	79.8	73.3	83.7	87.0	82.0
StyleMix [Hong, 2021]†	62.3	64.2	70.9	63.9	61.6	68.4	67.6	60.3	67.8	73.9	71.5	78.4
StyleCutMix [Hong, 2021]†	70.8	78.6	83.7	74.9	70.6	82.4	83.7	76.5	75.3	82.6	82.9	78.4
SuperMix [Dabouei, 2021a]‡	70.9	77.4	80.1	72.3	71.0	76.8	79.6	76.7	75.1	82.8	82.5	78.6
AlignMixup [Venkataramanan, 2021]†	74.2	79.9	84.1	75.1	72.8	83.2	84.1	80.3	77.2	85.0	87.8	85.0
ζ-Mixup [Abhishek, 2022]⋆	68.1	73.2	80.8	73.1	72.2	82.3	82.2	79.4	74.4	84.3	82.2	77.2
MultiMix (ours)	79.2	82.6	85.2	77.6	75.6	85.1	87.8	83.1	78.3	86.6	89.0	**88.2**
Dense MultiMix (ours)	**80.8**	**84.3**	**85.9**	**78.0**	**76.8**	**85.4**	**88.0**	**84.6**	**81.4**	89.0	**90.8**	88.0
Gain	**+6.6**	**+4.4**	**+1.8**	**+2.9**	**+2.9**	**+2.2**	**+3.9**	**+4.3**	**+4.2**	**+4.0**	**+3.0**	**+3.2**

Table 5.7 – *Out-of-distribution detection* using R-18. Det Acc (detection accuracy), AuROC, AuPR (ID) and AuPR (OOD): higher is better. ⋆: reproduced, †: reported by AlignMixup. ‡: reproduced, same teacher and student model. **Bold black**: best; Blue: second best; underline: best baseline. Gain: increase in performance. TI: TinyImagenet.

in Table 5.6. MultiMix has lower error than all SoTA methods and Dense MultiMix even lower.

Out-of-distribution detection This is another standard way to evaluate over-confidence. Here, *in-distribution* (ID) are examples on which the network has been trained, and *out-of-distribution* (OOD) are examples drawn from any other distribution. Given a mixture of ID and OOD examples, the network should predict an ID example with high confidence and an OOD example with low confidence, *i.e.*, the confidence of the predicted class should be below a certain threshold.

Following AlignMixup [Venkataramanan, 2021], we compare MultiMix and its variants with SoTA methods trained using R-18 on CIFAR-100 as ID examples, while using LSUN [Yu, 2015], iSUN [Xiao, 2010] and TI to draw OOD examples. We use detection accuracy, Area under ROC curve (AuROC) and Area under precision-recall curve (AuPR) as evaluation metrics. In Table 5.7, we observe that MultiMix and Dense MultiMix outperform SoTA on all datasets and metrics by a large margin. Although the gain of MultiMix and Dense MultiMix over SoTA mixup methods is small on image classification, they significantly reduce over-confident incorrect predictions and achieve superior performance on out-of-distribution detection.

Domain	Clipart	Real-World	Product	Art
DAML [Shu, 2021]	45.13	65.99	61.54	53.13
MultiMix (ours)	46.01	66.59	60.99	54.58
Dense MultiMix (ours)	**46.32**	**66.87**	**62.28**	**56.01**

Table 5.8 – *Generalizing to unseen domains.* Image classification using R-18 on Office-Home dataset [Venkateswara, 2017] under the open-domain setting, using the official settings of DAML [Shu, 2021]. Accuracy (%): higher is better.

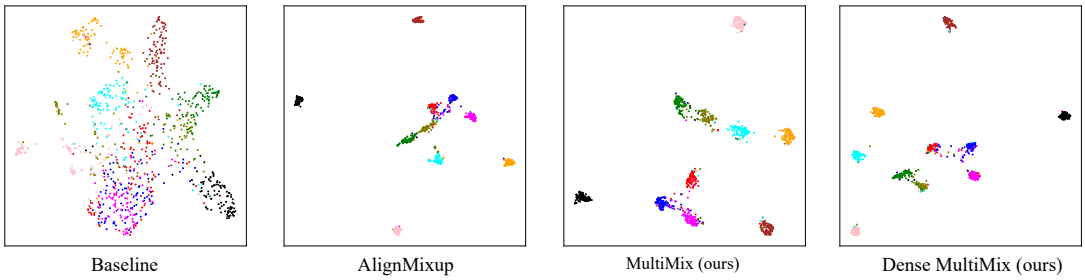


Figure 5.3 – *Embedding space visualization* for 100 test examples per class of 10 randomly chosen classes of CIFAR-100 with PreActResnet-18, using UMAP [McInnes, 2018].

5.4.5 Generalizing to unseen domains

We evaluate the ability of MultiMix and Dense MultiMix to generalize to unseen domains on the Office-Home dataset [Venkateswara, 2017] under the open-domain setting, using the official settings of DAML [Shu, 2021]. Table 5.8 shows that, while both MultiMix and DAML use the Dirichlet distribution to sample interpolation weights, MultiMix and Dense MultiMix generalize to unseen domains better than DAML. We hypothesize this is due to sampling an arbitrarily large number of samples. In addition, Dense MultiMix brings significant gain, up to nearly 3%.

5.4.6 Analysis of the embedding space

Qualitative analysis We qualitatively analyze the embedding space on 10 CIFAR-100 classes in Figure 5.3. We observe that the quality of embeddings of the baseline is extremely poor with severely overlapping classes, which explains its poor performance on image classification. All mixup methods result in clearly better clustered and more uniformly spread classes. AlignMixup [Venkataramanan, 2021] yields five somewhat clustered classes and five moderately overlapping ones. Our best setting, *i.e.*, Dense MultiMix, re-

sults in five tightly clustered classes and another five somewhat overlapping but less than all competitors.

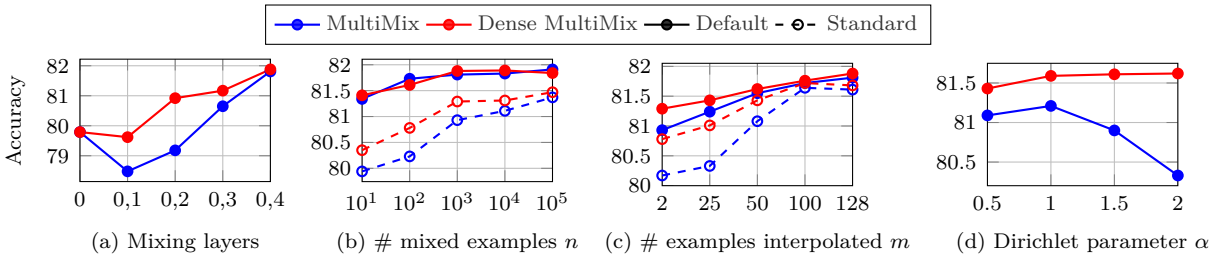
Quantitative analysis We also quantitatively assess the embedding space on the CIFAR-100 test set using alignment and uniformity [Wang, 2020]. *Alignment* measures the expected pairwise distance of examples in the same class. Lower alignment indicates that the classes are more tightly clustered. *Uniformity* measures the (log of the) expected pairwise similarity of all examples using a Gaussian kernel as a similarity function. Lower uniformity indicates that classes are more uniformly spread in the embedding space.

On CIFAR-100, we obtain alignment 3.02 for baseline, 2.04 for AlignMixup, 1.27 for MultiMix and 0.92 for Dense MultiMix. We also obtain uniformity -1.94 for the baseline, -2.38 for AlignMixup [Venkataramanan, 2021], -4.77 for MultiMix and -5.68 for Dense MultiMix. These results validate the qualitative analysis of Figure 5.3.

5.4.7 Manifold intrusion

Manifold intrusion [Guo, 2019b] can occur when mixed examples are close to classes other than the ones being interpolated in the embedding space. To evaluate for manifold intrusion, we define the *intrusion distance* (ID) as the minimum distance of a mixed embedding to the clean embeddings of all classes except the ones being interpolated, averaged over a mini-batch: $\mathrm{ID}(\widetilde{Z}, Z) = \frac{1}{|\widetilde{Z}|}\sum_{\tilde{z}\in\widetilde{Z}} \min_{z\in Z} \|\tilde{z} - z\|^2$. Here, $\widetilde{Z}$ is the set of mixed embeddings in a mini-batch and Z is the set of clean embeddings from all classes other than the ones being interpolated in the mini-batch. Intuitively, a larger $\mathrm{ID}(\widetilde{Z}, Z)$ denotes that mixed embeddings in $\widetilde{Z}$ are farther away from the manifold of other classes in Z, thereby preventing manifold intrusion.

Averaged over the training set of CIFAR-100 using Resnet-18, the intrusion distance is 0.46 for Input Mixup, 0.47 for Manifold Mixup, 0.45 for AlignMixup, 0.46 for MultiMix and 0.47 for Dense MultiMix. This is roughly the same for most SoTA mixup methods. This may be due to the fact that true data occupy only a tiny fraction of the embedding space, thus generated mixed examples lie in empty space between class-specific manifolds with high probability. The visualization in Figure 5.3 indeed shows that the embedding space is sparsely populated, even in two dimensions. This sparsity is expected to grow exponentially in the number of dimensions, which is in the order of 10^3.



(a) Mixing layers (b) # mixed examples n (c) # examples interpolated m (d) Dirichlet parameter α

Figure 5.4 – *Ablation study* on CIFAR-100 using R-18. (a) Interpolation layers (R-18 block; 0: input mixup). (b) Number n of interpolated examples per mini-batch with $m = b$ (Default) and $m = 2$ (Standard). (c) Number m of examples being interpolated, with $n = 1000$ (Default) and $n = 100$ (Standard). (d) Fixed value of Dirichlet parameter α.

5.4.8 Ablations

All ablations are performed using R-18 on CIFAR-100. We study the effect of the layer where we interpolate, the number n of generated examples per mini-batch, the number m of examples being interpolated and the Dirichlet parameter α. More ablations are given in the supplementary.

Interpolation layer For MultiMix, we use the entire network as the encoder f_θ by default, except for the last fully-connected layer, which we use as classifier g_W. Thus, we *interpolate embeddings* in the deepest layer by default. Here, we study the effect of different decompositions of the network $f = g_W \circ f_\theta$, such that interpolation takes place at a different layer. In Figure 5.4(a), we observe that mixing at the deeper layers of the network significantly improves performance. The same behavior is observed with Dense MultiMix, which validates our default choice.

It is interesting that the authors of input mixup [Zhang, 2018a] found that convex combinations of three or more examples in the input space with weights from the Dirichlet distribution do not bring further gain. This agrees with the finding of SuperMix [Dabouei, 2021a] for four or more examples. Figure 5.4(a) suggests that further gain emerges when mixing in deeper layers.

Number n of generated examples per mini-batch This is important since our aim is to increase the amount of data seen by the model, or at least part of the model. We observe from Figure 5.4(b) that accuracy increases overall with n and saturates for $n \geq 1000$ for both variants of MultiMix. The improvement is more pronnounced when $m = 2$, which is standard for most mixup methods. Our best solution, Dense MultiMix,

Method	Vanilla	Dense
Baseline	76.76	78.16
Input mixup [Zhang, 2018a]	79.79	80.21
CutMix [Yun, 2019]	80.63	81.40
Manifold mixup [Verma, 2019]	80.20	80.87
PuzzleMix [Kim, 2020a]	79.99	80.62
Co-Mixup [Kim, 2021b]	80.19	80.84
SaliencyMix [Uddin, 2021]	80.31	81.21
StyleMix [Hong, 2021]	79.96	80.76
StyleCutMix [Hong, 2021]	80.66	81.41
SuperMix [Dabouei, 2021a]‡	79.01	80.12
AlignMixup [Venkataramanan, 2021]	81.71	81.36
MultiMix (ours)*	**81.81**	81.84
MultiMix (ours)	**81.81**	**81.88**

Table 5.9 – *The effect of dense loss.* Image classification on CIFAR-100 using R-18. Top-1 accuracy (%): higher is better. ‡: reproduced with same teacher and student model. *: Instead of Dense MultiMix, we only apply the loss densely.

works best at $n = 1000$ and $n = 10,000$. We choose $n = 1000$ as default, given also that the training cost increases with n. The training speed as a function of n is given in the supplementary and is nearly constant for $n \leq 1000$.

Number m of examples being interpolated We vary m between 2 (pairs) and $b = 128$ (entire mini-batch) by using $\Lambda' \in \mathbb{R}^{m \times n}$ drawn from Dirichlet along with combinations (subsets) over the mini-batch to obtain $\Lambda \in \mathbb{R}^{b \times n}$ with m nonzero elements per column in (5.6),(5.7). We observe in Figure 5.4(c) that for both MultiMix and Dense MultiMix the performance increases with m. The improvement is more pronnounced when $n = 100$, which is similar to the standard setting ($n = b = 128$) of most mixup methods. Our choice of $m = b = 128$ brings an improvement of 1-1.8% over $m = 2$. We use this as our default setting.

Dirichlet parameter α Our default setting is to draw α uniformly at random from $[0.5, 2]$ for every interpolation vector (column of Λ). Here we study the effect of a fixed value of α. In Figure 5.4(d), we observe that the best accuracy comes with $\alpha = 1$ for most MultiMix variants, corresponding to the uniform distribution over the convex hull of the mini-batch embeddings. However, all measurements are lower than the default $\alpha \sim U[0.5, 2]$. For example, from Table 5.2(a) (CIFAR-100, R-18), Dense MultiMix has accuracy 81.93, compared with 81.59 in Figure 5.4(d) for $\alpha = 1$.

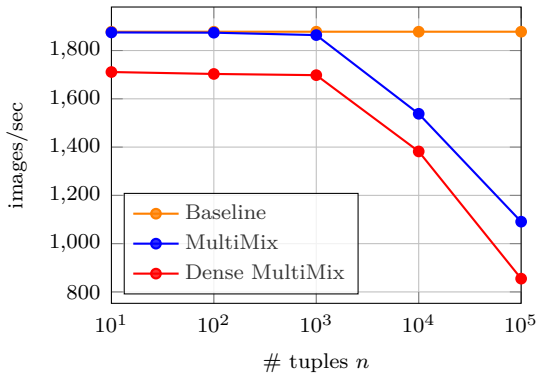


Figure 5.5 – *Training speed* (images/sec) of MultiMix and its variants *vs.* number of tuples n on CIFAR-100 using R-18. Measured on NVIDIA RTX 2080 TI GPU, including forward and backward pass.

Mixup methods with dense loss In Table 5.9 we observe that dense interpolation and dense loss improve MultiMix. Here, we study the effect of the dense loss only when applied to SoTA mixup methods; dense interpolation is not straightforward or not applicable in general with other methods.

Given a mini-batch of b examples, we follow the mixup strategy of the SoTA mixup methods to obtain the mixed embedding $\widetilde{Z}^j \in \mathbb{R}^{d\times b}$ for each spatial position $j = 1, \ldots, r$. Then, as discussed in subsection 5.3.3, we obtain the predicted class probabilities $\widetilde{P}^j \in \mathbb{R}^{c\times b}$ again for each $j = 1, \ldots, r$. Finally, we compute the cross-entropy loss $H(\widetilde{Y}, \widetilde{P}^j)$ (5.1) densely at each spatial position j, where the interpolated target label $\widetilde{Y} \in \mathbb{R}^{c\times b}$ is given by (5.4).

In Table 5.9, we observe that using a dense loss improves the performance of all SoTA mixup methods. The baseline improves by 1.4% accuracy (76.76 → 78.16) and manifold mixup by 0.67% (80.20 → 80.87). On average, we observe a gain of 0.7% brought by the dense loss. An exception is AlignMixup [Venkataramanan, 2021], which drops by 0.35% (81.71 → 81.36). This may be due to the alignment process, whereby the interpolated dense embeddings are not very far from the original. MultiMix and Dense MultiMix still improve the state of the art under this setting.

Training speed In Figure 5.5, we analyze the training speed of MultiMix and Dense MultiMix as a function of number n of interpolated examples. In terms of speed, MultiMix is on par with the baseline up to $n = 1000$, while bringing an accuracy gain of 5%. The best performing method—Dense MultiMix—is only slower by 10.6% at $n = 1000$ as compared to the baseline, which is arguably worth given the impressive 5.12% accuracy gain. Further

m	2	25	50	100
Input Mixup [Zhang, 2018a]	77.44	78.29	78.98	79.52
Manifold Mixup [Verma, 2019]	78.63	79.41	79.87	80.32
MultiMix (ours)	**80.90**	**81.30**	**81.60**	**81.80**

Table 5.10 – *Effect of batch size* $m < 128$. Image classification using R-18 on CIFAR-100. Top-1 accuracy (%): higher is better.

METHOD	u	h	ACC
Uniform	–	–	81.33
Attention (5.9)	CAM	softmax	81.21
	CAM	$\ell_1 \circ$ relu	81.63
	GAP	softmax	81.78
	GAP	$\ell_1 \circ$ relu	**81.88**

Table 5.11 – *Variants of spatial attention* in Dense MultiMix. Image classification on CIFAR-100 using R-18. Top-1 accuracy (%): higher is better. GAP: Global Average Pooling; CAM: Class Activation Maps [Zhou, 2016]; $\ell_1 \circ$ relu: ReLU followed by ℓ_1 normalization.

increasing beyond $n > 1000$ brings a drop in training speed, due to computing Λ and then using it to interpolate (5.6),(5.7). Because $n > 1000$ also brings little performance benefit according to Figure 5.4(b), we set $n = 1000$ as default for all MultiMix variants.

Using a smaller batch size We compare Input Mixup, Manifold Mixup and MultiMix for image classification using R-18 on CIFAR-100, with a batch size $b < 128$. By default, we use $n = 1000$ generated examples and $m = b$ examples being interpolated, following the same experimental settings described in subsection 5.4.1. As shown in Table 5.10, the increase of MultiMix accuracy with increasing batch size b is similar to the increase with the number m of examples being interpolated, as observed in Figure 5.4(c). This is to be expected because, m and b are increasing together. An exhaustive hyper-parameter sweep could result in a different observation; currently, hyper-parameters are adjusted to the default choice $m = b = 128$. We also observe that the performance improvement of MultiMix over Input or Manifold Mixup is higher for smaller batch size. This may be due to the ability of MultiMix to draw from a larger pool of interpolated examples.

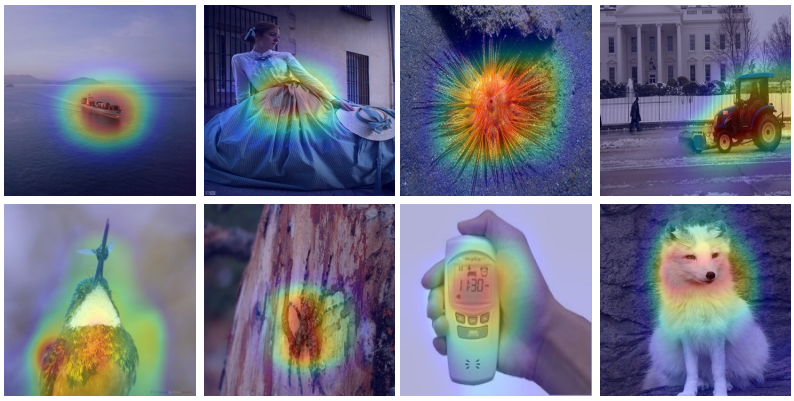


Figure 5.6 – *Attention visualization.* Attention maps obtained by (5.9) with u as GAP and h as $\ell_1 \circ \text{relu}$ using Resnet-50 on the validation set of ImageNet. The attention localizes the complete or part of the object with high confidence.

Dense MultiMix: Spatial attention In subsection 5.3.3, we discuss different options for attention in dense MultiMix. In particular, no attention amounts to defining a uniform $a = \mathbf{1}_r/r$. Otherwise, a is defined by (5.9). The vector u can be defined as $u = \mathbf{z}\mathbf{1}_r/r$ by global average pooling (GAP) of $\mathbf{z}$, which is the default, or $u = Wy$ assuming a linear classifier with $W \in \mathbb{R}^{d \times c}$. The latter is similar to class activation mapping (CAM) [Zhou, 2016], but here the current value of W is used online while training. The non-linearity h can be softmax or ReLU followed by ℓ_1 normalization ($\ell_1 \circ \text{relu}$), which is the default. Here, we study the affect of these options on the performance of dense Multimix.

In Table 5.11, we observe that using GAP for u and $\ell_1 \circ \text{relu}$ as h yields the best performance overall. Changing GAP to CAM or $\ell_1 \circ \text{relu}$ to softmax is inferior. The combination of CAM with softmax is the weakest, even weaker than uniform attention. CAM may fail because of using the non-optimal value of W while training; softmax may fail because of being too selective. Compared to our best setting, uniform attention is clearly inferior, by nearly 0.6%. This validates that the use of spatial attention in dense MultiMix is clearly beneficial. Our intuition is that in the absence of dense targets, assuming the same target of the entire example at every spatial position naively implies that the object of interest is present everywhere, whereas spatial attention provides a better hint as to where the object may really be.

We validate this hypothesis in Figure 5.6, where we visualize the attention maps obtained using our best setting with u as GAP and h as $\ell_1 \circ \text{relu}$. This shows that the attention map enables dense targets to focus on the object regions, which explains its superior performance.

Dense MultiMix: Spatial resolution We study the effect of spatial resolution on dense MultiMix. By default, we use a resolution of 4×4 at the last residual block of R-18 on CIFAR-100. Here, we additionally investigate 1×1 (downsampling by average pooling with kernel size 4, same as GAP), 2×2 (downsampling by average pooling with kernel size 2) and 8×8 (upsampling by using stride 1 in the last residual block). We measure accuracy 81.07% for spatial resolution 1×1, 81.43% for for 2×2, 81.88% for 4×4 and 80.83% for 8×8. We thus observe that performance improves with spatial resolution up to 4×4, which is the optimal, and then drops at 8×8. This drop may be due to assuming the same target at each spatial position. The resolution 8×8 is also more expensive computationally.

5.5 Discussion

The take-home message of this work is that, instead of devising smarter and more complex interpolation functions in the input space or intermediate features, it is more beneficial to use MultiMix, even though its interpolation is only linear. In this work, we combine three elements:

1. Increase the number n of generated mixed examples beyond the mini-batch size b.
2. Increase the number m of examples being interpolated from $m = 2$ (pairs) to $m = b$.
3. Perform interpolation in the *embedding* space rather than the input space.

Figure 5.4 shows that all three elements are important in achieving SoTA performance and removing any one leads to sub-optimal results. We discuss their significance and interdependence here.

Increasing the number n of generated examples The *expected risk* is defined as an integral over the underlying continuous data distribution. Since that distribution is unknown, the integral is approximated by a finite sum, *i.e.*, the *empirical risk*. A better approximation is the *vicinal risk*, where a number of augmented examples is sampled from a distribution in the vicinity of each training example, thus *increasing the number of loss terms per training example*. Input mixup [Zhang, 2018a] is inspired by the vicinal risk. However, as a practical implementation, all mixup methods still generate b mixed examples and thus incur b loss terms for a mini-batch of size b. As discussed in section 5.2, previous works have used more loss terms than b per mini-batch, but not for mixup.

Our hypothesis for the significance of element 1 is that more mixed examples, thus

more loss terms by interpolation, provide a better approximation of the expected risk integral. Dense interpolation further increases the number of loss terms, thus further improving the quality of approximation.

Increasing the number m of examples being interpolated As discussed in section 5.2, previous works, starting from input mixup [Zhang, 2018a] have attempted to interpolate $m > 2$ examples in the input space by sampling λ from Dirichlet or other distributions but have found the idea not effective for large m. Our finding is that element 2 becomes effective only by interpolating in the embedding space, that is, element 3.

Interpolating in embedding space Element 3 is originally motivated by Manifold Mixup [Verma, 2019], where "interpolations in deeper hidden layers capture higher level information [Zeiler, 2014]." ACAI [Berthelot, 2018] explicitly studies interpolation in the latent space of an autoencoder to produce a smooth semantic warping effect in data space. This suggests that nearby points in the latent space are semantically similar, which in turn improves representation learning. Mixed examples generated by sampling in the embedding space lie on the learned manifold. We hypothesize that the learned manifold is a good surrogate of the true, unknown data manifold.

A natural extension of this work is to settings other than supervised classification. A limitation is that it is not straightforward to combine the sampling scheme of MultiMix with complex interpolation methods, unless they are fast to compute in the embedding space.

Chapter 6

Learning Strong Image Encoders from Videos

Our previous chapters explored the potential of interpolation-based data augmentation for improving representation learning for image classification (chapter 3, chapter 5) and deep metric learning (chapter 4) in a supervised setting. While these methods effectively generate synthetic augmentations, we now venture beyond this approach to investigate the possibility of discovering natural augmentations inherent in real-world data. This shift aligns with the core theme of the thesis, which focuses on combining diverse learning objectives and modalities to enhance representation learning.

The inherent richness of videos presents a unique opportunity to explore natural augmentations. Unlike synthetic augmentations, videos naturally encompass diverse variations in pose, deformation, viewpoint, perspective, occlusion, and background clutter, offering a wealth of rich augmentations for learning robust representations. This eliminates the need for artificially generated augmentations such as mixup based interpolations, allowing the model to learn from the intrinsic complexities present within the video data.

To facilitate this exploration, we first introduce a novel dataset of open-source first-person videos specifically recorded for virtual "walking tours" inspired by [Wiles, 2022]). These videos possess several key advantages. Firstly, these videos exhibit a high density of diverse semantic categories within individual frames. Secondly, the videos directly capture the perspective of a human, minimizing the presence of cuts, special effects, and are long (1-3 hours). Lastly, the entire dataset is readily viewable, fostering transparency and facilitating deeper understanding of the learning process.

Our primary objective is to leverage the rich information embedded within these video frames to build robust representations. However, standard self-supervised learning methods such as DINO [Caron, 2021] often rely on establishing correspondences between different views. While it is relatively straightforward to establish correspondences in images, it becomes more challenging when dealing with temporal deformations, requiring some

form of object tracking. In videos with a large field of view or ego-motion, obtaining correspondences becomes even more difficult.

This chapter delves into addressing these challenges and explores the potential of natural video augmentations for self-supervised representation learning. We propose a novel approach that leverages the inherent variations within video data to learn robust representations, potentially surpassing the limitations of traditional methods that rely on synthetic augmentations. This effort was presented in as an Oral (top 1.2%) in the Twelfth International Conference on Learning Representations (ICLR), 2024.

6.1 Introduction

> *(To the question "Have you read all the books in here?")* No, only four of them. But I read those very, very carefully.
>
> Jacques Derrida

Learning from large scale datasets has been at the core of great progress. In particular, the field of self-supervised learning has allowed pretraining of neural networks to scale beyond the size of labelled datasets. By avoiding costly annotation, strong performance has been demonstrated by increasing the training dataset sizes into billions of images.

But how well are those images really used? At a rate of one image per second, a dataset of 1B images would take 317 years to watch. Yet, humans develop functioning visual systems *much faster i.e.* face recognition [Haan, 2001] and color sensitivity [Adams, 1987] is developed in three months, depth perception in five months [Campos, 1978] and visual acuity in six months [Sokol, 1978]. Besides potential genetic visual priors in humans, one stark difference is the *type* of data. Humans observe their visual surroundings in one continuous stream, only interrupted by sleep. Indeed, learning visual representations of images from videos is not new. However, previous works have found significant gaps in performance to image-pretrained models. They have mostly used object-centric videos scraped from the internet, and adapted image-based pretraining methods to use different frames as an extra form of data augmentation [Gordon, 2020; Parthasarathy, 2023].

In this work, we investigate two directions. First, in the direction of *data*, we introduce a new dataset of open-source first-person videos, recorded for the purpose of virtual "walking tours", inspired by [Wiles, 2022]. These videos have several advantages. Not only are the individual frames dense in semantic categories – much more so than movies, as we analyze – but these videos also directly represent the viewpoint of a human, contain few or no shot cuts nor special effects and are long (1-3h). Another benefit is their transparency: indeed, one can watch the whole dataset in one setting. The dataset we create contains 10 Walking Tours (WT) videos with CC-BY license.

Second, in the direction of the *method*, we develop a new self-supervised image-pretraining method that is uniquely suited for learning from natural, non-object-centric videos. Our approach is inspired by observing toddlers first learn to track objects and animals, then to recognize and differentiate them [Bomba, 1983; Quinn, 1993; Spelke, 2007]. Our method, called DoRA, is an end-to-end training approach that "tracks to learn to

Figure 6.1 – *Examples of frames from the Walking Tours dataset*, containing hours-long, continuous egocentric 4K videos from urban scenes in different cities, under CC-BY license. There are a large number of objects and actions in a variety and natural transition of places, *e.g.* residential area, park, market, waterfront, *etc.*, with natural transition of lighting conditions and object augmentations.

recognize": given a video clip, objects in an initial frame are implicitly **D**isc**o**vered and t**RA**cked across time. The tracked objects are incentivized to be diverse by introducing a Sinkhorn-Knopp clustering of patch embeddings; the tracked instances are used as a learning signal for a classical multi-view SSL loss.

Surprisingly, contrary to previous works, we find that our novel method obtains ImageNet-level performances by training on *a single WT video*, as evidenced by performances on segmentation and object detection downstream tasks. While humorously intentioned, Derrida's quote rings true to this finding and our results give some hope for alternative directions in SSL that depart from blind dataset scaling towards more efficient and smarter use of existing video data.

To summarize, our key contributions in this work are as follows:

1. We introduce a new dataset of 10 WT videos, with single-video and mixed-video splits. The latter is conveniently equal in size to ImageNet. We analyze their usefulness compared to existing video and image datasets.
2. We propose a new end-to-end self-supervised visual pretraining method called DoRA. It builds upon DINO but is tailored to promote tracking of multiple objects across frames. We use it to learn strong image encoders and trace the source of its improvements through extensive ablations.
3. We obtain strong performance on ADE20k segmentation and MS COCO detection, outperforming ImageNet-pretrained DINO, while instead pretraining on a single long video.

6.2 Related Work

Self-supervised learning of image encoders from video data is a very active area of research. Video, and more generally temporal streams, have long been theorized to be ideal signals for unsupervised learning [Wiskott, 2002]. In computer vision, early methods have been very diverse and included pretext tasks such as egomotion prediction [Agrawal, 2015; Jayaraman, 2015], active recognition [Jayaraman, 2016], pose estimation [Chakraborty, 2017], unsupervised object discovery [Croitoru, 2017], dense prediction [Pathak, 2017; Li, 2019b], optical flow [Mahendran, 2018; Xiong, 2021], frame order prediction [Misra, 2016], view-point matching [Sermanet, 2018; Pirk, 2020] or learning visual correspondences [Wang, 2019a].

More recently, there have been considerable advances in self-supervised learning using ImageNet, with the main theme being extracting multiple augmentations of an image [Chen, 2020b; Caron, 2021] and training models to pull them together/apart. These methods have since percolated to learning from video frames [Gordon, 2020; Parthasarathy, 2023; Tschannen, 2020; Wang, 2015; Orhan, 2020]. Similar to this work, TimeTuning [Salehi, 2023] leverages the passage of time in videos by not treating it as simple augmentations. However, in contrast to our work, it requires an already image-pretrained backbone. VITO [Parthasarathy, 2023] improves performance relative to ImageNet, by using VideoNet, a large YouTube dataset of 10s videos from a similar class distribution and the same number of examples as ImageNet. In this paper, we show that it is possible to obtain strong results from a *single* long video, with a very different visual distribution compared to ImageNet / VideoNet.

6.3 Walking Tours Dataset

6.3.1 Dataset collection and properties

We collect from YouTube a new dataset of urban scenes called "Walking Tours" (WTours, or WT) comprising 10 egocentric videos of a person walking in different cities in Europe and Asia. The cities include Amsterdam, Bangkok, Chiang Mai, Istanbul, Kuala Lampur, Singapore, Stockholm, Venice, and Zurich. We also include a video from a Wildlife safari. Examples are shown in Figure 6.1. These videos are captured in 4K resolution (3840 $\times$ 2160 pixels) at 60 frames-per-second and are under Creative Commons License (CC-BY). The minimum video duration is 59 minutes (Wildlife safari), the

Dataset	Domain	Ego	Pre	Bal	Annot	Avg. Dur (sec)	Dur (hr)	#Videos	Frame resolution
Diverse Pretraining									
Kinetics-400 [Kay, 2017]	Actions	✗	✓	✓	**Class**	10.2	851	400	340 × 255
AVA [Gu, 2018]	Actions	✗	✓	✓	**Class**	900	107.5	80	320 × 400
WebVid-2M [Bain, 2021]	Open	✗	✓	✗	**Weak**	18	13k	–	320 × 240
HowTo100M [Miech, 2019]	Instructions	✗	✓	✗	**Weak**	4	135k	–	–
Egocentric									
Epic-Kitchens [Damen, 2022]	Cooking	✓	✗	✗	**Loc.**	510	100	37	1920 × 1080
Ego-4D [Grauman, 2022]	Daily	✓	✗	✗	**Loc.**	1446	120	931	1920 × 1080
Meccano [Ragusa, 2023]	Industry	✓	✗	✗	**Loc.**	1247	849	20	1920 × 1080
Assembly-101 [Sener, 2022]	Assembly	✓	✗	✗	**Loc.**	426	167	362	1920 × 1080
ImageNet-aligned									
R2V2 [Gordon, 2020]	ImageNet	✗	✓	✓	**Class**	–	–	–	467 × 280
VideoNet [Parthasarathy, 2023]	ImageNet	✗	✓	✓	**Class**	10	3055	–	
Walking Tours (ours)	Urban	✓	✓	✗	**None**	4968	23	10	3840 × 2160

Table 6.1 – *Walking Tours* vs. *existing video datasets*. Ego: egocentric; Pre: used for pretraining; Bal: class balance control; Annot: annotation type. Weak: associated data per clip (text or other modality); Class: class label per frame or clip; Loc: localization per frame (*e.g.* bounding box, segmentation, mask 3D pose). Avg. Dur: average duration per video; Dur: total duration.

maximum is 2 hours 55 minutes (Bangkok) and the average is 1 hour 38 minutes. Such videos are particularly interesting for visual learning because of the following properties:

1. *Large number of objects and actions.* Each frame or clip taken from a video depicts several objects and actions, *e.g.* walking, riding a bike, sitting, drinking *etc.*
2. *Natural transition in lighting conditions.* In some videos, the lighting gradually transitions from bright (late afternoon) to dim (dusk) then to dark (post sunset).
3. *Natural transition in scenes.* The videos depict transitions between places, *e.g.* from city center to market place to residential areas to parks to water fronts *etc.*
4. *Natural object augmentations.* Continuous variation *e.g.* of pose, deformation, viewpoint, perspective distortion, relative object position, occlusion, background clutter.

The abundance of information within these videos, encompassing a multitude of objects and complex scenes, presents a formidable challenge for manual annotation or curation, making it appropriate for unsupervised pretraining. To the best of our knowledge, we are the first to propose an egocentric video dataset for pretraining and evaluate it on a wealth of downstream tasks.

In Table 6.2, we provide statistics of individual videos in the WTours dataset. In the following, we first compare WTours with other video datasets and then analyze WTours

Video	(a) Properties			(b) Analysis		
	Domain	#Frames (×1000)	Duration (min)	#Shots	#Objects / Frame (avg.)	#Classes
Amsterdam	Urban	147.4	72.6	0	48	684
Bangkok	Urban	314.7	153.0	12	44	703
Chiang Mai	Urban	122.2	87.5	2	40	711
Istanbul	Urban	122.4	82.8	0	42	604
Kuala Lampur	Urban	131.1	77.2	0	39	689
Singapore	Urban	174.0	71.0	0	49	732
Stockholm	Urban	119.7	70.4	0	32	590
Venice	Urban	197.8	90.0	0	39	701
Wildlife	Wildlife	85.7	59.3	1	12	374
Zurich	Urban	117.0	64.0	1	40	572
Average		**153.2**	**82.8**	**1.6**	**38.5**	**636**

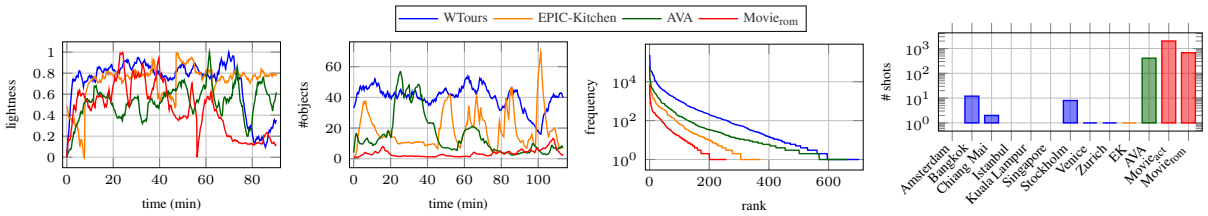
Table 6.2 – *Individual WTours video statistics.* (a) Properties. (b) Analysis: shots detected by [Castellano,]; objects and classes detected by Detic [Zhou, 2022b], trained on ImageNet-21k.

videos in terms of automatically extracted information, including lightness, shot changes and number of objects and categories depicted.

6.3.2 Comparison with other video datasets

In Table 6.1, we compare WTours with existing video datasets. Self-supervised pretraining on videos has been mostly limited to video datasets that rely on weak annotation in the form of video-text pairs [Bain, 2021; Miech, 2019] or even are curated, *e.g.* their class balance is controlled, even if their annotation is unused [Kay, 2017]. Their average clip duration is small, *e.g.* less than 20 sec, and their resolution is also small, limiting the capacity to detect objects at a greater distance. By contrast, WTours videos are continuous, hours-long at high resolution and provide natural transitions of scenes and viewing conditions. They are not curated and thus better suited for the self-supervised setting.

ImageNet-aligned datasets such as R2V2 [Gordon, 2020] and VideoNet [Parthasarathy, 2023] contain videos that are curated and annotated with the same distribution and classes as ImageNet, meant for pretraining image encoders. These videos are short, *i.e.* 10 seconds on average. By contrast, WTours consists of a continuous stream of egocentric video, where the average number of classes is close to that of ImageNet, as shown in subsection 6.3.3. The rich information contained in 4K resolution, together with a high number of objects in a frame, makes it appropriate for representation learning. Importantly, the continuity and absence of curation make it more realistic and more comparable with human learning.



(a) Lightness (b) #Objects per frame (c) Total #classes (d) Shots

Figure 6.2 – *Dataset analysis* of a WTours video compared with videos from Epic-Kitchens [Damen, 2022], AVA [Gu, 2018] and two entire movies, concatenated or cropped to match the duration of the WTours video. (a) Lightness *vs*. time. (b) Number of objects per frame *vs*. time. (c) Frequency of classes in entire video. (d) Number of shots. Objects detected by Detic [Zhou, 2022b], trained on ImageNet-21k.

Our dataset does not rely on a set of objects, human activities or other search terms but instead is *data-first* and more open-ended.

Egocentric videos Despite the large number of high-quality videos, *egocentric* video datasets [Damen, 2022; Grauman, 2022; Sener, 2022] have been used only for downstream tasks and thus come with extensive annotation. In comparison, WTours has 4-10 times longer average duration and twice the frame resolution. While WTours is smaller in terms of total duration and number of videos, it is scalable under the self-supervised setting since it requires no human labeling effort and more videos can be easily found, downloaded or even made. This makes collecting more data as simple as a walk in the park.

Very long video datasets. A large dataset of 10k WTours videos was created recently by [Wiles, 2022] but was not publicly released and not studied for self-supervised learning. Another dataset having hour-long videos is introduced in [Khan, 2020a], in the context of sports analytics; it has not been explored for self-supervised learning either.

6.3.3 Dataset analysis

In Figure 6.2, we analyse the properties of a single WTours video compared with videos of the same length from two other datasets, as well as two movie videos, an action movie and a romantic movie.

Variation in lightness We measure the change in perceived brightness using the lightness value (L) across consecutive frames. From Figure 6.2(a), we observe a gradual shift

at roughly 150 min into the WTours video, transitioning from bright to dim to dark. By contrast, Epic-Kitchens and AVA videos exhibit random brightness fluctuations, alternating between dim and bright conditions. Typically, self-supervised pretraining happens on datasets with uniform brightness levels. Datasets featuring such brightness variations are less expored.

Variation in number of objects Using Detic [Zhou, 2022b], a DETR-style object detector trained on ImageNet-21k, we detect objects in each frame. Figure 6.2(b) shows the number of objects per frame and Figure 6.2(c) shows their frequency in the entire video. We observe that the WTours video contains 703 unique object categories, while Epic-Kitchens has 373, AVA has 663 and Movie-2 has 259. The unique objects appear more frequently and there are more unique objects per frame in WTours than in the other datasets. This makes WTours semantically richer, despite coming from one continuous stream of video. Using videos with a large number of objects can encourage the model to capture complex relations and variations in the data. Detailed statistics of objects and classes per WTours video are given in Table 6.2(b). Except for the wildlife video, WTours videos in general contain 40 or more objects per frame and 600 or more unique object categories per video.

Variation in shots Egocentric videos are typically captured in a single uninterrupted take, with exceptions being post-processed special effects or cuts. In Figure 6.2(d), we find that, on average, WTours and Epic-Kitchens videos contain only one or two shots per entire video, while AVA contains 406, an action movie ($\text{Movie}_{\text{act}}$) [Skiptrace,] contains 2000 and a romantic movie ($\text{Movie}_{\text{rom}}$) [Central,] contains 667. The substantial number of shots in movies and AVA poses challenges for representation learning methods that rely on object tracking or optical flow. In subsection 6.5.5, we show that WTours significantly outperforms movies in downstream tasks, which may be attributed to the absence of cuts. The number of shots per WTours video is also given in Table 6.2(b).

6.4 Attention-based multi-object tracking

Our goal is to build robust representations by leveraging the rich information in video frames. Standard SSL frameworks [Chen, 2020b; Caron, 2020] often assume correspondences between different views. This is true whether using dense [Zhou, 2022a] or global

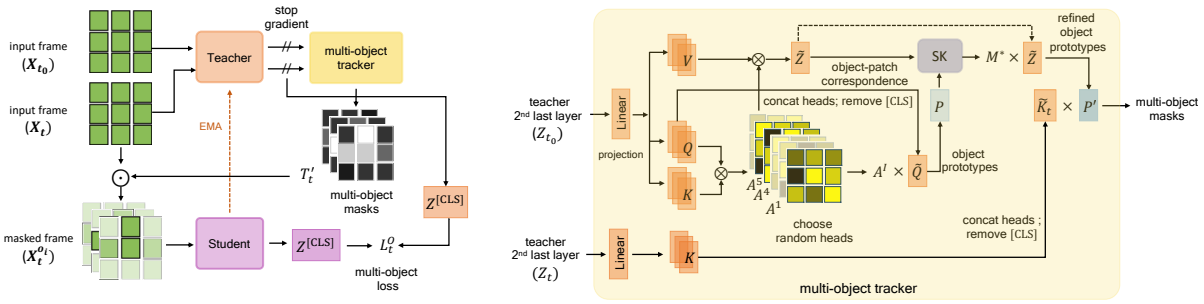


Figure 6.3 – *DoRA, our self-supervised image pretraining method from video.* (Left) From an input frame $\mathbf{X}_{t_0}$, the output of the second-last layer of the teacher model is used by a multi-object tracker to generate cross-attention maps T'_t with frame $\mathbf{X}_t$. We use those to mask $\mathbf{X}_t$ (6.7), feed it to the student model and apply a *distillation loss* L_t^{O} between [CLS] token embeddings (6.8). (Right) In the tracker, we obtain the query Q, key K and output Z embeddings. From the multi-head attention maps A^i (6.1), we draw a subset $\mathcal{I}$ of k heads and form object prototypes P by pooling over patch queries $\tilde{Q}$ (6.2). We refine them into P' to *discover distinct objects*, using Sinkhorn-Knopp (SK) to establish correspondences M^* between P and patch embeddings $\tilde{Z}$ (6.4) and pooling over $\tilde{Z}$ (6.5). We then *track the objects* over frames $\mathbf{X}_t$ by cross-attention T'_t with patch key embeddings $\tilde{K}_t$ (6.6).

representations by pooling [Caron, 2021]. While it is relatively straightforward to establish correspondences in images, it becomes more challenging when dealing with temporal deformations, requiring some form of object tracking [Salehi, 2023]. In videos with a large field of view or ego-motion, obtaining correspondences becomes even more difficult.

High-level idea We introduce DoRA, based on multi-object **Disco**very and t**RA**cking. As shown in Figure 6.3, it leverages the attention from the [CLS] token of distinct heads in a vision transformer to identify and consistently track multiple objects within a given frame across temporal sequences. On these, a teacher-student distillation loss is then applied. Importantly, we do not use any off-the-shelf object tracker or optical flow network. This keeps our pipeline simple and does not require any additional data or training. It also ensures that the learned representation is robust.

Preliminaries We are given a video clip consisting of T frames $\mathbf{X}_t \in \mathbb{R}^{h \times w \times c}$ for $t \in \{1, \ldots, T\}$, where $h \times w$ is the spatial resolution and c is the number of channels. Each frame is split into $n = hw/p^2$ non-overlapping patches of resolution $p \times p$. The patches are linearly projected into embeddings of dimension d and a [CLS] token embedding is

prepended. This representation is input to a *transformer encoder* [Dosovitskiy, 2021]. The *output embeddings* are $Z_t = g_\theta(\mathbf{X}_t) \in \mathbb{R}^{(n+1)\times d}$, where mapping g_θ includes the tokenizer and encoder, while θ denotes its learnable parameters. Given an embedding $Z \in \mathbb{R}^{(n+1)\times d}$, we write $Z = [Z^{[\text{CLS}]}; \tilde{Z}]$, where $Z^{[\text{CLS}]} \in \mathbb{R}^{1\times d}$ is the [CLS] token embedding and $\tilde{Z} \in \mathbb{R}^{n\times d}$ are the *patch embeddings*.

Following DINO [Caron, 2021], there is a student network with parameters θ and a teacher network with identical architecture and parameters θ' obtained as the *exponential moving average* (EMA) of θ according to $\theta' \leftarrow \alpha\theta' + (1-\alpha)\theta$. The encoder is followed by a *head* that includes an MLP and a scaled softmax, such that the output token embeddings can be interpreted as probabilities. We denote by f_θ the mapping that includes the tokenizer, encoder and head.

Discovering objects with multi-head attention Starting at a first frame $\mathbf{X}_{t_0}$, we obtain the query and key embeddings $Q, K \in \mathbb{R}^{(n+1)\times d}$ from the last transformer layer of the teacher network [1]. According to *multi-head* attention, these embeddings are partitioned as $Q = [Q^1, \ldots, Q^h]$, $K = [K^1, \ldots, K^h]$, where $Q^i, K^i \in \mathbb{R}^{(n+1)\times d/h}$ for $i = 1, \ldots, h$ and h is the number of heads. For each head i, the *self-attention* matrix $A^i \in \mathbb{R}^{(n+1)\times(n+1)}$ is based on the dot-product similarity between the query and key embeddings:

$$A^i := \text{softmax}\left(Q^i (K^i)^\top / \sqrt{d}\right) \in \mathbb{R}^{(n+1)\times(n+1)}. \tag{6.1}$$

Given an attention matrix $A \in \mathbb{R}^{(n+1)\times(n+1)}$, let $A^{[\text{CLS}]} := [a_{1,2}, \ldots, a_{1,n}] \in \mathbb{R}^{1\times n}$ be the *[CLS]-attention* vector between the [CLS] and patch embeddings, where $a_{i,j}$ is the element (i,j) of A. We draw at random a subset $\mathcal{I} := \{i_1, \ldots, i_k\}$ of $k < h$ heads and collect their [CLS]-attention vectors into $A^{\mathcal{I}} := [(A^{i_1})^{[\text{CLS}]}; \ldots; (A^{i_k})^{[\text{CLS}]}] \in \mathbb{R}^{k\times n}$. Intuitively, as expressed in rows of matrix $A^{\mathcal{I}}$, the different heads attend to different *objects* in the frame [Caron, 2021].

To represent the k objects in the embedding space, we use matrix $A^{\mathcal{I}} \in \mathbb{R}^{k\times n}$ to form linear combinations of patch embeddings $\tilde{Q} \in \mathbb{R}^{n\times d}$, obtaining *object prototypes*

$$P := A^{\mathcal{I}} \tilde{Q} \in \mathbb{R}^{k\times d}. \tag{6.2}$$

This can be seen as the representation of k different [CLS] tokens in the full embedding space, capturing k objects at frame t_0. Then, given the key embeddings $K_t \in \mathbb{R}^{(n+1)\times d}$ at

1. For simplicity, we drop t_0 from the notation.

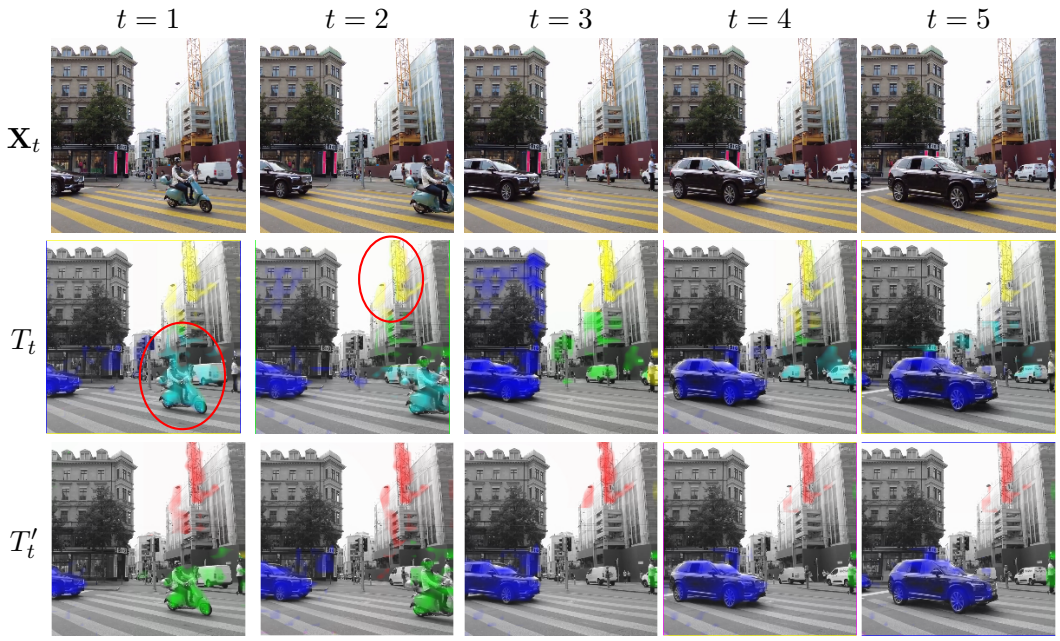


Figure 6.4 – For each input frame t of a video clip (top), cross-attention map $T_t \in \mathbb{R}^{k\times n}$ (6.3) (middle) and refined cross-attention map $T'_t \in \mathbb{R}^{k\times n}$ (6.6) (bottom), using Sinkhorn-Knopp algorithm. For each object, one row of T_t or T'_t is reshaped as $h/p \times w/p$ and upsampled to an $h \times w$ *attention map* overlaid on the input frame for $k = 3$ objects encoded in blue, red and green channel. Mixed colors yellow and cyan for T_t (middle, in red circle) indicate spatial overlap of two objects, while T'_t (bottom) yields three well separated objects shown in primary colors blue, red and green.

another frame t, we could track the objects by *cross-attention*

$$T_t := \text{softmax}\left(P\tilde{K}_t^\top/\sqrt{d}\right) \in \mathbb{R}^{k\times n}, \tag{6.3}$$

where $\tilde{K}_t \in \mathbb{R}^{n\times d}$. Unfortunately, we observe in Figure 6.4 that the k attention maps obtained this way are spatially overlapping, meaning that each attention map is not delineating a single object.

Establishing object-patch correspondences To discover spatially distinct objects, we propose to establish correspondences between prototypes and patch tokens. Let $Z = g_{\theta'}(\mathbf{X}_{t_0}) \in \mathbb{R}^{(n+1)\times d}$ be the output embeddings of the teacher network, still at frame t_0. We seek a correspondence between the rows of $P \in \mathbb{R}^{k\times d}$ and $\tilde{Z} \in \mathbb{R}^{n\times d}$, where $\tilde{Z}$ are the patch token embeddings.

The goal is to find a *transport plan* $M \in \mathbb{R}^{k\times n}$ that minimizes the expected pairwise

cost $C := -P\tilde{Z}^\top \in \mathbb{R}^{k\times n}$ between prototypes and patches, while incorporating an entropic regularizer with coefficient ϵ. Matrix M is non-negative with row-wise sum $1/k$ and column-wise sum $1/n$, representing a joint probability over P and $\tilde{Z}$ with uniform marginals. The minimal solution M^* is unique and can be found by forming the matrix $e^{-C/\epsilon}$ and then applying the Sinkhorn-Knopp (SK) algorithm [Cuturi, 2013], *i.e.*, iteratively normalizing its rows and columns:

$$M^* = \mathrm{SK}\left(\exp\left(P\tilde{Z}^\top/\epsilon\right)\right) \in \mathbb{R}^{k\times n}, \tag{6.4}$$

Observe the similarity with (6.1) and (6.3), where scaling is by $\sqrt{d}$ rather than ϵ, exp is included in softmax and normalization is on rows only rather than iterative. Then, similarly with (6.2), we use the optimal transport plan $M^* \in \mathbb{R}^{k\times n}$ to form linear combinations of patch embeddings $\tilde{Z} \in \mathbb{R}^{n\times d}$, obtaining the *refined object prototypes*

$$P' = M^*\tilde{Z} \in \mathbb{R}^{k\times d}. \tag{6.5}$$

Now, given the key embeddings $K_t \in \mathbb{R}^{(n+1)\times d}$ at another frame t, we track the objects by the *refined cross-attention*, similarly with (6.3):

$$T'_t := \mathrm{softmax}\left(P'\tilde{K}_t^\top/\sqrt{d}\right) \in \mathbb{R}^{k\times n}, \tag{6.6}$$

where $\tilde{K}_t \in \mathbb{R}^{n\times d}$. Indeed, Figure 6.4 confirms that each of the k resulting attention maps is associated with a spatially distinct object, thanks to the established correspondences.

In contrast to previous works that use SK in the context of self-supervised learning to force an equi-partitioning of images to cluster labels [Asano, 2020; Caron, 2020; Oquab, 2023], we rather use optimal transport to re-balance *spatial* correspondences to different objects.

Multi-object masking We use the cross-attention (6.6) to mask the input video clip for the student network, such that each masked clip can be considered as a *multi-object crop*. This crop plays a similar role with local crops in DINO [Caron, 2021], but it has arbitrary shape and tracks an object over video frames. In particular, given an input frame $\mathbf{X} \in \mathbb{R}^{h\times w\times c}$ with cross-attention matrix $T' \in \mathbb{R}^{k\times n}$ (6.6) and an object $i \in \{1,\dots,k\}$, we reshape the i-th row of T' as $h/p \times w/p$ and upsample to a $h \times w$ attention map to match the spatial resolution of $\mathbf{X}$, as shown in Figure 6.4. We repeat along the channel

dimension to form tensor $\mathbf{T}^i \in \mathbb{R}^{h \times w \times c}$ and we mask $\mathbf{X}$ as

$$\mathbf{X}^{o_i} := \mathbf{X} \odot \mathbf{T}^i, \tag{6.7}$$

where $\odot$ is the Hadamard product. Following DINO [Caron, 2021], given an input frame $\mathbf{X}_t$, we generate two standard resolution augmented *global views* $\mathbf{X}_t^a, \mathbf{X}_t^b$. We introduce a *multi-object loss*

L_t^{O} for frame t, applied to the [CLS] token between the teacher $f_{\theta'}$ output for one global view $\mathbf{X}_t^u$ and the student f_θ output for the masked version $\mathbf{X}_t^{v,o_i}$ of the other view $\mathbf{X}_t^v$ for $i \in \{1, \ldots, k\}$, where $u, v \in V = \{a, b\}$ and $u \neq v$:

$$L_t^{\mathrm{O}} := \sum_{u,v \in V} \mathbb{1}_{u \neq v} \sum_{i=1}^{k} f_{\theta'}(\mathbf{X}_t^u)^{[\mathrm{CLS}]} \log \left(f_\theta(\mathbf{X}_t^{v,o_i})^{[\mathrm{CLS}]} \right). \tag{6.8}$$

Following DINO [Caron, 2021] and iBOT [Zhou, 2022a], we apply the *multi-crop* strategy [Caron, 2020]. In particular, we generate m *local crops* $\mathbf{X}_t^{\ell_i}$ of smaller resolution for $i \in \{1, \ldots, m\}$. The *local loss* L_t^{LC} for frame t is applied to the [CLS] token between the teacher $f_{\theta'}$ output for a global view $\mathbf{X}_t^u$ and the student f_θ output for the local crop $\mathbf{X}_t^{\ell_i}$ for $i \in \{1, \ldots, m\}$:

$$L_t^{\mathrm{LC}} := \sum_{v \in V} \sum_{i=1}^{m} f_{\theta'}(\mathbf{X}_t^v)^{[\mathrm{CLS}]} \log \left(f_\theta(\mathbf{X}_t^{\ell_i})^{[\mathrm{CLS}]} \right) \tag{6.9}$$

The overall loss L is the sum of the multi-object loss L_t^{O} (6.8) and the local loss L_t^{LC} (6.9), averaged over all T frames:

$$L := \frac{1}{T} \sum_{t=1}^{T} (L_t^{\mathrm{O}} + L_t^{\mathrm{LC}}). \tag{6.10}$$

6.5 Experiments

6.5.1 Tasks and methods

We perform self-supervised pretraining on a single WT tour video in Venice (referred to as $\mathrm{WT}_{\mathrm{Venice}}$) or all 10 WT videos (referred to as $\mathrm{WT}_{\mathrm{all}}$) and compare with other image and video datasets. To evaluate the quality of the learned representations, we use frozen features for classification, unsupervised object discovery and video object segmentation. We fine-tune for semantic segmentation, object detection and object tracking. We compare

DoRA with SoTA SSL methods [Costa, 2022] using our settings. We provide more details in individual sections per task.

6.5.2 Implementation details

Code will be published as open-source code. We use ViT-S/16 [Dosovitskiy, 2021] as the backbone in all our experiments. For each mini-batch, we randomly sample clips from the video, consisting of $T = 8$ frames temporally separated by 1 second *i.e.* we sample one frame every 30. Objects discovered in the first frame are tracked over the following 7 frames. Since each frame contains several different objects, applying the standard multi-crop augmentation [Caron, 2020] to the entire frame would result in crops with very different visual content or *noisy* positive pairs. Instead, we apply multi-crop to a 300×300 crop that we first take from the frame. Following DINO [Caron, 2021], we obtain two *global crops* and six *local crops*. Masking (6.7) is applied to the global crops seen by the student for the multi-object loss (6.8), while local crops are seen directly by the student for the local loss (6.9). We train for 100 epochs by default.

Objects are discovered using attention heads, where the total number of heads is in ViT-S/16 is limited to $h = 6$. For the purpose of the ablation of the number k of objects for $k > h$ in Table 6.7b, we modify the MSA block in the final layer, resulting in configurations of 16 and 32 heads. Consequently, we can identify and track up to 16 and 32 objects within the video clip. To accomplish this, we decompose the query and key embeddings of dimension $d = 768$ into 16 and 32 subvectors, resulting in new feature dimensions of 24 and 12 respectively, as opposed to 64 for 6 heads. In Table 6.7b, we observe that tracking 16 or 32 objects results in overall poor performance possibly due to the small feature dimension, which encodes poor representations.

6.5.3 Hyperparameters

ImageNet-1k: Linear probing and k-NN We pretrain DoRA in a self-supervised setting with ViT-S/16 using DINO for 100 and 300 epochs. We use two global and six local crops for each clip and train on 8 A100 GPUs with a global batch size of $16 \times 8 = 128$. We use LARS [**<empty citation>**] with a learning rate of 5×10^{-4}, minimum learning rate of 1×10^{-6}, global crop scale of $[0.4, 1.0]$ and local crop scale $[0.05, 0.4]$.

For linear probing, we follow [Caron, 2021] and use the frozen features of the transformer backbone to train a linear classifier in a supervised setting. We use global batch

size of 1024 on the training set and evaluate on the validation set of ImageNet-1k. We use top-1 accuracy (%) as our evaluation metric. For k-NN, we freeze the backbone and extract features of training images, then use a k-nearest neighbour classifier with $k = 20$.

Pascal-VOC 2012: Object discovery We use the validation set of Pascal VOC 2012 [Everingham,], which comprises a total of 1449 images. Following LOST [Siméoni, 2021], we use the averaged self-attention map, extracted from the final layer of a our pre-trained ViT-S/16, to retain 80% of the mass. We use the Jaccard similarity J measured as overlap between predicted mask P and the ground truth mask G as $J(P, G) = \frac{G \cap P}{G \cup P}$. We also use CorLoc, which measures the number of correct predicted boxes, where a predicted box is said to be correct if its IoU ≥ 0.5.

ADE20k: Semantic segmentation We evaluate DoRA on ADE20k [Zhou, 2017] for semantic segmentation. The dataset includes 20,000 images in the training set and 2,000 images in the validation set. We use UperNet [Xiao, 2018] as the segmentation model and use DoRA pretrained on WT to initialize the backbone. Following the experimental settings in iBOT [Zhou, 2022a], we use AdamW [Loshchilov, 2019a] with an initial learning rate of 6×10^{-5}, weight decay of 1×10^{-2}, and linear warmup of 1,500 iterations. We fine-tune for 160,000 iterations with a batch size of 4.

MS-COCO: Object detection We evaluate DoRA for object detection and instance segmentation on MS-COCO. We use Cascade Mask R-CNN [Cai, 2019], which produces bounding boxes and instance masks simultaneously on the COCO dataset. We use a multi-scale training strategy, where we resize images to have a shorter side ranging between 480 and 800, ensuring that the longer side does not exceed 1,333 pixels. The learning rate is 1×10^{-4} and the weight decay is 0.05. During training, we fine-tune the entire network using a $1\times$ schedule, which involves 12 epochs with learning rate reductions by a factor of 10 at epochs 9 and 11. We explore different layer decay rates, specifically $0.65, 0.75, 0.8, 0.9$, with a rate of 1.0 indicating no decay.

To generate hierarchical feature maps, we utilize the features produced by layers 4, 6, 8, and 12 of our network and apply two deconvolutions for layer 4, one deconvolution for layer 6, identity mapping for layer 8, and max-pooling for layer 12. These post-processing steps enable the creation of hierarchical feature representations. It is important to note that we do not employ multi-scale testing in our experiments.

DAVIS-2017: Video object segmentation We assess the performance of DoRA for video object segmentation on DAVIS 2017 dataset [Pont-Tuset, 2017], which involves segmenting between 2 to 4 objects within the video frames. We follow DINO [Caron, 2021] and evaluate on video frames with a resolution of 480p. We apply label propagation on the attention map from our pretrained model and use mean region-based similarity $\mathcal{J}_m$ and mean contour-based accuracy $\mathcal{F}_m$ as our evaluation metrics.

GOT-10k: Object tracking We evaluate the object-tracking performance of DoRA on the GOT-10k dataset [Huang, 2019]. This is a large-scale benchmark for object tracking that contains 563 categories of common moving objects. The training set contains around 10,000 videos and the test set contains 180 videos. Another challenging aspect of this dataset is that the object classes in the training and test set are non-overlapping. We use the SeqTrack [Chen, 2023] codebase to evaluate the performance of different methods on this dataset. In particular, we initialize the encoder weights of SeqTrack with the self-supervised weights and keep them frozen during training. While training, we only update the parameters of the lightweight decoder which consists of 2 transformer blocks. We use all the default hyperparameters. We report mean average overlap (mAO) and success rate (SR) at different thresholds. The mAO measures the class-balanced average overlap between the ground truth and predicted bounding boxes whereas SR indicates the percentage of accurately tracked ground truth bounding boxes where the overlap crosses a certain threshold.

6.5.4 Comparison with State-of-the-art

Dense scene understanding Table 6.3(a) shows *semantic segmentation* by fine-tuning on ADE20k [Zhou, 2017] using UperNet [Xiao, 2018]. DoRA outperforms DINO by 3% mIoU, and 1.8% Acc_m. It is interesting to note that DORA pretrained on 200k frames of a *single* WTours video outperforms DINO pretrained on 1.3M images of ImageNet-1k

by 1.5% mIoU. A more comparable setting is DoRA pretrained on 1.5M frames of WT_{all}, which outperforms DINO pretrained on ImageNet by 3% mIoU.

Table 6.3(b) shows *object detection* and *instance segmentation* by fine-tuning on MS-COCO [Lin, 2014] using Cascade RCNN [Cai, 2019]. DoRA outperforms DINO by 2.4% mAP and 2.6% mIoU. DoRA pretrained on WT_{all} outperforms DINO pretrained on ImageNet by 0.8% mIoU and 1.2% mAP. This shows that pretraining on WTours videos significantly improves the generality of DoRA to dense prediction tasks, requiring only

Method	epochs	Pretrain	(a) Semantic seg.				(b) Object det.		(c) Instance seg.	
			mIoU	Gain	Acc_m	Gain	mAP	Gain	mIoU	Gain
ViT-S/16	100	None	25.1		33.3		28.6		24.3	
iBOT [Zhou, 2022a]	100	$\text{WT}_{\text{Venice}}$	33.9		43.3		37.6		33.0	
AttMask [Kakogeorgiou, 2022]	100	$\text{WT}_{\text{Venice}}$	33.6		42.7		36.5		32.5	
VITO [Parthasarathy, 2023]	300	VideoNet	39.4		–		44.0		–	
DINO [Caron, 2021]	100	IN-1k	33.9		44.3		39.9		35.1	
DoRA (ours)	100	WT_{all}	36.9		48.0		40.7		36.3	
DINO [Caron, 2021]	100	$\text{WT}_{\text{Venice}}$	32.4		43.7		37.1		32.1	
DoRA (ours)	100	$\text{WT}_{\text{Venice}}$	**35.4**	+3.0	**45.5**	+1.8	**39.5**	+2.4	**34.7**	+2.6

Table 6.3 – *Semantic segmentation, object detection and instance segmentation.* ViT-S/16 pretrained, then fine-tuned. $\text{WT}_{\text{Venice}}$: Walking Tours (ours), single video of *Venice*; WT_{all}: all videos. IN-1k: ImageNet-1k. (a) Semantic segmentation: fine-tuning on ADE20k using UperNet. mIoU: mean IoU; Acc_m: mean-class accuracy. (b) Object detection and (c) Instance segmentation: fine-tuning on MS-COCO using Cascade RCNN. mAP: mean average precision; mIoU: mean IoU.

one tenth of the total images.

Video understanding Table 6.4(a) shows video object segmentation by using frozen features on DAVIS-2017 [Pont-Tuset, 2017], which assesses the ability to segment an object over its dynamic temporal changes. DoRA captures detailed temporal deformations and outperforms baseline DINO by 3.4% $\mathcal{J}_m$ and 4.2% $\mathcal{F}_m$. Using only a *single* video for pretraining, DoRA achieves almost the same performance of DINO pretrained on ImageNet (56.4% *vs.* 57.4% $\mathcal{J}_m$). Table 6.4(b) shows multi-object tracking by fine-tuning on GOT-10k [Huang, 2021] using SeqTrack [Chen, 2023]. GOT-10k assesses the ability to track extremely fast moving objects, objects with illumination variation and low resolution. DoRA achieves significant gains between 4-6% over DINO.

Image classification and unsupervised object discovery We pretrain DoRA on WTours and then we keep it frozen on the downstream task, indicating the quality of the pretrained features. Table 6.5(a) shows *image classification* on ImageNet-1k, measuring accuracy for linear probing and k-nearest neighbor. Table 6.5(b) shows *unsupervised object discovery* on Pascal-VOC 2012,

using attention maps as segmentation masks to measure Jaccard similarity and CorLoc.

On both tasks, non-contrastive methods (DINO, iBOT, VICReg) outperform contrastive methods (SimCLR, SwAV), when pretrained on a single WT video. Importantly,

Method	epochs	Pretrain	(a) Video object segmentation						(b) Object tracking					
			$(\mathcal{J}\&\mathcal{F})_m$	Gain	$\mathcal{J}_m$	Gain	$\mathcal{F}_m$	Gain	mAO	Gain	$SR_{0.5}$	Gain	$SR_{0.75}$	Gain
ViT [Dosovitskiy, 2021]	100	None	26.9		25.4		28.3		23.1		19.0		3.4	
iBOT [Zhou, 2022a]	100	WT_{Venice}	57.4		56.7		58.0		41.5		47.5		16.6	
DINO [Caron, 2021]	100	IN-1k	59.4		57.4		61.4		46.4		54.3		24.1	
DoRA (ours)	100	WT_{all}	57.6		55.1		60.2		45.9		53.4		23.7	
DINO [Caron, 2021]	100	WT_{Venice}	54.6		53.0		56.2		37.4		41.4		13.4	
DoRA (ours)	100	WT_{Venice}	**58.4**	+3.8	**56.4**	+3.4	**60.4**	+4.2	**41.4**	+4.0	47.2	+5.8	18.2	+4.8

Table 6.4 – *Video object segmentation and object tracking.* ViT-S/16 pretrained, then frozen or fine-tuned. WT_{Venice}: Walking Tours (ours), single video from *Venice*; WT_{all}: all videos. IN-1k: ImageNet-1k. (a) Video object segmentation: frozen features on DAVIS-2017. $\mathcal{J}_m$: mean region similarity; $\mathcal{F}_m$: mean contour-based accuracy. (b) Multi-object tracking: fine-tuning on GOT-10k. mAO: mean average overlap; SR: success rate, threshold 50% and 75%.

Method	epochs	Pretrain	#Frames	(a) Classification				(b) Object discovery			
			(M)	LP	Gain	k-NN	Gain	Jacc.	Gain	CorLoc	Gain
SimCLR [Chen, 2020b]	100	WT_{Venice}	0.2	26.3		25.9		40.4		50.2	
SwAV [Caron, 2020]	100	WT_{Venice}	0.2	28.0		26.4		40.6		51.4	
iBOT [Zhou, 2022a]	100	WT_{Venice}	0.2	36.8		32.8		43.0		53.1	
AttMask [Kakogeorgiou, 2022]	100	WT_{Venice}	0.2	35.8		31.9		43.5		54.5	
VicReg [Bardes, 2021]	100	WT_{Venice}	0.2	36.5		30.1		42.7		52.1	
DINO [Caron, 2021]	100	WT_{Venice}	0.2	33.8		29.9		43.8		51.2	
DoRA (ours)	100	WT_{Venice}	0.2	**45.4**	+11.6	**33.8**	+3.9	**44.0**	+0.2	**56.2**	+5.0
DINO [Caron, 2021]	100	WT_{all}	1.5	36.6		31.1		42.9		55.8	
DoRA (ours)	100	WT_{all}	1.5	**45.3**	+8.7	**35.7**	+4.6	**44.3**	+1.4	**57.1**	+1.3

Table 6.5 – *Image classification and object discovery.* ViT-S/16 pretrained, then frozen. WT_{Venice}: Walking Tours (ours), single video from *Venice*; WT_{all}: all videos. (a) Classification top-1 accuracy (%) on validation set of ImageNet-1k. LP: linear probing. (b) Unsupervised object discovery on validation set of Pascal-VOC 2012. Jacc.: Jaccard similarity; CorLoc: Correct Localization.

Method	Epochs	Classification		Object disc.		Semantic Seg.		Object Det.	
		LP	k-NN	Jacc	CorLoc	mIoU	Acc_m	mAP	mIoU
DINO [Caron, 2021]	100	71.4	69.0	44.5	59.6	33.9	44.3	37.1	32.1
iBOT [Zhou, 2022a]	100	72.1	69.4	44.5	59.7	35.2	45.1	38.9	34.4
DoRA* (ours)	60	71.9	69.4	44.4	60.0	35.4	44.9	39.3	34.9
DoRA* (ours)	100	**72.2**	**69.6**	**44.8**	**60.2**	**35.8**	**45.1**	**39.9**	**35.1**

Table 6.6 – *Pretraining on ImageNet-1k.* ViT-S/16 pretrained, then frozen (classification and object discovery, same settings as Table 6.5) or fine-tuned (semantic segmentation and object detection, same settings as Table 6.3). DoRA*: DoRA without tracking; when pretrained for 60 epochs, it has the same training time as DINO and iBOT.

non-contrastive methods are also more efficient to train, since no negative pairs are used. Also on both tasks, DoRA outperforms DINO by a large margin, *e.g.* 11.6% LP and 3.9% k-NN on classification, when trained on a single WT video. Comparing DoRA on $\text{WT}_{\text{Venice}}$ with the WT_{all} dataset, the improvement brought by the full dataset is small when using DoRA, although it is 10 times larger.

Pretraining on ImageNet-1k We pretrain DoRA on ImageNet-1k and compare with SoTA methods on multiple tasks. Unlike videos, we discover objects but do not track them. Instead, images in a mini-batch are processed independently. Given an input image $\mathbf{X}$, we obtain refined object prototypes as usual (6.5), but the refined cross-attention (6.6) is with $\tilde{K}_t$ replaced by $\tilde{K}$ of the same image $\mathbf{X}$. The same image $\mathbf{X}$ is masked for the student (6.7). The loss is given again by (6.8) and (6.9) with $\mathbf{X}_t$ replaced by $\mathbf{X}$, averaged over the mini-batch. We refer to this version as *DoRA without tracking* or DoRA*.

DINO [Caron, 2021] and iBOT [Zhou, 2022a] use only one global crop for the student, while DoRA uses k object crops. To compensate, we perform an experiment where we pretrain DoRA* for 60 epochs and the competitors for 100, thus all methods having the same training time.

From Table 6.9, we observe that DoRA outperforms state-of-the-art self-supervised learning (SSL) methods like DINO and iBOT on image downstream tasks. This demonstrates that the multi-object loss not only enhances performance when pretrained on WTours videos but also achieves superior results when pretrained on ImageNet-1k images.

Method	Pretrain	#Frames (M)	LP	CorLoc
DINO	$\text{Movie}_{\text{rom}}$	0.19	34.9	51.5
DoRA	$\text{Movie}_{\text{rom}}$	0.19	**35.3**	**51.6**
DINO	K-400*	0.2	40.7	52.4
DoRA	K-400*	0.2	**43.0**	**55.2**
DINO	EK*	0.2	38.6	53.5
DoRA	EK*	0.2	**41.8**	**56.0**
DINO	$\text{WT}_{\text{Venice}}$	0.2	33.8	51.2
DoRA	$\text{WT}_{\text{Venice}}$	0.2	**44.5**	**56.2**

(a) *Video datasets*

Method	k	LP	CorLoc
DINO	✗	33.8	51.2
DoRA	1	39.9	53.9
DoRA	2	43.1	55.7
DoRA	3	**44.5**	**56.2**
DoRA	4	39.2	53.8
DoRA	5	36.7	50.3
DoRA	6	35.8	48.8
DoRA	16	28.3	48.5
DoRA	32	27.1	46.8

(b) *#Objects k on WT_{Venice}*

Method	SK	Mask	LP	CorLoc
DINO	✗	✗	33.8	51.2
DoRA	✗	Random	33.0	49.8
DoRA	✗	Object	42.5	55.3
DoRA	✓	Random	29.9	46.7
DoRA	✓	Object	**44.5**	**56.2**

(c) *SK and masking on WT_{Venice}*

Table 6.7 – *Effect of parameters.* ViT-S/16 pretrained, then frozen. (a) Different pretraining video dataset, (b) Number k of tracked objects. (c) Random or multi-object mask, without SK (6.3) and with SK (6.6). *: subset of videos with same total duration as a single WTours video. K-400: Kinetics-400, EK: Epic-Kitchens. LP: top-1 accuracy (%) of linear probing on the validation set of ImageNet-1k. CorLoc: correct localization on validation set of Pascal-VOC 2012.

6.5.5 Ablations

We examine the effect of using different pretraining video dataset and different options and parameters for DoRA, measuring performance of classification on ImageNet-1k [Deng, 2009] by linear probing (LP) accuracy and unsupervised object discovery on Pascal-VOC 2012 [Everingham,] by correct localization (CorLoc) [Siméoni, 2021].

Pretraining video dataset We study the impact of pretraining on diverse video datasets, encompassing *object-centric* videos such as Kinetics-400 (K-400) [Kay, 2017], *egocentric* videos like Epic-Kitchens (EK) [Damen, 2022] and a single movie, $\text{Movie}_{\text{rom}}$ [Central,]. To maintain uniformity in terms of the number of frames, we curate a subset of videos from K-400 and EK, such that their total duration is the same as a single WT video. In Table 6.7a, we observe that although K-400

is *object-centric*, pretraining on WTours videos yields superior performance on ImageNet and Pascal-VOC 2012. Pretraining on a single movie yields is inferior to both WTours and K-400 by a large margin. This is possibly due to the presence of cuts, which is shown in Table 6.8b.

Number of tracked objects We study the impact of the number k of objects. Objects are discovered using attention heads, where the total number of heads is in ViT-S/16 is $h = 6$. For $k > h$, we modify the MSA block as described in subsection 6.5.2. In Table 6.7b,

Video	LP	CorLoc
Amsterdam	**45.4**	54.5
Bangkok	42.1	54.3
Chiang Mai	44.9	55.5
Istanbul	44.5	54.6
Kuala Lampur	43.9	54.1
Singapore	42.7	54.7
Stockholm	44.1	54.7
Venice	44.5	**56.2**
Wildlife	44.0	54.9
Zurich	44.9	54.4
Mean	44.1	54.8

(a) *WT videos*

Method	PT	LP	CorLoc
DINO	WT	33.8	51.2
DoRA	Movie	35.3	51.6
DoRA	Movie†	39.8	54.8
DoRA	WT	**44.5**	**56.2**

(b) *Cuts*

Table 6.8 – *Effect of pretraining video and cuts.* ViT-S/16 pretrained, then frozen. (a) Different WTours video, using DoRA. (b) Effect of cuts. *: subset of videos with same total duration as a single WTours video. †: sampling without cuts. LP: top-1 accuracy (%) of linear probing on the validation set of ImageNet-1k. CorLoc: correct localization on validation set of Pascal-VOC 2012.

we observe that $k = 3$ works best. We hypothesize that this is a compromise between the number of objects that can be tracked and the multi-object loss (6.8) attempting to match small objects with the global crop.

Choice of masking and Sinkhorn-Knopp We explore the effect of using a multi-object mask (6.7) *vs.* random block-wise [Zhou, 2022a] and the effect of improving object-patch correspondence through SK in refined cross-attention (6.6) *vs.* (6.3). In Table 6.7c, we observe that a multi-object mask leads to a remarkable performance improvement even in the absence of SK. In fact, random block-wise mask undermines object-patch correspondence, making the effect of SK negative. By contrast, SK improves performance in the presence of multi-object mask.

Pretraining WT video We study the effect of pretraining on different videos of WTours. In Table 6.8a, we observe that the effect is minimal on both image classification and unsupervised object discovery. Notably, the fluctuation in illumination conditions within the *Bangkok* video influences the performance on image classification. It is also interesting to note that, while pretraining on *Amsterdam* is best on image classification, pretraining on *Venice* is best on object discovery. This could be due to the large overlap of objects in these videos with respect to the downstream datasets. However, the consistency of our method across diverse videos indicates that DoRA is robust to variations in

Method	Epochs	Classification		Object disc.		Semantic Seg.		Object Det.	
		LP	k-NN	Jacc	CorLoc	mIoU	Acc_m	mAP	mIoU
DINO [Caron, 2021]	100	71.4	69.0	44.5	59.6	33.9	44.3	37.1	32.1
iBOT [Zhou, 2022a]	100	72.1	69.4	44.5	59.7	35.2	45.1	38.9	34.4
DoRA* (ours)	60	71.9	69.4	44.4	60.0	35.4	44.9	39.3	34.9
DoRA* (ours)	100	**72.2**	**69.6**	**44.8**	**60.2**	**35.8**	**45.1**	**39.9**	**35.1**

Table 6.9 – *Pretraining on ImageNet-1k.* ViT-S/16 pretrained, then frozen (classification and object discovery, same settings as Table 6.5) or fine-tuned (semantic segmentation and object detection, same settings as Table 6.3). DoRA*: DoRA without tracking; when pretrained for 60 epochs, it has the same training time as DINO and iBOT.

scenes, number of objects and lighting conditions.

Presence of cuts We now analyse the effect of cuts in representation learning. Cuts are defined as instant transitions from one shot to the next, which is frequent in movies. In action movies, a single shot lasts around 4 seconds, while in romance movies, around 12 seconds on average [2]. To understand the effect of cuts, we compare pretraining on WTours videos and a romance movie. We use PySceneDetect [Castellano,] to extract the cut timestamps in the movie and we pretrain DoRA by sampling clips that do not intersect cuts; cuts naturally do not exist in WT videos. In Table 6.8b, we observe that the performance improves significantly in the absence of cuts, as tracking in DoRA will fail across a cut.

6.6 More visualizations

Figures 6.5 and 6.6 show example attention maps obtained using SK on different clips. These figures show that SK (6.6) leads to attention maps that exhibit spatial locality and are well aligned with objects in the scene. Remarkably, the masks seem to be even robust to occlusions, as shown in the sequence with a bicycle moving behind traffic lights.

6.7 Conclusion

We have introduced a dataset of 10 walking tour videos – first-person videos taken by people touring a city, with no cuts, high resolution and that are hours long. We show that

2. https://stephenfollows.com/many-shots-average-movie/

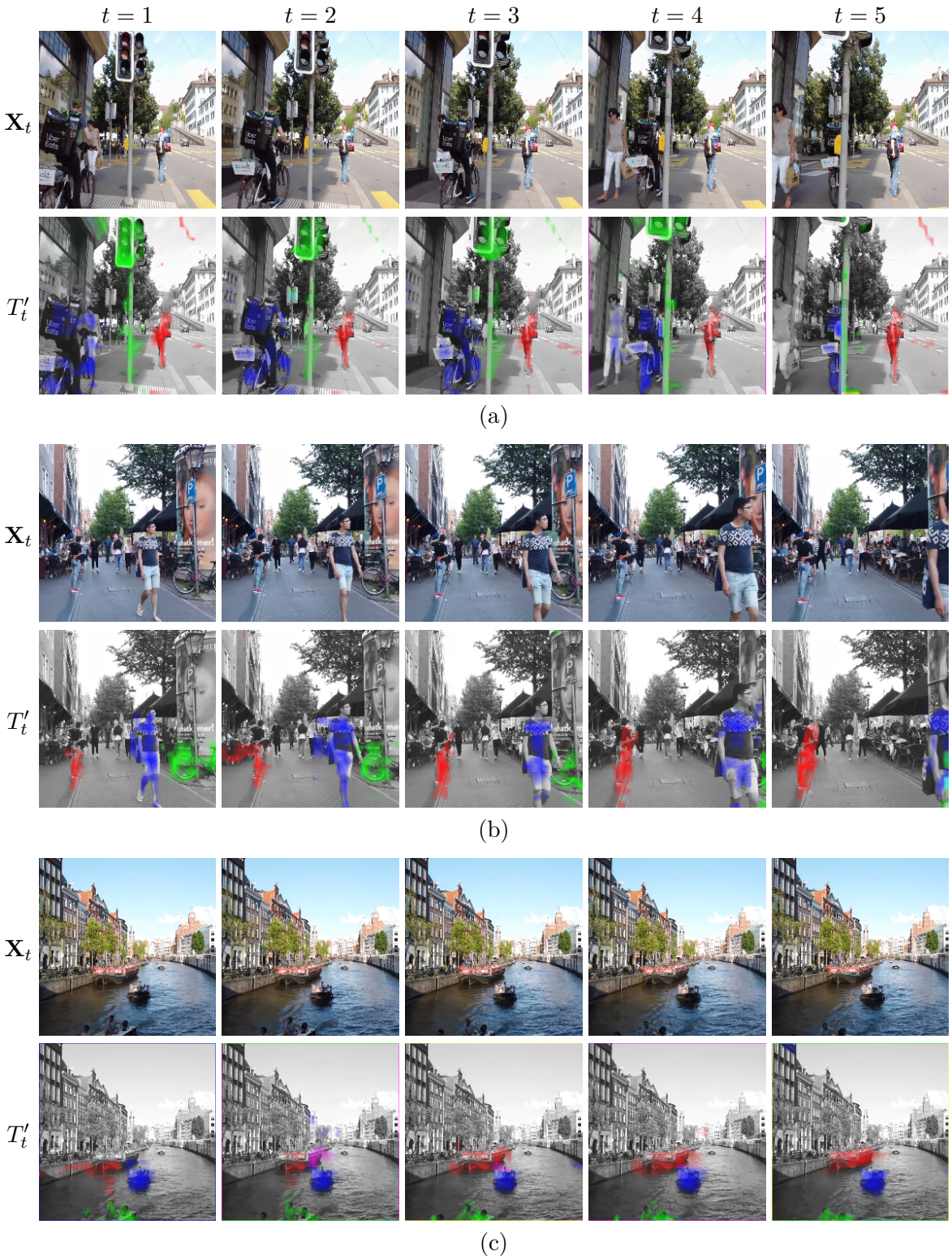


Figure 6.5 – For each input frame $\mathbf{X}_t$ of a video clip, refined cross-attention map $T'_t \in \mathbb{R}^{k \times n}$ (6.6), using Sinkhorn-Knopp. For each object, one row of T'_t is reshaped as $h/p \times w/p$ and upsampled to an $h \times w$ *attention map* overlaid on the input frame for $k = 3$ objects encoded in blue, red and green channel. T'_t yields three well separated objects shown in blue, red and green.

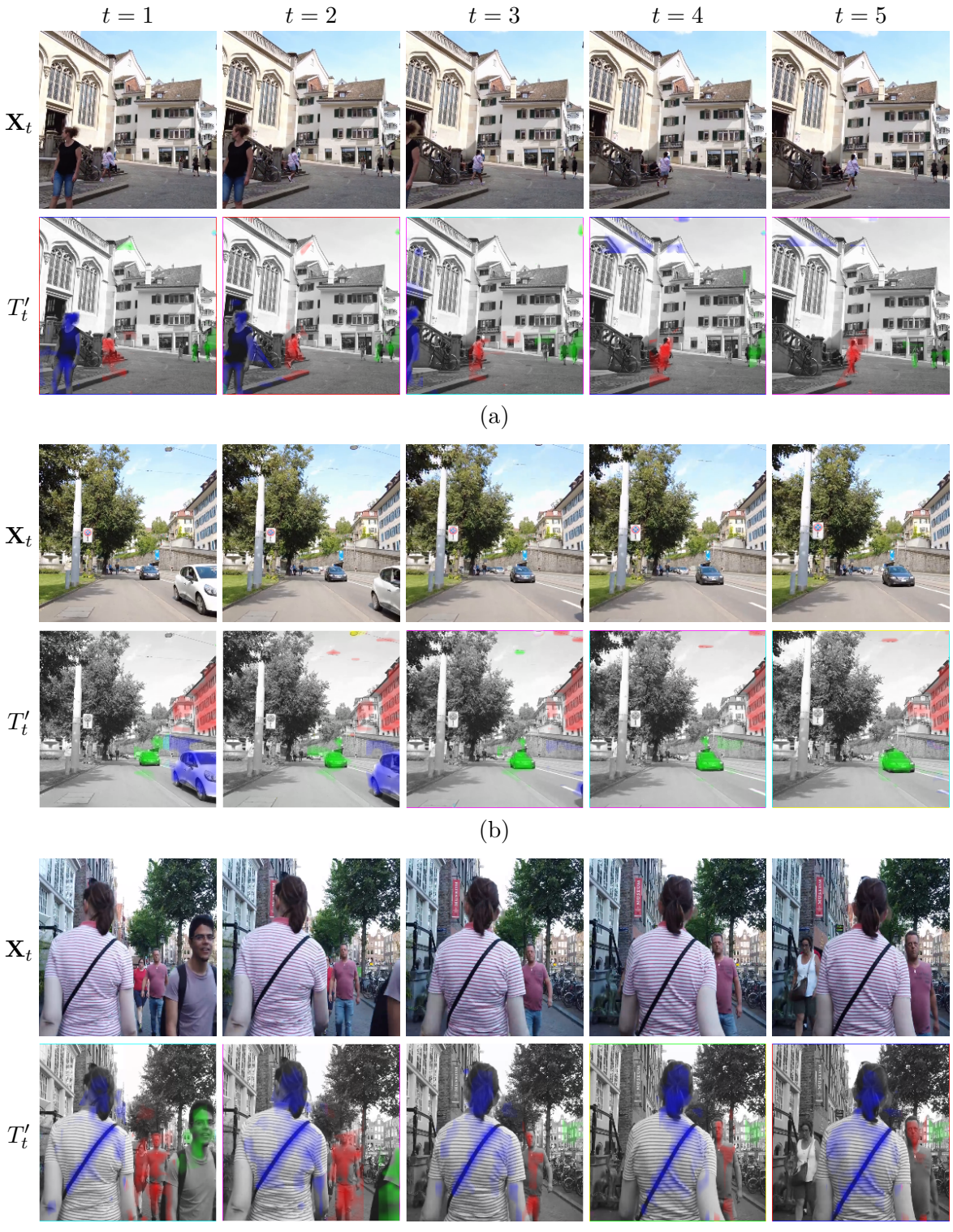


Figure 6.6 – For each input frame $\mathbf{X}_t$ of a video clip, refined cross-attention map $T'_t \in \mathbb{R}^{k \times n}$ (6.6), using Sinkhorn-Knopp. For each object, one row of T'_t is reshaped as $h/p \times w/p$ and upsampled to an $h \times w$ *attention map* overlaid on the input frame for $k = 3$ objects encoded in blue, red and green channel. T'_1 yields three well separated objects shown in blue, red and green.

learning from clips taken from these videos is surprisingly powerful: with an appropriately tailored self-supervised learning method for videos, we obtain representations that rival those obtained on ImageNet when transferring to popular downstream image and video tasks. This differs from previous state-of-the-art approaches to learning image encoders from video, which also obtain such results but require large video datasets, following closely the ImageNet blueprint.

Our proposed learning method DoRA is inspired by DINO, generalizing it to video by incorporating implicit multi-object tracking across video clips. We observe that the method leads to interesting emergent attention masks within the transformer model, that seem to latch on to particular objects, even through occlusions. This makes it uniquely suited to our newly introduced dataset.

Chapter 7

Conclusion

We recap the contributions outlined in this manuscript before providing an overview of key open problems and challenges in the field of representation learning.

7.1 Conclusions

AlignMixup In chapter 1, we discussed that one of the main challenges in interpolation based data augmentation methods like mixup is the potential lack of semantic coherence in the augmented samples. Mixup can lead to samples that are visually plausible but semantically inconsistent or unrealistic, potentially introducing noise and confounding factors during the training process. We have shown in chapter 3, interpolation of features by traversing along the manifold of representations from deeper layers of the network more likely results in realistic examples. Specifically, we have observed that alignment of features is a critical factor in achieving the desired improvements from the interpolation-based data augmentation approach. As compared to observations in Manifold Mixup [Verma, 2019], we have shown that mixup of a combination of input and latent representations of feature tensors from deeper layers is a simple and very effective pairwise data augmentation method. Our most interesting observations are as follows:

1. The idea of deformation as a natural way of interpolating images, where one image may continuously deform into another, aligns with the intuition that images can be smoothly transformed into each other, rather than just being linearly combined.
2. A key challenge identified is to make progress in the direction of a fully learned interpolation approach without compromising the speed and simplicity of the method, which would affect its wide applicability.

Metrix In chapter 4, we introduce a direct extension of the mixup technique from classification to metric learning. The key insight is that metric learning can be viewed as a binary classification problem of pairs of examples into "positive" and "negative" classes.

This observation allows the application of the mixup principle to metric learning, where the interpolation of labels affects the relative weighting of positives and negatives. The proposed approach is generic and can be applied to a wide range of loss functions that separate positives from negatives per anchor and involve component functions that are additive over examples. This is particularly beneficial for loss functions that require less mining, as the mixup-based approach provides a principled way of handling the positive and negative examples. Interestingly, we observed that:

1. We show that Metrix is completely agnostic with respect to the mixup method, opening the way to using more advanced mixup methods for metric learning. This allows for further improvements by incorporating more complex mixup techniques.
2. Interestingly, the multi-similarity loss function, which was not the state of the art without mixup, becomes the state of the art when using Metrix.
3. Because metric learning is about generalizing to unseen classes and distributions, our work may have applications to other such problems, including transfer learning, few-shot and continual learning.

MultiMix The key takeaway from chapter 5 is that a simple yet effective approach, MultiMix, can outperform more complex interpolation methods in the input space or intermediate features. The three critical elements of MultiMix - increasing the number of generated mixed examples, increasing the number of examples being interpolated, and performing interpolation in the embedding space - complement each other to provide state-of-the-art performance. By better approximating the expected risk integral through dense interpolation and leveraging the learned manifold in the embedding space, MultiMix demonstrates the following interesting properties:

1. Increasing the number of generated mixed examples per mini-batch provides a better approximation of the expected risk integral, which is typically estimated using a finite sum over the training data. By sampling more augmented examples from the vicinity of each training point, MultiMix can more accurately capture the underlying data distribution.
2. Interpolating in the embedding space, rather than the input space, allows MultiMix to leverage the learned manifold structure. This suggests that the learned manifold is a good proxy for the true, unknown data manifold, and that points nearby in the embedding space are semantically similar.

3. The key insight is that a relatively simple approach, with linear interpolation in the embedding space, can outperform more complex and sophisticated interpolation methods in the input space or intermediate features. This highlights the importance of carefully designing data augmentation techniques, rather than solely relying on increased model complexity.

DoRA Finally, in chapter 6, we focus towards capturing natural variations of objects, which offer rich semantic information present in real-world scenarios, using video data. We introduced a novel dataset of 10 walking tour videos, which are first-person videos taken by people touring a city, with no cuts, high resolution, and hours-long duration. We demonstrate that learning from clips extracted from these videos is remarkably powerful: by employing an appropriately tailored self-supervised learning method for videos, we obtain representations that rival those obtained on ImageNet when transferring to popular downstream image and video tasks. This contrasts with previous state-of-the-art approaches to learning image encoders from video, which also achieve such results but require large video datasets, closely following the ImageNet blueprint. Our proposed learning method, DoRA, is inspired by DINO and generalizes it to video by incorporating implicit multi-object tracking across video clips. In DoRA, we observed that:

1. It leads to the emergence of attention masks that can latch onto specific objects even through occlusions. This makes DoRA particularly well-suited for the walking tour video dataset, where the ability to track objects across clips is crucial.
2. Unlike previous state-of-the-art methods that rely on large video datasets and closely follow the ImageNet based downstream tasks, DoRA shows that it can achieve comparable results without the need for such extensive video data or large scale image datasets.
3. The dataset of walking tour videos, with their continuous, high-resolution, and lengthy nature, provides a rich source of information for self-supervised learning. The representations obtained from this dataset rival those learned on the large-scale ImageNet dataset, highlighting the potential of this real-world egocentric dataset.

7.2 What comes next?

Over the past couple of years, we have witnessed tremendous progress in the field of representation learning, particularly with the emergence of foundational models in VLMs,

LLMs, and diffusion models. However, as I reflect on the current state of the field, my personal and honest opinion is that I don't believe the challenges have been fully solved. Much of the research so far has been conducted in highly controlled settings, which are still far removed from real-world applications. In my view, there are still many open questions and remaining challenges that need to be addressed in representation learning.

Learning from Videos vs Large-scale text data Building on my previous work on DoRA in chapter 6, and drawing inspiration from Prof. Yann LeCun's recent talks on V-JEPA [Bardes, 2024], I've been struck by the stark contrast between the capabilities of large language models (LLMs) and the learning abilities of young children.

As Prof. LeCun pointed out, the largest LLMs have been trained on around 10 trillion tokens, which is equivalent to about 1 Exabyte of data. In comparison, a 4-year-old child has access to an estimated 1 quintillion bytes of data through their visual and auditory experiences over the course of 16,000 waking hours. That is 50 times more data than even the biggest LLMs have been exposed to. Moreover, Prof. LeCun noted that it would take a human 170,000 years to read all the high-quality text available on the internet, which is the primary training data for LLMs. This highlights the inherent limitations of text as a modality for learning about the world, as it is simply too low-bandwidth and scarce compared to the rich experiences that humans have access to from a young age.

In contrast, video data is more redundant and provides the kind of rich, multimodal information that can be leveraged for effective self-supervised learning. This redundancy is precisely what's needed for self-supervised learning to work well. So, while the progress in representation learning has been impressive, I believe there is still a lot of work to be done to truly capture the depth and breadth of human learning. By exploring new modalities, such as video, and drawing inspiration from the way children learn, I'm confident we can make further advancements in this field.

Synthetic data *vs.* Real-world data The use of synthetic data, such as from video games like GTA-5, can be a powerful complement to real-world data for self-supervised learning models as illustrated in Figure 7.1. Several works [Richter, 2016; Martinez, 2017] have demonstrated the potential of leveraging the vast amount of information available in gaming environments to pre-train models. This synthetic environments can generate an essentially unlimited amount of diverse data at a fraction of the cost of collecting real-world data. This allows self-supervised models to be trained on massive datasets. The

environments can also be programmed to include rare events such as sudden showers of rain or snow, dangerous scenarios due to natural calamities such as landslide, floods etc. and diverse conditions that may be difficult to capture in real-world data collection.

Frames from GTA-5

Frames from Walking Tour videos

Figure 7.1 – *Gaming* vs. *Real world videos* We observe a high similarity between real-world video frames and scenes from the video game GTA 5. Using these synthetic frames for self-supervised learning, can enable vision encoders to learn meaningful representations without needing large amounts of labeled data.

However, there may be a discrepancy between the statistical properties and visual fidelity of synthetic data compared to real-world data. Additionally, synthetic environments may struggle to capture the full complexity and subtleties of the real world, which could lead to overfitting or blind spots in the self-supervised model.

The most promising approach is to leverage both synthetic and real-world data for self-supervised pre-training and fine-tuning as shown in [Tian, 2024b; Tian, 2024a]. The synthetic data can provide the scale, diversity, and safety benefits to kickstart the self-supervised learning, while the real-world data can help bridge the domain gap and instill the model with real-world nuance.

Multi-modal information in real-world data The models developed in this manuscript operate on images or videos only. However, most real-world applications involve data from multiple modalities, such as videos that include audio, captions or hashtags, geolocation data, and other associated information.

This multimodal nature of real-world data presents an exciting opportunity to develop more robust and comprehensive representation learning systems. By ingesting and processing data from different modalities, these systems can learn richer and more generic representations that capture the inherent relationships and complementary information across the various data streams.

One promising approach in this direction is the development of vision-language models, which leverage both visual and textual information to learn powerful representations. These models, such as CLIP [Radford, 2021] and DALL-E [Ramesh, 2021], have demonstrated impressive capabilities in tasks like image classification, captioning, and even zero-shot learning, by tapping into the similarities between visual and linguistic data. Similarly, incorporating audio information alongside visual data can lead to even more comprehensive representations. Just as humans learn about the world through a combination of sight and sound, multimodal models that fuse visual and auditory cues can potentially capture a more holistic understanding of the environment and the objects within it.

By leveraging these diverse modalities, SSL algorithms can uncover deeper, more meaningful patterns in the data, leading to representations that are more transferable and applicable to a wider range of tasks and real-world scenarios. The redundancy and complementarity of the different data streams can serve as a powerful signal for the SSL models to learn robust and generalizable features. Overall, the shift towards multimodal representation learning is an exciting direction that holds great promise for advancing the state of the art in various applications, from computer vision and natural language processing to emodied AI and beyond. By embracing the richness of real-world data, we can develop AI systems that better mimic the way humans learn and understand the world around them.

BIBLIOGRAPHY

[Abhishek, 2022] Kumar Abhishek, Colin J Brown, and Ghassan Hamarneh. Multi-Sample ζ-mixup: Richer, More Realistic Synthetic Samples from a p-Series Interpolant. In: *arXiv preprint arXiv:2204.03323* (2022) (cit. on pp. 97, 107, 109, 111).

[Adams, 1987] Russell J Adams. An evaluation of color preference in early infancy. In: *Infant Behavior and Development* (1987) (cit. on p. 123).

[Agrawal, 2015] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In: *ICCV*. 2015 (cit. on p. 125).

[Ahn, 2019] Jiwoon Ahn, Sunghyun Cho, and Suha Kwak. Weakly Supervised Learning of Instance Segmentation with Inter-Pixel Relations. In: *CVPR*. 2019.

[AlBahar, 2019] Badour AlBahar and Jia-Bin Huang. Guided image-to-image translation with bi-directional feature transformation. In: *ICCV*. 2019.

[Allingham, 2021] James Urquhart Allingham, Florian Wenzel, Zelda E Mariet, Basil Mustafa, Joan Puigcerver, Neil Houlsby, et al. Sparse MoEs meet efficient ensembles. In: *arXiv preprint arXiv:2110.03360* (2021).

[Alvarez-Melis, 2018] David Alvarez-Melis and Tommi S Jaakkola. Gromov-Wasserstein alignment of word embedding spaces. In: *EMNLP*. 2018 (cit. on p. 53).

[Asano, 2020] Yuki M. Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In: *ICLR*. 2020 (cit. on p. 133).

[Bachman, 2019] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In: *NeurIPS* (2019) (cit. on p. 39).

[Bain, 2021] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In: *ICCV*. 2021 (cit. on pp. 126, 127).

[Bao, 2021] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. In: *arXiv preprint arXiv:2106.08254* (2021) (cit. on p. 43).

[Bardes, 2024] Adrien Bardes, Quentin Garrido, Jean Ponce, Michael Rabbat, Yann LeCun, Mahmoud Assran, et al. Revisiting Feature Prediction for Learning Visual Representations from Video. In: *arXiv preprint* (2024) (cit. on p. 150).

[Bardes, 2021] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In: *arXiv preprint arXiv:2105.04906* (2021) (cit. on p. 139).

[Bay, 2006] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In: *ECCV*. 2006 (cit. on p. 15).

[Beckham, 2019] Christopher Beckham, Sina Honari, Vikas Verma, Alex Lamb, Farnoosh Ghadiri, R Devon Hjelm, et al. On adversarial mixup resynthesis. In: *NIPS*. 2019 (cit. on pp. 49, 73, 99).

[Bell, 2015] Sean Bell and Kavita Bala. Learning visual similarity for product design with convolutional neural networks. In: *ACM Tansactions on Graphics (TOG)* (2015).

[Bello, 2019] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In: *ICCV*. 2019.

[Bengio, 2012] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In: *ICMLW on unsupervised and transfer learning*. 2012 (cit. on pp. 14, 15).

[Bengio, 2013] Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. Better mixing via deep representations. In: 2013 (cit. on pp. 22, 29, 47, 49, 181).

[Berthelot, 2018] David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In: *arXiv preprint arXiv:1807.07543* (2018) (cit. on pp. 49, 73, 99, 120).

[Bingham, 2008] Geoffrey P Bingham and Mats Lind. Large continuous perspective transformations are necessary and sufficient for accurate perception of metric shape. In: *Perception & Psychophysics* (2008) (cit. on p. 19).

[Bomba, 1983] Paul C Bomba and Einar R Siqueland. The nature and structure of infant form categories. In: *Journal of Experimental Child Psychology* (1983) (cit. on p. 123).

[Bommasani, 2021] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, et al. On the opportunities and risks of foundation models. In: *arXiv preprint arXiv:2108.07258* (2021) (cit. on p. 17).

[Boudiaf, 2020] Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, Pablo Piantanida, et al. A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses. In: *ECCV*. 2020.

[Bouthillier, 2015] Xavier Bouthillier, Kishore Konda, Pascal Vincent, and Roland Memisevic. Dropout as data augmentation. In: *arXiv preprint arXiv:1506.08700* (2015) (cit. on p. 30).

[Caesar, 2018] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. COCO-Stuff: Thing and stuff classes in context. In: *CVPR*. 2018.

[Cai, 2019] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: High quality object detection and instance segmentation. In: *IEEE TPAMI* (2019) (cit. on pp. 136, 137).

[Campos, 1978] Joseph J Campos, Susan Hiatt, Douglas Ramsay, Charlotte Henderson, and Marilyn Svejda. The emergence of fear on the visual cliff. In: *The development of affect* (1978) (cit. on p. 123).

[Cao, 2021] Jie Cao, Luanxuan Hou, Ming-Hsuan Yang, Ran He, and Zhenan Sun. ReMix: Towards Image-to-Image Translation with Limited Data. In: *CVPR*. 2021 (cit. on p. 60).

[Caron, 2018] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In: *ECCV*. 2018 (cit. on p. 71).

[Caron, 2020] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In: *NeurIPS*. 2020 (cit. on pp. 41, 129, 133–135, 139).

[Caron, 2021] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, et al. Emerging properties in self-supervised vision transformers. In: *ICCV*. 2021 (cit. on pp. 121, 125, 130, 131, 133–135, 137–140, 143).

[Carratino, 2022] Luigi Carratino, Moustapha Cissé, Rodolphe Jenatton, and Jean-Philippe Vert. On mixup regularization. In: *JMLR* (2022).

[Castellano,] Brandon Castellano. PySceneDetect. `https://github.com/Breakthrough/PySceneDetect` (cit. on pp. 127, 143).

[Central,] World Movie Central. The Night We Met. `https://www.youtube.com/watch?v=joIzqAueexA` (cit. on pp. 129, 141).

[Chadebec, 2022] Clément Chadebec, Elina Thibeau-Sutre, Ninon Burgos, and Stéphanie Allassonnière. Data augmentation in high dimensional low sample size setting using a geometry-based variational autoencoder. In: *IEEE TPAMI* (2022).

[Chakraborty, 2017] Prabuddha Chakraborty and Vinay P. Namboodiri. Learning to Estimate Pose by Watching Videos. In: *arXiv preprint arXiv:1704.04081* (2017) (cit. on p. 125).

[Chen, 2022] Jie-Neng Chen, Shuyang Sun, Ju He, Philip HS Torr, Alan Yuille, and Song Bai. TransMix: Attend to mix for vision transformers. In: *CVPR*. 2022 (cit. on pp. 96, 105, 107, 108).

[Chen, 2020a] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Gridmask data augmentation. In: *arXiv preprint arXiv:2001.04086* (2020) (cit. on p. 29).

[Chen, 2020b] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In: *ICML*. 2020 (cit. on pp. 40, 70, 125, 129, 139).

[Chen, 2017] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In: *CVPR*. 2017 (cit. on pp. 73, 98).

[Chen, 2023] Xin Chen, Houwen Peng, Dong Wang, Huchuan Lu, and Han Hu. SeqTrack: Sequence to Sequence Learning for Visual Object Tracking. In: *CVPR*. 2023 (cit. on pp. 137, 138).

[Chen, 2020c] Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, et al. PointMixup: Augmentation for Point Clouds. In: *ECCV* (2020) (cit. on p. 51).

[Choe, 2020] Junsuk Choe, Seong Joon Oh, Seungho Lee, Sanghyuk Chun, Zeynep Akata, and Hyunjung Shim. Evaluating weakly supervised object localization methods right. In: *CVPR*. 2020 (cit. on p. 66).

[Choe, 2019] Junsuk Choe and Hyunjung Shim. Attention-based dropout layer for weakly supervised object localization. In: *CVPR*. 2019 (cit. on pp. 30, 66).

[Choi, 2018] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: *CVPR*. 2018.

[Choi, 2020] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In: *CVPR*. 2020.

[Choy, 2016] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. Universal Correspondence Network. In: *NeurIPS*. 2016 (cit. on p. 51).

[Chuang, 2020] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. In: *NeurIPS*. 2020 (cit. on p. 36).

[Cinel, 2019] Caterina Cinel, Davide Valeriani, and Riccardo Poli. Neurotechnologies for human cognitive augmentation: current state of the art and future prospects. In: *Frontiers in human neuroscience* (2019) (cit. on p. 19).

[Costa, 2022] Victor Guilherme Turrisi da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. solo-learn: A Library of Self-supervised Methods for Visual Representation Learning. In: *JMLR* (2022) (cit. on p. 135).

[Croitoru, 2017] Ioana Croitoru, Simion-Vlad Bogolin, and Marius Leordeanu. Unsupervised learning from video to detect foreground objects in single images. In: *ICCV*. 2017 (cit. on p. 125).

[Csurka, 2004] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In: *Workshop on statistical learning in computer vision, ECCV*. Prague. 2004 (cit. on pp. 14, 15).

[Cubuk, 2018] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. AutoAugment: Learning augmentation policies from data. In: *arXiv preprint arXiv:1805.09501* (2018) (cit. on pp. 71, 105).

[Cubuk, 2019] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. AutoAugment: Learning augmentation strategies from data. In: *CVPR*. 2019 (cit. on p. 48).

[Cubuk, 2020] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. RandAugment: Practical automated data augmentation with a reduced search space. In: *CVPRW*. 2020 (cit. on p. 105).

[Cuturi, 2013] Marco Cuturi. Sinkhorn distances: lightspeed computation of optimal transport. In: *NeurIPS*. 2013 (cit. on pp. 50–52, 133).

[Dabouei, 2021a] Ali Dabouei, Sobhan Soleymani, Fariborz Taherkhani, and Nasser M. Nasrabadi. SuperMix: Supervising the Mixing Data Augmentation. In: *CVPR*. 2021 (cit. on pp. 97, 107–109, 111, 114, 115).

[Dabouei, 2021b] Ali et al. Dabouei. Supermix: Supervising the mixing data augmentation. In: *CVPR*. 2021 (cit. on p. 50).

[Dai, 2019] Zuozhuo Dai, Mingqiang Chen, Xiaodong Gu, Siyu Zhu, and Ping Tan. Batch dropblock network for person re-identification and beyond. In: *ICCV*. 2019 (cit. on p. 30).

[Dalal, 2005] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In: *CVPR*. 2005 (cit. on pp. 14, 15).

[Damen, 2022] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Jian Ma, Evangelos Kazakos, et al. Rescaling Egocentric Vision: Collection, Pipeline and Challenges for EPIC-KITCHENS-100. In: *IJCV* (2022) (cit. on pp. 126, 128, 141).

[DeGroot, 1983] Morris H DeGroot and Stephen E Fienberg. The comparison and evaluation of forecasters. In: *Journal of the Royal Statistical Society: Series D (The Statistician)* (1983) (cit. on p. 65).

[Dehaene, 2011] Stanislas Dehaene. The number sense: How the mind creates mathematics. OUP USA, 2011 (cit. on pp. 13, 180).

[Deng, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In: *CVPR*. 2009 (cit. on p. 141).

[Deng, 2019] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In: *CVPR*. 2019.

[Devlin, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *NAACL*. 2019 (cit. on p. 17).

[DeVries, 2017a] Terrance DeVries and Graham W Taylor. Dataset augmentation in feature space. In: *arXiv preprint arXiv:1702.05538* (2017).

[DeVries, 2017b] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. In: *arXiv preprint arXiv:1708.04552* (2017) (cit. on pp. 28, 47, 50, 66, 73, 99).

[Doersch, 2020] Carl Doersch, Ankush Gupta, and Andrew Zisserman. CrossTransformers: spatially-aware few-shot transfer. In: *NeurIPS*. 2020 (cit. on p. 51).

[Donahue, 2014] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, et al. Decaf: A deep convolutional activation feature for generic visual recognition. In: *ICML*. 2014 (cit. on p. 71).

[Dosovitskiy, 2021] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In: *ICLR*. 2021 (cit. on pp. 17, 105, 131, 135, 139).

[Dosovitskiy, 2013] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Unsupervised feature learning by augmenting single images. In: *arXiv preprint arXiv:1312.5242* (2013) (cit. on p. 48).

[Duan, 2018] Yueqi Duan, Wenzhao Zheng, Xudong Lin, Jiwen Lu, and Jie Zhou. Deep adversarial metric learning. In: *CVPR*. 2018 (cit. on p. 37).

[Dusenberry, 2020] Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, et al. Efficient and scalable bayesian neural nets with rank-1 factors. In: *ICML*. 2020.

[Elbattah, 2021] Mahmoud Elbattah, Colm Loughnane, Jean-Luc Guérin, Romuald Carette, Federica Cilia, and Gilles Dequen. Variational autoencoder for image-based augmentation of eye-tracking data. In: *Journal of Imaging* (2021).

[Engelbart, 1962] Douglas Engelbart. Augmenting human intellect: A conceptual framework. Summary report. In: *Stanford Research Institute, on Contract AF* (1962).

[Everingham,] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html (cit. on pp. 136, 141).

[Everingham, 2010] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. In: *IJCV* (2010) (cit. on p. 108).

[Faramarzi, 2022] Mojtaba Faramarzi, Mohammad Amini, Akilesh Badrinaaraayanan, Vikas Verma, and Sarath Chandar. PatchUp: A feature-space block-level regularization technique for convolutional neural networks. In: *AAAI*. 2022.

[Fukushima, 1980] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. In: *Biological cybernetics* (1980) (cit. on p. 15).

[Gastaldi, 2017] Xavier Gastaldi. Shake-shake regularization. In: *arXiv preprint arXiv:1705.07485* (2017).

[Gatys, 2016] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image Style Transfer Using Convolutional Neural Networks. In: *CVPR*. 2016.

[Gauthier, 1999] Isabel Gauthier, Michael J Tarr, Adam W Anderson, Pawel Skudlarski, and John C Gore. Activation of the middle fusiform'face area'increases with expertise in recognizing novel objects. In: *Nature neuroscience* (1999).

[Genevay, 2018] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In: *AISTATS*. 2018 (cit. on p. 51).

[Ghiasi, 2018] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In: *NeurIPS* (2018) (cit. on p. 30).

[Gibson, 1957a] James J Gibson. Optical motions and transformations as stimuli for visual perception. In: *Psychological Review* (1957) (cit. on p. 19).

[Gibson, 1957b] James J Gibson and Eleanor J Gibson. Continuous perspective transformations and the perception of rigid motion. In: *Journal of Experimental Psychology* (1957) (cit. on p. 19).

[Gokaslan, 2018] Aaron Gokaslan, Vivek Ramanujan, Daniel Ritchie, Kwang In Kim, and James Tompkin. Improving shape deformation in unsupervised image-to-image translation. In: *ECCV*. 2018.

[Goldberger, 2005] Jacob Goldberger, Sam Roweis, Geoffrey Hinton, and Ruslan Salakhutdinov. Neighbourhood Components Analysis. In: *NIPS*. 2005 (cit. on pp. 76, 77).

[Goodfellow, 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016 (cit. on pp. 14–16).

[Goodfellow, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, et al. Generative adversarial nets. In: *NeurIPS* (2014).

[Goodfellow, 2013] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In: *ICML*. 2013 (cit. on p. 30).

[Goodfellow, 2015] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In: *ICLR* (2015) (cit. on pp. 60, 107).

[Gordo, 2016] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. In: *ECCV*. 2016 (cit. on p. 71).

[Gordon, 2020] Daniel Gordon, Kiana Ehsani, Dieter Fox, and Ali Farhadi. In: *arXiv* (2020) (cit. on pp. 123, 125–127).

[Grabner, 2018] Alexander Grabner, Peter M Roth, and Vincent Lepetit. 3d pose estimation and 3d model retrieval for objects in the wild. In: *CVPR*. 2018.

[Graham, 2021] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, et al. LeViT: A Vision Transformer in ConvNet's Clothing for Faster Inference. In: *ICCV*. 2021.

[Grauman, 2022] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In: *CVPR*. 2022 (cit. on pp. 126, 128).

[Gu, 2018] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In: *CVPR*. 2018 (cit. on pp. 126, 128).

[Gu, 2020] Geonmo Gu and Byungsoo Ko. Symmetrical Synthesis for Deep Metric Learning. In: *AAAI*. 2020 (cit. on pp. 74, 83, 99).

[Gu, 2021] Geonmo Gu, Byungsoo Ko, and Han-Gyu Kim. Proxy Synthesis: Learning with Synthetic Classes for Deep Metric Learning. In: *AAAI*. 2021 (cit. on pp. 38, 74, 83, 85, 87, 99).

[Gulrajani, 2018] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In: *ICLR* (2018) (cit. on p. 56).

[Guo, 2019a] Changlu Guo, Márton Szemenyei, Yang Pei, Yugen Yi, and Wei Zhou. SD-UNet: A structured dropout U-Net for retinal vessel segmentation. In: *BIBE*. 2019 (cit. on p. 30).

[Guo, 2017] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In: 2017 (cit. on p. 65).

[Guo, 2019b] Hongyu Guo, Yongyi Mao, and Richong Zhang. Mixup as locally linear out-of-manifold regularization. In: *AAAI*. 2019 (cit. on pp. 50, 113).

[Haan, 2001] Michelle de Haan, Mark H Johnson, Daphne Maurer, and David I Perrett. Recognition of individual faces and average face prototypes by 1-and 3-month-old infants. In: *Cognitive development* (2001) (cit. on p. 123).

[Hadsell, 2006] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In: *CVPR*. 2006 (cit. on pp. 34, 73, 75, 77, 83, 86, 87, 98).

[Han, 2017] Kai Han, Rafael S Rezende, Bumsub Ham, Kwan-Yee K Wong, Minsu Cho, Cordelia Schmid, et al. Scnet: Learning semantic correspondence. In: *ICCV*. 2017 (cit. on p. 51).

[Harris, 2020] Ethan Harris, Antonia Marcu, Matthew Painter, Mahesan Niranjan, and Adam Prügel-Bennett Jonathon Hare. Fmix: Enhancing mixed sample data augmentation. In: *arXiv preprint arXiv:2002.12047* (2020) (cit. on pp. 47, 50).

[Havasi, 2020] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, et al. Training independent subnetworks for robust prediction. In: *arXiv preprint arXiv:2010.06610* (2020).

[He, 2022] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In: *CVPR*. 2022 (cit. on p. 42).

[He, 2017] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In: *ICCV*. 2017 (cit. on p. 17).

[He, 2016a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In: *CVPR*. 2016.

[He, 2016b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In: *CVPR*. 2016 (cit. on pp. 16, 60, 65, 82, 105).

[He, 2018] Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai. Triplet-center loss for multi-view 3d object retrieval. In: *CVPR*. 2018.

[Henaff, 2020] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In: *ICML*. 2020 (cit. on p. 40).

[Hendrycks, 2019a] Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In: *ICLR* (2019).

[Hendrycks, 2017] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In: *ICLR* (2017) (cit. on pp. 62, 63).

[Hendrycks, 2019b] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. In: *arXiv preprint arXiv:1912.02781* (2019) (cit. on pp. 73, 99, 107–109, 111).

[Hermans, 2017] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. In: *arXiv preprint arXiv:1703.07737* (2017) (cit. on pp. 73, 76, 77, 98).

[Hernández-García, 2018] Alex Hernández-García, Johannes Mehrer, Nikolaus Kriegeskorte, Peter König, and Tim C Kietzmann. Deep neural networks trained with heavier data augmentation learn features closer to representations in hit. In: *CCCN*. 2018.

[Hinton, 2015] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In: *arXiv preprint arXiv:1503.02531* (2015) (cit. on p. 71).

[Hinton, 2012] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. In: *arXiv preprint arXiv:1207.0580* (2012) (cit. on p. 30).

[Hjelm, 2018] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, et al. Learning deep representations by mutual information estimation and maximization. In: *arXiv preprint arXiv:1808.06670* (2018) (cit. on p. 39).

[Ho, 2020] Chih-Hui Ho and Nuno Nvasconcelos. Contrastive learning with adversarial examples. In: *NeurIPS*. 2020 (cit. on p. 36).

[Hong, 2021] Minui Hong, Jinwoo Choi, and Gunhee Kim. StyleMix: Separating Content and Style for Enhanced Data Augmentation. In: *CVPR*. 2021 (cit. on pp. 50, 59–65, 68, 107–109, 111, 115).

[Hsieh, 2016] Peng-Ju Hsieh, Yen-Liang Lin, Yu-Hsiu Chen, and Winston Hsu. Egocentric activity recognition by leveraging multiple mid-level representations. In: *ICME*. 2016.

[Huang, 2019] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. In: *IEEE TPAMI* (2019) (cit. on p. 137).

[Huang, 2021] Lianghua Huang, Xin Zhao, and Kaiqi Huang. GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild. In: *IEEE TPAMI* (2021) (cit. on p. 138).

[Huang, 2017] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In: *ICCV*. 2017 (cit. on p. 50).

[Inoue, 2018] Hiroshi Inoue. Data augmentation by pairing samples for images classification. In: *arXiv preprint arXiv:1801.02929* (2018) (cit. on p. 50).

[Iscen, 2018] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Mining on manifolds: Metric learning without labels. In: *CVPR*. 2018 (cit. on p. 36).

[Isola, 2017] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In: *CVPR*. 2017.

[Ivakhnenko, 1971] Alexey Grigorevich Ivakhnenko. Polynomial theory of complex systems. In: *IEEE transactions on Systems, Man, and Cybernetics* (1971) (cit. on p. 14).

[Jayaraman, 2015] Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. In: *ICCV*. 2015 (cit. on p. 125).

[Jayaraman, 2016] Dinesh Jayaraman and Kristen Grauman. Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. In: *ECCV*. 2016 (cit. on p. 125).

[Jia, 2020] Bin-Bin Jia and Min-Ling Zhang. Multi-dimensional classification via kNN feature augmentation. In: *Pattern Recognition* (2020) (cit. on p. 30).

[Kakogeorgiou, 2022] Ioannis Kakogeorgiou, Spyros Gidaris, Bill Psomas, Yannis Avrithis, Andrei Bursuc, Konstantinos Karantzalos, et al. What to hide from your students: Attention-guided masked image modeling. In: *ECCV*. 2022 (cit. on pp. 138, 139).

[Kalantidis, 2020] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. In: *NeurIPS* (2020) (cit. on pp. 38, 74, 83, 85, 87, 89, 99).

[Kay, 2017] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, et al. The kinetics human action video dataset. In: *arXiv preprint arXiv:1705.06950* (2017) (cit. on pp. 126, 127, 141).

[Khan, 2020a] Abdullah Aman Khan, Jie Shao, Waqar Ali, and Saifullah Tumrani. Content-aware summarization of broadcast sports videos: an audio–visual feature extraction approach. In: *Neural Processing Letters* (2020) (cit. on p. 128).

[Khan, 2020b] Adil Khan and Khadija Fraz. Post-training iterative hierarchical data augmentation for deep networks. In: *NeurIPS* (2020).

[Khosla, 2020] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, et al. Supervised contrastive learning. In: *NeurIPS*. 2020 (cit. on p. 70).

[Kim, 2021a] Jae Myung Kim, Junsuk Choe, Zeynep Akata, and Seong Joon Oh. Keep calm and improve visual feature attribution. In: *ICCV*. 2021 (cit. on p. 97).

[Kim, 2021b] Jang-Hyun Kim, Wonho Choo, Hosan Jeong, and Hyun Oh Song. Co-Mixup: Saliency Guided Joint Mixup with Supermodular Diversity. In: *ICLR*. 2021 (cit. on pp. 47, 49, 50, 57, 59–65, 73, 96, 99, 107–111, 115).

[Kim, 2020a] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In: 2020 (cit. on pp. 22, 31, 33, 47, 49, 50, 52, 57, 59–65, 73, 96, 99, 106–111, 115, 181).

[Kim, 2020b] Sungnyun Kim, Gihun Lee, Sangmin Bae, and Se-Young Yun. MixCo: Mix-up Contrastive Learning for Visual Representation. In: *NeurIPS Workshop on Self-Supervised Learning* (2020) (cit. on pp. 74, 99).

[Kim, 2020c] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In: *CVPR*. 2020 (cit. on pp. 34, 35, 73, 75–77, 82, 83, 86, 87, 99).

[Kingma, 2013] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In: *arXiv preprint arXiv:1312.6114* (2013) (cit. on pp. 66, 68).

[Kipf, 2021] Thomas Kipf, Gamaleldin F Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, et al. Conditional object-centric learning from video. In: *arXiv preprint arXiv:2111.12594* (2021).

[Knight, 2008] Philip A Knight. The Sinkhorn-Knopp algorithm: convergence and applications. In: *SIAM Journal on Matrix Analysis and Applications* (2008) (cit. on pp. 53, 54, 57).

[Ko, 2020] Byungsoo Ko and Geonmo Gu. Embedding expansion: Augmentation in embedding space for deep metric learning. In: *CVPR*. 2020 (cit. on pp. 38, 71, 74, 83, 99).

[Kolesnikov, 2020] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, et al. Big Transfer (BiT): General Visual Representation Learning. In: *ECCV*. 2020 (cit. on p. 71).

[Konno, 2018] Tomohiko Konno and Michiaki Iwazume. Icing on the cake: An easy and quick post-learnig method you can try after deep learning. In: *arXiv preprint arXiv:1807.06540* (2018).

[Krause, 2013] Jonathan Krause, Michael Stark, Jia Deng, and Fei-Fei Li. 3D Object Representations for Fine-Grained Categorization. In: *ICCVW* (2013) (cit. on p. 82).

[Krizhevsky, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. In: (2009) (cit. on pp. 14, 60, 105).

[Krizhevsky, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In: *NIPS* (2012) (cit. on pp. 16, 48).

[Kulis, 2012] Brian Kulis et al. Metric learning: A survey. In: *Foundations and trends in machine learning* (2012).

[Kumar, 2019] Varun Kumar, Hadrien Glaude, Cyprien de Lichy, and William Campbell. A closer look at feature space data augmentation for few-shot intent classification. In: *arXiv preprint arXiv:1910.04176* (2019) (cit. on p. 30).

[Kuo, 2020] Chia-Wen Kuo, Chih-Yao Ma, Jia-Bin Huang, and Zsolt Kira. Featmatch: Feature-based augmentation for semi-supervised learning. In: *ECCV*. 2020.

[Kuznetsova, 2020] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, et al. The open images dataset v4. In: *IJCV* (2020).

[LeCun, 1989] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, et al. Handwritten digit recognition with a back-propagation network. In: *Advances in neural information processing systems* (1989) (cit. on p. 15).

[Lee, 2021] Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. I-Mix: A Domain-Agnostic Strategy for Contrastive Representation Learning. In: *ICLR*. 2021 (cit. on pp. 74, 83, 85, 87, 89, 99).

[Lemley, 2017] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation learning an optimal data augmentation strategy. In: *Ieee Access* (2017).

[Li, 2021a] Boyi Li, Felix Wu, Ser-Nam Lim, Serge Belongie, and Kilian Q Weinberger. On feature normalization and data augmentation. In: *CVPR*. 2021.

[Li, 2020a] Hao Li, Xiaopeng Zhang, Qi Tian, and Hongkai Xiong. Attribute mix: Semantic data augmentation for fine grained recognition. In: *VCIP*. 2020.

[Li, 2020b] Pu Li, Xiangyang Li, and Xiang Long. Fencemask: a data augmentation approach for pre-extracted image features. In: *arXiv preprint arXiv:2006.07877* (2020) (cit. on p. 29).

[Li, 2022] Siyuan Li, Zedong Wang, Zicheng Liu, Di Wu, and Stan Z. Li. OpenMixup: Open Mixup Toolbox and Benchmark for Visual Representation Learning. In: *arXiv preprint arXiv:2209.04851* (2022).

[Li, 2019a] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting Local Descriptor Based Image-To-Class Measure for Few-Shot Learning. In: *CVPR*. 2019.

[Li, 2019b] Xueting Li, Sifei Liu, Shalini De Mello, Xiaolong Wang, Jan Kautz, and Ming-Hsuan Yang. Joint-task self-supervised learning for temporal correspondence. In: *NeurIPS* (2019) (cit. on p. 125).

[Li, 2017] Yao Li, Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. Mining mid-level visual patterns with deep CNN activations. In: *IJCV* (2017).

[Li, 2016] Yinan Li and Fang Liu. Whiteout: Gaussian adaptive noise regularization in deep neural networks. In: *arXiv preprint arXiv:1612.01490* (2016) (cit. on p. 30).

[Li, 2021b] Zhaowen Li, Zhiyang Chen, Fan Yang, Wei Li, Yousong Zhu, Chaoyang Zhao, et al. MST: Masked self-supervised transformer for visual representation. In: *NeurIPS*. 2021.

[Lifchitz, 2019] Yann Lifchitz, Yannis Avrithis, Sylvaine Picard, and Andrei Bursuc. Dense Classification and Implanting for Few-Shot Learning. In: *CVPR*. 2019.

[Lin, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, et al. Microsoft COCO: Common objects in context. In: *ECCV*. 2014 (cit. on pp. 17, 108, 137).

[Liu, 2022a] Jihao Liu, Boxiao Liu, Hang Zhou, Hongsheng Li, and Yu Liu. TokenMix: Rethinking image mixing for data augmentation in vision transformers. In: *ECCV*. 2022 (cit. on pp. 105, 107, 108).

[Liu, 2019] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, et al. Few-shot unsupervised image-to-image translation. In: *ICCV*. 2019.

[Liu, 2016a] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, et al. SSD: Single shot multibox detector. In: *ECCV*. 2016 (cit. on p. 108).

[Liu, 2022b] Wei Liu, Liyan Ma, and Mingyue Cui. Learning-based stereoscopic view synthesis with cascaded deep neural networks. In: *Journal of Advanced Computational Intelligence and Intelligent Informatics* (2022).

[Liu, 2018a] Xiaofeng Liu, Yang Zou, Lingsheng Kong, Zhihui Diao, Junliang Yan, Jun Wang, et al. Data Augmentation via Latent Space Interpolation for Image Classification. In: *ICPR)*. 2018 (cit. on pp. 73, 99).

[Liu, 2018b] Xiaofeng Liu, Yang Zou, Lingsheng Kong, Zhihui Diao, Junliang Yan, Jun Wang, et al. Data augmentation via latent space interpolation for image classification. In: *ICPR*. 2018 (cit. on p. 49).

[Liu, 2022c] Zicheng Liu, Siyuan Li, Di Wu, Zihan Liu, Zhiyuan Chen, Lirong Wu, et al. AutoMix: Unveiling the power of mixup for stronger classifiers. In: *ECCV*. 2022.

[Liu, 2016b] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. In: *CVPR*. 2016 (cit. on p. 82).

[Logothetis, 1996] Nikos K Logothetis and David L Sheinberg. Visual object recognition. In: *Annual review of neuroscience* (1996).

[Long, 2014] Jonathan Long, Ning Zhang, and Trevor Darrell. Do convnets learn correspondence? In: *NIPS*. 2014 (cit. on p. 51).

[Loshchilov, 2019a] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In: *ICLR*. 2019 (cit. on p. 136).

[Loshchilov, 2019b] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In: *ICLR* (2019) (cit. on p. 83).

[Lowe, 1995] David G Lowe. Similarity metric learning for a variable-kernel classifier. In: *Neural computation* (1995).

[Lowe, 2004] David G Lowe. Distinctive image features from scale-invariant keypoints. In: *IJCV* (2004) (cit. on pp. 14, 15).

[Madry, 2018] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In: *ICLR*. 2018 (cit. on pp. 60, 107).

[Mahendran, 2018] Aravindh Mahendran, James Thewlis, and Andrea Vedaldi. Cross Pixel Optical-Flow Similarity for Self-supervised Learning. In: *ACCV*. 2018 (cit. on p. 125).

[Mangla, 2020] Puneet Mangla, Vedant Singh, Shreyas Jayant Havaldar, and Vineeth N Balasubramanian. VarMixup: Exploiting the Latent Space for Robust Training and Inference. In: *arXiv preprint arXiv:2003.06566* (2020).

[Marr, 2010] David Marr. Vision: A computational investigation into the human representation and processing of visual information. MIT press, 2010 (cit. on p. 14).

[Martinez, 2017] Mark Martinez, Chawin Sitawarin, Kevin Finch, Lennart Meincke, Alex Yablonski, and Alain Kornhauser. Beyond grand theft auto V for training, testing and enhancing deep learning in self driving cars. In: *arXiv preprint arXiv:1712.01397* (2017) (cit. on p. 150).

[McCulloch, 1944] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biophysics. In: *The Journal of Symbolic Logic* (1944) (cit. on p. 14).

[McInnes, 2018] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. UMAP: Uniform Manifold Approximation and Projection. In: *The Journal of Open Source Software* (2018) (cit. on p. 112).

[Miech, 2019] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In: *ICCV*. 2019 (cit. on pp. 126, 127).

[Mirza, 2014] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. In: *arXiv preprint arXiv:1411.1784* (2014).

[Misra, 2016] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In: *ECCV*. 2016 (cit. on p. 125).

[Movshovitz-Attias, 2017] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In: *ICCV*. 2017 (cit. on pp. 34, 73, 75–77, 86, 99).

[Musgrave, 2020] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In: *ECCV*. 2020 (cit. on pp. 71, 73, 83, 98).

[Noh, 2015] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In: *ICCV*. 2015 (cit. on p. 97).

[O Pinheiro, 2020] Pedro O Pinheiro, Amjad Almahairi, Ryan Benmalek, Florian Golemo, and Aaron Courville. Unsupervised learning of dense visual representations. In: *NeurIPS*. 2020.

[Oh Song, 2016] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In: *CVPR*. 2016 (cit. on pp. 34, 36, 71, 73, 75, 76, 82, 83, 86, 98).

[Oord, 2018] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. In: *arXiv preprint arXiv:1807.03748* (2018) (cit. on pp. 36, 40).

[Oquab, 2023] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, et al. DINOv2: Learning Robust Visual Features without Supervision. In: *arXiv:2304.07193* (2023) (cit. on p. 133).

[Orhan, 2020] Emin Orhan, Vaibhav Gupta, and Brenden M Lake. Self-supervised learning through the eyes of a child. In: *NeurIPS* (2020) (cit. on p. 125).

[Parkhi, 2012] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and Dogs. In: *CVPR*. 2012.

[Parthasarathy, 2023] Nikhil Parthasarathy, SM Eslami, João Carreira, and Olivier J Hénaff. Self-supervised video pretraining yields strong image representations. In: *NeurIPS*. 2023 (cit. on pp. 123, 125–127, 138).

[Pathak, 2017] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In: *CVPR*. 2017 (cit. on p. 125).

[Patrini, 2020] Giorgio Patrini, Rianne van den Berg, Patrick Forre, Marcello Carioni, Samarth Bhargav, Max Welling, et al. Sinkhorn autoencoders. In: *Uncertainty in Artificial Intelligence*. 2020 (cit. on p. 51).

[Paulin, 2014] Mattis Paulin, Jérôme Revaud, Zaid Harchaoui, Florent Perronnin, and Cordelia Schmid. Transformation pursuit for image classification. In: *CVPR*. 2014 (cit. on p. 48).

[Perez, 2017] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. In: *arXiv preprint arXiv:1712.04621* (2017).

[Perronnin, 2007] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In: *CVPR*. IEEE. 2007 (cit. on p. 15).

[Pirk, 2020] Sören Pirk, Mohi Khansari, Yunfei Bai, Corey Lynch, and Pierre Sermanet. Online learning of object representations by appearance space feature alignment. In: *ICRA*. 2020 (cit. on p. 125).

[Pont-Tuset, 2017] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. In: *arXiv preprint arXiv:1704.00675* (2017) (cit. on pp. 137, 138).

[Qian, 2019] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In: *ICCV*. 2019 (cit. on pp. 34, 73, 75, 86, 99).

[Qin, 2020] Jie Qin, Jiemin Fang, Qian Zhang, Wenyu Liu, Xingang Wang, and Xinggang Wang. ResizeMix: Mixing Data with Preserved Object Information and True Labels. In: *arXiv preprint arXiv:2012.11101* (2020) (cit. on pp. 47, 50, 73, 99).

[Quinn, 1993] Paul C Quinn, Peter D Eimas, and Stacey L Rosenkrantz. Evidence for representations of perceptually similar natural categories by 3-month-old and 4-month-old infants. In: *Perception* (1993) (cit. on p. 123).

[Radford, 2021] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, et al. Learning transferable visual models from natural language supervision. In: *ICML*. 2021 (cit. on p. 152).

[Radosavovic, 2018] Ilija Radosavovic, Piotr Dollar, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data Distillation: Towards Omni-Supervised Learning. In: *CVPR*. 2018.

[Ragusa, 2023] Francesco Ragusa, Antonino Furnari, and Giovanni Maria Farinella. MECCANO: A Multimodal Egocentric Dataset for Humans Behavior Understanding in the Industrial-like Domain. In: *CVIU* (2023) (cit. on p. 126).

[Ramesh, 2021] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, et al. Zero-shot text-to-image generation. In: *ICML*. 2021 (cit. on p. 152).

[Reimers, 2019] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks. In: *EMNLP*. 2019 (cit. on p. 71).

[Ren, 2015] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In: *NIPS*. 2015 (cit. on pp. 17, 108).

[Richter, 2016] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In: *ECCV*. 2016 (cit. on p. 150).

[Robinson, 2021] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In: *ICLR* (2021) (cit. on p. 71).

[Rocco, 2018] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. End-to-end weakly-supervised semantic alignment. In: *CVPR*. 2018 (cit. on p. 51).

[Rohrbach, 2017] Anna Rohrbach, Atousa Torabi, Marcus Rohrbach, Niket Tandon, Christopher Pal, Hugo Larochelle, et al. Movie description. In: *IJCV* (2017).

[Ronneberger, 2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In: *MICCAI*. 2015 (cit. on p. 30).

[Rosenblatt, 1958] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. In: *Psychological review* (1958) (cit. on p. 14).

[Rubner, 2000] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. In: *IJCV* (2000) (cit. on p. 51).

[Rumelhart, 1986] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. In: *nature* (1986) (cit. on p. 15).

[Russakovsky, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, et al. Imagenet large scale visual recognition challenge. In: *IJCV* (2015) (cit. on pp. 16, 40, 60, 65, 82, 105).

[Ryan, 1986] Patrick J Ryan. Euclidean and non-Euclidean geometry: an analytic approach. Cambridge university press, 1986 (cit. on p. 27).

[Saito, 2020] Kuniaki Saito, Kate Saenko, and Ming-Yu Liu. Coco-funit: Few-shot unsupervised image translation with a content conditioned style encoder. In: *arXiv preprint arXiv:2007.07431* (2020).

[Salehi, 2023] Mohammadreza Salehi, Efstratios Gavves, Cees G. M. Snoek, and Yuki M. Asano. Time Does Tell: Self-Supervised Time-Tuning of Dense Image Representations. In: *ICCV* (2023) (cit. on pp. 125, 130).

[Sanakoyeu, 2019] Artsiom Sanakoyeu, Vadim Tschernezki, Uta Buchler, and Bjorn Ommer. Divide and conquer the embedding space for metric learning. In: *CVPR*. 2019 (cit. on p. 86).

[Schroff, 2015] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In: *CVPR*. 2015 (cit. on pp. 71, 73, 98, 99).

[Sener, 2022] F. Sener, D. Chatterjee, D. Shelepov, K. He, D. Singhania, R. Wang, et al. Assembly101: A Large-Scale Multi-View Video Dataset for Understanding Procedural Activities. In: *CVPR* (2022) (cit. on pp. 126, 128).

[Sermanet, 2018] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, et al. Time-contrastive networks: Self-supervised learning from video. In: *ICRA*. 2018 (cit. on p. 125).

[Sharif Razavian, 2014] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In: *CVPR*. 2014 (cit. on p. 15).

[Shen, 2016] Xu Shen, Xinmei Tian, Anfeng He, Shaoyan Sun, and Dacheng Tao. Transform-invariant convolutional neural networks for image classification and search. In: *ACMM*. 2016 (cit. on p. 30).

[Shorten, 2019] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. In: *Journal of big data* (2019).

[Shu, 2021] Yang Shu, Zhangjie Cao, Chenyu Wang, Jianmin Wang, and Mingsheng Long. Open domain generalization with domain-augmented meta-learning. In: *CVPR*. 2021 (cit. on p. 112).

[Simard, 1998] Patrice Y Simard, Yann A LeCun, John S Denker, and Bernard Victorri. Transformation invariance in pattern recognition—tangent distance and tangent propagation. In: *Neural networks: tricks of the trade*. 1998 (cit. on p. 48).

[Siméoni, 2019] Oriane Siméoni, Yannis Avrithis, and Ondrej Chum. Local features and visual words emerge in activations. In: *CVPR*. 2019 (cit. on p. 51).

[Siméoni, 2021] Oriane Siméoni, Gilles Puy, Huy V Vo, Simon Roburin, Spyros Gidaris, Andrei Bursuc, et al. Localizing objects with self-supervised transformers and no labels. In: *BMVC*. 2021 (cit. on pp. 136, 141).

[Simonyan, 2015] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In: *ICLR*. 2015 (cit. on pp. 14, 16, 65).

[Singh, 2017] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In: *ICCV*. 2017.

[Singh, 2018] Krishna Kumar Singh, Hao Yu, Aron Sarmasi, Gautam Pradeep, and Yong Jae Lee. Hide-and-seek: A data augmentation technique for weakly-supervised localization and beyond. In: *arXiv preprint arXiv:1811.02545* (2018) (cit. on p. 29).

[Skiptrace,] Skiptrace. Skiptrace. `https://www.youtube.com/watch?v=LbRNBQaO5a0` (cit. on p. 129).

[Sohn, 2016] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In: *NIPS*. 2016 (cit. on pp. 36, 73, 98).

[Sohn, 2020] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, et al. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In: *arXiv preprint arXiv:2001.07685* (2020) (cit. on p. 71).

[Sokol, 1978] Samuel Sokol. Measurement of infant visual acuity from pattern reversal evoked potentials. In: *Vision research* (1978) (cit. on p. 123).

[Spelke, 2007] Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. In: *Developmental science* (2007) (cit. on p. 123).

[Srivastava, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. In: *JMLR* (2014) (cit. on p. 30).

[Stamey, 1989] Thomas A Stamey, John N Kabalin, John E McNeal, Iain M Johnstone, Fuad Freiha, Elise A Redwine, et al. Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. II. Radical prostatectomy treated patients. In: *The Journal of urology* (1989) (cit. on pp. 13, 180).

[Summers, 2019] Cecilia Summers and Michael J Dinneen. Improved mixed-example data augmentation. In: *WACV*. 2019 (cit. on pp. 47, 50).

[Szegedy, 2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, et al. Intriguing properties of neural networks. In: *ICLR*. 2014 (cit. on p. 48).

[Takahashi, 2018] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Ricap: Random Image Cropping and Patching Data Augmentation for Deep Cnns. In: *ACML*. 2018 (cit. on pp. 47, 50).

[Teh, 2020] Eu Wern Teh, Terrance DeVries, and Graham W Taylor. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In: *ECCV*. 2020 (cit. on pp. 34, 73, 75, 77, 83, 86, 99).

[Thulasidasan, 2019] Sunil Thulasidasan, Gopinath Chennupati, Jeff Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In: *NeurIPS* (2019) (cit. on p. 65).

[Tian, 2024a] Yonglong Tian, Lijie Fan, Kaifeng Chen, Dina Katabi, Dilip Krishnan, and Phillip Isola. Learning vision from models rivals learning vision from data. In: *CVPR* (2024) (cit. on p. 151).

[Tian, 2024b] Yonglong Tian, Lijie Fan, Phillip Isola, Huiwen Chang, and Dilip Krishnan. Stablerep: Synthetic images from text-to-image models make strong visual representation learners. In: *Advances in Neural Information Processing Systems* (2024) (cit. on p. 151).

[Tokozume, 2018] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Learning from between-class examples for deep sound recognition. In: *ICLR*. 2018 (cit. on p. 50).

[Treneska, 2022] Sandra Treneska, Eftim Zdravevski, Ivan Miguel Pires, Petre Lameski, and Sonja Gievska. Gan-based image colorization for self-supervised visual feature learning. In: *Sensors* (2022).

[Tschannen, 2020] Michael Tschannen, Josip Djolonga, Marvin Ritter, Aravindh Mahendran, Neil Houlsby, Sylvain Gelly, et al. Self-supervised learning of video-induced visual invariances. In: *cvpr*. 2020 (cit. on p. 125).

[Uddin, 2021] A F M Uddin, Mst. Monira, Wheemyung Shin, TaeChoong Chung, and Sung-Ho Bae. SaliencyMix: A Saliency Guided Data Augmentation Strategy for Better Regularization. In: 2021 (cit. on pp. 47, 49, 50, 59–65, 73, 96, 99, 107–109, 111, 115).

[Vapnik, 1999] VN Vapnik. An Overview of Statistical Learning Theory. In: *IEEE Transactions on Neural Networks* (1999) (cit. on p. 96).

[Vasudeva, 2021] Bhavya Vasudeva, Puneesh Deora, Saumik Bhattacharya, Umapada Pal, and Sukalpa Chanda. Loop: Looking for optimal hard negative embeddings for deep metric learning. In: *CVR*. 2021 (cit. on p. 36).

[Vaswani, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, et al. Attention Is All You Need. In: *NeurIPS*. 2017.

[Venkataramanan, 2021] Shashanka Venkataramanan, Yannis Avrithis, Ewa Kijak, and Laurent Amsaleg. AlignMix: Improving representation by interpolating aligned features. In: *arXiv preprint arXiv:2103.15375* (2021) (cit. on pp. 25, 73, 77, 99, 106–113, 115, 116).

[Venkataramanan, 2024a] Shashanka Venkataramanan, Amir Ghodrati, Yuki M Asano, Fatih Porikli, and Amirhossein Habibian. Skip-attention: Improving vision transformers by paying less attention. In: *ICLR*. 2024 (cit. on p. 25).

[Venkataramanan, 2023] Shashanka Venkataramanan, Ewa Kijak, Yannis Avrithis, et al. Embedding space interpolation beyond mini-batch, beyond pairs and beyond examples. In: *NeurIPS*. 2023 (cit. on p. 25).

[Venkataramanan, 2022] Shashanka Venkataramanan, Bill Psomas, Ewa Kijak, Laurent Amsaleg, Konstantinos Karantzalos, and Yannis Avrithis. It takes two to tango: Mixup for deep metric learning. In: *ICLR*. 2022 (cit. on p. 25).

[Venkataramanan, 2024b] Shashanka Venkataramanan, Mamshad Nayeem Rizve, João Carreira, Yuki M Asano, and Yannis Avrithis. Is ImageNet worth 1 video? Learning strong image encoders from 1 long unlabelled video. In: *ICLR*. 2024 (cit. on p. 25).

[Venkateswara, 2017] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In: *CVPR*. 2017 (cit. on p. 112).

[Verma, 2019] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, et al. Manifold mixup: Better representations by interpolating hidden states. In: *ICML*. 2019 (cit. on pp. 22, 32, 47, 48, 50–52, 56, 58–65, 68, 71, 73, 76, 77, 84, 96, 97, 99, 101, 105, 107–111, 115, 117, 120, 147, 181).

[Villani, 2008] Cédric Villani. Optimal transport: old and new. 2008 (cit. on pp. 51, 52).

[Vinyals, 2016] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In: *arXiv preprint arXiv:1606.04080* (2016) (cit. on p. 71).

[Wah, 2011a] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Tech. rep. CNS-TR-2011-001. California Institute of Technology, 2011 (cit. on p. 82).

[Wah, 2011b] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Tech. rep. CNS-TR-2011-001. California Institute of Technology, 2011 (cit. on p. 66).

[Wang, 2014] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, et al. Learning Fine-Grained Image Similarity with Deep Ranking. In: *CVPR*. 2014 (cit. on pp. 34, 73, 75, 76, 98).

[Wang, 2020] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In: *ICML*. 2020 (cit. on pp. 72, 89, 113).

[Wang, 2015] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In: *ICCV*. 2015 (cit. on p. 125).

[Wang, 2019a] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In: *CVPR*. 2019 (cit. on p. 125).

[Wang, 2021] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense Contrastive Learning for Self-Supervised Visual Pre-Training. In: *CVPR*. 2021.

[Wang, 2019b] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In: *CVPR*. 2019 (cit. on pp. 34, 36, 71, 73–77, 83, 86, 94, 98, 100).

[Wang, 2019c] Yulin Wang, Xuran Pan, Shiji Song, Hong Zhang, Gao Huang, and Cheng Wu. Implicit semantic data augmentation for deep networks. In: *NeurIPS* (2019).

[Wei, 2022] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In: *CVPR*. 2022 (cit. on p. 43).

[Weinberger, 2009] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In: *JMLR* (2009) (cit. on pp. 73, 86, 98).

[Weinzaepfel, 2013] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. DeepFlow: Large displacement optical flow with deep matching. In: *ICCV*. 2013 (cit. on p. 51).

[Wen, 2016] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In: *ECCV*. 2016.

[Wen, 2020] Yeming Wen, Dustin Tran, and Jimmy Ba. BatchEnsemble: an alternative approach to efficient ensemble and lifelong learning. In: *arXiv preprint arXiv:2002.06715* (2020).

[Werbos, 1974] Paul Werbos. New tools for prediction and analysis in the behavioral science. In: *Ph. D. dissertation, Harvard University* (1974) (cit. on p. 15).

[Wiles, 2022] Olivia Wiles, Joao Carreira, Iain Barr, Andrew Zisserman, and Mateusz Malinowski. Compressed vision for efficient video understanding. In: *ACCV*. 2022 (cit. on pp. 121, 123, 128).

[Wiskott, 2002] Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. In: *Neural computation* (2002) (cit. on p. 125).

[Wohlhart, 2015] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3d pose estimation. In: *CVPR*. 2015.

[Wong, 2016] Sebastien C Wong, Adam Gatt, Victor Stamatescu, and Mark D McDonnell. Understanding data augmentation for classification: when to warp? In: *DICTA*. 2016 (cit. on p. 27).

[Wu, 2017] Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krähenbühl. Sampling Matters in Deep Embedding Learning. In: *ICCV*. 2017 (cit. on pp. 71, 73, 86, 98, 99).

[Wu, 2020] Mike Wu, Chengxu Zhuang, Milan Mosse, Daniel Yamins, and Noah Goodman. On mutual information in contrastive learning for visual representations. In: *arXiv preprint arXiv:2005.13149* (2020) (cit. on p. 36).

[Xiao, 2010] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In: *CVPR*. 2010 (cit. on pp. 63, 111).

[Xiao, 2018] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In: *ECCV*. 2018 (cit. on pp. 136, 137).

[Xie, 2020] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In: *CVPR*. 2020.

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

[Xie, 2021a] Zhenda Xie, Yutong Lin, Zheng Zhang, Yue Cao, Stephen Lin, and Han Hu. Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In: *CVPR*. 2021.

[Xie, 2021b] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, et al. Simmim: A simple framework for masked image modeling. In: *arXiv preprint arXiv:2111.09886* (2021).

[Xie, 2022] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, et al. Simmim: A simple framework for masked image modeling. In: *CVPR*. 2022 (cit. on p. 43).

[Xing, 2003a] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance Metric Learning with Application to Clustering with Side-Information. In: *NeurIPS*. 2003.

[Xing, 2003b] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance Metric Learning with Application to Clustering with Side-Information. In: *NIPS*. 2003 (cit. on pp. 71, 73, 98).

[Xiong, 2021] Yuwen Xiong, Mengye Ren, Wenyuan Zeng, and Raquel Urtasun. Self-supervised representation learning from flow equivariance. In: *ICCV*. 2021 (cit. on p. 125).

[Xu, 2016a] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In: *CVPR*. 2016.

[Xu, 2016b] Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, et al. Improved relation classification by deep recurrent neural networks with data augmentation. In: *arXiv preprint arXiv:1601.03651* (2016) (cit. on p. 27).

[Xuan, 2020] Hong Xuan, Abby Stylianou, and Robert Pless. Improved embeddings with easy positive triplet mining. In: *WACV*. 2020 (cit. on p. 86).

[Yamada, 2019] Yoshihiro Yamada, Masakazu Iwamura, and Koichi Kise. Shakedrop regularization. In: (2019) (cit. on p. 30).

[Yang, 2022] Suorong Yang, Weikang Xiao, Mengcheng Zhang, Suhan Guo, Jian Zhao, and Furao Shen. Image data augmentation for deep learning: A survey. In: *arXiv preprint arXiv:2204.08610* (2022) (cit. on p. 18).

[Yao, 2015] Leon Yao and John Miller. Tiny imagenet classification with convolutional neural networks. In: 2015 (cit. on pp. 60, 105).

[Yi, 2014] Dong Yi, Zhen Lei, and Stan Z. Li. Deep Metric Learning for Practical Person Re-Identification. In: *arXiv preprint arXiv:1703.07737* (2014) (cit. on pp. 76, 77).

[Yu, 2015] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. In: *arXiv preprint arXiv:1506.03365* (2015) (cit. on pp. 63, 111).

[Yu, 2018] Rui Yu, Zhiyong Dou, Song Bai, Zhaoxiang Zhang, Yongchao Xu, and Xiang Bai. Hard-aware point-to-set deep metric for person re-identification. In: *ECCV*. 2018.

[Yun, 2019] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In: *ICCV*. 2019 (cit. on pp. 32, 47, 48, 50, 59–66, 73, 96, 99, 105, 107–111, 115).

[Zagoruyko, 2016a] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In: *BMVC*. 2016.

[Zagoruyko, 2016b] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In: *BMVC*. 2016 (cit. on pp. 60, 105).

[Zeiler, 2014] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In: *ECCV*. 2014 (cit. on p. 120).

[Zhai, 2018] Andrew Zhai and Hao-Yu Wu. Classification is a strong baseline for deep metric learning. In: *arXiv preprint arXiv:1811.12649* (2018) (cit. on p. 83).

[Zhang, 2020] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. DeepEMD: Few-Shot Image Classification With Differentiable Earth Mover's Distance and Structured Classifiers. In: *CVPR*. 2020 (cit. on p. 51).

[Zhang, 2017] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In: *ICLR*. 2017 (cit. on p. 48).

[Zhang, 2018a] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In: *ICLR*. 2018 (cit. on pp. 23, 31, 48, 50–52, 58–66, 71, 73, 76, 95–97, 99, 101, 107–111, 114, 115, 117, 119, 120, 183).

[Zhang, 2022a] Lei Zhang, Na Jiang, Qishuai Diao, Zhong Zhou, and Wei Wu. Person Re-identification with pose variation aware data augmentation. In: *Neural computing and applications* (2022).

[Zhang, 2022b] Shaofeng Zhang, Meng Liu, Junchi Yan, Hengrui Zhang, Lingxiao Huang, Xiaokang Yang, et al. M-Mix: Generating Hard Negatives via Multi-sample Mixing for Contrastive Learning. In: *SIGKDD*. 2022.

[Zhang, 2018b] Xiaolin Zhang, Yunchao Wei, Jiashi Feng, Yi Yang, and Thomas S Huang. Adversarial complementary learning for weakly supervised object localization. In: *CVPR*. 2018 (cit. on p. 66).

[Zhang, 2018c] Xiaolin Zhang, Yunchao Wei, Guoliang Kang, Yi Yang, and Thomas Huang. Self-produced guidance for weakly-supervised object localization. In: *ECCV*. 2018.

[Zhao, 2018] Yiru Zhao, Zhongming Jin, Guo-jun Qi, Hongtao Lu, and Xian-sheng Hua. An adversarial approach to hard triplet generation. In: *ECCV*. 2018 (cit. on p. 37).

[Zheng, 2019] Wenzhao Zheng, Zhaodong Chen, Jiwen Lu, and Jie Zhou. Hardness-aware deep metric learning. In: *CVPR*. 2019 (cit. on pp. 37, 74, 99).

[Zhong, 2020] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In: *AAAI*. 2020 (cit. on pp. 29, 105).

[Zhou, 2016] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In: *CVPR*. 2016 (cit. on pp. 59, 65, 66, 103, 117, 118).

[Zhou, 2017] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In: *CVPR*. 2017 (cit. on pp. 136, 137).

[Zhou, 2022a] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, et al. iBOT: Image bert pre-training with online tokenizer. In: *ICLR*. 2022 (cit. on pp. 43, 129, 134, 136, 138–140, 142, 143).

[Zhou, 2022b] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting Twenty-thousand Classes using Image-level Supervision. In: *ECCV*. 2022 (cit. on pp. 127–129).

[Zhou, 2018] Yanzhao Zhou, Yi Zhu, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Weakly Supervised Instance Segmentation Using Class Peak Response. In: *CVPR*. 2018 (cit. on p. 98).

[Zhu, 2020a] Jianchao Zhu, Liangliang Shi, Junchi Yan, and Hongyuan Zha. AutoMix: Mixup Networks for Sample Interpolation via Cooperative Barycenter Learning. In: *ECCV*. 2020 (cit. on pp. 50, 51, 60, 73, 74, 99).

[Zhu, 2017] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: *ICCV*. 2017.

[Zhu, 2020b] Yuehua Zhu, Muli Yang, Cheng Deng, and Wei Liu. Fewer is More: A Deep Graph Metric Learning Perspective Using Fewer Proxies. In: *NeurIPS* (2020) (cit. on pp. 73, 99).

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

Université de Rennes

Abstract: The primary goal in computer vision is to enable machines to extract meaningful information from visual data, such as images and videos, and leverage this information to perform a wide range of tasks. To this end, substantial research has focused on developing deep learning models capable of encoding comprehensive and robust visual representations. A prominent strategy in this context involves pretraining models on large-scale datasets, such as ImageNet, to learn representations that can exhibit cross-task applicability and facilitate the successful handling of diverse downstream tasks with minimal effort.

To facilitate learning on these large-scale datasets and encode good representations, complex data augmentation strategies have been used. However, these augmentations can be limited in their scope, either being hand-crafted and lacking diversity, or generating images that appear unnatural. Moreover, the focus of these augmentation techniques has primarily been on the ImageNet dataset and its downstream tasks, limiting their applicability to a broader range of computer vision problems.

In this thesis, we aim to tackle these limitations by exploring different approaches to enhance the efficiency and effectiveness in representation learning. The common thread across the works presented is the use of interpolation-based techniques, such as mixup, to generate diverse and informative training examples beyond the original dataset. In the first work, we are motivated by the idea of deformation as a natural way of interpolating images rather than using a convex combination. We show that geometrically aligning the two images in the feature space, allows for more natural interpolation that retains the geometry of one image and the texture of the other, connecting it to style transfer. Drawing from these observations, we explore the combination of mixup and deep metric learning. We develop a generalized formulation that accommodates mixup in metric learning, leading to improved representations that explore areas of the embedding space beyond the training classes. Building on these insights, we revisit the original motivation of mixup and generate a larger number of interpolated examples beyond the mini-batch size by interpolating in the embedding space. This approach allows us to sample on the entire convex hull of the mini-batch, rather than just along linear segments between pairs of examples. Finally, we investigate the potential of using natural augmentations of objects from videos. We introduce a "Walking Tours" dataset of first-person egocentric videos, which capture a diverse range of objects and actions in natural scene transitions. We then propose a novel self-supervised pretraining method called DoRA, which detects and tracks objects in video frames, deriving multiple views from the tracks and using them in a self-supervised manner.

Keywords: Representation learning, mixup, deep metric learning, self-supervised learning

RÉSUMÉ

La conception de représentations efficaces est un élément central de nombreux systèmes d'Intelligence Artificielle (IA), et elle a considérablement évolué au cours des dernières décennies. La performance d'un système d'IA dépend fortement de la qualité de la représentation des données d'entrée. Tout comme les humains trouvent les calculs arithmétiques plus intuitifs en utilisant des chiffres plutôt que des binaires ou des chiffres romains [Dehaene, 2011], la représentation des entrées joue un rôle crucial dans la performance des systèmes d'apprentissage automatique (AA). Par exemple, dans un système simple d'AA chargé d'identifier le risque de cancer de la prostate, le système n'interagit pas directement avec le patient, mais plutôt avec un ensemble de variables, telles que les conditions cliniques et démographiques, recueillies par un spécialiste [Stamey, 1989]. Cet ensemble de variables constitue la représentation du patient vue par l'algorithme d'AA, qui apprend ensuite comment ces différentes variables interagissent pour faire des prédictions.

Au-delà des représentations textuelles et numériques couramment utilisées, les systèmes d'IA modernes se sont étendus pour englober une large gamme de modalités de données. La vision par ordinateur est l'une de ces modalités, qui traite la compréhension des données visuelles telles que les images et les vidéos. Les algorithmes de vision exploitent les représentations pour extraire des informations significatives des entrées visuelles, permettant aux systèmes d'IA de percevoir, d'interpréter et de prendre des décisions basées sur des indices visuels, tout comme les humains le font. Dans le domaine de la vision par ordinateur, la capacité à comprendre le contenu sémantique d'une image est cruciale pour une large gamme de tâches, telles que la classification, la recherche, la détection et la segmentation. De plus, les systèmes de vision par ordinateur peuvent tirer parti de données multimodales, où les images ou les vidéos sont associées à des informations textuelles, pour relever des défis tels que la légende d'images/vidéos et la réponse à des questions visuelles. Ces tâches impliquent souvent deux composants clés : un mécanisme pour extraire des informations de l'image et un mécanisme secondaire pour accomplir la tâche spécifique basée sur les informations extraites.

Nous proposons différentes approches visant à améliorer la performance et la robustesse des encodeurs d'images. Après un aperçu détaillé des différentes méthodes d'augmentation de données en classification d'images, apprentissage métrique et apprentissage auto-supervisé dans le ??, nous présentons les différentes contributions réalisées au cours de ce programme de doctorat dans ces différents contextes.

7.2.1 AlignMixup : une méthode naturelle d'interpolation

L'augmentation de données basée sur l'interpolation, comme le mixup, a montré qu'elle améliore la robustesse et la calibration des modèles [Verma, 2019]. Cependant, comme montré dans [Kim, 2020a], les images d'entrée mixtes ont tendance à paraître non naturelles et la sélection aléatoire des patches et le mélange de leurs étiquettes peuvent amener le classificateur à apprendre des caractéristiques non informatives. Cette limitation suggère que l'exploration de l'interpolation dans l'espace des caractéristiques, plutôt que dans l'espace d'entrée, pourrait être une direction intéressante à poursuivre.

[Bengio, 2013] montrent que traverser le long du manifold des représentations obtenues à partir des couches plus profondes du réseau entraîne plus probablement la découverte d'exemples réalistes. Ils émettent l'hypothèse que les représentations plus profondes apprises par les réseaux neuronaux ont tendance à mieux désentrelacer les facteurs sous-jacents de variation. Ces représentations désentrelacées peuvent être exploitées pour produire des chaînes de Markov à mélange plus rapide, ce qui signifie que les représentations plus profondes permettent en effet un meilleur mélange et génèrent des interpolations plus réalistes entre les points de données.

Motivés par cette observation, nous proposons d'interpoler les images dans l'espace des caractéristiques plutôt que dans l'espace image. Dans le chapter 3, nous montrons que l'idée de déformation est une manière naturelle d'interpoler les images, où une image peut se déformer en une autre, de manière continue. Pour ce faire, nous étudions l'alignement géométrique pour le mixup, basé sur des correspondances sémantiques explicites dans l'espace des caractéristiques. En particulier, nous alignons les tenseurs de caractéristiques de deux images, résultant en des correspondances douces. Nous établissons un nouvel état de l'art en classification d'images, robustesse aux attaques adversariales, calibration, localisation faiblement supervisée et détection des anomalies, surpassant des opérations de mixup plus sophistiquées sur plusieurs réseaux et

ensembles de données.

7.2.2 Extension du mixup à l'apprentissage métrique

Nous avons discuté d'AlignMixup, une technique qui interpole entre des caractéristiques alignées et améliore la performance sur les tâches de classification d'images. Cependant, ces méthodes de mixup ne se généralisent pas à différentes tâches telles que la recherche d'instance et l'apprentissage métrique. Sur cette base, nous explorons maintenant l'idée d'appliquer systématiquement le mixup dans le domaine de l'apprentissage métrique profond.

Il existe une similitude frappante entre l'utilisation de la similarité par paires en apprentissage métrique et l'utilisation de paires d'exemples en mixup pour les tâches de classification. Cette observation nous a conduit à explorer la possibilité d'interpoler entre les paires en apprentissage métrique en utilisant le mixup, de manière similaire à son fonctionnement en classification. Dans le chapter 4, nous introduisons le mixup dans le contexte de l'apprentissage métrique. Cependant, l'interpolation directe des paires d'embeddings présente un défi unique. Contrairement à la classification, les fonctions de perte en apprentissage métrique ne sont pas additives sur les exemples, ce qui rend l'interpolation directe des étiquettes cibles non triviale en utilisant le mixup traditionnel.

Pour relever ce défi, nous développons d'abord une formulation généralisée qui englobe les fonctions de perte d'apprentissage métrique existantes et la modifions pour accueillir le mixup. Cela contribue à une manière systématique d'interpoler les étiquettes, de sorte que le facteur d'interpolation affecte le poids relatif des positifs et des négatifs. Étant donné que l'interpolation entre toutes les paires possibles peut être coûteuse en termes de calcul, nous utilisons une stratégie d'interpolation linéaire efficace, la rendant significativement plus rapide que les méthodes d'interpolation non linéaires complexes. En introduisant le mixup dans l'apprentissage métrique et en développant une formulation généralisée avec une interpolation efficace, nous visons à améliorer la performance des tâches d'apprentissage métrique profond, en nous appuyant sur le succès du mixup en classification d'images et sur les insights obtenus d'AlignMixup dans le ??.

7.2.3 Interpolation au-delà du mini-lot, au-delà des paires et au-delà des exemples

Le chapitre précédent a exploré l'efficacité du mixup dans l'apprentissage métrique profond, où augmenter le nombre de termes de perte en interpolant entre toutes les paires d'embeddings a amélioré la performance sans surcharge computationnelle significative. Cela nous motive à explorer le potentiel d'étendre le mixup davantage dans les tâches de classification en générant plus d'exemples interpolés pendant l'entraînement. La motivation initiale du mixup [Zhang, 2018a], vise à augmenter les données d'entraînement en générant de nouveaux exemples par interpolation. Cependant, elle est limitée à l'interpolation entre paires d'exemples dans l'espace d'entrée, car la combinaison convexe de trois ou plus d'exemples n'apportait pas de gains supplémentaires.

Dans le chapter 5, nous revisitons la motivation initiale du mixup et augmentons le nombre d'exemples augmentés par interpolation dans l'espace d'embeddings. Au lieu d'interpoler dans l'espace d'entrée, nous générons un nombre arbitrairement grand d'exemples interpolés au-delà de la taille du mini-lot en interpolant l'ensemble du mini-lot dans l'espace d'embeddings. Géométriquement, cela se traduit par une interpolation entre tous les points, échantillonnant essentiellement des points sur l'enveloppe convexe du mini-lot. En augmentant le nombre de termes de perte par mini-lot de plusieurs ordres de grandeur à peu de coût supplémentaire, rendu possible par l'interpolation dans l'espace d'embeddings, nous montrons empiriquement des améliorations significatives par rapport aux méthodes de mixup de pointe sur quatre benchmarks différents, malgré l'interpolation étant uniquement linéaire.