POLITECNICO DI TORINO

Master degree in Computer Engineering

Master Thesis

Leveraging Relative-Norm-Alignment in higher norm feature space for Cross-Domain First Person Action Recognition

Supervisor
Prof.ssa Barbara Caputo
Dott. Mirco Planamente
Dott.ssa Chiara Plizzari

Laureando
Riccardo Zaccone

July 2022

Abstract

Recently First Person Action Recognition (FPAR) has gained great interest of the researchers' community, mainly due to the increasing spread of wearable devices, the release of large and well-annotated datasets and the huge investments in novel technologies e.g., autonomous drones and robots, self-driving systems. Although egocentric vision rapidly attracted the interest of the research community, this setup presents some important challenges, most notably the ego-motion and the domain shift in feature space. Approaches in the literature often exploit multiple modalities to help mitigating these problems. However, domain shifts affect each modality in a different way, so it is important to develop algorithms that can better leverage the complementarity among modalities to achieve model resilience across domains, allowing the model to better recognize actions under various domain shifts.

The literature proposes *domain adaptation* techniques to address such problems: they consist in methods to mitigate the performance drop that occurs when a model trained on *source* data is used on *target* data, and these data do not follow the same probability distribution. However, such techniques require some knowledge of the target distribution, and often such assumption is too strong. Domain Generalization techniques tackle this kind of scenario but, while for related tasks like image classification several methods exist, the literature in video domain generalization is still scarce.

This work focuses on domain adaptation in first person action recognition, by proposing an approach that takes advantage of the multi-modal nature of the perceptual input. Our approach is designed for a domain generalization scenario, but that can also be used in unsupervised domain adaptation scenario, taking further advantage from the availability of target data. We motivate our approach in light of recent progress in understanding problems and challenges of multi-modal training: in fact, jointly training multi-modal networks is harder than training their uni-modal counterparts, because different modalities separately overfit and generalize at different rates, leading to a sub-optimal joint optimization.

To this extend we study the relative norm alignment (RNA-Net) approach, and propose it as a valuable technique to leverage multi-modal correlations in input streams and as a valid regularizer, further proposing an extension that guides the norm alignment towards higher feature norm regions. Our experiments show that RNA-Net++ is able to effectively enhance the performance of the model it is applied to, by leveraging a *learn to re-balance* task that ensures a consensus mean feature norm among modality streams.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Artificial Intelligence, Machine Learning and Deep Learning have become commonly used buzzwords in the last decade, used by media in a interchangeable way, however without a clear idea of what those words mean. In fact, the concept of a machine exhibiting an "intelligent" behaviour is much older than the modern digital society. The process that lead humankind to the development of modern computers started with the idea that calculus, an activity that is indeed very intelligent since it implies abstracting the reality into numbers and operations between them; however, some aspects of calculus are repetitive, mechanic and in the end not very intelligent: some of the greatest minds of the past understood that, in order to express the full potential of the human mind, this kind of operations have to be automated by using machines. Instruments to assist calculus like the abacus are known from the very ancient times, but the first successful modern attempt to build a calculator is attributed to Blaise Pascal, who in 1642 invented the so-called "Pascalina", able to perform additions through wheels and gears. From that invention, others developed more complex systems: Leibneiz developed a machine able to perform additions, subtractions, multiplications, divisions and square roots, handling numbers up to 16 digits; Jacquard invented the concept of programming looms for the textile industry with punch cards, so the concept of storing instructions, an idea then integrated in the Babbage's "Machina Analitica" (1835). From that moment, the idea of a machine performing calculus was mature at least in theory, and some physical limits were overcome with the introduction of electromechanical components and then with the invention of electronics: from that moment the rest is history.

The idea of an artificial intelligence however is not so new as we may think: as the idea of automatic calculus took place, the idea of automatic reasoning became appealing. Ramon Llull invented a philosophical system known as the "Ars Generalis" (1308), conceived as a type of universal logic to prove the truth of Christian doctrine to interlocutors of all faiths and nationalities: Llull believes that he can solve every problem with mathematical precision: he starts from the assumption that every proposition is reducible to terms and complex terms are reducible to several simple terms or principles. This is believed to have influenced the the latter development of computation and artificial intelligence. Perhaps the most famous work, considered the cornerstone of the discussion on artificial intelligence, is an article from Alan Turing, entitled "Can Machines Think?": it is the article in which the conceptual experiment known as the "Turing test" is described. However, the mainstream culture advertise a different idea of artificial intelligence: there is plenty in the science fiction about humanoid robots taking over the humans, so sometimes it is associated with that particular trait of humans that we generically call intelligence, but that in fact we do not really know what it is. Nevertheless the idea of intelligence applied to machines is rather different: it has to do with exhibited behaviour than actual reasoning. The most straightforward definition would identify something as *intelligent* in relation of a human being, so in terms of fidelity to human performance. In many fields indeed the human performance is the supreme benchmarks with which the state of the art compares itself. Nowadays the use of *intelligence* applied to machines is very practical: can we make a machine able to perform a task such that it is useful for our purposes? Sometimes we need to process some information to obtain value, and often this processing is unfeasible for a human, given the volume of inputs. In practice, modern algorithms look at pattern into the data to extract useful information, to identify relations to exhibit a behaviour evaluated with a optimality metric. Most importantly, machine learning algorithms learn from data: very little prior knowledge about the task is integrated into the algorithm, because the answers we are searching for are hidden in patterns from data.

Machine learning is everywhere: recommendation systems monitors our online activity and purchases to propose products we might be interested in; virtual assistants make our preferences understood and bridge the gap in the human-computer interaction; autonomous driving algorithms promise to free us from the burden of driving; advances in robotics combined with cutting edge algorithms is giving us robots that can autonomously interact with a complex environment and perform complex tasks too dangerous for humans. The spreading of smart devices (like smartphones and wearables), as the integration of IoT devices in our society, increases the interest in

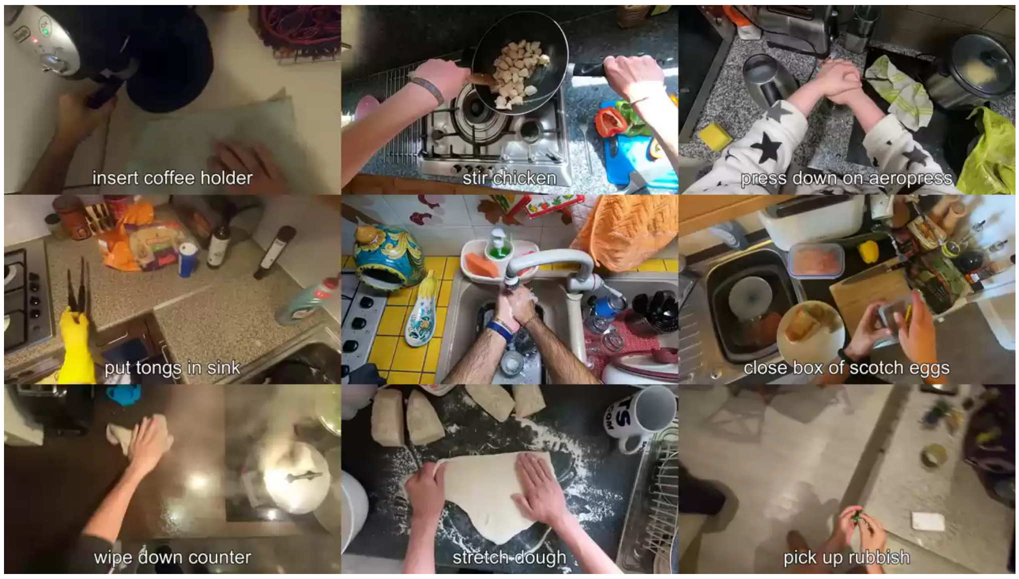


Figure 1.1. Example RGB frames with relative action annotations from several EPIC-Kitchens videos.

machine learning and deep learning: indeed these devices generate a lot of data, and their very purpose is linked to how these data can be exploited, and machine learning is the tool we need. The scenario is very complex and it is rapidly changing: artificial intelligence is a promising technology to advance humanity like never happened before, with all the associated challenges, from both scientific and social point of view. Deep Learning is a subset of machine learning that has neural networks as core components: the term *deep* refers to an architecture made of several layers of networks to extract useful representations from the raw data for the task at hand. In practice if machine learning generally uses shallow architectures in which, for example, the features used by a classier are hand-designed or anyway require a human intervention, deep architectures extract the relevant features from the data and use them as input to the final part of the network.

A field of Artificial Intelligence where Deep Learning architectures have become state of the art is Computer Vision: its goal is to enable computers and machines to gain high-level understanding from images or videos to solve all the various tasks associated with perception. Computer vision already beats humans in some tasks, like image classification. However, perceiving videos has been shown to be much more difficult. Recently, First Person Action Recognition (FPAR) has gained great interest from the researchers' community, mainly due to the increasing spread of wearable devices, the release of large and well-annotated datasets [13] and the huge investments in novel technologies, e.g., autonomous drones and robots, self-driving systems. The most notable difference with general action recognition is that visual information is collected from the human perspective, and the analysis of such egocentric data allows to study human behaviour in a more direct manner. Although egocentric vision has rapidly attracted the interest of the research community, this setup presents some important challenges. For example, ego-motion, arising from the movements of the actor's camera, introduces confusion between the actor's movements and the real action of the subject. Furthermore, the actions' surrounding environment frequently introduces bias into the dataset (environmental bias); as a result of a domain shift in the features space, models frequently rely on environmental characteristics to make predictions. Approaches in the literature often exploit multiple modalities to help mitigate these problems. For example, the auditory channel is not affected by ego-motion, so the prediction related to that modality would not be affected, and so this helps in making the model more robust. Conversely, domain shifts are not all of the same nature, and their impact can vary significantly across modalities; this means that domain shift affects each modality in its own unique way. Moreover, results from biological sciences indicate that humans perceive the world in a multi-modal way: multi-modality has a characteristics called *reentry* [14], the explicit interrelating of multiple simultaneous representations across modalities. In [15] authors provide an example: when a person experiences an apple, and immediately characterizes it as such, the experience is visual, but also invokes the smell of the apple, its taste, its feel, its heft, and a constellation of sensations and movements associated with various actions on the apple. Importantly, these multi-modal experiences are time-locked and correlated. As a consequence, it is of crucial relevance to develop algorithms that can better leverage the complementarity among modalities to achieve model resilience across domains, allowing the model to better recognize actions under various domain shifts.

1.1 Research goals

The literature proposes *domain adaptation* techniques to address such problems: they consist in methods to mitigate the performance drop that occurs when a model trained on *source* data is used on *target* data, and these data do not follow the same probability distribution. However, such techniques require some knowledge of the target distribution, in form of (unlabelled) target data. This requires that target data is available during training, and such assumption is often too strong: indeed the presence of multiple source or target domains further complicates the problem. This motivates to seek for a different approach that can generalize among domains without requiring access to a target distribution, that is in fact unknown. Domain Generalization techniques tackle this kind of scenario but, while for related tasks like image classification several methods exist, the literature in video domain generalization is still scarce. This work focuses on Domain Generalization in first person action recognition, by leveraging the multi-modal nature of the perceptual input.

1.2 Main Contributions

We propose an approach that takes advantage of the multi-modal nature of the perceptual input, obtaining models that can better leverage the complementarity among modalities, and so are more robust with respect to diverse domain shifts. Moreover it can be used also in Unsupervised Domain Adaptation scenario, taking further advantage from the availability of target data. It builds upon a recent approach for audio-visual domain generalization in first person action recognition, that has been proved to be promising [16]. In particular, we extend the approach to account for multi-modal domain generalization, proving its effectiveness in a much more challenging and real-world scenario: both the presence of different environments and the fact that the source and target domains are captured in different temporal moments make it a multi-shift problem ideal for proving the effectiveness of our method. This is the setting of the challenge released with the Epic-Kitchens-100 dataset, namely the Unsupervised Domain Adaptation challenge we face [17].

1.3 Thesis structure

This thesis is organized as follows:

- The first part analyzes the background knowledge of the field of study, namely the theoretical basics of machine learning, the basic concepts of deep learning as well the recent algorithms and techniques, with special attention to the computer vision field (chapter 2). Then we will present the state of the art in first person action recognition (chapter 3), from the formal description of the task, to multi-modal learning, the known architectures and the problems of Unsupervised Domain Adaptation (UDA) and Domain Generalization (DG).

- The second part is about our contributions: in chapter 4 we present the relative norm alignment (RNA-Net) approach, describing an implementation as a loss function. We further propose an extension that guides the norm alignment towards higher feature norm regions; our experiments show that RNA-Net++ is able to effectively enhance the performance of the model it is applied to by leveraging a *learn to re-balance* task. We motivate our approach in light of recent progress in understanding the problems and challenges of multi-modal training as a valuable technique to leverage multi-modal correlations in input streams and as a valid regularizer. In chapter 5 we validate the method through extensive experiments in first person action recognition, using the *EPIC-KITCHEN* dataset, commonly used as benchmark in this field of study. In chapter 6 we additionally show qualitative results analyzing the features in the embedding space resulting from our approach with respect to the baseline and extracting the class activation maps from the classifier, to prove that our method helps at attending the relevant parts of the input video.

- The last part describes the conclusions and the possible future directions of this work.

Part I

First part: Background

Chapter 2

Machine Learning and Deep Learning

This chapter and its subsections provide an overview of deep learning in general, starting from an historical perspective to understand why the development of methods and algorithms have lead us to the technology used nowadays. This first part is covered in Section 2.1, then the following chapters will focus on the concepts most relevant to this work. Then the next chapters will cover the neural networks, with specific focus on the the kind of ones used in this work. In Section 2.2, starting from the perceptron algorithm, the concept of feedforward neural network and the learning algorithm of backpropagation will be explained, following the demonstration in [18]. Starting from the simple neural network, in sections 2.3 and 2.4 the motivations behind the development of more advanced architectures will be examined, so the technologies of convolutional neural networks and residual neural networks. Finally, since analyzing videos involves considering properly the temporal dimension or, more in general, taking into account not only the spatial correlation of pixel in a image but also the correlation of different images, Section 2.5, starting from the recurrent neural networks, will introduce the cutting-edge Transformer architecture, that is emerging as promising approach in a variety of deep learning tasks.

2.1 AI, Machine Learning and Deep Learning

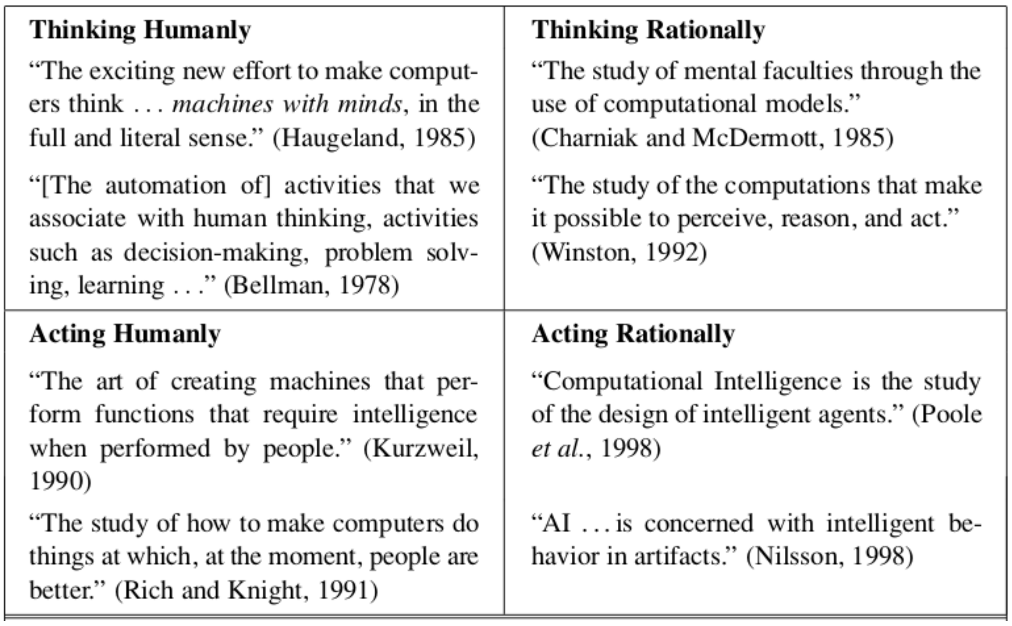
2.1.1 AI

Despite the tremendous speedup of recent progress in the field of artificial intelligence, the idea of making machinery that would allow automatic and intelligent behaviour is as old as the idea of automatic computation. In fact, as soon as humans figured it out how to make a machine to expand the human capabilities to compute, the idea of having machines able to act intelligently had been becoming compelling. However defining what *intelligent* means is as fascinating as intricate: the most straightforward answer would identify something as intelligent in relation of a human being, so in terms of fidelity to human performance. The fact is that, even limiting the concept to humans, giving a definitions of *intelligence* is still an open question without one definitive and precise answer.

Historically, definitions for *intelligent* has been concerned with *thinking* and *acting*, and measuring the success in terms of fidelity to human performance, or against an ideal performance measure, called *rationality*. In Figure 2.1 we see eight definitions of AI, laid out along the different combinations of thinking and acting. History of AI is pretty complex and outside the scope of this work, but there is an important historical passage that is worth to be highlighted: initial works on AI embodied the "physical system hypothesis", which states that “a physical symbol system has the necessary and sufficient means for general intelligent action”. Programs proving mathematical theorems and integrals solvers were successfully developed during this stage of AI. The limitation of such an approach became evident when many predictions about what computers would be able to do did not match the expectations. Motivated by the fact that programs did not know anything about the entities they were manipulating, in the ’70 *expert systems* were born: they leveraged a set of domain-specific rules encoded by humans to solve the task.

2.1.2 Machine Learning

Nowadays the approach taken by machine learning is quite different, more focused on learning from the data itself than by any human-encoded knowledge. In practice modern algorithms look at pattern into the data to extract useful information: just as an example, in the classification task an algorithms tries to identify the relation between the *features*, that is the characteristics the



Thinking Humanly	**Thinking Rationally**
"The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)	"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al.*, 1998) "AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

Figure 2.1. Possible definitions of intelligence, laid out into four categories . Image taken from [1]

sample, and the true label. The knowledge extracted from the data in the training set is the used to infer the true label on new unlabelled data.

More in general, the statistical learning framework of machine learning is based upon assumes the learner has access to a finite training set of instances $\mathcal{S} = \{(x_1, y_1), ..., (x_m, y_m)\}$ in the space $\mathcal{X} \times \mathcal{Y}$ (where $\mathcal{X}$ is the domain set and $\mathcal{Y}$ is the label space) that follow an unknown data distribution $\mathcal{D}$, and the goal of a learning algorithm is to output a predictor $h_\mathcal{S}$ that achieves the lowest possible error with respect to $\mathcal{D}$. The true error of a prediction rule h is defined as:

$$L_\mathcal{D}(h) \overset{\text{def}}{=} \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y] \overset{\text{def}}{=} \mathcal{D}(\{(x,y) : h(x) \neq y\}) \tag{2.1}$$

Because the learner has limited access to $\mathcal{D}$, the true error is not available, so the definition of error being used is called Empirical Risk, defined as:

$$L_\mathcal{S}(h) \overset{\text{def}}{=} \frac{|\{i \in [0,m) : h(x_i) \neq y_i\}|}{m} \tag{2.2}$$

Given this framework the best predictor any learning algorithm can output is the Bayes Optimal Predictor (here in the case $\mathcal{Y} = \{0,1\}$):

$$f_{\mathcal{D}}(x) = \begin{cases} 1, & \text{if } \mathbb{P}[y = 1|x] \geq 1/2 \\ 0, & otherwise \end{cases} \tag{2.3}$$

In general the definition of the risk function depends on the learning task at hand, and it is defined as the expected value of a loss function $l : H \times \mathcal{Z} \rightarrow \mathbb{R}_+$:

$$\begin{aligned} L_{\mathcal{D}}(h) &\stackrel{\text{def}}{=} \mathbb{E}_{z \sim \mathcal{D}}[l(h, z)] \\ L_{\mathcal{S}}(h) &\stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} l(h, z_i) \end{aligned} \tag{2.4}$$

Finally, the notion of PAC learnability gives information about both a lower bound for the lowest achievable error and the sample complexity of the hypothesis class H:

Definition 2.1. A hypothesis class $\mathcal{H}$ is agnostic PAC learnable with respect to a set $\mathcal{Z}$ and a loss function $l : \mathcal{H} \times \mathcal{Z} \rightarrow \mathbb{R}_+$, if there exist a function $m_{\mathcal{H}} : (0,1)^2 \rightarrow \mathbb{N}$ and a learning algorithm with the following property: for every $\epsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over $\mathcal{Z}$, when running the learning algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$, the algorithm returns a hypothesis class h such that, with probability of at least $(1 - \delta)$ over the choice of m training examples:

$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{S}}(h') + \epsilon \tag{2.5}$$

Particularly useful for this work form know on is to notice that any guarantee on the error with respect to an underlying distribution $\mathcal{D}$ for an algorithm that has access only to a sample $\mathcal{S}$ depends on the relationships between $\mathcal{D}$ and $\mathcal{S}$: the common assumption in statistical machine learning is the i.i.d. assumption, by which the examples in $\mathcal{S}$ are independently and identically distributed according to $\mathcal{D}$. This is important for this work because, when training any machine learning algorithm, the error we can expect when new unlabelled instances are fed to the predictor depends on the fact that the training set has been drawn from the same distribution or, equivalently, that the domain we are applying the predictor is similar to the one it has been trained on. Given these assumptions, machine learning has been proved to be the state-of-the-art tool for automatic learning.

2.1.3 The Bias-Complexity tradeoff

An important result in machine learning theory is the so-called No-Free-Lunch theorem, by which is it possible to state that no universal learner exist: more formally, modelling a task as an unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, there not exist a learning algorithm A and a training set size m, such that for every distribution $\mathcal{D}$, if A receives m samples from $\mathcal{S} \sim \mathcal{D}^m$, there is a high probability to obtain a predictor with low $L_{\mathcal{D}}(h)$. This phenomenon is related as *overfitting*, the situation in which the predictor h perfectly fits the training data, with very low or even zero $L_{\mathcal{S}}(h)$, but achieves high error on unseen instances (see figure 2.6(a) for a graphical example). To overcome such a problem, the learner is restricted to choose a predictor from a hypothesis class $\mathcal{H}$ that is believed to contain a predictor that achieves low error: the choice of $\mathcal{H}$ incorporates a prior knowledge of the task, that biases the learning algorithm, and for this reason it is also called *inductive bias.* A fundamental problem is how to properly restrict $\mathcal{H}$ such that it contains a predictor with low risk and it does not lead to overfitting. In fact, restricting the richness of $\mathcal{H}$ could lead to the opposite problem, called *underfitting*, in which the predictor chosen by the learning algorithm lacks the expressive power to effectively model the distribution $\mathcal{D}$, the learning task at hand. Let $h_{\mathcal{S}}$ be an hypothesis obtained using the ERM approach on $\mathcal{H}$, the true error of this predictor can be decomposed into:

$$L_{\mathcal{D}}(h_{\mathcal{S}}) = \epsilon_{app} + \epsilon_{est} \quad \text{where: } \epsilon_{app} = \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h), \quad \epsilon_{est} = L_{\mathcal{D}}(h_{\mathcal{S}}) - \epsilon_{app} \tag{2.6}$$

In this decomposition, the approximation error is the minimum risk achievable by a predictor in the hypothesis class, and it is an indicator of how much inductive bias it is introduced by restricting the hypothesis class. The lower bound for this term is the error of the Bayes predictor, the minimal error possible. The estimation error is the difference between the approximation error and the error achieved by the ERM predictor, and it results from the fact that the empirical risk is only an estimate of the true risk. This term depend on the training size (the larger the training size the lower the error) and on the complexity or richness of $\mathcal{H}$ (the more complex is $\mathcal{H}$, the higher the error). The objective of learning is to minimize the total risk, for this reason this is referred as *bias-complexity tradeoff.*

2.1.4 Loss functions

As previuosly said, a loss function is a function $l : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}_+$, and its choice strongly depend on the task at hand.

0-1 Loss

This loss function is used in binary or multiclass classification problems, and the random variable z ranges over the set of pairs $\mathcal{X} \times \mathcal{Y}$.

$$l_{0-1}(h,(x,y)) \overset{\text{def}}{=} \begin{cases} 0, & \text{if } h(x) = y \\ 1, & \text{if } h(x) \neq y \end{cases} \tag{2.7}$$

Square Loss

This loss is used in regression problems, namely when $\mathcal{Y} = \mathbb{R}$.

$$l_{sq}(h,(x,y)) \overset{\text{def}}{=} (h(x) - y)^2 \tag{2.8}$$

Cross-Entropy Loss

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1, and it is the most widely adopted in multiclass classification problems. Usually the true label is encoded as a one-hot vector containing only one 1, at the index corresponding to the category the instance is belonging to. The output of the classifier is encoded as a probability distribution over the set of classes by applying the softmax function, defined as

$$\begin{aligned} s &: \mathbb{R}^K \to \{o \in \mathbb{R}^K | o_i > 0, \sum_i^K o_i = 1\} \\ s(o)_j &= \frac{\exp o_j}{\sum_{k=1}^K \exp o_k} \quad \text{for } j = 1, ..., K \end{aligned} \tag{2.9}$$

The cross-entropy loss increases as the predicted probability diverges from the actual label. The cross-entropy loss penalizes more severely those predictions that are confident and wrong. Additionally, the error on some classes can be non-uniformly penalized by incorporating a weight vector $w \in \mathbb{R}^C$ in the definition: this is usually used when the dataset is unbalanced.

$$l_{ce}(h,(x,y)) \overset{\text{def}}{=} -\sum_{c=1}^{C} w_c(y_c \log(s(h(x))_c) \tag{2.10}$$

2.1.5 Deep Learning

Deep Learning is a subset of machine learning that has neural networks as core components: the term "deep" refers to an architecture made of several layers of networks to extract useful representations for the task at hand. In practice if machine learning generally uses shallow architectures in which, for example, the features used by a classier are hand-designed or anyway require a human intervention, deep architectures extract the relevant features from the data and use them as input to the final classifier. In such an approach, the features extracted by a layer are based on the ones extracted from the previous one, so from the bottom to the top of the network, the output captures higher-level features of the data, resulting in a hierarchy from which the name *deep learning*. The reasons behind the explosion of deep learning in real world applications in the last ten years lie in the recent progress of GPUs, that enable the massive computation needed to train deep learning models, and the availability of great amount of data to be used for training. This approach has been proved superior in fields like computer vision and natural language processing, in which is particularly difficult and unclear how to extract suitable features from the data for the task at hand. Because neural networks are the base of deep learning algorithms, it is necessary to describe how they are made, starting from the most basic building block, the perceptron.

2.2 From Perceptron to Feedforward neural networks

The perceptron has been introduced in 1958 by the scientist Frank Rosenblatt [19], who defined them as entities consisting of an input layer and an output layer. It is the archetype of an artificial neuron used in neural networks, and it is defined as a unit that characterized by being "on" and "off" based on the input it receives. The concept of an artificial neuron has been strongly inspired by the cognitive science area. From the mathematical point of view, the perceptron is an algorithm for learning the class of halfspaces, that is the

Algorithm 1 Perceptron Learning Algorithm

```
Require: a training set $(x_1, y_1), ..., (x_m, y_m)$, $w^{(1)}$ randomly initialized
  for $t = 1, 2, ...$ do
    if $\exists i \; s.t. \; y_i \langle w^{(t)}, x_i \rangle \leq 0$ then          ▷ Classification error
      $w^{(t+1)} = w^{(t)} + y_i x_i$
    else
      output $w^{(t)}$
    end if
  end for
```

hypothesis class designed for classification problems, that is:

$$\begin{aligned} L_d &= \{h_{\mathbf{w},b} : \mathbf{w} \in \mathbb{R}^d, b \in R\} \\ h_{\mathbf{w},b}(x) &= \langle \mathbf{w}, \mathbf{x} \rangle + b = (\sum_{i=1}^{d} w_i x_i) + b \\ \mathcal{H}_{S_d} &= sign \circ L_d = \{x \rightarrow sign(h_{\mathbf{w},b}(x)) : h_{\mathbf{w},b} \in L_d\} \end{aligned} \tag{2.11}$$

Given this formulation, the goal of an algorithm learning this hypothesis class is to find the set of weights $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that the classification error is the lowest possible: this means that the weight are changed only in there is at least one sample in the training set that is misclassified. If the image of the domain is $\mathcal{Y} = \{-1, +1\}$, then a sample is misclassified if $sign(\langle \mathbf{w}, \mathbf{x} \rangle) \neq y_i$, or equivalently $y_i \langle \mathbf{w}, \mathbf{x} \rangle \leq 0$. The Perceptron is an iterative algorithm that, starting from any vector $\mathbf{w}^{(0)}$, updates each time the weights such that at the next iteration $t+1$ the sample x_i will be correctly classified, repeating this process until convergence.

The perceptron convergence theorem (Block et al., 1962) says that, in the realizable case, the learning algorithm converges with all sample points correctly classified, with a bound on the number of steps needed equal to $(RB)^2$, where $R = max_i(||x_i||)$ and $B = min\{||\mathbf{w}|| : \forall i \in [0, m), \; y_i \langle \mathbf{w}^{(t)}, x_i \rangle \geq 1\}$. Restricting the problem to be realizable means assuming that the samples are linearly separable, meaning that a zero error solution does exist. In real world scenario, this assumption often does not hold, and the reason why the perceptron was revised after long time it was invented is because its expressive power is very limited, and this obstacle were overcome with multilayer networks and by the backpropagation learning algorithm.

A feedforward neural network can be described as a directed acyclic graph (from this the term *feedforward*) whose nodes correspond to neurons and edges correspond to links between them: in such networks the information passes onward from the input x, through intermediate calculations in the so called hidden layers, and finally to the output y. Each neuron is modelled as scalar function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ of the inputs, that is a weighted sum of the outputs of the neurons connected to its incoming edges. The function σ is called *activation function* and it is similar in idea to how a human brain is modelled, by neuron that "fire" when an appropriate input arrives. The activation function is responsible for the nonlinear behaviour of the network predictor, that is the characteristics that gives the expressive power needed to learn complex tasks: in fact is worthy to remind that a combination of linear functions is itself linear, so to model a nonlinear function of inputs a linear model is not sufficient. This makes also the loss function highly non convex, making it hard to train to find the global optima. In practice the key to train such models is still to use heuristics like SGD and the backpropagation algorithm to compute the gradients of the hidden layers from the output with respect to the inputs. A neural network is defined by a DAG $G = (V, E)$ and a weight function $w : E \rightarrow \mathbb{R}$ and the activation function σ, where V is the set of nodes (or neurons) and E is the set of edges. Fixed the architecture of the network as the triplet (V, E, σ), the hypotheses of its hypothesis class are represented by the set of combination of weights over the edges of the network. This means that learning a neural network predictor consists in learning the w parameters that optimize the loss function of the output with respect to the input.

$$\mathcal{H}_{V,E,\sigma} = \{h_{V,E,\sigma,w} : w \text{ is a mapping from } E \text{ to } \mathbb{R} \ \} \tag{2.12}$$

2.2.1 Activation functions

As said before, an activation function is a function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ present in each neuron: it is loosely inspired by how human neurons works, that fires only if their input is appropriate. In the context of neural networks, the output of the neuron is determined calculating $\sigma(a)$ where a is the weighted sum of the inputs to the neuron. Because a linear combination of neurons would lead to a linear function, thus not expanding the expressive capabilities of the network, activation function are designed to be non-linear: in this way the function implemented by the network is not linear anymore, so it can fit complex distributions. There are various activation functions commonly used:

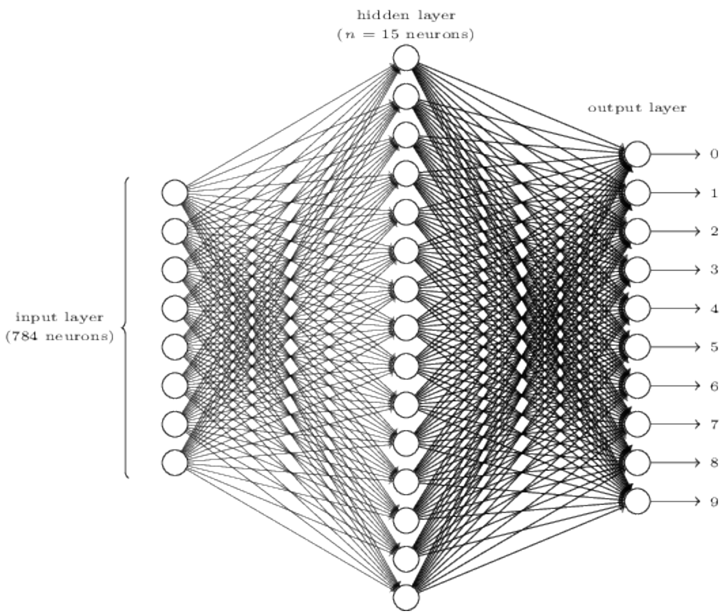


Figure 2.2. Graphical representation of a multilayer perceptron. Image taken from enlight.nyc

Binary Step Function

It is the most simple activation function (see figure 2.3(e)), it makes the neuron a threshold-based classifier, determining whether the neuron fires or not: the linear combination of the inputs should be in that case above the threshold for the neuron to fire. Given a threshold t, the expression of this function is:

$$\sigma_{step}(x) = \begin{cases} 1, & \text{if } x \geq t \\ 0, & \text{if } x < t \end{cases} \tag{2.13}$$

Step function is commonly used in primitive neural networks without hidden layer. In the context of more complex neural networks, the binary step function is usually not used, because its derivative computed during back-propagation is zero, and the weights would not be updated.

Sigmoid

The sigmoid function, from the class of logistic functions, is a non-linear function commonly used in place of binary step function, because its derivative is easy o demonstrate. Whatever the input is, it produces an output in

[0,1].

$$\begin{aligned} \sigma_{sig}(x) &= \frac{1}{1-e^{-x}} \\ \frac{d}{dx}\sigma_{sig}(x) &= \sigma_{sig}(x)(1-\sigma_{sig}(x)) \end{aligned} \tag{2.14}$$

A drawback of this function for the sake of learning is that the graph becomes significantly flatter in other places (see figure 2.3(a)). This implies that for values large in modulus, the gradients will be extremely small: this is undesirable because gradient values approaching zero indicate that the network is not truly learning. Additionally, this function is asymmetrical around zero, so the output of all neurons will have the same sign.

Hyperbolic Tangent

Differently from the sigmoid, the hyperbolic tangent function is symmetric around the 0 and its output ranges in $[-1{,}1]$. Moreover it has a steeper gradient than the *sigmoid* function, while its derivative being as simple as the sigmoid's one. For this reasons, it is generally chosen over the sigmoid (see figure 2.3(b)).

$$\begin{aligned} \sigma_{htan}(x) &= \tanh(x) \\ \frac{d}{dx}\sigma_{htan}(x) &= 1-\tanh(x)^2 \end{aligned} \tag{2.15}$$

ReLU and Leaky RELU

Perhaps the most widely used activation function, the REctified Linear Unit (RELU) is defined as the postive of its argument (see figure 2.3(c)). Used since the '60 in the context of visual feature extraction in hierarchical neural networks, more recent researches show it has strong biological motivations and mathematical justifications. It has been proved itself more efficient in training deep neural networks. Due to the fact that it does not simultaneously stimulate all neurons (sparse activation), it is more efficient to calculate (only comparison, addition and multiplication), and it is scale invariant. As drawbacks, because neurons that are never excited will have zero gradient on their weight and biases, this can result in the death of neurons that are never stimulated (a case of vanishing gradient problem).

$$\sigma_{relu}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{2.16}$$

To mitigate this problem and avoid the fact that it is not differentiable around 0, a linear variant called Leaky RELU can be used: it allows a small, positive gradient when the unit is not active (see figure 2.3(d)).

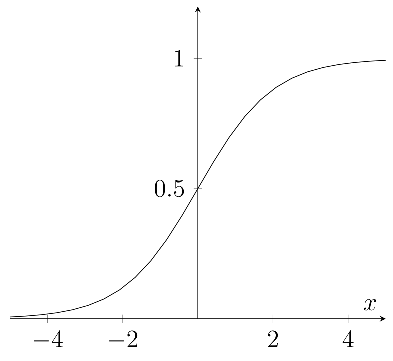
$$\sigma_{leaky-relu}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0.001x, & \text{if } x < 0 \end{cases} \tag{2.17}$$

2.2.2 Optimizing a neural network: GD and SGD

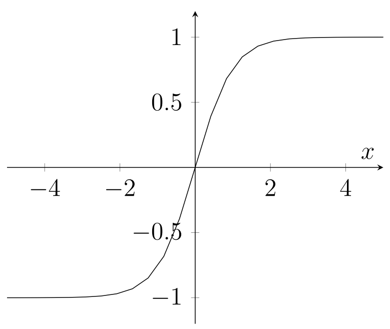
It has been demostrated that implementing the ERM rule with respect to $H_{V,E,sign}$ is an NP hard problem even for small networks; indeed such a problem cannot be circumvented by choosing a different σ or changing the structure of the network. For this reason, even if it cannot guarantee to converge to a global minimum, a widely used heuristics is to use the SGD procedure to learn the weights in conjunction with the backpropagation algorithm. The concepts behind the Gradient Descent procedure is to iteratively improve the solution of, for example, a $\min(\cdot)$ problem by taking a step along the negative of the gradient of the function at the current point. Since the underlying distribution of the training data is not known, it is not possible to calculate the true gradient so, to circumvent this problem, stochastic gradient descent allows the optimization procedure to take a step along a random direction, as long its expected value is the negative of the gradient. More formally, the gradient of a differentiable function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ at w is the vector of partial derivatives of f, namely $\nabla f(w) = (\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}_1}, ..., \frac{\partial f(\mathbf{w})}{\partial \mathbf{w}_d})$, and gradient descent iteratively updates the weights following the negative direction of the gradient, leading to the following update rule:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla f(\mathbf{w}^{(t)}) \tag{2.18}$$

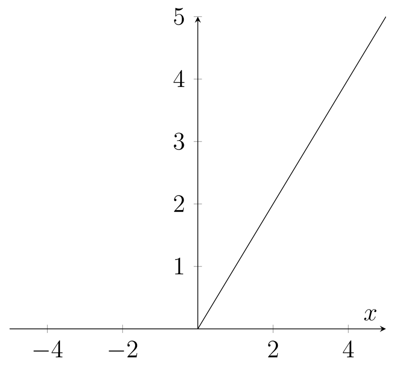
The parameter $\eta > 0$ is called *learning rate* and determines the step size toward the minimum: it is a common hyperparameter of neural networks that can be constant or can change according a scheduling strategy during the training procedure. When the problem is convex, the Gradient Descent procedure guarantees to find the global optimum, while when the problem is not convex it can find any of the local optima, for this reason it is important



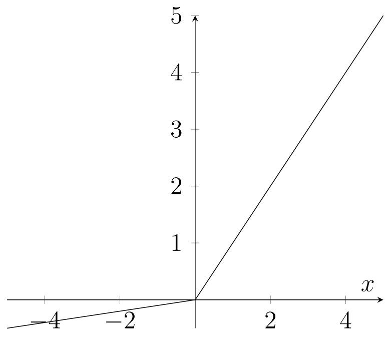
(a) Sigmoid



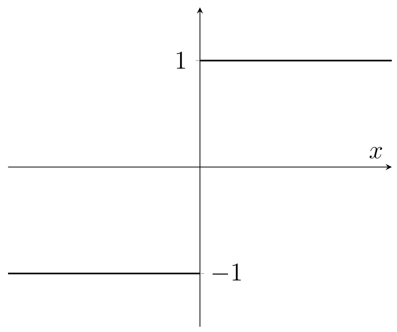
(b) Hyperbolic Tangent



(c) ReLU



(d) Leaky ReLU



(e) Binary step

Figure 2.3. Commonly used activation functions in neural networks

to correctly configure the learning rate, and there is not a standard optimal value for it. Gradient descent applies to differentiable convex functions: indeed another way to motivate gradient descent is to rely on Taylor's approximation of the function being learnt. In fact, the approximation of a function f around a point $\mathbf{w}$ can be expressed as $f(\mathbf{u}) \approx f(\mathbf{u}) + \langle \mathbf{u} - \mathbf{w}, \nabla f(\mathbf{w}) \rangle$, and

Algorithm 2 Stochastic Gradient Descent (SGD) for minimizing $f(\mathbf{w})$

```
Require: scalar $\eta > 0$, integer $T > 0$
  $\mathbf{w}^{(0)} = 0$
  for $t = 1, 2, ..., T$ do
      choose $\mathbf{v}_t$ at random from a distribution such that $\mathbb{E}[\mathbf{v}_t|\mathbf{w}^{(t)}] \in \partial f(\mathbf{w}^{(t)}$
      $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \mathbf{v}_t$
  end for
  output $\bar{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}^{(t)}$
```

in case the function is convex this approximation lower bounds f. Indeed, this is a possible characterization of convexity:

$$f(\mathbf{u}) \approx f(\mathbf{u}) + \langle \mathbf{u} - \mathbf{w}, \nabla f(\mathbf{w}) \rangle \tag{2.19}$$

In this way the problem of minimizing a function become the problem of jointly minimizing the approximation of f at $\mathbf{w}^{(t)}$ and the "goodness" of this approximation, that relies on the distance between $\mathbf{w}$ and $\mathbf{w}^{(t)}$, leading to the following formula:

$$\mathbf{w}^{(t+1)} = \arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w} - \mathbf{w}^{(t)}||^2 + \eta(f(\mathbf{w}^{(t)}) + \langle \mathbf{u} - \mathbf{w}, \nabla f(\mathbf{w}) \rangle) \tag{2.20}$$

The solution of this problem can be obtained by taking the first derivative with respect to $\mathbf{w}^{(t+1)}$ and comparing it to zero, leading to the update rule in equation 2.18. It can be proved that, if the function is convex and ρ-Lipschitz and $\mathbf{w}^* \in \arg\min_{\mathbf{w}:||\mathbf{w}|| \leq B} f(\mathbf{w})$, then the output vector $\bar{\mathbf{w}}$ of the Gradient Descent algorithm is a good approximation of the true minimum point, and the the extend of this approximation is bound to the number of steps T taken:

$$f(\bar{\mathbf{w}}) - f(\mathbf{w}^*) \leq \epsilon \rightarrow T \geq \frac{B^2\rho^2}{\epsilon^2} \tag{2.21}$$

As previously said, the Gradient Descent algorithm requires the loss function to be differentiable, and often it is not the case. In such cases, instead of the true gradient of the function, a *subgradient* $v \in \partial f(\mathbf{w})$ is used, that is a vector that satisfies equation 2.19. This principle is used in stochastic gradient descent, in which the direction of the update of equation 2.18 is allowed to be a random vector whose expected value is a subgradient of the function at the current vector $\mathbf{v}^{(t+1)}$.

In the context of neural networks, the SGD algorithm is used, but with some modifications to account for the non convexity of the learning problem:

1. The vector $\mathbf{w}$ is randomly initialized;
2. Instead of a fixed step size, η_t is variable during the training;
3. The vector in output by the algorithm is the best performing on a validation set;

Most importantly, the subgradient used in algorithm 2 is calculated by the *backpropagation* algorithm.

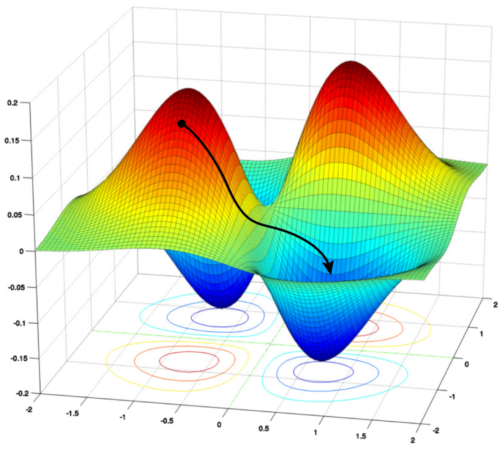


Figure 2.4. Graphical representation of gradient descent of a multidimensional function. Image taken from sciencesprings.wordpress.com

2.2.3 The backpropagation algorithm

The idea of backpropagation came around 1970, but its significance wasn't fully appreciated until David Rumelhart, Geoffrey Hinton, and Ronald Williams published a seminal paper [20] in 1986 when backpropagation was formally introduced as the learning procedure to train neural networks. In its essence, the backpropagation algorithm allows to calculate the gradients of a layer of the network as function of the next layer from bottom to top in a recursive manner, thus requiring only the final output and the current weights of the network. To explain how such computation works, it is necessary to recall the definition of the Jacobian of a function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ at $\mathbf{w} \in \mathbb{R}^n$ to be $J_{\mathbf{w}}(f) \in \mathbb{R}^{m \times n}$ such that $J_{\mathbf{w},(i,j)} = \frac{\partial f(\mathbf{w})}{\partial w_{ij}}$. Moreover, the chain rule allows to compute the jacobian of the composition of two functions $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ and $g : \mathbb{R}^k \mapsto \mathbb{R}^n$ as $J_{\mathbf{w}}(f \circ g) = J_{g(\mathbf{w})}(f) J_{\mathbf{w}}(g) \in \mathbb{R}^{k \times m}$.

For analysis convenience, instead of viewing the parameters of the networks as long row vectors as done previously, it is possible to decompose the network into the set of its layers $V = \uplus_{t=0}^{T} V_t$, where each layer is composed by a set of vertices (neurons) $V_t = \{v_{t,1}, ..., v_{t,k_t}\}\, k_t = |V_t|$, the set of edges $E = \cup_{t=0}^{T-1}(V_t \times V_{t+1})$ and the set of weights associated to each edge between V_t and V_{t+1}, $W_t \in \mathbb{R}^{k_{t+1} \times k_t}$.

Now the problem is how to calculate the (partial) derivative of this weights matrix $\mathbf{W}_{t-1}$ for all the layers between input and output. It is possible to notice that, since when calculating the partial derivative of the weights in $\mathbf{V}_{t-1}$ all other weights are fixed, it follows that the outputs of all the neurons in $\mathbf{V}_{t-1}$ denoted by $\mathbf{o}_{t-1}$ do not depend on the weights in $\mathbf{W}_{t-1}$. Then, the input to the neurons of V_t can be written as $a_t = \mathbf{W}_{t-1}\mathbf{o}_{t-1}$ and $\mathbf{o}_t = \sigma(\mathbf{a}_t)$, where σ is the activation function. Thus the loss, as a function of $\mathbf{W}_{t-1}$, can be written as:

$$g_t(\mathbf{W}_{t-1}) = l_t(\mathbf{o}_t) = l_t(\sigma(\mathbf{a}_t)) = l_t(\sigma(\mathbf{W}_{t-1}\mathbf{o}_{t-1}) \tag{2.22}$$

It is possible to define $\mathbf{w}_{t-1} \in \mathbb{R}^{k_{t-1}\, k_t}$ as the column vector obtained by concatenating the rows of $\mathbf{W}_{t-1}$ and then taking the transpose of the resulting long vector, and $\mathbf{O}_{t-1} \in \mathbb{R}^{k_t \times k_{t-1}\, k_t}$ as:

$$\mathbf{O}_{t-1} \overset{\text{def}}{=} \begin{pmatrix} \mathbf{o}_{t-1}^\top & 0 & \dots & 0 \\ 0 & \mathbf{o}_{t-1}^\top & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{o}_{t-1}^\top \end{pmatrix} \tag{2.23}$$

So $\mathbf{W}_t\mathbf{o}_{t-1} = \mathbf{O}_{t-1}\mathbf{w}_{t-1}$, and equation 2.22 can be rewritten as:

$$g_t(\mathbf{w}_{t-1}) = l_t(\sigma(\mathbf{O}_{t-1}\mathbf{w}_{t-1}) \tag{2.24}$$

Therefore, applying the chain rule, it is possible to write the Jacobian of the weights at layer $t-1$ of the loss function at layer t:

$$\begin{aligned} J_{\mathbf{w}_{t-1}}(g_t) &= J_{\sigma(\mathbf{O}_{t-1}\mathbf{w}_{t-1})}(l_t)\operatorname{diag}(\sigma'(\mathbf{O}_{t-1}\mathbf{w}_{t-1}))\mathbf{O}_{t-1} \\ &= J_{\mathbf{o}_t}(l_t)\operatorname{diag}(\sigma'(\mathbf{a}_t))\mathbf{O}_{t-1} \end{aligned} \tag{2.25}$$

So it is left to calculate the vector $\delta_t = J_{\mathbf{o}_t}(l_t)$ for every t, that is the gradient of l_t at $\mathbf{o}_t$: this can be accomplished by noting that the loss at layer t can be calculated using the loss at layer $t+1$, and that ultimately the loss at the last layer T is straightforward to calculate having the desired output

Algorithm 3 Backpropagation [18]

```
Require: example $(\mathbf{x}, y)$, weight vector $\mathbf{w}$, layered graph $(V, E)$
  denote layers of the graph $\mathbf{V}_0, ..., \mathbf{V}_T$ where $\mathbf{V}_t = \{v_{t,1}, ..., v_{t,k_t}\}$
  define $\mathbf{W}_{t,i,j}$ as the weight of $(v_{t,j}, v_{t+1,i})$, where $\mathbf{W}_{t,i,j} = 0$ if $(v_{t,j}, v_{t+1,i}) \notin E$

  set $\mathbf{o}_0 = \mathbf{x}$
  for $t = 1, ..., T$ do
      for $i = 1, ...k_t$ do
          set $a_{t,i} = \sum_{j=1}^{k_{t-1}} W_{t-1,i,j} o_{t-1,j}$
          set $o_{t,i} = \sigma(a_{t,i})$
      end for
  end for

  set $\delta_T = J_{\mathbf{o}_t}(l_T)$
  for $t = T-1, T-2, ..., 1$ do
      for $i = 1, ...k_t$ do
          $\delta_{t,i} = \sum_{j=1}^{k_{t+1}} W_{t,j,i} \delta_{t+1,j} \sigma'(\mathbf{a}_{t+1,j})$
      end for
  end for
  output : foreach edge $(v_{t,j}, v_{t+1,i}) \in E$ set the partial derivative to $\delta_{t,i} \sigma'(a_{t,i}) o_{t-1,j}$
```

and the result of the previous layers:

$$\begin{aligned} l_t(\mathbf{o}_t) &= l_{t+1}(\sigma \mathbf{W}_t \mathbf{o}_t)) \\ J_{\mathbf{o}_t}(l_t) &= J_{\sigma(\mathbf{W}_t \mathbf{o}_t)}(l_{t+1}) \mathrm{diag}(\sigma'(\mathbf{W}_t \mathbf{o}_t)) \mathbf{W}_t \\ &= J_{\mathbf{o}_{t+1}}(l_{t+1}) \mathrm{diag}(\sigma'(\mathbf{a}_{t+1})) \mathbf{W}_t \\ &= \delta_{t+1} \mathrm{diag}(\sigma'(\mathbf{a}_{t+1})) \mathbf{W}_t \end{aligned} \tag{2.26}$$

So, by calculating $\{\mathbf{a}_t, \mathbf{o}_t\}$ during the forward pass from the bottom to the top of the network, it is possible to calculate δ_t backwards and use them for calculating the partial derivatives according to equation 2.25.

2.2.4 Regularization

An alternative learning rule to minimize the empirical error in formula 2.2 is to jointly minimize the empirical error and a regularization function (*Regularized Loss Minimization*), that is a mapping $R : \mathbb{R}^d \to \mathbb{R}$, that intuitively

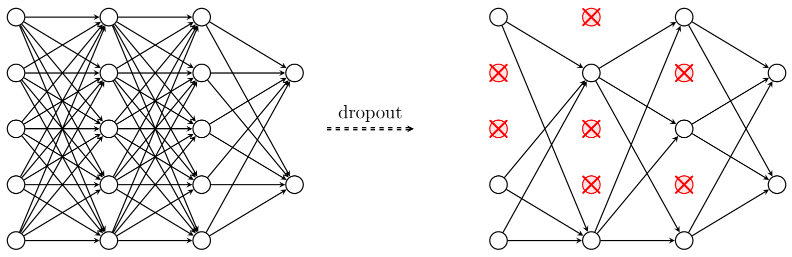


Figure 2.5. Example of dropout effect

measures the complexity of hypotheses, namely:

$$\arg\min_{\mathbf{w}}(L_{\mathcal{S}}(\mathbf{w}) + R(\mathbf{w})) \tag{2.27}$$

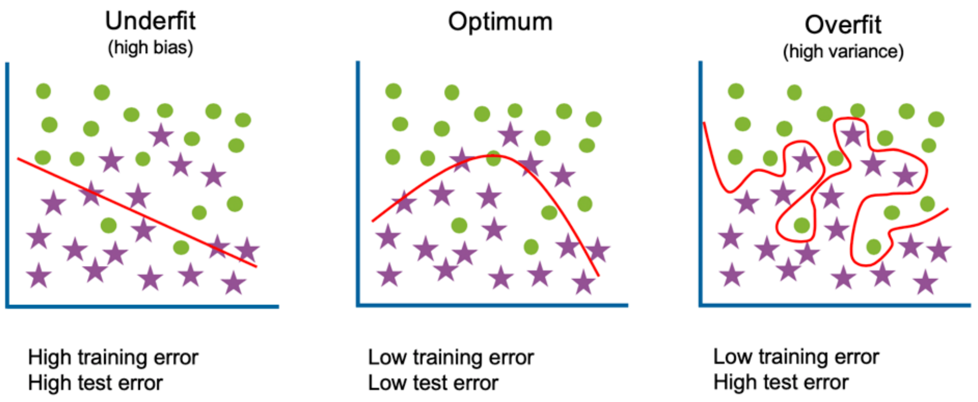
One of the most simple yet widely used and effective class of regularization functions is called Tikhonov regularization, and it is defined by $R(\mathbf{w}) = \lambda||\mathbf{w}||^2$. The parameter λ is usually called *weight decay*, and it is a common hyperparameter to be tuned.

Another well-known strategy in neural networks is dropout, which consists in "turning off" some neurons, so that they not influence the result during the forward and backward pass. More precisely, during each training stage, a random set of nodes is removed from the network with probability $1 - p$, by cutting down the the incoming and outgoing edges of those nodes.

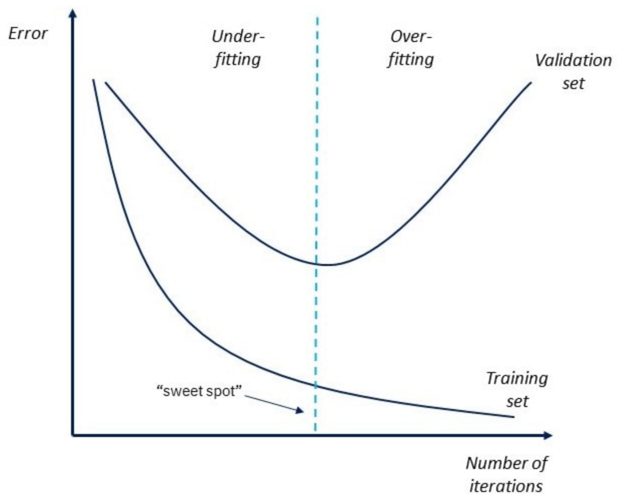
2.2.5 Batch Normalization

In 2015 two researchers from Google, Sergey Ioffe and Christian Szegedy, published a paper that highlighed a complication that occurs in training deep neural networks, namely the fact that the distributions of input to each layer are affected by the parameters of all preceding layers – so that small changes to the network parameters amplify as the network becomes deeper. Because of this change of distribution of layers' input, the layers need to continuously adapt to the new distribution. The change in the input distributions to a learning system is called *covariate shift*, and because this consideration is made for each part of the network, the researches referred to this phenomenon as *internal covariate shift*.

The proposed approach aims at eliminating it and, by doing so, accelerating the training: it consists in a normalization step that fixes the means and variances of layer inputs. Based on the fact that the network converges



(a) Decision boundaries of an underfitted, fitted and overfitted model



(b) Trend of training and validation error as function of training iterations

Figure 2.6. Representation of how underfitting and overfitting affect the model. Images taken from ibm.com

faster if its input are whitened, so linearly transformed to have zero means and unit variances, and decorrelated, and on the observation that each layer observes the inputs produced by the layers below, whitening the inputs to each layer would remove the drawbacks of internal covariate shift.

The approach taken consists in integrating in the learning process the fact that input are being normalized, and ensuring that, for any parameter values, the network always produces activations with the desired distribution so that the gradient of the loss with respect to the model parameters to account for the normalization, and for its dependence on the model parameters. Within this framework, the normalization is seen as a transformation of the a layer input $\mathbf{x}$ relative to the set of inputs X over the training data set $\hat{\mathbf{x}} = Norm(x, X)$. This would require an expensive calculation of $Cov[\mathbf{x}] = \mathbb{E}_{\mathbf{x}\in\mathbb{X}}[\mathbf{x}\mathbf{x}^\top] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{x}]^\top$, its inverse square-root and the derivatives for backpropagation.

As an alternative, it has been proposed to adopt two simplifications:

- instead of whitening the features in layer inputs and outputs jointly, to normalize each scalar feature independently;
- since the above formulation requiring the entire dataset to normalize activations is impractical in the context of stochastic optimization, the mean and variance values used for normalization are the estimates computer over a mini-batch.

Such an approach is hence called *Batch Normalization.* For a d-dimensional input $\mathbf{x} = (x^{(1)}, ..., x^{(d)})$, each dimension is normalized as:

$$\hat{\mathbf{x}} = \frac{\mathbf{x}^{(k)} - \mathbb{E}[\mathbf{x}^{(k)}]}{\sqrt{Var[\mathbf{x}^{(k)}]}} \tag{2.28}$$

Moreover, because only introducing this normalization would possibly change the expressive power of the network, since it could constrain the inputs to the same region of the loss function, additional parameters $\gamma^{(k)}$ and $\beta^{(k)}$ are learnt to scale and shift the normalized value:

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)} \tag{2.29}$$

During inference, because it is not desiderable to have the normalization of activations to depend on mini batch, i.e. the output must depend deterministically only on the input, the statistics are computed over the population rather than over a mini batch. In practice, this is usually implemented by the model keeping moving averages of minibatch means and variances, and use them during inference in place of the mini batch statistics. Since the means and variances are fixed during inference, the normalization is simply a linear transform applied to each activation. In [21] authors have discussed the use of moving average also during training, arguing that their use in place of actual mini batch statistics would cause the gradient optimization and the normalization to counteract each other.

Algorithm 4 Batch Normalization

```
Require: example $x$ over a mini-batch $B = \{x_1, ..., x_m\}$
  parameters to be learned: $\gamma, \beta$

  $\mu_\beta = \frac{1}{m}\sum_{i=1}^{m} x_i$

  $\sigma_\beta^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\beta)^2$

  $\hat{x_i} = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}}$

  output :
  $y_i = \gamma\hat{x_i} + \beta \equiv BN_{\gamma,\beta}(x_i)$
```

Effects of Batch Normalization

- **Batch Normalization allows to use much higher learning rates without the risk of divergence:** by normalizing activations throughout the network, it prevents small changes to the parameters from amplifying into larger and suboptimal changes in activations in gradients. Large learning rates may also increase the scale of layer parameters, which then amplify the gradient during backpropagation, leading to model explosion. In this case, using batch normalization the backpropagation trought a layer is unaffected by the scale of its parameters

- **Batch Normalization acts as a regularizer, reducing the need for Dropout:** it has been experimentally noted that in a batch-normalized network the need of dropout to reduce overfitting is reduced, due to the fact that in training the output of an example depends also on the mini batch.

- **Batch Normalization makes it possible to use saturating nonlinearities by preventing the network from getting stuck in the saturated regime of nonlinearity:** this saturation problem and the resulting vanishing gradients are addressed by means of non saturating nonlinearities such as ReLU and small learning rates. Batch Normalization ensures that the distribution of nonlinearity inputs to a layer remains more stable as the network trains, hence accelerating the training.

Variations: Batch Renormalization

A more recent paper suggest a variation of the batch normalization particularly effective when dealing with a small or non iid minibatches: in fact an important assumption when moving batch normalization in the context of stochastic optimization is that each mini batch produces estimates of the mean and variance of each activation. This assumption may not hold with small or non iid mini batches: in fact, the estimates become less accurate, affecting the resulting model. Moreover, as each example is used to compute the variance used in its own normalization, the normalization operation is less well approximated by an affine transform, which is what is used in inference. When dealing with non iid mini batches, because with batch norm the normalization of each example depends on examples in its mini batch, the examples interact at every layer, which can cause the model to overfit to the specific distribution of mini batches, while without it the loss computed for the mini batch decouples over the examples.

To ensure that the activations computed in the forward pass of the training step depend only on a single example and are identical to the activations computed in inference, [22] proposed an extension to Batch Normalization called *Batch Renormalization*: the aim is to unify the normalization procedure between training and inference while retaining the benefits of batch norm. It consists in a correction to the normalization, by noting that normalizing a particular node x using either the minibatch statistics or their moving averages, the result of these transformations are related by an affine transformation:

$$\frac{x_i - \mu}{\sigma} = \frac{x_i - \mu_\beta}{\sigma_\beta} \cdot r + d, \text{where } r = \frac{\sigma_\beta}{\sigma},\ d = \frac{\mu_\beta - \mu}{\sigma} \tag{2.30}$$

Taking $\sigma = \mathbb{E}[\sigma_\beta]$ and $\mu = \mathbb{E}[\mu_\beta]$ is falling back to batch norm, which sets $r = 1$, $d = 0$. The authors in [22] propose to treat parameters r and d as constants for the purpose of gradient computation, even if they were calculated from the mini batch itself. In this approach, the fixed r and d correct for the fact that the mini batch statistics differ from the population ones. This allows the above layers to observe the "correct" activations – namely, the ones that would be generated by the inference model. In practice usually the model is trained for a certain number of iterations with batchnorm alone, then the correction is introduced by imposing $r = 1$ and $d = 0$, and then relaxing the constraint gradually.

Batch Normalization and Internal Covariate Shift

Even though the reduction of the internal covariate shift has been widely accepted as explanation of Batch Normalization indisputable success, a more recent work [23] has rediscovered the underpinnings principles of how batch normalization helps optimization. Researchers have discovered that the link between the gain of batch normalization and the internal covariate shift is at most tenuous, and that the true reason of its success relies on how it affects the loss function. More specifically, batch normalization has been proved to make the landscape of underlying optimization problem significantly more smooth, that ensures that the gradients are more predictive and thus allows for use of larger range of learning rates and faster network convergence. More formally, researchers demonstrated that batch norm improves the Lipschitzness and Smoothness of the loss:

- (The effect of BatchNorm on the Lipschitzness of the loss). For a BatchNorm network with loss $\hat{L}$ and an identical non-BN network with (identical) loss L, it holds:

$$||\nabla_{y_j}\hat{L}||^2 \leq \frac{\gamma^2}{\sigma_j^2}\left(||\nabla_{y_j}L||^2 - \frac{1}{m}\langle 1, \nabla_{y_j}L\rangle^2 - \frac{1}{m}\langle \nabla_{y_j}L, \hat{y_j}\rangle^2\right) \tag{2.31}$$

- (The effect of BN to smoothness). Let $\hat{g_j} = \nabla_{y_j}L$ and $H_jj = \frac{\partial L}{\partial y_j \partial y_j}$ be the gradient and Hessian of the loss with respect to the layer outputs respectively. Then:

$$\left(\nabla_{y_j}\hat{L}\right)^\top \frac{\partial \hat{L}}{\partial y_j \partial y_j}\left(\nabla_{y_j}\hat{L}\right) \leq \frac{\gamma^2}{\sigma^2}\left(\frac{\partial \hat{L}}{\partial y_j \partial y_j}\right)^\top H_{jj}\left(\frac{\partial \hat{L}}{\partial y_j}\right) - \frac{\gamma}{m\sigma^2}\langle \hat{g_j}, \hat{y_j}\rangle \left\|\frac{\partial \hat{L}}{\partial y_j}\right\|^2 \tag{2.32}$$

If we also have that the H_{jj} preserves the relative norms of $\hat{g_j}$ and $\nabla_{y_j}\hat{L}$:

$$\left(\nabla_{y_j}\hat{L}\right)^\top \frac{\partial \hat{L}}{\partial y_j \partial y_j}\left(\nabla_{y_j}\hat{L}\right) \leq \frac{\gamma^2}{\sigma^2}\left(\hat{g_j}^\top H_{jj}\hat{g_j} - \frac{1}{m\gamma}\langle \hat{g_j}, \hat{y_j}\rangle \left\|\frac{\partial \hat{L}}{\partial y_j}\right\|^2\right) \tag{2.33}$$

As long as the Hessian and the term $\langle \hat{g_j}, \hat{y_j}\rangle$ are non-negative, the last theorem implies more predictive gradients than those of the standard network.

2.3 Convolution Neural Networks

Convolutional Neural Networks, or CNNs, are a subset of neural networks used to analyze data with a known, grid-like topology, as stated by [24]. In particular, CNNs have been proven effective to analyze visual imagery. In fact the way a computer system can work with images is by encoding it as the sequence of its pixels, where each pixel is commonly encoded as a integer number (greyscale images) or a tuple of three integers representing each the primary color component component (colored images in RGB). So the encoding of an image in usually a matrix in $\mathbb{R}^{3\times H\times W}$ with H and W being respectively the height and width of the image. Given that, usually images are high dimensional and the completely connected structure of multiplayer perceptrons introduces an enormous number of parameters. As described before, the number of parameters of a network is crucial in the bias-complexity tradeoff: the higher the number of parameters the bigger is the hypothesis class considered for learning the training set, so the higher the number of required samples during training to avoid overfitting. Conversely, reducing the hypothesis class could lead to underfitting, because the best predictor in it is not powerful enough to model the distribution to be learned. Indeed when evaluating a learning algorithm two measures have to be taken into account: its learning speed and its generalization performances. The main point of [24] is that good generalization performance can be obtained if some a priori knowledge about the task is built into the network: to this end, tayloring the network structure to the task can be thought as a way of reducing the set of possible functions that the network can implement without reducing too much its expressive power. Hence the challenge is to minimize the number of free parameters in the network, without reducing the size of the network to the point it cannot implement the desired function.

Another issue in using an unstructured MLPs with images is that they they have no built-in invariance with respect to translations, scale, or geometric distortions of the input: variations in the absolute positions of distinctive features in input are not relevant to the task. Using an MLP on handwritten digit recognition task, it has been observed that the network develops a set of matched filters to match an "average pattern" formed by superimposing all the training examples, and the classification is based on the computation of a weighted overlap between the input pattern and the "average prototype" [24]. This results in very low error on the training set (meaning that the task can be learned by the network) and high error on the test set. In principle, it is possible for a fully-connected network of sufficient size to learn how to

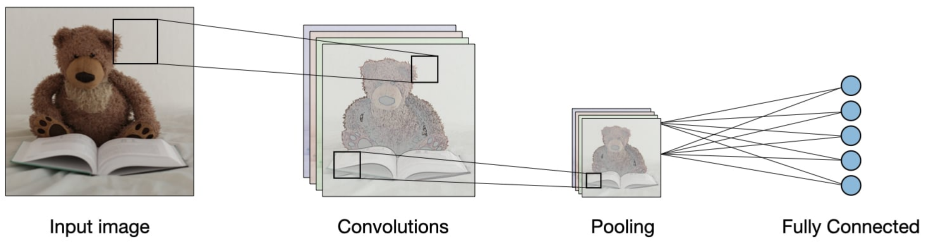


Figure 2.7. Overview of how a CNN works. Image taken from stanford.edu

produce outputs invariant to such variations, but this would probably result in multiple units with similar weight patterns positioned at various locations in the input, to detect the distinctive features wherever the appear on the input and a much larger training set would be necessary to cover the space of all possible variations [25]. Moreover, in fully connected architectures the input variables can be presented in any (fixed) order without affecting the outcome of the training, meaning that the topology is ignored, while images have a strong 2D local structure: variables represent pixels, and pixels in a spatial neighborhood are highly correlated. To this end, it is desirable to have a set of feature detectors that can extract local features independently of the position in input, and then use the hierarchical compositions of such features to recognize objects. Convolutional Neural Networks take advantage of such hierarchical feature extraction process to produce features that account for some degree of shift, scale and distortion invariance and are able to capture how local patterns interact to form the appearance of the image.

2.3.1 Convolutional Layer

In practice, convolutional layers perform a convolution on the input and transmit the result to the following layer: they do it by performing a convolution between a filter (or kernel) and the inputs received from the previous layer, sliding over them. This operation produces a *feature map*, that is a set of units whose weights are constrained to be identical. A complete convolutional layer is composed of several feature maps obtained with different filters, so that multiple features can be extracted at each location. Each filter has the same weights and bias when sliding on the input, and different filters have different sets of weights and biases to extract different feature maps.

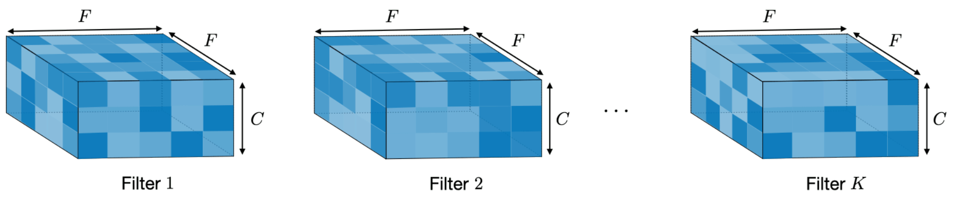


Figure 2.8. Convolutional filters in a CNN. Image taken from stanford.edu

Mathematically, a feature map is obtained using the following formula:

$$\texttt{Conv}(p,q) = \sum_i \sum_j f_{(i,j)} x_{(i+p,j+q)} \tag{2.34}$$

where f is the filter, x is the input of the previous layer and `Conv` is the resulting feature map. The indexes p and q are used to move on the input image while i and j are used to iterate over the filter's dimensions, multiplying each weight by the corresponding value of the input unit. The learning process for the convolutional layer then consists in learning the sets of weights and biases of the different filters, such that the features extracted are useful for the task. Three hyperparameters determine the output: *depth*, *stride*, and *padding*. The *depth* specifies the number of filters, the *stride* parameter determines of how many pixels is the filter moved while sliding across the image; finally the *padding* is used to increase the dimensions of the input with a variety of techniques (e.g. zero padding in images, adding zero pixels to the input data). Given these hyperparameters, the dimension of the filter $(H_f \times W_f)$ and the dimension of the input (e.g. the matrix associated with an image $M \in \mathbb{R}^{C \times H \times W}$, where C is the dimensionality of each pixel), the following simple formula can be used to obtain the output's dimensions:

$$\begin{aligned} H_{out} &= \frac{H - H_f + 2P}{S} + 1 \\ W_{out} &= \frac{W - W_f + 2P}{S} + 1 \\ C_{out} &= D \end{aligned} \tag{2.35}$$

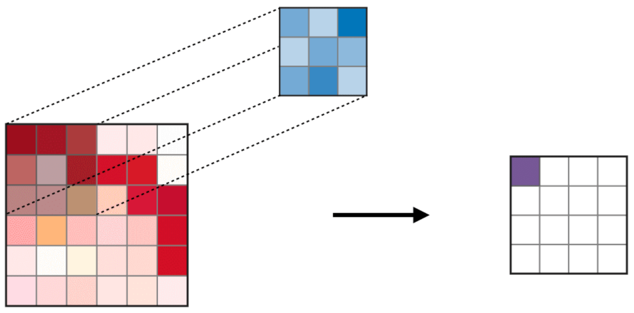


Figure 2.9. Convolutional operations in a CNN. Image taken from stanford.edu

2.3.2 Pooling layer

Once a features has been detected, its exact location becomes less important, only its approximate position with respect to other features is relevant. A simple way to reduce the precision with which the position of distinctive features are encoded in a feature map is to reduce the spatial resolution of the feature map. This can be achieved by using pooling layers, that compute a statistic on the feature map, scale it by a trainable coefficient and add a bias: widely used pooling operators are max pooling or average pooling, that respectively perform a the $max(\cdot)$ and the $avg(\cdot)$ operation on a feature map.

2.3.3 Fully-connected Layer

In deep learning the components described so far constitute what is called "(deep) feature extractor", that gathers relevant information from the input and eliminates irrelevant variabilities. The final part of the network is composed by a classifier, which discriminates the features representation extracted into the desired output categories. More in general, the classifier part is implemented by a MLP capable of learning potentially non-linear functions between high-level features.

2.3.4 Class Activation Maps (CAMs)

It has been demonstrated that the convolutional units of various layers of convolutional neural networks (CNNs) behave as object detectors despite the absence of supervision on the location of the object. The authors of [2] showed that this behavior could be generalized in order to begin locating

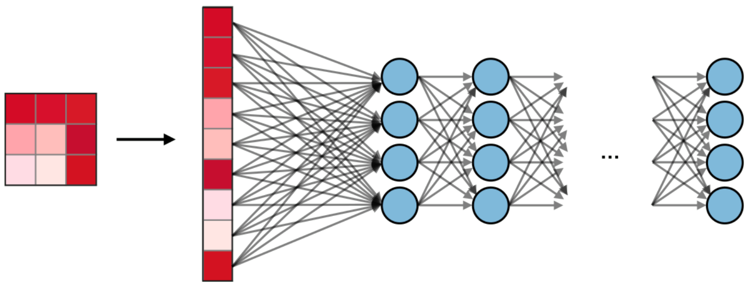


Figure 2.10. Fully connected layer in a CNN on higher level features. Image taken from stanford.edu

precisely which areas of an image are being used for discrimination. By using class activation maps, which are weighted activation maps created for each image, this method makes it possible to visualize CNNs. Class activation map for a particular category indicates the discriminative image regions used by the CNN to identify that category. These activation maps are obtained by by projecting back the weights of the output layer on to the convolutional feature maps: intuitively, based on prior works, each unit is activated by some visual pattern within its receptive field. In this way, the class activation map is simply a weighted linear sum of the presence of these visual patterns at different spatial locations. Finally the images in figure 6.2 are obtained by up-sampling the class activation map to the size of the input image, such that they translates to regions highlighting the regions most relevant to the predicted category. More formally, given an image let $f_k(x,y)$ the activation of unit k in the last convolutional layer at spatial location (x,y). Then, for unit k, denote result of performing global average pooling as $F_k = \sum_{(x,y)} f_k(x,y)$: so for a given class c, the input to the softmax, S_c, is $\sum_k w_k^c F_k$, where w_k^c is the weight corresponding to class c for unit k, and indicates the importance of F_k for class c. Given that, we obtain:

$$S_c = \sum_k w_k^c \sum_{(x,y)} f_k(x,y) = \sum_{(x,y)} \sum_k w_k^c f_k(x,y) \tag{2.36}$$

Then the class activation map for class c is defined as: $M_c(x,y) = \sum_k w_k^c f_k(x,y)$. In this way it is evident that $S_c = \sum_{(x,y)} M_c(x,y)$, and hence $M_c(x,y)$ determinates the importance of the activation at location (x,y) leading to the classification of an image to class c.

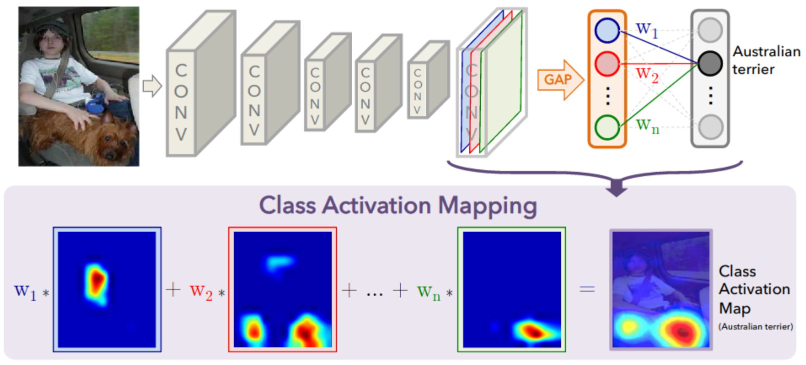


Figure 2.11. Graphical representation of how Class Activation Mapping works. Image taken from [2]

2.4 Residual Neural Networks

Convolutional Neural Networks, if compared with unstructured fully-connected networks, encode a prior knowledge about the task, namely the correlations present in the input, such that the learning algorithm can leverage less complex hypothesis classes to find the optimal predictor without overfitting the training data. Most importantly, the goodness of the predictor depends on the engineering of the network structure. More in detail, since the pattern extraction from data is hierarchical, the depth of the networks has become of paramount importance, especially with tasks of increasing complexity (e.g. datasets with real-world variabilities and larger label space): more complex tasks not only require bigger datasets but also demand for more powerful architectures, capable of capturing those variabilities that are not present in smaller tasks like handwritten digit recognition. The importance of CNNs depth has become clear yet in 2012, when the authors of [3] showed the potentialities of Deep Convolutional Neural Networks for large scale image classification with their AlexNet model, and noted that removing any convolutional layer (each of which contains no more than 1% of the model's parameters) resulted in inferior performance (about 2% loss). Other researches investigated more in depth the effect of network depth to performance: [26] fixed other parameters of the architecture, and steadily increased the depth of the network by adding more convolutional layers (up to 19 weight layers), keeping a small (3×3) convolution filters in all layers.

However, as models become deeper and more complex, they become more

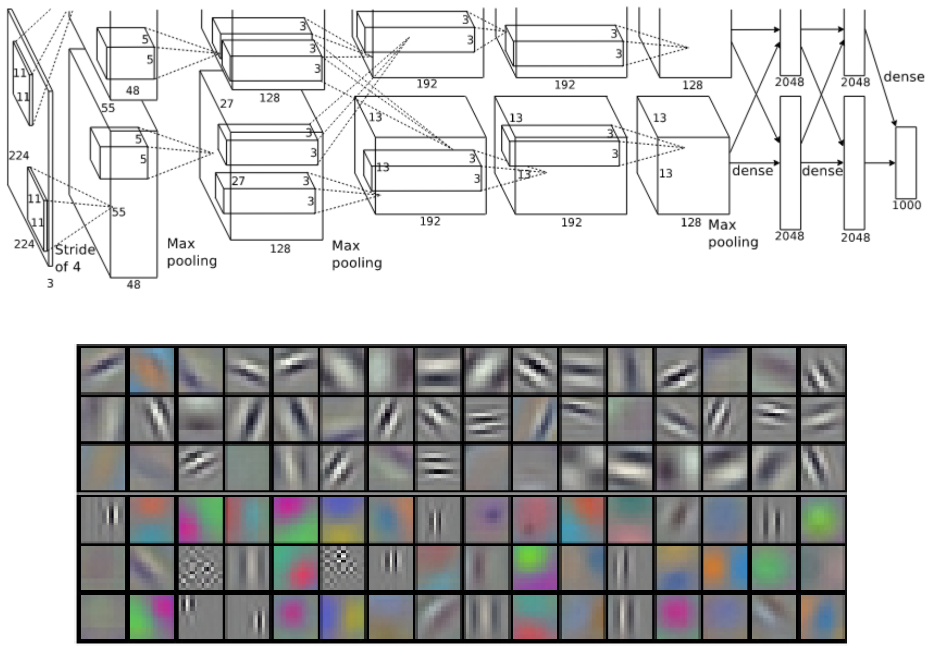


Figure 2.12. Architecture of AlexNet [3] (top) and corresponding convolutional kernels

difficult to train. Driven by the result of increasing the network depth, the authors of [4] started by this question: "Is learning better networks as easy as stacking more layers?". As it was already known, deep networks tend to suffer from vanishing or exploding gradients [27, 28], that has been largely addressed by normalized initialization and intermediate normalization layers [21], which enable networks with tens of layers to start converging using SDG and backpropagation.

With these problems addressed, a *degradation problem* has been exposed: with the network depth increasing, accuracy gets saturated and then degrades rapidly: counterintuitively, this phenomenon is not caused by overfitting, because adding additional layers to a sufficiently deep model results in increased training error. The degradation problem underlies the fact that there are underlying difficulties with optimizing a deeper model: by constructing a deeper model starting from a shallower architecture adding only identity mappings, authors of [4] showed that the optimization of a deeper architecture did not find a solution comparably good to the newly deeper model constructed from the shallower one, that is a model with no worse training error. To address this issues the authors proposed a *deep residual learning* framework, in which

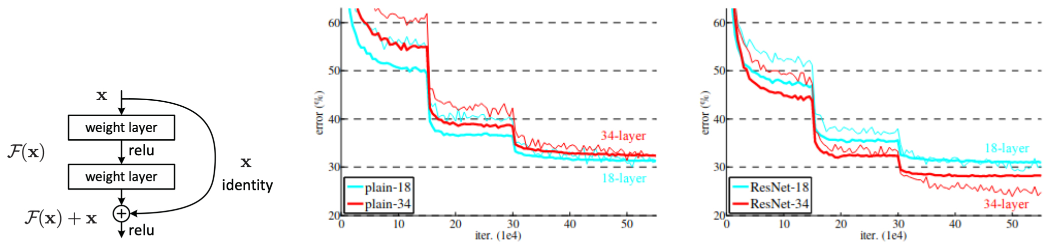


Figure 2.13. Graphical representation of a residual block (left) and comparison of error rates of the two networks without (left) and with residual blocks. Thin curves denote training error, and bold curves denote validation error of the center crops [4]

instead of making a group of stacked layers fit a desired underlying mapping $H(x)$, the network is build in a way such that it fits a residual mapping $F(x) \stackrel{\text{def}}{=} H(x) - x$, by casting the original mapping into $F(x) + x$. [4] suggest this change through the idea that optimizing the residual mapping is more straightforward than optimizing the initial, unreferenced mapping.

This reformulation is motivated by the degradation problem and the aforementioned construction of deeper network from a shallower one by adding identity mappings. In particular, if the added layers can be constructed as identity mappings, a deeper model should have training error no greater than its shallower counterpart. With the residual learning reformulation, if identity mappings are optimal, the solvers may simply drive the weights of the multiple nonlinear layers toward zero to approach identity mappings. If the optimal function is closer to an identity mapping than to a zero mapping, it should be easier for the solver to find the perturbations with reference to an identity mapping, than to learn the function as a new one.

The formulation $F(x) + x$ is achieved by *shortcut connections*, that skip one or more layers: they perform identity mapping, and their outputs are added to the outputs of the stacked layers, without requiring extra parameter nor computational complexity. The entire network can still be trained end-to-end by SGD with backpropagation.

2.5 Modern Architectures

This sections introduces two major modern architectures that have been used and in NLP and computer vision and that in recent years have been proven theirselves effective in video processing: recurrent neural networks and transformers. Their main importance in such fields is related to the encoding of temporal dynamics: in NLP, words in sentences have an intrinsic temporal dependence between each other, since sentences and groups of sentences have their meaning encoded not only in the set of words used, but also in their order. Similarly, frames in a video are tied together by the temporal causality imposed by the action being observed. For example, the actions "open the fridge" and "close the fridge" may share the same set of frames, but their order disambiguate the two actions, so the temporal information is useful to be captured, and simple CNNs were not designed to accomplish this task.

To model sequences, some design criteria need to be met; among these, an algorithm to be used to capture sequential information need to to:

- handle variable length sequences;
- track long-term relationships;
- mantain information about order;
- share parameters across the sequence.

Applications of sequence modelling can be characterized by the relationship mapping the input to the desired output:

- the input is a sequence, but the output has not temporal dynamics (many-to-one relationship): an examples of this class of problems is sentiment classification, in which the input is natural language and the output is a label representing the sentiment (e.g. positive, negative or neutral)
- the input is not a sequence, but the desired output has a temporal component (one-to-may relationship): an example of this kind is image captioning;
- the input is sequential, as well as the output (many-to-may relationship): an examples of this class of problems is machine translation, where the input is text in a language and the output is the same text in another language.

Two major architecture to handle such sequence modelling are Recurrent Neural Networks (RNNs) and Transformers.

2.5.1 Recurrent Neural Networks

The idea of RNNs can be build by starting analyzing the perceptron algorithm, that is the foundation of MLPs, and adding the necessary components to handle sequential data. It is possible to view a simple feedforward network as a function that, given and input $\mathbf{x}_t$ gives as output $\hat{y_t}$, for each time step t, but the problem is that this model is unable to relate the prediction $\hat{y_t}$ with the predictions at previous time steps $[\hat{y_t}, \hat{y_{t-1}}, ..., \hat{y_1}]$. The idea of RNNs is to relate the network's computations at a particular time step to its prior history and its memory of the computations from those prior time steps. In RNNs this recurrent relation is modelled by having an internal memory, o state, that is maintained time step to time step and passed forward the network across time. In this way the output of the network depends not only on the input at a particular time step but also on the state from the prior time step $\mathbf{H}_{t-1}$, in functional form $\hat{y_t} = f(\mathbf{X}_t, \mathbf{H}_{t-1})$; the state $\mathbf{H}_t$ is updated at each time step as the sequence is processed. So it is possible to describe this process of passing information from the previous iteration to the hidden layer with the following mathematical notation: let it be he the hidden state and the input at time step t respectively as $H_t \in \mathbb{R}^{n \times h}$ and $\mathbf{X}_t \in \mathbb{R}^{n \times d}$ where n is number of samples, d is the number of inputs of each sample and h is the number of hidden units. The network will have weight matrices from input to hidden state $\mathbf{W}_{xh} \in \mathbb{R}^{d \times h}$, from hidden state to hidden state $\mathbf{W}_{hh} \in \mathbb{R}^{h \times h}$, from hidden state to output $\mathbf{W}_{ho} \in \mathbb{R}^{h \times o}$ and bias parameters $\mathbf{b}_h \in \mathbb{R}^{1 \times h}$ and $\mathbf{b}_o \in \mathbb{R}^{1 \times o}$, where o is the output dimensionality. Then, being ϕ an activation function, the following equations describe the update of the hidden state and the output:

$$\begin{aligned} \mathbf{H}_t &= \phi_h(\mathbf{X}_t W_{xh} + \mathbf{H}_{t-1}\mathbf{W}_{hh} + \mathbf{b}_h) \\ \mathbf{O}_t &= \phi_o(\mathbf{H}_t \mathbf{W}_{ho} + \mathbf{b}_o) \end{aligned} \tag{2.37}$$

Since the recursive definition of $\mathbf{H}_t$, the output includes traces of the hidden states at all previous time steps.

Backpropagation through time (BPTT)

The standard backpropagation algorithm described in 3 is adapted to the case of RNNs by unfolding the structure of the net to resamble a simple feedforward network. In practice, when forwarding $\mathbf{X}_t$ through the network the hidden state $\mathbf{H}_t$ and the output state $\mathbf{O_t}$ is computed one step at a time, and the loss function is computed as the sum of every loss term for each time step l_t:

$$L(\mathbf{O},\mathbf{Y}) = \sum_{t=1}^{T} l_t(\mathbf{O}_t,\mathbf{Y}_t) \tag{2.38}$$

Since there are three weight matrices $\mathbf{W}_{xh}$, $\mathbf{W}_{hh}$ and $\mathbf{W}_{ho}$, it is needed to compute the partial derivative w.r.t. to each of them:

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{W}_{ho}} &= \sum_{t=1}^{T} \frac{\partial l_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \frac{\partial \phi_o}{\mathbf{W}_{ho}} = \sum_{t=1}^{T} \frac{\partial l_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{H}_t \\
\frac{\partial L}{\partial \mathbf{W}_{hh}} &= \sum_{t=1}^{T} \frac{\partial l_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \frac{\partial \phi_o}{\partial \mathbf{H}_t} \cdot \frac{\partial \mathbf{H}_t}{\partial \phi_h} \cdot \frac{\partial \phi_h}{\partial \mathbf{W}_{hh}} \\
&= \sum_{t=1}^{T} \frac{\partial l_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \cdot \frac{\partial \mathbf{H}_t}{\partial \phi_h} \cdot \frac{\partial \phi_h}{\partial \mathbf{W}_{hh}} \\
\frac{\partial L}{\partial \mathbf{W}_{xh}} &= \sum_{t=1}^{T} \frac{\partial l_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \frac{\partial \phi_o}{\partial \mathbf{H}_t} \cdot \frac{\partial \mathbf{H}_t}{\partial \phi_h} \cdot \frac{\partial \phi_h}{\partial \mathbf{W}_{xh}} \\
&= \sum_{t=1}^{T} \frac{\partial l_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \cdot \frac{\partial \mathbf{H}_t}{\partial \phi_h} \cdot \frac{\partial \phi_h}{\partial \mathbf{W}_{xh}}
\end{aligned} \tag{2.39}$$

Since $\mathbf{H}_t$ depends on the previous time step, it is possible to explicit this in the above equations:

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{W}_{hh}} &= \sum_{t=1}^{T} \frac{\partial l_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \sum_{k=1}^{t} \frac{\partial \mathbf{H}_t}{\partial \mathbf{H}_k} \cdot \frac{\partial \mathbf{H}_k}{\partial \mathbf{W}_{hh}} \\
&= \sum_{t=1}^{T} \frac{\partial l_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \sum_{k=1}^{t} (\mathbf{W}_{hh}^{\top})^{t-k} \cdot \mathbf{H}_k \\
\frac{\partial L}{\partial \mathbf{W}_{xh}} &= \sum_{t=1}^{T} \frac{\partial l_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \sum_{k=1}^{t} \frac{\partial \mathbf{H}_t}{\partial \mathbf{H}_k} \cdot \frac{\partial \mathbf{H}_k}{\partial \mathbf{W}_{xh}} \\
&= \sum_{t=1}^{T} \frac{\partial l_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \sum_{k=1}^{t} (\mathbf{W}_{hh}^{\top})^{t-k} \cdot \mathbf{X}_k
\end{aligned} \tag{2.40}$$

From these equations it is evident that the optimization algorithm needs to store the powers of $\mathbf{W}_{hh}^{k}$ as each term l_t is calculated, and L can become large. For this reason, a common issues is exploding gradients: a common solution is to use a *Truncated BPTT*, that consists in truncating the sum at certain size, limiting in fact the number of the time steps considered for gradient computation, and doing so dynamically limiting the number of hidden layers. Another problem is vanishing gradients: in fact, if there are small values (< 1) in $\mathbf{W}_{hh}$ this causes the gradient to decrease going deeper in the hidden layer, limiting the contribution of states far from the last time step. This problem motivated the introduction of the long short term memory units (LSTMs), that are more complex recurring units, using gates to selectively add or remove information within each recurring unit.

LSTMs

Long Short-Term Memory Units (LSTMs) [29] were designed to properly handle the vanishing gradient problem, allowing for a longer-term memory (over 1000 time steps). To this end, LSTMs store more information outside of the traditional neural network flow in structures called gated cells: there is an output gate $\mathbf{O}_t$ to read entries of the cell, an input gate $\mathbf{I}_t$ to read data into the cell and a forget gate $\mathbf{F}_t$ to reset the content of the cell. Being $\mathbf{W}_{xi}$, $\mathbf{W}_{xf}$, $\mathbf{W}_{xo} \in \mathbb{R}^{d\times h}$ and $\mathbf{W}_{hi}$, $\mathbf{W}_{hf}$, $\mathbf{W}_{ho} \in \mathbb{R}^{h\times h}$ the weight matrices and $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o \in \mathbb{R}^{1\times h}$ their respective biases and σ the sigmoid function, the computations performed by each gate are:

$$\begin{aligned}\mathbf{O}_t &= \sigma(\mathbf{X}_t\mathbf{W}_{xo} + \mathbf{H}_{t-1}\mathbf{W}_{ho} + \mathbf{b}_o)\\ \mathbf{I}_t &= \sigma(\mathbf{X}_t\mathbf{W}_{xi} + \mathbf{H}_{t-1}\mathbf{W}_{hi} + \mathbf{b}_i)\\ \mathbf{F}_t &= \sigma(\mathbf{X}_t\mathbf{W}_{xf} + \mathbf{H}_{t-1}\mathbf{W}_{hf} + \mathbf{b}_f)\end{aligned} \tag{2.41}$$

Moreover there is a candidate memory cell $\widetilde{\mathbf{C}}_t \in \mathbb{R}^{n\times h}$, having its own weights $\mathbf{W}_{xc} \in \mathbb{R}^{d\times h}$, $\mathbf{W}_{hc} \in R^{h\times h}$ and biases $b_c \in \mathbb{R}^{1\times h}$:

$$\widetilde{\mathbf{C}}_t = tanh(\mathbf{X}_t\mathbf{W}_{xc} + \mathbf{H}_{t-1}\mathbf{W}_{hc} + \mathbf{b}_c) \tag{2.42}$$

Finally the new memory content depends on the content at previous time step, on the forgetting gate and on the input gate:

$$\begin{aligned}\mathbf{C}_t &= \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \widetilde{\mathbf{C}}_t\\ \mathbf{H}_t &= \mathbf{O}_t \odot tanh(\mathbf{C}_t)\end{aligned} \tag{2.43}$$

In this way, LSTMs make the backpropagation through time algorithm more stable, mitigating against the vanishing gradient problem having a smoother flow of gradients.

Limitations of RNNs

Despite all the advantages and the practical success of recurrent models, there are some drawbacks:

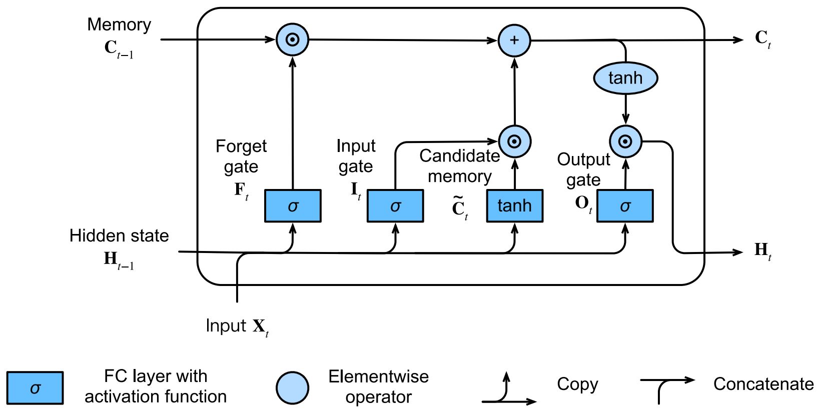

Memory $\mathbf{C}_{t-1}$
$\mathbf{C}_t$
tanh
Forget gate $\mathbf{F}_t$
Input gate $\mathbf{I}_t$
Candidate memory $\tilde{\mathbf{C}}_t$
Output gate $\mathbf{O}_t$
σ
tanh
Hidden state $\mathbf{H}_{t-1}$
$\mathbf{H}_t$
Input $\mathbf{X}_t$
FC layer with activation function
Elementwise operator
Copy
Concatenate

Figure 2.14. Illustration of computations in a LSTM cell. Image taken from d2l.ai

- **Encoding bottleneck:** all the content to be passed forward the network must be first encoded into a suitable representation, and in this process information can be lost, especially in case of long sequences;

- **Slowness, poor exploitation of parallelization algorithms:** RNNs are not computationally efficient, mostly because by construction they require sequential computations, time step by time step, so they can't fully exploit the parallelization available by using GPUs;

- **Not so long-term memory:** RNNs do not scale with sequences longer than thousands of elements.

In response for these limitations, the attention mechanism has been developed and used as a principle for an architecture called *Transformer*.

2.5.2 Transformers

Transformers [5] were introduced in 2017, and it is a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. Specifically, they rely on self-attention for relating different positions of a single sequence in order to compute a representation of the sequence.

Transformers are based on a encoder-decoder structure, where recurrence units are completely dismissed. Each element of the input sequence is transformed in an embedding similar to how it is done with RNNs, but additionally the consumed representation is enriched with positional information, that encodes the notion of order in the input sequence (position-aware encoding). Having these features, the attention function can be described as mapping a query and a set of key-value pairs to an output. The output is computed as a weighted sum of the values, whose weight depends on a similarity function between the query and the key. The attention function is computed on a set of queries, keys and values simultaneously, represented by matrices **Q**, **K** and **V**, with the following operation:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{QK}^\top}{\sqrt{d_k}})\mathbf{V} \tag{2.44}$$

Where d_k is the dimension of queries and keys and d_v is the dimension of values. These operations form what is called "scaled dot-product attention", and in the transformer architecture this structure is repeated h times by linearly projecting the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. The attention mechanism is then computed in parallel and at the end the resulting features are concatenated and fed to a linear layer, forming what is called "Multi-Head Attention". Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions.

$$\begin{aligned} MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Concat(head_1, ..., head_h)\mathbf{W}^O \text{ , where} \\ head_i = Attention(\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V) \end{aligned} \tag{2.45}$$

Where the projections are parameter matrices $\mathbf{W}_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{model}}$ ($d_{model} = 512$).

Encoder and decoder stacks

The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers: a multi-head self-attention block, and a position wise

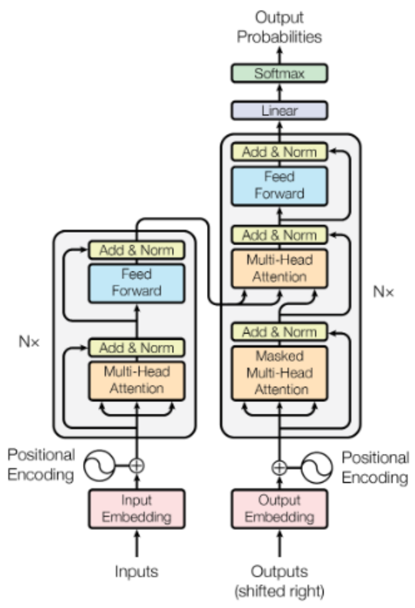


Figure 2.15. Transformer overall architecture, image from [5]

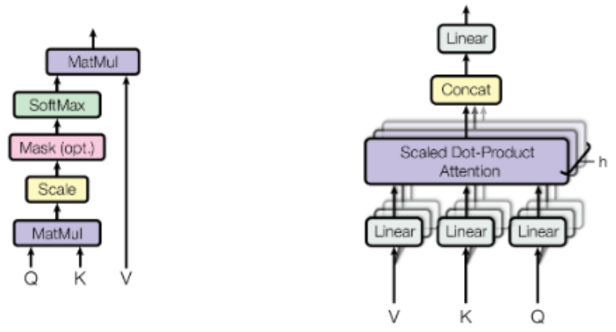


Figure 2.16. Scaled Dot-Product Attention (left) and Multi-Head Attention (right) blocks in the transformer architecture, image from [5]

fully connected feed-forward network. Additionally, around each of these sub-layers it is added a residual connection and a layer normalization. The decoder is also composed of a stack of $N = 6$ identical layers, with the addition of a third sub-layer, which performs multi-head attention over the output of the encoder stack, with similar residual connections and layer normalizations. The self-attention is modified to prevent positions from attending to subsequent positions. Transformers have been used mostly on NLP tasks (like in BERT [30] and GPT-3 [31] architectures) but also on biological sequences (like in Alpha Fold 2) and most recently even in computer vision, with the introduction of Vision Transformers [32].

Chapter 3

First Person Action Recognition

This chapter introduces the main task that is been treated in this work, starting from the main solutions of the state-of-art to the proposed enhancement in the context of multi-modal learning. This chapter introduces the background of this work on First Person Action Recognition in multi-modal context. The first section describes the task of first person action recognition, the second will introduce a recently released large-scale dataset commonly used as benchmark in this field, while the fourth introduces the concept of multi-modal learning. The latter sections will describe the state of the art architectures for dealing with video processing and introduce the concepts of Domain Adaptation and Domain Generalization, with particular focus on how they apply to video action recognition.

3.1 Task

Action Recognition is a multiclass classification problem, where the objective is to assign the correct action label to input videos. The domain set $\mathbb{X} = \{V_0, ..., V_N\}$ is the the set of videos, while the label set $\mathbb{Y}$ is the (fixed) set of actions any video can belong to. In principle the task is similar to classical classification, however there is a significant difference in the input data, and consequently in how to fully exploit them to obtain best performances. To this end, two factors must be taken into account:

- a video could be represented by means of the (ordered) sequence of frames that constitutes it, e.g. $V_i = \{F_0, ...F_M\}$ and so falling back to image classification on these frames, but there are additional perceptual inputs that can be used to perform classification (multi-modal learning);

- with respect to classical image classification, a video contains also a temporal dynamics that is often crucial to perform the classification, so the need to ad-hoc solutions to effectively incorporate this information. Resorting to an example, frames in a video are tied together by the temporal causality imposed by the action being observed. For example, the actions "open the fridge" and "close the fridge" may share the same set of frames, but their order disambiguate the two actions.

In addition to this, First Person Action recognition belongs to the egocentric-vision world, in which the recording equipment is worn by the observer: this means that the viewpoint is the observer's one, and that the recording shows higher degree of change with respect to fixed third person camera: environment and lighting conditions, occlusions caused by the observer interaction, viewpoint changes. At the same time, the possibility to extract value from these data opens interesting opportunities due to the recent spread of wearable devices:

- Having wearable technologies capable of collecting fine-grained data means that it is possible to further exploit deep approaches, that usually require a great amount of data;
- Being able to develop mature technology for first person action recognition enables such devices to be equipped with additional intelligence, with benefits in human-robot interaction or human assistance as well in automatic video segmentation for a variety of human centered actions.

To this end, here we summarize the principal desiderata for first person action recognition:

- Deal with *egomotion*, a phenomenon arising from the rapid and involuntary motion of the wearable device, which inevitably moves around with the user;
- Effectively encode the temporal dimension and generalize to variable sequences length and speed;
- Work with fine-grained actions: this is the problem of distinguishing very specific actions such as "unscrew filter from aeropress" or "remove packaging from salmon" instead of coarse-grained interactions such as

"preparing a meal". This requires reasoning not only about the general action performed and the principal object of interest but also considering how the participant is interacting with the environment and with surrounding items;

- Generalize to different environments: given the complexity of the task, it is unmanageable to have enough samples to account for any possible variation of the same action, since environment the high variability of the environment. This means that a good algorithm should also be invariant of the environment in which the action is performed, which implies that the model should not rely on environmental characteristics but on the relevant part of the input. This issue is also refereed as extracting discriminative yet domain invariant features, and it is a problem we will discuss more in detail later.

- Exploit multi-modal information to make predictions stronger and more reliable: as humans, it has been known that different perceptual stimuli are often combined in a way that enhance our understanding of the environment, adaptively relying on perceptual information from other senses when the considered one is not sufficient. The current state of the art has not yet found an optimal way to leverage different perceptual inputs; this problem of multi-modal learning will be discussed more in detail later.

3.2 EPIC-KITCHEN Dataset

In this section we provide a detailed description of the EPIC-Kitchens dataset [13], which a recent released large-scale, multi-modal dataset of fine-grained actions recorded in first person, that has been commonly used as benchmark in this field. The availability of a good dataset is crucial for research in deep learning: in recent years significant progress in many domains such as image classification and object detection are due not only to advances in the theory of deep learning, but also to the availability of large-scale image benchmarks, such as ImageNet [33], ADE [34] or VOC [35]. However, in the video understanding field only a small number of annotated video datasets were available to the research community, and this has been a reason for slower success.

Various datasets have been released recently: most of these consisted of a collection of short videos, each recording a single action performed by the user. Hollywood in Homes [36] was the first attempt to collect longer videos showing humans performing various tasks at their home: videos were recorded in a scripted way (meaning that participants sequentially follow a series of instructions), causing them to be less natural and lacking the progression and multi-tasking of actions that occur in real life. Thus, EPIC-KITCHENS seeks to be a dataset that reflects the demands of egocentric vision, including rich annotations, the capacity to perform multiple tasks on it, and multi-environment recorded. It is made up of 55 hours of recorded video, 11.5 million frames, bounding boxes around the objects the user interacts with, and other data that was gathered in the native kitchens of 32 participants from 10 different nationalities. The participants were instructed to record all of their routine kitchen tasks, regardless of the sequence length. The recordings, which combine sound and video, also demonstrate the natural multitasking that people engage in, such as washing a few dishes while they cook, increasing the realism and difficulty.

Participants were instructed to record every time they went into the kitchen for at least three days straight, stopping the recording only when they left the kitchen. To ensure that the camera only records one person's activities, they were asked to spend the entirety of the recordings alone in the kitchen. The tool used was a head-mounted GoPro with a mounting that could be adjusted to control the angle for various environments and participant heights. Stereo audio was captured from the Go-Pro's built-in microphone, at a sampling rate of $48000kHz$ and a bit rate of $128kb/s$. The camera was set to linear field of view (fov), 59.94 fps and Full HD resolution of 1920×1080. Considering how difficult it is to crowdsource annotations for such lengthy videos, participants contributed a rough initial annotation themselves. Then, after the recording, they were asked to narrate the actions taken using a hand-held recording device. The authors attempted to group all of the collected verbs into semantic classes in order to adhere to the standard multi-class approach in which each sample is assigned to a single class. First, they attempted to achieve automatic clustering using WordNet/Word2Vec combined with Part-of-Speech techniques to distinguish nouns and verbs, as well as turning to manual correction when the results of applying these techniques were unsatisfactory. The dataset was divided into two main splits:

- **Seen Kitchens (S1):** each kitchen is seen in both training and testing, with a proportion of training/test set of 80%-20%.

- **Unseen kitchens (S2):** this divides the participants/kitchens so that all sequences of the same kitchen are either in training or testing. Complete sequences for 4 participants are hold out for the testing purposing, accounting for the 7% of the split.

The challenges defined by the dataset's authors are:

- **Action Recognition:** it consists in assigning a (verb, noun) label to a trimmed video segment. The evaluation metrics are top-1/5 accuracy for verb, noun and action (verb+noun), calculated for all segments as well as unseen participants and tail classes.

- **Action Detection:** it consists in detecting the start and the end of each action in an untrimmed video, and assigning a (verb, noun) label to each detected segment. The evaluation metric is the Mean Average Precision (mAP) @ IOU 0.1 to 0.5.

- **Action Anticipation:** it consists in predicting the (verb, noun) label of a future action observing a segment preceding its occurrence. The evaluation metric is the top-5 recall averaged for all classes, as defined here, calculated for all segments as well as unseen participants and tail classes.

- **Domain Adaptation for action recognition:** it consists in assigning a (verb, noun) label to a trimmed segment, following the Unsupervised Domain Adaptation paradigm, meaning that the training is performed on a labelled source domain, and the model needs to adapt to an unlabelled target domain. The evaluation metrics are top-1/5 accuracy for verb, noun and action (verb+noun), on the target test set.

- **Multi-instance Retrieval**:

  - **Video to text:** given a query video segment, rank captions such that those with a higher rank are more semantically relevant to the action in the query video segment.

  - **Text to video:** given a query caption, rank video segments such that those with a higher rank are more semantically relevant to the query caption.

  The evaluation metric is the normalised Discounted Cumulative Gain (nDCG) and Mean Average Precision (mAP).

3.2.1 EPIC-KITCHENS-100 Dataset

The first version of the EPIC-KITCHENS dataset was released in 2018: from its release it has been successful and many researchers took up the various challenges it enables. As result of it success, the authors of [13] extended this dataset, that will be called from here on EK-55 for brevity, leading to the release of EK-100, a collection of 100 hours, 20M frames, 90K actions in 700 variable-length videos, capturing long-term unscripted activities in 45 environments, adopting a similar procedure used for EK-55. New data was collected in the following way: the 32 participants of EK-55 project were contacted to record additional footage, 16 of them agreed to participate and half of them had moved house during the two years period from EK-55. Other than using a refined annotation pipeline, that allowed denser and more complete annotations of actions in untrimmed videos, the value of this new dataset for our purposes is in the investigations that authors of [17] carried out:

- **Test of time:** This challenge involves evaluating how models trained on EK-55 videos perform on videos collected two years later. It has been noted that when new data is evaluated, the performance of the same model suffers noticeably. This issue, known as a domain gap, results from discrepancies between the test set used to evaluate the models and the dataset on which they were trained. This domain gap can be explained by a variety of factors, including shifting contexts, labels, participants, and locations (even the same person's behavior can change over time), as well as shifting data collection techniques (hardware has changed) or shifting environmental conditions (some participants have moved house).
- **Scalability test:** It involves assessing how model performance scales with additional annotated data. According to analyses, when 50% of new data is added to training, results show a significant improvement, albeit saturating, especially for participants who cannot be seen. These findings suggest that better models must be created and that having more diverse data is advantageous over simply having more data.

To better explore this concept of compound domain gap, rather then leveraging domain labels for each factor accountable for change (e.g. recording camera, location, time-of-day), which anyway are provided, because it is also difficult to provide these labels for other change factors (e.g. appearance changes, change in participants behaviour), authors propose a new challenge on unsupervised adaptation for action recognition.

Unsupervised Domain Adaptation (UDA) for Action Recognition

This is the challenge we use to evaluate our results, so we provide some details. The action recognition task is similar to how it has been described above, but with some difference in the data available for training and test. In particular, it consist in using a labelled source domain classical for supervised learning with the possibility to use unlabelled target domain samples to perform domain adaptation. Specifically the following splits are provided, in such a way the this challenge tackles the *test of time* issue:

- **Source:** labelled training data from 16 participants (collected in 2018) and
- **Target:** unlabelled footage from the same 16 participants of EK-55, further splitted into:
  - **Target Train:** it is composed by unlabelled videos used during domain adaptation;
  - **Target Test:** it is composed by videos used for evaluation.

The challenges of this task stem from the source and target domains belong to distinct training distributions due to the collection of videos two years later. The differences in these distributions account for all the variabilities mentioned earlier, and constitute what is called *domain shift*. In the literature, this kind of task is usually performed by using two different dataset for representing the source and target distributions (e.g. UCF and Olympics datasets): in practice one of them is used in supervised way (e.g. exploiting the class labels), while the other is use to perform adaptation, without using the class labels. EPIC-KITCHENS-100 is the first to propose a within-dataset domain adaptation challenge in video. Some of the additional challenges of video UDA with respect to image UDA are: aligning temporal information across domains, attending to relevant transferable frames, and avoiding non-informative background frames.

3.3 Multimodal FPAR

Recent advances in sensors, mobile computing, wearable devices and deep learning techniques have lead to the idea of efficiently using more than one data source to improve the performance of learning algorithms. As it is known in biology, perceptual information coming from different senses can be combined to have a more complete understanding of the environment. As compared with uni-modal learning, multi-modal learning has several advantages:

- **Data complementarity:** This is due to the fact that cues from various modalities can augment or complement one another. It is possible to comprehend this by examining various learning domains: it is important to take into account both the literal meaning of the words as well as the nonverbal contexts in which these words appear. Nonverbal contexts include vocal patterns and facial expressions. To this end, [37] leverages multiple modalities (video, audio and text) to capture the dynamic nature of nonverbal intents by shifting word representations based on the accompanying nonverbal behaviors. A similar idea is used in Multi-modal Emotion Recognition (MER) [38, 39] and in sentiment analysis for learning how the context affect the prediction [40, 41, 42].
- **Model robustness:** models leveraging more than one modalities can show robustness against a modality missing at test time by leveraging the remaining ones, or taking advantage from the data complementary to not be confounded by similar inputs in one modality belonging to different label class.
- **Performance superiority:** for the above motivations, multi-modal models often exhibits better performances

In the context of first person action recognition, several studies have been conducted in an attempt to improve the performances by taking advantage of two or more data sources (e.g., audio and video) [43, 44, 45]. To this extend, it is possible to model the domain space $\mathbb{X} = \{V_0, ..., V_N\}$ by characterizing the video instances as the set of (ordered) sequences of corresponding modalities, $V_i = \{V_i^{M_0}, ..., V_i^{Mz}\}$, where M_j represent the modality in use. In our case we use RGB, (Optical) Flow and Audio modalities, so a video segment, that is a portion of a video with in which there is an action, can be represented by $V_i = \{V_i^R, V_i^F, V_i^A\}$, being V_i^R, V_i^F and V_i^A the sequences of instances of the corresponding modality belonging to the same segment. As belonging to different perceptual kind, different modality inputs come from different distributions, so the problem of multi-modal learning consists in how to effectively extract features from these different data stream, so learning the proper embedding, such that the classification task can benefit from it. In a domain adaptation perspective, this means finding feature representations that are discriminative, such that the classificator can more easily distinguish among classes, and domain invariant, so that they capture only the characteristics of input that are relevant for the task and not irrelevant domain-dependant features. In fact this kind of characteristics of the input can harness the model performance when it is deployed to the real-world, because the model may will rely on features encountered during training that are different when actually using the model. An example of this issue is relying on appearance characteristics of an object (e.g. the colour of a mug) to predict the class it belongs to: clearly such a model is biased to what encountered during training, and special care is needed to "guide" the network to search for other characteristics that instead are domain-invariant, so they are distinctive of the object, without the need of exposing the model to the too wider range of variabilities needed to learn the best representation. Another challenge of integrating multiple modalities to the same classification task is leveraging their complementarity, isolating possible noise and avoid conflicts, so optimizing not only with respect to the single modality, but jointly finding the feature extraction that works best considering the data as a whole. The multi-modal approach has been widely employed in literature [8, 12, 46, 47, 48]. RGB data streams are usually combined with optical flow ones since the latter encode motion information which can be complementary to the former, and the same time potentially more domain-invariant, since lacks the appearance details from the RGB stream. Moreover, recently also audio modality has been demonstrated itself effective in the egocentric context [7, 49, 50]. The next sections will describe the principal architecture for video processing and the principal strategies for fusing modality representations.

3.3.1 Fusion strategies for multi-modal learning

As studies on the development of embodied cognition suggest, simultaneous multi-modal stimuli are crucial in human perceptual learning. However, with deep learning models, numerous factors are to be faced when dealing with fusing multiple modalities: the representations extracted by a neural network can be very different as different is the encoding of each modality in the input data. For the task at hand, one modality may have more informative content than another, and one modality can be more or less challenging to network to be learned.

For this reason, fusion strategies define how a joint embedding for multi-modal inputs is created. There are three common techniques [51] used to tackle the fusion of different data streams.

- **Early fusion**: low-level modality-specific features are extracted from modality streams, and fused together before being fed to the final (shared) classifier. The representations of modalities can be very different but, being fused before the final classification layer, features from different modalities can interact by means of the part of the network that is after the fusing point;
- **Late fusion**: each stream is separate and perform all the operation from the inputs to the classification, and at the end all the classifications scores are fused together.
- **Hybrid fusion**: in this case the different modalities are first treated with an early fusion approach and then with a late fusion one. As a result different predictions scores are created and they are finally combined together.

In addition to decide where to fuse the features being extracted from the different streams, another issue is how to perform fusion. In [7] three mid-level fusion mechanisms are compared:

- **Concatenation**: the feature maps of each modality are concatenated and a fully-connected layer is used to model the cross-modal relations;
  $$f^{concat} = \phi(\mathbf{W}[V_i^{M_0}, ..., V_i^{Mz}] + b) \tag{3.1}$$
- **Context gating**: it aims at recalibrating the strength of the activations of different units with a self-gating mechanism;
  $$f^{context} = \sigma(\mathbf{W}h + b) \odot h \tag{3.2}$$
  where h is the multi-modal fusion with concatenation
- **Gating fusion**: a gate neuron takes as input the features from all modalities to learn the importance of one modality w.r.t. all modalities.

$$\begin{aligned} h_i &= \phi(\mathbf{W}_i m_i + b_i)\forall i \\ z_i &= \sigma(\mathbf{W}_{zi}[V_i^{M_0}, ..., V_i^{Mz}] + b_{zi})\forall i \\ f^{gating} &= \sum_{i=1}^{M} z_i \odot h_i \end{aligned} \tag{3.3}$$

However there are also more complex fusion methods that aim at enhancing the information sharing across modalities, like [52] that makes use of a transformer-based architecture that uses "fusion bottlenecks" for modality fusion at multiple layers.

3.3.2 Opportunities and pitfalls

Multi-modal inputs give the opportunity to exploit a wider range of characteristics of the task at hand: while the different inputs share the same semantic, they offer a different perspective that can complement each other, recognizing discriminative signals from the most appropriate input, enhancing confidence in the predictions, hence leading to a more robust model. However, from an architectural perspective, there is no clear winner about the optimal way to train on multi-modal inputs: considering an end-to-end training of a multi-modal and a uni-modal network on a task with multiple modalities, the multi-modal network should match or outperform its uni-modal counterpart, because it is given more information. Surprisingly, often this is not the case, and it has been observed in multiple works on video classification. In fact, training multi-modal classification networks is harder than training their uni-modal counterparts: having multiple modalities opens the possibility to leverage data complementarity, but networks tend to be more complex, i.e. with much more trainable parameters. For this reason, multi-modal networks are often more prone to overfitting due to their increased capacity. Moreover, different modalities overfit and generalize at different rates, so their joint training with a unified optimization strategy can be sub-optimal. This motivates the need of finding effective architecture and regularization techniques designed ad-hoc for multi-modal tasks, like the one proposed in [53] .

3.4 Architectures

One of the first deep architectures for action recognition [6] extends the deep convolutional networks, that has been proven effective in image classification, to action recognition in video data by employing a two stream model, decoupling the part that recognizes the spatial information from the one that encodes temporal relationships. The spatial stream is dedicated to the appearance of the video, and performs image recognition from still frames, while the temporal stream is trained to recognise action from motion in the form of dense optical flow. The two streams are then fused following a late-fusion approach. Both networks are base on Convolutional Neural Networks, and the spatial net is pretrained on ImageNet to leverage the availability of large amounts of annotated image data. This idea behind this architecture stems the two-streams hypothesis, according to which the human visual cortex contains two pathways: the ventral stream (which performs object recognition)

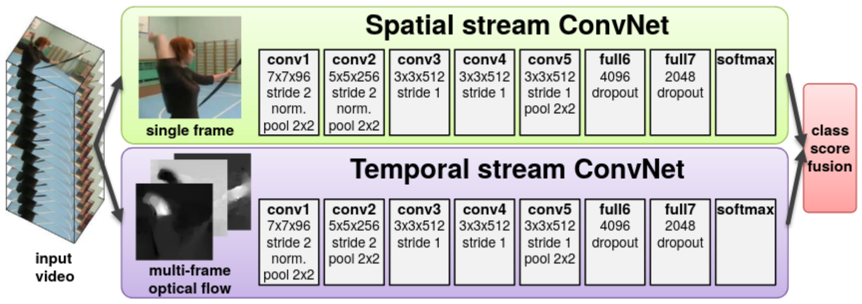


Figure 3.1. Structure of the two-stream convolutional network proposed in [6]

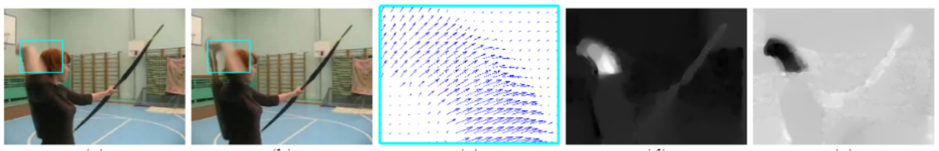


Figure 3.2. Illustration of optical flow. From left to right: a pair of consecutive video frames with the area around a moving hand outlined with a cyan rectangle; a close-up of dense optical flow in the outlined area; horizontal an vertical component of the displacement vector field. Image taken from [6]

and the dorsal stream (which recognises motion). However this two-stream network fail in modeling long-range temporal structure. This is mainly due to their limited access to temporal context as they are designed to operate only on a single frame (spatial networks) or a single stack of frames in a short snippet (temporal network) [46]. To this end, different deep architectures have been developed in recent years.

3.4.1 Temporal Segment Networks (TSN)

According to the authors of [46], the simple two-stream network fails in modeling long-range temporal structure, since their limited access to temporal context as they are designed to operate only on a single frame (spatial networks) or a single stack of frames in a short snippet. Approaches that rely on intensive temporal sampling at a fixed interval (*dense sampling*) usually incur in excessive computational cost when used on long video sequences, so limiting their application in context where the temporal dynamic is longer than the maximum sequence extend and introducing the risk of missing useful information. To address this issue, Temporal Segment Network (TSN)

[46], aims to utilize the visual information of the entire videos to perform video-level prediction, by operating on a sequence of short snippets sparsely sampled from the entire video and computing a consensus among the individual snippets' predictions as final video-level prediction. In such a framework, the model parameters are optimized calculating the loss function on video-level predictions, other than on snippet-level predictions which were used in two-stream ConvNets. Formally, being V_i a video that has been divided in K segments $S_1, ..., S_K$ of equal duration, the results of the prediction is computed as follows:

$$TSN(T_1, T_2, ..., T_K) = \mathcal{H}(\mathcal{G}(\mathcal{F}(T_1; \mathbf{W}), \mathcal{F}(T_2; \mathbf{W}), ..., \mathcal{F}(T_K; \mathbf{W}))) \tag{3.4}$$

where:

- $[T_1, T_2, ..., T_K]$ is the sequence of snippets, each of which has been sampled from its corresponding segment S_k;
- $\mathcal{F}(T_k; \mathbf{W})$ is the function represented by the ConvNet with parameters $\mathbf{W}$, that takes as input a short snippet T_k and outputs a prediction score for the snippet;
- $\mathcal{G}$ is the segmental consensus function, that takes as input the prediction score for the snippets and outputs a consensus of class hypothesis;
- $\mathcal{H}$ is the prediction function for the video-level scores from the consensus function, in this case the widely used softmax function.

Combining with standard categorical cross-entropy loss, the final loss function regarding the segmental consensus is:

$$\mathcal{L}(y, \mathbf{G}) = -\sum_{i=1}^{C} y_i \left(G_i - log \sum_{j=1}^{C} \exp\ G_j \right) \tag{3.5}$$

with C being the number action classes and y_i the ground truth for sample i. Although the choice of K and $\mathcal{G}$ is free, authors of [46] have chosen $K = 3$ and a evenly averaging functions as segmental consensus function.

Choosing properly the segmental consensus function, the temporal segment network is differentiable, allowing for the simultaneous optimization of model parameters utilizing all snippets via the following loss function:

$$\frac{\partial \mathcal{L}(y, \mathbf{G})}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{G}} \sum_{k=1}^{K} \frac{\partial \mathcal{G}}{\partial \mathcal{F}(T_k)} \frac{\partial \mathcal{F}(T_k)}{\partial \mathbf{W}} \tag{3.6}$$

3.4.2 Temporal Binding Network (TBN)

Authors of [7] took a different perspective in dealing with multi-modal networks. The potential temporal asynchrony between the discriminative cues in different streams are the core of the work: previous multi-modal fusion architectures for action recognition perform temporal aggregation within each modality before fusing them or, when fusing them before temporal aggregation, they synchronize the input across modalities [54]. Nevertheless in some actions, for example "breaking an egg into a pan", the distinct sound of cracking the egg, the motion of separating the egg and the change in appearance of the egg occur at different frames/temporal positions within the video [7]. For this reason fusing modalities with synchronized output may not capture this kind of dynamics. To this end, authors propose fusing inputs within a Temporal Binding Window (TBW), by allowing asynchronous inputs from the different modalities withing the TBW, implementing a mid-level fusion approach. The concept of TBW is inspired by the evidence in neuroscience in how humans perceive the world across different perceptual stimuli. More formally, using the same notation of equation 3.4 restricting the notation to the case of two modalities, synchronous fusion approaches like [54] perform classification using a feature extractor that produces a representation for each time step j and a segmental consensus function that perform temporal aggregation over all time steps.

$$SyncNet(T_{m_1}, T_{m_2}) = \mathcal{H}(\mathcal{G}(\mathcal{F}_{sync}(T_{k,m_1}, T_{k,m_2}; \mathbf{W}))) \quad k = \left\lceil j\frac{r_2}{r_1} \right\rceil \tag{3.7}$$

This approach also need to take into account modalities with not matching frame rates, by sampling at time steps multiple of all modalities frame rate, therefore approximating an exact synchronous fusion. TBN in contrast fuses modalities within a range of temporal offset, with the constraint of being contained within a finite time window (TBW):

$$TBN(T_{m_1}, T_{m_2}) = \mathcal{H}(\mathcal{G}(\mathcal{F}_{tbw}(T_{k,m_1}, T_{k,m_2}; \mathbf{W}))) \quad k \in \left[\left\lceil \frac{r_2}{r_1} - b \right\rceil, \left\lceil \frac{r_2}{r_1} + b \right\rceil \right] \tag{3.8}$$

where $\mathcal{F}_{tbw}$ is a multi-modal feature extractor that combines inputs within a binding window of width $\pm b$. According to authors, this sampling within the temporal window:

- enables straightforward scaling to multiple modalities with different frame rates;

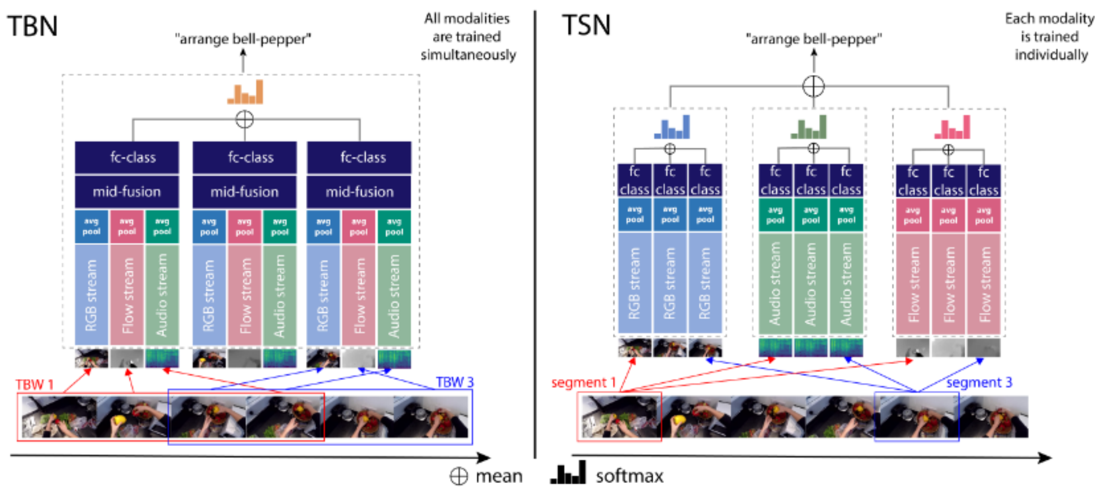


Figure 3.3. Architectural differences between TSN and TBN: in TBN modalities are fused within a window and following a mid-fusion approach, then the score logits are averaged. In TSN instead each modality is trained individually and then score logits are averaged in a late-fusion fashion. Image taken from [7]

- allows training with a variety of temporal shifts, accommodating, say, different speeds of action performance;
- provides a natural form of data augmentation.

Inside each TBW, a per-modality ConvNet extracts mid-level features, that then fused by concatenation, and then fed to a fully-connected layer, making multi-modal predictions per TBW. All the predictions of multi-modal representations are then aggregated for video-level predictions. The size of the temporal window b is variable, specifically calculated relative to the action video length, but can be set independently of the number of segments the actions is divided into, allowing for overlapping temporal windows.

3.4.3 Inflated 3D (I3D) Convolutional Network

Authors of [8] have taken a rather different approach towards video modelling, by directly modifying the structure of convolutional layers, encoding the temporal dynamics as an additional dimension in kernels. As such, I3D builds on existing image classification architectures, but inflates their filters

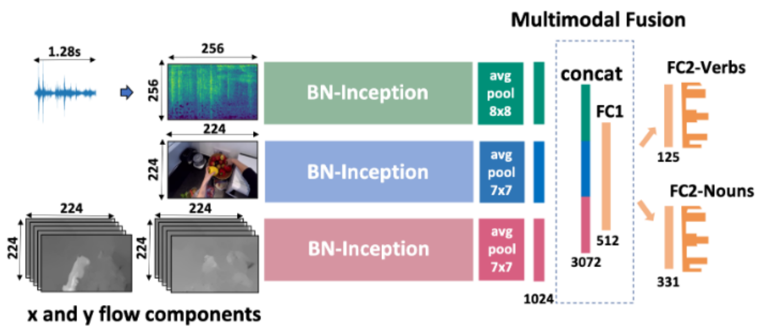


Figure 3.4. Illustration of a single TBN block. Image taken from [7]

and pooling kernels (and optionally their parameters) into three dimensions, resulting in deep spatio-temporal classifiers. Since in image classification relying on ConvNets pretrained on ImageNet has been proved an effective approach, the authors of [8] provide a way of bootstrapping 3D filters from 2D filters. Note that inflating a 2D model into a 3D one presents notable advantages with respect to directly using a 3D ConvNet: they do create hierarchical representation of spatio-temporal data, but the resulting model has many more parameters than 2D ConvNets, and they seem to preclude the benefit of ImageNet pretraining. For these reasons, previous works have defined shallower architectures retrained from scratch, but results were not competitive again the state of the art.

On the contrary, an I3D model share the parameters for the inflated dimension, so can benefit from an (implicit) ImageNet pretraining. To this extend, by noting that an image can be converted into a video by copying it repeatedly into a video sequence, 3D models can be implicitly pretrained on ImageNet by satisfying what they call "the boring-video fixed point": the pooled activations on a boring video should be the same as on the original single-image input. This is achieved by repeating the weights of the 2D filters N times along the time dimension, and rescaling them by dividing by N. In addition to the RGB stream, the authors found valuable to keep the two-stream architecture and use also Optical Flow using another I3D network, perhaps because the flow keeps a recurrent information that the RGB stream lacks.

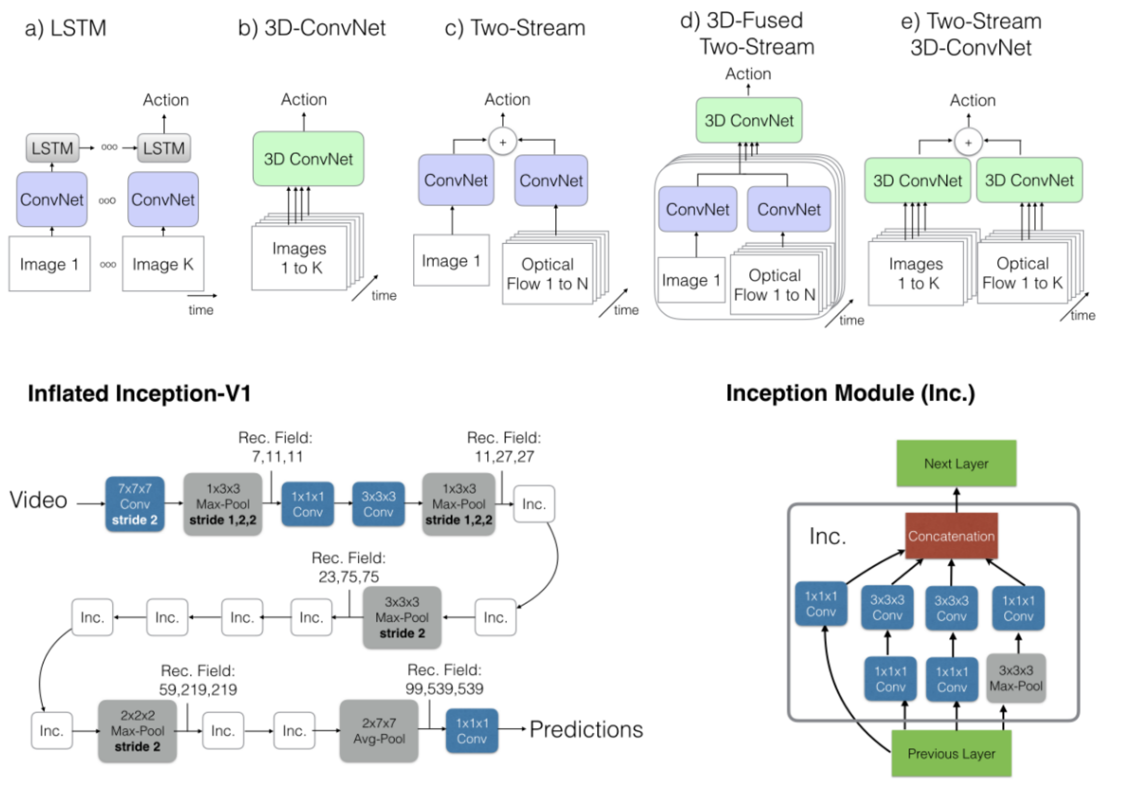


Figure 3.5. Top: Video architectures considered in this paper, where K is the total number of frames in a video and N is the subset of neighboring frames of the video. Bottom: from left to right, the Inflated Inception-V1 architecture and its detailed inception submodule. Images taken from [8]

3.4.4 Temporal Shift Module (TSM)

Altought 3D CNNs can jointly learn spatial and temporal features, their computational cost is large, and this makes their deployment on edge devices difficult, since they cannot be applied to real-time video recognition. On the opposite, 2D CNNs operate independently over the time dimension, so they lack proper temporal modelling but are computationally cheaper. The authors of TSM [9] propose a module that can achieve the performance of 3D CNNs mantaining 2D CNNs' complexity. The intuition behind the TSM method is the fact that the convolution operation consists of *shift* and *multiply-accumulate* operations. Taking as an example a convolution with kernel $W = (w_i, w_2, w_3)$ over an infinite 1D input X, the convolution operation can be written as $Y_i = w_1 X_{i-1} + w_2 X_i + w_3 X_{i+1}$. Hence it is possible to decouple the shift operation from the multiply-accumulate:

$$X_i^{-1} = X_{i-1}, \quad X_i^0 = X_i, \quad X_i^{+1} = X_{i+1}$$
$$Y = w_1 X^{-1} + w_2 X^0 + w_3 X^{+1}$$

The method of TSM consists in shifting the channels along the temporal dimension forward and backward by ± 1 (that computes the shift operation mentioned above), and computing the multiply-accumulate operation with a standard 2D convolutional layer. When applied in an online context, since frames at future time steps are not available, TSM performs the shift only backward. However, simply applying the shift to all or most of the channels there are negative effects with respect to the baseline (TSN):

- **Decreased efficiency due to significant data migration:** the shift, while requiring no computing, involves data movement, that increases the memory footprint and inference time on the hardware;
- **Performance deterioration due to a reduced ability to model spatially:** the relocation of a portion of the channels to neighboring frames causes the information contained in the channels to be inaccessible for the current frame, which may impair the 2D CNN spatial modeling capabilities;

The authors address the first issue by a partial shift approach (e.g. 1/8 of the channels), to reduce data movement, and the second issue by placing the shift module inside the residual branch in a residual block (residual shift) rather than inserting the module in-place, to balance the model's capacity for spatial and temporal feature learning.

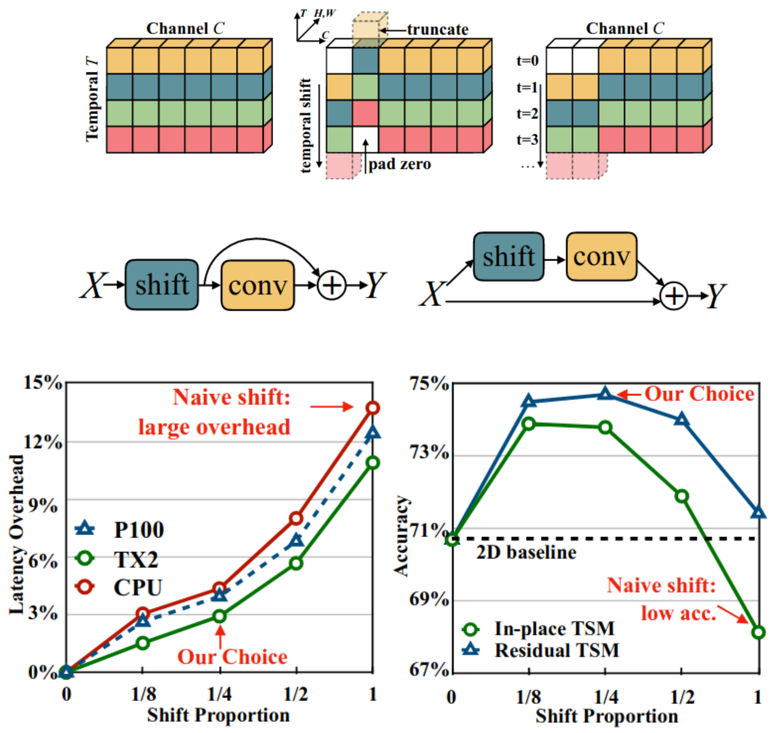


Figure 3.6. Top: the basic TSM block in its online and offline version; the conceptual structure of in-place TSM and residual TSM. Bottom: from left to right, latency overhead of TSM due to data movement; comparison of accuracy reached by in-place TSM and residual TSM. Images taken from [9]

3.5 Unsupervised Domain Adaptation (UDA)

As mentioned earlier, Unsupervised Domain Adaptation (UDA) refers to the task of adapting a model trained in a supervised manner in a domain to another, different domain without being provided with the class labels, so without being able to use the ground through in a supervised way. Different approaches present in the domain adaptations literature, more often in the field of image classification, can be grouped by the kind of techniques used:

- **self-supervised learning:** this kind of approaches often exploit the data used for the main classification task to generate a different task, that is believed to to enhance the generalizability of the network with respect to the main task, without requiring additional annotations. This auxiliary task in called *pretext task*, that exploits inherent data attributes to automatically generate surrogate labels: part of the knowledge of the input data is removed, and the learning task consists in recovering it. Possible pretext tasks are those that rely on original visual cues and involves geometric transformations (translation, scaling, rotation, ecc.), like [55]. More importantly, the pretext task is used to transfer its knowledge to the main I, for example supervised classification.

- **adversarial learning:** this kind of approaches aim at helping the features extractor of a deep architecture to extract features that are discriminative for the main learning (supervised) task on the source domain and invariant with respect to the shift between the source and target domain. The techniques usually consist in challenging the feature extractor with an adversarial task: [56, 57, 58] use a gradient reversal layer paired with a domain classifier. The parameters of the domain classifier are optimized in order to minimize their error on the training set, while the parameters of the underlying deep feature mapping are optimized for jointly minimizing the loss of the label classifier and maximizing the loss of the domain classifier.
- **data augmentation:** this kind of approaches work at data-level, exploiting variants of GANs [59] to synthesize new images or using an adversarial data augmentation procedure [60], with the aim of reducing the domain gap.
- **contrastive learning:** this kind of approaches aim at minimizing the distance between source and target domain in the feature space by pushing closer cross-domain inputs belonging to the same label class and pulling apart those that do not. In particular, given an anchor image from one domain, the objective is to minimize its distances to cross-domain samples from the same class relative to those from different categories, like in [61].
- **feature alignment strategies:** methods that belongs to this category focus on learning domain invariant data representations by minimizing different domain shift measures [62, 63, 64, 65], or on feature norm matching between source and target domains [66].

3.5.1 UDA in Action Recognition

With respect to unsupervised domain adaptation for images, videos are more challenging because of the temporal dynamics, and the problem remains largely unexplored. The state of the art methods can be however grouped

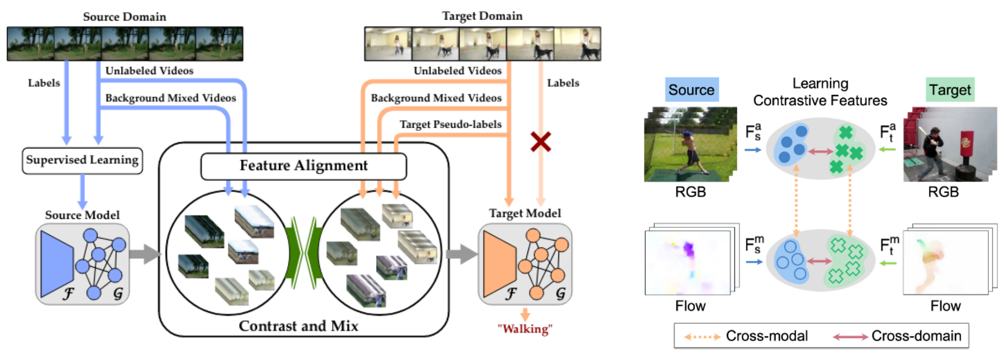


Figure 3.7. Overview of Comix approach (left) and the the contrastive cross-modal learning approach proposed in [10] (right). Images taken from [11, 10]

into the categories above outlined. CoMix [11] belongs to the contrastive learning framework, and aims at learning discriminative invariant feature representations by utilizing temporal contrastive learning to maximize the similarity between encoded representations of an unlabeled video at two different speeds as well as minimizing the similarity between different videos played at different speeds. The temporal contrastive objective is further enriched with background mixing of a video from one domain to a video from another domain, allowing additional positives examples per anchor. In addition, it also integrates a supervised contrastive learning objective, enhancing the discriminability of the latent space by using target pseudo-labels. This approach however does use multi-modal inputs, therefore not accounting for the multi-modal nature of action videos. MM-SADA [12] is one of the state of the art in multi-modal domain adaptation for action recognition: it uses a two-stream model that uses RGB frames and optical flow, leveraging adversarial and self-supervised approaches to extract domain invariant features and to learn modality correspondence. More in detail, the architecture employs per modality feature extractors, with domain classifiers trained in an adversarial manner like in [56], incorporating a self supervision alignment classifier that determines whether modalities are sampled from the same or different actions, finally merging the modality prediction scores following a late fusion scheme. Both the alignment techniques are trained on source and unlabeled target data, while the action classifier is only trained with labelled source data. A more recent approach [10] leverages a contrastive approach

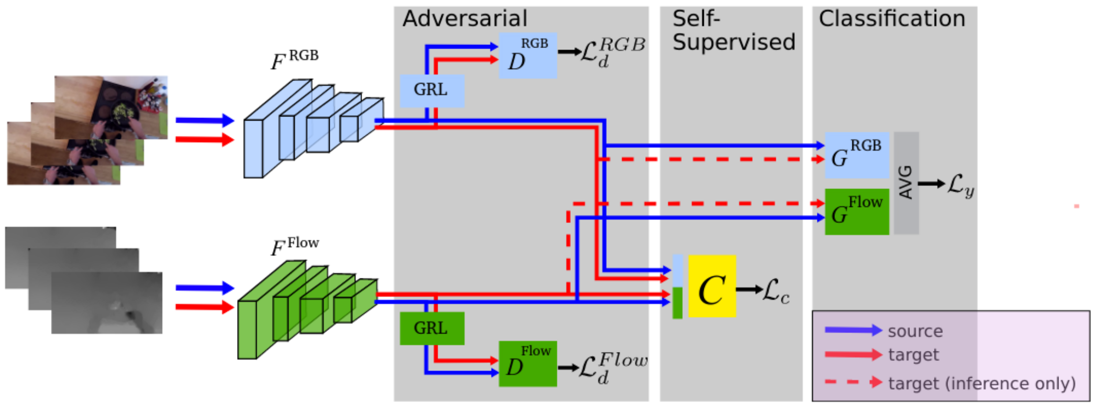


Figure 3.8. Proposed architecture of MM-SADA approach. Image taken from [12]

using multiple modalities of videos, by regularizing cross-modal and cross-domain feature representations, enriching the feature space with additional connection across modalities and better alignment across domains. More in detail, the cross-modal contrastive pushes together feature representation of the same video of different modalities and pulls apart features of different videos of different modalities, while the cross-domain loss (separately for each modality) pushes together the features of videos sharing the same action label for both domains and pulls apart the features of videos having different action labels, leveraging pseudo-labels for the target domain.

3.6 Domain Generalization

Domain Generalization (DG) refers to the task of learning a model from data belonging to one or more domains in a supervised manner, but at the same time having this model not to degenerate when tested with data coming from a different domain. With respect to Unsupervised Domain Adaptation, in a DG setting the target domain in unknown at training time: in particular the data distribution of the target domain in not available. This makes it a more challenging scenario, and some of the approaches used for UDA are therefore not applicable. For example, approaches like the ones proposed in [55, 60] can still be used, while other approaches need to use the some knowledge of the target domain. Among the methods for image DG, [67] aims at optimizing the feature space such that its semantic structure is insensitive to different training domains, and generalize better to new unseen domains. The approach proposed in [68] uses adversarial autoencoders to learn a generalized latent feature representation across domains for domain generalization, by imposing the Maximum Mean Discrepancy (MMD) measure to align the distributions among multiple seen source-domain. [69] uses a meta-learning approach to improve feature extractor training, and deliver a better model for both homogeneous and heterogeneous DG problems.

3.6.1 DG in First Action Recognition

The literature in DG for video processing is scarce, and in general presents additional challenges with respect to image DG. The key finding is that in videos the spatial and temporal domain shift coexist: while the domain shift caused by the variations of the appearance in video frames can be partially solved by image domain generalization methods, the temporal domain shift remains to be explored. Authors of [70] claim that the temporal shift is due to unexpected absence or misalignment of short-term video events (called also local temporal relations) across distant domains, so they propose to use the short-term video relations to generate adversarial examples to augment the source set, using an approach similar to [60], and then exploiting them along with the global-relation features to maintain the discriminability. Nevertheless such an approach do not consider the multi-modal nature of actions, lacking the ability to exploit additional perceptual cues.

Part II

Second part: Our Contribution

Chapter 4

Multi-Modal Domain Generalization via Relative Norm Alignment (RNA)

In this chapter we will describe our contribution to the research of domain generalization in multi-modal learning, focusing on application in first person action recognition. More concretely, we will describe the intuition behind the Relative Norm Alignment (RNA) when training a deep multi-modal feature extractor, and then present an implementation of the idea as a loss function. We will show how the approach is domain agnostic, meaning that it can be used as a off-the-shelf component for multi-modal learning, and how it can easily extended to the UDA scenario, achieving additional gain. We will discuss how adding a relative norm alignment affects the training, and why it can be interpreted as a regularization technique for multi-modal streams. Finally we show how it is possible to induce RNA to work in the higher norm feature space, and the rationale behind this choice.

4.1 Intuitions and motivations

4.1.1 Domain generalization and regularization

As stated earlier in this work in section 2.1.3, any machine learning is affected by what is called the bias-complexity tradeoff: there exist a relation between the inductive bias introduced in choosing the hypothesis class the learning algorithm will learn the predictor from and the training set. Specifically, this takes form in the estimation error, that is the difference between the error achieved by the chosen predictor $h_{\mathcal{S}}$ and the best possible predictor belonging to the hypothesis class, with respect to the underlying (and unknown) distribution from which training and test data are drawn. Using this kind of error decomposition, overfitting occurs when the estimation error is high.

In a practical scenario, estimating the ϵ_{app} from equation 2.6 is difficult, so another error decomposition is used:

$$L_{\mathcal{D}}(h_{\mathcal{S}}) = (L_{\mathcal{D}}(h_{\mathcal{S}}) - L_{\mathcal{V}}(h_{\mathcal{S}})) + (L_{V}(h_{\mathcal{S}}) - L_{\mathcal{S}}(h_{\mathcal{S}})) + L_{\mathcal{S}}(h_{\mathcal{S}}) \tag{4.1}$$

where

- the first term is bounded by $\sqrt{\frac{\log(2|H|/\delta)}{2m_V}}$ with probability $(1-\delta)$ over the choice of m_V samples of validation set, sampled independently of H;
- the second term is the estimate of the estimation error $\hat{\epsilon}_{est}$: if large, the algorithm suffers from overfitting;
- the third term is the estimation of the approximation error $\hat{\epsilon}_{app}$: if large, the algorithm suffers from undefitting. In case it is small and we known that the hypothesis class is powerful enough, this means that it is a good estimate of $L_{\mathcal{D}}(h^*)$, having defined $\epsilon_{est} = L_{\mathcal{D}}(h_{\mathcal{S}}) - L_{\mathcal{D}}(h^*)$, where $h^* \in \arg\min_{h\in\mathcal{H}} L_{\mathcal{D}}(h)$.

However, this scheme refers to the standard setting in statistical learning, namely the fact that training and test set are drawn from the same underlying data distribution, so that the empirical error is a proxy for the true error that the predictor will achieve when tested with new data. In domain adaptation, the objective is to learn a predictor that not only considers a source training set made of labelled data, but also a target domain for which labels are not available. A rigorous model of domain adaptation has been formalized by [62], by bounding the error with respect to the target distribution with a measure of divergence between them, in fact for any hypothesis $h, h' \in H$:

$$|L_{\mathcal{S}}(h, h') - L_{\mathcal{T}}(h, h')| \leq \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{T}}) \tag{4.2}$$

where

$$\begin{aligned} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{T}}) &= 2 \sup_{h,h' \in \mathcal{H}} |\mathbb{P}_{x \sim \mathcal{D}_{\mathcal{S}}}[h(x) \neq h'(x)] - \mathbb{P}_{x \sim \mathcal{D}_{\mathcal{T}}}[h(x) \neq h'(x)]| \\ &= 2 \sup_{h,h' \in \mathcal{H}} |L_{\mathcal{S}}(h, h') - L_{\mathcal{T}}(h, h')| \\ &\geq 2|L_{\mathcal{S}}(h, h') - L_{\mathcal{T}}(h, h')| \end{aligned} \tag{4.3}$$

In domain generalization this is not applicable, since any statistics on the target domain is unavailable. So in our setup we seek to find a predictor that "generalize enough" to take into account samples drawn from a distribution different from, but somehow linked to, the one seen in training. To better understand the problem, we can rephrase the problem of generalization into an overfitting problem. In a classical setup, overfitting is detected when, during training, the error on the training set keeps decreasing, while the error on a validation set starts increasing. Most notably, assuming that the hypothesis class the algorithm can learn the predictor from to is large enough, observing a very low training error and high validation error means that we have two choices:

- *getting more examples:* recalling the standard assumption of the statistical learning by which the training set is drawn i.i.d. from the a data distribution $\mathcal{D}$, having a larger training sample means capturing more information about the distribution so the training error increases, intuitively because it becomes more difficult for a predictor to "explain" the diversity of the instances;

- *reducing the complexity of the hypothesis class $\mathcal{H}$:* it practically means decreasing the number of free parameters, and this is commonly achieved with the help of regularization techniques.

However to frame the problem of generalization as overfitting with respect to an unknown distribution, it is not possible to only look at the common validation error: the validation set would be drawn from the same distribution as the source domain, so we would not notice a drop in the generalization performance of the algorithm. To this extend, it is possible to refer to the validation set of the target domain, and define the "lack of generalization" as "overfitting on the source domain", with respect to the target. In this context the first choice above corresponds to the ability to access the target domain, which is not possible as per the definition of domain generalization, so the other way is to regularize the learning objective, such that we restrict the hypothesis class in such a way we still preserve those predictors that are optimal for the task. Indeed one interpretations on our approach is as a (multi-modal) regularization technique: we constrain the learning algorithm to search for a predictor that, other than minimizing the empirical error on the source domain training set, satisfies the *relative norm alignment* constraint, that we will explain the section 4.2.

4.1.2 Multi-Modal optimization

As highlighted in section 3.3.2, training multi-modal networks is harder than training the uni-modal counterparts. One of the identified reasons is overfitting: a multi-modal has usually more free parameters than a uni-modal one; in fact, because of inputs multi-modal inputs have different nature, usually architectures use at least a dedicated feature extractor per modality, and sometimes even more (e.g. those one using late-fusion approaches). Another issue, not related to architectural choices but rather related to the nature of data of different modalities, is due to the fact that different modalities separately overfit and generalize at different rates, so the joint optimization of the related branches is sub-optimal. However, architectures that do not use joint training by using pre-trained uni-modal features will most likely have sub optimal performances, due to the fact that the inter-modality correlations are not captured, thus missing to fully exploit data complementarity. To this extend, authors of [53] have proposed a metric to quantitatively describe the problem of overfitting of multi-modal networks, namely the overfitting-to-generalization ratio (OGR): their Gradient-Blending solution aims at finding optimal weights for the loss functions associated to the single modalities, to re-weight them into a multi-modal loss that ensures an equal rate of overfitting and generalization of all modalities. Here we spot the same problem from a different perspective: because different modalities overfit and generalize at different rates, it could happen that the easier-to-converge network, associated with the modality containing the most of the informative content, became predominant for the classification task: this means that the classifier will mostly rely on the corresponding perceptual input, not exploiting the complementary information from other modalities. The amount of information content with respect to other modalities is given by their mean feature norm: a strong correlation between the mean feature norms and the informative content for classification has been noted in [71, 72, 73]. In particular, in [72] authors show that the cross-entropy loss promotes well-separated features with a high norm value. The Smaller-Norm-Less-Informative assumption used in [74] implies moreover that a modality representation with a smaller norm is less informative during inference. Based on these findings, we conjecture that a norm imbalance between different modalities affects the classifier to prefer the higher mean feature norm modality, affecting the performance of smaller mean feature norm modalities as well as the global classification task, that misses important cues from other modalities, leading to a less robust, so less generalizable model. We believe that the relative norm alignment approach enjoys both the aforementioned advantages: it ensures an equal rate of learning from all modalities and constraints the learning algorithm such that the model extract features with rebalanced mean norm, with clear improvement in taking advantage of data complementarity.

4.2 Proposed approach

To constrain the different features to be norm-aligned we introduce a new task for the feature extractors of the different modalities. Specifically we want the network to learn how extract features whose expected values is the same across modalities because it is in doing do that the network is constrained to leverage multi-modal norm correlations. To do so, our approach is based on a loss function that penalizes a norm misalignment between feature norms, applied to the features out of the deep feature extractor, before modality fusion. Formally we denote $h(x_i^m) \overset{\text{def}}{=} (||\cdot||_2 \odot f^m)(x_i^m)$ as the L_2-norm of the features f^m of the m-th modality, and define the mean-feature-norm distance between two modality norms f^{m_i} and f^{m_j} as:

$$\delta(h(x_k^{m_i}), h(x_k^{m_j})) \overset{\text{def}}{=} |\mathbb{E}[h(X^{m_i})] - \mathbb{E}[h(X^{m_j})]| \tag{4.4}$$

To minimize the δ distance of all the modalities we propose the Relative Norm Alignment (RNA) loss. Let it be $M = \{m_1, ..., m_m\}$ the set of available modalities, ordered by their mean feature norm $\mathbb{E}[h(X^{m_i})]$, then:

$$\mathcal{L}_{RNA} \overset{\text{def}}{=} \sum_{i<j} \left(\frac{\mathbb{E}[h(X^{m_i})]}{\mathbb{E}[h(X^{m_j})]} - 1 \right)^2 \tag{4.5}$$

where the expected value of the feature norm can be approximated by its unbiased estimator, the mean over the batch: $\mathbb{E}[h(X^{m_i})] = \frac{1}{B}\sum_{x_i^m \in \mathcal{X}^m} h(x_i^m)$ for the m-th modality and B is the batch size. Finally, the total loss function optimized by the learning algorithm is:

$$\mathcal{L} = \mathcal{L}_{cross-entropy} + \lambda \mathcal{L}_{RNA} \tag{4.6}$$

4.2.1 Discussion and variations

The rationale behind the norm alignment is to force the network to full exploit the features of various modality; the advantage is twofold:

- Imposing this constraint to the learning algorithm means restricting the hypothesis class $\mathcal{H}$ to the subset of hypotheses that extract equally informative features from different modalities, relying on the evidence that a modality representation with a smaller norm is less informative during inference;

- The part of the network after the feature extractor has the chance to learn to work in the normalized feature space during training.

For these reasons a static normalization, without any learning objective would not achieve the same result; there are two points in the architecture stack in which one could place a normalization operator that would achieve a norm balancing: at input level or at feature level. A normalization at input level simply won't work: it is not feasible in a DG context because of missing information about target domain statistics and would be not suitable with the use of pretrained models. On the other hand, simply normalizing the feature norms just after their extraction does not work equally well, because only the second aspect is achieved: there could be an advantage just for the classifier to be presented with normalized features, but there would be no alignment of learning from multiple modalities and no regularization of individual streams. Besides these downsides, static feature re-balance would contrast with the Smaller-Norm-Less-Informative assumption: the higher norm of a feature would not result from having extracted additional informative content, but as a result of an affine transformation. Moreover, we believe that a learning induced re-balance leverages networks' non-linearities to extract features in a embedding space where they are more discriminative and yet more generalizable. For this reasons it is crucial that the network learns how to re-balance the feature norms by itself.

Not only it is important to learn how to re-balance feature norms of different modalities, but also how to induce this learning objective. More specifically, the learning objective in equation 4.5 mathematically requires the ratio of the feature norms to be equal to one, that means that there must an optimal feature norm value, that RNA loss achieves as consensus on the ratio. The existence of such value would mean that an alternative and equal performing loss formulation satisfying equation 4.4 is:

$$\mathcal{L}_{HNA} \overset{\text{def}}{=} \sum_{m_i \in M} (\mathbb{E}[h(X^{m_i})] - k)^2 \tag{4.7}$$

being k the optimal value just mentioned. This formulation however have a few downsides:

- The optimal value k is unknown, it depends on the data and on the network, so it becomes an hyperparameter difficult to tune properly;
- Being k fixed, $\mathbb{E}[h(X^{m_i})]$ for some $m_i \in M$ can be far away from it, especially at the beginning of the training. This means a potentially high loss value, to be counteract with a proper weight for the loss, that for this reason is a hyperparameter difficult to tune;
- Modalities do not explicitly interact: applying equation 4.7 to the joint training is equal to apply it to the single stream network corresponding to the modality m_i, being k fixed and not dependent on the observed batch.

An alternative valid formulation of the constraint in equation equation 4.4 is the following:

$$\mathcal{L}_{RNA}^{sub} \overset{\text{def}}{=} (\mathbb{E}[h(X^{m_i})] - \mathbb{E}[h(X^{m_j})])^2 \tag{4.8}$$

However this formulation has some of the problems mentioned earlier: high discrepancy in mean feature norm corresponding to different modalities would reflect in higher loss value, requiring a careful tuning of the loss weight, with consequent sensitivity on that value. The structure of equation 4.5, other than inducing an optimal equilibrium between the two embeddings, pushes the network to take larger steps when the ratio of the two modality norms is too far from one, resulting in faster convergence.

4.2.2 Extension to UDA scenario

Under the UDA setting it is possible to take advantage of unlabelled target data, so RNA loss can be applied also to target data. More formally, given the input data from source and target domains respectively denoted as X_S and X_T, it is possible to extend the definition in equation 4.4 also to target data instances $x_{s,i} \in X_S$ and $x_{t,i} \in X_T$ and use the same formulation in equation 4.5 to apply RNA loss to source and target samples, resulting in:

$$\mathcal{L}_{RNA} = \mathcal{L}^s_{RNA} + \mathcal{L}^t_{RNA} \tag{4.9}$$

Applying RNA loss also on target data gives the network the opportunity to learn how the information content is distributed on the different modalities for the target distribution, by means of a different norm unbalance. The output predictor is hence able to account for a variety of unbalance and, because seeing also target means that the "learn to re-balance" task is harder, it has a stronger regularization effect.

4.3 Further variations

The loss formulation introduced in equation 4.5 adaptively incrementally re-balances the mean feature norm associated with features of multiple modality, reaching a consensus among them. However, sometimes it is possible that one of the network accomplishes this task more easily than the others: the scenario we are in is domain agnostic, features can come from input of very diverse nature and in principle each modality employs the feature extractor works best with it. In such a complex case, it is possible that the problem of a network to learn faster than the other can reoccur for the RNA task, and this can result in networks associated with modalities with lower feature norms to move the common feature norm at convergence to a lower value. Recalling that larger norm convey more information, it is desirable to avoid such decrease of the consensus mean feature norm. To this extend, it is possible to combine RNA loss with the Stepwise Adaptive Feature Norm approach of [66], that encourages a feature norm enlargement at the step size of Δr with respect to individual examples, based on their feature norms calculated by the past model parameters in the last iteration. In particular, since the introduction of this loss would contrast with RNA and would encourage an unbounded increase to the mean feature norms, we applied this loss only to the features associated with the lowest mean norm, doing so encouraging a rebalance towards an higher mean feature norm. More formally, given

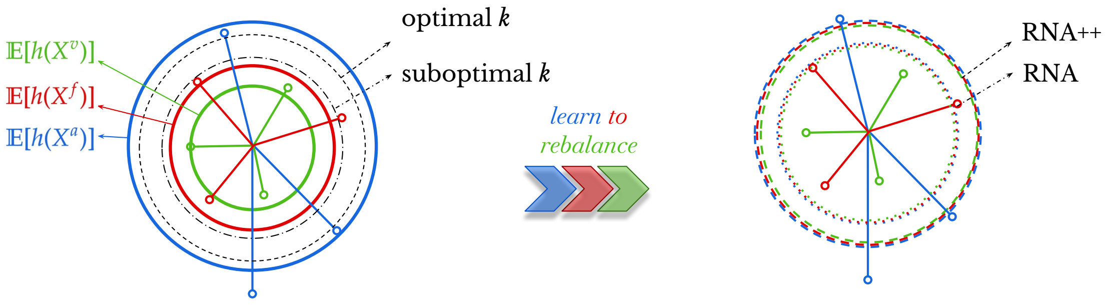


Figure 4.1. RNA-Net++ induces the network not only to rebalance mean feature norms of different modalities, but also to push the consensus mean feature norm towards higher norm feature space.

$M = \{m_1, ..., m_m\}$ the set of available modalities, ordered by their mean feature norm $\mathbb{E}[h(X^{m_i})]$, the $SAFN_{min}$ loss is defined as:

$$\mathcal{L}_{SAFN_{min}} \overset{\text{def}}{=} \frac{1}{B}\sum_{i=1}^{B}(h(x_i^{m_1}) - \Delta r)^2 \tag{4.10}$$

As result of our experiments, we found that RNA benefits from this integration, showing superior performance over applying RNA or $SAFN_{min}$ losses alone.

Chapter 5

Experimental Results

In this chapter we will present the experimental results that confirm the intuition behind the relative norm alignment approach described in chapter 4. The first part of the chapter presents the main results, starting from the approach applied to EK-55 dataset, demonstrating its effectiveness against the state of the art, although in a simplified scenario, where the actions are less fine-grained and only three domain shifts are considered. From these results, we extend the approach porting it to the more complex scenario of the EK-100 Unsupervised Domain Adaptation challenge described in section 3.2.1. The second part of the chapter consist in an ablation study on the algorithmic and architectural choices that lead us to the final formulation of proposed method: in particular we will compare the *learn to re-balance* task to a simple feature alignment, demonstrating that the value of the method in primarily in the induced task. Finally we will evaluate the robustness of our approach against the state of the art, demonstrating that the RNA-Net++ is better at optimizing with respect to a single modality even when trained with multi-modal input.

5.1 Experimental setting

5.1.1 Dataset

Along the experiment we used two datasets: the EK-55 dataset in the first part, where we show that RNA is a promising multi-modal learning method domain generalization in first person action recognition. From those results, we consider extending it to a more complex scenario using the EK-100 dataset, and competing for the corresponding Unsupervised Domain Adaptation challenge.

EK-55

State of the art algorithms compare themselves against top-1 verb classification in two different settings: *cross-domain*, meaning that the training set and the test set belong to different domains, so different kitchens, and *supervised*, meaning that training set and test set belong to the same domain, so the same underlying data distribution. In practice, among the 32 kitchens (considered as different domains), the three biggest are selected: kitchen P08 is referred as D1, kitchen P01 as D2 and kitchen P22 as D3. Given these tree subset of the dataset, 9 combinations can be defined: 6 of them are referred as cross-domain, while the remaining 3 describe a supervised setting. For example the wording $D_i \rightarrow D_j$ referred to a model means that D_i is treated as source domain and D_j as target domain. In UDA context, in means that the training set of D_i has been used to train the model in a supervised way, while the training set of D_j has been used in an unsupervised way during the training: finally the model's performance are evaluated on the test set of D_j.

EK-100

The aforementioned setting, however challenging, is simplified and unrealistic: performance are measured only against verb prediction, the considered domains are few and the label space is relatively small, in fact the prediction is made among only 8 verbs. The first version of the EPIC-KITCHENS dataset was released in 2018: from its release it has been successful and many researchers took up the various challenges it enables. As result of it success, two years later EK-100 was released, a collection of 100 hours, 20M frames, 90K actions in 700 variable-length videos, capturing long-term unscripted activities in 45 environments. Along with the dataset, authors of [17] proposed the Unsupervised Domain Adaptation challenge (see section 3.2.1). The training sets are additionally split into source and target: the difference lies in the fact the target training set in not annotated, so it is used to perform domain adaptation with respect to the test set domain. Specifically, the splits provided as source and target corresponds to the same participants of EK-55 dataset, but the target splits are the same actions recorded two years later.

The scenario of the UDA challenge is a much more challenging test case: performances are evaluated in terms of top-1/5 test accuracy on verb, noun and action prediction, on 16 different domains. Importantly the verb label and noun label spaces are composed respectively by 94 and 300 different values, making the action prediction task much more fine-grained. Additionally, the dataset presents variabilities due to changes in locations, viewpoints, labels and participants (the behaviour of the same person can change over time), but also differences in data recording methodology (the hardware has changed) or changes in the environment over time (some participants have moved house).

5.1.2 Implementation details

Experiments on EK-55

For the experiments on EK-55, the network is composed of two streams, one for each modality m, with distinct feature extractor F_m and classifier G_m . The RGB stream uses I3D [8] as done in [12], while the audio feature extractor uses the BN-Inception model pretrained on ImageNet, which proved to be a reliable backbone for the processing of audio spectrograms [7]. Each feature extractor produces a 1024-dimensional representation f_m, then fed to the classifier G_m. The modalities are fused following a late-fusion approach by summing the score logits, and the cross entropy loss is used to train the network. We compare our approach considering the base loss formulation and the complete method in DA, made by the RNA loss, the GRL and attentive entropy, that we refer to as RNA-Net.

The network is trained for 9k iterations using the SGD optimizer. The learning rate for RGB is set to $1e-3$ and reduced to $2e-4$ at step $3k$, while for audio, the learning rate is set to $1e-3$ and decremented by a factor of 10 at steps $\{1000, 2000, 3000\}$. The batch size is set to 128, and the weight λ of $\mathcal{L}_{RNA}$ (see equation 4.5) is set to 1.

Experiments on EK-100

The EK-100 UDA challenge is based on pre-extracted features publicly released and available to contestants. In practice, this data come from applying state of the art architectures (like the ones described in this work in section 3.4) to different modality streams of the dataset, to extract single modality features, and the task of contestants is to further process these features to enhance the performance of the base model. Feature splits follow the same logic of the dataset's ones: source and target train and validation sets are provided with labels, while target test data are provided unlabelled. We used the officially provided extracted features from TBN and TSM (see section 3.4.2 and 3.4.4), comparing the two best temporal aggregation approaches: the one referred as TBN consist in an average pool of temporal features, the one referred as TBN-TRN uses the same features but using the temporal aggregation method described in [75]. On those architectures, we compare different methods: source-only refers to no domain adaptation strategy adopted; TA^3N refers to the approach proposed in [76]. Our approach builds upon TA^3N by adding an extra SE [77] layer per modality, that performs feature norm alignment of multi-modal stream. We further propose to extend the approach by pushing the norm alignment towards higher feature norms region, using the formulation in equation 4.10: we refer to such formulation as RNA-Net++.

The weight for RNA-Net has been searched in {0.1, 0.5, 1, 2, 3, 4, 5, 10, 15, 20} when using the SE layer [77] or SMR [78] for norm alignment and in {0.05, 0.1, 1, 2, 3, 4, 5} when using a simple linear layer, and $\lambda = 10$ and $\lambda = 0.1$ have been chosen respectively. Additionally, for RNA-Net++ the search additionally involved the two hyperparameters for equation 4.10, namely $\Delta r \in \{0.3, 0.5, 1\}$ and its weight in the final loss formulation $\lambda_{SAFN_{min}} \in \{0.1, 0.5, 1, 2, 3\}$: the chosen values are finally $\lambda_{SAFN_{min}} = 2$ and $\Delta r = 1$.

5.1.3 Results section structure

In this section we describe how we compare our approach to the state of the art methods and how we validate our choices. In section 5.2 we present our main results both on EK-55 and on the setting of the EK-100 UDA challenge. More in detail, in section 5.2.1 we propose comparisons both in DG and UDA scenario, grouping the algorithms we compare our method to in: *image-based* domain adaptation and *multi-modal* approaches. As common practice in the literature, each method is compared against a baseline consisting of the same backbone with no domain adaptation strategy adopted. In section 5.2.2 we present the results of our approach across several combinations of modalities, in the setting of EK-100 UDA challenge, comparing each method with the two best performing architectures. Section 5.3 explores more in depth some aspects of our approach: in section 5.3.1 we compare the various architectures across different modality combinations, while in section 5.3.2 we evaluate the choice of the additional architectural part added to perform norm-alignment, across the best architectures. Moreover, to prove the point of section 4.2.1 about the contribution of a mere feature normalization, in section 5.3.3 we compare the performance of RNA with an approach that statically (e.g. without a learning task) aligns the mean norm of modality features. In particular, we define a *Static Norm Alignment* (SNA) as a input level re-normalization on pre-trained input features as follows:

$$SNA = \frac{X^{m_i}}{||X^{m_i}||^2} \cdot k \tag{5.1}$$

where k is the final value of the feature norms of X^m. The optimal value of k is unknown, so in the experiment we searched it into three values: the feature norm corresponding to the modality having respectively the minimum or maximum norm or the average of all modalities' feature norms. Hence we define:

$$\begin{aligned} SNA_{min} &= \frac{X^{m_i}}{||X^{m_i}||^2} \cdot \min_{m_j} ||X^{m_j}||^2 \quad \forall m_i \\ SNA_{max} &= \frac{X^{m_i}}{||X^{m_i}||^2} \cdot \max_{m_j} ||X^{m_j}||^2 \quad \forall m_i \\ SNA_{avg} &= \frac{X^{m_i}}{||X^{m_i}||^2} \cdot \frac{1}{M} \sum_{j=1}^{M} ||X^{m_j}||^2 \quad \forall m_i \end{aligned} \tag{5.2}$$

To end the ablation study, in section 5.3.4 we propose an experiment aimed at demonstrating that our approach leads to a more robust model, in fact it less suffers from a perturbation of input consisting in a missing modality at test-time.

5.2 Main Results

5.2.1 EK-55

All the results reported in this section refer to a multi-modal scenario in which RGB and Audio modalities are used. Table 5.1 refers to a multi-DG scenario, in which more than one domain is used as source domain, while the target is always one different domain at a time. We refer to the baseline as *Deep All*, that is when the backbone architecture is used without employing other domain adaptive strategies, and all the source domains are fed to the network. We compared our approach with methods of different type: IBN-Net and Gradient Blending belong to image-based domain generalization approaches, while MM-SADA is a multi-modal UDA approach, but we adapted here to the DG scenario by retaining only the self-supervised task so (refer to section 3.5.1 for details about MM-SADA). The second group in table 5.1 refer to methods that exploit multi-modality of input data. Results show that RNA outperforms the mentioned approaches by a large margin, achieving a +6.4% accuracy with respect to the baseline.

Table 5.2 shows comparison with popular domain adaptation methods in UDA scenario: GRL challenges the feature extractor with an adversarial task to extract domain invariant features; MMD is based on minimizing a discrepancy measure between distributions; and AdaBN consists in modulating the statistics from the source domain to the target domain in the batchNorm layers across the network. It is possible to see that adopting our approach alone, performance are similar to the ones of the complete MM-SADA method, while our complete DA method gives additional performance gain, making RNA-Net the best in class method.

Method	D2, D3 $\rightarrow$ D1	D3, D1 $\rightarrow$ D2	D1, D2 $\rightarrow$ D3	Mean
DeepAll	43.19	39.35	51.47	44.67
IBN-Net	44.46	49.21	48.97	47.55
MM-SADA (Only SS) [12]	39.79	52.73	51.87	48.13
Gradient Blending [53]	41.97	48.80	51.43	47.27
TBN [7]	42.35	47.45	49.20	46.33
Transformer [79]	42.78	47.38	51.79	47.32
Cross-Modal Transformer [80]	40.87	43.57	54.88	46.44
SE [77]	42.82	42.81	51.07	45.56
Non-Local [81]	45.72	43.08	49.49	46.10
RNA loss	45.65	51.64	55.88	**51.06**

Table 5.1. Top-1 accuracy (%) of RNA in Multi Source DG scenario, compared to other methods of the state of the art

5.2.2 EK-100

The results reported in this section refer to the setting of the EK-100 UDA challenge. As previously anticipated, this setting consist in 16 different domains, corresponding to participants, and the shift between source and target domains consist in having recorded the actions of the same participants two years later.

Table 5.3 shows the performance of RNA-Net++ against the state of the

Method	Mean
Source-Only	41.87
GRL [56]	43.67
MMD [63]	44.86
AdaBN [82]	41.92
MM-SADA (only SS) [12]	46.44
RNA loss	47.71
MM-SADA (SS+GRL) [12]	47.75
RNA-Net	**48.30**

Table 5.2. Top-1 accuracy (%) of RNA-Net in UDA scenario, compared to other methods of the state of the art

art methods, across various combination of the three modalities. We compare each method against its *source-only* approach (that is training on source and testing directly on target data). For RGB-Flow modality combination, we report also the results using the pre-extracted features from TSM. In this case we note more moderate gains from using the Temporal Attentive Alignment strategy of TA^3N: this could be motivated by the fact that TSM more explicitly takes into consideration the encoding of the temporal dimension. Overall, results indicates that our approach consistently outperforms the other methods across different modalities and architectures combinations. In particular, we obtain a +1.62% (TBN) and +1.52% (TBN-TRN) with respect to the source-only in the most complete scenario where all the three modalities are used. This shows that the temporal attentive alignment as well as the methods used to fuse frame level features into video level ones are complementary to the norm alignment objective.

Modality	Method	Architecture	Top-1 Accuracy(%)		
			verb	noun	action
R+F+A	source only	TBN	47.10	28.30	18.66
	DAAA [83]	TBN	47.96	29.08	19.19
	source only	TBN	47.11	27.95	18.44
	TA^3N	TBN	47.22	29.04	18.94
	RNA-Net++	TBN	48.65	30.04	**20.06**
	source only	TBN-TRN	46.58	27.97	19.09
	TA^3N	TBN-TRN	47.48	28.35	19.25
	RNA-Net++	TBN-TRN	49.37	29.51	**20.61**
R+F	source only	TBN	41.51	25.13	14.27
	TA^3N	TBN	43.03	28.10	16.36
	RNA-Net++	TBN	43.49	28.18	**16.64**
	source only	TBN-TRN	43.79	26.36	16.34
	TA^3N	TBN-TRN	44.56	27.15	17.25
	RNA-Net++	TBN-TRN	45.80	27.53	**17.76**
	source only	TSM	48.49	28.19	18.32
	TA^3N	TSM	48.63	28.63	18.44
	RNA-Net++	TSM	48.56	28.56	**18.54**
	source only	TSM-TRN	47.16	26.97	17.73
	TA^3N	TSM-TRN	48.28	27.44	18.34
	RNA-Net++	TSM-TRN	47.63	27.59	**18.41**
R+A	source only	TBN	40.64	24.80	14.84
	TA^3N	TBN	40.72	25.75	15.18
	RNA-Net++	TBN	41.70	26.20	**15.91**
	source only	TBN-TRN	40.61	24.47	15.31
	TA^3N	TBN-TRN	40.56	25.03	15.74
	RNA-Net++	TBN-TRN	41.93	25.42	**16.45**

Table 5.3. Top-1 (%) of RNA-Net in UDA scenario on EK-100, compared to the source-only

5.3 Ablation study

In this section we analyze different aspects of our method in the context of EK-100 UDA challenge.

5.3.1 Comparing temporal aggregation methods

We started from an initial study about temporal aggregation methods: in the base architecture, 5 frames per action video are sampled for different modalities, then the corresponding features are concatenated along the last dimension to form multi-modal features. After being fed to a linear layer, the features are temporally aggregated. Table 5.4 shows the results for different architectures: TBN refers to the standard average pooling of temporal features, TBN-TRN uses the temporal aggregation described in [75], APN use a multi-level adversarial pyramid network of attention blocks and RNN uses a recurrent neural network. From these results, we selected the two best performing architectures, namely TBN and TBN-TRN, to carry out our study.

Modality	Architecture	Top-1 Accuracy(%)		
		verb	noun	action
R+F+A	TBN	47.22	29.04	<u>18.94</u>
	TBN-TRN	47.48	28.35	**19.25**
	APN	46.15	26.74	17.95
	RNN	46.43	23.61	16.03
R+F	TBN	43.03	28.10	<u>16.36</u>
	TBN-TRN	44.56	27.15	**17.25**
	APN	42.78	25.52	15.84
	RNN	41.46	22.32	13.13
R+A	TBN	40.72	25.75	<u>15.18</u>
	TBN-TRN	40.56	25.03	**15.74**
	APN	39.76	23.80	14.88
	RNN	40.87	21.20	13.24

Table 5.4. Comparison of different temporal aggregation methods across different combination of input modalities in UDA context

5.3.2 Adding a feature interaction module

Our architectural changes refer to part of the network before the temporal aggregation module, in fact the loss formulation in 4.5 requires a network that accomplishes the norm alignment task. Most importantly, because in the base architecture multi-modal features are fed into a linear layer before the temporal aggregation, we cannot rely on tail components of the network to specifically take into account the activations of single modalities.

For this reason we evaluated different layers to be added to perform RNA. "Linear" refers to a simple linear layer per modality, SENet [77] consist in a squeeze-and-excitation module per modality, that interact by means of RNA, while SMR refers to the Semantic Mutual Refinement module described in [78], by which modality streams already interact via self-cross modal gating, in a way similar to [77]. When applying RNA, we consider the output features of the added layer after a dropout when using SENet or the simple linear layer, while when using SMR we apply the loss only features out of cross-gating, namely the modality features refined by transferable knowledge from other modalities.

As it is possible to see from table 5.5, the relative norm alignment always enhances the performances with respect to the base added layer, confirming that it is always beneficial to re-align modality norms. Adding a linear layer degrades the performance and even adding RNA-Net does not bridge the gap. Using SMR, performances are already good in source only, meaning that the module effectively makes modalities interact, but integrating RNA-Net gives additional gain.

5.3.3 Comparing norm alignment techniques

In this section we compare different strategies to obtain a feature norm alignment between different modality features. In particular we first compare approaches that introduce a *learn to re-balance* task with approaches that achieve the same objective by static input-level re-normalization. Then among those approaches, we compare plain RNA-Net with its Hard formulation and finally with the formulation coming from equation 4.10.

Results in table 5.6 confirm the intuitions in section 4.2.1: the static norm alignment approach gives some gain over the base method, resulting from having the classifier working in normalized space, with slightly higher gain using the SNA_{max} approach, consistently over the choice of the architecture. However the induced norm alignment task performs better: HNA

Added layer	Method	Architecture	Top-1 Accuracy(%)		
			verb	noun	action
SE [77]	source only	TBN	47.11	27.95	18.44
	TA^3N	TBN	48.09	29.40	19.37
	RNA-Net	TBN	48.43	29.45	**19.63**
	source only	TBN-TRN	46.58	27.97	19.09
	TA^3N	TBN-TRN	48.72	28.56	19.78
	RNA-Net	TBN-TRN	48.72	28.76	**20.01**
Linear	source only	TBN	45.31	20.86	13.70
	TA^3N	TBN	47.32	28.23	18.67
	RNA-Net	TBN	47.42	28.24	**18.70**
	source only	TBN-TRN	46.87	25.00	17.18
	TA^3N	TBN-TRN	47.46	27.59	19.01
	RNA-Net	TBN-TRN	47.66	27.67	**19.05**
SMR [78]	source only	TBN	48.51	29.80	20.14
	TA^3N	TBN	48.50	30.05	20.28
	RNA-Net	TBN	48.64	30.01	**20.30**

Table 5.5. Comparison of different modules for RNA-Net, in UDA context using all three modalities (RGB-Flow-Audio)

performs slightly better than RNA-Net, at the cost of having carefully and extensively searched the best combination of hyperparameters; conversely, RNA-Net achieves good performances with little parametrization. Finally, by encouraging a rebalance towards an higher mean feature norm with loss formulation in equation 4.10, RNA-Net++ achieves even higher performance with respect to HNA.

5.3.4 Evaluating robustness

Training multi-modal networks is harder than training the uni-modal counterparts: as discussed in section 3.3.2, different modalities separately overfit and generalize at different rates, so the joint optimization of the related branches is sub-optimal. To test how different methods deal with this problem, here we propose an ablation study in which a model trained on all three modalities is being tested with only two modalities. Intuitively, a more

Architecture	Method	Top-1 Accuracy(%)		
		verb	noun	action
TBN [7]	source only	47.11	27.95	18.44
	TA^3N	47.22	29.04	18.94
	SNA$_{min}$	47.33	29.55	19.28
	SNA$_{avg}$	47.22	29.56	19.24
	SNA$_{max}$	47.23	29.53	19.31
	RNA-Net	48.43	29.45	19.63
	HNA	48.13	29.75	19.71
	RNA-Net++	48.65	30.04	**20.06**
TBN-TRN [76]	source only	46.58	27.97	19.09
	TA^3N	47.48	28.35	19.25
	SNA$_{min}$	47.26	28.84	19.38
	SNA$_{avg}$	47.52	28.81	19.38
	SNA$_{max}$	47.61	28.71	19.40
	RNA-Net	48.72	28.76	20.01
	HNA	48.80	28.96	20.18
	RNA-Net++	49.37	29.51	**20.61**

Table 5.6. Ablation on different choices of feature norm alignment: RNA-Net, HNA and different variations of SNA, using all three modalities

robust method is able to guarantee higher performances in such situation in which part of the training information is missing. This implicitly means that the network is able to better optimize each modality stream separately, so avoiding to relying too much on the stronger modality. Table 5.7 shows the results of this experiment: how it is possible to see, RNA-Net is consistently more robust than the baseline, achieving higher accuracy over all the modality combinations.

Dropped Modality	Method	Architecture	Top-1 Accuracy(%)		
			verb	noun	action
R	source only	TBN	43.58	21.35	14.67
	TA^3N	TBN	44.70	21.74	14.60
	RNA-Net	TBN	46.13	22.86	**15.80**
	source only	TBN-TRN	43.07	20.78	14.63
	TA^3N	TBN-TRN	44.59	20.78	14.89
	RNA-Net	TBN-TRN	46.78	22.41	**16.42**
F	source only	TBN	36.99	23.78	13.51
	TA^3N	TBN	36.74	24.08	13.41
	RNA-Net	TBN	37.96	24.69	**14.38**
	source only	TBN-TRN	35.95	22.81	13.41
	TA^3N	TBN-TRN	36.09	22.95	13.71
	RNA-Net	TBN-TRN	37.49	23.24	**14.08**
A	source only	TBN	40.22	23.63	13.25
	TA^3N	TBN	40.89	24.83	13.51
	RNA-Net	TBN	41.03	24.48	**13.93**
	source only	TBN-TRN	41.46	24.22	14.29
	TA^3N	TBN-TRN	41.65	24.67	14.43
	RNA-Net	TBN-TRN	42.23	24.29	**14.50**

Table 5.7. Testing the robustness of RNA-Net with respect to baseline methods. The models have been trained on all three modalities (RGB-Flow-Audio), and at test time one modality has been dropped.

Chapter 6

Qualitative Results and Visualization

In this chapter we show qualitative results of the application of our approach on a two-stream network trained on RGB and Audio streams, referring to the setting of the experiments on EK-55 dataset. We propose two qualitative ways of evaluating the contributions: the first one is by analyzing how well separable are the features of source and target samples in the embedding implemented by the deep feature extractors; the second one is by noting how the class activation maps (CAMs [2]) change after introducing the *learn-to-rebalance* task.

6.1 Analysis of the alignment of source and target videos in the action embedding

As previously mentioned, one of the problems in first person action recognition is that the actions' surrounding environment frequently introduces bias into the dataset (environmental bias) and, as a result, a domain shift in the feature space, with consequent drop in performance. To this extend, the objective of an algorithm tackling the domain shift problem is to extract feature that are discriminative of the action, so that the classifier can achieve good performances, and yet domain invariant, so that the model's performance won't drop as consequence of using it with data following a different data distribution. A common way to qualitatively evaluate the feature extractor capabilities is to perform a dimensionality reduction in two or three dimensions and plot the points in the resulting subspace.

For our purposes, we used UMAP on the features of source and target data extracted by a two-stream model trained on source data, and colored the points based on the fact that the corresponding data is from source or target dataset. What we search for is a situation of maximal homogeneity: this would mean that the distribution of the samples is very similar, hence that there is no domain shift; conversely, a clear separation of those features indicates the presence of domain shift. Figure 6.1 refers to features extracted by a two stream model trained in a multi-DG scenario, separately per modality. As it is possible to notice, figures on the right tend to depict a situation in which source and target features are less separable from each other. This qualitatively indicates a mitigated domain shift in the feature space as a result of leveraging multi-modal correlations via relative norm alignment. The effect is even more glaring in figure 6.2, where the model is trained on a single source: this could be motivated by the fact that the model in multi-DG scenario, having seen more than one different domain during supervised training, has an inherently enhanced generalization capability. Indeed, in this last setting there is more room for improvement, hence our approach bridges the gap between the two distributions.

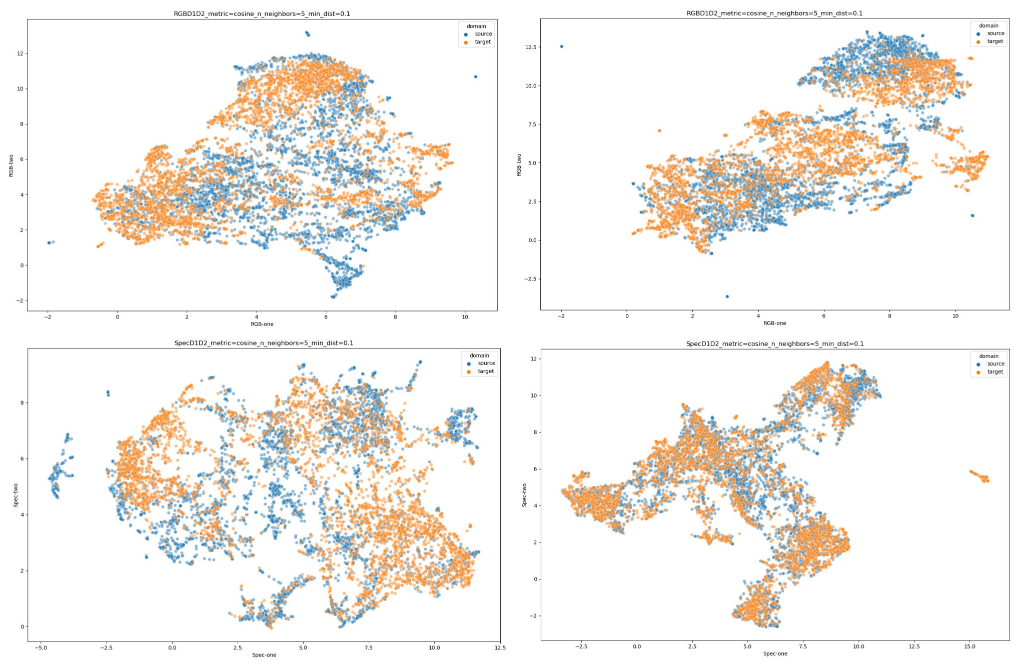


Figure 6.1. UMAP embedding of features extracted by the feature extractors of: the baseline two-stream network (left) and our method (right), using multiple sources (D1, D2) and one target (D3). From top to bottom: embeddings corresponding to RGB and Audio features.

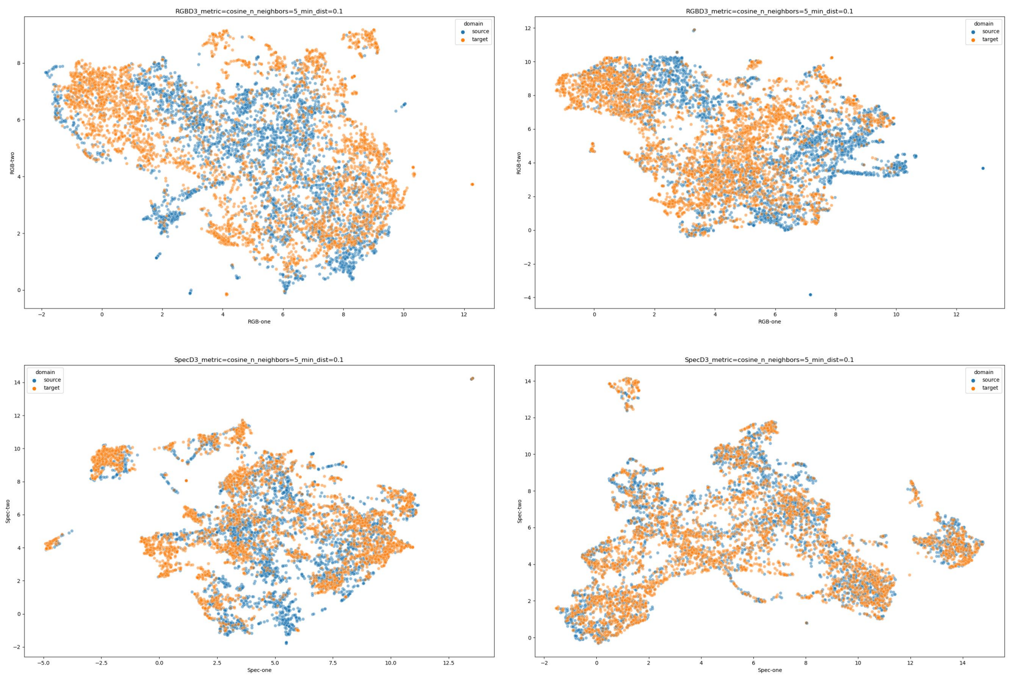
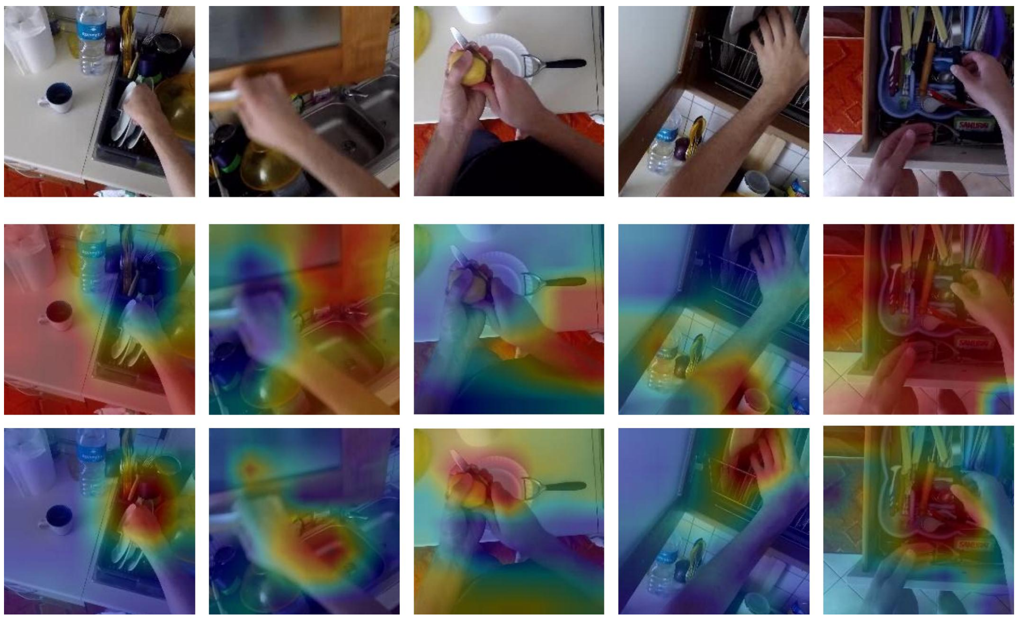


Figure 6.2. UMAP embedding of features extracted by the feature extractors of: the baseline two-stream network (left) and our method (right), using a single source (D3) and multiple targets (D1, D2). From top to bottom: embeddings corresponding to RGB and Audio features.

6.2 Analysis of network's attention

As previously mentioned, environmental bias result in a domain shift in the feature space makes often the network to focus on irrelevant parts of the input frames to predict the action label. A commonly used method in the literature to visualize the attention a convolutional neural network is analyzing the Class Activation Maps (CAMs, described in section 2.3.4).

The images in figure 6.2 clearly show that our approach, by leveraging the *learn-to-rebalance* task, improves the network's ability to correctly identify the image regions that correlate best with the represented action. This effect results from the network's tendency to "choose" which features to favor (i.e., those that are more general) when it unifies the norms. As a result, it tends to reduce those features it deems irrelevant, i.e., those more domain-specific (such as the background) that have a negative impact on generalization.



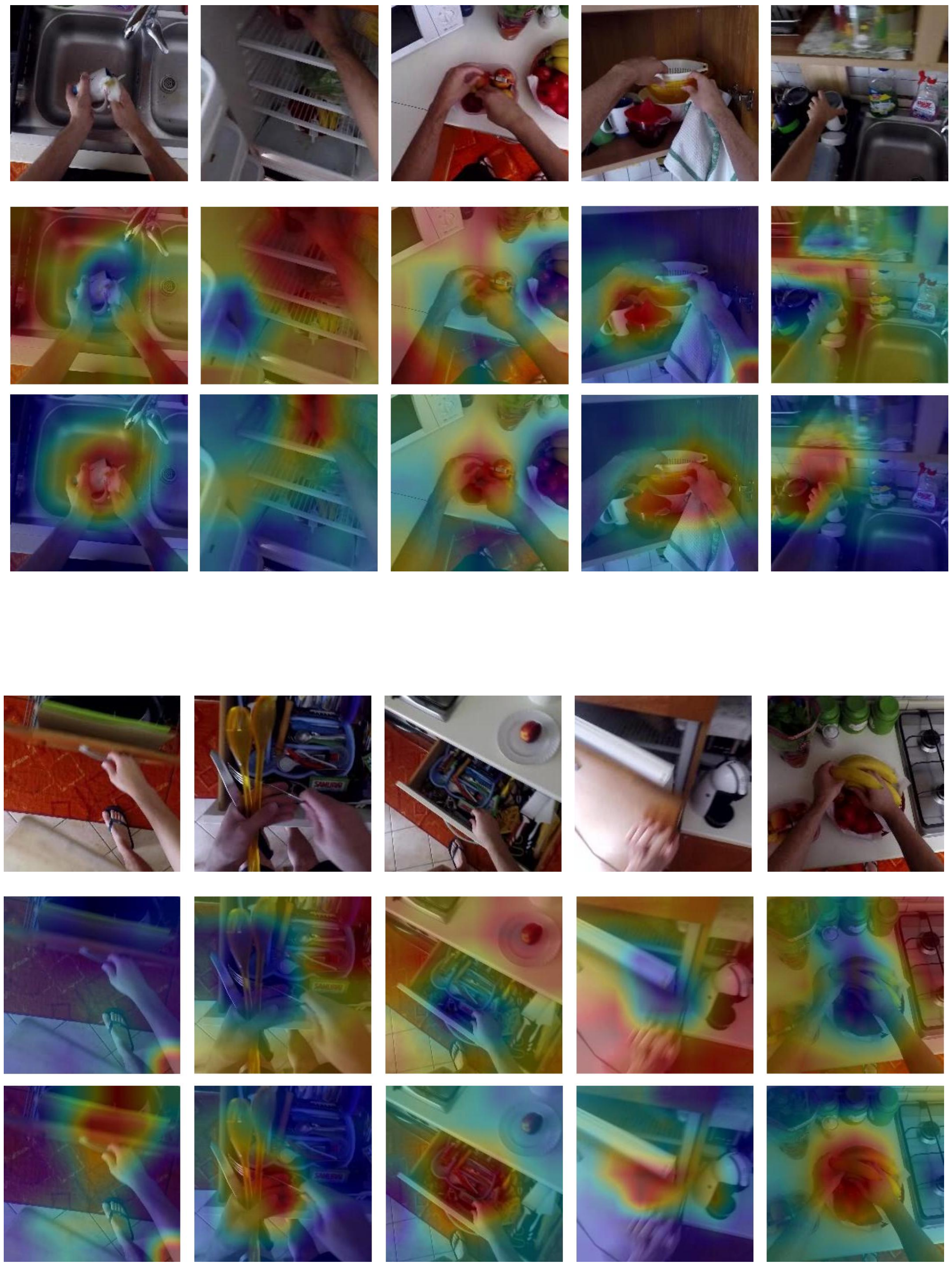


Figure 6.3. Class activation maps corresponding to: the baseline two-stream network (second row) and our method (third row), trained on D1 and D2, applied to target (D3).

Chapter 7

Conclusions

In this work we tackled the problem of domain shift in first person action recognition by exploiting the multi-modality of input data of this task, adopting the Epic-Kitchens dataset [13] for experiments. In particular, our approach stems from two observations: the first one is that domain shifts are not all of the same nature, and their impact can vary significantly across modalities, affecting each modality in its own unique way; the second one is that, when training a multi-stream model, different modalities separately overfit and generalize at different rates, so the joint optimization of the related branches is suboptimal.

To this extend we proposed an approach that takes advantage of the multi-modal nature of the perceptual input, to obtain models that can better leverage the complementarity among modalities, and so are more robust with respect to diverse domain shifts. It belongs to domain generalization methods, but it can be used also in Unsupervised Domain Adaptation scenario, taking further advantage from the availability of target data. In particular, in this work we extended a recent promising approach for audio-visual domain generalization to multi-modal domain generalization in a much more challenging and real-world scenario, that is the setting of the Unsupervised Domain Adaptation challenge released with the Epic-Kitchens-100 dataset. The method ensures a consensus mean feature norm among modality streams, so an equal rate of learning from all modalities, and improves the network's capabilities to leverage data complementarity present in perceptual inputs corresponding to the different modalities. Furthermore, our extension guides the norm alignment towards higher feature norm regions, relying on the *larger norm more transferable* assumption [66]: the experiments validate this intuition, in fact RNA-Net++ is able to effectively enhance the performance of the model. As an ablation study, we showed the importance of *learning-to-rebalance*: a static feature normalization, even allowing the feature extractor to work in the normalized feature space during training, does not have any regularization effect. Qualitative results on the feature space show that our approach effectively helps at bridging the domain gap and that, as a result, improves the network's ability to correctly identify the image regions that correlate best with the represented action.

Our approach is simple to implement, requires little parametrization and has been demonstrated itself effective and of potential interest for many other research fields. Furthermore, it represents a novel approach towards domain adaptation in multi-modal learning from the theoretical point of view, as it exploits the feature norm of different modalities as correlated information content. In particular, their relative magnitude during training has been proved to affect how the networks can exploit the shared semantic between modalities, leverage complementarity and enhance robustness.

Bibliography

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

[2] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," 2015.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[6] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *CoRR*, vol. abs/1406.2199, 2014.

[7] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen, "Epic-fusion: Audio-visual temporal binding for egocentric action recognition," *CoRR*, vol. abs/1908.08498, 2019.

[8] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," 2018.

[9] J. Lin, C. Gan, and S. Han, "Tsm: Temporal shift module for efficient video understanding," 2019.

[10] 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," 2021.

[11] A. Sahoo, R. Shah, R. Panda, K. Saenko, and A. Das, "Contrast and mix: Temporal contrastive video domain adaptation with background mixing," 2021.

[12] J. Munro and D. Damen, "Multi-modal domain adaptation for fine-grained action recognition," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Conference on Computer Vision and Pattern Recognition (CVPR), (United States), pp. 119–129, Institute of Electrical and Electronics Engineers (IEEE), Aug. 2020. Computer Vision and Pattern Recognition ; Conference date: 14-06-2020 Through 19-06-2020.

[13] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "The EPIC-KITCHENS dataset: Collection, challenges and baselines," *CoRR*, vol. abs/2005.00343, 2020.

[14] G. M. Edelman, *Neural Darwinism : the theory of neuronal group selection.* Basic Books, 1987.

[15] L. Smith and M. Gasser, "The development of embodied cognition: Six lessons from babies," *Artif. Life*, vol. 11, p. 13–30, jan 2005.

[16] M. Planamente, C. Plizzari, E. Alberti, and B. Caputo, "Domain generalization through audio-visual relative norm alignment in first person action recognition," 2021.

[17] D. Damen, H. Doughty, G. M. Farinella, A. Furnari, E. Kazakos, J. Ma, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "Rescaling egocentric vision," *CoRR*, vol. abs/2006.13256, 2020.

[18] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014.

[19] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, 1962.

[20] 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.

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 448–456, JMLR.org, 2015.

[22] S. Ioffe, "Batch renormalization: Towards reducing minibatch dependence in batch-normalized models," *CoRR*, vol. abs/1702.03275, 2017.

[23] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," 2018.

[24] Y. LeCun, "Generalization and network design strategies," in *Connectionism in Perspective* (R. Pfeifer, Z. Schreter, F. Fogelman, and

L. Steels, eds.), (Zurich, Switzerland), Elsevier, 1989. an extended version was published as a technical report of the University of Toronto.

[25] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision* (D. Forsyth, J. Mundy, V. di Gesu, and R. Cipolla, eds.), Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp. 319–345, Springer Verlag, 1999. International Workshop on Shape, Contour and Grouping in Computer Vision ; Conference date: 26-05-1998 Through 29-05-1998.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[27] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[28] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010.

[29] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, 11 1997.

[30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[31] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[32] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[34] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[35] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” in *International Journal of Computer Vision*, June 2010.

[36] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, “Hollywood in homes: Crowdsourcing data collection for activity understanding,” *CoRR*, vol. abs/1604.01753, 2016.

[37] Y. Wang, Y. Shen, Z. Liu, P. P. Liang, A. Zadeh, and L.-P. Morency, “Words can shift: Dynamically adjusting word representations using nonverbal behaviors,” 2018.

[38] S. Zhao, G. Jia, J. Yang, G. Ding, and K. Keutzer, “Emotion recognition from multiple modalities: Fundamentals and methodologies,” *IEEE Signal Processing Magazine*, vol. 38, pp. 59–73, nov 2021.

[39] S. Yoon, S. Byun, and K. Jung, “Multimodal speech emotion recognition using audio and text,” 2018.

[40] S. Poria, E. Cambria, D. Hazarika, N. Majumder, A. Zadeh, and L.-P. Morency, “Context-dependent sentiment analysis in user-generated videos,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Vancouver, Canada), pp. 873–883, Association for Computational Linguistics, July 2017.

[41] D. Kiela, H. Firooz, A. Mohan, V. Goswami, A. Singh, P. Ringshia, and D. Testuggine, “The hateful memes challenge: Detecting hate speech in multimodal memes,” 2020.

[42] F. Chen, Z. Luo, Y. Xu, and D. Ke, “Complementary fusion of multi-features and multi-modalities in sentiment analysis,” 2019.

[43] R. Salakhutdinov and G. Hinton, “Deep boltzmann machines,” in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics* (D. van Dyk and M. Welling, eds.), vol. 5 of *Proceedings of Machine Learning Research*, (Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA), pp. 448–455, PMLR, 16–18 Apr 2009.

[44] M. Alam, M. Bennamoun, R. Togneri, and F. Sohel, “A deep neural network for audio-visual person recognition,” in *Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference*, vol. N/A, (United States), pp. 1–6, IEEE, Institute of Electrical and Electronics Engineers, 2015. Biometrics Theory, Applications and Systems (BTAS) 2015 ; Conference date: 08-09-2015 Through 11-09-2015.

[45] J. H. Koo, S. W. Cho, N. R. Baek, M. C. Kim, and K. R. Park, “Cnn-based multimodal human recognition in surveillance environments,” *Sensors*, vol. 18, no. 9, 2018.

[46] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool, "Temporal segment networks: Towards good practices for deep action recognition," 2016.

[47] S. Sudhakaran, S. Escalera, and O. Lanz, "LSTA: long short-term attention for egocentric action recognition," *CoRR*, vol. abs/1811.10698, 2018.

[48] A. Furnari and G. M. Farinella, "Rolling-unrolling lstms for action anticipation from first-person video," *CoRR*, vol. abs/2005.02190, 2020.

[49] A. Cartas, J. Luque, P. Radeva, C. Segura, and M. Dimiccoli, "Seeing and hearing egocentric actions: How much can we learn?," *CoRR*, vol. abs/1910.06693, 2019.

[50] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen, "Slow-fast auditory streams for audio recognition," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 855–859, 2021.

[51] K. Bayoudh, R. Knani, F. Hamdaoui, and A. Mtibaa, "A survey on deep multimodal learning for computer vision: advances, trends, applications, and datasets," *The Visual Computer*, no. 9, 2021.

[52] A. Nagrani, S. Yang, A. Arnab, A. Jansen, C. Schmid, and C. Sun, "Attention bottlenecks for multimodal fusion," 2021.

[53] W. Wang, D. Tran, and M. Feiszli, "What makes training multi-modal classification networks hard?," 2019.

[54] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," 2016.

[55] S. Bucci, A. D'Innocente, Y. Liao, F. M. Carlucci, B. Caputo, and T. Tommasi, "Self-supervised learning across domains," 2020.

[56] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," 2014.

[57] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," 2016.

[58] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," 2017.

[59] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[60] R. Volpi, H. Namkoong, O. Sener, J. Duchi, V. Murino, and S. Savarese, "Generalizing to unseen domains via adversarial data augmentation," 2018.

[61] R. Wang, Z. Wu, Z. Weng, J. Chen, G.-J. Qi, and Y.-G. Jiang, "Cross-domain contrastive learning for unsupervised domain adaptation," 2021.

[62] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, p. 151–175, may 2010.

[63] M. Long and J. Wang, "Learning transferable features with deep adaptation networks," *CoRR*, vol. abs/1502.02791, 2015.

[64] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 2208–2217, PMLR, 06–11 Aug 2017.

[65] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," 2016.

[66] R. Xu, G. Li, J. Yang, and L. Lin, "Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation," 2018.

[67] Q. Dou, D. C. Castro, K. Kamnitsas, and B. Glocker, "Domain generalization via model-agnostic learning of semantic features," 2019.

[68] H. Li, S. J. Pan, S. Wang, and A. C. Kot, "Domain generalization with adversarial feature learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5400–5409, 2018.

[69] Y. Li, Y. Yang, W. Zhou, and T. M. Hospedales, "Feature-critic networks for heterogeneous domain generalization," 2019.

[70] Z. Yao, Y. Wang, J. Wang, P. S. Yu, and M. Long, "Videodg: Generalizing temporal relations in videos to novel domains," 2019.

[71] Y. Zheng, D. K. Pal, and M. Savvides, "Ring loss: Convex feature normalization for face recognition," 2018.

[72] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, "NormFace," in *Proceedings of the 25th ACM international conference on Multimedia*, ACM, oct 2017.

[73] R. Ranjan, C. D. Castillo, and R. Chellappa, "L2-constrained softmax loss for discriminative face verification," 2017.

[74] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," 2018.

[75] B. Zhou, A. Andonian, and A. Torralba, "Temporal relational reasoning in videos," *CoRR*, vol. abs/1711.08496, 2017.

[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 (ICCV)*, October 2019.

[77] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," 2017.

[78] L. Yang, Y. Huang, Y. Sugano, and Y. Sato, "Epic-kitchens-100 unsupervised domain adaptation challenge for action recognition 2021: Team m3em technical report," 2021.

[79] P. Morgado, Y. Li, and N. Vasconcelos, "Learning representations from audio-visual spatial alignment," 2020.

[80] Y. Cheng, R. Wang, Z. Pan, R. Feng, and Y. Zhang, "Look, listen, and attend: Co-attention network for self-supervised audio-visual representation learning," in *Proceedings of the 28th ACM International Conference on Multimedia*, ACM, oct 2020.

[81] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," 2017.

[82] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, "Adaptive batch normalization for practical domain adaptation," *Pattern Recognition*, vol. 80, pp. 109–117, 2018.

[83] A. Jamal, V. Namboodiri, D. Deodhare, and K. Venkatesh, "Deep domain adaptation in action space," Jan. 2019. 29th British Machine Vision Conference, BMVC 2018, BMVC 2018 ; Conference date: 03-09-2018 Through 06-09-2018.