DISSERTATION

submitted
to the
Combined Faculty of the Natural Sciences and for Mathematics
of the
Ruberto-Carola Heidelberg University, Germany
for the degree of
Doctor of Natural Sciences

Put forward by

Biagio Brattoli
Born in Foggia, Italy
Oral examination:

Visual Similarity Using Limited Supervision

Advisor: Prof. Dr. Björn Ommer

The challenge of vision should not be answering the question 'what is this?', but asking instead 'what is it like?', thus linking the input with an analogous representation in memory.

[**Moshe Bar**. The proactive brain: memory for predictions (2009), Philosophical Transaction of the Royal Society, 364, 1235-1243]

Abstract

The visual world is a conglomeration of objects, scenes, motion, and much more. As humans, we look at the world through our eyes, but we understand it by using our brains. From a young age, humans learn to recognize objects by association, meaning that we link an object or action to the most similar one in our memory to make sense of it.

Within the field of Artificial Intelligence, Computer Vision gives machines the ability to see. While digital cameras provide eyes to the machine, Computer Vision develops its brain. To that purpose, Deep Learning has emerged as a very successful tool. This method allows machines to learn solutions to problems directly from the data. On the basis of Deep Learning, computers nowadays can also *learn* to interpret the visual world. However, the process of learning in machines is very different from ours. In Deep Learning, images and videos are grouped into predefined, artificial categories. However, describing a group of objects, or actions, with a single integer (category) disregards most of its characteristics and pair-wise relationships. To circumvent this, we propose to expand the categorical model by using *visual similarity* which better mirrors the human approach.

Deep Learning requires a large set of manually annotated samples, that form the training set. Retrieving training samples is easy given the endless amount of images and videos available on the internet. However, this also requires manual annotations, which are very costly and laborious to obtain and thus a major bottleneck in modern computer vision.

In this thesis, we investigate visual similarity methods to solve image and video classification. In particular, we search for a solution where human supervision is marginal. We focus on Zero-Shot Learning (ZSL), where only a subset of categories are manually annotated. After studying existing methods in the field, we identify common limitations and propose methods to tackle them. In particular, ZSL image classification is trained using only discriminative supervision, i.e. predefined categories, while ignoring other descriptive characteristics. To tackle this, we propose a new approach to learn shared features, i.e. non-discriminative, thus descriptive characteristics, which improves existing methods by a large margin. However, while ZSL has shown great potential for the task of image classification, for example in case of face recognition, it has performed poorly for video classification. We identify the reasons for the lack of growth in the field and provide a new, powerful baseline.

Unfortunately, even if ZSL requires only partial labeled data, it still needs supervision during training. For that reason, we also investigate purely unsupervised methods. A successful paradigm is self-supervision: the model is trained using a surrogate task where supervision is automatically provided. The key to self-supervision is the ability of deep learning to transfer the knowledge learned from one task to a new task. The more similar the two tasks are, the more effective the transfer is. Similar to our work on ZSL, we also studied the common limitations of existing self-supervision approaches and proposed a method to overcome them. To improve self-supervised learning, we propose a policy net-

work which controls the parameters of the surrogate task and is trained through reinforcement learning.

Finally, we present a real-life application where utilizing *visual similarity with limited supervision* provides a better solution compared to existing parametric approaches. We analyze the behavior of motor-impaired rodents during a single repeating action for which our method provides an objective similarity of behavior, facilitating comparisons across animal subjects and time during recovery.

Zusammenfassung

Die visuelle Welt ist ein Konglomerat von Objekten, Szenen, Bewegungen und vielem mehr. Als Menschen betrachten wir die Welt durch unsere Augen, wohingegen die visuellen Informationen von unserem Gehirn verarbeitet werden. Schon in jungen Jahren lernen Menschen Objekte durch Assoziation zu erkennen, was bedeutet, dass wir ein Objekt oder eine Handlung mit unserem Wissen, das wir über mehrere Jahre aufgebaut haben verknüpfen, um einen Sinn daraus zu ziehen.

Im Bereich der künstlichen Intelligenz verleiht Computer Vision Maschinen die Fähigkeit zu sehen. Während Digitalkameras als die Augen einer Maschine agieren, befasst sich Computer Vision damit das "Gehirn" einer Maschine weiterzuentwickeln. Zu diesem Zweck hat sich Deep Learning als sehr erfolgreiches Werkzeug herausgestellt. Diese Methode ermöglicht es Maschinen, Lösungen für Probleme direkt aus den Daten zu erlernen. Auf der Grundlage von Deep Learning können Computer *lernen* die visuelle Welt eigenständig zu interpretieren. Der Lernprozess in Maschinen unterscheidet sich jedoch stark von unserem. In Deep Learning werden Bilder und Videos in vordefinierte künstliche Kategorien eingeteilt. Bei der Beschreibung einer Gruppe von Objekten oder Aktionen mit einer einzelnen Zahl (Kategorie) werden jedoch viele ihrer Merkmale und paarweisen Beziehungen nicht berücksichtigt. Um dies zu umgehen, schlagen wir vor, das kategoriale Modell durch Verwendung von *visuellen Ähnlichkeiten* zu erweitern, da dies eher die menschliche Vorgehensweise widerspiegelt.

Deep Learning erfordert eine große Menge an manuell annotierten Beispielen, die den Trainings-Datensatz bilden. Glücklicherweise ist das Zusammenstellen eines Datensatzes angesichts der fast endlosen Anzahl von Bildern und Videos im Internet einfach. Ein Trainings-Datensatz erfordert allerdings auch manuelle Annotationen, deren Beschaffung sehr kostspielig und mühsam ist und daher einen großen Engpass für moderne Computer Vision Methoden darstellt.

In dieser Dissertation untersuchen wir visuelle Ähnlichkeitsmethoden um die Klassifizierung von Bildern und Videos zu verbessern. Insbesondere suchen wir nach Lösungen die nur eine geringfügige Kontrolle von Menschen erfordert. Dabei konzentrieren wir uns vorwiegend auf Zero-Shot Learning (ZSL), bei dem nur eine Teilmenge von Kategorien manuell mit Annotationen versehen wird. Aufgrund einer von uns durchgeführten detaillierten Analyse von existierenden ZSL Methoden war es uns möglich methodenübergreifende Limitierungen zu identifizieren. Darüber hinaus schlagen wir verschiedene Ansätze vor, um die gefundenen Limitierungen zu beseitigen. Wir beschäftigen uns insbesondere mit der Einschränkung, dass existierende ZSL Methoden, Modelle zur Bild-Klassifierungen nur unter der Verwendung von diskriminierender Überwachung (vordefinierte Kategorien) trainieren während andere beschreibende Merkmale ignoriert werden. Daher schlagen wir einen neuen Ansatz vor, bei dem gemeinsame, d.h. nicht diskriminierende, Merkmale gelernt werden. Diese Vorgehensweise verbessert existierende Methoden um ein Vielfaches. Während ZSL sein großes Potenzial im Lösen von Bild-Klassifizierungs Problemen gezeigt hat, erreicht es im Falle von Video-Klassifizierungen allerdings nur mangelhafte Leistungen. Wir identifizieren

die Gründe für die geringen Verbesserungen, die in den letzten Jahren in diesem Bereich erreicht wurden und bieten eine neue, leistungsstarke Basis an.

Leider benötigt ZSL immer einen Datensatz der zumindest teilweise annotiert ist. Aus diesem Grund untersuchen wir zusätzlich Methoden, die ausschließlich auf unbeaufsichtigten (nicht annotiert) Datensätzen basieren. Ein erfolgreiches Paradigma in diesem Bereich ist die Selbstüberwachung: Das Modell wird mithilfe einer Ersatzaufgabe trainiert, bei der die Überwachung automatisch erfolgt. Der Schlüssel zur Selbstüberwachung liegt in der Fähigkeit von Deep Learning. Das von einer Aufgabe erlernte Wissen kann dank Deep Learning auf eine neue Aufgabe übertragen werden. Je ähnlicher die beiden Aufgaben sind, desto effektiver funktioniert die Übertragung. Ähnlich zu unserer Arbeit in ZSL haben wir auch hier die allgemeinen Limitierungen bestehender Ansätze untersucht und bieten eine neue Methode an, die diese umgeht. Um das selbstüberwachte Lernen zu verbessern, schlagen wir ein Strategie-Netzwerk vor, das die Parameter der Ersatzaufgabe steuert und durch bestärkendes Lernen geschult wird.

Schließlich stellen wir eine reale Anwendung vor, bei der die Verwendung von *visuellen Ähnlichkeiten mit begrenzter Überwachung* eine bessere Lösung als bestehende parametrische Ansätze bietet. Wir analysieren das Verhalten von Nagetieren, die an einer neurologischen Erkrankung leiden. Unsere Methode ermöglicht ein objektives Auswerten von Videos, die die Tiere, während sie eine bestimmte Bewegung wiederholt ausführen, zeigen. Insbesondere vergleichen wir die Erfolgsrate von unterschiedlichen Tieren und untersuchen die Verbesserungen die durch eine Behandlung auftreten.

Acknowledgements

I would like to thank my advisor Prof. Dr. Björn Ommer for giving me the chance to work on my passion over the last years. Also, I own him the critical thinking that I build over my PhD.

I am deeply grateful to my colleagues who made the office feel like home. In particular, I want to thank Sabine, Miguel and Timo for their friendship. I am grateful for all the people I shared memories over the past years: Artsiom, Niko, Karsten, Nawid, Tobias, Pablo, Patrick, Johannes, Michael. A huge thanks goes to Pamela for her big help in boring paperwork and the morning laughs.

Keeping the best for last, I thank Uta for constantly testing my patients in stressful situations, but also for her strengths that kept me motivated and was key to a successful PhD.

Contents

Chapter 1

Introduction

Artificial intelligence[139] (AI) gives machines the ability to perceive the world and make decisions. Similarly to the invention of the steam engine, AI may bring a new industrial revolution that will change society[138]. Machines do not get tired and can access a huge amount of data, abilities which humans lack. Because of that, AI has a strong impact on several industries, particularly in those fields where objectively analyzing a large flow of data is crucial, such as finance[169] and cyber-security[1]. Concurrently, AI is entering our homes and everyday life. Products like Echo from Amazon and Home from Google can recognize and partially understand human language. Automatic translators are getting better every year, surpassing human level in some cases[113]. A single algorithm[145] can defeat the best human chess and go players.

A fundamental part of AI is the ability to understand the visual world. Cameras give eyes to computers, however "seeing", i.e. understanding the content of an image or video, is more complex and still an open research topic. Although, digital images are just a series of numbers, humans can naturally interpret them by recognizing objects, scenes, places, postures, and other properties. Computer vision[51] is the field of research that develops algorithms which enable machines to "see". The core of vision is the recognition of patterns in images[36, 35]. More specifically, objects are made of parts[48] which in turn are made of basic components, such as lines and curves. For example, a face is typically made of two eyes, a nose, and a mouth, and a person is typically made of a face, arms, legs, etc... The relative arrangement of parts forms a specific patter which can be leveraged to recognize the object. In this sense, vision can be defined as pattern recognition in images and videos. Therefore, given a set of predefined object/action categories, the algorithm is developed to recognize the specific patterns that distinguish the categories from each other. For example, an algorithm that separates horses from zebras will search for stripes on the animal.

However, designing by hand an algorithm that can search for those discriminative patterns is nearly impossible. Fortunately, using machine learning, we can *learn* an approximate solution directly from the data. CV algorithms can discover, or *learn*, the discriminative patterns necessary to distinguish object categories. Commonly, the learning process requires a human to define the target categories and annotate each data sample with one of those categories. Methods

of this nature are referred to as *supervised* and the collection of annotated data used for the learning process is called "training set" or "training data". Labelling each image (or video) in the training set entails that the human annotator has a clear distinctions of what defines an object (or action) and how to effectively group samples into non-overlapping categories.

However, what represents an object, a scene or an action is not well defined. A way to describe complex objects is through a hierarchical structure. For example, a car is made of wheels, doors, and other constituents. This raises the question: Should the car be labelled as a whole or each part independently? Are we satisfied to recognize the object as a "car" or do we need more fine-grained categories such as the car model and brand? A similar issue appear with actions: when does an action start or end? In case of a person jumping, when does the "jumping" start? Is it the exact moment when the person lifts his feet from the ground or a few seconds earlier when it prepares for the jump? Should we split the jump in phases, such as ready, going up, going down? There is no general answer to these questions[117, 86], as the answer depends on the context and the problem to be solved. However, even when the problem is well defined, the subjective interpretation of the object could introduce human bias through the annotations. Using the earlier example, if a part of the car is occluded, do we also label the missing part? Also, how much of the car can be missing for it to be still considered a "car"? Developing a single model to solve computer vision will therefore not be possible as long as scientists force discrete categories on a continuous visual world[105]. Currently, the common approach is to develop very specialized methods. For example, one model is able to recognize faces, while another gives cars the ability to recognize driving street and obstacles.

In this thesis, we research a more general vision model by substituting the concept of "category" with "visual similarity"[105, 117, 86]. Furthermore, we limit the need for human annotations during the learning process which has two major benefits: decreasing the annotation bottleneck allowing for larger training set and reducing the human bias within the annotation process[64].

In the remainder of the section, we examine the concepts of object classification, which apply to any computer vision task.

Visual Similarity Humans do not explore and learn about the world in discrete categories, but through exemplars[105], meaning that we associate any new object with the most similar one in our memory. This concept can also be applied to machines, thus instead of asking the model "Which category does this belong to?", we ask "What is this similar to?" [117]. In this way, it is possible to develop a more flexible model of the visual world, closer to reality. For example, two dogs are more similar to each other than to a cat, and two german shepherds will be more similar than a chihuahua. However, learning visual similarity is a hard task. A perfect model would require a human annotator to compare every pair of images existing in the training set, which is infeasible due to the cost growing exponentially with the number of samples. This brings us directly to the second topic of this thesis: reducing the amount of annotation needed for training, i.e. limiting human supervision.

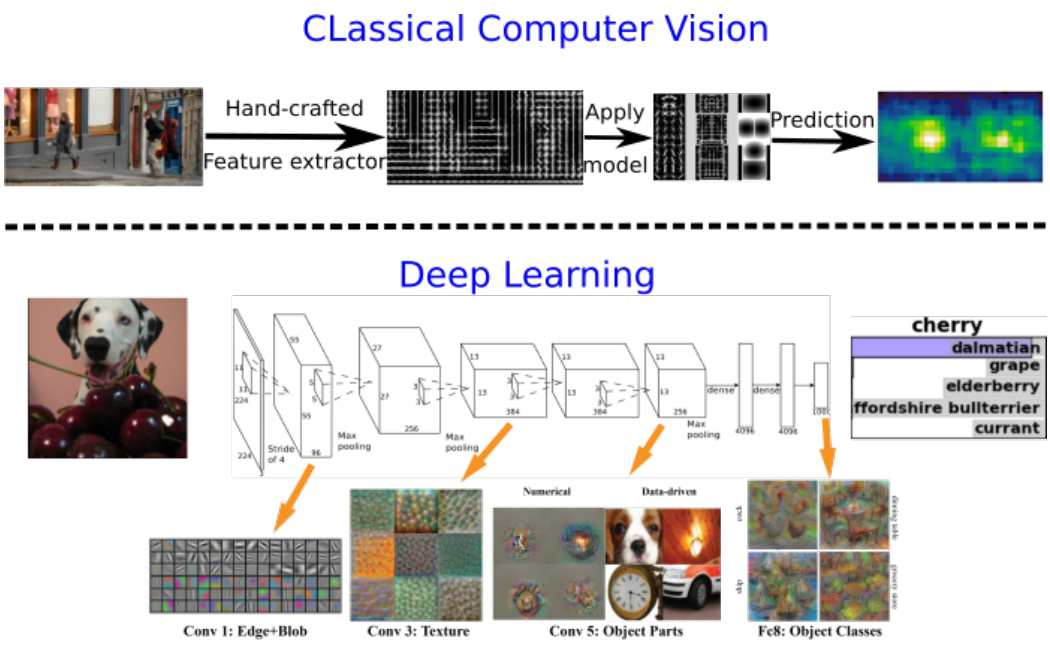

Figure 1.1: Comparison between a classical computer vision (CV) pipeline and a deep learning model. (Top) A classical pipeline extracts salient information from the input image using hand-crafted features. Learned filters are then applied to solve the task. In this example, taken from Felzenszwalb et al.[48], the HOG features[31] emphasizes strong edges. A part-based model, trained using SVM[29], detects a person within the image. (Bottom) This figure, taken from https://www.cc.gatech.edu/ hays/compvision/proj6/, shows the feature extractors automatically learned by the network. Hand-crafted features are not necessary to encode the image. The network shows a hierarchical representation, from low level information, such as edges, to more abstract concepts, object parts, to the complete object.

Limited Human Supervision Until three decades ago, the main obstacle of computer vision was the lack of digital images. Thanks to technological innovations, the abundance of cameras and the internet, it is easy to collect an infinite amount of images and videos nowadays. However, simply having the data is not enough to train a ML model. Each training sample still requires meticulous manually annotation. This created a new bottleneck in ML: Annotation -providing millions of images with thousands of object categories is very costly and time-consuming. Furthermore, human annotation is prone to error - either because of practical mistakes or because of ambiguities in the task itself. This and the constraint of the visual world into a categorical system make it hard to accurately reflect reality. These issues motivate the search for new methods to learn the model using less manual annotation.

In this thesis, we cover two types of limited supervision: zero-shot (ZSL) and unsupervised learning. In ZSL, the model is learned on a limited number of categories and tested on a new set of categories, never seen during training. Unsupervised methods are approaches that do not require any human supervision for training and learn representations directly from the data.

1.1 Deep Representation Learning for Computer Vision

As mentioned earlier in this chapter, a computer vision algorithm recognizes an object by looking for discriminative patterns within the image. Since finding those patterns is a hard task, computer vision researchers use machine learning (ML) to learn salient patterns directly from the training data[19] For example, given a simple dataset of handwritten digits, a classical ML classifier (e.g. decision trees, support vector machine, multi-layer perceptron) could be applied directly on the raw pixels, achieving good performances. However, a real-life task has a much higher complexity than a simple white digit on black background. Therefore a computer vision task can not be solved by a classifier applied directly on the raw image. For this reason, the image is processed before usage, to remove redundant information and retain only crucial ones. In other words, the classifier needs to be invariant to uninformative characteristics, such as brightness, tiny spatial shifts, rotation or occlusion, and focus on crucial features like edges, corners, or motion in videos. The algorithm to break down an image into essential information or features is called "feature extractor". Extracting the right information is a very difficult task and has been a relevant research topic for decades. For this reason, in classical computer vision, researchers would focus on hand-crafting specialized feature extractors instead of improving on the classifier responsible for the actual prediction. The output of the feature extractor is what we call a *visual representation*[106]. Over the last four decades, researchers have developed many features extractors, some focused on shape (Hough lines[10]), others on motion (optical flow[5], HOOF), or color gradient (SIFT[98] and HOG[31]). However, in the last decade a new paradigm became popular in computer vision, so much so that classical feature extractors are becoming obsolete. This new paradigm is called Deep Learning (DL)[91, 57] and is based on artificial neural networks (ANN)[92], a ML method able to automatically learn feature extractor and classifier concurrently. ANN have been around for three decades, however, because of limitations in data and computation, they were never powerful enough to compete with other methods. However, the exponential rise of available and specialized compute made ANNs much more attractive in recent years.

Arguably, the high effectiveness of DL compared to classical CV stems from better visual representations [85]. In particular, the DL representation is[91]:

- Hierarchical: the model is composed of several layers where the output (representation) of a layer is the input to another. (Hence the term "deep" in the name.)

- Learned: the filters are directly learned from the data to solve the specific task. In early layers, some DL filters do resemble the hand-crafted ones, however most of the filters retrieve new information.

- Large capacity: the number of filters in a deep learning model is orders of magnitude larger than the biggest model using hand-crafted features.

Hierarchy, training, and capacity enables the model to learn more complex and abstract concepts[176]. For example, to detect a person, the neural network learns that the body has a face, which is composed of eyes, nose, and mouth, which in turn are made of lines and curves[176]. Fig. 1.1 shows a comparison between a classical pipeline and a modern deep learning model.

1.2 Transfer learning for Self-supervision

Humans have the ability to transfer the knowledge learned from one task to another. For example, a tennis player is better at badminton than a person who never played either because the tennis player can utilize what he learned from one sport (tennis) in the other (badminton). This ability is called Transfer Learning (TL) and is essential for our everyday life, preventing us from learn each small activity from scratch. In machine learning the same concept applies[152, 37], however it has been less successful in classical ML because the learned classifier is typically very specialized. On the other hand, the hierarchical nature of deep neural network allows them to extract more generic features, less specialized on the actual task that they are learning. Therefore, after training a network on a task, it is possible to *transfer* the general features to a new task. The first phase comprising the initial training takes the name of "pre-training" and the second containing the adjustment to a secondary task, is referred to as "fine-tuning". The final evaluation task is called "downstream task". As an example, let's say that our pre-training task is to distinguishing dogs from cats and the downstream task is distinguishing dog's breed. To solve the first task, the network needs to learn classical features of the two animals, such as face, legs, fur, and so on. When solving the second task, even if it is data is notably different, the network can utilize the previous knowledge and does not need to learn it from scratch. Naturally, the closer the pre-trained task is to the downstream task, the more effecting the transfer is. In fact, transfer learning can be used to measure similarity between two tasks[2].

Transfer learning is particularly effective when the first task has a large amount of training data available while the second task only has few annotated data. The most common example is object detection[132, 96]: annotating the full image with a single label (image classification) is much faster than pointing at all objects within the image (object detection). Therefore, it is common practice to pre-train the network on image classification and fine-tune on object detection, since more annotated training samples are available for the first task. This is important because, in general, the more data is used for training, the stronger the visual representation will be. Unsupervised representation learning fills this niche perfectly as theoretically it has an infinite amount of data available. Pre-training can be done in an unsupervised fashion first to produce a very powerful visual representation, and then fine-tuned on the downstream task where annotations are limited.

Unsupervised methods Unsupervised deep learning methods can be distinguished in two main categories:

- Generative models[67, 81, 58]: the task is to approximate the data distribution or reconstruct each single sample.
- Self-supervision[38, 114]: the model is trained using a surrogate task for which labels are freely available, typically by transforming the input and using the network to identify the transformation.

Generative models have been a first attempt to a generic unsupervised representation[83, 128]. However, these methods tend to focus too much on fine-grained features of the image (such as background), since they need to approximate the data distribution. Moreover, they are very costly since generating the image requires extra computation. Self-supervision on the other hand as recently shown promise to learn unsupervised representation. Unlike reconstruction-based approaches, self-supervised training is closer to the downstream task since it utilizes a similar objective function (called loss) instead of reproducing the input image. In Chapter 4, we will review some of the most common self-supervision methods and propose a framework for learning a generic representation for images and videos.

1.3 Contributions

This thesis provides the following contributions:

- We highlight an important weakness of metric learning, where existing methods are based solely on discriminative features, ignoring crucial inter-class relations.
- Our novel training forces the network to learn inter-class characteristics together with classical discriminative features, boosting any existing metric learning method by a large margin.
- Research in zero-shot learning for video classification is making very slow progress. We identify the reasons behind that: existing models are shallow, hence the low performance, and unreasonably complex, making the methods not reproducible. Finally, used training sets are unnecessarily small and don't exploit modern dataset.
- We propose a new zero-shot learning protocol for video classification , which supports a faster progress in the field, and a strong baseline method easy to reproduce and modify.
- We train a controller via reinforcement learning to adjust the hyper-parameters of the surrogate task to maximize the generalization capability of the learned representation.
- We apply self-supervision to encode behavior in a representation for objectively quantify motor-skill changes in rodents and compare behavior across animals.

- We compare animal behavior during training and rehabilitation, showing how visual similarity can assist drug evaluation and comparison of rehabilitation methods.

1.4 Thesis Organization

Chapter 2. Before discussing unsupervised learning, we study zero-shot learning (ZSL) as an intermediate problem between unsupervised and supervised learning. Specifically, we train a model on labeled data and test on classes never seen during training, for example, recognize a bird speci or car model. ZSL is typically solved by learning similarities between images, called metric learning, instead of classifying them into predefined categories. In Chapter 2, we discuss existing methods and propose our contribution to improve the state-of-the-art. The proposed method was previously published at ICCV 2019.

Chapter 3. In the last few years, researchers have worked on improving ZSL for images and were able to apply to a real-world applications. At the same time, ZSL for video classification has not made much progress and is far from being used in a real product. In Chapter 3 we study the current state of the field and analyze why the progress is slow. Moreover, we provide a new, strong baseline and evaluation method which could help the field progress at an increased pace. This work was published at CVPR 2020.

Chapter 4. After ZSL, this thesis focuses on unsupervised learning, changing the problem from having some supervision to a total lack thereof. In particular, we study the paradigm of self-supervision: the network is trained using a surrogate task for which labels are freely available. In Chapter 4 we give an overview of existing self-supervised approaches and propose an unsupervised method to learn a video representation. Moreover, we boost self-supervised methods by combining images and videos into a single framework. Finally, we improve self-supervision even further by optimizing the surrogate task during training using an automatic controller. We utilize modern reinforcement learning methods to learn the controller. The work discussed in Chapter 4 was published as part of ECCV 2018.

Chapter 5. In Chapter 5, we apply our unsupervised video representation to a real world scenario: to objectively evaluate behavior. In particular, given several rodents performing the same, repetitive action, we compare their motor functions to tell the level of impairment of each animal. This is instrumental in comparing different rehabilitation methods and drugs. Moreover, we can predict the complex brain status by only watching the behavior. This work was partially published at CVPR 2017. Follow-up work is under review for a journal publication. The behavior analysis was a joint contribution with Uta Buechler.

Chapter 6. Finally, we draw the conclusion on visual similarity when the supervision is incomplete and discuss future direction.

Chapter 2

Image Classification of Unseen Objects

The goal of Zero-shot learning (ZSL) [89, 122] is recognizing classes never seen during training. ZSL belongs to limited supervision since it requires that only a limited set of classes are available during training while testing on a different set of classes. Typically, the challenge is limited to a single object type (cars, birds, faces, ...) and the task is more a fine-grained classification: given a specific objects, distinguish its fine-grained variations (car model, bird species, face ID). Distinguish faces by training on airplanes is unrealistic.

Metric learning is the standard way to tackle zero-shot learning. Instead of predicting the exact object class, the model learns the visual similarity between objects. In this way, the model can generalize to classes never seen during training.

By far, the most successful field for ZSL is face recognition. Every day millions of devices are unlocked using face ID technology. This technology reached very high precision in the latest years thanks to metric learning.

In this chapter, we tackle a common limited supervision scenario (ZSL). Our contribution is a method to boost the visual representation for boosting state-of-the-art.

After formally define metric learning (Sec.2.1.1) and providing a short list of related work, we motivate our contribution (Sec. 2.2) and describe our method in details (Sec. 2.3). Finally, we evaluate the method showing how it boost performances for many previous methods (Sec. 2.4). The work done in this chapter was previously published in Roth *et al.* [136].

2.1 Background

In this section, we are going to formally define the metric learning algorithm and provide an overview of existing methods.

2.1.1 Preliminaries

In metric learning, the network learns to extract discriminative features and project the images from pixel space to a lower-dimensional euclidean space. The

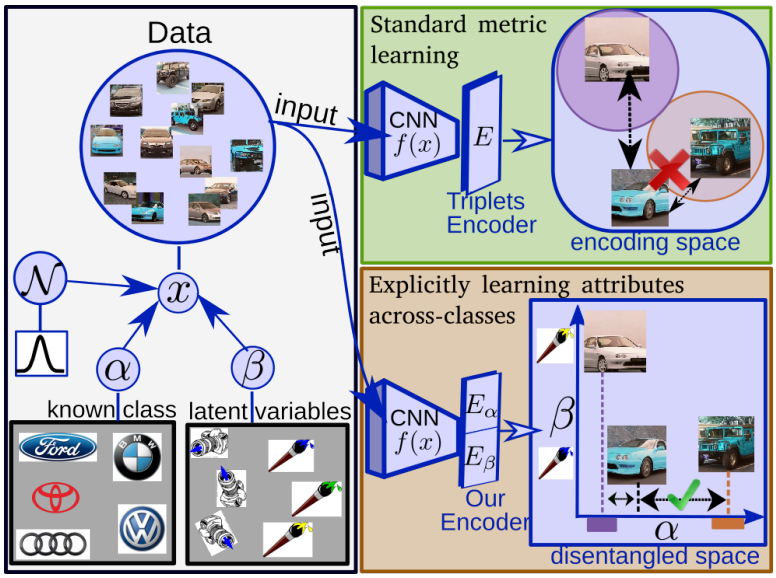

Figure 2.1: (Left) Images can be described by combinations of latent characteristics and white noise. (Green) Standard metric learning encoders extract class-discriminative information α while disregarding object-specific properties β (e.g. color, orientation). Achieving invariance to such characteristics requires substantial training data. (Brown) Instead, the model can explain them away by learning their structure explicitly. Our novel approach explicitly separates class-specific and shared properties during training to boost the performance of the discriminative encoding.

encoder E projects images x_i of class y nearby in the encoding space, while samples from different classes are far apart.

The visual representation $f(x)$ is extracted using a neural network $f : \mathbb{R}^{Height \times Width \times 3} \rightarrow \mathbb{R}^F$, which is then passed to the embedding $E : \mathbb{R}^F \rightarrow \mathbb{R}^D$. Typically, E is implemented as a fully connected layer of dimension D. f and E can be combined in a single neural network, therefore they can be learned jointly using standard backpropagation. The network is trained by enforcing that $d_{ij} < d_{ik}$ if $y_j = y_i$ and $y_k \neq y_i$, where $d_{ij} = ||E(f(x_i)) - E(f(x_j))||^2$ is the euclidean distance between the images x_i and x_j. Formally, the loss is defined as $l = \max\left(d_{ij} - d_{ik} + m, 0\right)$ where m is a margin parameter and $y_j = y_i$ and $y_k \neq y_i$. Many variants of this loss have been proposed recently, with margin loss[167] (adding an additionally learnable margin β) proving to be best.

2.1.2 Related Work

Metric learning is effective in various computer vision applications, such as object retrieval [119, 167], zero-shot learning [167] and face verification [28, 143]. The triplet paradigm [143] is the standard in the field and much work has been done to improve upon the original approach. As an exponential number of possible triplets makes the computation infeasible, many papers propose solutions for

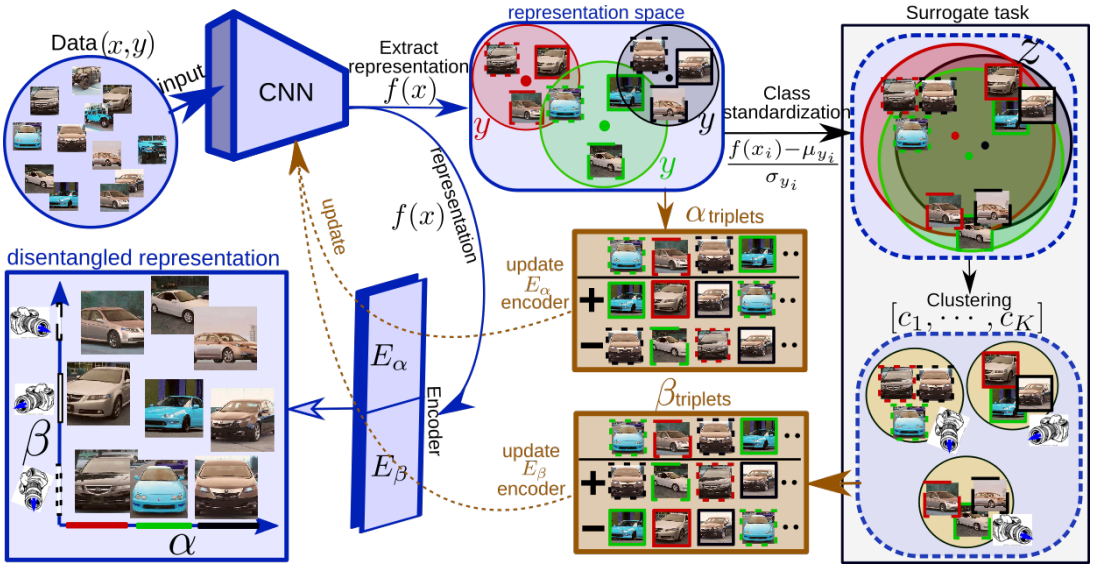

Figure 2.2: **Overview of our approach**. We aim to learn two separate encoding spaces s.t. class information α extracted by E_α is free from shared properties β by explicitly describing them through an auxiliary encoder E_β. Given a set of image/label pairs (x, y), their CNN feature representation $f(x)$ groups images by both class-specific (car model) and shared (orientation, color) characteristics. We separate these by training the class-discriminative encoder E_α with ground-truth labels (*boundary color*). Simultaneously, an auxiliary encoder E_β is trained on labels from a surrogate task (right) to explain away interclass features. The required surrogate labels are generated by standardizing the embedded training data per class and performing clustering. This recovers labels representing the shared structures β (*contour line-styles*). Training both tasks together, E_α learns a robust, β-free encoding, which is now explicitly explained by E_β.

mining triplets more efficiently [167, 143, 63, 54, 72]. Recently, Duan *et al.* [41] have proposed a generative model to directly produce hard negatives. ProxyNCA [112] generates a set of class proxies and optimizes the distance of the anchor to said proxies, solving the triplet complexity problem. Others have explored orthogonal directions by extending the triplet paradigm, e.g. making use of every sample in the (specifically constructed) batch at once [119, 148], enforcing an angular triplet constraint [160], minimizing a cluster quality surrogate [118] or optimizing the overlap between positive and negative similarity histograms [155]. Also, ensembles have been quite successfully used by combining multiple encoding spaces [120, 121, 175, 52] to maximize their efficiency.

Our work makes use of class-agnostic grouping of our data (see e.g. [11, 12]) and shares similarities with proposals from Liu *et al.*. [94], who explicitly decompose images into class-specific and intra-class embeddings using a generative model, as well as Bai *et al.*. [9], who, before training, divide each image class into subgroups to find an approximator for intra-class variances that can be included in the loss. However, unlike [9, 94], we explicitly search for structures shared between classes instead of modelling the intra-class variance per sample [94] or class [9]. Also, unlike [9], we assume class-independent intra-class variance and

iteratively train a second encoder to model intra-class features, thereby purifying the main encoder from non-discriminative features and achieving significantly better results.

Finally, some works have exploited the latent structure of the data as a supervisory signal [114, 116, 24, 21, 23, 142, 141]. In particular, Caron *et al.*. [24] learn an unsupervised image representation by clustering the data, starting from a Sobel filter before initialization. Our approach includes such latent data structures in a similar way, however, we use it as auxiliary information to improve upon the metric learning task.

2.2 Motivation

The pixel space is a very high dimensional space composed of structured dimensions, but also many unstructured, noisy dimensions. If we ignore the unstructured components, we can describe an image using a few dimensions where the structured information is encoded. A typical computer vision algorithm learns salient latent characteristics for a specific task. For example in object classification, discriminative characteristics (e.g. car shape) are used to group the images according to predefined classes. In order to eliminate the unstructured information (e.g. random clutter, occlusion, image brightness), we force the neural network to become invariant to them, for example by using data augmentation. However, there is still structured information which is not used for the classification task because they are shared among classes, but cannot be simply disregarded as noise (e.g. viewpoints and notions of color).

Carefully extracting the shared features becomes especially important in metric learning. Learning similarities is a complex task that requires a very detailed description, or encoding, of the image. Therefore, finding a strong set of latent characteristics is crucial. Explicitly handling those characteristics shared across classes should, therefore, benefit the model (Lin *et al.* [94]), as it can better explain the object variance within a class. Take for example a model trained only on white cars of a certain category. This model will very likely not be able to recognize a blue car of the same category (Fig. 2.1 top-right). In this example, the encoder ignores the concept of "color" for that particular class, even though it can be learned from the data as a latent variable shared across all cars (Fig. 2.1 bottom-right). This is a typical generalization problem and is traditionally solved by providing **more labeled data**. However, besides being a costly solution, metric learning models need to also generalize to unknown classes, a task which should work independently from the amount of labels provided.

Several work [94, 73, 9] have shown that explicitly modeling intro-class variation is beneficial for the model. For example, spatial transformer layers [73] explicitly learn the possible rotations and translations of an object category.

Our model learns the classical metric learning task of discriminating between classes, but at the same time, it learns the shared characteristics of the objects. The discriminative and shared features are learned by two separate encoders, respectively the class and auxiliary encoder. While we use ground-truth labels to learn the class encoder, the auxiliary encoder is trained through a novel surrogate task that extracts class-independent information in an unsupervised way.

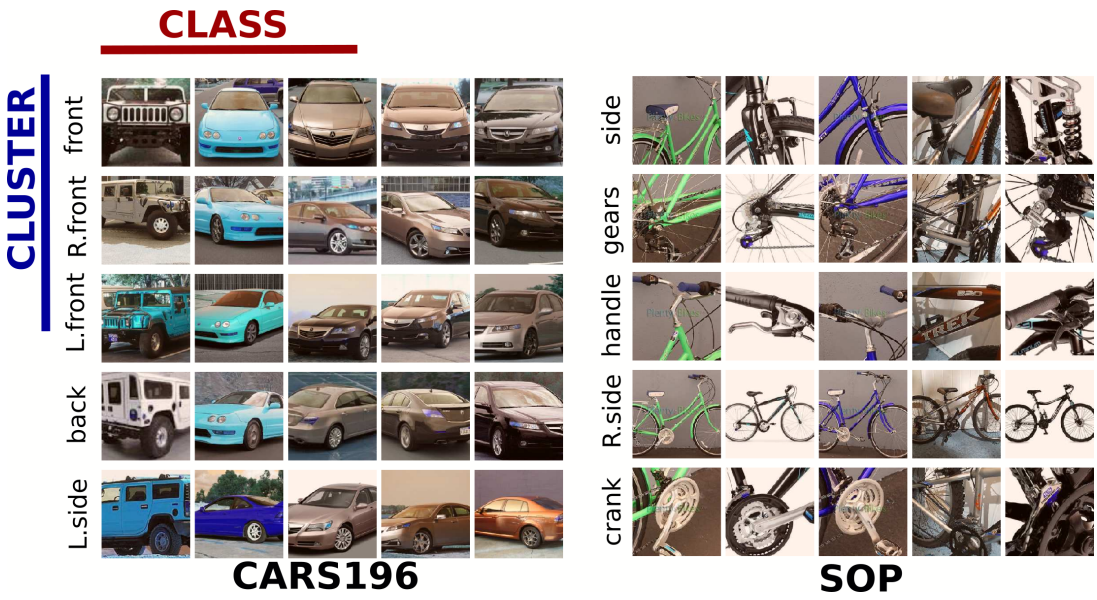

Figure 2.3: Example of clustering the data based on Z (see Sec2.3.2) for two datasets: CARS196[84] and SOP[119]. We group the dataset into 5 clusters (rows) and select the first 5 classes (columns) with at least one sample per cluster. For each entry, we selected the sample closest to the centroid per class. On the left is our interpretation of the cluster structure. The results show that subtraction of the class-specific features by standardization helps to group images based on more generic properties, like car orientation and bike parts.

Finally, we extract the shared characteristics learned by the auxiliary encoder from the discriminative ones adopting a mutual information loss. This purifies the class encoder from non-discriminative, shared characteristics.

This solution can be utilized with any standard metric learning loss, as shown in the result section (Sec. 2.4.3). Our approach is evaluated on three standard benchmarks for zero-shot learning, CUB200-2011 [158], CARS196 [84] and Stanford Online Products [119], as well as two more recent datasets, In-Shop Clothes [185] and PKU VehicleID [95]. The results show that the proposed approach consistently enhances the performances of existing methods.

2.3 Approach

The main idea behind our method is the inclusion of class-shared characteristics into the metric learning process to help the model explain them away. In doing so, we would gain robustness to intrinsic, not-discriminative properties of the data, which is contrary to the common approach of simply forcing invariance towards them. However, three main problems arise with this approach, namely: *(i)* Extracting both class and class-independent characteristics using a single encoder is infeasible and detrimental to the main goal. *(ii)* We lack the labels for extracting these latent properties. *(iii)* We need to explicitly remove unwanted properties from the class embedding. We propose solutions to each of these problems in sections 2.3.1, 2.3.2 and 2.3.3.

Algorithm 1 Training a model via MIC

Input: data X, full encoder E, inter-/intra class encoders $\{E_\alpha, E_\beta\}$, CNN f, class targets Y_α, batchsize bs, clusternumber C, update frequency T_U, (adversarial) mutual information loss l_d and weight γ, projection network R, gradient reversal op r, metric learning loss functions for $E_{\alpha,\beta}$ $l_{\alpha,\beta}$

$Y_\beta \leftarrow$ `Cluster(Stand(Embed(`X, E, f`)), `C`)`
$epoch \leftarrow 0$
while *Not Converged* **do**
  repeat
    $b_\alpha, b_\beta \leftarrow$ `GetBatch(`X, Y_α, Y_β, bs`)`
    $e_{\alpha,\beta} \leftarrow$ `Embed(`$b_{\alpha,\beta}$, $E_{\alpha,\beta}$, f`)`
    $L^\alpha \leftarrow l_\alpha(e_\alpha, Y_\alpha) + \gamma \cdot l_d(e^r_\alpha, \mathrm{R}(e^r_\beta))$
    $E_\alpha, f \leftarrow$ `Backward(`L^α`)`
    $e_{\alpha,\beta} \leftarrow$ `Embed(`$b_{\alpha,\beta}$, $E_{\alpha,\beta}$, f`)`
    $L^\beta \leftarrow l_\beta(e_\beta, Y_\beta) + \gamma \cdot l_d(e^r_\alpha, \mathrm{R}(e^r_\beta))$
    $E_\beta, f \leftarrow$ `Backward(`L^β`)`
  until *end of epoch*;
  if $epoch \bmod T_U == 0$ **then**
    $Y_\beta \leftarrow$ `Cluster(Embed(`X,E_β,f`), `C`)`
  end
  $epoch \leftarrow epoch + 1$
end

2.3.1 Auxiliary Encoder

To separate the process of extracting both inter- and intra-class (shared) characteristics, we utilize two separate encodings: a class encoder E_α which aims to extract class-discriminative features and an auxiliary encoder E_β to find shared properties. These encoders are trained together (Fig.2.2). To efficiently train the underlying deep neural network, the two encoders share the same image representation $f(x)$ which is updated by both during training. In the first training task, the class encoder E_α is trained using the provided ground truth labels $y_1, \cdots, y_N$ associated with each image $x_1, \cdots, x_N$ with N the number of samples. A respective, metric-based loss function can be selected arbitrarily (such as a standard triplet loss or the aforementioned margin loss), as this part follows the generic training setup for metric learning problems. Because labels are not provided for the training of our auxiliary encoder, we define an automatic process to mine shared latent structure information from the original data. This information is then used to provide a new set of training labels to train our auxiliary encoder (Fig.2.2 right). As the training scheme is now equivalent to the primary task, we may choose from the same set of loss functions.

2.3.2 Extracting Inter-class Characteristics

We seek a task which, without human supervision, spots structured characteristics within the data while ignoring class-specific information. As structured proper-

R@k	Dim	1	2	4	NMI
DVML[94]	512	52.7	65.1	75.5	61.4
BIER[120]	512	55.3	67.2	76.9	-
HTL[54]	512	57.1	68.8	78.7	-
A-BIER[121]	512	57.5	68.7	78.3	-
HTG[182]	-	59.5	71.8	81.3	-
DREML[174]	9216	63.9	75.0	83.1	67.8
Semihard[143]	-	42.6	55.0	66.4	55.4
Semihard*	128	57.2	69.4	79.9	63.9
MIC+semih	128	58.8	70.8	81.2	66.0
ProxyNCA[112]	64	49.2	61.9	67.9	64.9
ProxyNCA*	128	57.4	69.2	79.1	62.5
MIC+ProxyNCA	128	60.6	72.2	81.5	64.9
Margin[167]	128	63.6	74.4	83.1	69.0
Margin*	128	62.9	74.1	82.9	66.3
MIC+margin	128	**66.1**	**76.8**	**85.6**	**69.7**

Table 2.1: Recall@k for k nearest neighbor and NMI on CUB200-2011 [158]. Our model outperforms all previous approaches, even those using a larger number of parameters. (*) indicates our best re-implementation with ResNet50.

R@k	Dim	1	2	4	NMI
HTG[182]	-	76.5	84.7	90.4	-
BIER[120]	512	78.0	85.8	91.1	-
HTL[54]	512	81.4	88.0	92.7	-
DVML[94]	512	82.0	88.4	93.3	67.6
A-BIER[121]	512	82.0	89.0	93.2	-
DREML[174]	9216	86.0	91.7	95.0	76.4
Semihard[143]	-	51.5	63.8	73.5	53.4
Semihard*	128	65.5	76.9	85.2	58.3
MIC+semih	128	70.5	80.5	87.4	61.6
ProxyNCA[112]	64	73.2	82.4	86.4	-
ProxyNCA*	128	73.0	81.3	87.9	59.5
MIC+ProxyNCA	128	75.9	84.1	90.1	60.5
Margin[167]	128	79.6	86.5	90.1	**69.1**
Margin*	128	80.0	87.7	92.3	66.3
MIC+margin	128	**82.6**	**89.1**	**93.2**	68.4

Table 2.2: Recall@k for k nearest neighbor and NMI on CARS196 [84]. DREML[174] is not comparable given the large embedding dimension. (*) indicates our ResNet50 re-implementation.

R@k	Dim	1	10	100	NMI
DVML[94]	512	70.2	85.2	93.8	**90.8**
BIER[120]	512	72.7	86.5	94.0	-
ProxyNCA[112]	64	73.7	-	-	-
A-BIER[121]	512	74.2	86.9	94.0	-
HTL[54]	512	74.8	88.3	94.8	-
Margin[167]	128	72.7	86.2	93.8	**90.7**
Margin*	128	74.4	87.2	94.0	89.4
MIC+margin	128	**77.2**	**89.4**	**95.6**	90.0

Table 2.3: Recall@k for k nearest neighbor and NMI on Stanford Online Products [119]. (*) indicates our ResNet50 re-implementation.

ties are generally defined by characteristics shared among several images, they create homogeneous groups. To find these, clustering offers a well-established solution. This algorithm associates images to surrogate labels $c_1, \cdots, c_N$ with $c_i \in [1, \cdots, C]$ and C being the predefined number of clusters. However, applied directly to the data, this method is biased towards class-specific structures since images from the same class share many common properties, like color, context, and shape, mainly injected through the data collection process (e.g. a class may be composed of pictures of the same object from multiple angles).

To remove the characteristics shared within the class, we apply normalization guided by the ground truth classes. For each class y we compute the mean μ_y and standard deviation σ_y based on the features $f(x_i), \forall x_i : y_i = y$. Then we obtain the new standardized image representation $Z = [z_1, \cdots, z_N]$with $z_i = \frac{f(x_i) - \mu_{y_i}}{\sigma_{y_i}}$, where the class influence is now reduced. Afterwards, the auxiliary encoder E_β can be trained using the surrogate labels $[c_1, \cdots, c_N]$ produced by clustering the space Z.

To be effective, a strong prior is needed. It is a standard procedure for deep metric learning to initialize the representation backend f with weights pre-trained on ImageNet. This provides a sufficiently good starting point for clustering, which is then reinforced through training E_β.

Fig.2.3 shows some examples of clusters detected using our surrogate task. This task and the encoder training are summarized in Fig.2.2.

2.3.3 Minimizing Mutual Information

The class encoder E_α and auxiliary encoder E_β can then be trained using the respective labels. As we utilize two different learning tasks, E_α and E_β learn distinct characteristics. However, as both share the same input, the image features $f(x)$, a dependency between the encoders can be induced, therefore leading to both encoders learning some similar properties. To reduce this effect and to constrain the discriminative and shared characteristics into their respective encoding space, we introduce a mutual information loss, which we compute through an

R@k	Dim	1	10	30	50
BIER[120]	512	76.9	92.8	96.2	97.1
HTG[182]	-	80.3	93.9	96.6	97.1
HTL[54]	512	80.9	94.3	97.2	97.8
A-BIER[121]	512	83.1	95.1	97.5	98.0
DREML[174]	9216	78.4	93.7	96.7	-
Margin*	128	84.5	95.7	97.6	98.3
MIC+margin	128	**88.2**	**97.0**	**98.0**	**98.8**

Table 2.4: Recall@k for k nearest neighbor and NMI on In-Shop [185]. (*) indicates our best re-implementation with ResNet50

Test Splits		Small		Large	
R@k	Dim	1	5	1	5
MixDiff+CCL[95]	-	49.0	73.5	38.2	61.6
GS-TRS[9]	-	75.0	83.0	73.2	81.9
BIER[120]	512	82.6	90.6	76.0	86.4
A-BIER[121]	512	86.3	92.7	81.9	88.7
DREML[174]	9216	88.5	94.8	83.1	92.4
Margin*	128	85.1	92.4	80.4	88.9
MIC+margin	128	**86.9**	**93.4**	**82.0**	**91.0**

Table 2.5: Recall@k for k nearest neighbor and NMI on PKU VehicleID[95]. DREML[174] is not comparable given the large embedding dimension. (*) our best ResNet50 re-implementation

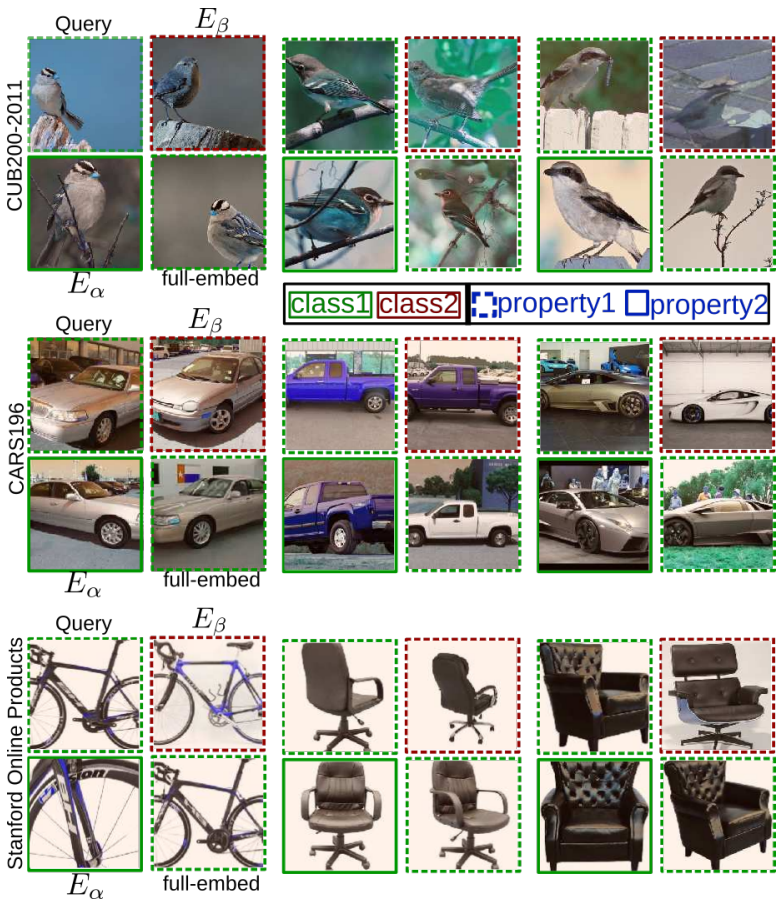

Figure 2.4: Qualitative nearest neighbor evaluation for CUB200-2011, CARS196 and SOP based on E_α and E_β encodings and their combination. The results show that E_β leverages class-independent information (posture, parts) while E_α becomes independent to those features and focuses on the class detection. The combination of the two reintroduces both.

adversarial setup

$$l_d = -\left(E_\alpha^r(f(x)) \odot R(E_\beta^r(f(x)))\right)^2 \tag{2.1}$$

with R being a learned, small two-layered fully-connected neural network with normalized output projecting E_β to the encoding space of E_α. $\odot$ stands for an elementwise product, while the r superscript notes a gradient reversal layer [53] which flips the gradient sign s.t. when trying to minimize l_d, i.e. maximizing correlation, the similarity between both encoders is actually decreased. A similar method has been adopted by [121], where shared information is minimized between an ensemble of encoders. In contrast, our goal is to transfer non-discriminate characteristics to an auxiliary encoder. Finally, as l_d scales with R, we avoid trivial solutions (e.g. $R(E_\beta) \to \infty$) by enforcing $R(E_\beta)$ to have unit length, similar to E_α and E_β.

Finally, the total loss L to train our two encoders and the representation f is computed by $L = l_\alpha + l_\beta + \gamma l_d$, where γ weights the contribution of the mutual information loss with respect to the class triplet loss l_α and the auxiliary triplet

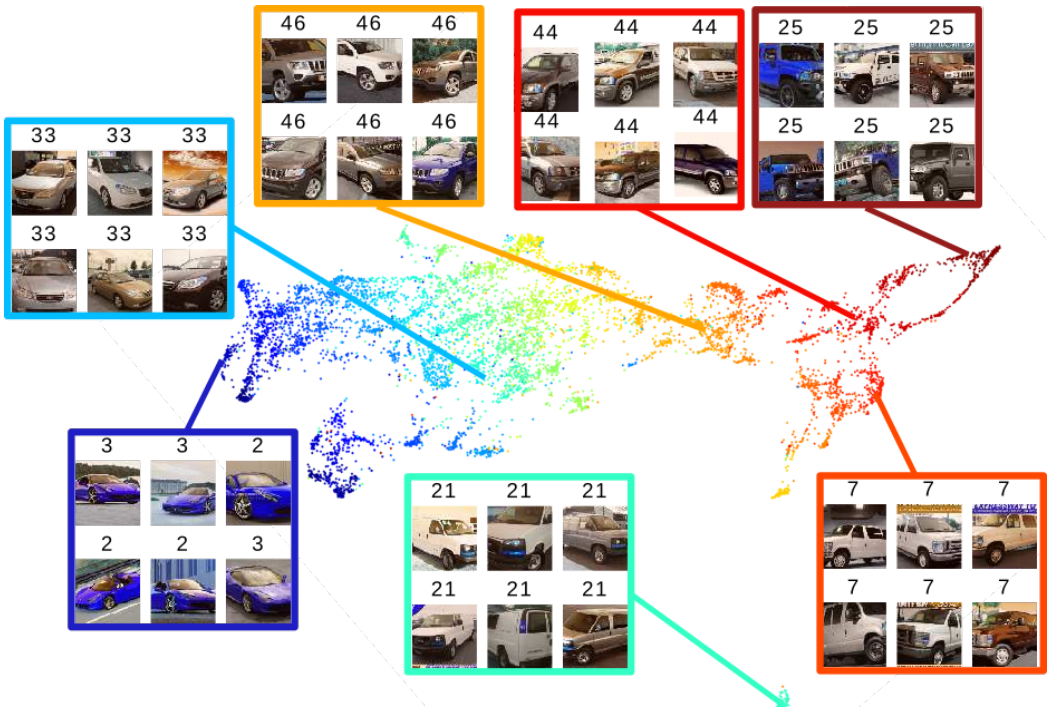

Figure 2.5: UMAP projection of E_α for CARS196. Seven clusters are selected, showing six images near the centroid and their ground-truth labels. We see that the encoding extracts class-specific information and ignores other (e.g. orientation).

loss l_β. The full training is described in Alg. 1.

2.4 Experiments

In this section we offer a quantitative and qualitative analysis of our method, also in comparison to previous work. After providing technical information for reproducing the results of our model, we give some information regarding the standard benchmarks for metric learning and provide comparisons to previous methods. Finally, we offer insights into the model by studying its key components.

2.4.1 Implementation Details

We implement our method using the PyTorch framework [123]. As baseline architecture, we utilize ResNet50 [66] due to its widespread use in recent metric learning work. All experiments use a single NVIDIA GeForce Titan X. Practically, class and auxiliary encoders E_α and E_β use the same training protocol (following [167] with embedding dimensions of 128) with alternating iterations to maximize the usable batch-size. The dimensionality of the auxiliary encoder E_β is fixed (except for ablations in sec. 2.5) to the dimensionality of E_α to ensure similar computational efficiency compared to previous work. However, due to GPU memory limitations, we use a batchsize of 112 instead of a proposed 128, with no relevant changes in performance.

During training, we randomly crop images of size 224×224 after resizing to 256×256, followed by random horizontal flips. For all experiments, we use the original images without bounding boxes. We train the model using Adam [80] with a learning rate of 10^{-5} and set the other parameters to default. We set the triplet parameters following [167], initializing $\beta = 1.2$ for the margin loss and $\alpha = 0.2$ as fixed triplet margin. Per mini-batch, we sample $m = 4$ images per class for a random set of classes, until the batch size is reached. For γ (Sec. 2.3.3 eq.) we utilize dataset-dependent values in $[100, 2000]$ determined via cross-validation.

After class standardization, the clustering is performed via standard k-means using the faiss framework [76]. Using the hyperparameters proposed in this paragraph, the computational cost introduced by our approach is 10-20% of total training time. For efficiency, the clustering can be computed on GPU using faiss[76]. The number of clusters is set before training to a fixed, problem-specific value: 30 for CUB200-2011 [158], 200 for CARS196 [84], 50 for Stanford Online Products [119], 150 for In-Shop Clothes [185] and 50 for PKU VehicleID [95]. We update the cluster labels every other epoch. Notably, however, our model is robust to both parameters since a large range of parameters give comparable results. Later in section 2.5 we study the effect of cluster numbers and cluster label update frequencies for each dataset in more detail to motivate the chosen numbers. Finally, class assignments by clustering, especially in the initial training stages, becomes near arbitrary for samples further away from cluster centers. To ensure that we do not reinforce such a strong initial bias, we found it beneficial to ease the class constraint by randomly switching samples with samples from different cluster classes (with probability $p \leq 0.2$).

2.4.2 Datasets

Our model is evaluated on five standard benchmarks for image retrieval typically used in deep metric learning. We report the Recall@k metric [74] to evaluate image retrieval and the normalized mutual information score (NMI) [101] for the clustering quality. The training and evaluation procedure follows the standard setup as used in [167].

CARS196[84] with 196 car models over 16,185 images. We use the first 98 classes (8054 images) for training and the remaining 98 (8131 images) for testing.

Stanford Online Products[119] with 120,053 product images in 22,634 classes. 59,551 images (11,318 classes) are used for training, 60,502 (11,316 classes) for testing.

CUB200-2011[158] with 200 bird species over 11,788 images. Train and Test Sets contain the first and last 100 classes (5,864/5,924 images) respectively.

In-Shop Clothes[185] with 72,712 clothing images in 7,986 classes. 3,997 classes are used for training and 3,985 classes for evaluation. The test set is divided into a query set (14,218 images) and a gallery set (12,612 images).

PKU VehicleID[95] with 221,736 surveillance images of 26,267 vehicles with shared car models. We follow [95] and use 13,134 classes (110,178 images) for training. Testing is done on a predefined small and large testing subset with 7,332 (small) and 20,038 (large) images respectively.

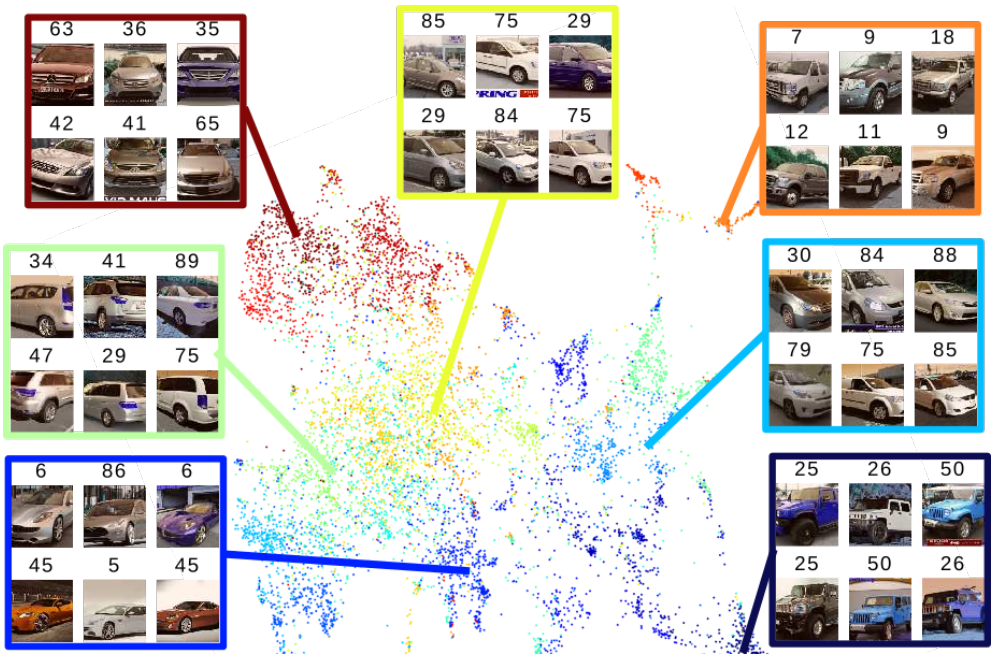

Figure 2.6: UMAP projection of E_β for CARS196. Seven clusters are selected, showing six images near the centroid and their GT labels. The result shows that the encoding extracts intrinsic characteristics of the object (car) independent from GT classes.

2.4.3 Quantitative and Qualitative Results

In this section we compare our approach with existing models from recent literature. Our method is applied on three different losses, the standard triplet loss with semi-hard negative mining [143], Proxy-NCA [112] and the state-of-the-art margin loss with weighted sampling [167]. For full transparency, we also provide results with our re-implementation of the baselines.

The results show a consistent gain over the state of the art for all datasets , see tables 2.1, 2.2, 2.3, 2.4 and 2.5. In particular, our approach achieves better results than more complex ensembles. On CUB200-2011, we outperform even DREML [174] which trains 48 ResNet models in parallel.

Qualitative results are shown in Fig.2.4: the class encoder E_α retrieves images sharing class-specific characteristics, while the auxiliary encoder E_β finds intrinsic, class-independent object properties (e.g. posture, context). The combination retrieves images with both characteristics.

2.5 Ablations

In this section, we investigate the properties of our model and evaluate its components. We qualitatively examine the proposed encoder properties by checking recalled images for both and study the influence of E_β on the recall performance,

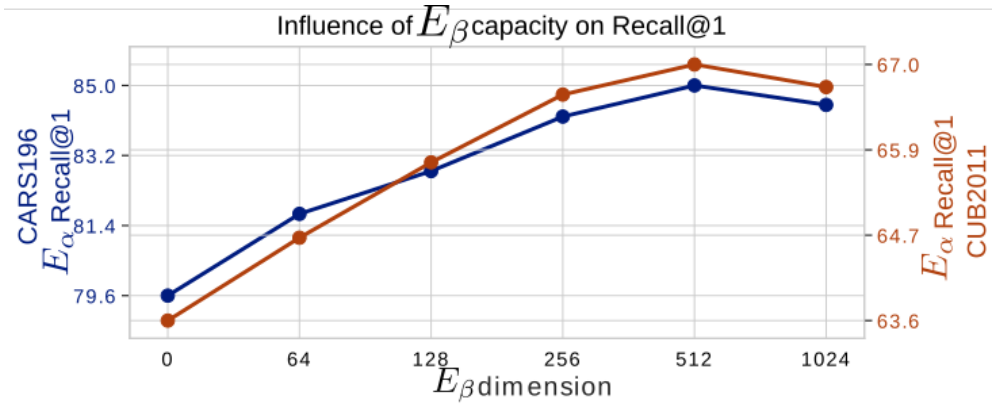

Figure 2.7: Evaluation of E_α as a function of the E_β capacity. For CARS196 [84] and CUB200-2011 [158], we plot E_α Recall@1 against the E_β dimension during training. The results show that the increase in capacity of E_β and thus the ability to learn properties shared among classes directly benefits the class encoder E_α.

see Section 2.5.1. In Section 2.5 we measure the relation between the intra-class variance and the capacity of our auxiliary encoder E_β. Also, ablation studies are performed to examine the relevance of each pipeline component and hyper-parameter. We primarily utilize the most common benchmarks CUB200-2011, CARS196 and SOP.

2.5.1 Embedding Properties

Firstly, we visualize the characteristics of the class encoder E_α (Fig.2.5) and auxiliary encoder E_β (Fig.2.6) by projecting the embedded test data to two dimensions using UMAP[104]. The figures show E_α extracting class-discriminative information while E_β encodes characteristics shared across classes (e.g. car orientation).

To evaluate the effect of the auxiliary encoder E_β on the class encoder E_α, we study the properties of the class encoding as function of the capability of E_β to learn shared characteristics. First, we study the performance of E_α on CARS196[84] and CUB200-2011[158] relative to the auxiliary encoder dimension. Utilizing varying E_β dimensionalities, Fig.2.7 shows a direct relation between E_β capacity and the retrieval capability. E_β with dimension 0 indicates the baseline method [167]. For all other evaluations, the E_β dimension is equal to E_α to keep the computational cost comparable to the baseline [167] (see Sec.2.4.1).

To examine our initial assumption that learning shared characteristics produces more compact classes, we study the intra-class variance by computing the mean pairwise distances per class, averaged over all classes. These distances are normalized by the average inter-class distance, approximated by the distance between two class centers. Summarized in fig.2.8 we see higher intra-class variance for basic margin loss (E_β dimension equal to 0). But more importantly, the class compactness is directly related to the capacity of the auxiliary encoder E_β.

We also offer a qualitative evaluation of the surrogate task in Fig.2.3. After class-standardization, the clustering recognizes latent structures of the data shared across classes.

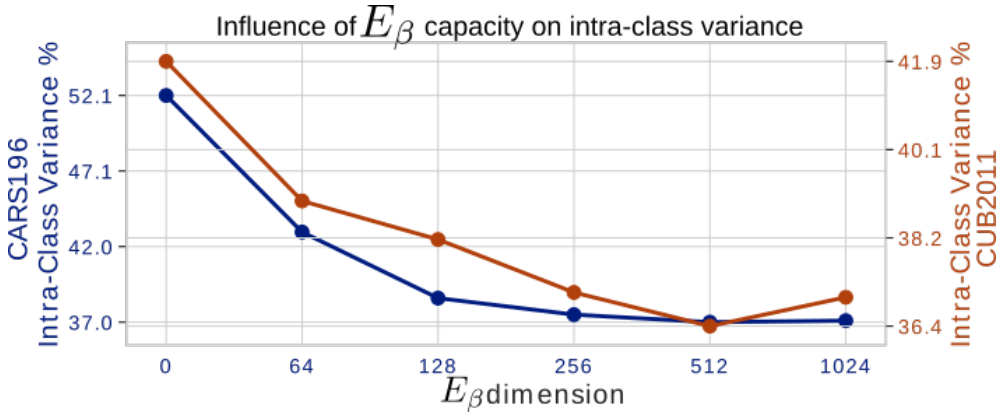

Figure 2.8: Measure of the intra-class variance in the class embedding E_α as function of the auxiliary encoder E_β dimension. The result shows that the intra-class variance decreases with an increase in E_β capacity. This points towards E_β making it easier for E_α to disregard class-independent information.

Clust	Stand	MutInfo	CARS	CUB	SOP
-	-	-	80.0	62.9	73.2
+	-	-	79.2	59.1	71.9
+	+	-	81.3	64.9	75.8
+	+	+	**82.6**	**66.1**	**77.2**

Table 2.6: Ablation study: Relevance of different contributions. Each component is crucial for reaching the best performance. (Clust: E_β training with clusters, Stand: standardization before clustering (Sec. 2.3.2), MutInfo: mutual information loss (Sec. 2.3.3))

2.5.2 Testing Components and Parameters

In order to analyze our modules, we evaluate different models, each lacking one of the proposed contribution, see tab. 2.6. The table shows how each component is needed for the best performance. Comparing to the baseline in the first line, we see that simply introducing an additional task based on clustering the data deteriorates the performance, as we add another class-discriminative training signal that introduces worse or even contradictory information. However, by utilizing standardization, we allow our second encoder to explicitly learn new features to support the class encoder instead of working against it, giving a significant performance boost. A final mutual information loss emphasises the feature separation to improve the results further.

Our approach can be combined with most existing metric learning losses, which we evaluate on ProxyNCA[112] and triplet loss with semihard sampling[143] in Tab.2.1 and 2.2. On both CARS196 and CUB200-2011, we see improved image retrieval performance.

To examine the newly introduced hyper-parameters, Fig.2.9 compares the performances on the three benchmarks using a range of cluster numbers. The plot shows how the number of clusters influences the final performances, meaning

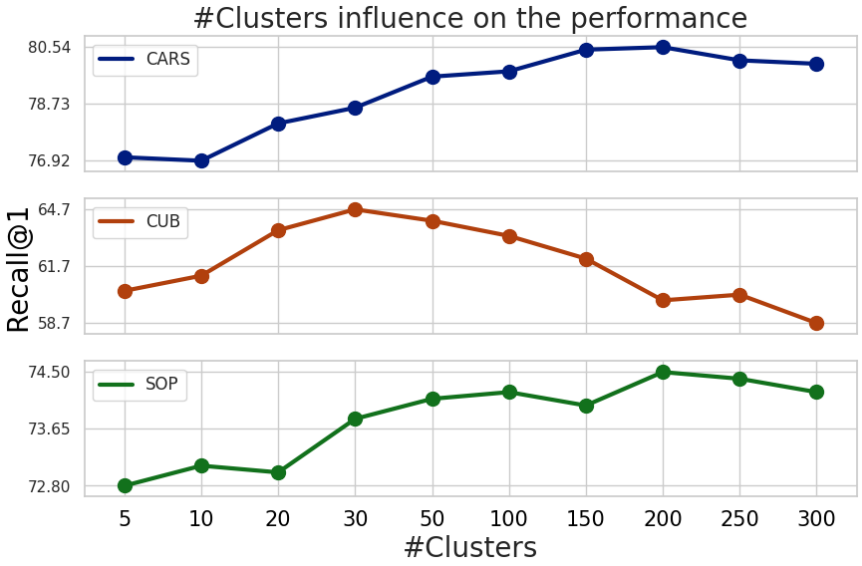

Figure 2.9: Ablation study: influence of the number of clusters on Recall@1. A fixed cluster label update period of 1 was used with equal learning rate and consistent scheduling.

the quality of the latent structure extracted by the auxiliary encoder E_β is crucial for a better classification. At the same time, an optimal performance, within a range of $\pm 1\%$ Recall@1, is reached by a large set of cluster values, making the model robust to this hyper-parameter. For these cumulative tests, a higher learning rate and less training epochs were used to both reduce computation time and avoid overfitting to the test set. Based on these examinations, we set a fixed, but dataset-dependent cluster number for all other training runs, see Sec. 2.4.1.

A similar evaluation has been performed on the update frequency for the auxiliary labels (Fig.2.10). Updating the cluster frequently clearly provides a boost to our model, suggesting that the auxiliary encoder E_β improves upon the initial clustering. However, within a reasonable range of values (between an update every 1 to 10 epochs) the model has no significant drop in performance. Thus we fix this parameter to update every two epochs for all the experiments.

2.6 Summary

In this chapter, we tackled zero-shot learning, a form of limited supervision where labels are available only for a limited number of classes, while new unseen classes appear during testing. ZSL is an example where learning visual similarity is crucial for solving the task.

Our main contribution is a novel extension for standard metric learning methods to incorporate structured intra-class information into the learning process. We do so by separating the encoding space into two distinct subspaces. One incorporates information about class-dependent characteristics, with the remaining encoder handling shared, class-independent properties. While the former is trained using standard metric learning setups, we propose a new learning task for

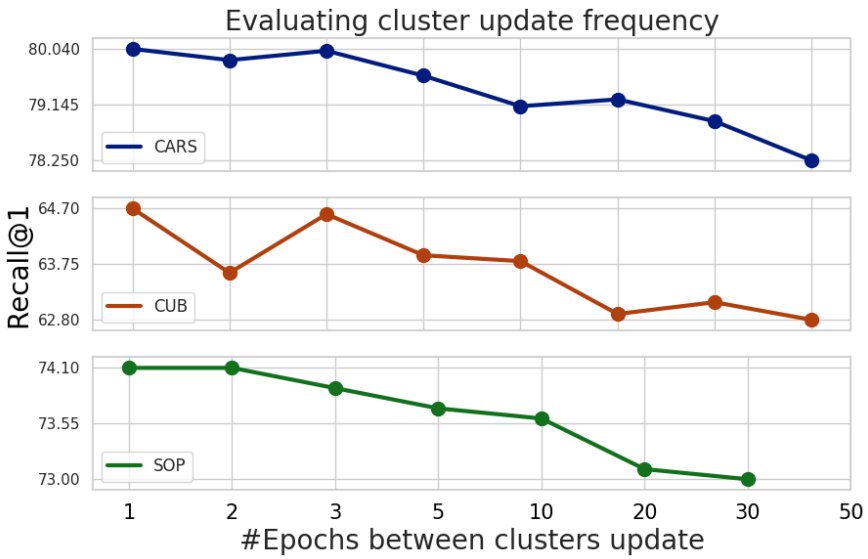

Figure 2.10: Ablation study: influence of the cluster label update frequency on Recall@1. An optimal number of clusters (see Sec. 2.4.1) and consistent scheduling was used.

the second encoder to learn shared characteristics and explain a combined training setup. Experiments on several standard image retrieval datasets show that our method consistently boosts standard approaches, outperforming the current state-of-the-art methods and reducing intra-class variance.

Chapter 3

Video Classification of Unseen Actions

As mentioned in the previous chapter, Zero-Shot Learning (ZSL) [89, 122] is a powerful paradigm of limited supervision where a single model can be trained on a set of classes and tested on any new class as long as the two domains are not too different from each other. ZSL for image classification is very popular in research given its success in several practical applications. However, video classification could also really benefit from ZSL since training a model for video is particularly costly and data are harder to annotate. Therefore, producing a single visual representation for solving video classification should be a top priority of computer vision research.

ZSL for image classification is very successful, reaching high performances on standard benchmarks. Aditionally, many real-world products are based on metric learning. However, ZSL for video recognition did not follow the same progress: state-of-the-art algorithms achieve very low performances on easy dataset and only a few new methods are published every year. In this chapter, we study existing methods for ZSL video classification to identify the reason behind this slow progress in the field. In particular, we find a lack of reproducibility to be a key weakness and propose a strong baseline that is easy to reproduce for future development. Our proposed baseline outperforms all previous methods by learning the model end-to-end for the first time in the field. Moreover, we establish a standard evaluation protocol with the aim of keeping the ZSL premises respected.

In this chapter, we learn a general visual representation for video classification in the limited supervision paradigm of ZSL. Our contributions are: a thorough analysis of existing methods and their weaknesses; a well-defined training and evaluation protocol; a strong baseline easy to reproduce for boosting future work in the field. The work done in this chapter was previously published in Brattoli et al.[22].

3.1 Background

Before diving into the method, we give a quick overview of existing work in the field of video classification, in the fully supervised and zero-shot case.

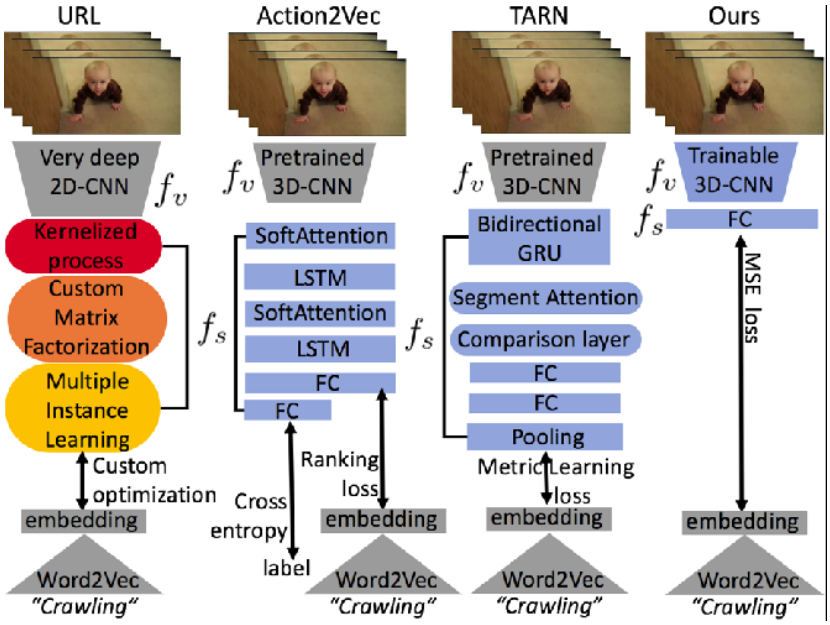

Figure 3.1: Our e2e model is simple but powerful. URL [184], Action2Vec [61] and TARN [18] are state-of-the-art approaches. Gray blocks represent modules fixed during training. Colors (blue, red, orange, yellow) indicate modules trained in separate stages.

Video classification: Existing methods can be categorized into two groups: 2D networks [146, 161] and 3D networks [25, 47, 62, 102, 153, 154]. The pioneer work of Simonyan and Zisserman [146], uses only up to 5 frames sampled randomly from the video. During testing, the visual representation is extracted from more frames and averaged over the video clip. This implied that looking at a large chunk of the video was important during inference but wasn't strictly required during training. On the contrary, Wang *et al.* [161] showed that sampling multiple frames throughout the video during training could improve performance, suggesting that temporal context is instrumental. However, modern 3D networks [25, 47, 153] showed that sampling only 16 frames, typically consecutive ones, is sufficient for achieving the best performances. Increasing training input from 16 to 128 frames improved performance only marginally.

In this chapter, we apply the basic sampling concept of state-of-the-art video classification to the ZSL problem. This enables us to train the network end-to-end, making the model more effective but also much simpler compared to previous work – as shown in Fig. 3.1.

Zero shot video classification: Differently from standard video classification, existing ZSL approaches [18, 61, 127, 177, 184] firstly extract visual features from all video frames using a pretrained network, typically C3D [153] pretrained on Sports-1M. Then, a shallow temporal model is trained to map the visual features to a semantic space [107]. Learning semantic embedding is a good proxy for generalizing to new classes that are not present in the training set. Therefore, inference reduces to finding the test class whose embedding is the nearest-neighbor of the model's output. Word2Vec [107] is commonly used to produce the ground-truth word embeddings. An alternative approach is to use manually crafted class

attributes [70]. We decided not to pursue the manual approach as it harder to apply in general scenarios.

Hahn *et al.* [61] and Bishay *et al.* [18] are two state-of-the-art methods. They extract C3D features from 52 clips of 16 frames from each video following the Protocol from Tran et al. [153]. A recurrent neural network [34, 68] encodes all frame features into a single vector, which is then mapped to the Word2Vec embedding using a fully connected layer. Fig. 3.1 illustrates this two approaches. The experiments are performed on a single dataset by splitting the classes in half for training and the other half for testing. The strength of extracting the visual features offline using a pretrained network is that most of the video frames can fit in the GPU memory during training. However, we show that this has little benefit respect to train the model end-to-end, thus extracting the visual features online.

Similarly to Zhu *et al.* [184], we also learn a generic visual representation from a large dataset and test on separate, smaller datasets. However, we also leverage the strength of modern 3D CNNs. In comparison, Zhu *et al.* [184] utilize the very deep ResNet200 [66], pretrained on still images (ImageNet [137]), ignoring the temporal context.

A major issue with the standard evaluation method is that actions are overlapping between pre-training or training data and target dataset, as pointed out by Roitberg *et al.* [135]. For example, Zhu *et al.* [184] train on the full ActivityNet [44] dataset, which has 23 classes in their training datasets that overlap with the test dataset. The situation is similar for all other methods to varying degrees.

Inductive VS Transductive: Transductive ZSL methods [4, 109, 162, 163, 172, 171, 173] use test samples during training without their annotation. We do not consider this case because a real case scenario is typically *inductive* where test data is fully unknown at training time.

Alternative approaches use generative models to compensate for the gap between semantic and visual distributions [109, 178]. Unfortunately, performance is limited by the inability to fine-tune the visual embedding. We show fine-tuning is crucial to generalize across datasets.

3.2 Motivation

Video data sourcing and annotation are particularly expensive in terms of time and monetary cost. For this reason, ZSL could be particularly beneficial. A great testbed could be a large set of human actions since they all shared a common structure: human body, motion and object interaction. Moreover, many public datasets are available for this domain [44, 59, 78, 79, 87, 149]. In case of full supervision, 3D convolutional neural networks (CNNs) proved successful [47, 153, 154]. How well modern deep networks can recognize human actions in the ZSL setting is, however, an open question.

All existing inductive methods for ZSL action recognition [4, 18, 61, 109, 127, 162, 163, 171, 172, 173, 177, 184] have a good trade-off between training efficiency and using prior knowledge since the visual representation is extracted using pretrained network offline. The algorithm only trains the mapping from visual to

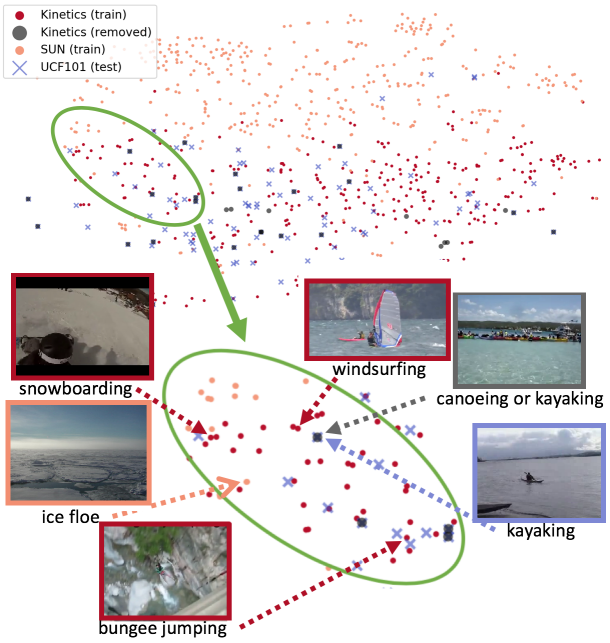

Figure 3.2: Training and test classes, t-SNE [100] visualization of Word2Vec embeddings. Red dots represent training classes we used, and gray dots training classes we removed to separate training and test data. Crosses represent test classes. Pictures are actual dataset videoframes.

semantic embedding, implemented as a shallow model. Low training space complexity of shallow models allows them to benefit from long video sequences and large feature extractors. Fig. 3.1 shows a representation of three state-of-the-art and our model.

In contrast, successful computer vision algorithms for many vision tasks, such as image classification [66], object detection [130, 132, 147] and segmentation [27, 65, 181], rely on a fully differentiable network that can be trained end-to-end (e2e). Being able to fine-tune the visual representation on the downstream task is arguably at the core of deep networks' success across machine learning domains [14]. Furthermore, training the full model increases the network capacity to store information available in large datasets [13, 66]. This poses a question: How can an e2e ZSL system compete with current methods?

We provide several contributions to the ZSL video classification problem:

Methodological: We adapt action recognition standards to ZSL developing the first e2e-trained model. Our method is simple (Fig. 3.1) and effective (Fig. 3.4), outperforming all previous work. Moreover, we provide an easy pretraining to specifically tackle ZSL weaknesses.

Testing: Following Roitberg *et al.* [135], we propose a fair ZSL evaluation that enforces a realistic ZSL setting. In particular, we train a model on a single large dataset and evaluate on multiple test datasets. The sets of training and test classes are ensured to be disjoint using an ad-hoc class distance measure.

Analysis: We analyze the ZSL problem in detail, searching for strengths and weaknesses of a typical ZSL model for human action recognition. For example, we found out that a large variety of classes is to be preferred over a

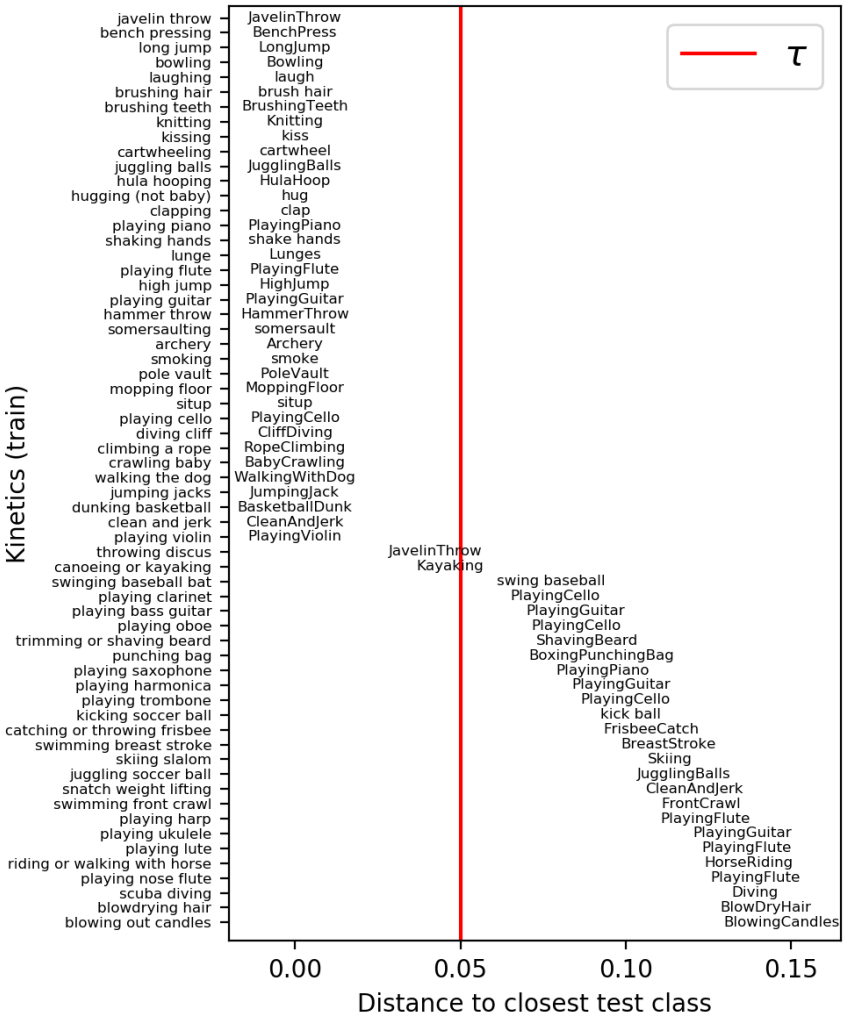

Figure 3.3: Removing overlapping training and test classes. The y-axis shows Kinetics classes closest to the test sets UCF and HMDB. x-axis shows the distance (see Eq. 3.4) of the corresponding closest test class. In our experiments, we removed training classes closer than $\tau = 0.05$ to the test set – to the left of the red line in the figure.

large number of samples.

Our model, training and evaluation code, are available at github.com/bbrattoli/ZeroShotVid

3.3 Zero-shot Action Classification

We first carefully define ZSL in the context of video classification. This will allow us to propose not only a new ZSL algorithm, but also a clear evaluation protocol that we hope will direct future research towards practical ZSL solutions. We stay within the inductive setting, as described in Sec. 3.1.

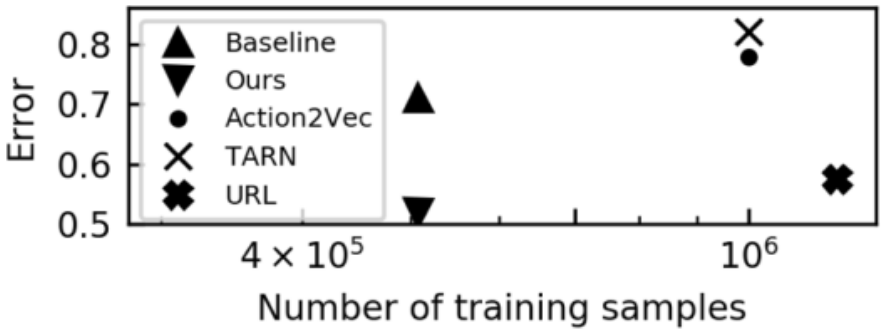

Figure 3.4: (Top) Our model is state-of-the-art (error computed on the UCF test dataset.)

3.3.1 Problem Setting

A video classification task is defined by a training set (source) $D_s = \{(x_1, c_1), \cdots, (x_{N_s}, c_{N_s})\}$ consisting of pairs of videos x and their class labels c, and a video-label test set D_t. In addition, previous work often uses pretraining datasets D_p as explained in Sec. 3.1.

Intuitively, ZSL is any procedure for training a classification model on D_s (and possibly D_p) and then testing on D_t *where* D_t *does not overlap with* $D_s \cup D_p$. How this overlap is defined varies. Sec. 3.3.3 proposes a definition that is more restrictive than those used by previous work, and forces the algorithms into a more realistic ZSL setting.

ZSL classifiers need to generalize to unseen test classes. One way to achieve this is using nearest-neighbor search in a semantic class embedding space.

Formally, given a video x, we infer the corresponding semantic embedding $z = g(x)$ and classify x as the nearest-neighbor of z in the set of embeddings of the test classes. Then, a trained classification model $M(\cdot)$ outputs

$$M(x) = \underset{c \in D_t}{\operatorname{argmin}} \cos\left(g(x), \mathrm{W2V}(c)\right). \tag{3.1}$$

where $\cos$ is the cosine distance and the semantic embedding is computed using the Word2Vec function [107] $\mathrm{W2V}\colon \mathcal{C} \to \mathbb{R}^{300}$.

The function $g = f_s \circ f_v$ is a composition of a visual encoder $f_v\colon x \mapsto y$ and a semantic encoder $f_s\colon y \mapsto z \in \mathbb{R}^{300}$.

3.3.2 End-to-end Training

In previous work, the visual embedding function f_v is either hand-crafted [173, 184] or computed by a pretrained deep network [18, 61, 163, 184]. It is fixed during optimization, forcing model development to focus on improving f_s. Resulting models need to learn to transform fixed visual embeddings into meaningful semantic features and can be very complex, as shown in Fig. 3.1 (Bottom).

Instead, we propose to optimize both f_v and f_s at the same time. Such e2e training offers multiple advantages:

1. Since f_v provides a complex computation engine, f_s can be a simple linear layer (see Fig. 3.1).

2. We can implement the full model using standard 3D CNNs.
3. Pretraining the visual embedding on a classification task is not necessary.

End-to-end optimization using the full video is unfeasible due to GPU memory limitations. Our implementation is based on standard video classification methods which are effective even when only a small snippet is used during training, as discussed in detail in Sec 3.1. Formally, given a training video/class pair $(x, c) \in D_s$ we extract a snippet x^t of 16 frames at a random time $t \leq (\text{len}(x) - 16)$. The network is optimized by minimizing the loss

$$L = \sum_{(x,c)\in D_s} \|\text{W2V}(c) - (f_s \circ f_v)(x^t)\|^2. \tag{3.2}$$

Inference procedure is similar but pools information from multiple snippets following Wang *et al.* [161]. Sec. 3.4.4 details both our training and inference procedures.

To better understand our method's performance under various experimental conditions, we implemented a baseline model that uses identical f_s, f_v and training data, but fixes f_v's weights to values pretrained on the classification task (available out-of-the-box in the most recent PyTorch implementation, see Sec. 3.4.4). This was necessary since we were not able to access implementations of any of the state-of-the-art methods ([18, 61, 184]). Unfortunately, our own re-implementations achieved results far below numbers reported by their authors, even with their assistance.

3.3.3 Towards Realistic ZSL

To ensure that our ZSL setting is realistic, we extend the methods of [135] that carefully separates training and test data. This is cumbersome to achieve in practice, and has not been attempted by most previous work. We hope that our clear formulation of the training and evaluation protocols will make it easy for future researchers to understand the performance of their models in true ZSL scenarios.

Non-overlapping training and test classes: Our first goal is to make sure that $D_s \cup D_p$ and D_t have "non-overlapping classes". The simple solution – to remove source class names from target classes or *vice-versa* – does not work, because two classes with slightly different names can easily refer to the same concept, as shown in Fig. 3.3. A distance between class names is needed. Equipped with such a metric, we can make sure training and test classes are not too similar. Formally, let $d\colon \mathcal{C} \to \mathcal{C}$ denote a distance metric on the space of all possible class names $\mathcal{C}$, and let $\tau \in \mathbb{R}$ denote a similarity threshold. A video classification task fully respects the zero-shot constraint if

$$\forall c_s \in D_s \cup D_p, \ \min_{c_t \in D_t} d(c_s, c_t) > \tau. \tag{3.3}$$

A straightforward way to define d is using semantic embeddings of class names. We define the distance between two classes to be simply

$$d(c_1, c_2) = \cos(\text{W2V}(c_1), \text{W2V}(c_2)) \tag{3.4}$$

where cos indicates cosine distance. This is consistent with the use of the cosine distance in the ZSL setting as we do in Eq. 3.1. Fig. 3.2 shows an embedding of training and test classes after we removed from Kinetics classes overlapping with test data using the procedure outlined above. Fig. 3.3 shows the distribution of distances between training and test classes in our datasets. There is a cliff between distances very close to 0 and larger than 0.1. In our expeirments we use $\tau = 0.05$ as a natural, unbiased threshold.

Different training and test video domains: We argue that video domains of $D_s \cup D_p$ and D_t should differ. In previous work, the standard evaluation protocol is to use one dataset for training and testing, using 10 random splits. This does not account for domain shifts that happen in real world scenarios due to data compression, camera artefacts, and so on. For this reason ZSL training and test datasets should ideally have disjoint video sources.

Multiple test datasets: A single ZSL model should perform well on multiple test datasets. As outlined above, previous works train and test anew for each available dataset (typically UCF and HMDB). In our experiments, training happens only once on the Kinetics dataset [79], and testing on all of UCF [149], HMDB [87] and ActivityNet [44].

3.3.4 Easy Pretraining for Video ZSL

In a real-world scenario a model is trained once and then deployed on diverse unseen test datasets. A large and diverse training dataset is crucial to achieve good performance. Ideally, the training dataset would be tailored to the general domain of inference – for example, a strong ZSL surveillance model to be deployed at multiple unknown locations would require a large surveillance and action recognition dataset.

Sourcing and labeling domain-specific video datasets is, however, very expensive. On the other hand, annotating images is considerably faster. Therefore, we designed a simple dataset augmentation scheme which creates synthetic training videos from still images. Sec. 3.5 shows that pretraining our model using this dataset boosts performance, especially if available training data is small.

We convert images to videos using the Ken Burns effect: a sequence of crops moving around the image simulates video-like motion. Sec. 3.4.1 provides more details.

Our experiments focus on the action recognition domain. In action recognition (as well as in many other classification tasks), location and scenery of the video is strongly predictive of action category. Because of this we choose SUN [168], a standard scene recognition dataset. Fig. 3.2 shows the complete class embedding of our the scene dataset's class names.

3.4 Experimental Setup

To facilitate reproducibility, we describe our training and evaluation protocols in detail. The protocols propose one way of training and evaluating ZSL models that is consistent with our definitions in Sec. 3.3.3.

3.4.1 Datasets

UCF101 [149] has 101 action classes primarily focused around sports, with 13320 videos sourced from YouTube. *HMDB51* [87] is divided into 51 human actions focused around sports and daily activities and contains 6767 videos sourced from commercial videos and YouTube. *ActivityNet* [44] contains 27,801 untrimmed videos divided in 200 classes focusing on daily activities with videos sourced using web search. We extracted only the labeled frames from each video. *Kinetics* [79] is the largest currently available action recognition dataset, covering a wide range of human activity. The first version of the dataset contains over 200K videos divided in 400 categories. The newest version has 700 classes for a total of 541624 videos sourced from YouTube. *SUN397* [168] (see Sec. 3.3.4) is a scene understanding image dataset. It contains 397 scene categories for a total of over 100K high-resolution images. We converted it to a simulated video dataset using the Ken Burns effect: To create a 16-frame video from an image, we randomly choose "start" and "end" crop locations (and crop sizes) in the image, and linearly interpolate to obtain 16 crops. Each of them are then resized to 112×112.

3.4.2 Training Protocol

Our experiments in Sec. 3.5 use two training methods:

Training Protocol 1: Remove from Kinetics 700 all the classes whose distance to any class in UCF $\cup$ HMDB is smaller than τ (see Eq. 3.4). This results in a subset of Kinetics with 664 classes, which we call Kinetics 664. As explained in Sec. 3.3.3, this setting is already more restrictive than that of the previous methods, which train new models for each test dataset.

Training Protocol 2: Remove from Kinetics 700 all the classes whose distance to any class in UCF $\cup$ HMDB $\cup$ *ActivityNet* is smaller than τ (see Eq. 3.4). This results in a subset of Kinetics with 605 classes which we call Kinetics 605. This setting is even more restrictive, but is closer to true ZSL. Our goal is to show that it is possible to train a single ZSL model that applies to multiple diverse test datasets.

Figure 3.2 shows a t-SNE projection of the semantic embeddings of all Kinetics 700 classes, as well as the 101 UCF classes and the classes we removed to obtain Kinetics 664.

3.4.3 Evaluation Protocol

We tested our model using two protocols: the first follows Sec. 3.3.3 to emulate a true ZSL setting, the second is compatible with previous work. Both Evaluation Protocols apply the same model to multiple test datasets.

Evaluation Protocol 1: In order to make our results comparable with previous work, we use the following procedure: Randomly choose half of the test dataset's classes, 50 for UCF and 25 for HMDB. Evaluate the classifier on that test set. Repeat ten times and average the results for each test dataset.

Evaluation Protocol 2: Previous work uses random training/test splits of UCF [149] and HMDB [87] to evaluate their algorithms. However, we train on a separate dataset Kinetics 664 / 605 and can test on full UCF and HMDB. This

Dataset	VisualFeat	UCF	HMDB	Activity
URL [184]	ResNet200	42.5	51.8	-
DataAug [173]	-	18.3	19.7	-
InfDem [134]	I3D	17.8	21.3	-
Bidirectional [163]	IDT	21.4	18.9	-
FairZSL [135]	-	-	23.1	-
TARN [18]	C3D	19	19.5	-
Action2Vec [61]	C3D	22.1	23.5	-
Ours(605classes)	C3D	41.5	25.0	24.8
Ours(664classes)	C3D	43.8	24.7	-
Ours(605classes)	R(2+1)D_18	44.1	29.8	**26.6**
Ours(664classes)	R(2+1)D_18	**48**	**32.7**	-

Table 3.1: Comparison with the state-of-the-art on standard benchmarks. We evaluate on half test classes following Evaluation Protocol 1 (Sec. 3.4.3). Ours(605classes) indicates we removed all training classes that overlap with UCF, HMDB, or ActivityNet. Ours(664classes) indicates we removed only training classes overlapping with UCF and HMDB. We outperform previous work in both scenarios. Sec. 3.1 argues that URL's results are not compatible with other works as their training and test sets overlap and their VisualFeat is an order of magnitude deeper.

allows us to return more realistic accuracy scores. The evaluation protocol is simple: evaluate the classifier on all 101 UCF classes and all 51 HMDB classes.

3.4.4 Implementation Details

In our experiments, f_v (see Sec. 3.3.1) is the PyTorch implementation of R(2+1)D_18 [154] or C3D[153]. In the pretrained setting, we use the out-of-the-box R(2+1)D_18 pretrained on Kinetics 400[79], while C3D is pretrained on Sports-1M[78]. In the e2e setting, we initialize the model with the pretrained=False argument. The visual embedding $f_v(x)$ is BxTx512 where B is the batch size and T is the number of clips per video. We use $T = 1$ for training, and $T = 25$ for evaluation in Tables 3.1 and 3.2. The clips are 16 frames long and we choose them following standard protocols established by Wang *et al.* [161]. We average $f_v(x)$ across time (video snippets) similarly to previous approaches [153, 184]. f_s is a linear classifier with 512x300 weights. The output of $f_s \circ f_v$ is of shape Bx300.

We follow standard protocol in computing semantic embeddings of class names [18, 171, 184]. Word2Vec [107] – in particular, the gesim [131] Python implementation – encodes each word. We average multi-word class names. In rare cases of words not available in the pretrained W2V model (for example, 'rubiks' or 'photobombing') we manually change the words (see the code for more details). Formally, for a class name consisting of N words $c = [c^1, \cdots, c^N]$, we embed it as $\mathrm{W2V}(c) = \sum_{i=1}^{N} \mathrm{W2V}(c^i) \in \mathbb{R}^{300}$. We set τ to 0.05 following the analysis in Sec. 3.3.3 based on Fig. 3.3.

To minimize the loss of Eq. 3.2 we use the Adam optimizer [80], starting with

Method	UCF		HMDB		Activity	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
URL [184]	34.2	-	-	-	-	-
664classes	**37.6**	**62.5**	**26.9**	**49.8**	-	-
605classes	35.3	60.6	24.8	44.0	**20.0**	**42.7**

Table 3.2: Evaluation on all test classes. In contrast to Table 3.1, here we report results of our method applied to all three test datasets using Evaluation Protocol 2 (Sec. 3.4.3). We applied a single model trained on classes dissimilar from all of UCF, HMDB and ActivityNet. Nevertheless, we outperform URL [184] on UCF101. URL authors do not report results on full HMDB51. Remaining previous work do not report results on neither full UCF101 nor full HMDB51.

a learning rate of $1e-3$. Batch size is 22 snippets, with 16 frames each. The model trained for 150 epochs, with a tenfold learning rate decrease at epochs 60 and 120. All experiments are performed on the Nvidia Tesla V100 GPU.

Following [153], we reshaped each frame's shortest side to 128 pixels, and cropped a random 112x112 patch on training and the center patch on inference.

3.5 Results

Our experiments have two goals: compare our method to previous work and investigate our method's performance vs the baseline (see Sec. 3.3.2.) The first is necessary to validate that e2e ZSL on videos can outperform more complex approaches that use pretrained features. The latter will allow us to understand under what conditions e2e training can be particularly beneficial.

3.5.1 Comparison to the State of the Art

Table 3.1 compares our method to existing approaches. We followed our Training and Evaluation Protocol 1, as described in Sections 3.4.2 and 3.4.3. Our protocols are more restrictive than that of previous methods: we removed training classes that overlap with test classes, introduced domain shift, and applied one model to multiple test datasets. Despite this, we outperform previous video-based methods by a large margin. Furthermore, when testing on UCF we outperform URL [184] which uses a network an order of magnitude deeper than ours – 18 vs 200 layers – and 23 classes overlap between training and testing (see Sec. 3.1).

3.5.2 Comparison to a Baseline Method

Our baseline method described in Sec. 3.3.2 uses a fixed, pretrained visual feature extractor but is otherwise identical to our e2e method. This allows us to study the benefits of e2e training under Evaluation Protocol 2, (see Sections 3.4.2 and 3.4.3). Using all test classes provides a more direct evaluaition of the method.

Training dataset size: To investigate the effect of training set size on performance we subsampled Kinetics 664 uniformly at random, then re-trained and

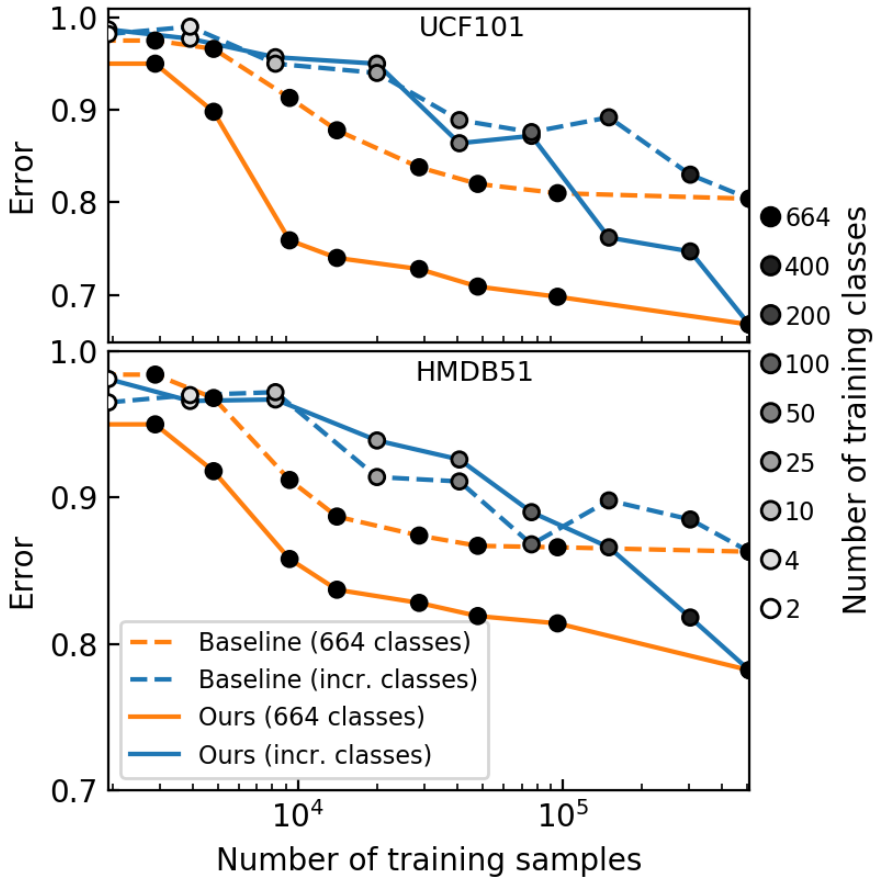

Figure 3.5: Number of training classes matters in ZSL. Orange curves show performance on subsets of Kinetics 664, as we keep all the training classes and increase the subset size. The blue curves, whose markers become progressively brighter, indicate a separate experiment where we increased the number of training classes starting from 2, all the way up to 664 (Sec. 3.5.2). For any given training dataset size, performance on test data is much better with more training classes. In addition, when few training classes are available the e2e model is not able to outperform the baseline.

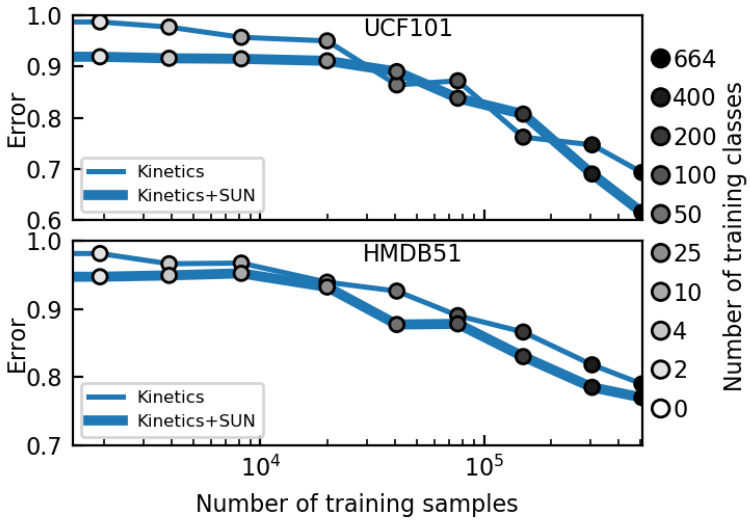

Figure 3.6: Augmented pretraining with videos-from-images. We trained our algorithm on progressively smaller subsets of Kinetics 664 classes (Sec. 3.5.2). We compared the results to training on the same dataset, after pretraining the model on our synthetic SUN video dataset (Sec. 3.5.3). The pretraining procedure boosts performance up to 10% points.

re-evaluated the model. Fig. 3.5 shows that the e2e algorithm consistently outperforms the baseline on both datasets. Both algorithms' performance is worse with smaller training data. However, the baseline flattens out at about 100K training datapoints, whereas our method's error keeps decreasing. This is expected, as the e2e model has more capacity.

Number of training classes: In many video domains diverse data is difficult to obtain. Small datasets might not only have few datapoints, but also contain only a few training classes. We show that the number of training classes can impact ZSL results as much as training dataset size.

To obtain Fig. 3.5 we subsampled Kinetics 664 class-wise. We first picked 2 Kinetics 664 classes at random, and trained the algorithm on those classes only. We repeated the procedure using 4, 10, 25, 50, 100, 200, 400 and all 664 classes. Naturally, the fewer classes the fewer datapoints the training set contained. This results are compared in Fig. 3.5 with the procedure described above, where we removed Kinetics datapoints at random – independent of their classes.

The figure shows that it is better to have few training samples from a large number of classes rather than many from a very small number of classes. This effect is more pronounced for the e2e model rather than the baseline.

Training dataset class diversity: We showed that ZSL works better with more training classes. If we have a limited budget for collecting classes and datapoints, how should we choose them? We investigated whether the set of training classes should emphasize fine differences (e.g. "shooting basketball" vs "passing basketball" vs "shooting soccerball" and so on) or diversity.

In Fig. 3.7 we selected 50 training classes in four ways: (Top Left) We randomly choose 50 classes from the whole Kinetics 664 dataset, trained the algorithm on these classes, and ran inference on the test set. We repeated this process

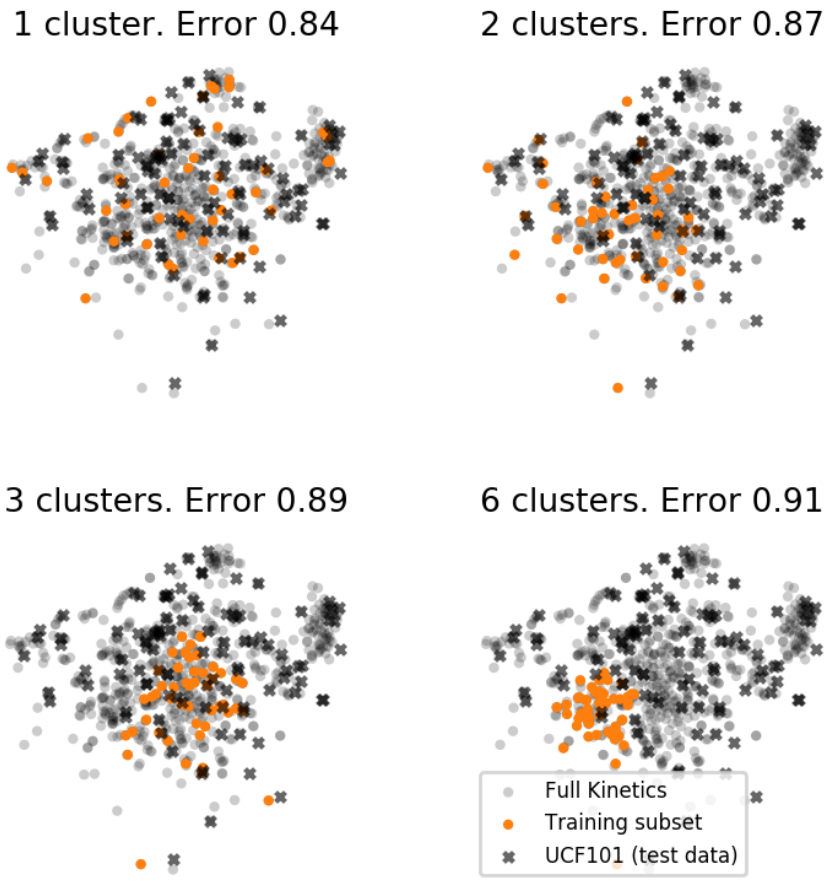

Figure 3.7: Diverse training classes are good for ZSL. Here we trained our algorithm on subsets of 50 Kinetics 664 classes. (Top left) Training classes picked uniformly at random. (Top right) We clustered Word2Vec embeddings of classes into two clusters, then trained and evaluated separately using each cluster, and averaged the results. (Bottom) Here we averaged the results of training using three and six clusters. The figure shows that the more clusters, the less diverse the training classes were semantically. At the same time, less diversity caused higher errors.

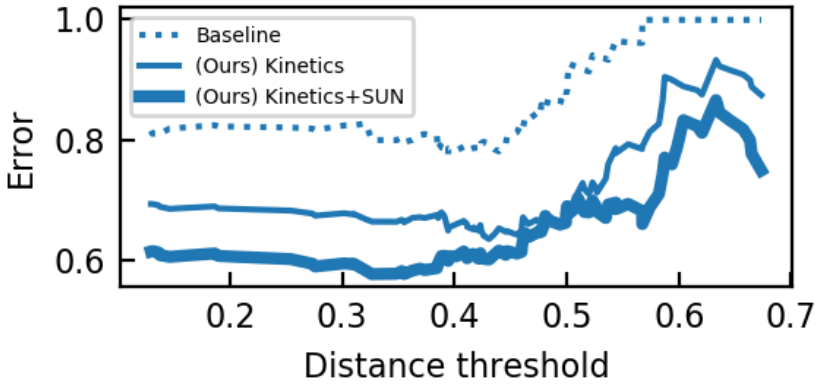

Figure 3.8: Error as test classes move away from training. For each UCF101 test class, we computed its distance to 10 nearest neighbors in the training dataset. We arranged all such distance thresholds on the x-axis. For each threshold, we computed the accuracy of the algorithms *on test classes whose distance from training data is larger than the threshold*. In other words, as x-axis moves to the right, the model is evaluated on cumulatively smaller, but harder test sets.

ten times and averaged inference error. (Top Right) We clustered the 664 classes into 2 clusters in the Word2Vec embedding space, and chose 50 classes at random within one of the clusters, trained and ran inference. We then repeated the procedure ten times and averaged the result. (Bottom) Here we chose 50 classes in one of 3 clusters (Left) and one of 6 clusters (Right), trained, and averaged inference results of 10 runs. The figure shows that test error for our method increases as class diversity decreases. This result is not obvious, since the task becomes harder with increasing class diversity.

3.5.3 Easy Pretraining with Images

Previous section showed that class count and diversity are important drivers of ZSL performance. This inspired us to develop the pretraining method described in Sec. 3.3.4: we pretrain our model on a synthetic video dataset created from still images from the SUN dataset. Fig. 3.6 shows that this simple procedure consistently decreases test errors by up to 10%. In addition, Fig. 3.8 shows that this initialization scheme makes the model more robust to large domain shift between train and test classes. The following section describes the latter finding in more detail.

3.5.4 Generalization and Domain Shift

A good ZSL model will generalize well to test classes that differ significantly from training classes. To investigate the performance of our models under heavy domain shift, we computed their accuracy on subsets of test data with growing distance from the training dataset. We first trained our model on Kinetics 664. Then, for a given semantic distance threshold τ (see Sec. 3.3.3), we computed accuracy of the model on the set of UCF classes whose mean distance from the

UCF101 accuracy			50 classes		101 classes	
e2e	Augment	Multi	Top-1	Top-5	Top-1	Top-5
			26.8	55.5	19.8	40.5
✓			43.0	68.2	35.1	56.4
✓	✓		45.6	73.1	36.8	61.7
✓		✓	48.0	74.2	37.6	62.5
✓	✓	✓	**49.2**	**77.0**	**39.8**	**65.6**

Table 3.3: Ablation study. Numbers represent classification accuracy. "50 classes" uses Evaluation Protocol 1 (Sec. 3.4.3.) "101 classes" uses Evaluation Protocol 2. e2e: training the visual embedding as opposed to fixed, pretrained baseline (Sec. 3.3.2). Augment: pretrain using the SUN augmentation scheme (Sec. 3.5.3). Multi: At test time, extract multiple snippets from each video and average the visual embeddings (Sec. 3.4.4).

closest 10 Kinetics 664 classes is larger than τ. Fig. 3.8 shows that the baseline model's (pretrained on a large dataset but not trained e2e) performance drops to zero at around $\tau \sim 0.57$. Our e2e method performs much better, never dropping to zero accuracy for high thresholds. Finally, using the SUN pretraining method further increases the model's robustness.

3.5.5 Ablation Study

Table 3.3 studies contributions of different elements of our model to its performance. The performance is low when the visual embedding is fixed. The e2e approach improves the performance by a large margin. Our class augmentation method further boosts performance. Finally it helps to extract linearly spaced snippets from a video on testing, and average their visual embeddings. Using 25 snippets improves considerably the performances without influencing the training time of the model.

3.5.6 Backbone Choice

Tab. 3.4 compares the accuracy of three 3D convolutional backbones on two kinetics versions using our Training Protocol 1 (Sec. 3.4.2). For this comparison we also tried using the full Kinetics 400/700 datasets, without removing overlapping test classes. The table shows that adding the 6% of the training classes most overlapping with the test set yields an unexpected >40% accuracy boost for UCF and 25% on HMDB. This proves that the zero-shot learning constraint is non-trivial.

			UCF		HMDB	
Network	Train	classes	50	101	25	51
C3D	K400	361	33.7	25.7	17.0	13.3
R3D_18	K400	361	37.2	29.0	20.4	16.8
R(2+1)D_18	K400	361	38.7	30.6	22.0	18.1
C3D	K700	664	40.3	33.1	22	17.0
R3D_18	K700	664	41.2	34.2	23.6	19.0
R(2+1)D_18	K700	664	**43.0**	**35.0**	**25.8**	**20.6**
R(2+1)D_18	K400	400	50.1	44.5	27.2	22.5
R(2+1)D_18	K700	700	54.6	49.7	30.5	25.6

Table 3.4: Accuracy of different backbone architectures trained on the first (K400) and last (K700) version of Kinetics [19]. The models are evaluated on a single clip (16 frames).

3.6 Analysis

3.6.1 SUN Pretraining: Easier Task or Better Representation?

Sec. 3.3.4 shows that pretraining on a scenes dataset (SUN397) improves ZSL performance. In this section, we ask whether the boost is due to better model generalization or simply because the source domain becomes closer to the target domain.

Per each UCF101 test class, Fig. 3.9 shows the W2V distance to Kinetics train classes as well as (Kinetics + SUN) train classes. Test classes that got more than 10% closer to training data are marked in color. The right subplot, however, shows that the model trained on (Kinetics + SUN) boosts the accuracy of many classes – in particular, the accuracy of many classes that are *not* among the colored ones rose significantly. The model pretrained on SUN data increases performance on many classes which are not close to SUN data. We conclude that pretraining on SUN allows the model to generalize better over almost all test classes, not only the ones close to SUN data.

3.7 Training Class Diversity

We expand the analysis of Sec. 3.5.2 and Fig. 3.7, by testing the influence of training class diversity on both UCF and HMDB. Fig. 3.10 correlates model performance with training class density. For this experiment, we selected 50 train classes with different density in the Word2Vec space, using the same clustering approach we used in Sec. 3.5.2. Per each diversity value, we select 50 classes and train a model multiple times to compute the standard deviation. Fig. 3.10 shows that test error decreases as training classes become more diverse. At the same time, the standard deviation decreases, indicating that for compact classes, the

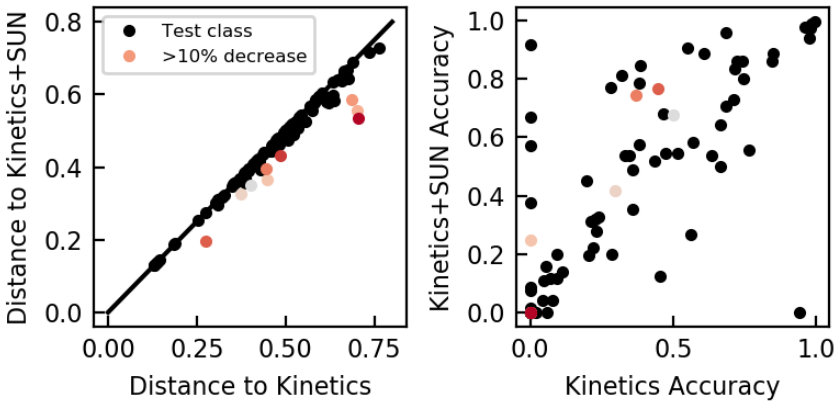

Figure 3.9: Each dot represents a UCF101 test class. Test class accuracy (right) and distance (left) to the train set (Kinetics664) for two models: one with random initialization and one pretrained on SUN (see Sec 3.6). A colored dot indicates a test class that reduces its distance to the train set by more than 10% when SUN is included on training.

performance highly depends on where in the class space we sample the classes, which is something we only know once the test set is available.

This outcome is not obvious, since we might expect the task to become harder when class variance increases (given the same number of training datapoints). However, we do not observe decrease in performance. Therefore, we can conclude that the model can only benefit from a high variety within the train class distribution. This new insight can be useful during training dataset collection.

3.8 Analyze the Model Capability Action per Action

What does better or worse accuracy indicate for specific classes? We break down the change in performance between models for each UCF101 test class.

3.8.1 Direct Comparison by Sorting Classes

In Sec. 3.5.2, we evaluated the model using error aggregated over all the test classes. It is also interesting to know whether the network is getting better at recognizing specific classes, or improves across the board?

Fig. 3.11 shows the accuracy on each UCF test class for three models: baseline, e2e trained on Kinetics, and e2e pretrained on SUN397 and then trained on Kinetics. We sorted the classes from hardest to easiest for each model. Fig. 3.12 shows the same information, zoomed in on worst and best actions only. The two plots show that some of the actions which are difficult for the baseline model are correctly classified by our e2e models. On the other hand, the inverse situation is rare. In addition, the actions which are correctly classified by the baseline are also easily identified by our models.

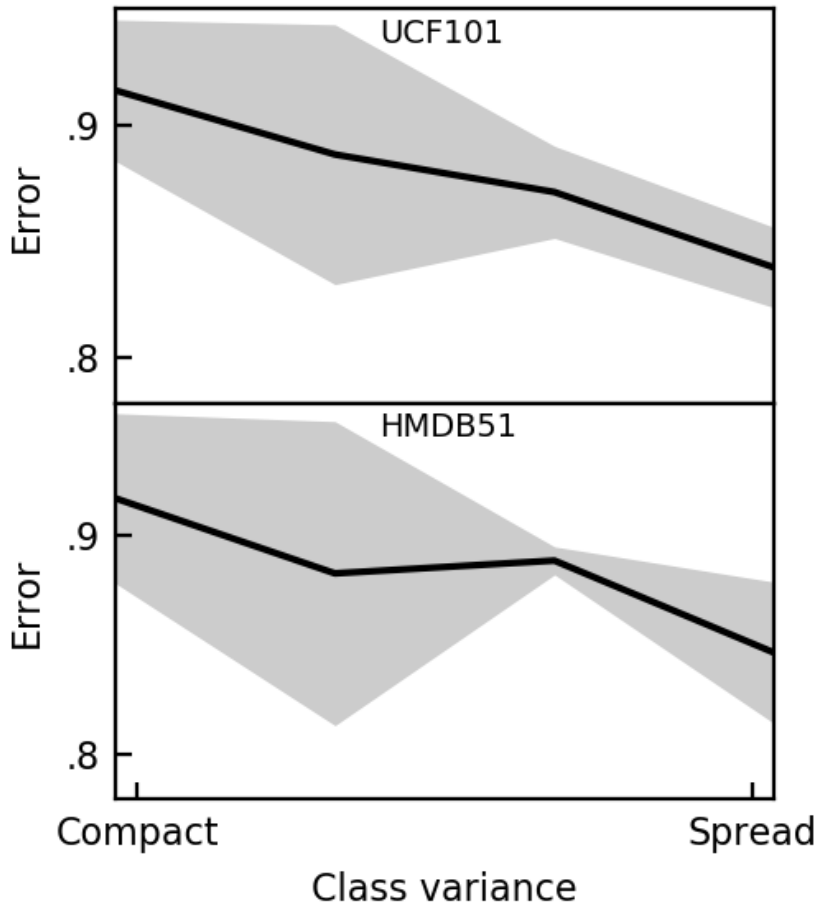

Figure 3.10: Performance of the e2e model trained on 50 Kinetic664 classes and tested on UCF and HMDB. The 50 classes are chosen based on diversity in their W2V embedding (see Fig. 3.7, for details). The more semantically diverse the training classes, the lower the error.

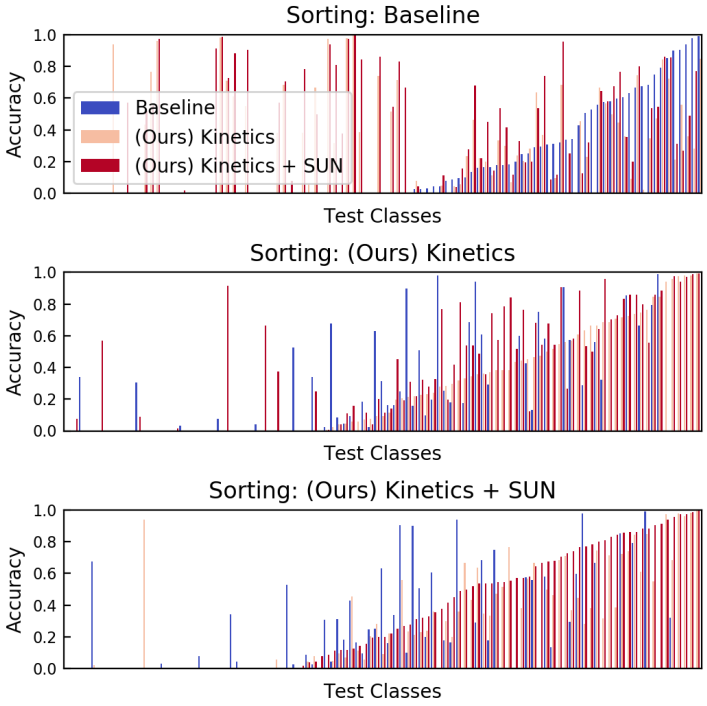

Figure 3.11: Accuracy on each UCF101 test class, for three models. Each subplot uses different model's accuracies to sort the classes, otherwise the numbers are the same.

In addition, the results of e2e trained on Kinetics and e2e pretrained on SUN and trained on Kinetics are highly correlated, but the second achieves overall better performances. This suggests that SUN provides *complementary* information to Kinetics which are useful for the target task. On the other hand, the baseline is less correlated with the e2e results, suggesting that the fixed visual features have a lot to learn and *should be fine-tuned*.

3.9 Summary

In this chapter, we studied the limited supervision task ZSL for video classification. We provide a method that uses semantic information from a natural language model to learn visual similarities between actions, crucial for generalizing to unseen actions. Our contributions are several: we providing a strong baseline, studied the ZSL problem in detail for the field of human actions and defined a better evaluation protocol.

Our baseline is developed by following practices from recent video classification literature. We train the first e2e system for video recognition ZSL. Our evaluation protocol is stricter than that of existing work and measures more realistic zero-shot classification accuracy. Even under this stricter protocol, our method outperforms previous works whose performance was measured with training and

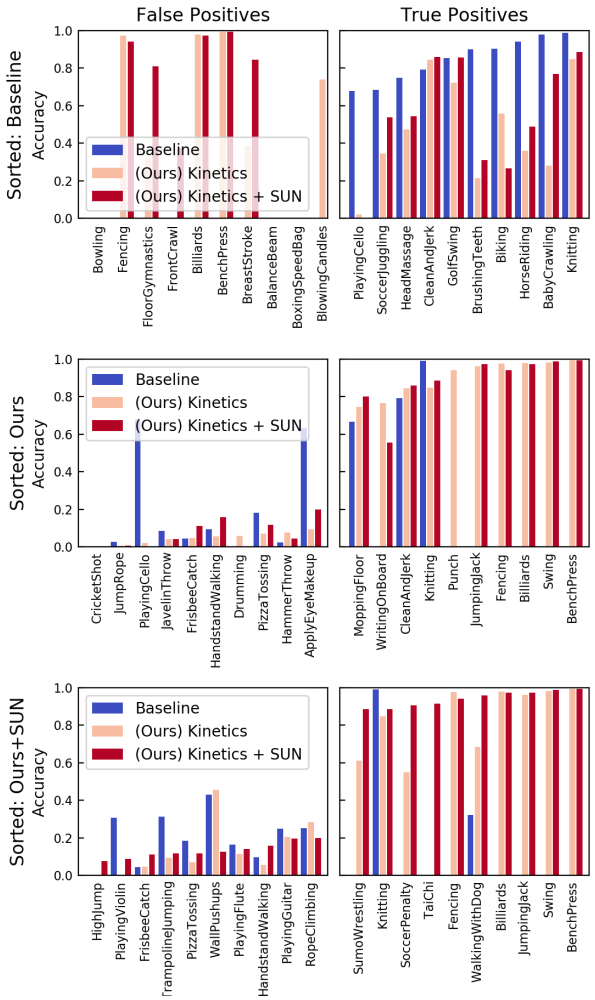

Figure 3.12: Accuracy on best and worst 10 classes for each model.

test sets overlapping and sharing domains. Through a series of directed experiments, we showed that class diversity is crucial for good generalization.Guided by this insight, we formulated a simple pretraining technique that boosts ZSL performance.

Our model is easy to understand and extend. Our training and evaluation protocols are easy to use with alternative approaches. We made our code available github.com/bbrattoli/ZeroShotVideoClassification, encouraging the community to build on our insights and create a strong foundation for future video ZSL research.

Chapter 4

Improving Self-Supervision for Better Visual Representation

In the last two chapters, we produced a visual representation that can generalize from training to unseen classes. In this chapter, we will go one step further and develop an unsupervised visual representation, i.e. without using human annotation.In particular, we focus on the paradigm of self-supervision which uses a surrogate task where labels are automatically available. More in detail, the common approach is to transform the input data and ask the network to recognize the applied transformation. To solve the task, the network needs to extract crucial semantic information from the data, thus learning a useful visual representation that can be transferred to new tasks.

Self-supervision is a well-established paradigm with a large collection of methods. Therefore, we do not add one more method to the pile, but propose a contribution on a meta-level. Every surrogate task has a free parameter which determines how to transform the input sample. Typically, this parameter is chosen randomly for each sample during training. Instead, we could control it to optimize the training. The assumption is that a specific transformation is more effective on certain data. Base on this assumption, we develop an automatic controller trained using reinforcement learning.

More details on this will be provided in Sec 4.3, but first we provide a short collection of related work on the relevant topics in Sec 4.1. This work was previously published in Buechler *et al.* [23].

4.1 Background: Meta-learning

Curriculum Learning: In 2009 Bengio *et al.* [15] proposed curriculum learning (CL) to enhance the learning process by gradually increasing the complexity of the task during training. CL has been utilized by different deep learning methods [60, 150, 26] with the limitation that the complexity of samples and their scheduling during training typically has to be established a priori. Kumar et. al [88] define the sample complexity from the perspective of the classifier, but still manually define the scheduling. In contrast, our policy dynamically selects the permutations based on the current state of the network.

Meta-Learning for Deep Neural Networks: Recently, methods have proposed ways to improve upon the classical training of neural networks by, for example, automatizing the selection of hyper-parameters [7, 129, 186, 46, 120]. Andrychowicz *et al.* [7] train a recurrent neural network acting as an optimizer which makes informative decisions based on the state of the network. Fan *et al.* [46] propose a system to improve the final performance of the network using a reinforcement learning approach which schedules training samples during learning. Opitz *et al.* [120] use the gradient of the last layer for selecting uncorrelated samples to improve performance. Similar to [7, 46, 120] we propose a method which affects the training of a network to push towards better performances. In contrast to these supervised methods, where the image labels are fixed, our policy has substantial control on the training of the main network since it can directly alter the input data by proposing permutations.

4.2 Self-Supervision

Convolutional neural networks (CNNs) have demonstrated to learn powerful visual representations from large amounts of tediously labeled training data [85]. However, since visual data is cheap to acquire but costly to label, there has recently been great interest in learning compelling features from unlabeled data. Without any annotations, self-supervision based on surrogate tasks, for which the target value can be obtained automatically, is commonly pursued [110, 93, 49, 11, 114, 38, 108, 115, 99, 124, 90, 30, 55, 140]. In colorization [90], for instance, the color information is stripped from an image and serves as the target value, which has to be recovered. Various surrogate tasks have been proposed, including predicting a sequence of basic motions [99], counting parts within regions [115] or embedding images into text topic spaces [124]. In contrast to the majority of recent self-supervised learning approaches, Doersch *et al.* [39] and Wang *et al.* [165] combine surrogate tasks to train a multi-task network. Doersch *et al.* [39] choose 4 surrogate tasks and evaluate a naive and a mediated combination of those. Wang *et al.* [165], besides a naive multi-task combination of these self-supervision tasks, use the learned features to build a graph of semantically similar objects, which is then used to train a triplet loss. Since they combine heterogeneous tasks, both methods use an additional technique on top of the self-supervised training to exploit the full potential of their approach.

The key competence of visual understanding is to recognize structure in visual data. Thus, breaking the order of visual patterns and training a network to recover the structure provides a rich training signal. This general framework of permuting the input data and learning a feature representation, from which the inverse permutation (and thus the correct order) can be inferred, is a widely applicable strategy. It has been pursued on still images [114, 116, 38, 30, 39] by employing spatial shuffling of images (especially permuting jigsaws) and in videos [110, 93, 49, 21] by utilizing temporally shuffled sequences. Buechler *et al.* [23] proposes that since spatial and temporal shuffling are both ordering tasks, which only differ in the ordering dimension, they should be addressed jointly. Since this

is a more complete method, addressing both image and video tasks, we will use this as our baseline. More details will be given in Sec. 4.2.1.

4.2.1 Baseline: Spatiotemporal Jigsaw Puzzle

Our goal is to control a surrogate task to optimize the training. Therefore, we need to choose one of the existing self-supervised methods and build our controller on top of that. We chose Buechler *et al.* [23] which learns a general visual representation for images and video following the paradigm of Jigsaw puzzle. In this section, we describe in details the method.

The visual representation is trained using a CNN backbone (CaffeNet [75] architecture up to pool5) initialized from scratch (see Fig. 4.1C). The spatial component receives in input an image divided in $m \times m$ regular grid of tiles as suggested by [114](Fig. 4.1B top). A video sequence composed of u frames is shuffled temporally and given as input to the temporal network.

To avoid redundancy in the formulation, we will refer to a generic input $x = (x_1, x_2, \dots)$ for both a sequence of frames (temporal task) or image tiles (spatial task). Given an index permutation $\psi_i = (\psi_{i,1}, \psi_{i,2}, \cdots)$, we define the shuffled input as

$$\psi_i(x) := \left(x_{\psi_{i,1}}, x_{\psi_{i,2}}, \dots\right). \tag{4.1}$$

The set of all possible permutations $\Psi^\star$ is limited for practical reasons using the heuristic proposed by Noroozi and Favaro [114], by sampling a set $\Psi \subset \Psi^\star$ of maximally diverse permutations $\psi_i \in \Psi$. We iteratively include the permutation with the maximum Hamming distance $d(\bullet, \bullet)$ to the already chosen ones. The set of permutations is different for the spatial and temporal tasks since we use less permutations for the latter. For simplicity, we are going to explain our approach based on a general Ψ without referring to a specific task.

On top of the visual representation (CNN backbone, Fig. 4.1(C)) we need a classifier that actually solves the Jigsaw task by identifying the permutation applied to the input. The output of the Pool5 is flattened and given to a fully connected layer of size 1024, typically referred to as fc6. We obtain one fc6 output for each input part with a total size of $u \times 1024$. For the spatial network, the parts are stacked $1 \times u * 1024$ and passed to another fully connected layer (fc7); for the temporal network, a recurrent neural network (LSTM [68]) summarizes the parts into a single output 1×512 (see Fig. 4.1(C) and Sect. 4.4 for implementation details). The output of fc7 or the LSTM is then processed by a final fully connected layer which estimates the permutation ψ_i applied to the input sample. Using a limited number of permutation and assign a categorical number to each of them, allow us to train the network end-to-end using a standard cross-entropy loss. The output activation $\varphi_i, i \in \{1, \dots |\Psi|\}$ of the classifier corresponds to the permutation $\psi_i \in \Psi$. Given that the output activation value of a classifier indicates how certain the network is on the prediction, this can be important information for the selection of the permutation. We will follow up on this in Sec 4.3.

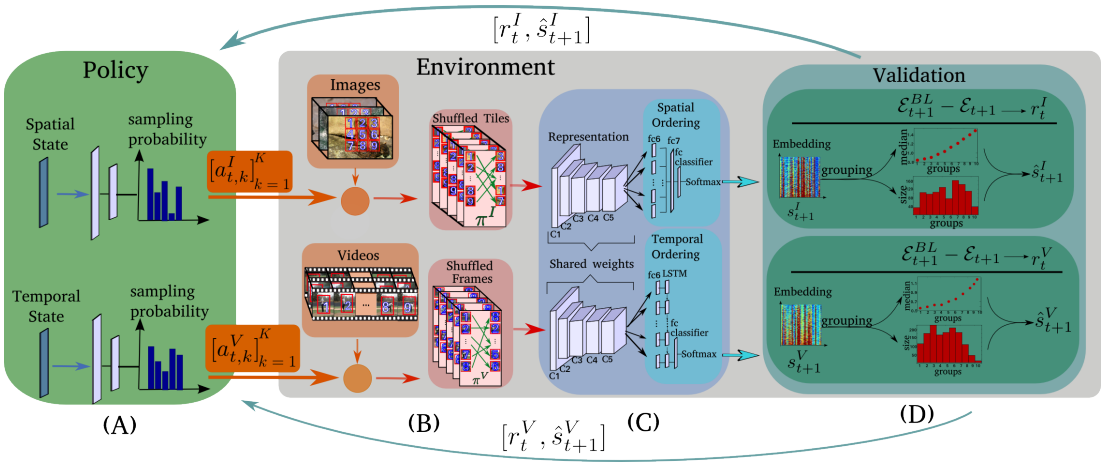

Figure 4.1: (A) Deep RL of a policy for sampling permutations. (B) Permuting training images/videos by the proposed actions of (A) to provide self-supervision for our network architecture (C). (D) Evaluating the update network (C) on validation data to receive reward and state.

The spatial and temporal components are trained in parallel: each task produces a gradient which are then averaged and back-propagated through the entire network, including the shared backbone.

4.3 Our Contribution

We observe that there has been unused potential in self-supervision based on ordering: Previous works [93, 49, 114, 116, 21] have *randomly* selected the permutations used for training the CNN. However, can we not find permutations that are of higher utility for improving a CNN representation than the random set? For instance, given a 3×3 jigsaw grid, shuffling two neighboring image patches, two patches in faraway corners, or shuffling all patches simultaneously will learn the structure of different granularity. Thus diverse permutations will affect the CNN in a different way. Moreover, the effect of the permutations on the CNN changes during training since the state of the network evolves. During learning, we can examine the previous errors the network has made when recovering order and then identify a set of best-suited permutations. Therefore, wrapped around the standard back-propagation training of the CNN, we have a reinforcement learning algorithm that acts by proposing permutations for the CNN training. To learn the function of proposing permutations we simultaneously train a policy and self-supervised network by utilizing the improvement over time of the CNN network as a reward signal.

In previous works [110, 93, 49, 114, 21], for each training sample one permutation is randomly selected from a large set of candidate permutations $\psi_i \in \Psi$. Selecting the data permutation independent from the input data is beneficial as it avoids overfitting to the training data (permutations triggered only by specific samples). However, permutations should be selected conditioned on the state of the network that is being trained to sample new permutations according to their utility for learning the CNN representation.

4.3.1 Meta-learner for Improving Self-supervision

A Markov Decision Process for Proposing Permutations: We need to learn a function that proposes permutations conditioned on the network state and independent from samples x to avoid overfitting. Knowingly, the state of the network cannot be represented directly by the network weights, as the dimensionality would be too high for learning to be feasible. To capture the network state at time step t in a compact state vector s, we measure performance of the network on a set of validation samples $x \in X_{val}$. Each x is permuted by some $\psi_i \in \Psi$. A forward pass through the network then leads to activations φ_i and a softmax activation of the network,

$$y_i^\star = \frac{exp(\varphi_i)}{\sum_k exp(\varphi_k)}. \tag{4.2}$$

Given all the samples, the output of the softmax function indicates how good a permutation ψ_i can already be reconstructed and which ones are hard to recover (low $y_i^\star$). Thus, it reflects the complexity of a permutation from the view point of the network and $y_i^\star$ can be utilized to capture the network state s. To be precise, we measure the network's confidence regarding its classification using the ratio of correct class l vs. second highest prediction p (or highest if the true label l is not classified correctly):

$$y_l(x) = \frac{y_l^\star(x) + 1}{y_p^\star(x) + 1}, \tag{4.3}$$

where $x \in X_{val}$ and adding 1 to have $0.5 \leq y_l \leq 2$, so that $y_l > 1$ indicates a correct classification. The state s is then defined as

$$s = \begin{bmatrix} y_1(x_1) & \dots & y_1(x_{|X_{val}|}) \\ \vdots & & \vdots \\ y_{|\Psi|}(x_1) & \dots & y_{|\Psi|}(x_{|X_{val}|}), \end{bmatrix} \tag{4.4}$$

where one row contains the softmax ratios of a permutation ψ_i applied to all samples $x \in X_{val}$ (see Fig. 4.1(D)). Using a validation set for determining the state has the advantage of obtaining the utility for all permutations ψ_i and not only for the ones applied in the previous training phase. Moreover, it guarantees the comparability between validations applied at different time points independently by the policy. The action $a = (x, \psi_i) \in A = X \times \Psi$ of training the network by applying a permutation ψ_i to a random training sample x changes the state s (in practice we sample an entire mini-batch of tuples for one training iteration rather than only one). Training changes the network state s at time point t into s' according to some transition probability $T(s'|s,a)$. To evaluate the chosen action a we need a reward signal r_t given the revised state s'. The challenge is now to find *the action* which maximizes the expected reward

$$R(s,a) = \mathbb{E}[r_t | s_t = s, a], \tag{4.5}$$

given the present state of the network. The underlying problem of finding suitable permutations and training the network can be formulated as a Markov Decision Process (MDP)[151], a 5-tuple $< S, A, T, R, \gamma >$, where S is a set of states s_t, A is a set of actions a_t, $T(s'|s,a)$ the transition probability, $R(a,s)$ the reward and $\gamma \in [0,1]$ is the discount which scales future rewards against present ones.

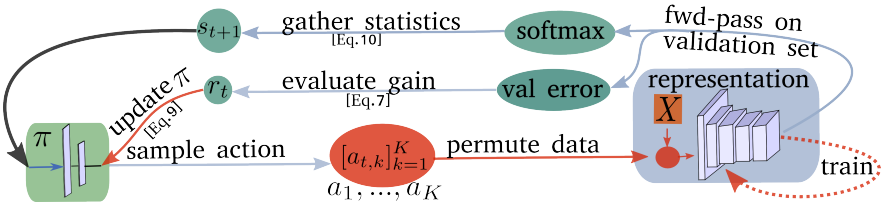

Figure 4.2: Training procedure of π. The policy proposes actions $[a_{t,k}]_{k=1}^{K}$ to permute the data X, used for training the unsupervised network. The improvement of the network is then used as reward r to update the policy.

Defining a Policy: As a reward r_t we need a score which measures the impact the chosen permutations have had on the overall performance in the previous training phase. For that, the error

$$\mathcal{E} := 1 - \frac{1}{|\Psi| \cdot |X_{val}|} \sum_{l=1}^{|\Psi|} \sum_{x \in X_{val}} \delta_{l, \underset{p=\{1,\ldots,|\Psi|\}}{\text{argmax}}\, y_p^\star(x)} \tag{4.6}$$

with δ the Kronecker delta, can be used to assess the influence of a permutation. To make the reward more informative, we compare this value against a baseline (BL), which results from simply extrapolating the error of previous iterations, i.e. $\mathcal{E}_{t+1}^{BL} = 2\mathcal{E}_t - \mathcal{E}_{t-1}$. We then seek an action that improves upon this baseline. Thus, the reward r_t obtained at time point $t+1$ (we use the index t for r at time step $t+1$ to indicate the connection to a_t) is defined as

$$r_t := \mathcal{E}_{t+1}^{BL} - \mathcal{E}_{t+1}. \tag{4.7}$$

We determine the error using the same validation set as already employed for obtaining the state. In this way no additional computational effort is required.

Given the earlier defined state s of the network and the actions A we seek to learn a policy function

$$\pi(a|s,\theta) = P(a_t = a|s_t = s, \theta_t = \theta), \tag{4.8}$$

that, given the θ parameters of the policy, proposes an action $a = (x, \psi_i)$ for a randomly sampled training data point x based on the state s, where $\pi(a|s,\theta)$ is the probability of applying action $a \in A$ at time point t given the state s. The parameters θ can be learned by maximizing the reward signal r. It has been proven that a neural network is capable of learning a powerful approximation of π [111, 151, 144]. However, the objective function (maximizing the reward) is not differentiable. In this case, Reinforcement Learning (RL)[151] has now become a standard approach for learning π in this particular case.

Policy Gradient: There are two main approaches for attacking deep RL problems: Q-Learning and Policy Gradient. We require a policy which models action probabilities to prevent the policy from converging to a small subset of permutations. Thus, we utilize a Policy Gradient (PG) algorithm which learns a stochastic policy and additionally guarantees convergence (at least to a local optimum) as opposed to Q-Learning. The objective of a PG algorithm is to maximize the

expected cumulative reward (Eq. 4.5) by iteratively updating the policy weights through back-propagation. One update at time point $t+1$ with learning rate α is given by

$$\theta_{t+1} = \theta_t + \alpha \Big(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} \Big) \nabla \log \pi(a|s, \theta), \tag{4.9}$$

Action Space: The complexity of deep RL increases significantly with the number of actions. Asking the policy to permute a sample x given the full space Ψ leads to a large action space. Thus, we dynamically group the permutations into $|C|$ groups based on the state of the spatiotemporal network. The permutations which are equally difficult or equally easy to classify are grouped at time point t and this grouping changes over time according to the state of the network. We utilize the state s (Eq. 4.4) as input to the grouping approach, where one row s_i represents the embedding of permutation ψ_i. A policy then proposes one group $c_j \in C$ of permutations and randomly selects one instance $\psi_i \in c_j$ of the group. Then a training data point x is randomly sampled and shuffled by ψ_i. This constitutes an action $a = (x, \psi_i)$. Rather than directly proposing individual permutations ψ_i, this strategy only proposes a set of related permutations c_j. Since $|C| << |\Psi|$, the effective dimensionality of actions is significantly reduced and learning a policy becomes feasible.

Network State: To obtain a more concise representation $\hat{s} = [\hat{s}_j]_{j=1}^{|C|}$ of the state of the spatiotemporal network (the input to the policy), we aggregate the characteristics of all permutations within a group c_j. Since the actions are directly linked to the groups, the features should contain the statistics of c_j based on the state of the network. Therefore we utilize per group (i) the number of permutations belonging to c_j and (ii) the median of the softmax ratios (Eq. 4.3) over the (ψ_i, x) pairs with $\psi_i \in c_j$ and $x \in X_{val}$

$$\hat{s} = [|c_j|, median \left([s_i]_{\psi_i \in c_j} \right)]_{j=1}^{|C|}. \tag{4.10}$$

The median over the softmax ratios reflects how well the spatiotemporal network can classify the set of permutations which are grouped together. Including the size $|c_j|$ of the groups helps the policy to avoid the selection of very small groups which could lead to overfitting of the self-supervised network on certain permutations. The proposed $\hat{s}$ have proven to be an effective and efficient representation of the state. Including global features, as for example the iteration or learning rate utilized in previous work [45, 46], does not help in our scenario. It rather increases the complexity of the state and hinders policy learning. Fig. 4.1(D) depicts the validation process, including the calculation of state $\hat{s}$ and the reward r.

Training Algorithm: We train the self-supervised network and the policy simultaneously, where the training can be divided in two phases: the self-supervised training and the policy update (see Fig. 4.2 and Algorithm 1 in section A of the Supplementary Material). The total training runs for T steps. Between two steps t and $t+1$ solely the self-supervised network is trained (π is fixed) using SGD

for several iterations using the permutations proposed by π. Then, $\hat{s}$ is updated using the validation procedure explained above. At each time step t an episode (one update of π) is performed. During episode t, the policy proposes a batch of K actions $[a_t]_{k=1}^{K}$, based on the updated state $\hat{s}_t$, which are utilized to train the self-supervised network for a small amount of iterations. At the end of the episode, another validation is applied to determine the reward r_t for updating π (Eq. 4.9). The two phases alternate each other until the end of the training.

Computational Extra Costs during Training: With respect to the basic self-supervised training, the extra cost for training the policy derives only from the total number of episodes $\times$ the time needed for performing an episode. If the number of SGD iterations between two policy updates t and $t+1$ is significantly higher than the steps within an episode, the computational extra costs for training the policy is small in comparison to the basic training. Fortunately, sparse policy updates are, in our scenario, possible since the policy network improves significantly faster than the self-supervised network. We observed a computational extra cost of $\sim$40% based on the optimal parameters. Previous work, [45, 186] which utilize deep RL for meta-learning, need to repeat the full training of the network several times to learn the policy, thus being several times slower.

4.4 Implementation Details

All deep networks are implemented using the PyTorch[1] framework.

Spatiotemporal Jigsaw: The shared basic model of the spatiotemporal network up to pool5 has the same architecture as CaffeNet [75] with batch normalization[71] between the conv layers. The still images utilized for the spatial task are chosen from the training set of the Imagenet dataset [137]. For training our model with the temporal task, we utilize the frames from split1 of the human action dataset UCF-101 [149]. We use 1000 initial permutations for both tasks ($|\Psi| = 1000$). we use SGD with a starting learning rate of 0.001 which we reduce after 200k iterations by a factor of 10. Our network runs in total for 350k iterations. We use a batchsize of 128 for both spatial and temporal tasks. For the spatial classification branch, the fc6-layer has a size of 1024, fc7 has 4096 dimensions. For the temporal task we use an fc6-layer with 512 neurons and one LSTM layer with the hidden dimension of 256. The policy network is trained using the ADAM optimizer with a starting learning rate of 0.01. For the temporal task, we randomly crop a patch with the size of 224x224 per frame and resize to 75x75. For the spatial task, each tile has the size of 75x75. Having the same input as the temporal task simplifies the implementation phase. As in [114], conv1 has stride 2 during the unsupervised training, and it is changed to 4 during all evaluation experiments. As augmentation, we randomly crop each tile/frame, apply a random color jittering to each of them and normalize the tiles/frames separately. For the spatial task we divide an input image x into 9 non-overlapping parts. For the temporal task, we randomly select 8 frames from each video.

[1] `http://pytorch.org/`

Methods	UCF101[149]					PascalVOC07[42]				
	Top1	Top5	Top10	Top20	Top50	Top1	Top5	Top10	Top20	Top50
Random	18.8	25.7	30.0	35.0	43.3	17.6	61.6	75.5	85.5	94.2
Jigsaw [114]	19.7	28.5	33.5	40.0	49.4	39.2	71.6	82.2	89.5	96.0
OPN [93]	19.9	28.7	34.0	40.6	51.6	33.2	67.1	78.5	87.0	94.6
Ours	**25.7**	**36.2**	**42.2**	**49.2**	**59.5**	**54.3**	**73.0**	**83.0**	**89.9**	**96.2**

Table 4.1: Evaluating the visual representation using an unsupervised nearest neighbor approach on split1 of UCF-101 and Pascal VOC 2007 dataset. We compare the results gained by (i) a random initialization, (ii) a spatial approach [114], (iii) a temporal method [93], and (iv) spatiotemporal jigsaw + our meta-learner. For extracting the features based on the weights of (ii) and (iii) we utilize their published models

Our meta-learner: To train the policy we use the Policy Gradient algorithm REINFORCE (with moving average subtraction for variance reduction) and add the entropy of the policy to the objective function which improves the exploration and therefore prevents overfitting (proposed by [166]). The policy network contains 2 FC layers, where the hidden layer has 16 dimensions. We use K-means clustering for grouping the permutations in 10 groups. The validation set contains 100 ($|X_{val}| = 100$) samples and is randomly sampled from the training set (and then excluded for training).

4.5 Experiments

Our ablation study evaluates quantitatively our contribution, showing that a strategy better than random can be found automatically. The visual representation is tested quantitatively and qualitatively on a variety of contrasting vision tasks, including image classification, object detection, object segmentation, and action recognition (Section 4.6).

Datasets: *UCF-101* [149] contains 101 different action classes and over 13k clips. *HMDB-51* [87] contains 51 classes and around 7k clips. The *Imagenet* [137] benchmark consists of ∼1.3M images divided in 1000 objects category. The *Pascal VOC 2007* [42] dataset consists of 9,963 images, containing 24,640 annotated objects which are divided in 20 classes. Based on the default split, 50% of the images belong to the training/validation set and 50% to the testing set. We use the provided bounding boxes of the dataset to extract the individual objects, whereas patches with less than 10k pixels are discarded.

4.5.1 Nearest Neighbor Search

Nearest neighbor search is a good unsupervised method for evaluating the visual representation. We test the learned representation on two datasets: Pascal VOC for images and UCF101 for videos.

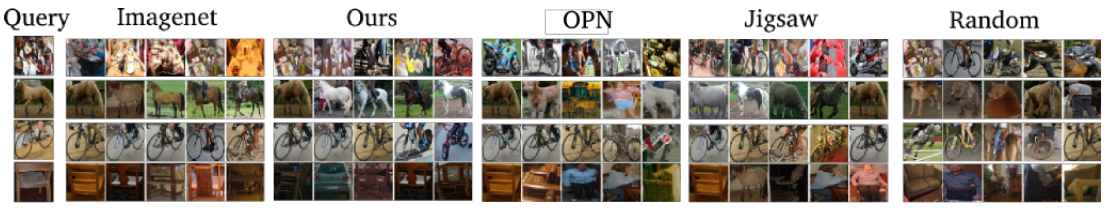

Figure 4.3: Given a query from the Pascal VOC test set, we provide the top5 nearest neighbor from the train set based on pool5 using cosine distance. We compare the models from (i) supervised training with the Imagenet classification task, (ii) our meta-learner(+spatiotemporal jigsaw) approach, (iii) OPN as a temporal approach [93], (iv) Jigsaw as a spatial method [114] and (v) a random initialization.

For each image/frame we resize the input to 227×227 as standard procedure, then extract the pool5 activation. For each image/video in the test set, top$k$ NN are retrieved from the train set using cosine distance. For UCF101, the representation is the average between pool5 from 10 frames per video. If the class of a test sample appears within the top$k$ train samples, it is considered correctly predicted. Finally, the mean accuracy [%] is reported averaged across test samples.

Tab. 4.1 shows the accuracy for $k = 1, 5, 10, 20, 50$ computed on UCF-101 and Pascal VOC 2007, respectively. It can be seen, that our model achieves the highest accuracy for all k, meaning that our method produces more informative features for object/video classification. Note, that especially the accuracy of Top1 is much higher in comparison to the other approaches.
We additionally evaluate our features qualitatively by depicting the Top5 nearest neighbors in the training set given a query image from the test set (see Fig. 4.3). We compare our results with [114, 93], a random initialization, and a network with supervised training using the Imagenet dataset.

4.6 Transfer Capabilities of the Unsupervised Visual Representation

After training our visual representation without using labels, we evaluate its generalization quality by testing on different visual tasks. For the following experiments, we initialize all networks with our trained model up to conv5 and fine-tune on the specific task using standard evaluation procedures.

Imagenet 4.2: The backbone is initialized using our unsupervised representation and frozen during fine-tuning. The classifier is fine-tuned on the Imagenet challenge [137] on top of conv5. The classifier is either a single linear layer, as introduced by Zhang *et al.* [179], or a two-layer network, proposed by Noroozi and Favaro [114]. Tab. 4.2 shows that our visual representation obtains more than 2% over the best model with a comparable architecture and almost 4% in the linear task.

Action recognition: The experiment is performed using the PyTorch im-

Method	Non-Linear	Linear
Imagenet	59.7	50.5
Random	12.0	14.1
Videos [164]	29.8	-
OPN* [93]	29.6	-
Context [38]	30.4	29.6
Colorization[179]	35.2	30.3
BiGan[40]	34.8	28.0
Split-Brain[180]	-	32.8
NAT[20]	36.0	-
Jigsaw[114]	34.6	27.1
Ours	38.2	**36.5**
RotNet+[55]	**43.8**	**36.5**

Table 4.2: Imagenet challenge [137] using our visual representation up to pool5. We train a Linear[179] and Non-linear[114] classifier while keeping the features (pool5) frozen. (*: indicates our implementation of the model, +: indicates bigger architecture due to missing groups in the conv layers)

Method	UCF-101	HMDB-51
Random	47.8	16.3
Imagenet	67.7	28.0
Shuffle&Learn [110]	50.2	18.1
VGAN [157]	52.1	-
Luo et. al [99]	53.0	-
OPN [93]	56.3	22.1
Jigsaw* [114]	51.5	22.5
Ours	**58.6**	**25.0**

Table 4.3: Testing the unsupervised visual representation on the action recognition task. The unsupervised representation is fine-tuned on UCF-101 and HMDB-51 and the test accuracy [%] is reported. '*': Jigsaw (Noroozi and Favaro [114]) do not provide results for this task, we replicate their results using our PyTorch implementation

plementation [2] provided by Wang *et al.* [161]. The network architecture and hyperparameters are retained from our model. The input is a single frame from each video for training and testing. As dataset, we use UCF-101 and HMDB-51, averaging the accuracy over the three provided splits. Our method outperforms the state-of-the-art by 2.3% on UCF-101 and 2.9% on HMDB-51. Notice that HMDB-51 is never used during training, showing that our visual representation generalizes better than other methods.

[2] https://github.com/yjxiong/temporal-segment-networks

Method	Classification[42]	Detection[42]	Segmentation[43]
Imagenet	78.2	56.8	48.0
Random	53.3	43.4	19.8
OPN[93]	63.8	46.9	-
Color17[90]	65.9	-	38.4
Counting[115]	67.7	51.4	36.6
PermNet[30]	69.4	49.5	37.9
Jigsaw[114]	67.6	53.2	37.6
RotNet[55]+	73.0	**54.4**	39.1
Ours	**74.2**	52.8	**42.8**

Table 4.4: Our visual representation is tested on three different visual tasks on Pascal VOC: (i) multi object classification, (ii) detection (VOC07) and (iii) segmentation (VOC12). For (i) and (ii) we show the mean average precision (mAP), for (iii) the mean intersection over union (mIoU). ('+': significantly larger conv layers)

Pascal VOC: Our unsupervised visual representation is then evaluated on different vision tasks. On Pascal VOC07 [42], we test on multi-object classification using the procedure described in [82]. On the same dataset, we evaluate on object detection following the experimental protocol described in [132]. Then, using the Pascal VOC12 [43] dataset, we test on object segmentation using FCN [96]. For all tasks, we initialize the backbone with our representation and fine-tune the whole network on the downstream task. Previous methods using deeper networks, such as [165, 39], are omitted from Tab. 4.4. The results in Tab. 4.4 show that we significantly improve upon the other approaches. Our method outperforms even [55] in object classification and segmentation, which uses batch normalization also during fine-tuning and uses a larger network due to the group parameter in the conv layers.

4.7 Ablation Study

In this section, we compare the baseline method (S+T) with our improved version using the controller policy (P) during training.

Method	S	S+P	T	T+P	S+T	S+T+P
Pascal	67.6	71.3	64.1	65.9	72.0	**74.2**
UCF-101	51.5	54.6	52.8	55.7	57.3	**58.6**

Table 4.5: We compare the different models on the multi-object classification task using the Pascal VOC07 and on the action recognition task using UCF-101. (S):**S**patial task, (S+P):**S**patial task + **P**olicy, (T): **T**emporal task, (T+P):**T**emporal task + **P**olicy, (S+T):**S**patial and **T**emporal task simultaneously, (S+T+P):all approaches simultaneously

Unsupervised Feature Evaluation: In Fig. 4.5 the models are evaluated

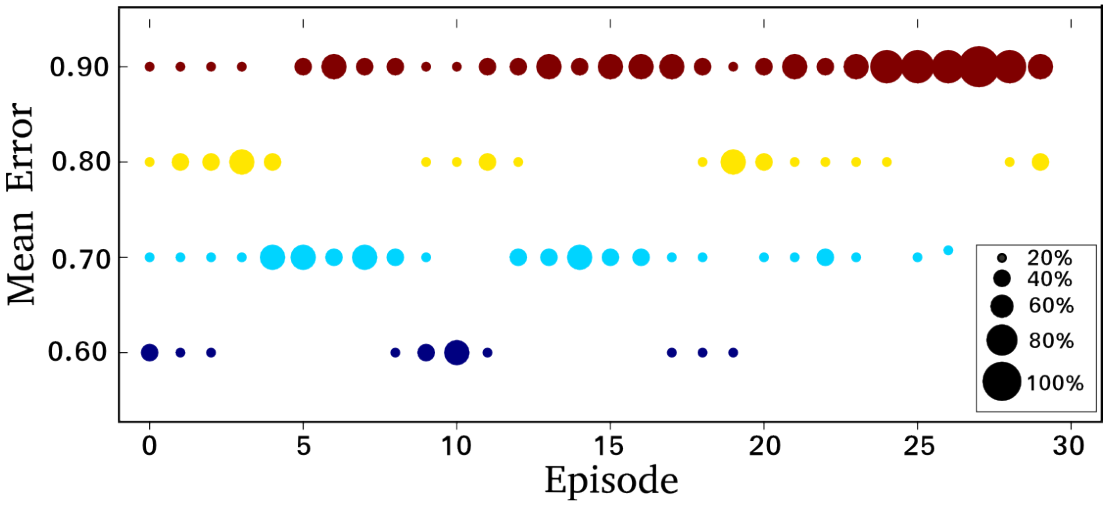

Figure 4.4: Permutations chosen by the policy in each training episode. For legibility, ψ_i are grouped by validation error into four groups. The policy, updated after every episode, learns to sample hard permutations more often in later iterations

on the Pascal VOC object classification task without any further fine-tuning by extracting pool5 features and computing cosine similarities for nearest neighbor search as described in section 4.5.1. This unsupervised evaluation shows how well the unsupervised features can generalize to a primary task, such as object classification. Fig. 4.5 shows, that each of the three models (Spatial, Temporal and Spatiotemporal) has a substantial gain when the CNN is trained using the policy. Our final model, composed of the spatiotemporal task with policy (S+T+P), reaches almost the supervised features threshold ("imagenet" line in Fig. 4.5).

Supervised Fine-Tuning: In Tab. 4.5, a supervised evaluation has been performed starting from the unsupervised visual representation. Each model is fine-tuned on the multi-class object classification task on Pascal VOC 2007 and video classification using UCF-101. The results are consistent throughout the unsupervised evaluation, showing that the methods with RL policy (S+P, T+P, and S+T+P) improve over the baseline models.

Policy Learning: Fig. 4.4 shows the permutations chosen by the policy while it is trained at different episodes (x-axis). This experiment aims to analyze the learning behavior of the policy. For this reason, we initialize the policy network randomly and the CNN model from an intermediate checkpoint (average validation error 72.3%). Per episode, the permutations are divided into four complexities (based on the validation error) and the relative count of permutations selected by the policy is shown per complexity. Initially, the policy selects the permutations uniformly in the first three episodes, but then learns to sample with higher frequency from the hard permutations (with high error; top red) and less from the easy permutations (bottom purple), without overfitting to a specific complexity but mixing the hard classes with intermediate ones.

Fig. 4.6 depicts the spatial validation error over the whole training process of the spatiotemporal network with and without the policy. The results are consistent with the unsupervised evaluation, showing a faster improvement when

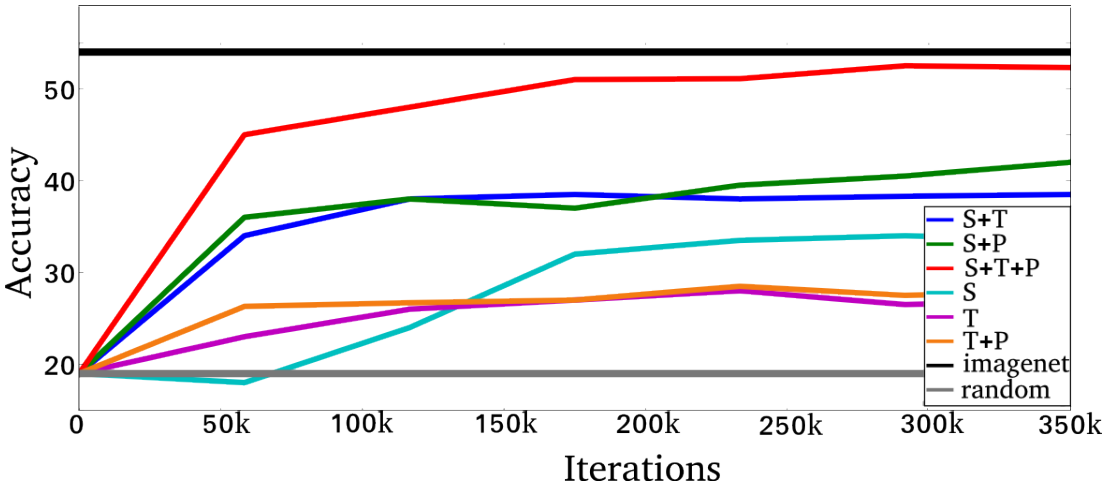

Figure 4.5: The test accuracy from Top1 nearest neighbor search evaluation on VOC07 is used for comparing different ablations of our architecture during training. The curves show a faster improvement of the features when the policy (P) is used

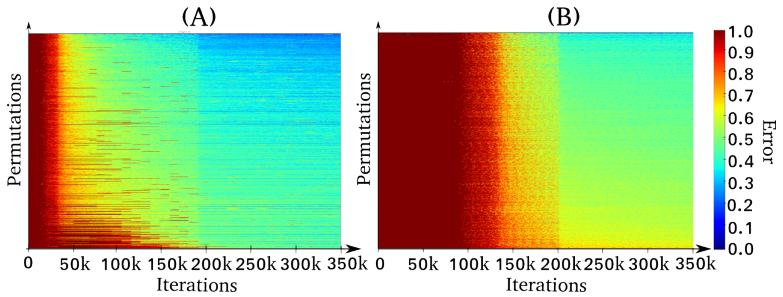

Figure 4.6: Error over time of the spatial task, computed using the validation set and sorted by the average error. Each row shows how the error for one permutation evolves over time. (A): with Policy, (B): without policy

training with the permutations proposed by the policy than with random permutations. Note that (B) in Fig. 4.6 shows a uniform improvement over all permutations, whereas (A) demonstrates the selection process of the policy with a non-uniform decrease in error.

4.8 Policy Detailed Analysis

In this section, we will provide extra details about our controller policy, in particular how we found the hyper-parameters and their effect on the final result.

4.8.1 Size of Validation Set

In Sec. 4.3.1 we mention the usage of 100 images for the validation set X_{val}. Fig. 4.7 shows an evaluation of the optimal size for the validation set based on the mean and standard deviation of the error (y-axis) using several randomly sampled validation sets with size $|X_{val}| = 10, 20, 50, 100$ or 200 at different time steps (x-

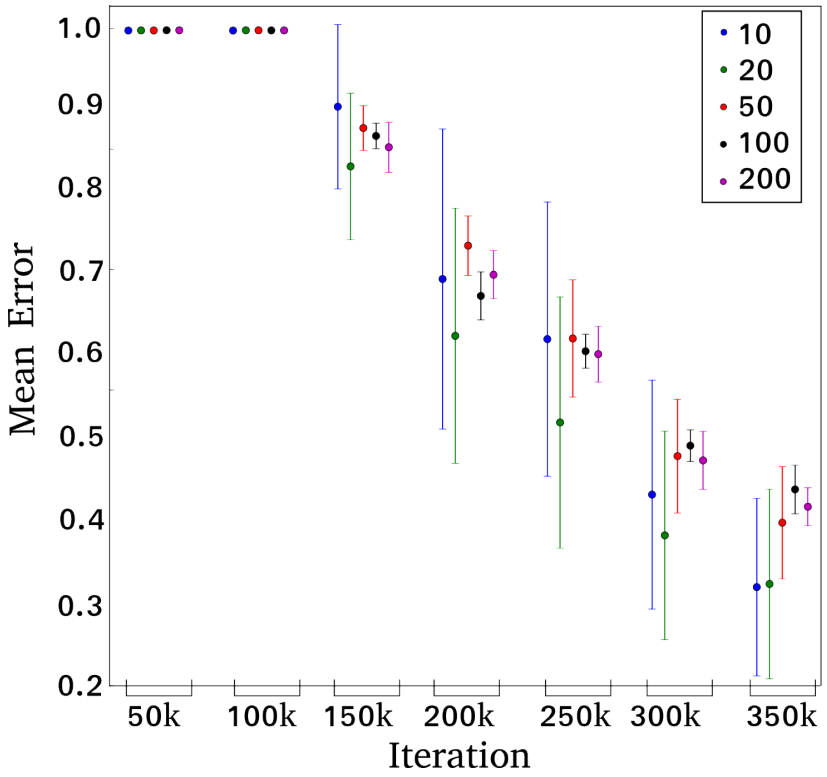

Figure 4.7: Evaluation of the optimal size for X_{val}.

axis). We randomly sample 5 different sets per size and compute for every set the mean error given the checkpoints of the self-supervised network trained without policy at iteration 50k, 100k, 150k, 200k, 250k, 300k and 350k. Fig. 4.7 then shows the mean and standard deviation over the 5 sets regarding a specific size and iteration. While the overall tendency of the error over the consecutive training iterations is similar for all validation sizes, $|X_{val}| = 10, 20, 50$ show comparably large standard deviation. For the other two sizes there is only little difference which motivates our choice of $|X_{val}| = 100$.

4.8.2 Number of Groups

We declare in Sect. 4.1 the choice of 10 groups which we are going to analyze subsequently. We use the softmax ratios $y_i^\star$ (Eq. 4.3) to determine the complexity of a permutation from the view point of the network. Fig. 4.8 shows the distribution of all $y_i^\star$ over the (ψ_i, x) pairs with $\psi_i \in \Psi$ and $x \in X_{val}$ (all entries of s, Eq.4.4) at time point $300k$ as histogram. We compute this distribution for all ψ_i which are part of a group. We then test the distributions of the different groups for equality using the Kolmogorov-Smirnov-Test (KS-test; Null-Hypothesis is that the distributions are the same). If the p-value returned by the KS-test is smaller than a predefined significance value $\alpha = 0.01$ the Null-Hypothesis can be rejected and the distributions are assumed to be different. We utilize this measure to identify groups which have a similar distribution and should therefore be grouped together. In this way, we can find the optimal amount of groups without having two separate groups with the same distribution/difficulty. Fig. 4.9 depicts the

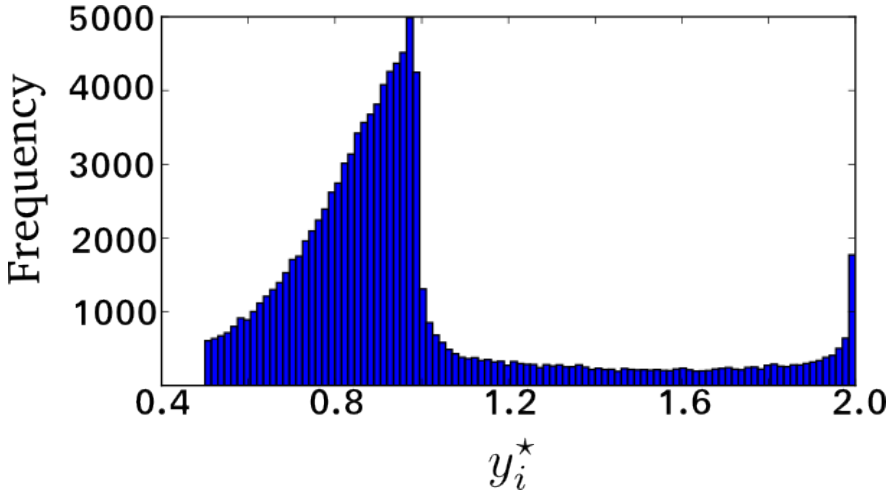

Figure 4.8: Frequency of the softmax ratios $y_i^\star$ given all entries of s (Eq. 4.4) at time point $300k$.

matrices of pairwise p-values for $|C| = 10, 15$ and 20 at time point $150k$, $250k$ and $350k$. It can be seen, that there are already several groups for $|C| = 15$ where the Null-Hypothesis cannot be rejected anymore (values higher than α), i.e. $|C| = 15$ is already to high for avoiding groups of similar complexity. Therefore, 10 groups seems to be the best choice for the clustering approach.

4.8.3 Baseline Error $\mathcal{E}^{BL}$ Description

In Eq. 4.7 we define the baseline error $\mathcal{E}_{t+1}^{BL}$ as the minimum error that the policy network needs to achieve to receive a positive reward. Fig. 4.10 illustrates more in details the use of this baseline with respect to the reward computation. The baseline $\mathcal{E}_{t+1}^{BL}$ is computed by linear extrapolation based on the error $\mathcal{E}_{t-1}$ and $\mathcal{E}_t$ in the previous time points $t-1$ and t. For extrapolating $\mathcal{E}_{t+1}^{BL}$ we use the equation

$$f(u_3) = f(u_1) + \frac{u_3 - u_1}{u_2 - u_1}(f(u_2) - f(u_1)). \tag{4.11}$$

where a point $(u, f(u))$ corresponds to our errors $(t, \mathcal{E}_t)$ and the extrapolated point $f(u_3)$ corresponds to our baseline error $\mathcal{E}_{t+1}^{BL}$. Substituting $\{u_1, u_2, u_3\}$ with $\{t-1, t, t+1\}$ and $f(u)$ with $\mathcal{E}_t$ results in

$$\mathcal{E}_{t+1}^{BL} = \mathcal{E}_{t-1} + \frac{(t+1) - (t-1)}{t - (t-1)}(\mathcal{E}_t - \mathcal{E}_{t-1}) = 2\mathcal{E}_t - \mathcal{E}_{t-1}. \tag{4.12}$$

4.8.4 How Decisive Are the Permutations?

In this section we evaluate the impact on performance that permutation selection has during training. In particular, we use our trained policy for selecting the permutations and evaluate the model after one epoch. As baseline we use the random policy which selects the permutations uniformly at random. Moreover,

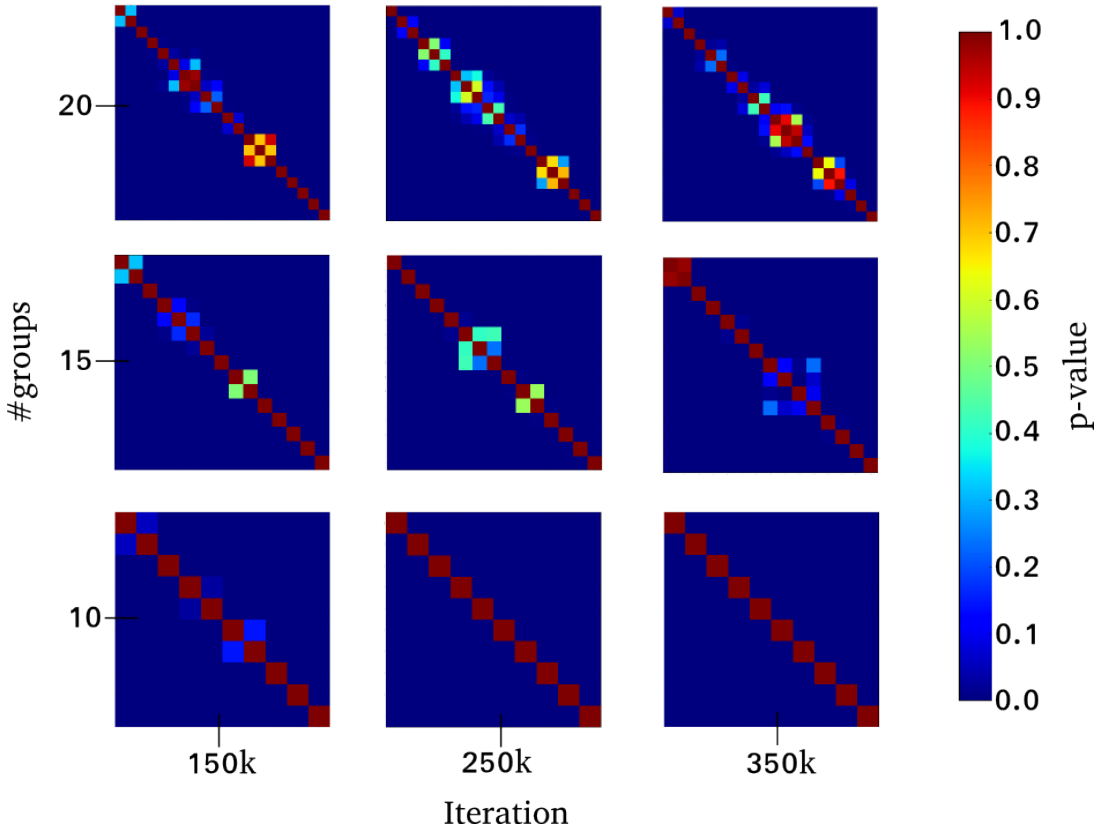

Figure 4.9: Pairwise p-values for a different amount of groups at several checkpoints.

we evaluate the permutations which are discarded by our policy. Therefore, we utilize an inverse policy in order to understand the importance of the permutation selection. It turns out that the inverse policy impairs training, producing features worse than the random policy ($(78 \pm 16)\%$, see Tab. 4.6), while our policy always increases the performance with respect to the random policy.
The validation accuracy shown in Tab. 4.6 refers to unsupervised training. We initialize the network from a given checkpoint and train for one epoch following one of the three policies. The final result is the ratio between our/inverse policy and the baseline random policy. Then we average over several checkpoints.

4.8.5 Permutation Selection of Our Policy

As discussed in Sec. 4.3, defining the complexity of a permutation should depend on the state of the network and not, for example, only on the degree of shuffling independently of the network. For this reason, we utilize the validation error as input for our policy. When illustrating how often permutations with a particular shuffling (Hamming distance to the not-shuffled sequence) are selected by our policy during the training process (see Fig. 4.11) one should not be able to recognize a specific pattern, as for example a curriculum that selects easy samples (strongly permuted) at earlier iterations and harder ones (only small changes) at later iterations. Fig. 4.11 shows that our trained policy does not follow a simple curriculum learning procedure. It selects the permutations only based on the

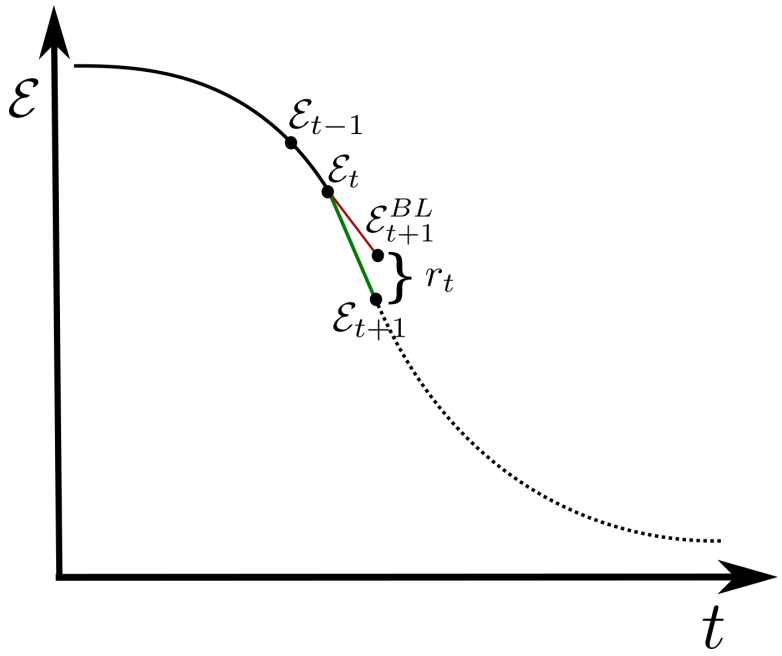

Figure 4.10: The reward r_t is positive when the error $\mathcal{E}_{t+1}$, obtained by training the self-supervised network using the policy, is below the extrapolated baseline error $\mathcal{E}^{BL}_{t+1}$

Table 4.6: Validation accuracy after one epoch of training using the policy in the left column. The accuracy is relative to the random policy. The experiment is repeated for several checkpoints, the reported accuracy is the mean and std over those repetitions. The performances when using the inverse policy are worse than the random policy baseline, while our policy always outperforms the baseline.

Method	Relative Accuracy
RL policy	$(125 \pm 14)\%$
Inverse policy	$(78 \pm 16)\%$

state of the network as can be seen in Fig. 4.4. Qualitative examples of chosen permutations, depicted in Fig. 4.12, confirm this behavior as no correlation between the degree of shuffling and the training iteration is visible.

4.8.6 Extra Computational Costs

Policy Cost Calculation

In this section we derive the computational cost of using the policy during training relative to the computation of the basic self-supervised training. Including the policy introduces three additional phases in the training algorithm: action sampling (policy inference), update of the policy, and validation for computing state $\hat{s}$ and reward r. The inference and update of the policy are omitted from the calculation since their cost is orders of magnitude lower with respect to the main network, given the minor size of the policy network. Therefore the cost of

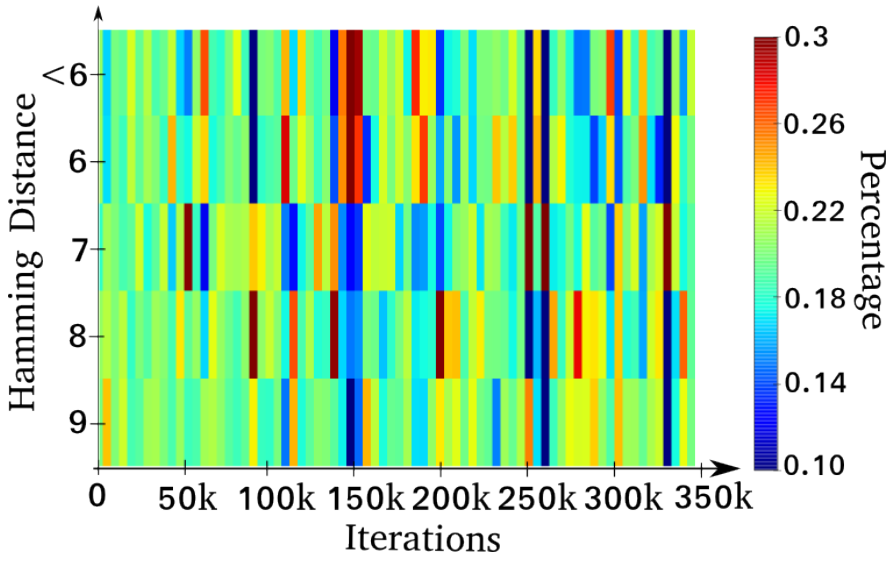

Figure 4.11: Percentage of permutations chosen by the policy at diverse training iterations. The permutations are structured using the Hamming distance to the not-shuffled sequence.

the policy derives from the computational cost V of the validation phase. In fact, for each sample in the validation set (100 samples following Sec. 4.8.1), the main network performs one forward pass per each of the 1000 permutations, resulting in

$$V = \frac{100 \cdot 1000}{128} \approx 780 \tag{4.13}$$

where 128 is the batch size. The validation phase is then performed twice per episode t, at the beginning (computing $\hat{s}_t$) and the end of the episode (for the reward r_t). The final computational cost of using the policy is calculated by multiplying the episode cost by the number of episodes $T = 90$ performed during the entire training

$$CC = T \cdot (2 \cdot V). \tag{4.14}$$

The number of total updates T is set to 90 since the policy does not benefit from an higher frequency of updates (no additional performance gain), which would only increase the computational cost.
We can compute the policy cost in relation to the self-supervised training as

$$\frac{CC}{I} = \frac{T \cdot (2 \cdot V)}{I} = \frac{90 \cdot (2 \cdot 780)}{350000} \approx 40\%. \tag{4.15}$$

given the total number of iterations $I = 350k$ to train the self-supervised network.

Performance Relative to Computation

Fig. 4.5 shows the performances of the unsupervised features during training, based on the iterations of the self-supervision network. Since our goal was to compare the convergence speed of the main network with and without policy,

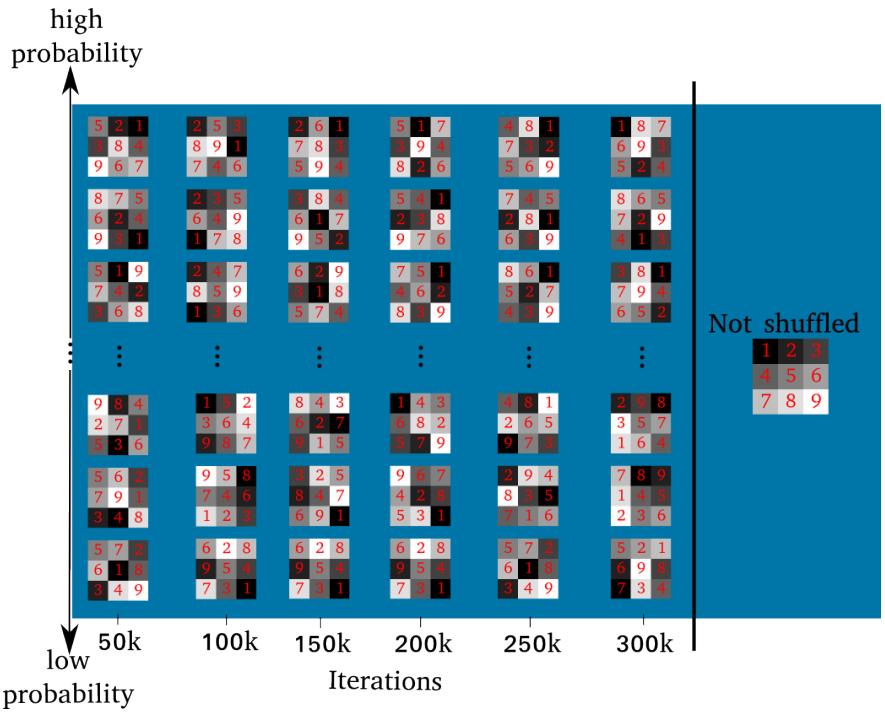

Figure 4.12: Qualitative examples of permutations with a high or low probability to be chosen by the policy at different time points.

Fig. 4.5 does not consider the additional iterations necessary for training the policy. For Fig. 4.13 we normalize the x-axis by taking into account the episodes needed to train the policy network. Since the extra cost mainly derives from the forward passes of the self-supervised network during the validation phase, we use the total number of forward passes on the x-axis. Fig. 4.13 shows that, even considering the extra computational cost needed to train the policy, there is a big advantage of using the policy during training.

4.8.7 Activation

Fig. 4.14,4.15 show the top activation for different conv5 units of our self-supervised trained model following the approach described by Zhou *et al.* [183]. In short, we run all images of a particular dataset through the network and output the top activations per unit contained in the conv5 layer. In Figure E.4.14 we show three neurons over three different datasets: Imagenet, Pascal VOC, and UCF-101. Due to the number of images included in the Imagenet dataset, we only use the test set for visualizing the top activations. For UCF-101 we use one frame per video contained in the training set of split1. Having a consistent activation across different datasets shows the transfer capability of our feature representation. In particular the first row of Figure E.4.14 is the activation of a unit responding to eyes, the second recognizes faces and the third reacts to sky in landscapes. Figure E.4.15 shows additionally that the network learns to recognize very particular object parts, like the tires of a four-wheel vehicle.

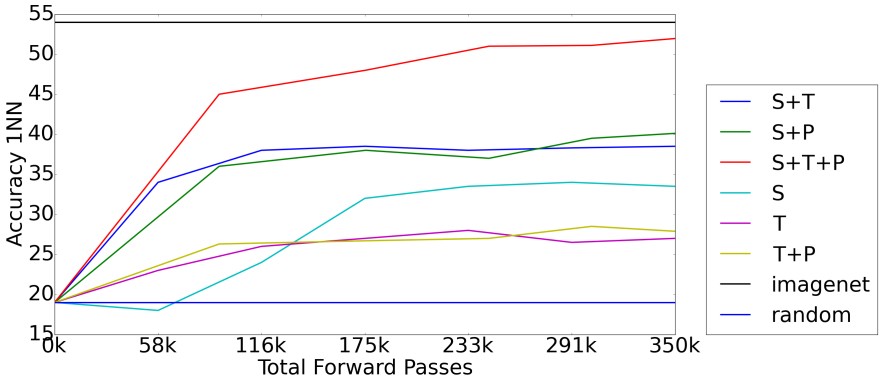

Figure 4.13: Unsupervised object classification on Pascal VOC 2007 per number of total forward passes computed by the self-supervised network. The x-axes contains the unsupervised training plus the validation for the policy. 'Random' and 'Imagenet' are computed using respectively random weights and features trained with labels on ImageNet.

Figure 4.14: Rows: Top activations of 3 different conv5 neurons across three datasets (columns). Note, that the neurons exhibit the same behavior in all datasets; The first unit focuses on eyes, the second on faces and the third on sky in a landscape.

Figure 4.15: Top activations of a single neuron firing on car wheels. The same neuron is evaluated on different images of Imagenet (1st row) and Pascal VOC (2nd row).

4.9 Summary

In this chapter, we studied self-supervision, a paradigm to learn visual representation without human annotation. The visual representation can then be used directly to compute similarities between images and videos, or as initialization for a downstream task.

To sample data permutations, which are at the core of any surrogate ordering task, we have proposed a policy based on RL requiring relatively small extra computational cost. Therefore, the sampling policy adapts to the state of the network that is being trained. As a result, permutations are sampled according to their expected utility. In experiments on diverse tasks ranging from image classification and segmentation to action recognition in videos, our adaptive policy for spatiotemporal permutations has shown favorable results compared to the state-of-the-art.

Since the publication of this work in Buechler *et al.* [23], new self-supervision methods have been proposed for images and video, based on the Jigsaw puzzle idea. For example, two important papers are Noroozi *et al.* [116] for images and

Xu *et al.* [170] for video.

Chapter 5

Behavior Analysis for Rodents using Unsupervised Visual Representation

In the previous chapter, we introduced several methods to learn a good visual similarity, bypassing the use of human annotation. In this chapter, we apply the learned visual representation to a real-life application: motor behavior analysis. In summary, the task is to compare the motor-skills across different animal subjects to quantify their degree of illness and studying their improvement during rehabilitation.

In this chapter, we introduce the concept of behavior analysis and describe the challenges and contributions in Sec. 5.1. Then we provide more details about our method (Sec. 5.2.1) and evaluate qualitatively and quantitatively the unsupervised visual similarity (Sec. 5.2.2). Finally, we test our method on several biomedical tasks: motor-skill training and rehabilitation (Sec. 5.3), predicting the brain status from behavior (Sec. 5.4).

The work presented in this chapter was partially published in Brattoli *et al.* [21] and it is currently under review for publication.

5.1 Behavior Analysis

Behavior analysis (BA) is the study of motor function during a single, repeating action. BA is a major tool for biomedical research, for example for clinical study, since it is a non-invasive method to study the change of impairment during rehabilitation. Existing methods require physical or virtual markers to track the subject limbs, which is time-intensive.

Our contribution is an unsupervised visual similarity of posture and behavior. Our method provides an objective comparison of behavior across subjects without using keypoints or annotations. The method is evaluated on rodents with neurological diseases for several applications: rehabilitation and neuro-function correlation.

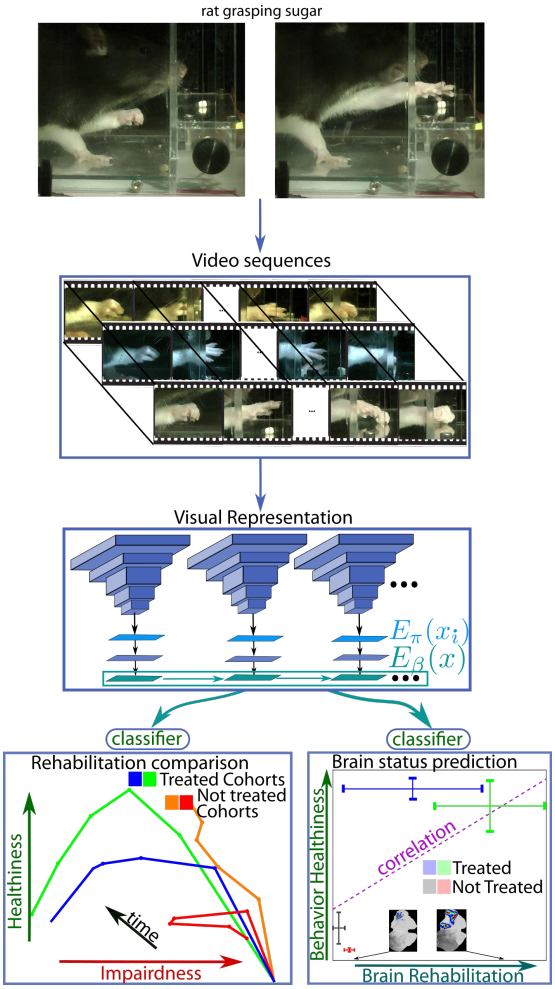

Figure 5.1: Our framework for behavior analysis. From the top: exemplary frames of the rats grasping sugar pellets; the proposed neural network for behavior encoding; examples of the applications enabled by our approach.

5.1.1 Definition

Study the elaborate internal processes of the brain is challenging and invasive. Therefore, we study their visible result: motor behavior, i.e. the voluntary dynamic change of posture. In many fields of biomedical research, the quantification of motor behavior constitutes an essential, non-invasive diagnostic strategy [16].

Complex movements, such as grasping, are coordinated by the brain and the signal is sent through the spinal cord to control the muscle activation. The analysis of the output, i.e. behavior, is crucial for the understanding of the brain function. BA can lead to detect and classify distinct functional deficits. Moreover, it can give us direct feedback on the treatment, allowing for individual adjustments [50].

5.1.2 Motivation

Videos of behavior recorded during the long-term recovery after neurological diseases provide an easily available, rich source of information to evaluate and adjust drug application and rehabilitative treatment paradigms.

The common practice for behavior analysis is to place physical markers on body joints which are easily detected by the algorithm [17, 156, 97, 69]. However, placing markers on the body can alter the behavior of the animal subjects in the process or being very time consuming, and annoying, for human patients.

A good alternative is marking the body parts on the video recording [133, 33, 126, 8]. This solves the previous problems, but requires time-intensive, error-prone human annotations.

Using machine learning, more advanced algorithms can track virtual keypoints, reducing the number of annotations needed [103, 8]. However, these models require tedious posture annotation of large amounts of training images [6]. Even using transfer learning, modern methods still require hundreds of labeled frames for a single animal.

Apart from the manual annotation, existing methods have another major weakness: the disease effect on the motor-skills has to be known *before* detecting keypoints, since the user needs to select specific body-parts to track. However, a true diagnostic tool should *discover* and localize deviant behavior, rather than *only confirm* it. Firstly, this makes BA, not objective, since different annotators may favor different body-parts. Second, the analysis might miss important neural mechanisms if the right body-parts are not tracked. This means that it is not possible to discover new characteristics of the disease, but only confirm those known to the user.

5.1.3 Contribution

We propose a fully automatic, unsupervised diagnostic support system for behavior analysis based on objective comparison of motor functions across subjects. A summary of the approach is shown in Fig. 5.1. Using an unsupervised training procedure avoids tedious labeling and an annotator bias and supports an objective analysis. Given reference videos of healthy and impaired behavior, a query video sequence can be compared to the references for discovering characteristic behavior. Being subject agnostic is crucial for an unbiased comparison. Moreover, we can compare the behavior of the same subject during a recovery process to quantify the improvement. Thus, we are closing the gap between merely extracting keypoints [103, 8, 133, 33, 126, 56, 125] and a direct behavior classification [77, 32] that is rather opaque to the user. Our novel algorithm for movement analysis is promising for diverse applications in the field of biomedical research and was evaluated on rodents subjects with stroke.

5.1.4 Experiment setup

We analyzed the recovery of impaired forelimb function in a rat stroke model where a stroke partially destroyed the sensorimotor cortex in one hemisphere. The effect on the impaired motor function was assessed using recordings of the

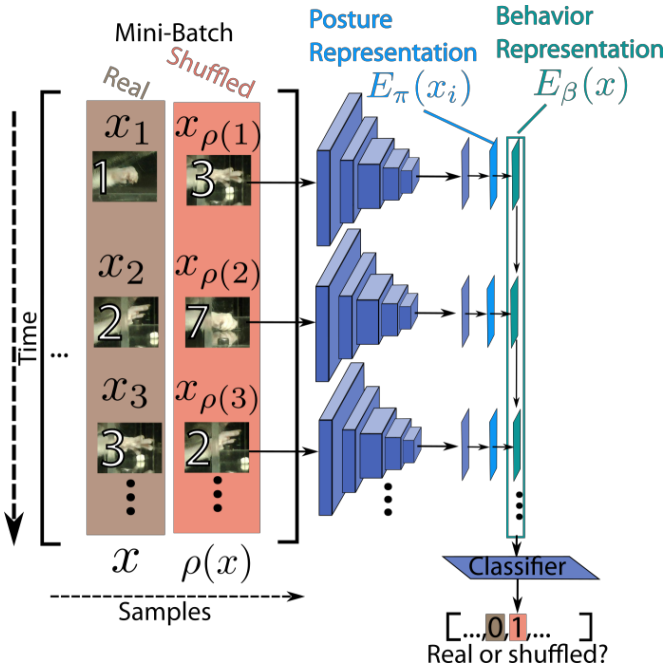

Figure 5.2: The training procedure of Brattoli *et al.* [21] for learning the behavior representation without manual annotation. The network is trained to extract a posture and behavior embedding ($E_\pi(x_i)$ and $E_\beta(x)$) only by forcing it to distinguish the characteristic structure of behavior sequences from shuffled sequences.

animals grasping a sugar pellet on a transparent shelf with minimum opening. The animals were separated into four groups based on the treatment: "Stimulation and Training", "Stimulation", "Delayed training" and "No Treatment". Recordings (50 frames/sec, consumer camera) have been taken during the initial training of the animals before the stroke and during recovery, which lasted up to 5 weeks after insult.

5.2 Approach

Before diving into the experiment application, we provide more details about the method and evaluate the learned visual similarity.

5.2.1 Self-supervision

As introduced in Sec. 5.1.2, a good method for comparing behavior needs to be subject agnostic, enabling the comparison of behavior across individuals despite their difference in appearance. This model should be trained without using human annotation since those are tedious and inject the annotator bias in the data. Using the self-supervision method proposed in Brattoli *et al.* [21] (Fig. 5.2), the network learns to extract the characteristic behavior from a video sequence without any user intervention using a surrogate task. During training, the network recognize normal behavior sequences $x = (x_1, x_2, ...)$ from the same sequence with frames $x_{\rho_{(i)}}$ randomly permuted by a permutation ρ (Fig. 5.2). The network

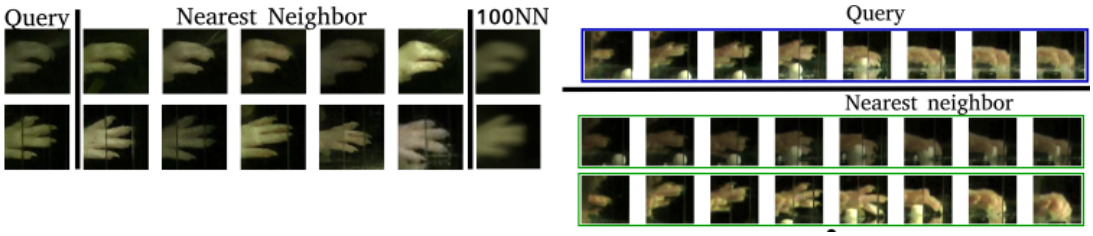

Figure 5.3: Nearest neighbors using our unsupervised visual similarity. (Left) Posture similarity: For each query, we show five nearest neighbors using cosine similarity on $E_\pi(x_i)$ and the rgb average of 100 NN, which show a consistent posture with almost no blurriness. (Right) Sequence similarity: we show two nearest neighbors selected using cosine similarity on $E_\beta(x)$ for a single query.

can solve this auxiliary task only by recognizing the paw posture $E_\pi(x_i)$ and the temporal dynamics while grasping, the behavior $E_\beta(x)$. Sequences from different animal subjects are mix together during training, pushing the model to ignore the differences in appearance and focus on the posture, enforcing the subject agnostic representation.

The behavior encoding $E_\beta(x)$ is a good visual representation that can be used for visual similarity to objectively compare behavior across subjects. Before showing application of our method, we carefully evaluate $E_\pi(x_i)$ and $E_\beta(x)$ in Sec. 5.2.2.

5.2.2 Evaluation

In this section, we evaluate the leaned encodings $E_\pi(x_i)$ and $E_\beta(x)$ both qualitatively and quantitatively, showing that they can be used for visual similarity of posture and motion sequence across subjects, therefore are a good candidate for behavior analysis.

Visual Similarity. We evaluate the visual similarity using the learned unsupervised encodings. Fig. 5.3 (Left) shows the nearest neighbor based on our posture encodings $E_\pi(x_i)$. Even though no supervision has been used, the model can retrieve images representing the same posture from different subjects. In particular, we provide the five nearest neighbors and an rgb average over the hundred closest samples. The consistent posture over the 100 NN shows that the posture similarity is strong. Similarly, we can retrieve similar motion using $E_\beta(x)$, shown in Fig. 5.3 (Right).

In Fig. 5.4, we project a thousands paw frames on a 2D plane using t-SNE [100]. The results shows that $E_\pi(x_i)$ extracts salient posture information and ignores appearance. Moreover, even thought the model was not explicitly trained for this, the time dimension is represented as a circle: the grasp starts at the top right and continues clockwise.

Comparison with Grund-Truth. For our approach, we compute a "healthiness score" by classify every sequence as healthy or impaired and calculate the

Figure 5.4: Projection of the posture representation $E_\pi(x_i)$ to a 2D plane using t-SNE. Similar postures are projected nearby and the time dimension appear as a circle, showing that the representation extracts useful posture information and rejects appearance characteristics.

percentage of healthy sequences per cohort. A linear classifier is trained on our visual representation to distinguish healthy and impaired.

Fig. 5.5 shows the healthiness score (HS) and the ground truth (GT) values provided by experts [3]. Our visual representation reaches the good results with a high correlation of 0.933 ± 0.005 using a simple linear classifier.

5.3 Grasping Sugar: Training and Rehabilitation

In this section, we apply our method to compare behavior over time. In particular, we show how the grasping action improves during training and how it changes during rehabilitation, depending on the treatment. To study the improvement, we need a reference behavior to compare with. Therefore, we define a "baseline" behavior which is the best possible grasping skills, obtained at the end of training, right before the stroke, which we call "0d".

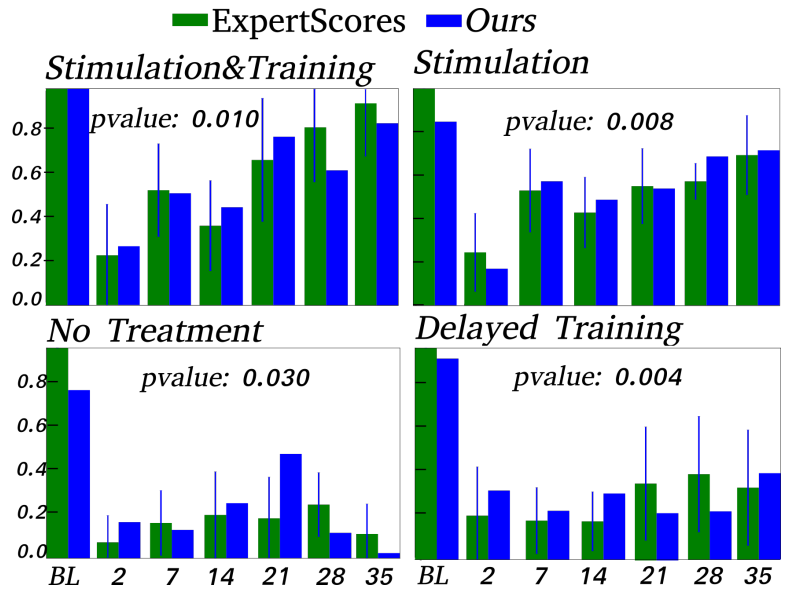

Figure 5.5: Comparing our model with ground-truth scores [3] provided by the experts. The bar-plot shows the healthiness score (higher is healthier). Our score is computed using a classifier to predict the ratio of healthy sequences per cohort. The results of our model correlate with GT at 0.933 ± 0.005.

Train to Grasp. Fig. 5.6 shows, every four days, the average similarity of all subjects to baseline. The figure shows that the improvement is continues as expected, suggesting that our encoding is providing a good representation.

Additionally, we can identify for each time point of learning how the behavior differs from the reference: the posture representation allows to spot postures (rows in Fig. 5.7) that are significantly over-represented in contrast to the skilled reference (red) as well as the ones that are missing (blue) at each point in time. Here the postures are mapped from the multidimensional E_π to 1D on the y-axis using t-SNE. The result shows that non-grasping postures (bottom) are more frequent in early stages while grasping postures (top), which precisely target the sugar pellet, are unlikely. During learning, the posture distribution then converges to the skilled reference baseline.

Therapies Comparison. An objective measure of behavior is crucial for testing drugs and treatments. In Fig. 5.8, we compare the effect of different treatments on the impaired rat respect to our reference baseline behavior. Specifically, the animals are divided into four treatment cohorts:

- "Stimulation and Training": a combination of optogenetic stimulation and physical treatments are used during rehabilitation.
- "Stimulation": only the optogenetic stimulation are used on the subjects, without physical treatment.
- "Delayed Training": only a physical treatment after a couple of weeks from the stroke.

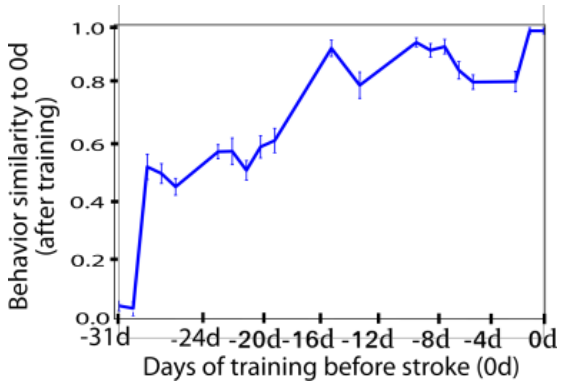

Figure 5.6: Similarity to skilled animals (0d) during training. The similarity to trained animals gradually increases over successive training days.

- "No Treatment": as the name suggests, these subjects have received no treatments.

We expected the first category to obtain the best recovery result.

Fig. 5.8 shows the similarity to a healthy baseline (left) and to impaired reference (right), immediately post-stroke, for each week of recovery. As expected, animals treated with optogenetic stimulation (green, blue) steadily improve during rehabilitation and having almost no similarity with the post-stroke behavior. In contrast, groups with no treatment (red) or only rehabilitative training (orange) reveal behavior that is similar to neither reference (top, center), suggesting an inadequate compensation differing significantly from the true recovery of impaired function.

The experiments show the behavior encoding E_β to be an effective means to compare different therapies after a disease and to diagnose the resulting changes in motor function.

5.4 Study Brain Through Behavior

As mentioned in Sec. 5.1.2, body motion is the output of complex brain functionality. Therefore, by looking at the behavior, we can predict the status of the brain. In this section, we propose two experiments where we correlate the behavior encoded in our unsupervised representation, with some information describing the brain status.

5.4.1 Neuronal Rewiring Correlation

For this experiment, we look at the brain "offline", after the animal is deceased. In particular, we counted the BDA positive fibers [159] in the damaged brain hemisphere post-mortem. The higher is the count, the more rehabilitated was the animal after treatment. Fig. 5.9 shows two brain slices where the count is low (left) and high (right) in the damaged hemisphere. This count, which describes the status of the brain after rehabilitation, is compared to the healthiness score defined in Sec. 5.2.2 for the last day of recordings. The plot shows a significant

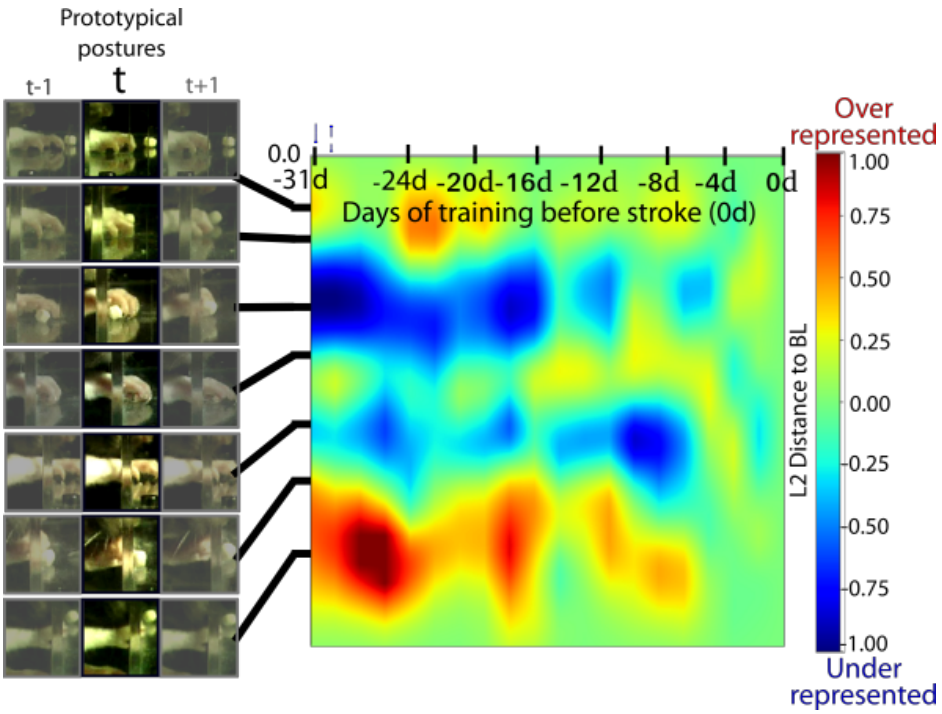

Figure 5.7: Relative frequency of individual postures per day of training compared to trained animals (0d). green: same frequency, red: more frequent, and blue: less frequent than day 0. Examples of postures are shown on the left (t column), including temporal context ($t-1$ and $t+1$ columns) to better understand the natural ordering of the postures along the vertical axis. The plot shows that correct hand closure at grasping occurs significantly less early in training.

correlation of $r \sim 0.7$ between the healthiness measurement based on our visual similarity and the fiber count in the brain.

The results in Fig. 5.9 show that our behavior study could provide an alternative to the highly invasive brain study.

5.4.2 Optogenetic Stimulation

While the previous experiment was a study of the brain "offline", in this paragraph we study the changes of behavior "online". In particular, we use a technique called "Optogenetics" to deactivate targeted brain functions in-vivo, basically simulating a stroke, which slightly alter the motor skills. The effects are reversible the moment the laser is turned off. For this experiment, we point the laser at three positions in the motor cortex. By simultaneously measuring the resulting changes in behavior we can quantify the importance of individual cortical circuits for a particular motor function.

The animals are divided into two cohorts:

- "Treatment": the animals are treated in a way that the laser affects the brain.
- "Control": the animals did not receive any treatment and the laser should

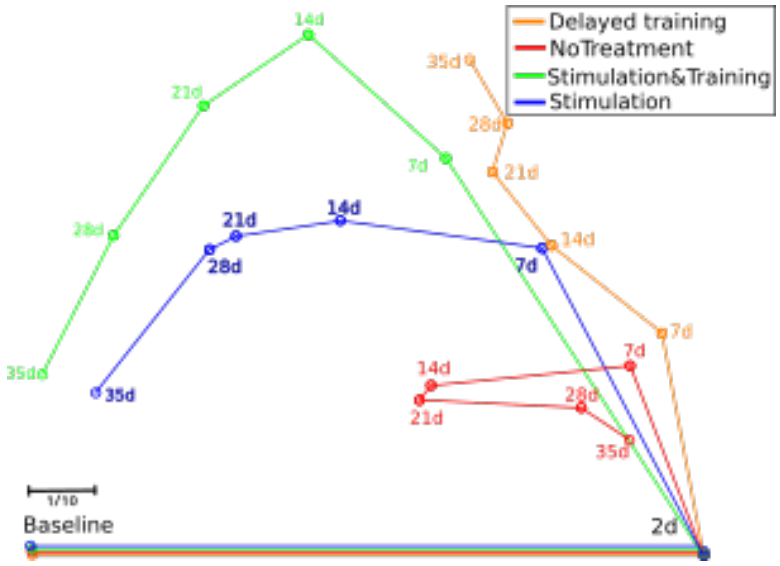

Figure 5.8: Comparison of different treatments on the rats' dataset during 35 days of rehabilitation. The x-axes measure behavior similarity to healthy baseline (left) and 2days post-stroke impaired (right). Rehabilitation successfully restores skilled motor function to bring behavior close to pre-stroke for the treatment cohorts (green, blue). Without treatment (orange, red), the behavior is altered, but still remains far from baseline, indicating inadequate compensation of motor-function.

have no effect on them.

The experiment evaluates whether based on our behavior representation a classifier can predict if optogenetics was perturbing a grasp (Fig. 5.10). As expected, the classifier only performs at chance level for controls and is significantly better ($68 \pm 9\%$ test accuracy per grasp) to discriminate light-perturbed behavior in treated animals. Only for these, it can recognize the light-driven modification of specific motor functions. Since such altered behavior is not present in every trial of a grasp, the goal had to be a significant improvement over control, but not finding differences in every grasp.

5.5 Summary

In this chapter, we utilize our unsupervised visual similarity as a tool for representing the animal behavior. Behavior analysis is a major instrument for the objective comparison of rehabilitative treatments and drugs, as well as evaluating the status of a disease.

For the behavior analysis to be effective, the visual similarity needs to be detailed and subject agnostic. Therefore, in Sec. 5.2.2 we evaluated our method before moving on to the application. Our approach was then tested on two tasks: studying the change of behavior over time and predicting the brain status by solely looking at the animal movement. The results show that our behavior similarity

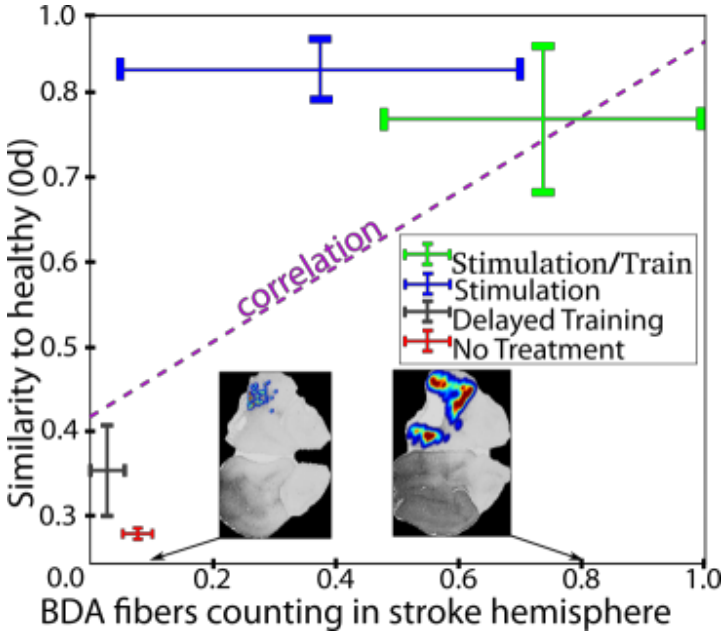

Figure 5.9: Relation between cortical rewiring (number of BDA positive fibers post-mortem in the hemisphere affected by the stroke, horizontal axis) and our behavior representation (vertical). Two examples, low and high fiber density, are shown. The experiment indicates that the behavior and rewiring are correlated ($p \sim 0.7$).

can be used to compare the effect of different treatments during rehabilitation and could substitute invasive measures to "look into the brain".

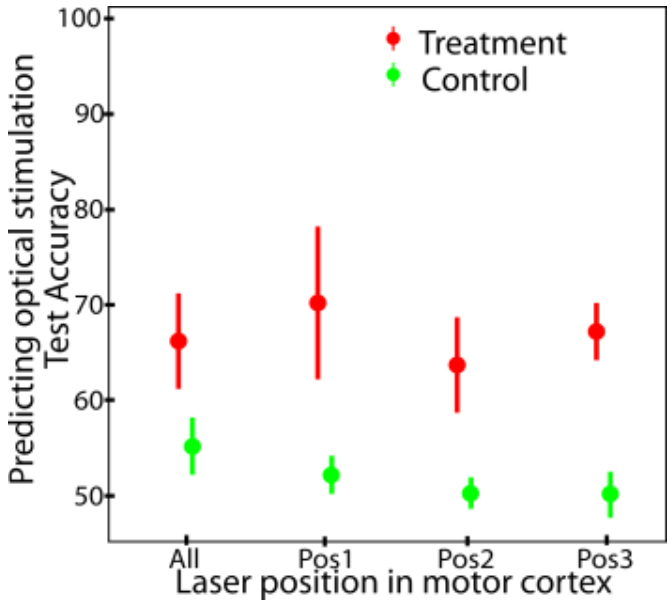

Figure 5.10: In-vivo deactivation of brain function in the motor cortex using optogenetic stimulation triggers subtle, reversible changes in behavior. Only based on our behavior representation, a classifier is then able to predict for test sequences whether there was optical stimulation. As would be expected, the classifier only achieves chance level for a control cohort but it performs significantly better ($68 \pm 9\%$ accuracy) on the treated animals.

Chapter 6

Conclusion

This thesis proposed several approaches to learn a visual representation for image and video similarity in two scenarios with limited supervision. In this chapter, after summarizing the thesis, we draw the conclusion and propose some interesting directions for future work.

6.1 Summary

Chapter 1 At the beginning of this thesis, we introduced the benefits of using deep learning for computer vision and its limitations. In particular, we identify two major issues both originating from the need of human supervision.
The first weakness is the representation of the visual world. Humans tend to group every object in the image in predefined, artificial categories. This view is reductive since it ignores many characteristics of the object, such as function and aspect. In our work, we relax the categorical assumption by learning visual similarities instead, therefore classifying objects only relative to each other, avoiding artificial classes.
The second weakness is that annotating training samples is very costly, in particular for more complex tasks (e.g. object detection and segmentation). If we could bypass the annotation bottleneck, we could harvest the huge amount of data available nowadays to produce very powerful visual representation. In this thesis, we investigated two setup in which human supervision is limited: zero-shot learning (ZSL) and self-supervision.
We then talked about the important ability of transferring the knowledge from a task to another, called transfer learning, intrinsic of deep learning. Transfer learning is crucial when it comes to limited supervision since we can solve a task where only few labels are available by leveraging a much large dataset.

Chapter 2 ZSL has proven to be an effective way under condition of limited supervision to only known object categories and generalize to unknown ones. ZSL is particularly effective when source and target distribution come from the same underlying data generator, for example objects such as faces, birds and cars. State-of-the-art models are trained using metric learning, which maximizes the similarity between images of the same category, while maximizing distance to samples from distinct categories. This discriminative approach, however, is very

simplistic and ignores the underling structure of the representation space, including similarity across categories, i.e. inter-class similarity. Many relaxation to this problem have proven successful over the past years. Nevertheless, the inter-class similarity is still being ignored. We show that including this type of information is very effective to boost any kind of metric learning objective. To capture inter-class characteristics, we developed an unsupervised method which removes class-specific features and searches for patterns across categories. For example, when training a representation over cars, no matter what the car model is, the network should know that object characteristics such as "wheels" and "doors" are shared among all categories.

Chapter 3 Given the success of ZSL in the image domain, we studied the problem also for video classification. The task is to produce a model that learns to compare actions in videos so that it can recognize new actions which have never seen during training. For example, given several actions during training, such as "playing football" and "playing piano", the model should recognize test categories such as "playing basketball" and "playing violin". In particular, we focus on human activities. Labelling videos is very time intensive relatively to images, therefore being able to only annotate part of the data and generalize to any new category could really benefit the field. Instead, the field is way behind: performances are very low and the yearly growth is very little. We investigated the causes of this slow progress and propose a new, strong baseline to help the field grow faster. In particular, we identify the major weaknesses in a lack of common benchmark due to non-reproducible models and ineffective use of the network potential, which is kept frozen after pre-training. Therefore, we propose a new baseline model which outperforms all previous approaches while being much simpler by using an end-to-end approach.

Chapter 4 As mentioned above, one goal of this thesis is to reduce the human supervision for training the visual representation. ZSL is a very effective approach to recognize new, unlabelled categories, however it still requires a large labelled dataset during training. The next step is to get rid of human annotation and for that self-supervision is a good candidate. The key of self-supervision is to train the visual representation using a surrogate task where labels are freely available. Typically, the surrogate task transforms the input sample and the network is tasked to recognize the transformation. The transformation is typically chosen randomly during training. We argue that the training can be optimized, therefore producing more robust visual representation, based on which transformation is chosen. Therefore, we propose a meta-learning approach to control the surrogate task during training to boost the visual representation strength even further. Our reinforcement learning controller can select the transformation based on the network status. Our approach was tested only on permutations, which is a very popular transformation for self-supervision. In future works, the RL controller could also be tested on other surrogate tasks, like rotation prediction, or even a multi-task combination of surrogate tasks.

Chapter 5 Finally, we tested a modern self-supervised task on a real-application. We used an unsupervised video sequence encoder to compare animal behavior. An objective metric of behavior is crucial for biomedical fields. For example, it could be used to compare treatments and rehabilitative drugs, or measuring the state of a disease that damages motor skills. However, manual annotation is costly and human bias might alter the model. Thus an unsupervised video representation is a good fit to solve the task. We tested this method on a dataset of rodents grasping a sugar pellet through a slit. Our behavior similarity was successful throughout a series of biomedical experiments. For example, we were able to detect the changes in behavior during training and rehabilitation, and we could correlate the animal movement with the brain status.

6.2 Discussion and Future Work

Discussion In this thesis, we investigated visual similarity as an alternative approach to artificial categories. In parallel, we studied deep learning models that require minimal to no supervision. The contribution of this thesis is to find and tackle common weaknesses of existing models and propose a real-world application.

The major issue with using limited supervision is the weak training signal. Since the model is trained on a task different to the downstream task, the visual representation is not guaranteed to generalize well across tasks. Fully supervised methods do not have this issue, since they are directly trained on the final task. Previous works focused on closing the gap between these two types of signal. This work has shown that it is still possible to improve upon state-of-the-art by proposing alternative approaches to boost existing methods. We also have shown how to use visual similarity in combination with a fully unsupervised approach for behavior encoding. This approach can be adapted to other applications, such as video anomaly detection or performance evaluation in sports.

Future work Fully supervised methods are easy to train, but require a fully annotated dataset. Unsupervised methods require no human annotations, but might not generalize well on the downstream task. Major improvements have been done by the research community in ZSL and self-supervision over the past few years. However, it has been shown that the yearly improvement is converging. Therefore, we seek new paradigms.

For future work, a hybrid solution is key: use self-supervision on a large collection of unlabelled data while, in parallel, train on a few labelled samples. In this way, we can exploit the strengths of both methods, and reduce their weaknesses. This is a form of semi-supervised learning that is receiving a lot of attention lately because of its promising performances.

Publications

This dissertation has led to the following scientific publications:

- Brattoli B., Büchler U., Wahl AS, Schwab ME, Ommer B., LSTM Self-Supervision for Detailed Behavior Analysis. In IEEE Computer Vision and Pattern Recognition (CVPR), 2017.
- Brattoli B., Büchler U., Ommer B. Improving, Spatiotemporal Self-Supervision by Deep Reinforcement Learning. In European Conference on Computer Vision (ECCV), 2018.
- Brattoli B., Roth K., Ommer B., MIC: Mining Inter-class Characteristics for Improving Metric Learning. In IEEE International Conference on Computer Vision (ICCV), 2019.
- Brattoli B., Tighe J., Zhdanov F., Perona P., Chalupka K., Rethinking Zero-shot Video Classification: End-to-end Training for Realistic Applications. In IEEE Computer Vision and Pattern Recognition (CVPR), 2020.

The following publication is currently under submission:

- Brattoli B., Büchler U., Dorkenwald M., Reiser P., Filli L., Helmchen F., Wahl AS, Ommer B., UBAM: Unsupervised Behavior Analysis and Magnification.

Bibliography

[1] Artificial intelligence and the future of cybersecurity. *Proceedings of the ACM Conference on Computer and Communications Security*, 10 2011.

[2] A. Achille, M. Lam, R. Tewari, A. Ravichandran, S. Maji, C. C. Fowlkes, S. Soatto, and P. Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6430–6439, 2019.

[3] M. Alaverdashvili and I. Q. Whishaw. A behavioral method for identifying recovery and compensation: hand use in a preclinical stroke model using the single pellet reaching task. *Neuroscience & Biobehavioral Reviews*, 37(5):950–967, 2013.

[4] I. Alexiou, T. Xiang, and S. Gong. Exploring synonyms as context in zero-shot action recognition. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 4190–4194. IEEE, 2016.

[5] L. Alvarez, J. Weickert, and J. Sánchez. Reliable estimation of dense optical flow fields with large displacements. *International Journal of Computer Vision*, 39(1):41–56, 2000.

[6] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[7] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

[8] A. Arac, P. Zhao, B. H. Dobkin, S. T. Carmichael, and P. Golshani. Deepbehavior: A deep learning toolbox for automated analysis of animal and human behavior imaging data. *Frontiers in systems neuroscience*, 13:20, 2019.

[9] Y. Bai, F. Gao, Y. Lou, S. Wang, T. Huang, and L. Duan. Incorporating intra-class variance to fine-grained visual recognition. *CoRR*, abs/1703.00196, 2017.

[10] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. In *Readings in computer vision*, pages 714–725. Elsevier, 1987.

[11] M. A. Bautista, A. Sanakoyeu, and B. Ommer. Deep unsupervised similarity learning using partially ordered sets. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, 2017.

[12] M. A. Bautista, A. Sanakoyeu, E. Tikhoncheva, and B. Ommer. Cliquecnn: Deep unsupervised exemplar learning. In *Advances in Neural Information Processing Systems*, pages 3846–3854, 2016.

[13] S. Beery, G. Van Horn, and P. Perona. Recognition in terra incognita. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[14] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[15] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[16] G. J. Berman. Measuring behavior across scales. *BMC biology*, 16(1):23, 2018.

[17] G. J. Berman. Measuring behavior across scales. *BMC biology*, 16(1):23, 2018.

[18] M. Bishay, G. Zoumpourlis, and I. Patras. Tarn: Temporal attentive relation network for few-shot and zero-shot action recognition. *arXiv preprint arXiv:1907.09021*, 2019.

[19] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[20] P. Bojanowski and A. Joulin. Unsupervised learning by predicting noise. *arXiv preprint arXiv:1704.05310*, 2017.

[21] B. Brattoli, U. Büchler, A. S. Wahl, M. E. Schwab, and B. Ommer. Lstm self-supervision for detailed behavior analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] B. Brattoli, J. Tighe, F. Zhdanov, P. Perona, and K. Chalupka. Rethinking zero-shot video classification: End-to-end training for realistic applications. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2020.

[23] U. Büchler, B. Brattoli, and B. Ommer. Improving spatiotemporal self-supervisionby deep reinforcement learning. In *IEEE Conference on European Conference on Computer Vision (ECCV)*, 2018.

[24] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. *CoRR*, abs/1807.05520, 2018.

[25] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[26] H.-S. Chang, E. Learned-Miller, and A. McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems*, pages 1003–1013, 2017.

[27] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[28] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings*, pages 539–546. IEEE, 2005.

[29] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[30] R. S. Cruz, B. Fernando, A. Cherian, and S. Gould. Deeppermnet: Visual permutation learning. In *CVPR*, 2017.

[31] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[32] F. De Chaumont, R. D.-S. Coura, P. Serreau, A. Cressant, J. Chabout, S. Granon, and J.-C. Olivo-Marin. Computerized video analysis of social interactions in mice. *Nature methods*, 9(4):410, 2012.

[33] A. I. Dell, J. A. Bender, K. Branson, I. D. Couzin, G. G. de Polavieja, L. P. Noldus, A. Pérez-Escudero, P. Perona, A. D. Straw, M. Wikelski, et al. Automated image-based tracking and its application in ecology. *Trends in ecology & evolution*, 29(7):417–428, 2014.

[34] R. Dey and F. M. Salemt. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

[35] E. Diamant. Modeling visual information processing in brain: a computer vision point of view and approach. In *International Symposium on Brain, Vision, and Artificial Intelligence*, pages 62–71. Springer, 2007.

[36] J. J. DiCarlo, D. Zoccolan, and N. C. Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

[37] C. B. Do and A. Y. Ng. Transfer learning for text classification. In *Advances in Neural Information Processing Systems*, pages 299–306, 2006.

[38] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[39] C. Doersch and A. Zisserman. Multi-task self-supervised visual learning. *arXiv preprint arXiv:1708.07860*, 2017.

[40] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[41] Y. Duan, W. Zheng, X. Lin, J. Lu, and J. Zhou. Deep adversarial metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[42] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[43] 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.

[44] B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015.

[45] Y. Fan, F. Tian, T. Qin, X.-Y. Li, and T.-Y. Liu. Learning to teach. In *International Conference on Learning Representations*, 2018.

[46] Y. Fan, F. Tian, T. Qin, and T.-Y. Liu. Neural data filter for bootstrapping stochastic gradient descent. *ICLR*, 2016.

[47] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. *arXiv preprint arXiv:1812.03982*, 2018.

[48] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[49] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[50] L. Filli, T. Sutter, C. S. Easthope, T. Killeen, C. Meyer, K. Reuter, L. Lörincz, M. Bolliger, M. Weller, A. Curt, et al. Profiling walking dysfunction in multiple sclerosis: characterisation, classification and progression over time. *Scientific reports*, 8(1):4984, 2018.

[51] D. A. Forsyth and J. Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[52] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[53] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, Jan. 2016.

[54] W. Ge. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–285, 2018.

[55] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018.

[56] A. Gomez-Marin, N. Partoune, G. J. Stephens, and M. Louis. Automated tracking of animal posture and movement during exploration and sensory orientation behaviors. *PloS one*, 7(8):e41642, 2012.

[57] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[58] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[59] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017.

[60] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003*, 2017.

[61] M. Hahn, A. Silva, and J. M. Rehg. Action2vec: A crossmodal embedding approach to action learning. *arXiv preprint arXiv:1901.00484*, 2019.

[62] K. Hara, H. Kataoka, and Y. Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

[63] B. Harwood, B. Kumar, G. Carneiro, I. Reid, T. Drummond, et al. Smart mining for deep metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2821–2829, 2017.

[64] J. Hays and A. Efros. Where in the world? human and computer geolocation of images. *Journal of Vision*, 9(8):969–969, 2009.

[65] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[66] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[67] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

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

[69] Y. Huang, M. Kaufmann, E. Aksan, M. J. Black, O. Hilliges, and G. Pons-Moll. Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 37:185:1–185:15, Nov. 2018. Two first authors contributed equally.

[70] H. Idrees, A. R. Zamir, Y.-G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155:1–23, 2017.

[71] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[72] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Mining on manifolds: Metric learning without labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7642–7651, 2018.

[73] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[74] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2011.

[75] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[76] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[77] M. Kabra1, A. A. Robie1, M. Rivera-Alba1, S. Branson, and K. Branson. Jaaba: interactive machine learning for automatic annotation of animal behavior. *Nature methods*, 10, 2012.

[78] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[79] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[80] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[81] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[82] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015.

[83] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[84] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[85] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[86] J. K. Kruschke. Alcove: an exemplar-based connectionist model of category learning. *Psychological review*, 99(1):22, 1992.

[87] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[88] M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2010.

[89] H. Larochelle, D. Erhan, and Y. Bengio. Zero-data learning of new tasks. In *AAAI*, 2008.

[90] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. *arXiv preprint arXiv:1703.04044*, 2017.

[91] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[92] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[93] H.-Y. Lee, J.-B. Huang, M. K. Singh, and M.-H. Yang. Unsupervised representation learning by sorting sequences. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[94] X. Lin, Y. Duan, Q. Dong, J. Lu, and J. Zhou. Deep variational metric learning. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[95] H. Liu, Y. Tian, Y. Wang, L. Pang, and T. Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2167–2175, 2016.

[96] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[97] M. M. Loper, N. Mahmood, and M. J. Black. MoSh: Motion and shape capture from sparse markers. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 33(6):220:1–220:13, Nov. 2014.

[98] D. G. Lowe. Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image, Mar. 23 2004. US Patent 6,711,293.

[99] Z. Luo, B. Peng, D.-A. Huang, A. Alahi, and L. Fei-Fei. Unsupervised learning of long-term motion dynamics for videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[100] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[101] C. Manning, P. Raghavan, and H. Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010.

[102] B. Martinez, D. Modolo, Y. Xiong, and J. Tighe. Action recognition with spatial-temporal discriminative filter banks. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[103] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 21(9):1281–1289, 9 2018.

[104] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[105] D. L. Medin and M. M. Schaffer. Context theory of classification learning. *Psychological review*, 85(3):207, 1978.

[106] G. Mesnil, A. Bordes, J. Weston, G. Chechik, and Y. Bengio. Learning semantic representations of objects and their parts. *Machine learning*, 94(2):281–301, 2014.

[107] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[108] T. Milbich, M. Bautista, E. Sutter, and B. Ommer. Unsupervised video understanding by reconciliation of posture similarities. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[109] A. Mishra, V. K. Verma, M. S. K. Reddy, S. Arulkumar, P. Rai, and A. Mittal. A generative approach to zero-shot and few-shot action recognition. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 372–380. IEEE, 2018.

[110] I. Misra, C. L. Zitnick, and M. Hebert. Unsupervised learning using sequential verification for action recognition. In *IEEE European Conference on Computer Vision (ECCV)*, 2016.

[111] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[112] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 360–368, 2017.

[113] G. Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*, 2017.

[114] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *IEEE European Conference on Computer Vision (ECCV)*, 2016.

[115] M. Noroozi, H. Pirsiavash, and P. Favaro. Representation learning by learning to count. *arXiv preprint arXiv:1708.06734*, 2017.

[116] M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash. Boosting self-supervised learning via knowledge transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[117] R. M. Nosofsky. Attention, similarity, and the identification–categorization relationship. *Journal of experimental psychology: General*, 115(1):39, 1986.

[118] H. Oh Song, S. Jegelka, V. Rathod, and K. Murphy. Deep metric learning via facility location. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5382–5390, 2017.

[119] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016.

[120] M. Opitz, G. Waltner, H. Possegger, and H. Bischof. Bier-boosting independent embeddings robustly. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5189–5198, 2017.

[121] M. Opitz, G. Waltner, H. Possegger, and H. Bischof. Deep metric learning with bier: Boosting independent embeddings robustly. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[122] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, pages 1410–1418, 2009.

[123] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[124] Y. Patel, L. Gomez, M. Rusiñol, C. Jawahar, and D. Karatzas. Self-supervised learning of visual features through embedding images into text topic spaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[125] A. Pérez-Escudero, J. Vicente-Page, R. C. Hinz, S. Arganda, and G. G. De Polavieja. idtracker: tracking individuals in a group by automatic identification of unmarked animals. *Nature methods*, 11(7):743, 2014.

[126] S. M. Peters, I. J. Pinter, H. H. Pothuizen, R. C. de Heer, J. E. van der Harst, and B. M. Spruijt. Novel approach to automatically classify rat social behavior using a video tracking system. *Journal of neuroscience methods*, 268:163–170, 2016.

[127] A. Piergiovanni and M. S. Ryoo. Learning shared multimodal embeddings with unpaired data. *CoRR*, 2018.

[128] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[129] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. *ICLR*, 2016.

[130] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[131] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[132] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[133] A. A. Robie, K. M. Seagraves, S. R. Egnor, and K. Branson. Machine vision methods for analyzing social interactions. *Journal of Experimental Biology*, 220(1):25–34, 2017.

[134] A. Roitberg, Z. Al-Halah, and R. Stiefelhagen. Informed democracy: voting-based novelty detection for action recognition. *arXiv preprint arXiv:1810.12819*, 2018.

[135] A. Roitberg, M. Martinez, M. Haurilet, and R. Stiefelhagen. Towards a fair evaluation of zero-shot action recognition using external data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[136] K. Roth, B. Brattoli, and B. Ommer. Mic: Mining interclass characteristics for improved metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8000–8009, 2019.

[137] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[138] S. Russell. Artificial intelligence: The future is superintelligent. *Nature*, 548(7669):520, 2017.

[139] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2010.

[140] A. Sanakoyeu, M. A. Bautista, and B. Ommer. Deep unsupervised learning of visual similarities. *Pattern Recognition*, 78:331–343, 2018.

[141] A. Sanakoyeu, V. Tschernezki, U. Büchler, and B. Ommer. Divide and conquer the embedding space for metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[142] N. Sayed, B. Brattoli, and B. Ommer. Cross and learn: Cross-modal self-supervision. In *German Conference on Pattern Recognition (GCPR)*, 2018.

[143] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[144] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[145] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[146] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Conference on Neural Information Processing Systems (NIPS)*, 2014.

[147] B. Singh, M. Najibi, and L. S. Davis. Sniper: Efficient multi-scale training. In *Advances in Neural Information Processing Systems*, pages 9310–9320, 2018.

[148] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016.

[149] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[150] Ö. Sümer, T. Dencker, and B. Ommer. Self-supervised learning of pose embeddings from spatiotemporal relations in videos. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 4308–4317. IEEE, 2017.

[151] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[152] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[153] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[154] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[155] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems*, pages 4170–4178, 2016.

[156] C. E. Vargas-Irwin, G. Shakhnarovich, P. Yadollahpour, J. M. Mislow, M. J. Black, and J. P. Donoghue. Decoding complete reach and grasp actions from local primary motor cortex populations. *Journal of neuroscience*, 30(29):9659–9669, 2010.

[157] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Conference on Neural Information Processing Systems (NIPS)*, 2016.

[158] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. *Computation Neural Systems Technical Report*, 2011.

[159] A.-S. Wahl, U. Büchler, A. Brändli, B. Brattoli, S. Musall, H. Kasper, B. V. Ineichen, F. Helmchen, B. Ommer, and M. E. Schwab. Optogenetically stimulating intact rat corticospinal tract post-stroke restores motor control through regionalized functional circuit formation. *Nature communications*, 8(1):1187, 2017.

[160] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin. Deep metric learning with angular loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2593–2601, 2017.

[161] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Val Gool. Temporal segment networks: Towards good practices for deep action recognition. In *IEEE European Conference on Computer Vision (ECCV)*, 2016.

[162] Q. Wang and K. Chen. Alternative semantic representations for zero-shot human action recognition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 87–102. Springer, 2017.

[163] Q. Wang and K. Chen. Zero-shot visual recognition via bidirectional latent embedding. *International Journal of Computer Vision*, 124(3):356–383, 2017.

[164] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015.

[165] X. Wang, K. He, and A. Gupta. Transitive invariance for self-supervised visual representation learning. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[166] R. J. Williams and J. Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

[167] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017.

[168] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492. IEEE, 2010.

[169] M. Xie. Development of artificial intelligence and effects on financial system. In *Journal of Physics: Conference Series*, volume 1187, page 032084. IOP Publishing, 2019.

[170] D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie, and Y. Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10334–10343, 2019.

[171] X. Xu, T. Hospedales, and S. Gong. Semantic embedding space for zero-shot action recognition. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 63–67. IEEE, 2015.

[172] X. Xu, T. Hospedales, and S. Gong. Transductive zero-shot action recognition by word-vector embedding. *International Journal of Computer Vision*, 123(3):309–333, 2017.

[173] X. Xu, T. M. Hospedales, and S. Gong. Multi-task zero-shot action recognition with prioritised data augmentation. In *European Conference on Computer Vision*, pages 343–359. Springer, 2016.

[174] H. Xuan, R. Souvenir, and R. Pless. Deep randomized ensembles for metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 723–734, 2018.

[175] Y. Yuan, K. Yang, and C. Zhang. Hard-aware deeply cascaded embedding. In *Proceedings of the IEEE international conference on computer vision*, pages 814–823, 2017.

[176] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[177] B. Zhang, H. Hu, and F. Sha. Cross-modal and hierarchical modeling of video and text. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 374–390, 2018.

[178] C. Zhang and Y. Peng. Visual data synthesis via gan for zero-shot video classification. *arXiv preprint arXiv:1804.10073*, 2018.

[179] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.

[180] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. *arXiv preprint arXiv:1611.09842*, 2016.

[181] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[182] Y. Zhao, Z. Jin, G.-j. Qi, H. Lu, and X.-s. Hua. An adversarial approach to hard triplet generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 501–517, 2018.

[183] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.

[184] Y. Zhu, Y. Long, Y. Guan, S. Newsam, and L. Shao. Towards universal representation for unseen action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9436–9445, 2018.

[185] S. Q. X. W. Ziwei Liu, Ping Luo and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[186] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.