POLITECNICO DI TORINO

Master Degree in Computer Engineering

Master Thesis

Source-Free Domain Adaptation for Video Action Recognition

Supervisor
Prof.ssa Barbara Caputo

Co-supervisors:
Dott. Mirco Planamente
Dott.ssa Chiara Plizzari

Candidate
Julian Neubert

March 2023

Abstract

With the advent of media streaming, Action Recognition (AR) has become progressively important for various applications such as intelligent video surveillance, sports analysis, and human-computer interaction. An open challenge is that video analysis systems heavily rely on the environment in which the activities are recorded, which inhibits their ability to recognize actions when they are recorded in unfamiliar surroundings or different lighting conditions. This problem, known in the literature as domain shift, has long been studied in the image classification domain and has recently started to attract attention also in the activity classification community. Most of the researchers in the field addressed this issue by reducing the problem to an unsupervised domain adaptation (UDA) setting, where an unlabeled set of samples from the target is available during training. However, nowadays data are distributed on different devices and usually contain private information, e.g., those on personal phones or from surveillance cameras. Existing UDA methods need to access the source data during learning to adapt, which is not efficient for data transmission and may violate the data privacy policy.

In this thesis, we address an interesting but challenging unsupervised DA setting with only a trained source model provided as supervision, which is referred to as Source-Free Domain Adaptation (SFDA). We propose a solution that aims to enhance a model's robustness to distribution shift by working on the feature space. The key insight is that widespread pretrained feature extractors are able to extract general domain-independent features, while most of the performance degradation stems from application-specific classification layers. Thus, we compute a pseudo-prototype representation of each class from unlabeled target data during the adaptation step. This allows us to classify samples based on their similarity to the prototypes, thereby improving the model's applicability to unseen domains. Our method does not modify any network parameters, resulting in minimal computational overhead.

We evaluate the efficacy of our method on the two most established benchmarks for video action recognition, namely UCF-HMDB and EPIC-KITCHENS. As video data is usually associated with multi-modal information, e.g., RGB, optical flow, and audio, we also investigate how to apply our method across various information channels of videos to increase consistency across individual predictions and make our model overall more robust. Results show our method stably improves performance on unseen domains, outperforming existing source-free adaptation methods by a large margin.

Contents

List of Tables III

List of Figures IV

1 Introduction 1
1.1 Research Goals and Contribution 3

2 Background 5
2.1 Deep Learning 5
2.1.1 Perceptrons 6
2.1.2 Multi-Layer Perceptrons 7
2.1.3 Activation Functions 8
2.1.4 Backpropagation 10
2.1.5 Loss Functions 12
2.1.6 Regularization 13
2.1.7 Normalization 16
2.2 Convolutional Neural Networks 18
2.2.1 Convolutions 18
2.2.2 Pooling 19
2.2.3 Architectures 20
2.3 Sequential Data 21
2.3.1 Recurrent Neural Networks 21
2.3.2 Transformers 23

3 Related Works 25
3.1 Action recognition 25
3.1.1 Egocentric Action Recognition 25
3.1.2 Temporal Information 26
3.1.3 Multi-Modal Data 28
3.1.4 Datasets 30
3.2 Cross-Domain Learning 33
3.2.1 Domain Generalization 33

3.2.2 Unsupervised Domain Adaptation 34
3.2.3 Source-Free Domain Adaptation 39
3.2.4 Test-Time Adaptation 41

4 Proposed Solution 43
4.1 Motivation 44
4.2 Predictions from Pseudo-Prototypes 45
4.3 Problem Formulation 47

5 Experiments 49
5.1 Implementation Details 49
5.2 Comparison with the State-of-the-Art 50
5.2.1 Comparison with Test-Time Adaptation 51
5.2.2 Architecture-Independent Relative Norm Alignment 54
5.3 Class Imbalance 55
5.3.1 Imbalance in the Dataset 57
5.3.2 Imbalance in the Classifier 57
5.4 Multi-Modal Learning 58
5.5 Ablation Study 59
5.5.1 Which is the best feature? 59
5.5.2 Increasing Centroid Robustness by Filtering Samples 62
5.5.3 Can centroids be improved iteratively? 62
5.5.4 How do we measure similarity between features? 62

6 Conclusion 65

Bibliography 66

List of Tables

4.1 Different settings in domain adaptation 44
5.1 Comparison with SOTA on UCF-HMDB$_{full}$ 50
5.2 Comparison with SOTA on EPIC-Kitchens-55 51
5.3 Architecture-indpendent comparison on UCF-HMDB$_{full}$ 54
5.4 Architecture-indpendent comparison on EPIC-Kitchens-55 54
5.5 Synergies with MTRAN* on EPIC-Kitchens-55 55
5.6 Average accuracy on EPIC-Kitchens-55 56
5.7 Average per-class accuracy on EPIC-Kitchens-55 56

List of Figures

1.1 Cross-Domain Performance Loss 2
1.2 Domain adaptation 3
2.1 Perceptron 7
2.2 Multi-Layer Perceptron 8
2.3 Activation Functions 11
2.4 Early stopping 14
2.5 Image Augmentations 15
2.6 Dropout 16
2.7 Normalization Layers 18
2.8 Convolution 19
2.9 Pooling 20
2.10 Recurrent Neural Network Architectures 22
2.11 LSTM cell 23
3.1 EPIC Kitchens 26
3.2 Inflated Inception-V1 Architecture 27
3.3 Modalities 29
3.4 Optical Flow 30
3.5 Spectrogram 31
3.6 HMDB51 & UCF101 32
3.7 EPIC-Kitchens-100 Distribution of Classes 33
3.8 Deep Adaptation Network 34
3.9 Domain Adversarial Neural Network 35
3.10 Multi-Modal Self-Supervised Domain Adaptation 36
3.11 Cross-modal Interactive Alignment 38
3.12 Transfer Sequential Variational Autoencoder 39
3.13 Attentive Temporal Consistency Network 41
3.14 Multimodal and Temporal Relative Alignment Network 42
5.1 Accuracy of T3A over share of test dataset seen 52
5.2 Architecture-Independent Comparison 55
5.3 Class Imbalance in EPIC-Kitchens-55 56
5.4 Class Distributions and Classifier Bias 58
5.5 The Impact of Multi-Modal Training 59

5.6 Ablation over Feature Selection 60
5.7 Ablation over the Number of Samples 61
5.8 Ablation over the Number of k-means Iterations 63
5.9 Ablation over the Similarity Measure in the Feature Space 64

Chapter 1

Introduction

Artificial intelligence has long been fascinating people and sparked their imagination. With the advent of deep learning the field has seen rapid development, making intelligent systems a reality in our everyday lives. Virtual assistants can be controlled with our voice and understand complex commands, computational photography helps us take better pictures, and recommendation systems provide us with suggestions on which shows to watch and what music to listen to.

Computer Vision has been a central field in driving the progress of deep learning. It consists of designing algorithms and techniques that allow computers to extract high-level information from visual data, such as images or videos. Typical tasks include classification, object detection, segmentation, instance retrieval, and image generation. State-of-the-art models are nowadays capable of super-human performance on some of these tasks, most notably image classification, where models recognize and classify images more accurately than humans.

However, despite recent advancements more complex problems still remain a challenging area of research. One of those is applying deep learning for video understanding. Videos are significantly more complex than images. First, they come with additional information on the temporal dimension, encoding motion information that is not represented in static images. Incorporating temporality into the learning process is crucial for most video-related tasks. For instance, many human activities consist of a series of individual motions, where the order of these motions defines the overall action. However, adding this extra temporal information significantly increases computational complexity. In fact, modern 3D CNN video architectures have several times more parameters than 2D CNNs for image understanding. Secondly, videos encode additional information on multiple information channels. Alternative modalities, such as audio track or apparent motion (optical flow), can assist in identifying the content of a video, providing valuable clues about objects and their interactions. Nonetheless, multi-modal learning presents its own set of challenges, introducing noise into the training process, which can lead to reduced performance on each modality individually.

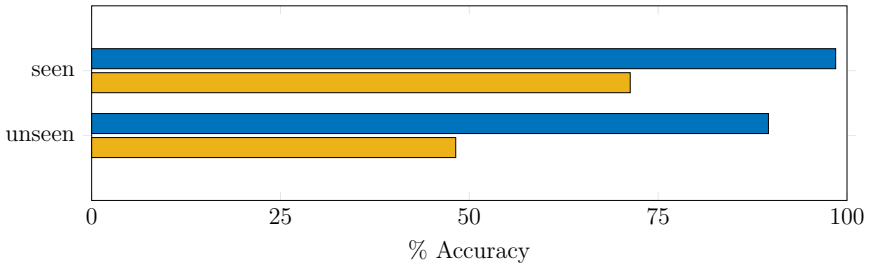


Figure 1.1: Accuracy on **UCF-HMDB**$_{full}$ and **EPIC-Kitchens-55** when testing in a familiar "seen" environment and an "unseen" domain.

Another key challenge across all machine learning algorithms is their vulnerability to the domain shift, which occurs when there is a discrepancy in terms of data distribution difference between the training and evaluation data. For instance, this can be observed in images from a stylistic point of view, such as the difference between a *photograph* of a cat and a *sketch* of a cat. On videos, this problem is referred to as "environmental bias". This problem arises from the network's heavy reliance on the environment in which the activities are recorded, which inhibits the network's ability to recognize actions when they are conducted in unfamiliar (unseen) surroundings. For example, changes in illumination, viewpoint, object locations, and surroundings between training and test data have a drastic impact on the performance in video understanding tasks. To give an intuition of this performance drop, figure 1.1 highlights how much the accuracy of a state-of-the-art classifier decreases when applied to test data with a different distribution of the training data.

The most common approach to address this issue is Unsupervised Domain Adaptation (UDA), where a model is trained on both labeled data from the seen domain, which is referred to as source, and unlabeled data from the same distribution as the test set, which is referred to as target. The availability of information from target simplifies the adaptation process and offers interesting ways to study the domain gap from a theoretical perspective, leading to significant improvements over a model trained on source data only. However, requiring data from both domains is a strong limitation, hindering the usage of UDA methods in many real-world applications where privacy policies and proprietary licensing do not allow sharing data between model providers and users. Viable alternatives have been proposed, such as Domain Generalization (DG), where data from target is not available, and the model exploits multiple source domains to adapt to *any* unseen target distribution, or Test-Time Adaptation (TTA), where the model adapts on test samples on-the-fly during evaluation. However, both DG and TTA are subject to tight restrictions that limit their potential for improvement, particularly when the data has high variability and the source data is not representative [1, 2].

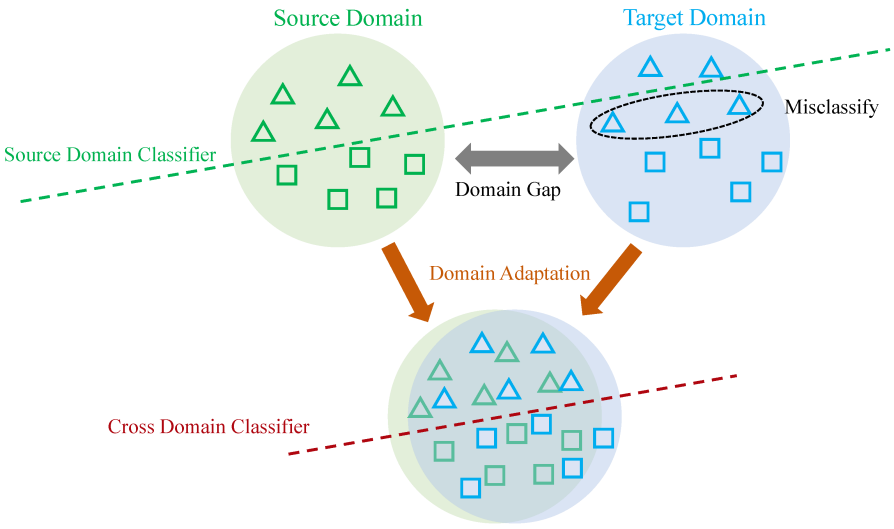


Figure 1.2: Domain adaptation aims to improve cross-domain performance [3].

In this thesis, we investigate the newest setting of Source-Free Domain Adaptation (SFDA). It requires adapting a model to a target domain without any access to information from the source domain. Due to its definition, it makes the most of the available data, integrating information from target while conforming to strict data-sharing limitations. We explore this setting on video understanding, which is one of the most challenging tasks nowadays, but it offers the playground for extensive analysis due to the complexity and richness of the data.

1.1 Research Goals and Contribution

In this thesis we propose a novel method to improve the cross-domain accuracy of action recognition models in the interesting, yet challenging SFDA setting. Compared to previous publications in this area our method has the key advantage of having no specific requirements for the model architecture, making it highly flexible and able to profit from future developments in action recognition architectures. Furthermore, our method is unique among UDA literature for action recognition in that it is completely backpropagation-free, directly data dependent, and does not introduce any learnable parameters, making it computationally efficient, able to adapt to multiple domains simultaneously, and avoiding many pitfalls associated with gradient-based training that often lead to degraded performance. These points play an especially important role in the context of wearable devices, a central field of application for video SFDA, where computational resources are limited and fast and reliable adaptation to new environments is crucial. Finally, our method only works on the classifier, taking extracted features as input, meaning it synergizes

well with competing SFDA methods that focus on improving the feature extractor.

To this end, we propose a pseudo-prototypical classification module to replace the final linear classification layer of the model, which is often biased and severely affected by domain shift. Each class is represented by a prototype, which is the mean feature vector of all samples from the target dataset belonging to that class, and classification is done based on the nearest centroid. Since data in the target domain is unlabeled, our method uses the predictions of the source model as pseudo labels to assign samples to a prototype's support set. To improve the quality and robustness of the centroids we use the average multi-modal feature vector over several clips to represent each sample, only consider features with low prediction entropy for the support set, and iteratively improve the centroids in a k-means-like fashion.

We evaluate the efficacy of our method on two of the most important datasets for this setting, UCF-HMDB$_{full}$ and EPIC-Kitchens-55. Since videos are generally associated with multi-modal information, we investigate how to apply our method to different information channels to increase consistency among individual predictions. Through a theoretical analysis, we investigate how domain shift affects linear classification layers and how our method circumvents this. Deriving from these observations we show how our method increases fairness by specifically improving accuracy on underrepresented classes. Furthermore, we show how our method benefits from previous achievements in video SFDA, by combining our classification module with an adapted feature extractor to further improve cross-domain accuracy. Comparing our results with the current state-of-the-art shows that we achieve a significant improvement over existing SFDA methods on all studied datasets.

To summarize, our contributions are threefold:

- We highlight the problem of imbalanced classes in the context of domain adaptation and its effect on the classification layer;
- We develop a backpropagation-free architecture-independent classification solution for source-free domain adaptation;
- We validate our method on the UCF-HMDB$_{full}$ and EPIC-Kitchens-55 datasets obtaining state-of-the-art results.

Chapter 2

Background

This chapter provides an overview of key topics in deep learning, focusing on those areas that are related to the topic of this thesis. Section 2.1, begins by providing a broad introduction to deep learning, briefly summarizing the historical development and core concepts. Then, section 2.2 takes a deeper dive into methods specific to image understanding. Finally, section 2.3 introduces the most common approaches to dealing with sequential data.

2.1 Deep Learning

The idea of artificial neural networks can be traced back to the 1940s and 1950s when researchers first began to explore the idea of using computational models to simulate the way the brain processes information. McCulloch and Pitts [4] laid the theoretical groundwork in 1943 by providing a mathematical description of neural networks and their incredible computational power. The first practical implementations followed shortly after, with Farley and Clark [5] designing self-organizing maps and Rosenblatt et al. [6], who introduced the perceptron, a general algorithm for binary classification. However, it was not until the late 1980s and early 1990s that the field of deep learning really began to take off. Innovations such as the backpropagation algorithm by Rumelhart et al. [7] allowed more complex networks to be trained for a variety of tasks.

As artificial neural networks were originally inspired by how the brain works, we can find many parallels to biology, and even today researchers look to biology for new ideas. The basic building block of neural networks is the artificial neuron or "unit", analogous to the neuron or nerve cell in the brain. Each neuron is connected to several other neurons and receives electrical signals from them via its dendrites. If these excitations exceed a certain threshold in strength and number, the neuron 'fires' and transmits an electrical signal itself along its axon. Individually these cells follow a simple pattern, but many of them working together can create complex

systems that are able to solve abstract problems. Our brains enable us to see, understand and interact with the world around us in this manner.

This complexity is also where the difference between deep learning and shallow learning lies. Most classical machine learning methods are classified as shallow learning, they require high-level measurements or hand-engineered features to work as intended. The advantage of using pre-computed features is that this often enables the use of smaller and faster models. Deep learning on the other hand is designed for applications where only plain data is available which does not immediately provide useful features. A typical example of this is computer vision, one of the first areas where deep learning proved its incredible potential. Attempting to define abstract and generic features based on only pixel values as input proved to be a major challenge for researchers. Deep learning overcame this challenge by automatically extracting the features it needed at the cost of requiring larger datasets. Whether shallow learning or deep learning should be applied to a given task largely depends on the type of data, its availability, and the complexity of the feature extraction process.

2.1.1 Perceptrons

This section starts by introducing the perceptron, a simple binary classification algorithm, and precursor to modern neural networks. The model consists of a single artificial neuron or unit. Each unit has a set of weights, $w_1, w_2, \ldots, w_n$, and a bias term, b. It receives input variables $x_1, x_2, \ldots, x_n$ and gives one output $\hat{y}$, which is computed as:

$$\hat{y} = f(x) = \begin{cases} 1 & \mathbf{x} \cdot \mathbf{w} + b > 0 \\ 0 & \mathbf{x} \cdot \mathbf{w} + b \leq 0 \end{cases} \tag{2.1}$$

with $\mathbf{x} \cdot \mathbf{w}$ being the dot product defined as

$$\mathbf{x} \cdot \mathbf{w} = \sum_{i=1}^{n} w_i x_i \tag{2.2}$$

We find good parameters for w and b using a supervised learning algorithm. Given pairs of inputs and outputs $(\mathbf{x}, y)$ we try to minimize the discrepancy between the true output y and the prediction $\hat{y}$ of the perceptron. For each iteration and training sample in the dataset, we update the parameters with

$$w_i = w_i + \eta * (y - \hat{y}) * x_i \tag{2.3}$$

and

$$b = b + \eta * (y - \hat{y}) \tag{2.4}$$

where η denotes the learning rate. We repeat these steps until the average error rate over the training set is below a given threshold.

Even though the perceptron is a very simple classifier that is only able to draw linear decision boundaries, it has found many applications in shallow learning and remains an important concept for artificial neural networks.

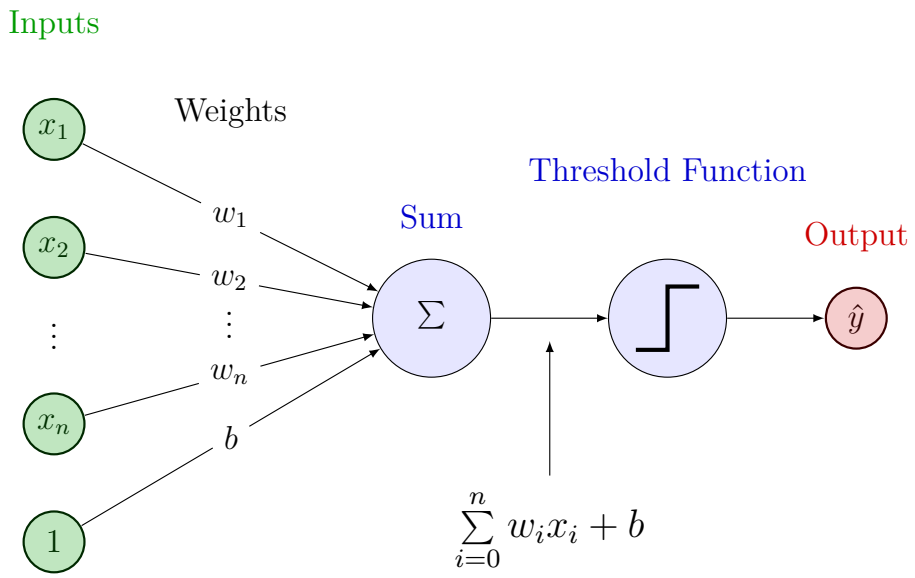


Figure 2.1: Perceptron A schematic of the different parts of the perceptron. For simplicity, the bias can be integrated into the model weights by appending another dimension with the value 1 to the inputs.

2.1.2 Multi-Layer Perceptrons

The natural extension of the perceptron is the multi-layer perceptron (MLP), or feed-forward network. MLPs consist of many individual artificial neurons working in harmony to overcome most of the limitations of using a single perceptron. Specifically, units are stacked into layers, where each layer aggregates the individual outputs and gives forwards them as a vector. Formally a layer is defined as

$$\hat{\mathbf{y}} = f_\theta(\mathbf{x}) = \phi(\mathbf{x} \cdot \mathbf{W} + \mathbf{b}) \tag{2.5}$$

where $\mathbf{x}$ is the layers input, $\mathbf{W}$ is the weight matrix, $\mathbf{b}$ is the bias, ϕ is the activation function, and $\hat{\mathbf{y}}$ is the output. The activation function is a generalization and replaces the threshold function of the perceptron. Section 2.1.3 explains its importance and presents some commonly used examples.

Furthermore, multiple layers can be appended to each other, feeding the output from one layer as input to the next. These intermediate values are called features. With each added layer the network can represent more abstract features. An MLP consists of at least three layers, the *input layer*, one or more *hidden layers*, and

finally the *ouput layer*. The *Universal Approximation Theorem* states that a theoretical network with a single hidden layer and an infinite number of units in this layer can model any continuous function. Networks with at least one hidden layer can draw any convex decision boundary and if there are two or more hidden layers the network is able to draw arbitrary decision boundaries. But in practice, the number of units in a network is limited. So instead of having very wide layers with many units, researchers exploit the network's abstraction capabilities and increase the depth of the network to model more complex functions.

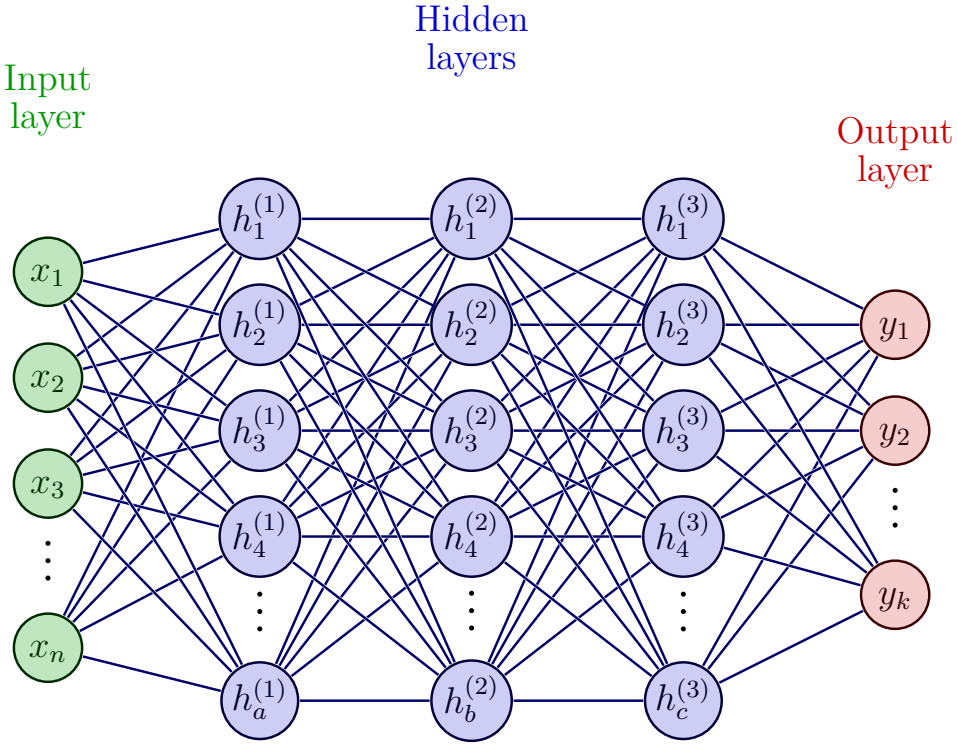


Figure 2.2: Multi-Layer Perceptron A schematic showing the structure of an MLP. This network has 3 hidden layers with a, b, and c units respectively. The connections represent weights, that are summed in each unit. Bias and activation functions are part of the hidden units and are not shown explicitly.

2.1.3 Activation Functions

An essential part of neural networks is the activation function. Without it, the network would just be a composition of linear functions, itself a linear function. Choosing an activation function that adds *non-linearity* is thus imperative. Another important feature of the activation function is its derivative, specifically how it affects the gradient during backpropagation (see section 2.1.4). Finally, the activation function should not add significant computational complexity to the network,

neither during inference nor when computing the gradient. This section will briefly present the most important activation functions, which can also be seen in figure 2.3 with their derivatives.

Binary Step This simplistic activation function was already introduced in section 2.1.1 as the activation function of perceptrons. It features a hard cutoff at $x = 0$ giving exactly two possible outputs, 0 and 1. As such, it naturally lends itself to the binary classification task. However, it is not well suited for use in general neural networks, as its derivative is always 0 regardless of the input, making it impossible to train the network using gradient-based methods.

$$f(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{2.6}$$

Identity Networks consisting of a single layer can also be equipped with the linear activation function or identity function, its continuous output domain being useful for regression tasks. For MLPs on the other hand, this choice makes little sense. Since this function adds no non-linearity, the combination of layers of an MLP collapse into a linear function, nullifying the MLPs abstraction capabilities. Because of this, the function sees only limited practical use. Furthermore, its derivative is constant and independent of the input x, making it impossible to train a network with the backpropagation algorithm. Nevertheless, it remains interesting from a theoretical perspective and can be part of more complex activation functions, such as the ReLU further down.

$$f(x) = x \tag{2.7}$$

sigmoid One of the first functions that were used to add non-linearity to neural networks is the sigmoid function. It is also known as the logistic function and is used in the logistic regression algorithm. Its most important features are that it is smooth and it fixes outputs to the interval $[-1,1]$. The downside of limiting the output range this way is the potential for *saturation*. It occurs when the gradient approaches 0 for very large negative or positive input values. This hinders the gradient flow during training and slows down the learning process. Additionally, it increases the network's sensitivity to the initialization of its parameters. Nonetheless, the sigmoid function has seen wide use and was the de-facto default in the early days of deep learning in the 1990s.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

tanh Similar to the sigmoid is the hyperbolic tangent function. The main difference is that the output of the tanh function is bound in the interval $[-1,1]$.

Otherwise, it shares many of the sigmoid's advantages and disadvantages. In practice, however, tanh tends to perform better than sigmoid, replacing it in many applications for most of the 1990s and 2000s.

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.9}$$

ReLU Recent convolutional neural networks mainly rely on the Rectifier Linear Unit (ReLU) activation function. It is simple, efficient, and does not suffer from gradient saturation, significantly accelerating training. Regardless of its popularity, however, there are some limitations. Its derivative is 0 for all negative inputs, meaning the gradient can not backpropagate further. This may lead to 'dead' units, that do not contribute to the network and never update.

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{2.10}$$

ELU Many variations of the ReLU function have been proposed to address its shortcomings. Leaky ReLUs and parametric ReLUs for example, propose replacing the constant output for negative inputs with a linear function with a very small gradient. Alternatively, the exponential linear unit (ELU) [8] defines an exponential function for negative inputs. This improves learning characteristics and evaluation performance of the network according to the authors. ELUs have a hyper-parameter α that is usually set to 1 resulting in a smooth function.

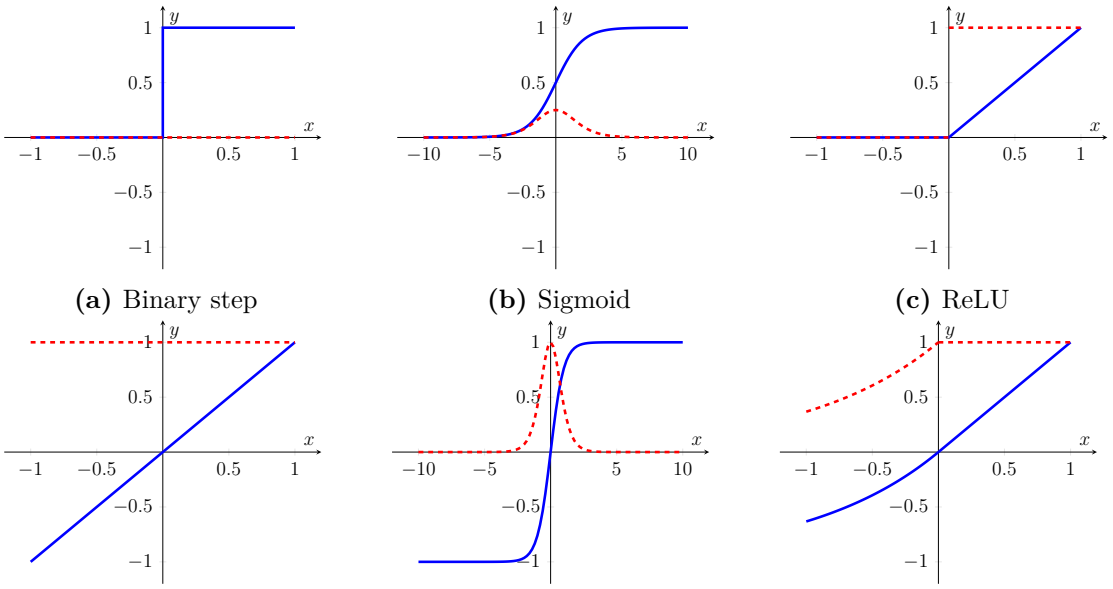
$$f(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases} \tag{2.11}$$

Other Activation Functions So far this section discussed only the most important activation functions, but many more exist, each with its advantages and disadvantages. An interesting example that features non-monotonous behavior are radial basis functions, like the Gaussian. Broomhead and Lowe [9] used this extensively in 1988 when they presented a model for interpolation and function approximation.

Some activation functions go even further, acting not on each input individually, but rather on the entire input space. The softmax activation function is an essential example in this category. It is used in virtually all networks for multi-class classification to transform the network's outputs into a discrete probability distribution over the classes.

2.1.4 Backpropagation

Backpropagation [7] played an important role in the success of deep learning, facilitating the training of multi-layer perceptrons and more complex architectures. A



(a) Binary step **(b)** Sigmoid **(c)** ReLU

(d) Linear **(e)** Hyperbolic tangent **(f)** ELU

Figure 2.3: Activation functions These plots show commonly used activation functions in **blue** and their derivatives in **dashed red**.

learning objective J is defined to find the optimal model parameters θ^* minimizing a loss function $\mathcal{L}$ over a set of training data

$$\theta^* = \arg\min_{\theta} J(\theta) \tag{2.12}$$

$$J(\theta) = \frac{1}{n}\sum_{i=0}^{n} \mathcal{L}(x_i, y_i; \theta) \tag{2.13}$$

where x_i is the ith input data and y_i its associated label. Depending on the task at hand, different loss functions can be utilized, section 2.1.5 gives an overview of the most commonly used.

Suitable objective functions are differentiable with respect to the model parameters. Similar to how the input flows through the network's layers, the gradient is propagated backward through the network following the chain rule. Finally, the parameters are nudged into the direction of the steepest loss descent

$$\theta^{t+1} = \theta^t - \eta * \nabla J(\theta^t) \tag{2.14}$$

where θ^t are the parameters at the t-th iteration, η is the learning rate and ∇J is the derivative of the objective function J with respect to the parameters θ^t.

An inherent problem with using the chain rule in very deep network architectures is that the chain of multiplied gradients gets very long in earlier layers. This becomes a problem if the gradients are consistently smaller or larger than 1. When

gradients become very large or are approaching zero, it is called the exploding or vanishing gradient problem respectively. Section 2.1.3 briefly discusses the importance of how the gradient behaves during backpropagation, as unstable gradients are one of the main issues limiting the depth of networks. An effect that is amplified in recurrent neural networks with long input sequences, as explained in section 2.3.1.

Stochastic gradient descent (SGD) addresses another common issue in large datasets. Deep learning is very data-intensive, making it infeasible to compute the loss over the entire dataset at each step. Instead, the use of mini-batches is proposed, smaller subsets of the dataset that are resampled at every iteration to approximate the global data distribution.

Slight statistical variations between the mini-batches remain, however, which can lead to instability during the learning process. Momentum [10] proposes to update the weights with an exponential moving average of the gradient, to iron out these variations. Other developments of SGD exist, to improve on different aspects. Adam [11] for example, additionally tracks the second-order derivative for each parameter, to find an optimal individual learning rate.

2.1.5 Loss Functions

Loss functions are a crucial component of the training process, as they quantify the performance of a model and define the learning objective. In this context, the mathematical properties of the loss function are important, particularly its gradient, which is propagated through the network to update its parameters. Thus, the loss function needs to be smooth and behave predictably during differentiation, having a gradient that points in the general direction of a good local minimum. Furthermore, there should be few local optima and they should give similar results to the global optimum.

Many different loss functions exist, depending on the requirements of the task. For regression problems, the most commonly used function is the mean squared error (MSE)

$$\mathcal{L}(y_i, \hat{y}_i) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.15}$$

where y_i is the true output and $\hat{y}_i$ is the predicted output for the i-th sample. It is applied directly to the model's outputs to push predictions closer to the targets, penalizing outliers more than close misses. Variations of the distance measure can also be used, like the L1 distance.

Classification models, on the other hand, are generally trained with a cross-entropy (CE) loss function

$$\mathcal{L}(y_i, \hat{y}_i) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{2.16}$$

This loss measures the distance between two discrete probability distributions: the output of the network after applying the softmax transformation, and the target label as a one-hot vector.

Other loss functions are commonly seen to address specific problem settings. Examples include self-supervised loss functions in contrastive learning, added terms for regularization as in section 2.1.6, or compositions of several loss functions to achieve multiple objectives and increase robustness.

2.1.6 Regularization

Overfitting is a common problem that can affect all machine learning algorithms, however, deep learning models are particularly sensitive. It occurs when a model fits the training data 'too well', resulting in decreased accuracy on unseen data. This loss of generalization capabilities should be avoided at all costs. Many techniques address this, ranging from choosing a different model to changes in the dataset and learning process. In this section, we present some common approaches that can be applied to neural networks.

Weight Decay Generally speaking, overfitting occurs when a model is too complex to be justified by the data. Weight decay restricts the amplitude of a model's parameters θ by adding a norm penalty to the objective function. Several choices for the norm exist, the most common being the L1 and L2 norms. In the case of the L2 norm the objective function J changes to

$$\tilde{J}(\theta) = J(\theta) + \lambda \|\theta\|_2^2 \tag{2.17}$$

where λ is a hyperparameter that controls the regularization impact.

Early Stopping Early stopping is a simple technique that does not require changing the network's parameters or objective function. Instead, it lets the training process run only as long as generalization improves, and stops as soon as this condition is violated. Generalization performance can be measured with the loss or accuracy on an independent validation dataset. Figure 2.4 shows how both the training and validation loss decrease initially until the validation loss increases again. With early stopping the final model is the one with the lowest validation loss.

data augmentation Most methods to prevent overfitting act on the model or training process. Data augmentation, on the other hand, aims to enhance the training data, so that it is too complex for the model to overfit. It increases variance in the dataset by applying random distortions to the data samples. Possible augmentations range from general methods such as adding Gaussian noise to more

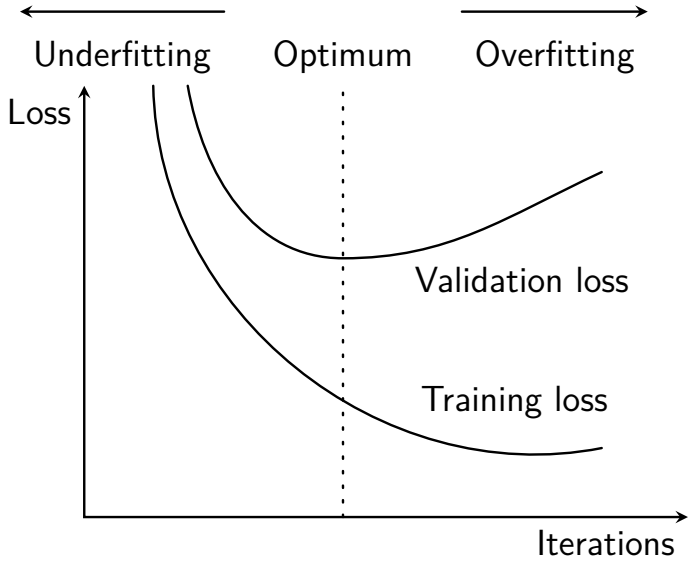


Figure 2.4: Early stopping The figure outlines how the training and validation loss may evolve during training over a number of iterations. Initially both losses decrease, until the model starts overfitting, leading to an increased validation loss while the training loss continues to decrease. The optimal model with the best generalization performance has the lowest validation loss.

task-specific augmentations like rotation or random cropping of images. Figure 2.5 shows some commonly used image augmentations.

On the other hand, augmentations may add noise to the dataset which hinders the training process and reduces the model's final performance. Thus, the choice of transformations depends on the model and task at hand. Generally speaking, however, data augmentation is considered good practice, aiding with both generalization and improving overall performance. In extreme cases, it may even be used to generate artificial datasets for tasks with very limited real-world data.

label smoothing The hypothesis behind label smoothing is that a model that is overly confident in its predictions is overfitting. To prevent this, Laplace smoothing is added to the label vector, pushing the model to make more balanced predictions. This method is simple to implement and adds very little computational complexity. Laplace smoothing is defined as

$$\tilde{y} = (1 - \alpha)y + \alpha/K \tag{2.18}$$

where y is the one-hot representation of the label, α the smoothing hyperparameter, K the number of classes, and $\tilde{y}$ the resulting label vector.

dropout Finally, dropout [13] is a regularization method that is specific to deep learning. It refers to randomly excluding units from the computation, replacing

Figure 2.5: Image augmentations This figure shows common data augmentations applied to RGB images. It is part of a large library of augmentations [12].

their outputs with 0. This encourages the network to learn redundant internal feature representations, making it more robust to unseen features. To make use of the redundancy for evaluation, dropout is only enabled during the training process. Figure 2.6 shows a schematic of an MLP with dropout applied.

In 2013 Wan et al. introduced dropconnect [14] as a generalization of dropout. Instead of dropping entire units, random connections between the units are dropped, replacing the inputs with 0. This follows the same general logic as dropout but allows for more variation between model iterations.

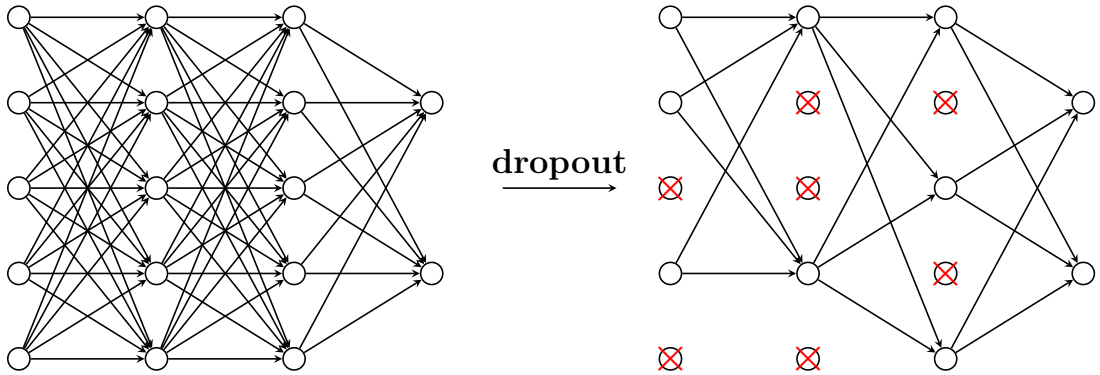


Figure 2.6: Dropout Dropout randomly excludes a percentage of units from a network for each forward pass during training, increasing redundancy and robustness in internal feature representations.

2.1.7 Normalization

This section explores the internal covariate shift and how normalization layers address this issue in neural networks. Normalization layers have an additional regularizing effect on the network, however, we discuss them separately from section 2.1.6, as the primary objective of normalization is different.

Most machine learning models assume their data to follow a normal distribution. As this is not always the case it has become standard practice to normalize data before any training or predictions are made. In neural networks, however, this issue is more complex. As the data flows through the layers each layer applies a set of transformations potentially violating the assumption of normality for the next layer. This effect is called the *internal covariate shift*. Its effects are even more apparent during training, when the networks parameters are constantly updated and thus the distribution of the ouput features changes. Normalization layers are designed to limit this issue and its implications. In the following the various types of normalization layers are presented.

Batch normalization Batch norm was introduced by Ioffe and Szegedy [15] in 2015. Their paper was the first to analyze the internal covariate shift and propose normalization layers as a solution. To this day it remains the most widely used normalization layer and has been incorporated in many state-of-the-art architectures [16].

Given a batch of inputs, each input having a spatial and a channel dimension, the mean μ and variance σ^2 are computed over all spatial dimensions and inputs, separately for each channel. After normalizing the data it can be shifted and rescaled with two learnable parameters β and γ to retain the network's representation power.

The normalization layer's output $\hat{x}_i$ is defined as

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \sigma^2_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \tag{2.19}$$

$$\hat{x}_i^{(k)} = \gamma^{(k)} \frac{x_i^{(k)} - \mu_{\mathcal{B}}^{(k)}}{\sqrt{(\sigma_{\mathcal{B}}^{(k)})^2 + \epsilon}} + \beta^{(k)} \tag{2.20}$$

where $\mathbf{x}_i^{(k)}$ is kth channel dimension of the ith sample of the input batch and m the batch size of batch $\mathcal{B}$. This operation ensures normally distributed data with location and scale explicitly defined by optimized network parameters.

Batch normalization is mainly important during training to account for the covariate shift due to updated parameters. Since the parameters are fixed for inference, batch norm is instead replaced by a linear transformation. For this, running estimates are kept during training via exponential moving averages

$$\tilde{\mu} = (1 - \lambda)\mu_{\mathcal{B}} + \lambda\tilde{\mu} \tag{2.21}$$

$$\tilde{\sigma} = (1 - \lambda)\sigma_{\mathcal{B}} + \lambda\tilde{\sigma} \tag{2.22}$$

where λ is the momentum hyperparameter. These estimates are used in place of the computed mean and variance for inference.

Other Normalization Layers Batch normalization, however, has one major shortfall, its dependence on the batch size. It works well for large batches giving meaningful statistics but becomes increasingly unstable for smaller batches. In response, many alternative normalization layers have been proposed, following the same ideas but choosing different dimensions over which the normalization statistics are computed.

Layer norm [17] takes the mean and variance over all spatial and feature dimensions of each individual sample. Instance norm [18] on the other hand calculates them over all spatial features of each feature, behaving as batch norm would with a batch size of 1. Group norm [19] tries to strike a balance between the two, extending the computation over groups of channels. Figure 2.7 illustrates this concept more intuitively.

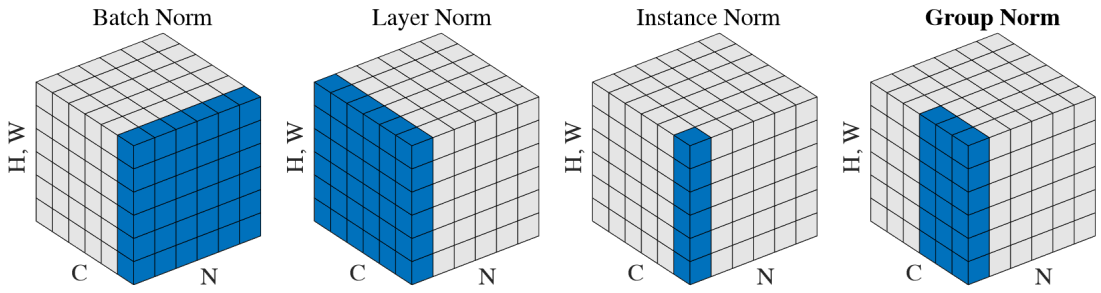


Figure 2.7: Normalization layers This figure provides an overview of different types of normalization layers. H and W are the height and width representing the spatial dimension, C is the channel dimension and N the batch axis. The blue boxes indicate the sets of values which share the same mean and variance. Figure taken from [19]

2.2 Convolutional Neural Networks

Image understanding and classification are complex disciplines in computer vision that aim to develop algorithms that are able to extract abstract information from images. Interesting challenges and a wide range of potential applications keep the research community engaged. Thus it is no wonder, that this area set the stage for the rise of deep learning. Early attempts starting in the 1960s sought to tackle the issue by using traditional machine learning models on top of hand-crafted features [20], but they saw only limited success.

While simple feed-forward networks are theoretically able to implement arbitrary functions, they have several fundamental limitations that prevent them from being successful in this field. Notably, objects in images are position invariant, meaning the object is the same, no matter where in the image it is located. Additionally, the meaning of a pixel depends on those around it, a relationship that MLPs do not capture explicitly.

A new class of neural networks was required, the convolutional neural network (CNN). Inspired by Hubel and Wiesel's [21] discovery, that the neurons in the visual cortex in cats had a limited receptive field, Fukushima [22] presented the neocognitron featuring convolutional layers. Further innovations like the backpropagation algorithm [7] and increasingly complex architectures kept advancing the field.

2.2.1 Convolutions

Convolutions are mathematical operators that express an interaction between two functions f and g defined as

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt \tag{2.23}$$

If f and g are discrete functions, this equation simplifies to

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m] \tag{2.24}$$

In computer vision, convolutional operators are typically defined as 2D matrices that slide over an image. They are can be used for example to blur images or detect edges. In CNNs they are treated as learnable parameters to extract features from the input data. Figure 2.8 shows what this 2D convolutional operation looks like.

Analog to the linear layers of the MLP, convolutional layers are part of the architecture of a neural network. This addresses two of the main drawbacks of MLPs with images, the convolutional operator is spatially invariant and addresses the locality of features. To increase the representational power of convolutional layers, several filters can be used at each stage, returning an additional dimension in the output feature map, the channel dimension.

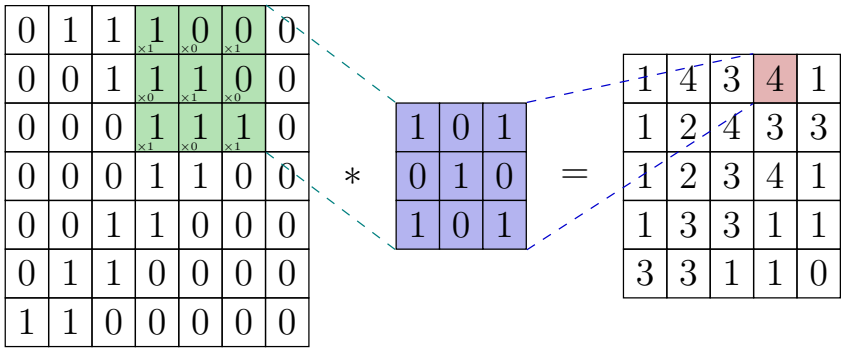
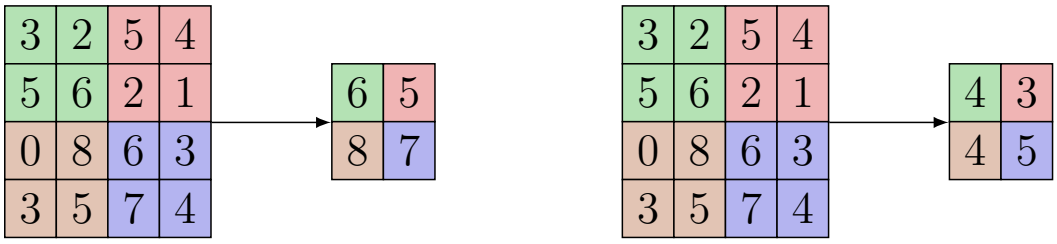


Figure 2.8: Convolution The discrete 2D convolution involves sliding the filter (middle) over the input (left) and computing the weighted sum as the output at that position (right).

2.2.2 Pooling

Another important part of convolutional neural networks are pooling layers. While the main purpose of convolutional layers is to extract information, pooling layers are designed to reduce the dimensionality of latent features. They summarize features in patches of the feature map and thus downsample the input space and increase local translation invariance.

Typical summarizing functions in pooling layers are the max function and averaging. They are efficient and maintain important information of the feature map. Other pooling layers exist, like min pooling or L2 pooling, but they are used much less frequently. Figure 2.9 shows a simplified comparison of max pooling and average pooling.



(a) Max pooling **(b)** Average pooling

Figure 2.9: Pooling A simple example of two pooling layers with 2×2 kernels and 2×2 stride applied to the same input.

2.2.3 Architectures

One of the most important architectures in the development of CNNs is LeNet-5 by Yann LeCun's research team [23]. Built for optical character recognition of US postal codes still finds application today as a minimalistic CNN. It features three convolutional layers and two pooling layers as part of the feature extractor followed by two fully connected layers for classification.

The next step in the architectural evolution of CNNs was taken by Krizhevsky [24] with AlexNet. It features a much deeper architecture which empowered it to win the prestigious ImageNet [25] challenge in 2012. GPU accelerated training made it feasible to train such a large architecture, a method that would persist with later research.

In 2014, Szegdy et al. [26] presented Inception net and won the ImageNet challenge. They proposed a significantly more complex architecture than previous networks. An important assumption of this architecture is that features are present at different scales in the input. The researchers address this by applying several differently-sized convolutions at the same network depth. These parallel convolutions were bundled into Inception modules that could be chained together. The very deep architecture required several auxiliary loss functions throughout the network to help guide the gradient during training. Aggressive downsampling at the first layer helps to limit the computational cost of the network.

Inception net proved very powerful and several improvements have been added to later iterations of the base model. Inception v2 [27] saw the factorization of convolutions into a number of smaller convolutions, reducing the number of parameters and computations and increasing the representational power. Inception v3 [27] added several regularization techniques, most notably batch norm layers.

Alongside Inception, ResNet [16] is one of the most widespread architectures today. Its largest version won the 2015 ImageNet challenge. The defining feature of ResNet is the skip connection, which allows layers to be skipped, greatly helping the gradient flow thus allowing much deeper architectures. ResNet exists in different

sizes enabling adjustment to the computational constraints of the application.

2.3 Sequential Data

Many real-world applications deal with sequential data. This can be in the form of chronologically ordered measurements, the frames of a video, notes from a song, or the words in a text. Specialized models are required to properly caption the context of logically ordered inputs.

Early attempts at this followed similar developments in computer vision by using 1D convolutions, as the same locality of features principle discussed in section 2.2 applies to sequential data. To better handle sequences of arbitrary lengths, researchers proposed encoding the contextual information with a hidden state vector using recurrent neural networks (RNNs). Long short-term memory (LSTM) networks are an improvement upon vanilla RNNs that are designed to be more stable on very long sequences. Current state-of-the-art models rely on transformer architectures with self-attention modules, that allow the model to focus on specific parts of the input sequence, for each output, regardless of sequence length.

2.3.1 Recurrent Neural Networks

The fundamental idea behind recurrent neural networks is that the model encodes the context of an input sample x_t at time step t in a hidden state vector h_t. Following hidden states h_{t+1} and the model's output $\hat{y}_t$ can then be computed from the input and the hidden state.

$$\hat{y}_t = f_\theta(x_t, h_t) \tag{2.25}$$

$$h_{t+1} = g_\theta(x_t, h_t) \tag{2.26}$$

Computing the next hidden state from all previous ones using the same operation gives RNNs their name. It results in a highly versatile model, that can handle arbitrary length input and output sequences and be naturally adapted to different settings of sequential data processing, as shown in figure 2.10.

Long-Short-Term Memory An important limitation of RNNs is the vanishing or exploding gradient problem. When backpropagating the gradient over long sequences, the gradients are multiplied with the weights at each step, leading to very small or large values if the gradients are consistently below or above 1. LSTMs propose to limit the number of interactions with the hidden state by adding a series of gates while keeping to the basic principle of storing context in the hidden state.

Specifically, LSTM cells consist of three gates, an input gate, an output gate, and a forget gate. Each gate consists of a single fully connected layer followed by a

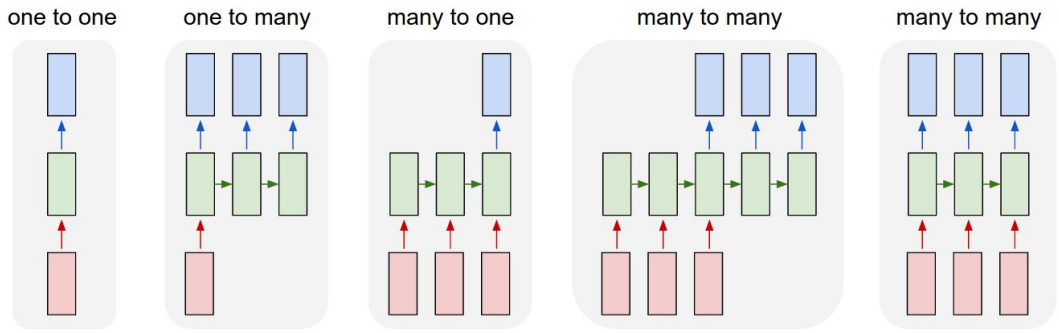


Figure 2.10: Recurrent Neural Network Architectures Different architectural setups of RNNs, from one-to-one vanilla models without recurrency to different forms of sequence inputs and outputs with recurrency. Red rectangles represent input vectors, hidden states are green and output vectors are shown as blue rectangles. Image from [28].

sigmoid activation function, to convert the output to a weight factor. At time step t, the cell takes the encoded model input x_t along with the previous cell state c_{t-1}, and hidden state h_{t-1} to compute the next cell state c_t and hidden state h_t. The hidden state h_t doubles as the LSTM cell's output. h_{t-1} and x_t play an important role in controlling the gates to regulate the flow of all information passing through the cell.

The output of gate g_t at time t can be computed as

$$g_t = \sigma(W_g x_t + U_g h_{t-1} + b_g) \tag{2.27}$$

where W_g and U_g are the weights and b_g the bias specifically associated with gate g, and σ is the sigmoid activation function (see section 2.1.3). Furthermore, the new intermediate cell state is calculated from the layers input and hidden state as

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2.28}$$

Finally, the new cell state is computed by passing the old and new cell states through their respective gates, applying element-wise multiplication, and summing the results

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.29}$$

where f_t and i_t are the outputs of the forget gate and input gate respectively, and $\odot$ is the Hadamard product. The cell's output and new hidden state is then the Hadamard product of the new cell state with the output gate o_t

$$h_t = o_t \odot \tanh(c_t) \tag{2.30}$$

Figure 2.11 gives a visual explanation of the LSTM cell.

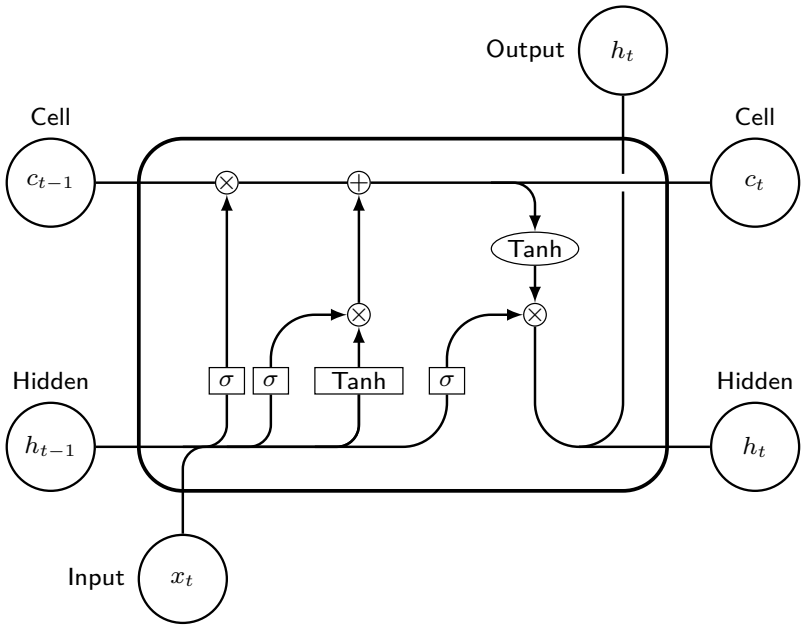


Figure 2.11: LSTM cell This figure visualizes the structure of an LSTM cell. LSTMs improve RNN's abilities to learn from long input sequences with LSTM cells by improving the flow of information through the use of input, output, and forget gates, indicated as σ activation functions. The gates are controlled by the hidden state h_{t-1} and the input x_t, to manipulate the cell state c_{t-1} and finally return the new hidden state h_t as output.

2.3.2 Transformers

LSTMs mark an evolutionary step in the development of models for text and speech understanding. However, they face the same underlying issues as standard RNNs: instability during training and the representational bottleneck of the hidden state. To address this, researchers proposed the concept of attention. Attention allows the model to deliberately focus on specific parts of the input sequence when predicting each output.

Attention First, the inputs $x_1, x_2, \ldots, x_n$ are encoded in the latent space as

$$h_i = f_E(x_i, h_{i-1}) \tag{2.31}$$

Next, an alignment score e_{ij} is computed between the input h_i the hidden state s_j of the decoder

$$e_{ij} = f_{att}(h_i, s_j) \tag{2.32}$$

The function f_{att} used to compute this alignment score can be as simple as the dot product or as complex as an MLP with its own trainable parameters. Using the softmax function the alignment scores $e_{1j}, e_{2j}, \ldots, e_{nj}$ are turned into attention

weights $a_{1j}, a_{2j}, \ldots, a_{nj}$. These weights are multiplied with the encoded inputs

$$c_j = \sum_{i=1}^{n} a_{ij} h_i \tag{2.33}$$

to get a representation that is meaningful for the current state s_j of the output sequence. Finally, the decoder computes the new state s_{j+1} and output from c_j.

Transformers Vaswani et al. [29] take the concept of attention a step further and introduce *self-attention*. They propose to transform the inputs h_i into key k_i, value v_i, and query q_i vectors

$$k_i = f_K(h_i) \quad v_i = f_V(h_i) \quad q_i = f_Q(h_i) \tag{2.34}$$

where f_K, f_V, and f_Q are the multi-layer perceptrons. Then, alignment scores e_{ij} are calculated as the dot product of the query q_j and key vectors k_i and transformed again into attention weights a_{ij} using the softmax function. Similar to the standard attention model, the weights a_{ij} are multiplied with the value vectors v_i to compute the outputs c_j of the layer. However, thanks to the use of key-value pairs, this can be done independently for each sample of the sequence, without keeping track of a state vector s_j.

Multi-headed attention employs multiple self-attention mechanisms at the same abstraction layer to focus on separate parts of the sequence, facilitating the modeling of different contexts. Transformers stack many multi-headed attention layers in both the encoder and decoder, to model high-level abstract relationships within the input sequence. Currently, transformers are the state of the art for sequence processing, replacing RNNs and LSTMs in most applications.

Chapter 3

Related Works

This chapter provides an overview of the relevant literature for our work. Section 3.1 explores action recognition, explaining the task, the importance of temporal and multimodal information, datasets, and important architectures. Section 3.2 takes a look at the cross-domain scenario and its sub-settings, presenting important conventional approaches and methods specific to action recognition.

3.1 Action recognition

Action recognition is a classification problem in computer vision, that aims to identify what kind of activity is performed in a short video clip. In most cases, this amounts to assigning a verb label to that activity, like *walking* or *climbing*. Some datasets [30] additionally record interactions with objects, increasing the difficulty by requiring the correct activity verb as well as the correct object noun to be identified. This section presents the action recognition task and explores some of the challenges specific to this setting.

3.1.1 Egocentric Action Recognition

Egocentric Action Recognition is a special case of action recognition where the videos are filmed from a first-person perspective. This adds many challenges and unique aspects to the classification task. First of all, the input video is recorded with a limited field of view and never shows the full body posture or movement of the person performing the action. Furthermore, the camera itself experiences a lot of movement leading to drastically changing view angles, as well as frequent object occlusion.

On the other hand, additional information can be inferred in this special setting. Changes in the camera perspective keep the relevant action always in the center of the frame and help determine the relative locations of key objects in 3D space.

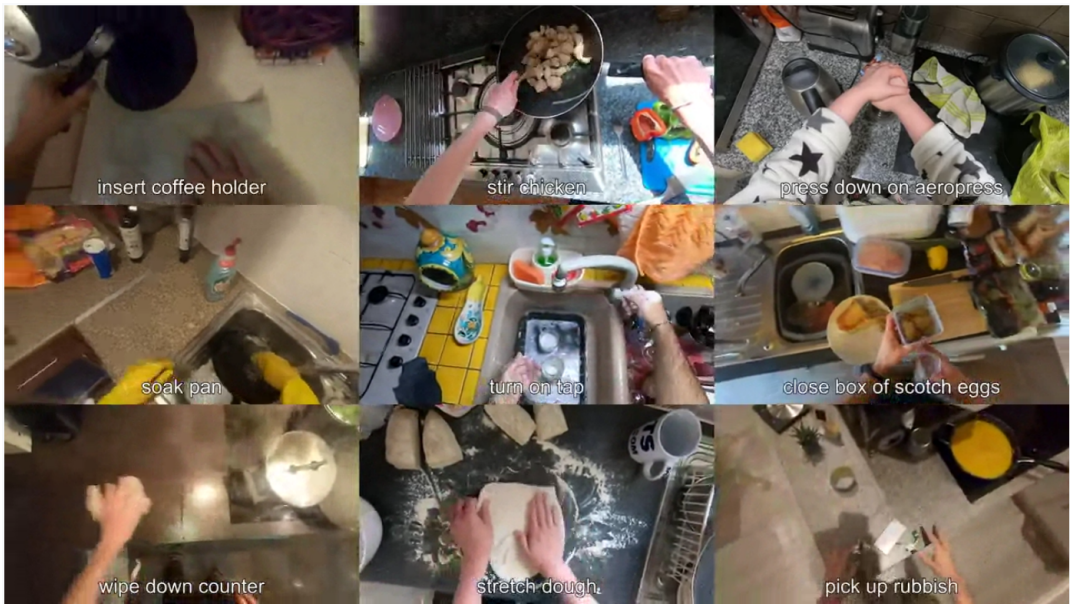


Figure 3.1: EPIC Kitchens Some example actions with their verb and noun labels from the EPIC Kitchens [30] egocentric action recognition dataset.

Tracking the gaze direction, for example, helps models focus on relevant parts of the input [31, 32]. Similarly, some researchers have suggested the use of hand positions to identify hand-object interactions and assist in action classification [33].

Finally, the distribution and structure of action classes is different in egocentric datasets. Traditional action recognition videos include classes like *walking* and *running*, which are difficult to distinguish from a first-person point of view. The biggest public egocentric dataset [34], on the other hand, contains videos of people performing typical kitchen activities, like *mix batter* and *stir chicken*, which are more dependent on egocentric-specific information. Figure 3.1 shows some sample actions and screenshots from egocentric videos of the EPIC Kitchens dataset.

3.1.2 Temporal Information

The key difference between image and video classification lies in the additional temporal information of videos. Different formats exist to encode temporal information, such as event cameras, RGB differences, and flow images. In most cases, however, videos are treated as sequences of RGB frames, extracted at a predefined frame rate.

The relevance of temporal information varies between action recognition datasets [35] and some researchers achieve impressive results with regular 2D CNNs from image recognition architectures [36, 37]. Nonetheless, temporality is a defining feature of this field and often carries essential information. Especially in egocentric

action recognition, camera angles, and viewpoints often change, indicating new positions where relevant actions take place [31, 32]. Furthermore, this positional variance can give the environment a three-dimensional meaning and move objects in and out of occlusion. Finally, some actions require temporal context to be distinguishable. In the EPIC Kitchens dataset [30] for example both the action of opening and closing a cupboard exist. The difference between these two actions is entirely dependent on the temporal context, highlighting the need for powerful architectures able to capture this information.

In the following, we briefly present the most important models that have been proposed to extract both spatial and temporal features from RGB frame sequences.

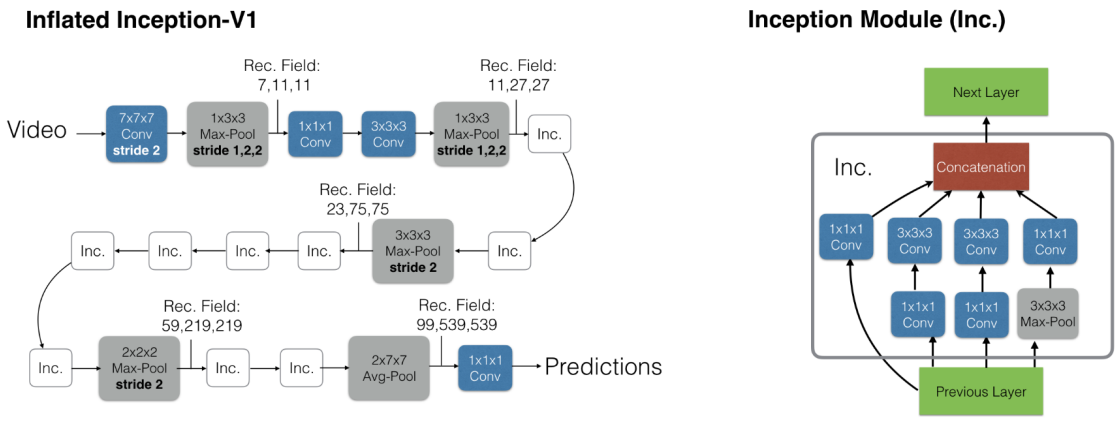


Figure 3.2: Inflated Inception-V1 Architecture A detailed overview of the I3D inflated Inception-V1 architecture. The left figure shows a high-level overview, while the right side shows an Inception module in detail. Figure from [38].

2D Convolutions Earlier approaches relied entirely on applying 2D CNNs on single frames. The extracted features could then be averaged or concatenated in a process called fusion. Fusion could either occur in lower layers or higher up in the network architecture, resulting in the *early fusion* and *late fusion* approaches respectively. Karpathy et al. [39] show in their study that these methods achieve competitive results and benefit directly from new advances made in image classification.

Later advancements improved the fusion process by applying more complex architectures from sequence processing or specialized fusion modules. LSTMs have been used with great success [40, 41] even in complex settings such as unsupervised learning [42] and domain adaptation [43]. Analog to the developments in sequence processing, attention-based methods [44, 45] and transformers [46, 47, 48] have grown in popularity.

Notable examples of special fusion modules include temporal relational networks [49], which feed different-length sequences of frame-level features into separate

MLPs to model temporal relations over distinct time intervals. Temporal shift modules [50] enhance temporal pooling by replacing a small percentage of features in convolutional layers with features from adjacent frames in the past or future. Temporal aggregation modules [51] enhance this step by adding a depthwise 1x1 convolution before shifting along the temporal axis. Finally, Sahoo et al. [52] propose graph convolutions to fuse temporal features.

3D Convolutions A different approach to temporal modeling has been proposed with 3D CNNs. They naturally extend the 2D convolutional filters of image-based CNNs along the temporal axis, resulting in 3D spatiotemporal filters. These filters operate on $h \times w \times f \times c$ inputs called clips, each clip consisting of f consecutive $h \times w \times c$ frames.

3D CNNs have been proposed several times [53, 54, 55], however, their drastically increased complexity requires much more computational power and presents difficulties during training. Inflated 3D (I3D) models [38] were a major breakthrough in this regard, taking a pretrained BN-Inception net [27] and inflating the 2D filters to three dimensions while keeping the original parameters. This involves copying them along the temporal axis, but down-scaling them so that their output does not change when applied to a static video consisting of a single repeated frame.

While most 2D and 3D CNNs are adaptations of models directly copied from image recognition, some video-specific architectures have been proposed in recent years. Similar to the origins of 2D CNNs, Slowfast [56] also takes inspiration from the biological visual cortex. It features a slow pathway with a powerful CNN that works on frames sampled with a low frame rate, and a fast pathway, that works with a high frame rate and a fraction of the channels of the slow pathway, to save computations. The intuition is that the slow pathway extracts complex spatial features, while the fast pathway captures temporal information, resulting in an efficient and capable architecture.

3.1.3 Multi-Modal Data

Another characteristic feature of videos is their inherent multi-modal nature. Modalities are different sources or encodings of the input that may contain complementary information. Examples include measurements collected from auxiliary sensors, such as the audio track, camera position and movement, and event-based videos, as well as information that can be extracted from the source video with advanced algorithms, for example, optical flow or RGB differences. Figure 3.3 shows some example modalities.

Training models to extract features from multiple modalities can simultaneously boost performance [38] and robustness [57]. Different modalities often contain additional information [58] that is not available in other modalities. Especially in the cross-domain scenario, many methods exploit a distinct and often uncorrelated

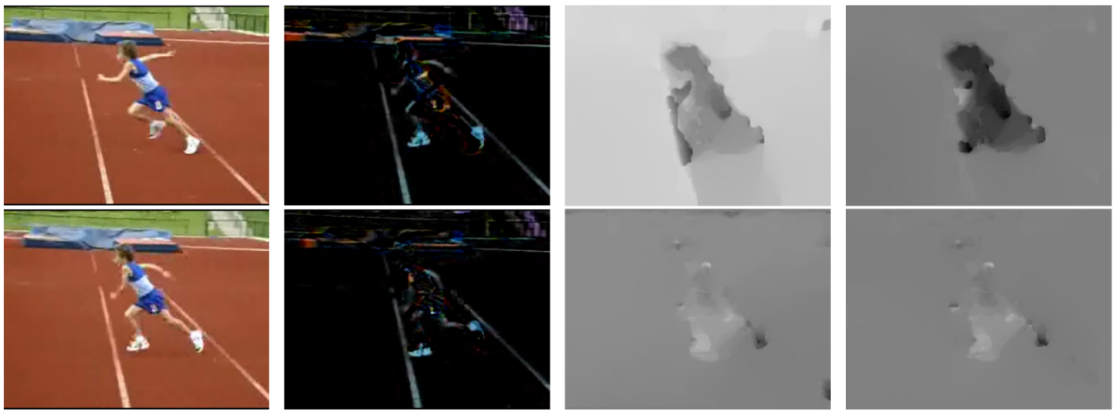


Figure 3.3: Modalities Examples of different modalities for two different images. From left to right: RGB images, RGB difference, optical flow, warped optical flow. Images from [36].

domain shift in each of the modalities [59, 60].

Similar to temporal information, different techniques exist to integrate multi-modal data. Most models revert to simple early [36] or late fusion [59, 38] of features, while others integrate multi-modal learning with the extraction of temporal features [61] or propose dedicated fusion architectures [62].

This section presents the most relevant modalities for action recognition

Optical Flow Optical flow is the most commonly used auxiliary modality for action recognition as it records the apparent motion of objects between frames with a 2D motion vector at each pixel. While this information is present in each pair of RGB frames as well, it is highly abstract and difficult to implicitly extract with a neural network. Figure 3.4 shows a visualization of what the optical flow modality looks like. Extracting optical flow requires complex and slow algorithms [63], however, efforts have been made to leverage deep learning in this context [64].

Audio Another widely available and commonly used modality is audio [66, 67]. This modality is especially useful in the context of object interactions [30] as objects made from different materials often make distinct sounds. Cooking food in a metal pot for example will sound different than chopping paprika on a wooden cutting board.

It is common practice to transform the audio track into a spectrogram that is then evaluated by a pretrained 2D CNN. The default architecture for this is BN-Inception [27] pretrained on ImageNet [25], which has proven to give reliable results [61]. Figure 3.5 shows a comparison of spectograms of different sounds at different resolutions.

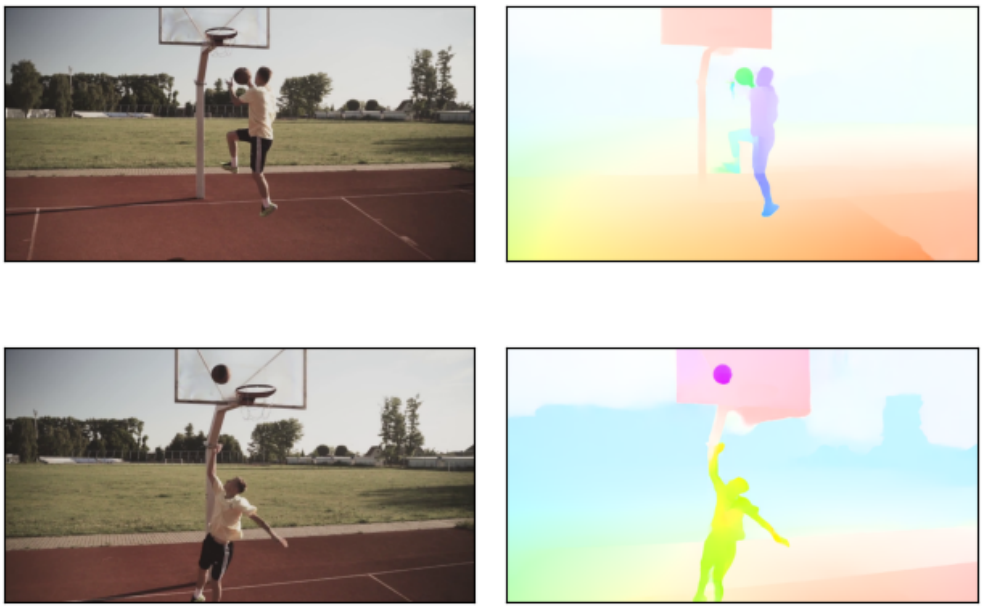


Figure 3.4: Optical flow The left column shows two RGB frames of a video, while the right side shows the corresponding extracted flow images. Color indicates the direction of the flow vectors and color intensity their magnitude. Images from [65].

Other modalities Other modalities have been proposed to augment the information passed to the model. These range from straight-forward RGB differences [36] to the implementation of novel sensors. Event cameras [68] are neuromorphic sensors that asynchronously record brightness changes of individual pixels to capture information at extremely high temporal resolution, simultaneously reducing redundancy by ignoring still parts of the image. Plizarri et al. [69] show that event-based videos classified by common video-processing architectures are competitive with state-of-the-art RGB and Flow multi-modal models.

3.1.4 Datasets

A multitude of datasets exists for action recognition. This section introduces the most relevant ones for this work.

Kinetics Kinetics [72] is one of the largest action recognition datasets, featuring 700 different action classes with over 700 videos each. The videos were scraped from YouTube and cover a diverse range of human actions, including a variety of human-object and human-human interactions. Due to its comprehensiveness, it is often used to pretrain action recognition models, similar to ImageNet in 2D computer vision. Carreira et al. [38] show that pretraining on Kinetics can significantly boost

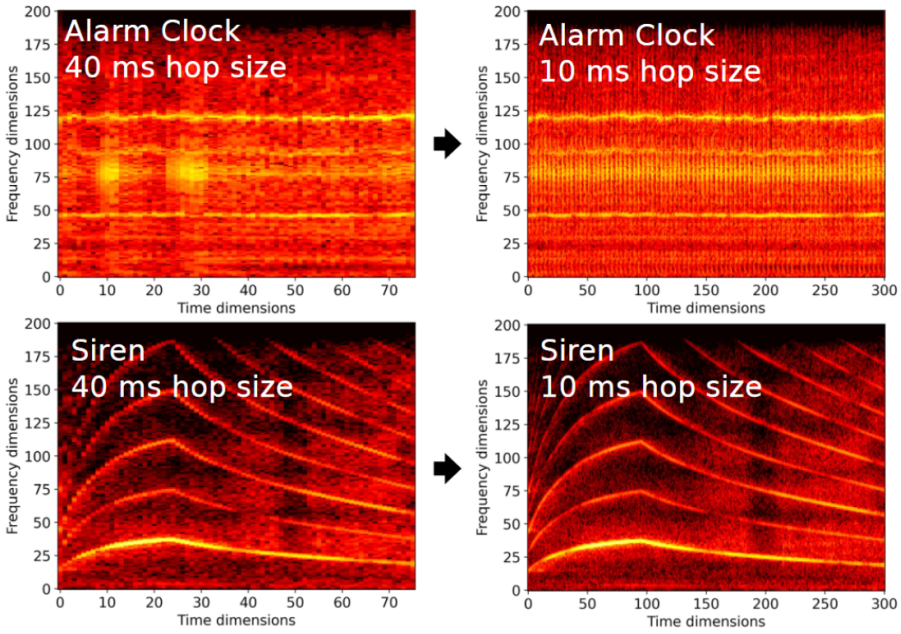


Figure 3.5: Spectrogram This figure shows the spectrogram for two distinct sounds (Alarm and Siren). The left column has a lower resolution due to the larger hop size, while the right column shows higher-resolution spectrograms. Figure from [70]

a model's performance, also on other action recognition datasets.

UCF-HMDB This dataset is a standard benchmark for domain adaptation models for action recognition. It consists of two independent datasets: HMDB51 [73] and UCF101 [74]. HMDB51 was the largest action recognition dataset at the time of publishing, comprising 51 classes and over 7000 videos from YouTube and movie scenes. Similarly, UCF101 is the latest iteration in a series of related datasets with videos downloaded from YouTube. It is slightly newer than HMDB51 and consists of 101 distinct classes. Figure 3.6 shows sample images and classes from both HMDB51 and UCF101 datasets. Due to domain adaptation being a relatively new area of research with specific requirements for datasets to show a clear domain shift, not many datasets dedicated to this task exist yet. In practice, researchers often resort to combining different datasets that share common classes. Examples include UCF-HMDB$_{small}$ [75], Kinetics-Gameplay [76], and UCF-HMDB$_{full}$ [76]. However, some newer datasets include dedicated setups for domain adaptation to provide a unified framework [30] for benchmarks.

Epic Kitchens EPIC Kitchens 55 revolutionized the field as the first large-scale dataset for egocentric action recognition dataset. Its initial release [30] featured over 55 hours of videos recorded with a variety of head-mounted cameras in 32

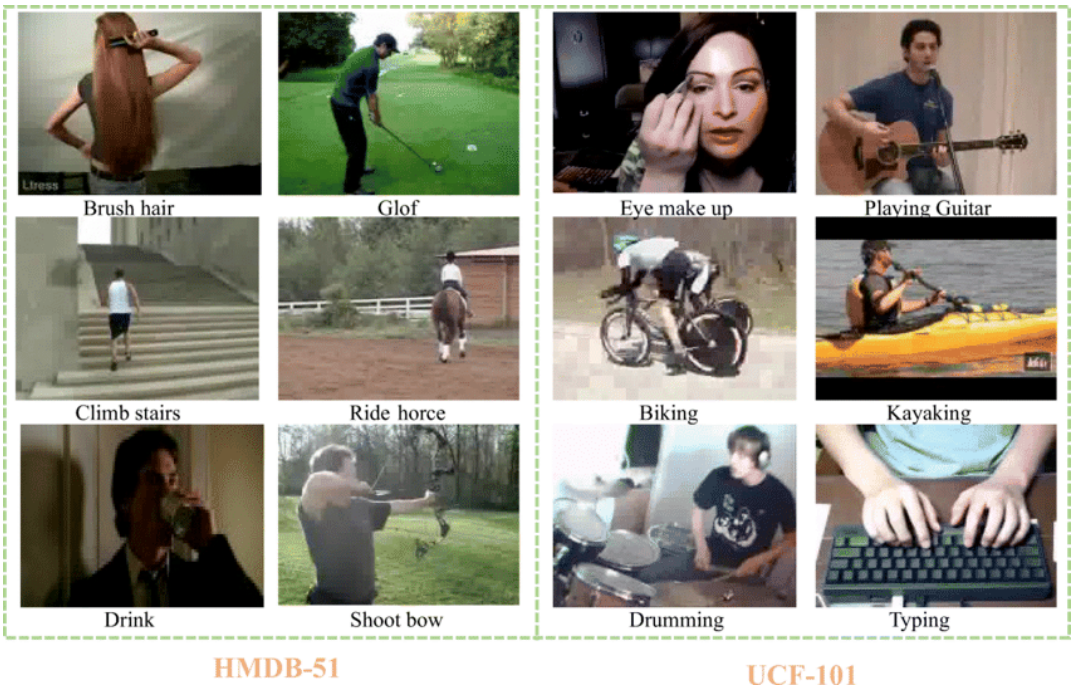
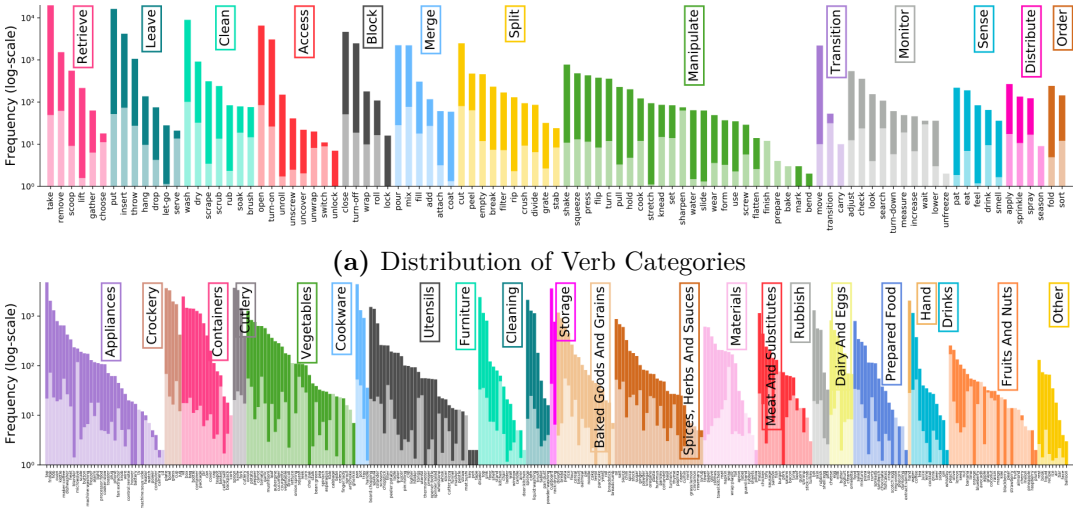


Figure 3.6: HMDB51 & UCF101 Some examples of actions in the HMDB51 (left) and UCF101 (right) datasets. Figure from [71].

different kitchens all over the world. Many different annotation types exist for a variety of tasks including action recognition, action detection, action anticipation, domain adaptation, multi-instance retrieval, and object detection. A reworked version of the dataset [34], EPIC Kitchens 100 contains even more videos from recurring kitchens and some new kitchens. Figure 3.1 shows a few examples of actions with their verb and noun labels.



(a) Distribution of Verb Categories

(b) Distribution of Noun Categories

Figure 3.7: EPIC-Kitchens-100 Distribution of Classes Distribution of verb and noun categories of the EPIC-Kitchens-100 dataset [34].

3.2 Cross-Domain Learning

Machine Learning models are generally trained under the assumption of independently and identically distributed data in both the training and test dataset. State-of-the-art deep learning models achieve impressive results across a variety of tasks and settings as long as this assumption holds. When models are evaluated on a different domain or dataset, however, the assumption breaks, resulting in a significant decrease in performance [77]. The difference between the source domain, on which the model was trained, and the target domain, on which the model is evaluated, is called the domain shift. Addressing this loss of performance across domains is an intense area of research. This section summarizes important publications for different settings: domain generalization (DG) in section 3.2.1, unsupervised domain adaptation (UDA) in section 3.2.2, source-free domain adaptation (SFDA) in section 3.2.3, and test-time adaptation (TTA) in section 3.2.4.

3.2.1 Domain Generalization

Domain generalization comprises methods that adapt a model from one or more source domains to an unknown target domain and can be roughly divided into three overall categories [78]. Data manipulation aims to augment source data or generate new data samples to increase a model's robustness and includes common domain shifts in the training process. Representation learning, on the other hand, attempts to improve internal feature representations by extracting domain-invariant features and feature disentanglement. Finally, a number of publications propose

new learning strategies that lead to more general models.

RNA-Net Relatively little research has gone into domain generalization specifically for action recognition. Planamente et al. [57] propose a relative norm alignment loss in the form of RNA-Net. Their approach seeks to augment multi-modal training by encouraging the feature representations of each modality to be similar in magnitude. This avoids the common problem of one modality having a significantly stronger effect on the final prediction.

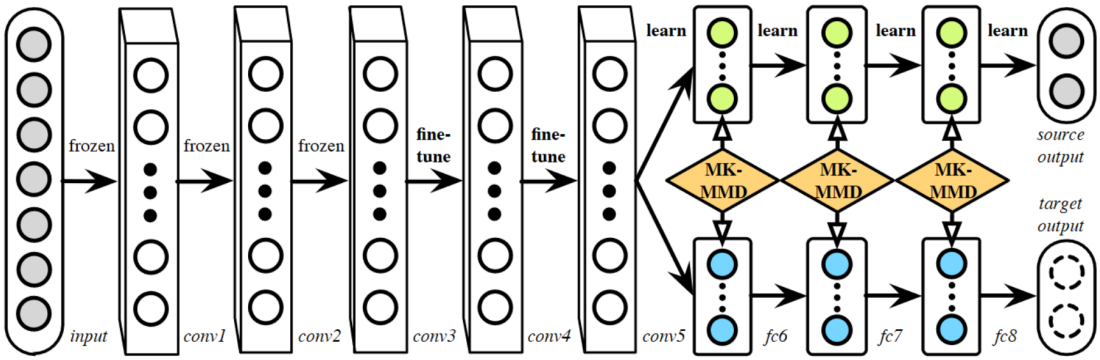


Figure 3.8: Deep Adaptation Network [79] For adaptation, the first three convolutional layers are frozen, the last two are fine-tuned, and the task-specific fully-connected layers are adapted with multiple kernel maximum mean discrepancy.

3.2.2 Unsupervised Domain Adaptation

Unsupervised domain adaptation (UDA) is the most widely studied and general setting of domain adaptation. During adaptation models simultaneously have access to labeled source data and unlabeled target data. Many different methods have been proposed over the years, ranging from discrepancy-based approaches and domain adversarial networks to pseudo-labeling and reconstruction-based methods.

Early research attempted to find an embedding space that minimizes the discrepancy between the source and the target domain. A commonly used pattern is to map samples into a kernel Hilbert space using maximum mean discrepancy (MMD) [80, 81]. Other publications propose to model the datasets from different domains as lying on a continuous manifold [82, 83], connected by geodesic curves. Intermediate datasets can then be sampled along the curve to observe a gradual change in the data distribution.

Maximum mean discrepancy [85] is another method that aims to minimize the statistical variance of the model between the source and the target domain. Long et al. [79] propose multiple kernel maximum mean discrepancy (MK-MMD) to adapt the classifier module of the network while freezing the earlier layers of the feature

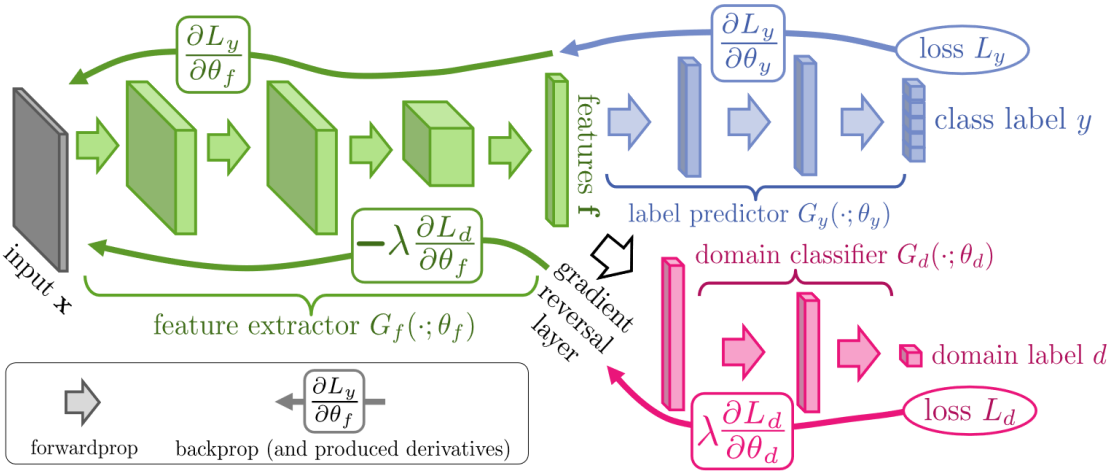


Figure 3.9: Domain Adversarial Neural Network DANNs [84] include an adversarial network (pink) that is trained to distinguish features from the source and the target domain, pushing the feature extractor (green) to learn domain-invariant features.

extractor and fine-tuning later layers in a supervised manner. Figure 3.8 illustrates the architecture of deep adaptation networks using MK-MMD to adapt domains. They further develop this work with joint MMD [86] to improve gradient flow and capture more complex domain shifts.

A different approach is taken by adversarial domain adaptation. In this case, an adversarial network is trained to distinguish between samples from the source domain and the target domain. Through the gradient reversal layer, the discriminator loss pushes the network to encode source and target features in a domain-invariant way [84]. This adversary is usually applied to high-level features to adapt the feature extractor, but the method can also be applied directly to inputs [87]. Figure 3.9 shows a general CNN architecture during adaptation with a domain adversarial network. Asymmetric adversarial domain adaptation [88] takes a similar approach, replacing the domain discriminator with an autoencoder that is only trained to reconstruct samples from the source domain, thereby pushing the feature extractor to model the target domain in a similar manner.

Below we present important UDA for action recognition publications.

MM-SADA Multi-modal self-supervised domain adaptation (MM-SADA) [59] is the first technique, that exploits the multi-modal nature of videos for domain adaptation. Its architecture consists of a two-stream I3D network with separate feature extractors and classifiers for each modality with the final classification score being the average of the predictions on each modality. Adaptation is done over source and target data simultaneously and consists of two components. Within-modal alignment performs adversarial domain adaptation separately on both feature extractors,

while multi-modal alignment is achieved with a binary domain correspondence classifier which learns to decide whether the features of the two domains belong to the same action class or different action classes. It is trained in a self-supervised manner by sampling positive samples as clips from the same video, possibly at different times, and negative samples as clips from different actions. The correspondence module should be as shallow as possible so that the feature extractors are trained to encode the necessary information to distinguish actions. The final loss for adaptation is a weighted sum of the supervised action classifier loss, the adversarial within-modal alignment losses, and the self-supervised correspondence loss.

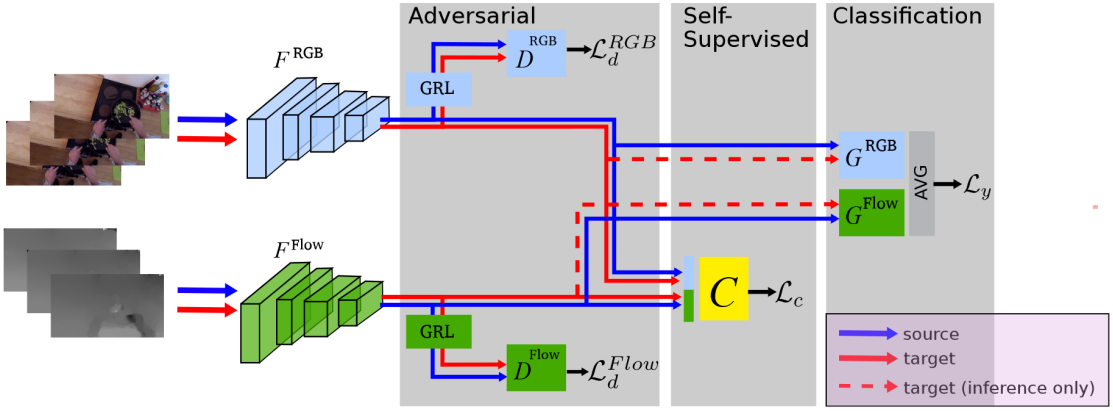


Figure 3.10: Multi-Modal Self-Supervised Domain Adaptation Architecture of MM-SADA [59]. F^{RGB} and F^{Flow} are the feature extractors for each modality, and G^{RGB} and G^{Flow} the classifiers. Final predictions are the average of all modality predictions. Within-modal adaptation is achieved by the adversarial domain discriminators D^{RGB} and D^{Flow}, while the correspondence module C is responsible for multi-modal alignment. The adaptation loss is a weighted sum of the adversarial losses $\mathcal{L}_d^{RGB}$ and $\mathcal{L}_d^{Flow}$, the correspondence loss $\mathcal{L}_c$, and the classification loss $\mathcal{L}_d^{RGB}$.

CoMix Sahoo et al. [52] propose contrast and mix (CoMix) to exploit the temporal dynamics and background invariance of actions with a contrastive learning approach. Their model has a unique architecture consisting of a graph convolutional network with three layers on top of the I3D feature extractor. During adaptation, the model is trained with a weighted average of three different losses.

A temporal contrastive loss $\mathcal{L}_{tcl}$ enforces features to be invariant to the video recording speed by sampling positive samples as clips from the same video at different frame rates and negative samples as clips from different videos.

$$\mathcal{L}_{tcl}(V_f^i, V_s^i) = -\log \frac{h(z_f^i, z_s^i)}{h(z_f^i, z_s^i) + \sum\limits_{\substack{j=i, j\neq i \\ v\in\{s,f\}}}^{B} h(z_f^i, z_v^j)} \tag{3.1}$$

where V_f^i is fast version of the i-th video, V_s^i is its slow version, z_f^i and z_s^i are the respective feature representations, and $h(u, v) = \frac{u^T v}{\tau \|u\|_2 \|v\|_2}$ is the exponential cosine similarity with temperature hyperparameter τ.

The temporal contrastive loss $\mathcal{L}_{tcl}$ can be augmented with background mixing or pseudo labels. Pseudo labels are derived from centroids and added into a supervised contrastive loss [89], the target pseudo-labels loss $\mathcal{L}_{tpl}$. For the background mixing loss $\mathcal{L}_{bgm}$, the backgrounds of videos in equation 3.1 ($\mathcal{L}_{tcl}$) are shuffled, replacing the backgrounds of positive samples with backgrounds of videos from the opposite domain, pushing the model to focus on the action in the foreground. Backgrounds **BG** are extracted via temporal median filtering and replaced by a weighted average sum similar to mixup

$$\hat{V}^{i\{s\}} = (1 - \lambda) \cdot V^{i\{s\}} + \lambda \cdot \mathbf{BG}^{i\{t\}} \tag{3.2}$$

$$\hat{V}^{i\{t\}} = (1 - \lambda) \cdot V^{i\{t\}} + \lambda \cdot \mathbf{BG}^{i\{s\}} \tag{3.3}$$

where $\hat{V}^{i\{s\}}$ is the background-mixed version of video $V^{i\{s\}}$, $\{s\}$ and $\{t\}$ are the source and target domains respectively, and λ is the mixing hyperparameter.

Finally, source samples are trained with a standard cross entropy loss $\mathcal{L}_{ce}$. The overall loss includes background mixing and pseudo-labeling as

$$\mathcal{L}_{CoMix} = \lambda_{ce}\mathcal{L}_{ce} + \lambda_{bgm}\mathcal{L}_{bgm} + \lambda_{tpl}\mathcal{L}_{tpl} \tag{3.4}$$

CIA Cross-modal Interactive Alignment (CIA) is a recent UDA method proposed by Yang et al. [62] to further exploit the use of multi-modal data for domain adaptation. They present a special architecture to fuse modalities on top of a standard feature extractor, that consists of two parts, a Mutual Complementarity module (MC), and a Spacial Consensus module (SC). The MC is responsible for sharing complementary information across modalities, by reweighing the channels of each feature map with a weight vector obtained from the globally averaged features of the other modalities fed through an MLP. The SC has a similar effect as self-attention but does not introduce additional parameters that could be affected by the domain shift. Instead, it sums the Pearson correlation between the feature maps of different modalities at different resolutions to calculate a consensus map C with which it reweighs the spatial feature maps of each modality. This architecture is designed to improve inter-modality communication with the MC and generalization performance with the SC. Finally, adaptation is done with two domain adversarial classifiers, one at the video level and one at the feature level of each modality. Figure 3.11 illustrates the overall architecture of CIA.

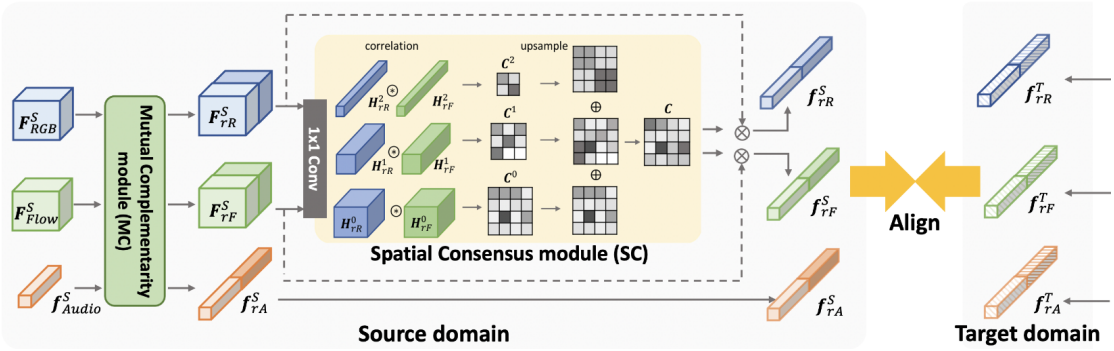


Figure 3.11: Cross-modal Interactive Alignment This figure details the architecture of CIA [62]. The Mutual Complementarity module allows for the exchange of information between the three input modalities, RGB, Flow, and Audio. The Spatial Consensus module is conceptually similar to attention, but it is more robust to domain shift as it does not introduce additional parameters. Domain alignment is done with a domain adversarial network.

TranSVAE The Transfer Sequential Variational Autoencoders (TranSVAE) [43] is a state-of-the-art domain adaptation model that takes a radically different approach from previous work by attempting to disentangle domain-specific and action-specific features. Firstly, Wei et al. propose a VAE-based architecture to disentangle domain-specific and action-specific information in the latent feature representation. Secondly, they do not adapt their model end-to-end but work on pre-extracted features from a standard I3D architecture trained on Kinetics [38]. Figure 3.12 illustrates their architecture, which notably consists of an encoder, a decoder, and a Bi-directional LSTM. The latent space is split into an action-specific domain-invariant sequential component to capture action semantics and an action-invariant static component to encode domain-specific information. A Kullback–Leibler divergence (KL)-based mutual independence loss ensures that the two components store complementary information. Furthermore, a euclidean distance-based contrastive triplet loss pushes static domain encodings from different domains to be further apart than encodings of the same domain. Analogously an adversarial loss enforces domain-independent representations of the dynamic action-specific features. Finally, the task-specific classification loss and a VAE reconstruction loss complete the setup. Figure 3.12 provides an overview of the architecture and where each loss is computed.

Other approaches This section presented the most relevant publications for this work, but many more methods have been proposed to solve UDA for action recognition. Common patterns include a domain adversary component [76, 90], a contrastive loss [60], auxiliary tasks [90], entropy minimization [76], or some form of domain-shift derived attention [76].

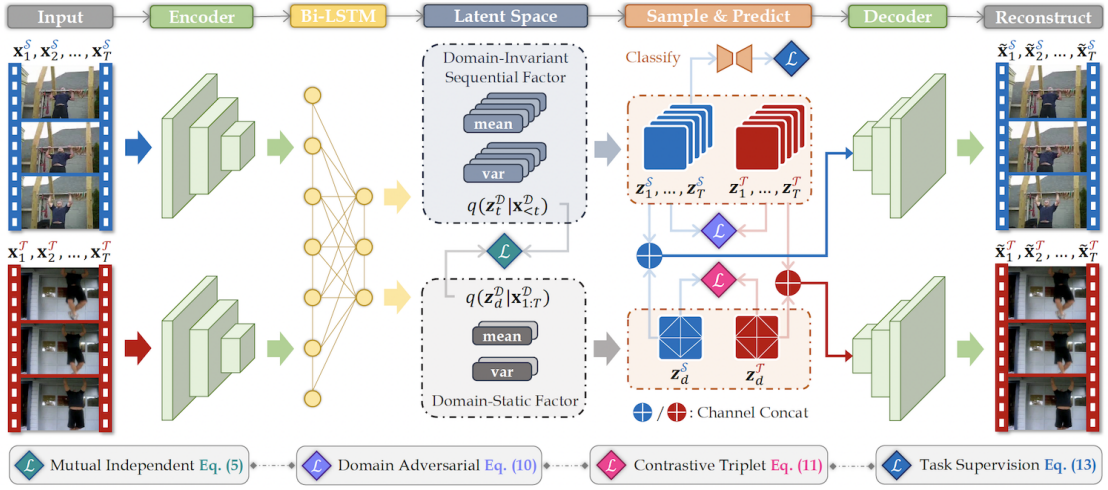


Figure 3.12: Transfer Sequential Variational Autoencoder This figure details the TranSVAE [43] architecture. Overall, the architecture is an autoencoder operating on pre-extracted features. It consists of the encoder, a bi-directional LSTM, and the decoder. The latent space consists of two disentangled factors, a sequential factor for action-specific features and a static factor for domain-specific features. A mutual independence loss on the two factors, domain adversarial alignment of action features, and a contrastive loss on the domain features ensure disentanglement. The AE loss and task supervision complete the model's loss for training.

3.2.3 Source-Free Domain Adaptation

Source-free domain adaptation (SFDA) is a special setting within domain adaptation where the model does not have access to any source data during adaptation. It finds many applications in the real world where data privacy concerns may prevent the publication of training data. However, most conventional domain adaptation techniques violate this additional restriction. Thus, specialized methods are required to address this issue.

How strictly the absence of source information is enforced, however, is not clearly defined. Some researchers propose methods that follow a more relaxed approach which allows the model to be prepared on source data with the specific intent to help with eventual adaptation [91, 37]. This can involve applying domain generalization methods or encoding source-domain information in the network. Others propose their models in a stricter setting, which requires the source model to be trained in a generic way without domain adaptation-specific modifications [92, 93, 94]. Either way, the SFDA setting requires that the source and target domain can not be accessed at the same time.

Notable publications in SFDA include source-free domain adaptation via distribution estimation (SFDA-DE) [95], which proposes to sample surrogate features from estimated source distributions around robust centroid clusters generated by a

k-means algorithm. Kundu et al. [94] propose universal SFDA, which is designed to also tackle the difficult open set DA problem by generating negative samples to emulate unseen classes and using centroid-based clustering to improve discriminability between classes. Style translation was proposed by Hou et al. [92] to apply the style of the target domain to artificially generated source images and adjust batch norm layers accordingly. Computing centroids to generate pseudo labels or compute attention weights is a common pattern in SFDA [96], with Chen et al. bringing the concept to the context of action recognition. Source-free hypothesis transfer (SHOT) [97] enhances the pseudo-labeling concept by using it to make feature clusters in the source domain tighter and fine-tune the feature extractor on the target domain along with an information maximization (IM) loss.

SFDA for action recognition has received much less attention than the more general UDA scenario. Below we present recent publications which we use as a benchmark for this work.

ATCoN Xu et al. propose Attentive Temporal Consistency Networks [98] (ATCoN) to exploit consistency requirements among the different levels of spatial and temporal features in temporal relation networks [49]. While ATCoN is able to achieve major improvements over the source-only baseline in the cross-domain scenario, it is a highly architecture-specific approach. TRNs extract spatial features from each frame and fuse them into a variety of spatial and local temporal features, which are then combined in the global temporal feature map. ATCoN consists of several losses that ensure consistency in the form of KL divergence and metric distance between the different features and partial predictions in both a fixed source model and the adapted model. Furthermore, it adds an information maximization (IM) loss, a cross-entropy loss with centroid-based pseudo-labels, and a local weight module (LWM) to provide temporal attention. Figure 3.13 gives an overview of the network architecture during adaptation.

MTRAN Multimodal and Temporal Relative Alignment Networks (MTRAN) [99] mark a significant improvement on previous SFDA methods for action recognition. The authors propose a unique architecture, which uses a standard I3D model to extract clip-level features, but instead of following the common practice of averaging predictions over five clips, they employ a transformer to fuse the features of several clips before adding a multi-modal fusion layer and a classification layer for the final prediction. The adaptation is then only performed on the transformer, fixing both the feature extractor and classifier. Their novel adaptation loss takes inspiration from mixup [100] and attempts to move the internal representation of target features to be closer to those of the source domain. By splitting the target dataset into samples with high prediction entropy and low prediction entropy, two splits are obtained, one with source-like samples that suffers less from the domain

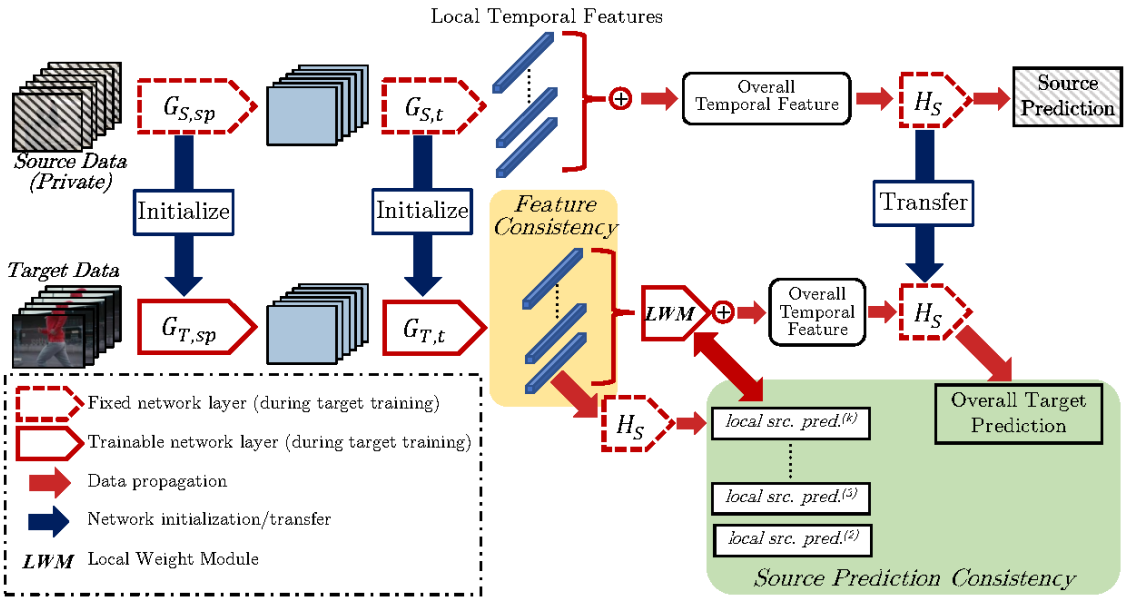


Figure 3.13: Attentive Temporal Consistency Network During adaptation, ATCoN [98] keeps track of both the model that is being adapted and a fixed source model. KL divergence and metric distance losses ensure consistency within the adapted model and with the source model, while an IM loss, a pseudo-labeled cross-entropy loss, and the local weight module (LWM) improve cross-domain performance.

shift, and one with target-like samples that are more affected. Intermediate features are sampled with mixup, and a euclidean loss trains the network to represent more target-like samples in a more source-like way. This step is repeated after the temporal and multi-modal layers, along with an information maximization loss and a pseudo-label loss.

3.2.4 Test-Time Adaptation

In some cases, adapting a model to the target domain before evaluation is infeasible, due to the unavailability of sufficient training data. Sun et al. address this problem by proposing the setting of test-time training [101], where the model is adapted at test-time on each batch individually. This may be extended to online adaptation where updates from previous batches are retained to continuously improve the model. Test-time adaptation (TTA) is a highly restrictive setting due to the limited amount of information available at each adaptation step, but also because of performance limitations, as the inference process should not be slowed down excessively by the adaptation.

In their original paper, Sun et al. [101] propose rotation prediction as an auxiliary task to adapt the entire network on a batch of samples. Later methods often

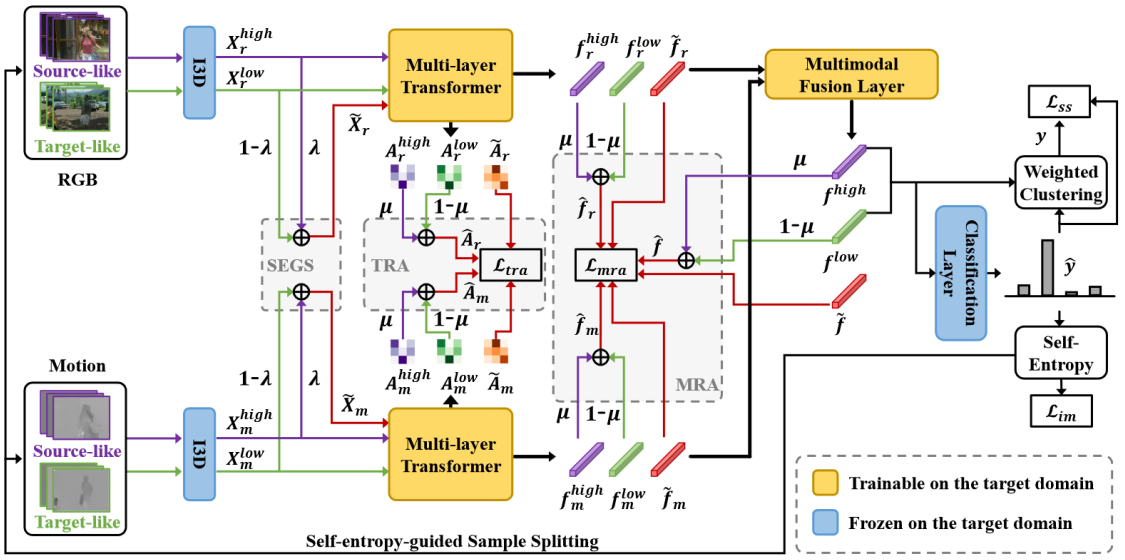


Figure 3.14: Multimodal and Temporal Relative Alignment Network Adaptation architecture of MTRAN [99]. During adaptation, only the transformers and fusion layer are updated, while parameters of the I3D feature extractors and classifier are frozen. The target dataset is split by prediction entropy into source-like and target-like features. A mixup-inspired alignment loss pushes internal representations of target-like multi-modal and temporal features to appear to be more source-like. This alignment loss is augmented by a pseudo-label loss and an IM loss.

focus on updating the batch norm statistics [102] as an essential part of the adaptation process. Single image test time adaptation [103] enables adapting the model on batches consisting of a single image, by estimating its batch-norm statistics on a wide range of augmentations and computing a weighted average of the source and estimated target statistics. Test time entropy [104] employs an entropy minimization loss, which has been studied extensively in SFDA as a self-supervised loss [97, 76] to update the statistics. Finally, T3A [105] proposes to replace the final classification layer with a robust centroid distance classifier.

Test-time adaptation for action recognition has not received attention until very recently, with Planamente et al. [106] proposing the use of self-supervised losses in combination with training on multiple source domains and using other domain generalization methods.

Chapter 4

Proposed Solution

Domain shift is an issue that persists in video action recognition, however, most research focuses on the unsupervised domain adaptation setting. While many interesting advances have been made in this area, its application to the real world is limited. UDA requires both the source and the target datasets to be available during the adaptation process, which carries two major drawbacks. Primarily, it requires the source dataset to be available for the adaptation process. This raises privacy concerns, as the source dataset may contain sensitive information about individuals or confidential data that give a competitive advantage. Instead, it is common for companies to provide plug-and-play models that are pretrained on proprietary data. Furthermore, in the UDA setting models need to be adapted separately for each pair of source and target domains. Transferring the model to a new domain necessitates the adaptation to be redone, requiring both the source dataset and the new target dataset. Together this means, that UDA can not realistically be applied by the end user. Finally, UDA methods require the target dataset to be available as well, meaning the model provider can not perform adaptation either due to the same privacy concerns.

Domain generalization helps mitigate this issue by providing techniques that enable training on multiple source datasets simultaneously and improve model robustness. However, DG is fully done at training time and does not integrate any knowledge from the target domain. Thus, methods often do not hold up under general assumptions and still suffer from domain shift [107]. Source-free domain adaptation, and by extension test time adaptation, addresses this by working on the target domain without any knowledge of the source domain. TTA is a relatively new research area [101] which is able to provide significant performance enhancements with little additional computational load. However, it is also a very strict setting, working in real-time with limited data. Many recent works update the model parameters on a per-batch basis, relying on large batch sizes to improve accuracy and robustness. This is not in line with real-world problems which generally require immediate predictions, making batch collection infeasible. Therefore,

we see SFDA as an important setting to investigate, having a significant and direct impact on realistic problem settings.

So far, source-free domain adaptation has been paid limited attention by the action recognition research community. Prior works include ATCoN [98] and MTRAN [99], with both methods being designed to work with a specific model architecture. This represents a major limitation, as network architectures often change and evolve over time. A more general approach was proposed with SFTADA [37], however, it employs centroid-based attention in the source model, which is replaced with a source-trained MLP to estimate the attention in the target model. This violates both the strict absence of source information in the SFDA setting, as well as the standard practice of using the same network architecture in the source and target domains.

With this work, we want to address the gap in the research by proposing a general, architecture-independent method to adapt a model under strict source-free conditions.

	source domain		target domain	
	training	adaptation	adaptation	online
DG	✓	-	-	-
UDA	✓	✓	✓	-
SFDA	-	-	✓	-
TTA	-	-	-	✓

Table 4.1: Different settings in domain adaptation

4.1 Motivation

Our underlying hypothesis states that the extracted features of the network are sufficient to keep most of the network's discriminative performance. In fact, the main reason for the performance drop across domains comes from the shallow classification module which operates on the feature space. Most modern architectures for action recognition use the same backbone, either ResNet-101 for spatial features or I3D for combined spatiotemporal feature extraction. Additional modules to improve multi-modal or temporal feature extraction may be added, however, the majority of the features are extracted within the backbone. These backbones come pretrained on large datasets, usually ImageNet [25] for ResNet and Kinetics [72] for I3D. The feature extractors are then trained on the source domain together with a classification layer. Nonetheless, they retain much of their general feature-extracting capabilities from pretraining and are thus able to extract robust domain-invariant features. The classifier, on the other hand, is dependent on the number of classes and is usually trained from a random initialization on

the source dataset. We thus hypothesize, that the classifier contains much more domain-specific information and is responsible for the majority of the cross-domain performance drop on the target domain. Furthermore, it encodes dataset-specific biases like the class imbalance and relevance of features for each class.

Recent TTA methods address the domain shift by training the model on each batch using self-supervised entropy-based losses. This can range from updating just the batch norm statistics [103] to training the batch norm parameters, classifier [106], or the entire model end-to-end [104]. However, while these methods often improve the per-batch performance, they do not improve the general performance, in fact, when training online with more than one batch the model quickly collapses. These self-supervised losses alone are thus unsuitable for the SFDA setting. Nonetheless, they do find some application in the form of regularization, combined with other SFDA methods that guide the model to adapt in a more general manner.

Using prototype-generated pseudo-labels to guide the adaptation process is common practice in source-free domain adaptation literature [97, 94, 37, 99] and to a lesser extent in general unsupervised domain adaptation [52, 108]. Generally, these prototypes are simply the mean feature vectors with only minor variations between proposals. Centroid-based pseudo-labels have several useful properties: they are computationally light, agnostic to the underlying architecture, combine well with other losses, and implicitly reduce prediction entropy by encouraging decision boundaries to pass through low-density regions in the feature space. Instead of using centroids to just generate pseudo-labels, we propose to replace the classification layer with a centroid-based classifier entirely, directly benefiting from these qualities. Furthermore, this allows us to forego gradient-based optimization, significantly reducing adaptation overhead and leaving the source model intact. We can thus seamlessly adapt a single model to several independent target domains, by using separate centroids for each domain. Finally, our proposed method is robust with respect to continuous domain shift and can improve over time, as new samples can be added to the support sets, directly updating the centroids.

4.2 Predictions from Pseudo-Prototypes

Following Iwasawa and Matsuo [105] we propose to replace the standard classification module of deep learning models with an efficient prototypical classifier built under self-supervision from unlabeled training samples on the target domain, without changing the network's parameters.

Conceptually, the classification model $f = h_\omega \circ g_\psi$ can be split into two parts, a feature extractor g_ψ and a classifier h_ω. Generally, the classifier is a very shallow network consisting of just a single layer, defined by the weight matrix ω. Thus, the

predicted probability of a sample x belonging to class k equates to

$$p(\hat{y}_i = k|x_i, \psi, \omega) = \frac{\exp(\omega^k \cdot g_\psi(x_i))}{\sum_j \exp(\omega^j \cdot g_\psi(x_i))} \tag{4.1}$$

where ω^k is the kth row ω and $\hat{y} = f(x)$. Thus, ω^k can be seen as a template representing the kth class, with the dot product as a similarity measure in the feature space. Class probabilities of a sample x are then computed as the softmax over the similarity of its feature representation $g_\psi(x)$ with each class template ω^k. As discussed in section 4.1, we assume that the feature extractor g_ψ has strong generalization capabilities and produces robust domain-agnostic features, while the classifier h_ω holds domain-specific biases and suffers strongly from the domain shift. Therefore, we propose to replace the source-specific template ω^k with a target-specific estimate from a training dataset in the target domain.

Given an unlabeled dataset X in the target domain, we must first establish which features contribute to which prototype. Thus, the model assigns each sample $x_i \in X$ a pseudo label $\hat{y}_i$ using the source classifier $\hat{y}_i = f(x_i)$. Finally, the prototype representing class c_k is computed as the mean normalized feature vector of all samples belonging to class k

$$c_k = \frac{1}{N} \sum_{\substack{i \\ \hat{y}_i = k}} \frac{g_\psi(x_i)}{\|g_\psi(x_i)\|} \tag{4.2}$$

At test time predictions are then made using the softmaxed distance of a sample's feature vector to the classes centroid

$$p(\hat{y}_i = k|x_i, \psi) = \frac{\exp(g_\psi(x_i) - c_k)}{\sum_j \exp(g_\psi(x_i) - c_j)} \tag{4.3}$$

To improve the method's robustness to outliers and the accuracy of the assigned pseudo-labels, the set of features to compute each centroid following equation 4.2 can be reduced to a subset of all samples. Iwasawa and Matsuo [105] propose to select the features of the m samples with the lowest prediction entropy for each class, which works well on many datasets. Further improvements can be obtained by replacing the pseudo labels extracted from the source classifier with the newly generated centroid-based pseudo-labels in equation 4.2. This process can be repeated several times, updating the centroids similarly to the k-means algorithm.

Finally, we make some adjustments specific to the video setting. For multi-modal training, we use the joint predictions to generate the same pseudo-labels for all modalities in equation 4.2. Similarly, when computing centroids on only a subset of features, the same subsets of samples are used across all modalities. Furthermore, video action recognition models are usually evaluated on five clips for each video. We exploit this additional information by improving initial pseudo-labels and computing the centroids over the mean feature of 5 separate clips.

4.3 Problem Formulation

Formally, a domain $\mathcal{D} = \{\mathcal{X}, \mathcal{P}(X)\}$ consists of a feature space $\mathcal{X}$ with random input variables $X \in \mathcal{X}$ and a probability distribution $\mathcal{P}(X)$ over the feature space. A task $\mathcal{T} = \{\mathcal{Y}, \mathcal{P}(Y|X)\}$ comprises labels Y in a labels space $Y \in \mathcal{Y}$ and a distribution of expected outputs for each input $\mathcal{P}(Y|X)$. Machine learning models are designed to learn a hypothesis $h : \mathcal{X} \mapsto \mathcal{Y}$ that assigns labels Y to inputs X.

In a supervised setting this happens by sampling inputs from the source domain $\mathcal{D}^S = \{\mathcal{X}^S, \mathcal{P}(X^S)\}$ and making the network learn the hypothesis h^S to predict the corresponding outputs according to the source task $\mathcal{T}^S = \{\mathcal{Y}^S, \mathcal{P}(Y^S|X^S)\}$. The network is then evaluated on the target domain $\mathcal{D}^T$ and task $\mathcal{T}^T$. Generally, machine learning algorithms assume the source and target domain and task to be the same $\mathcal{D}^S = \mathcal{D}^T$, $\mathcal{T}^S = \mathcal{T}^T$. If this assumption is violated the learned hypothesis h^S is not valid anymore and the model's performance may degrade. The difference between two domains is called the domain shift.

Domain shift can occur for many reasons, due to statistical variation nearly every dataset contains a small amount of domain shift between its splits, even if they are collected following the same protocol. Larger domain shifts can be caused by external factors such as different camera sensors, differences in the environment, weather, season, time of day, or lighting. One of the first datasets dedicated to capturing the domain shift is the Office-31 dataset [109], which features images of everyday office objects downloaded from amazon and captured with a DSLR camera and a webcam.

Transfer learning is the process of learning a hypothesis h^T in one domain utilizing information from another domain $\mathcal{D}^S$, $\mathcal{T}^S$ to improve the result. The area has received much attention and it is common practice nowadays to initialize models with weights pretrained on large datasets [110, 111].

Applying pretraining often enables greatly reduced training times, as the network already starts with a good initialization of parameters. Furthermore, it can boost performance and improve generalization capabilities, by providing an initial understanding of features that may not be entirely learnable from the target training data. Finally, transfer learning makes it possible to achieve good results on tasks or datasets that are too small to be normally used, because collecting the data is so difficult.

Domain adaptation is a subset of transfer learning, where the task $\mathcal{T}^S = \mathcal{T}^T$ and feature space $\mathcal{X}^S = \mathcal{X}^T$ are the same between the source and target domain, but the samples in the datasets follow a different probability distribution $\mathcal{P}(X)^S \neq \mathcal{P}(X)^T$.

The adaptation process can be roughly divided into three stages, training on the source dataset, adapting the model, and evaluation on the target dataset. Differences between domain adaptation settings are mostly defined by what data is available at which stage. On the one extreme domain generalization exists, where the model is exclusively trained on the source domain, without any knowledge of

potential target domains on which the model could be evaluated. Unsupervised domain adaptation is the most widely studied setting, in which the model is jointly trained on labeled source data and unlabeled target data. Source-free domain adaptation is a subset of UDA, where the supervised training of the source model and its unsupervised adaptation are strictly separated, with no data being shared between the two steps outside of the model itself. Finally, test time adaptation sits at the other end of the spectrum by simultaneously evaluating and adapting the model on a batch per-batch basis. Table 4.1 gives an overview of the different settings that were discussed in this section.

Chapter 5

Experiments

This chapter presents the experiments we performed to evaluate the effectiveness of our method. Firstly, section 5.1 defines the technical details of our implementation and the experimental setup. We then start by comparing our method with the state-of-the-art in source-free domain adaptation, unsupervised domain adaptation, and test-time adaptation in section 5.2. The later sections are intended to give a theoretical analysis of our method. Section 5.3 looks at the dataset bias and its effect on neural networks. Furthermore, section 5.4 highlights the advantages of multi-modal learning. Finally, section 5.5 performs an ablation study on our method, comparing the effect of each individual component.

5.1 Implementation Details

Datasets We evaluate our proposed method on the two most commonly used datasets for recent works in domain adaptation for action recognition: **EPIC Kitchens 55** [30] has egocentric videos from three different domains and **UCF-HMDB**$_{full}$ [74, 73] following the setting proposed by Chen et al. [76].

Implementation Our model uses a two-stream I3D backbone pretrained on kinetics with averaged late fusion to process multi-modal inputs. Flow frames are extracted using the TV-L1 algorithm [63]. Clips consist of 16 frames following the dense sampling strategy with random cropping, scale jitters, and horizontal flipping as augmentations. We follow the standard practice of evaluating our model using the average prediction on 5 clips per video unless otherwise stated.

Computational Resources Our experiments were run in two different setups: a workstation with two NVIDIA GeForce GTX 1070 GPUs; on a single NVIDIA-V100 GPU provided by hpc@polito, which is a project of Academic Computing

within the Department of Control and Computer Engineering at the Politecnico di Torino (http://www.hpc.polito.it).

5.2 Comparison with the State-of-the-Art

Most research on domain adaptation for action recognition has focused on the more general unsupervised domain adaptation (UDA) scenario, with source-free domain adaptation (SFDA) receiving much less attention. Therefore, we compare our results to state-of-the-art publications in both settings, however, the most relevant benchmarks for us remain the SFDA methods: ATCoN [98], SFTADA [37], SHOT [97] and MTRAN [99].

method	backbone	SF	MM	U→H	H→U	avg
ATCoN [98]	ResNet-101	✓	-	79.7	85.3	82.5
SFTADA [37]	ResNet-101	✓	✓	87.2	91.2	89.2
STCDA [112]	I3D	-	-	83.1	92.1	87.6
CoMix [52]	I3D	-	-	86.7	93.9	90.3
CIA + TA3N [62]	I3D	-	-	91.94	94.57	93.26
TransVAE [43]	I3D	-	-	87.78	**98.95**	93.37
TA3N [76, 90]	I3D	-	-	81.4	90.5	85.95
SAVA [90]	I3D	-	-	82.2	91.2	86.7
MM-SADA [59, 60]	I3D	-	✓	84.2	91.1	87.65
CIA [62]	I3D	-	✓	90.56	94.22	92.39
[60]	I3D	-	✓	84.7	92.8	88.75
SHOT [97, 99]	I3D	✓	✓	89.72	91.77	90.75
MTRAN [99]	I3D	✓	✓	**92.22**	95.27	93.75
ours	I3D	✓	✓	**92.22**	**98.95**	**95.59**

Table 5.1: Comparison with SOTA on UCF-HMDB$_{full}$ This table compares our results with other domain adaptation methods on the UCF-HMDB$_{full}$ dataset. The second and third column state whether the proposed method is source-free (SF) and uses multi-modal data (MM). The best values of each column are indicated in **bold** writing. Our method outperforms all previous SFDA and UDA techniques on all shifts.

UCF-HMDB$_{full}$ Table 5.1 shows that our method outperforms not just previous SFDA methods, but all UDA methods as well, on both domain shifts, with an average accuracy of 95.59%, an improvement of 5.74% over the source-only model. However, one should note that each method reports different source-only and target-only performances, making a direct comparison difficult. These differences can be the result of differences in architecture, experimental setting, training process, or simply statistical variations. For example, two commonly used backbones, are

ResNet-101 [16] and I3D [38]. As a 3D CNN I3D has significantly more parameters than ResNet-101 and generally achieves better results.

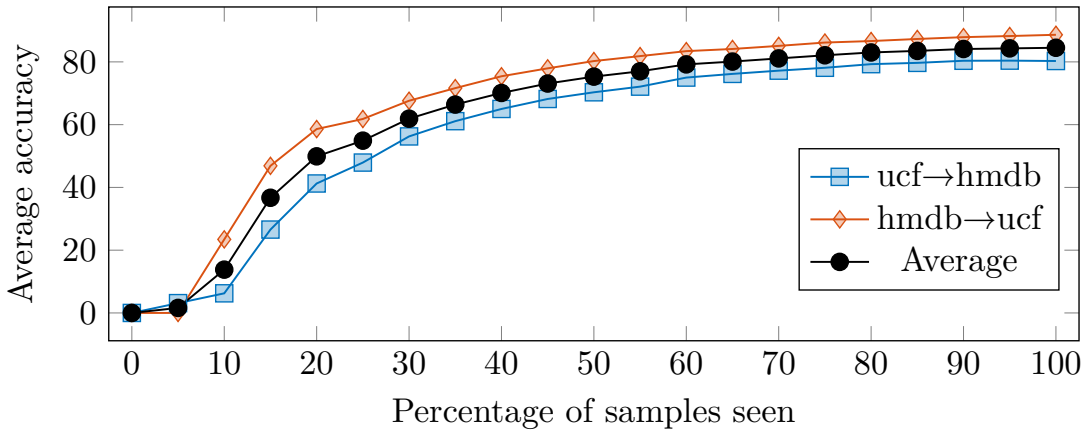
EPIC-Kitchens-55 Compared to UCF-HMDB$_{full}$, EK55 is a much more challenging dataset, both in the supervised setting and even more so for domain adaptation. Table 5.2 reports our results in comparison with state-of-the-art methods and shows that our method outperforms previous SFDA work and closes the gap to the more general UDA setting.

method	SF	MM	D2→D1	D3→D1	D1→D2	D3→D2	D1→D3	D2→D3	avg
DANN [84, 52]	-	-	38.3	38.8	37.7	42.1	36.6	41.9	39.23
TA3N [76, 43]	-	-	40.9	39.9	34.2	44.2	37.4	42.8	39.90
CoMix [52]	-	-	38.6	42.3	42.9	49.2	40.9	45.2	43.18
TransVAE [43]	-	-	**50.3**	48	50.5	**58**	**50.3**	**58.6**	**52.62**
MMD [79, 62]	-	✓	46.6	39.2	43.1	48.5	48.3	55.2	46.82
MM-SADA [59]	-	✓	48.2	50.9	49.5	56.1	44.1	52.7	50.25
[60]	-	✓	49.5	51.5	50.3	56.3	46.3	52	50.98
STCDA [112]	-	✓	49	**52.6**	52	55.6	45.5	52.5	51.20
CIA [62]	-	✓	49.8	52.2	**52.5**	57.6	47.8	53.2	52.18
SHOT [97, 99]	✓	✓	44.09	53.99	40.77	36.45	49.03	45.34	44.95
MTRAN [99]	✓	✓	46.33	**58.15**	42.21	38.12	**52.33**	46.12	47.21
ours	✓	✓	**52.06**	50.46	**45.6**	**52.27**	41.07	**55.13**	**49.43**

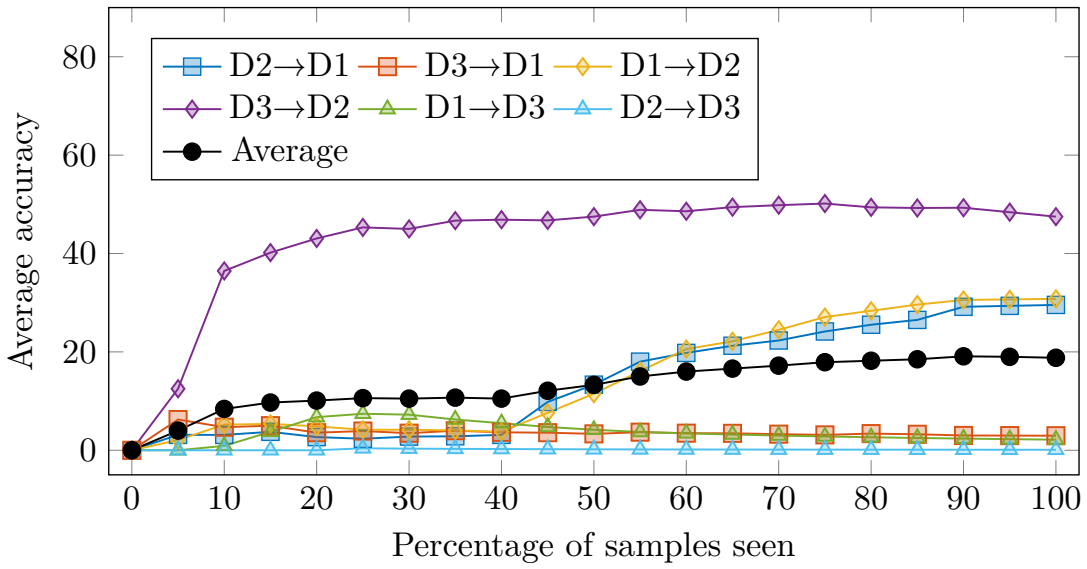
Table 5.2: Comparison with SOTA on EPIC-Kitchens-55 This table compares our results with other UDA methods on EPIC-Kitchens-55. Columns 2 and 3 indicate whether the model was adapted in the source-free setting (SF) and if it uses multiple modalities (MM). The best results for SF are highlighted in **bold blue** and for non-SF methods in **bold**. Our experiments show that we are able to beat all state-of-the-art source-free methods and close the gap between SFDA and the more relaxed UDA.

5.2.1 Comparison with Test-Time Adaptation

Test-time adaptation (TTA) methods often operate under the source-free assumption, with the additional constraint of no access to a training dataset in the target domain. It is thus interesting to compare our proposed SFDA method with state-of-the-art TTA techniques for action recognition. Specifically, we benchmark our results against a variety of self-supervised losses utilized for action recognition TTA by Planamente et al. [106] and Peirone [113], and Test-Time Template Adjusters (T3A) [105], a method that is conceptually similar to ours. Tables 5.3 and 5.4 show the results of this analysis on both datasets.



(a) UCF-HMDB$_{full}$



(b) EPIC-Kitchens-55

Figure 5.1: Accuracy of T3A over share of test dataset seen When using T3A [105] to adapt a model to a new domain, the predictions improve over time, as more target samples are added to the support set and the centroids become more robust and representative. This figure shows how the average accuracy over all seen data develops over time, in relation to how much of the target dataset has been evaluated. On UCF-HMDB$_{full}$ T3A quickly improves, having competitive accuracy on the later few batches, however, the overall average accuracy stays below the source-only benchmark. For the more difficult EPIC-Kitchens-55 dataset, T3A struggles to adapt. While on D3→D2 it improves immediately, D1→D2 and D2→D1 take much longer to find good centroids, and the remaining domain shifts fail completely.

T3A Despite the similarity between T3A and our method, applying T3A does not lead to any improvements over the source-only baseline, but rather a drastic performance decrease. To explain these unexpected observations one has to look at the way T3A functions. We run T3A in the online learning scenario, where the support set for each centroid persists between subsequent batches. Figure 5.1 shows how the average accuracy over the seen part of the test dataset develops as more batches are evaluated and the support set grows. A "warmup" period can be observed across all analyzed data shifts when T3A has not yet gathered enough features to make reliable predictions. In most cases, this issue is eventually resolved. For UCF-HMDB$_{full}$ the final accuracy almost reaches the source-only baseline, however, on the more difficult EPIC-Kitchens-55 dataset, T3A performs much worse and totally fails to adapt on three of the domain shifts.

Of course, these results are highly dependent on the batch size at test-time, since the main bottleneck of T3A appears to be the number of samples included in the centroid computations. When adaptation is done on a per-batch basis with a batch size of 16, the model collapses entirely, making predictions that are indistinguishable from random guessing. If the batch size is equal to the total number of samples in the test dataset, all samples are included in the support set and performance significantly increases, outperforming all other SFDA and TTA methods. This, however, is an unrealistic setting, since TTA requires predictions to be made while simultaneously adapting the model. Collecting the samples of the entire test dataset would mean separating the adaptation and evaluation stages, breaking the fundamental assumption of TTA. Furthermore, doing so would violate the strict separation of training and test data, making the adapted model biased toward the already-seen test data.

Self-Supervised Losses Following previous work on TTA for action recognition [106, 113] we conduct experiments with a variety of self-supervised losses, specifically entropy, information maximization, minimum class confusion [114], complementary entropy, and relative norm alignment [106]. Furthermore, we evaluate methods, that propose to adjust the batch norm layers, such as TENT [104] and TENT-C [106].

While these methods show great promise for adapting models that were prepared with DG techniques or trained on multiple source modalities [113], the results do not seem to translate to the general case. Table 5.4 shows that the best method, MCC with three adaptation steps per batch, shows a decrease in performance on most shifts of EPIC-Kitchens-55 when compared to the source-only baseline. On the easier UCF-HMDB$_{full}$ dataset all methods achieve a minor improvement on the source-only baseline of just under 1%.

However, none of the model updates generalize to data outside of the batch they were computed on. When keeping updates of previous batches in the online TTA scenario all models quickly collapse, either always predicting a single class or

making the predictions indistinguishable from random guesses. Thus, these TTA methods do not translate well to the SFDA setting except as auxiliary losses to regularize the main adaptation loss.

method	setting	U→H	H→U	avg
source-only		84.40	94.75	89.57
MCC [114]	TTA	85.94	95.14	90.54
T3A [105]	online-TTA	80	88.64	84.32
MTRAN* [99]	SFDA	91.67	95.28	93.47
ours	SFDA	**92.22**	**98.95**	**95.59**
target-only		97.50	99.48	98.49

Table 5.3: Architecture-indpendent comparison on UCF-HMDB$_{full}$

method	setting	D2→D1	D3→D1	D1→D2	D3→D2	D1→D3	D2→D3	avg
source-only		46.33	46.79	47.6	55.33	**43.02**	50.31	48.23
MCC [114]	TTA	47.79	44.34	**51.87**	55.33	40.73	45.69	47.29
T3A [105]	online-TTA	29.55	2.96	30.78	47.49	2.15	0.01	18.8
MTRAN* [99]	SFDA	50.92	43.58	51.07	**55.33**	41.48	53.08	48.91
ours	SFDA	**52.06**	**50.46**	45.6	52.27	41.07	**55.13**	**49.43**
target-only		64.6	64.6	76.4	76.4	72.9	72.9	71.3

Table 5.4: Architecture-indpendent comparison on EPIC-Kitchens-55

5.2.2 Architecture-Independent Relative Norm Alignment

Multi-modal and temporal relative alignment networks (MTRAN) [99] are a recently proposed SFDA method for action recognition that achieves state-of-the-art performance across several benchmark datasets. Our main criticism of MTRAN, however, is its architecture-specific formulation that makes it unsuitable for real-world problems, where architectures evolve over time and depend on specific task requirements.

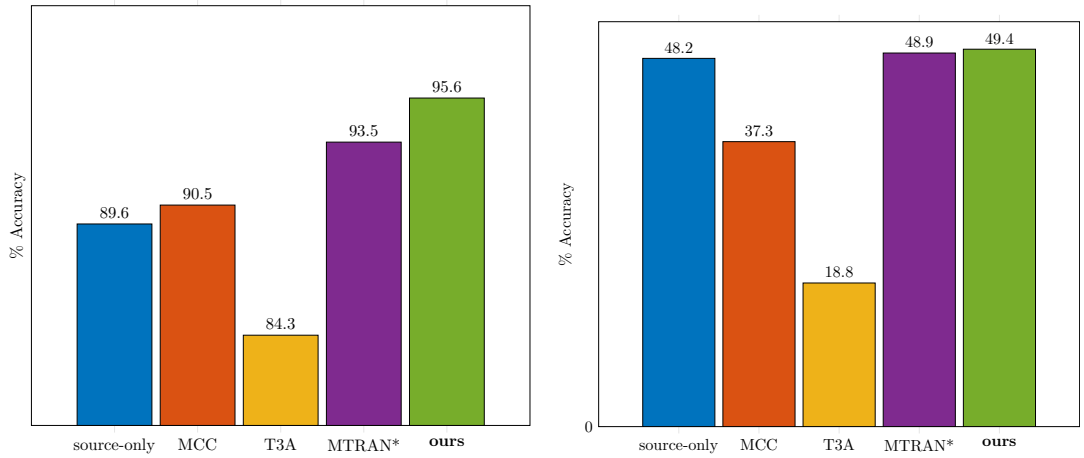
To this end, we implement MTRAN*, a naive architecture-independent version of MTRAN, and apply it to our two-stream I3D architecture. Instead of freezing the feature extractor and adapting the transformer and multi-modal fusion layer, MTRAN* adapts the feature extractor itself. Other than this, MTRAN* retains the IM and pseudo-label loss from MTRAN but changes the relative alignment losses to operate on the I3D backbone, mixing raw inputs and aligning features separately for each modality.

Tables 5.3 and 5.4 show that our generalized MTRAN* version performs on par with MTRAN, even achieving a slightly higher accuracy on EPIC-Kitchens-55. Furthermore, we note that the pseudo labels for MTRAN* are generated by

method	setting	D2→D1	D3→D1	D1→D2	D3→D2	D1→D3	D2→D3	avg
source-only		46.33	46.79	47.6	55.33	**43.02**	50.31	48.23
MTRAN* [99]	SFDA	50.92	43.58	**51.07**	55.33	41.48	53.08	48.91
ours	SFDA	52.06	50.46	45.6	52.27	41.07	**55.13**	49.43
ours + MTRAN*	SFDA	**55.05**	**50.69**	50.00	**55.87**	40.76	53.29	**50.94**
target-only		64.6	64.6	76.4	76.4	72.9	72.9	71.3

Table 5.5: Synergies with MTRAN* on EPIC-Kitchens-55

our proposed method and MTRAN* itself solely adjusts the feature extractor. We thus expect our method to synergize well with MTRAN*, especially on difficult datasets like EPIC-Kitchens-55, where the domain shift in the feature extractor plays a more important role. In fact, table 5.5 confirms this hypothesis, showing that the combination of our method with MTRAN* achieves even better results.



(a) UCF-HMDB$_{full}$ **(b) EPIC-Kitchens-55**

Figure 5.2: Architecture-Independent Comparison This figure compares the average accuracy of several methods where we reran the experiments on the same architecture as our method. Note that these methods are defined in different settings: MCC is a TTA method, T3A was run as online TTA, and MTRAN* and our method are both SFDA

5.3 Class Imbalance

Not all actions are equal, some are performed much more frequently than others. Figure 3.7 shows that taking something is the most common task in EPIC-Kitchens-100 while bending objects occurs much more rarely. Imbalances in the relative frequency of classes have a significant impact on the performance of machine learning models. Some datasets, like HMDB51 [73], are artificially balanced, to allow

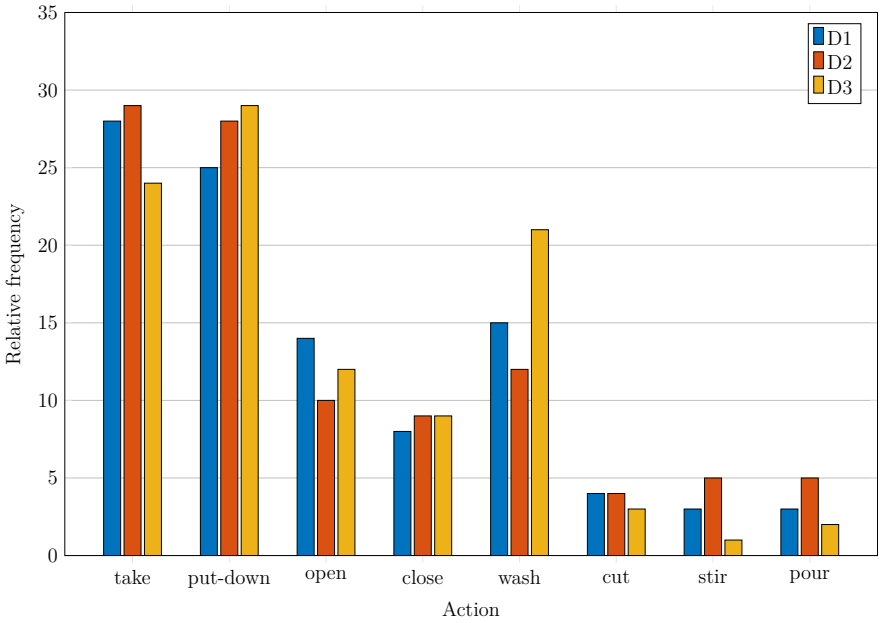


Figure 5.3: Class Imbalance in EPIC-Kitchens-55 This figure shows the relative frequency of each class in each of the three kitchens of the UDA EPIC-Kitchens-55 dataset.

for a fair evaluation of each class. Real-world datasets, however, are usually not adjusted, making the class imbalance an additional challenge to overcome. In the cross-domain scenario, this becomes an even bigger factor, as the distribution of classes may change from one domain to the next. This section explores the impact of imbalanced classes in the EPIC-Kitchens-55 dataset.

method	D2→D1	D3→D1	D1→D2	D3→D2	D1→D3	D2→D3	avg
source-only	46.33	46.79	47.6	55.33	43.02	50.31	48.23
ours	52.06	50.46	45.6	52.27	41.07	55.13	49.43
target-only	64.6	64.6	76.4	76.4	72.9	72.9	71.3

Table 5.6: Average accuracy on EPIC-Kitchens-55

method	D2→D1	D3→D1	D1→D2	D3→D2	D1→D3	D2→D3	avg
source-only	35.89	33.39	37.23	53.78	23.64	36.84	36.80
ours	59.80	53.17	56.97	63.67	40.55	47.46	53.60
target-only	67.95	67.95	79.76	79.76	55.83	55.83	67.85

Table 5.7: Average per-class accuracy on EPIC-Kitchens-55

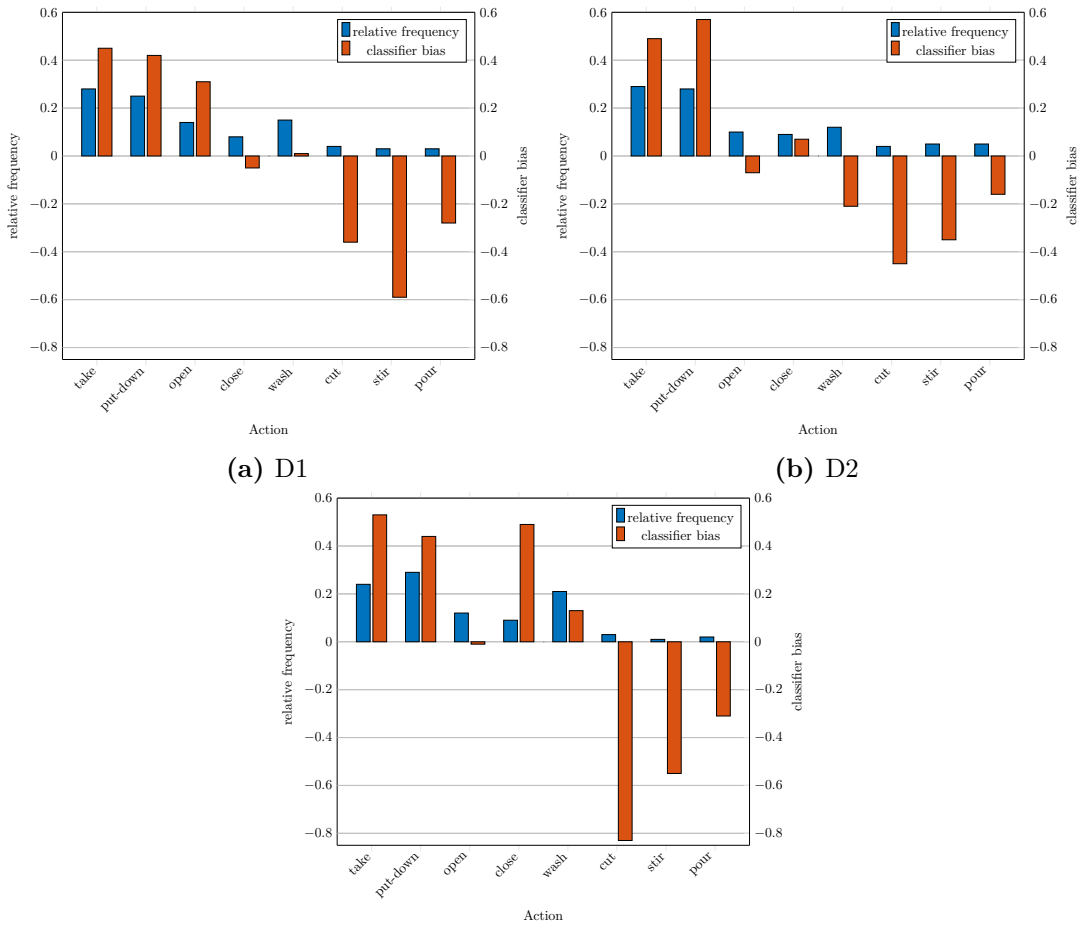
5.3.1 Imbalance in the Dataset

Figure 5.3 shows how the UDA EPIC-Kitchens-55 dataset is affected by the imbalanced class problem, with two dominant classes, three classes with very few samples, and different distributions of classes in each of the three kitchens. This naturally impacts the results, when comparing the average accuracy in table 5.6 with the average per-class accuracy in table 5.7 we observe that the average per-class accuracy is 3.45% lower than the average sample accuracy in the supervised setting, meaning that less frequent classes are identified with lower accuracy. In the cross-domain source-only scenario this discrepancy is dramatically increased, to 11.43%.

5.3.2 Imbalance in the Classifier

Section 4 previously discussed the role of the classification layer in the loss of performance across domains. To analyze this in the context of imbalanced classes, we use the bias parameter of the classification layer as a proxy for the biased internal representation of class imbalances. Figure 5.4 plots the classifier bias of the target-only model and the actual class bias in the dataset for each class and domain, confirming a strong correlation between the two. When applying the source-only model to a different domain this bias remains fixed, while the layer outputs change, resulting in the much lower per-class accuracy on the source-only dataset.

The inherent absence of this class bias in our centroid-based classifier is one of the central arguments in favor of our method. We thus expect our method to be much more robust with respect to this type of domain shift. In fact, tables 5.6 and 5.7 show that our method achieves a higher average per-class accuracy than average sample accuracy. This means that our method achieves higher accuracies classifying less common classes, meaning it is more fair and robust to dataset bias [2].



(a) D1 **(b)** D2

(c) D3

Figure 5.4: Class Distributions and Classifier Bias These figures plots the relative frequency of each class and classifier bias for each domain of the EPIC-Kitchens-55 dataset. A strong correlation between the two variables can be observed.

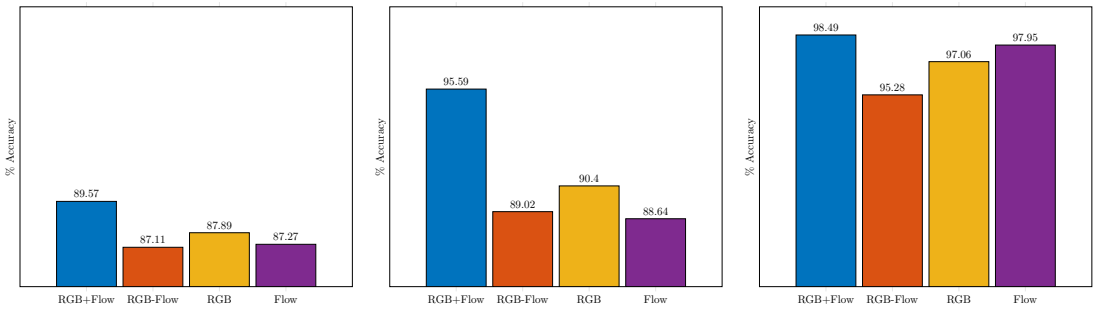
5.4 Multi-Modal Learning

In this section, we analyze the effect multi-modal learning has on the base model and our proposed method by looking at the results on the UCF-HMDB$_{full}$ dataset. Figure 5.5 compares the average accuracy of our method with the source-only and target-only on a variety of modalities.

Firstly, we observe that training and evaluating the model on both RGB and flow modalities achieves the best results across all experiments, showing that the base model is able to extract complementary information from each modality. In the cross-domain scenario, this additional information is even more relevant, as we expect each domain to experience a unique domain shift, enabling a multi-modal to

average out much of the cross-domain statistical variation. Comparing the source-only and target-only experiments confirms this hypothesis as the relative improvement of using multiple modalities increases in the source-only case. Furthermore, we note that our method especially profits from the complementary information in each modality, achieving a much higher improvement on the multi-modal model.

Finally, our experiments highlight the difficulty of training multi-modal models. When evaluating only the RGB modality of a multi-modal model, it performs significantly worse than the RGB model. This presents a known issue and some research has gone into improving multi-modal models in this regard [57, 58].



(a) source-only **(b) ours** **(c) target-only**

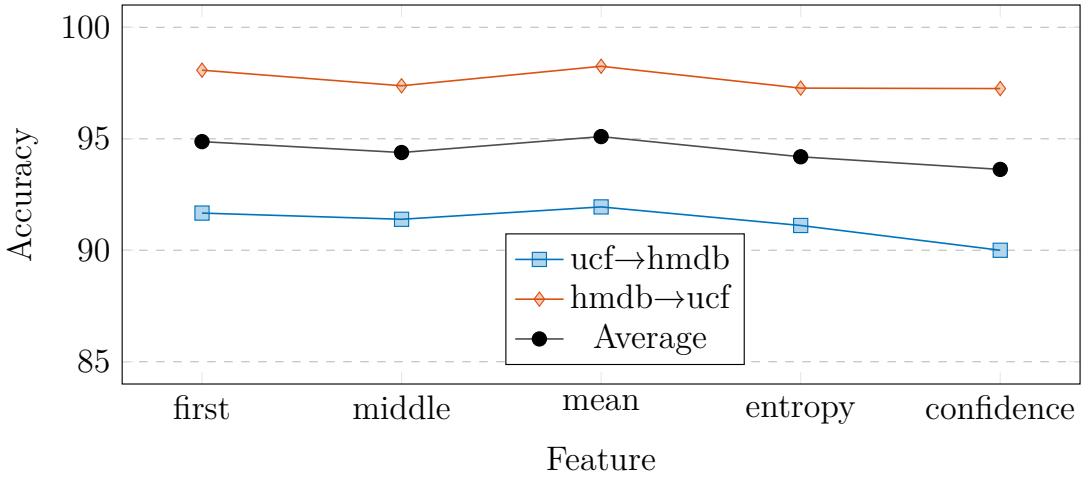
Figure 5.5: The Impact of Multi-Modal Training This figure compares our model with the source-only and target-only on different modalities. Specifically, we report the average accuracy on UCF-HMDB$_{full}$ for the following settings: trained and evaluated on RGB+Flow (blue), trained on RGB+Flow and evaluated on RGB (orange), trained and evaluated on RGB (yellow), trained and evaluated on Flow (purple).

5.5 Ablation Study

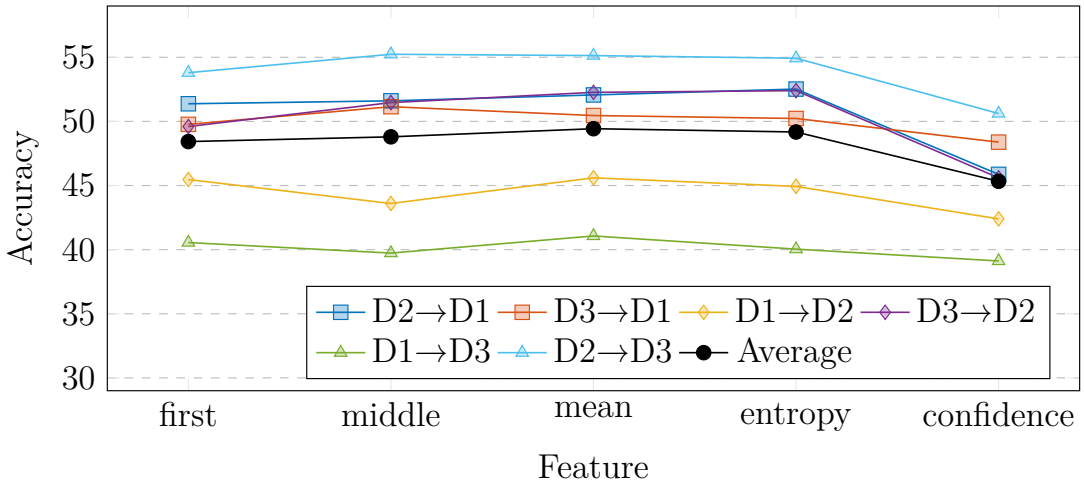
In this section, we perform a brief ablation study on the most important design choices of our proposed method. Specifically, we analyze the accuracy depending on the choice of features to represent a multi-clip video, using only a subset of features to compute the centroids, updating centroids iteratively similar to k-means, and different distance measures in the feature space.

5.5.1 Which is the best feature?

It is standard practice for action recognition models to be evaluated on the average prediction over 5 clips from the same video to get a more robust prediction. Since the initial step of our proposed method involves assigning pseudo-labels to the samples in order to assign them to a centroid, it suggests itself to improve these predictions in a similar manner. Additionally to using the 5 clips to improve our

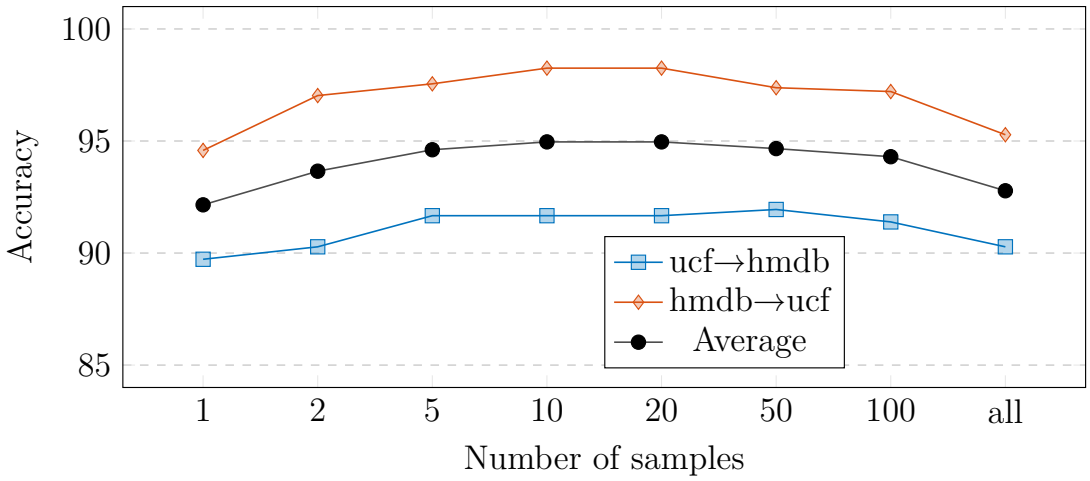


(a) UCF-HMDB$_{full}$

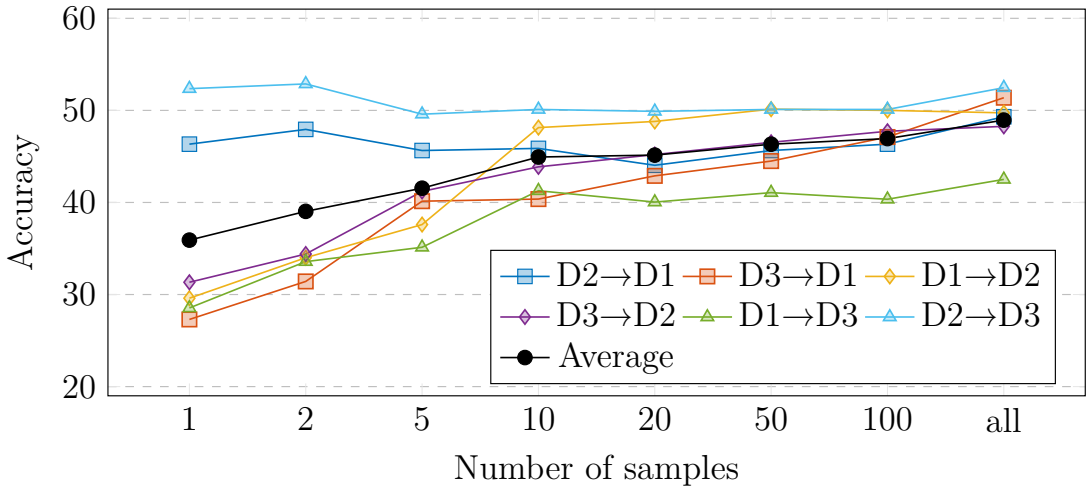


(b) EPIC-Kitchens-55

Figure 5.6: Ablation over Feature Selection This figure shows the effect of selecting the features of different clips to represent a sample when computing the centroid of all target train sample features of each class. Observations show, that consistently picking the same features, such as from the first clip or middle clip gives good results, while picking a clip based on prediction entropy or confidence reduces the accuracy of the resulting centroids. The best features, however, are the average features over all five clips.



(a) UCF-HMDB$_{full}$



(b) EPIC-Kitchens-55

Figure 5.7: Ablation over the Number of Samples This figure shows the impact of reducing the set features to compute the centroids. For these experiments the n features with the lowest prediction entropy were chosen for each class. While UCF-HMDB$_{full}$ benefits from filtering features, EPIC-Kitchens-55 steadily increases performance with each sample added on all shifts.

pseudo-labels, we attempt to select the clip with the best features to represent the video.

Figure 5.6 shows that a variety of feature selection methods provide good results, however, the best accuracy is generally achieved by the mean feature of all the clips. Interestingly, selecting the features with the lowest prediction entropy or highest confidence leads to lower accuracy, indicating that features with these characteristics are not good representations of the class overall.

5.5.2 Increasing Centroid Robustness by Filtering Samples

Similar to T3A [105], we reduce the support set of each centroid by selecting the m features with the lowest prediction entropy. Our experiments in figure 5.7 show that some datasets like UCF-HMDB$_{full}$ benefit from this, while the more difficult EPIC-Kitchens-55 does not. Note how the results on EPIC-Kitchens-55 improve with each additional sample. This may indicate that the amount of available data is a major limitation and that a larger dataset would allow us to achieve better results.

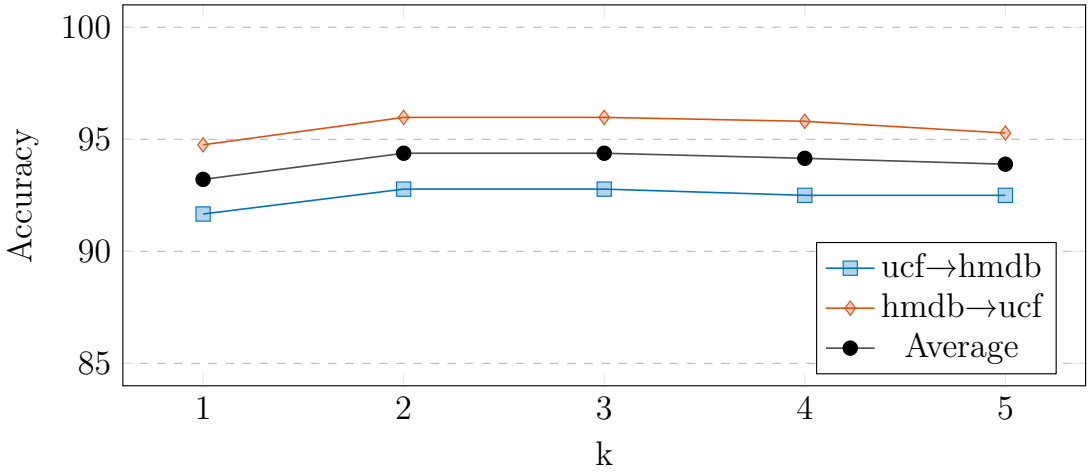
Furthermore, we observe that while confidence is a better indicator of the correctness of a prediction, entropy is better to determine the usefulness of a feature for the centroids. Thus it seems that misclassified samples in a centroid's support set do not directly harm performance, but may actually increase the robustness of the centroids by reducing the negative impact of outliers.

5.5.3 Can centroids be improved iteratively?

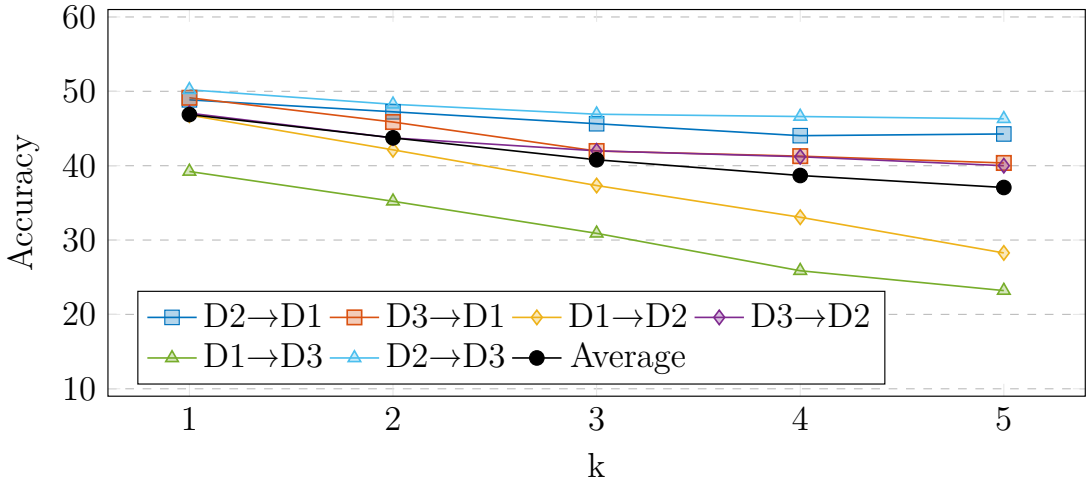
Previous works on SFDA that rely on centroid-based pseudo labels often propose to update the centroids iteratively, similar to k-means. In this case, the initial centroid assignments are determined by the source classifier, while subsequent iterations use the predictions by the centroid-based classifier itself [97, 98]. Figure 5.8 shows the impact the number of k-means iterations has on the final accuracy. On UCF-HMDB$_{full}$ updating the centroids seems to work well, with performance peaking after 2 iterations. However, for the more difficult EPIC-Kitchens-55, the performance decreases with each iteration. This could indicate that the feature space is more complex, has non-linear clusters, or that feature clusters are overlapping.

5.5.4 How do we measure similarity between features?

Finally, we investigate a selection of similarity measures on the feature space, specifically euclidean and angular distance, cosine similarity, and the dot product. All of these measure similarity or distance in euclidean space, and with the exception of the dot-product, all achieve similarly good results.

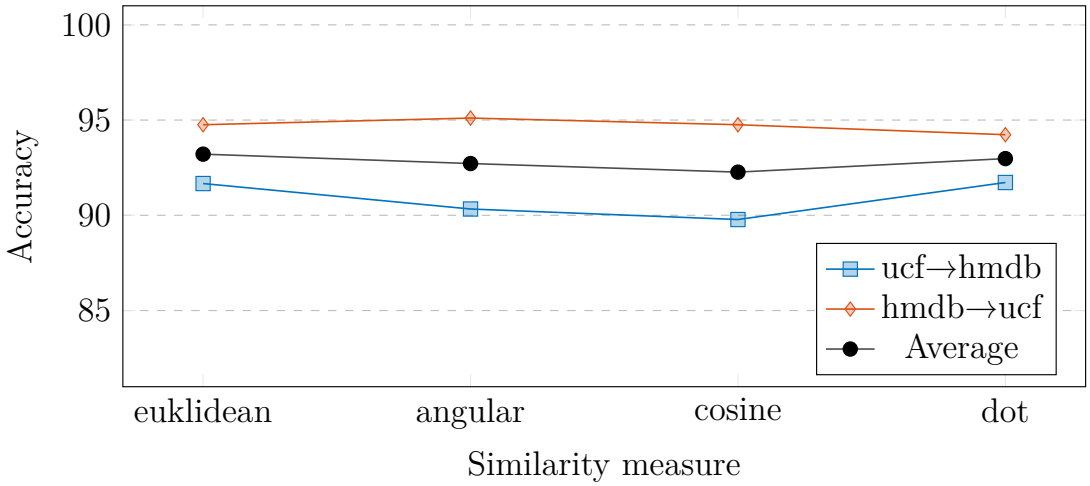


(a) UCF-HMDB$_{full}$

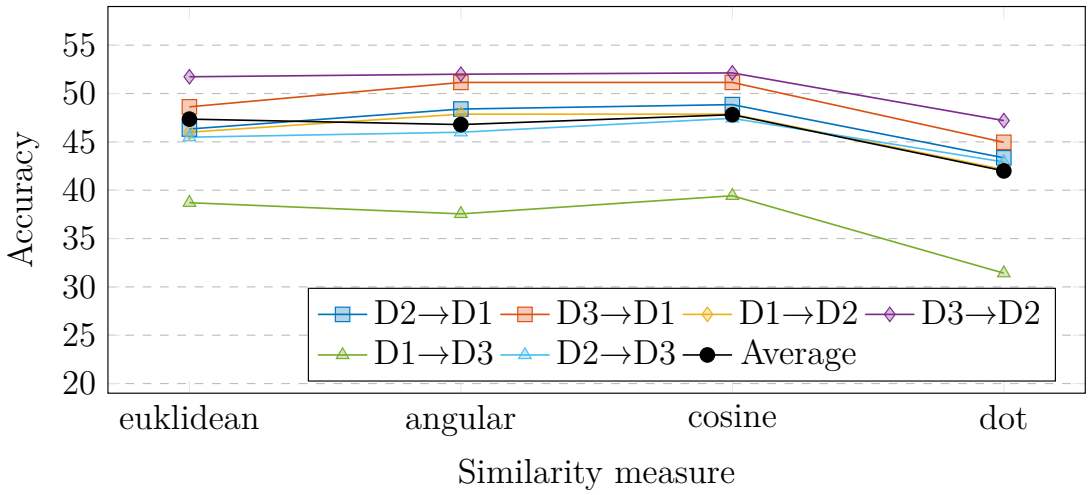


(b) EPIC-Kitchens-55

Figure 5.8: Ablation over the Number of k-means Iterations This figure shows the impact of computing centroids in an iterative fashion similar to k-means. On UCF-HMDB$_{full}$ we observe some improvements, when updating the centroids 2 or 3 times, while on EPIC-Kitchens-55 performance continuously degrades with each iteration.



(a) UCF-HMDB$_{full}$



(b) EPIC-Kitchens-55

Figure 5.9: Ablation over the Similarity Measure in the Feature Space This figure shows how our method performs with different similarity measure between sample features and centroids. Across all datasets the euclidean distance appears to be a solid choice, with angular distance and cosine achieving very similar results. While using the dot product works equally well on UCF-HMDB$_{full}$, a major performance drop can be observed on all shifts in the EPIC-Kitchens-55 dataset.

Chapter 6

Conclusion

This work proposes a novel source-free domain adaptation method, which replaces the linear classification layer with a centroid-based classifier. We show that our method improves cross-domain performance without modifying the network parameters themselves, making it more efficient than traditional SGD-based adaptation methods. Our experiments show, that we are able to beat all previous SFDA methods on two of the most important datasets for domain adaptation in action recognition.

A central finding of this work is that much of the domain bias in neural networks stems from the classification layer that is specific to each dataset, while the feature extractor is more robust thanks to pretraining on large datasets. We exploit this by replacing the linear classifier with a centroid-based classifier, which generalizes butter as it has no trainable parameters, is directly data-dependent, and does not explicitly encode any class bias. Furthermore, it requires no gradient-based adaptation of the network parameters themselves, making it computationally more efficient than previous SFDA methods. Finally, our method integrates well with previous centroid-based SFDA techniques that adapt the feature extractor, allowing for further performance enhancements.

One of the main limitations of our method is its restriction to the classification task, which is shared with many other SFDA methods. However, we see much promise in exploiting the feature space of pre-trained general feature extractors and encourage further investigation in this direction.

Bibliography

[1] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le, "Rethinking pre-training and self-training," *Advances in neural information processing systems*, vol. 33, pp. 3833–3845, 2020.

[2] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *CVPR 2011*, pp. 1521–1528, IEEE, 2011.

[3] Y. Shi, X. Ying, and J. Yang, "Deep unsupervised domain adaptation with time series sensor data: A survey," *Sensors*, vol. 22, no. 15, p. 5507, 2022.

[4] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[5] B. Farley and W. d. Clark, "Simulation of self-organizing systems by digital computer," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 76–84, 1954.

[6] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[8] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[9] D. S. Broomhead and D. Lowe, "Radial basis functions, multi-variable functional interpolation and adaptive networks," tech. rep., Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[10] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, PMLR, 2013.

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

[12] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *Information*, vol. 11, no. 2, p. 125, 2020.

[13] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[14] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International conference on machine learning*, pp. 1058–1066, PMLR, 2013.

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[16] 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*, pp. 770–778, 2016.

[17] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[18] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[19] Y. Wu and K. He, "Group normalization," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

[20] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[21] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, p. 574, 1959.

[22] K. Fukushima, "Neocognitron," *Scholarpedia*, vol. 2, no. 1, p. 1717, 2007.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[28] A. Karpathy, "The unreasonable effectiveness of recurrent neural networks,"

May 2015.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[30] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "Scaling egocentric vision: The epic-kitchens dataset," in *European Conference on Computer Vision (ECCV)*, 2018.

[31] K. Min and J. J. Corso, "Integrating human gaze into attention for egocentric activity recognition," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1069–1078, 2021.

[32] Z. Zhang, D. Crandall, M. Proulx, S. Talathi, and A. Sharma, "Can gaze inform egocentric action recognition?," in *2022 Symposium on Eye Tracking Research and Applications*, pp. 1–7, 2022.

[33] A. Bandini and J. Zariffa, "Analysis of the hands in egocentric vision: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[34] D. Damen, H. Doughty, G. M. Farinella, A. Furnari, J. Ma, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100," *International Journal of Computer Vision (IJCV)*, vol. 130, p. 33–55, 2022.

[35] C.-F. R. Chen, R. Panda, K. Ramakrishnan, R. Feris, J. Cohn, A. Oliva, and Q. Fan, "Deep analysis of cnn-based spatio-temporal representations for action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6165–6175, 2021.

[36] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks for action recognition in videos," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 11, pp. 2740–2755, 2018.

[37] P. Chen and A. J. Ma, "Source-free temporal attentive domain adaptation for video action recognition," in *Proceedings of the 2022 International Conference on Multimedia Retrieval*, pp. 489–497, 2022.

[38] 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*, pp. 6299–6308, 2017.

[39] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[40] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4694–4702, 2015.

[41] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634, 2015.

[42] P. Wei, L. Kong, X. Qu, X. Yin, Z. Xu, J. Jiang, and Z. Ma, "Unsupervised video domain adaptation: A disentanglement perspective," *arXiv preprint arXiv:2208.07365*, 2022.

[43] P. Wei, L. Kong, X. Qu, X. Yin, Z. Xu, J. Jiang, and Z. Ma, "Unsupervised video domain adaptation: A disentanglement perspective," *arXiv preprint arXiv:2208.07365*, 2022.

[44] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, "Aˆ 2-nets: Double attention networks," *Advances in neural information processing systems*, vol. 31, 2018.

[45] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7794–7803, 2018.

[46] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?," in *ICML*, vol. 2, p. 4, 2021.

[47] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6836–6846, 2021.

[48] J. Chen and C. M. Ho, "Mm-vit: Multi-modal video transformer for compressed video action recognition," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1910–1921, 2022.

[49] B. Zhou, A. Andonian, A. Oliva, and A. Torralba, "Temporal relational reasoning in videos," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 803–818, 2018.

[50] J. Lin, C. Gan, and S. Han, "Tsm: Temporal shift module for efficient video understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7083–7093, 2019.

[51] Q. Fan, C.-F. R. Chen, H. Kuehne, M. Pistoia, and D. Cox, "More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[52] A. Sahoo, R. Shah, R. Panda, K. Saenko, and A. Das, "Contrast and mix: Temporal contrastive video domain adaptation with background mixing," *Advances in Neural Information Processing Systems*, vol. 34, pp. 23386–23400, 2021.

[53] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part VI 11*, pp. 140–153, Springer, 2010.

[54] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[55] 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*, pp. 4489–4497, 2015.

[56] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6202–6211, 2019.

[57] M. Planamente, C. Plizzari, E. Alberti, and B. Caputo, "Domain generalization through audio-visual relative norm alignment in first person action recognition," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1807–1818, 2022.

[58] W. Wang, D. Tran, and M. Feiszli, "What makes training multi-modal classification networks hard?," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12695–12705, 2020.

[59] J. Munro and D. Damen, "Multi-modal domain adaptation for fine-grained action recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 122–132, 2020.

[60] D. Kim, Y.-H. Tsai, B. Zhuang, X. Yu, S. Sclaroff, K. Saenko, and M. Chandraker, "Learning cross-modal contrastive features for video domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13618–13627, 2021.

[61] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen, "Epic-fusion: Audio-visual temporal binding for egocentric action recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5492–5501, 2019.

[62] L. Yang, Y. Huang, Y. Sugano, and Y. Sato, "Interact before align: Leveraging cross-modal knowledge for domain adaptive action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14722–14732, 2022.

[63] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l 1 optical flow," in *Pattern Recognition: 29th DAGM Symposium, Heidelberg, Germany, September 12-14, 2007. Proceedings 29*, pp. 214–223, Springer, 2007.

[64] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8934–8943, 2018.

[65] "Optical flow," 2017.

[66] A. Cartas, J. Luque, P. Radeva, C. Segura, and M. Dimiccoli, "How much does audio matter to recognize egocentric object interactions?," *arXiv preprint arXiv:1906.00634*, 2019.

[67] A. Cartas, J. Luque, P. Radeva, C. Segura, and M. Dimiccoli, "Seeing and hearing egocentric actions: How much can we learn?," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[68] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, "Cifar10-dvs: an event-stream dataset for object classification," *Frontiers in neuroscience*, vol. 11, p. 309, 2017.

[69] C. Plizzari, M. Planamente, G. Goletto, M. Cannici, E. Gusso, M. Matteucci, and B. Caputo, "E2 (go) motion: Motion augmented event stream for egocentric action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19935–19947, 2022.

[70] H. Liu, X. Liu, Q. Kong, W. Wang, and M. D. Plumbley, "Learning the spectrogram temporal resolution for audio classification," *arXiv preprint arXiv:2210.01719*, 2022.

[71] G. Jiang, X. Jiang, Z. Fang, and S. Chen, "An efficient attention module for 3d convolutional neural networks in action recognition," *Applied Intelligence*, pp. 1–15, 2021.

[72] J. Carreira, E. Noland, C. Hillier, and A. Zisserman, "A short note on the kinetics-700 human action dataset," *arXiv preprint arXiv:1907.06987*, 2019.

[73] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," in *2011 International conference on computer vision*, pp. 2556–2563, IEEE, 2011.

[74] 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.

[75] W. Sultani and I. Saleemi, "Human action recognition across datasets by foreground-weighted histogram decomposition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 764–771, 2014.

[76] M.-H. Chen, Z. Kira, G. AlRegib, J. Yoo, R. Chen, and J. Zheng, "Temporal attentive alignment for large-scale video domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6321–6330, 2019.

[77] J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. Mit Press, 2008.

[78] J. Wang, C. Lan, C. Liu, Y. Ouyang, T. Qin, W. Lu, Y. Chen, W. Zeng, and P. Yu, "Generalizing to unseen domains: A survey on domain generalization," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[79] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*, pp. 97–105, PMLR, 2015.

[80] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE transactions on neural networks*, vol. 22, no. 2, pp. 199–210, 2010.

[81] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola, "A kernel

method for the two-sample-problem," *Advances in neural information processing systems*, vol. 19, 2006.

[82] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *2011 international conference on computer vision*, pp. 999–1006, IEEE, 2011.

[83] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *2012 IEEE conference on computer vision and pattern recognition*, pp. 2066–2073, IEEE, 2012.

[84] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[85] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[86] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *International conference on machine learning*, pp. 2208–2217, PMLR, 2017.

[87] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3722–3731, 2017.

[88] J. Yang, H. Zou, Y. Zhou, Z. Zeng, and L. Xie, "Mind the discriminability: Asymmetric adversarial domain adaptation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, pp. 589–606, Springer, 2020.

[89] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in neural information processing systems*, vol. 33, pp. 18661–18673, 2020.

[90] J. Choi, G. Sharma, S. Schulter, and J.-B. Huang, "Shuffle and attend: Video domain adaptation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, pp. 678–695, Springer, 2020.

[91] H. Xia, H. Zhao, and Z. Ding, "Adaptive adversarial network for source-free domain adaptation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9010–9019, 2021.

[92] Y. Hou and L. Zheng, "Source free domain adaptation with image translation," *arXiv preprint arXiv:2008.07514*, 2020.

[93] Y. Liu, W. Zhang, and J. Wang, "Source-free domain adaptation for semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1215–1224, 2021.

[94] J. N. Kundu, N. Venkat, R. V. Babu, *et al.*, "Universal source-free domain

adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4544–4553, 2020.

[95] N. Ding, Y. Xu, Y. Tang, C. Xu, Y. Wang, and D. Tao, "Source-free domain adaptation via distribution estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7212–7222, 2022.

[96] S. Yang, Y. Wang, J. Van De Weijer, L. Herranz, and S. Jui, "Generalized source-free domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8978–8987, 2021.

[97] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in *International Conference on Machine Learning*, pp. 6028–6039, PMLR, 2020.

[98] Y. Xu, J. Yang, H. Cao, K. Wu, M. Wu, and Z. Chen, "Source-free video domain adaptation by learning temporal consistency for action recognition," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIV*, pp. 147–164, Springer, 2022.

[99] Y. Huang, X. Yang, J. Zhang, and C. Xu, "Relative alignment network for source-free multimodal video domain adaptation," in *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 1652–1660, 2022.

[100] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[101] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt, "Test-time training with self-supervision for generalization under distribution shifts," in *International conference on machine learning*, pp. 9229–9248, PMLR, 2020.

[102] F. You, J. Li, and Z. Zhao, "Test-time batch statistics calibration for covariate shift," *arXiv preprint arXiv:2110.04065*, 2021.

[103] A. Khurana, S. Paul, P. Rai, S. Biswas, and G. Aggarwal, "Sita: Single image test-time adaptation," *arXiv preprint arXiv:2112.02355*, 2021.

[104] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization," *arXiv preprint arXiv:2006.10726*, 2020.

[105] Y. Iwasawa and Y. Matsuo, "Test-time classifier adjustment module for model-agnostic domain generalization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2427–2440, 2021.

[106] M. Plananamente, C. Plizzari, and B. Caputo, "Test-time adaptation for egocentric action recognition," in *Image Analysis and Processing–ICIAP 2022: 21st International Conference, Lecce, Italy, May 23–27, 2022, Proceedings, Part III*, pp. 206–218, Springer, 2022.

[107] I. Gulrajani and D. Lopez-Paz, "In search of lost domain generalization," *arXiv preprint arXiv:2007.01434*, 2020.

[108] J. Lv, K. Liu, and S. He, "Differentiated learning for multi-modal domain adaptation," in *Proceedings of the 29th ACM International Conference on*

Multimedia, pp. 1322–1330, 2021.

[109] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, pp. 213–226, Springer, 2010.

[110] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[111] A. Ng, "Nuts and bolts of building ai applications using deep learning," *NIPS Keynote Talk*, 2016.

[112] X. Song, S. Zhao, J. Yang, H. Yue, P. Xu, R. Hu, and H. Chai, "Spatio-temporal contrastive domain adaptation for action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9787–9795, 2021.

[113] S. A. Peirone, *EGO-T^ 3: Test Time Training for Egocentric videos.* PhD thesis, Politecnico di Torino, 2022.

[114] Y. Jin, X. Wang, M. Long, and J. Wang, "Minimum class confusion for versatile domain adaptation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pp. 464–480, Springer, 2020.