Deep learning methods for Action Unit detection

Gauthier Tallec

Ecole doctorale 391 : Sciences mécaniques, acoustique, électronique et robotique

THÈSE

pour obtenir le grade de docteur délivré par

Sorbonne Université

Spécialité doctorale "Informatique"

présentée et soutenue publiquement par

Gauthier Tallec

Deep learning methods for Action Unit detection

Directeur de thèse : **Kevin Bailly**
Co-encadrant de thèse : **Arnaud Dapogny**

Jury

Matthieu Cord	Professeur, Sorbonne Université, ISIR	Président du Jury
Mohamed Daoudi	Professeur, IMT Lille Nord Europe, CRIStAL	Rapporteur
David Picard	Professeur, Ecole des Ponts Paristech, LIGM	Rapporteur
Stéphane Canu	Professeur, INSA Rouen, LITIS	Examinateur
Antitza Dantcheva	Chargé de Recherche, INRIA Sophia Antipolis	Examinatrice
Kevin Bailly	Maître de conférence, Sorbonne Université, ISIR	Encadrant
Arnaud Dapogny	Chercheur, Datakalab	Encadrant

Résumé

La détection d'Action Unit (AU) consiste à décrire automatiquement les expressions faciales par les activations musculaires qu'elles impliquent. L'intérêt est de fournir une représentation bas niveau qui peut ensuite aider l'apprentissage de tâches d'analyse faciale de plus haut niveau. Cependant, c'est un problème difficile. En effet, les bases de données disponibles ne présentent pas une grande variété de sujets et contiennent beaucoup plus d'images de visages neutres que de visages expressifs. De plus, les AU sont des mouvements subtils du visage et sont donc difficiles à annoter. Ainsi, on dispose de peu de données et certaines des annotations sont susceptibles d'être fausses. Par conséquent, il est peu probable qu'un réseau entraîné sur ces bases de données soit capable de généraliser efficacement. Dans cette thèse, on explore trois pistes pour améliorer les performances de généralisation des détecteurs d'AU.

Premièrement, on cherche à exploiter les dépendances entre les AU pour structurer les prédictions du réseau. Pour ce faire, on utilise des réseaux de neurones récurrents multi-tâches qui traitent les tâches séquentiellement et aident la prédiction de chaque tâche en utilisant les résultats précédents. Cependant, ces méthodes nécessitent d'imposer un ordre sur un ensemble de tâches qui n'a pas d'ordre naturel. Le choix de cet ordre est important. En effet il a été montré qu'il pouvait impacter les performances du réseau. Pour sélectionner les ordres pertinents, on introduit les Multi-Order Networks (MONET) qui apprennent plusieurs tâches ainsi que l'ordre dans lequel elles doivent être prédites. Dans un premier temps, on montre que MONET est capable d'apprendre l'ordre de prédiction optimal dans un environnement contrôlé. Dans un second temps, on montre que MONET surpasse les architectures multi-tâches de base sur plusieurs problèmes de détection d'attributs avec différents types de dépendances entre les tâches. Finalement, on démontre que MONET dépasse les performances de l'état de l'art en détection d'AU sur DISFA et BP4D.

Dans un deuxième temps, on part de l'observation selon laquelle chaque AU modifie une région connue du visage. On tente donc de guider notre détecteur pour qu'il prête attention aux zones pertinentes pour chaque AU. Pour cela, on s'inspire

du succès des transformeurs pour la vision. Concrètement, on évalue plusieurs stratégies de guidage de l'attention multi-tête des transformeurs en utilisant soit des points de repères, soit des cartes de ségmentation du visage. Le résultat de cette étude est que, quelle que soit la nature de l'a priori, forcer les différentes têtes d'un transformeur à prêter attention à des zones différentes permet d'améliorer les performances d'un détécteur d'AU sur BP4D et DISFA.

Enfin, on aborde le problème du bruit dans les annotations d'AU. Pour cela, on utilise d'abord une stratégie de type label smoothing pour réduire la confiance du réseau et ainsi atténuer l'influence des exemples bruités. Cependant, on constate que le label smoothing nuit aux performances de détection. Pour expliquer cette baisse, on suppose que l'application du label smoothing dans des scénarios déséquilibrés aggrave le manque de confiance dans la classe minoritaire. Pour contourner cela, on propose le Vanilla Asymetric Label Smoothing (VALS) qui utilise des coefficients de lissage distincts pour les exemples positifs et négatifs. VALS permet donc de réduire la surconfiance dans la classe majoritaire tout en laissant la classe minoritaire intact. On affine cette stratégie avec le Robin Hood Label Smoothing (RHLS) qui lisse uniquement la classe majoritaire avec un coefficient qui est ajusté en fonction des fréquences empiriques des AU. On montre que les performances de VALS et RHLS sont prometteuses sur BP4D et surpassent les résultats de l'état de l'art sur DISFA.

Abstract

Action Unit (AU) detection aims at automatically caracterizing facial expressions with the muscular activations they involve. Its main interest is to provide a low-level face representation that can be used to assist higher level affective computing tasks learning. Yet, it is a challenging task. Indeed, the available annotated databases display limited face variability and are imbalanced toward neutral expressions. Furthermore, as AU involve subtle face movements they are difficult to annotate so that some of the few provided datapoints may be mislabeled. Therefore, a deep neural network trained on such datasets is likely to overfit and to consequently display poor generalisation performance. In this thesis, we explore three different ways to improve the generalisation performance of an AU detector.

First, we aim at exploiting known inter-dependencies between Action Units to better structure the prediction of the network. To do so, we leverage multi-task recurrent neural networks that process tasks sequentially and help each task prediction with the results of the previous ones. However, such method requires imposing an arbitrary order into an unordered task set. Crucially, it has been shown that this ordering choice impacts predictive performance. For that purpose, we propose Multi-Order Networks (MONET) that jointly learns multiple tasks along with the order in which they should be predicted. Experimentally, we first show that MONET is able to learn the optimal prediction order in a controlled environment. Second, we validate MONET architecture by showing that MONET outperform existing multi-task baselines on multiple attribute detection problems chosen for their wide range of dependency settings. More importantly, we demonstrate that MONET significantly extends state-of-the-art AU detection performance on both DISFA and BP4D.

Second, we build upon the observation that each Action Unit is locally constrained to a known region of the face. Consequently we attempt to guide our deep Action Unit detector towards paying attention to AU-relevant zones. For that purpose, we draw inspiration from the recent success of vision transformer architectures and their multi-head attention mechanism. More precisely we benchmark several strategies to guide the multi-head attention of vision transformers

with either landmarks-based and face parsing-based priors. The main outcome of our study is that, whatever the nature of the prior, forcing different attention heads to pay attention to different zones is an useful strategy as it improves AU detection performance on both DISFA and BP4D.

Finally, we tackle the noisy annotation problem of AU detection. To this end, we first attempt to use label smoothing to reduce the network confidence and consequently mitigate the influence of noisy examples on the training. However, we experimentally observe that label smoothing degrades AU detection performance. To explain this drop in performance, we hypothetize that applying label smoothing in imbalanced scenarios worsens the pre-existing under-confidence in the minority class. To circumvent that issue, we propose a Vanilla Asymetric Label Smoothing (VALS) that use separate smoothing coefficients for positive and negative examples. Consequently VALS provides the flexibility to reduce over-confidence in the majority class while leaving the minority class untouched. We further refine this strategy by proposing Robin Hood Label Smoothing (RHLS) that smoothes only the majority class with a smoothing coefficient that is scaled based on AU empirical frequencies. Experimentally, we show that both VALS and RHLS display promising performance on BP4D and outperforms state-of-the-art results on DISFA.

Contents

Chapter 1

Introduction

The Facial Action Coding System (FACS) [16] is a language used to describe facial expressions that have been developped by the psychologists Paul Ekman and Wallace Friesen. It aims at decomposing facial cues from an anatomical perspective using unitary muscular activations called Action Units (AU). A detailed decomposition of two faces based on FACS is provided in figure 1.1.

Compared to Ekman's 7 Basic Emotions [15] that is another main face representation system, FACS has several advantages: First, as its name indicates, Ekman's 7 Basic Emotions model classifies faces into 7 emotion-related classes (Anger, Contempt, Disgust, Enjoyment, Fear, Sadness and Surprise). The prob-

Figure 1.1: Two faces decomposed using the Facial Action Coding System [42]

lem with this representation is that the 7 classes show some degree of subjectivity. This means that, based on their life experience, two annotators may assign different labels to the same image. On the contrary, FACS is based on muscular activations. It is therefore less open to interpretation and is likely to provide labels that are consistent across annotators.

The second advantage of FACS compared to the Basic Emotion model is the precision of the face representation it provides. Indeed, the Basic Emotion only provides very coarse information regarding a face. In contrast, FACS include more than 30 AUs that can be combined to finely describe and discriminate a wide range of facial expressions. Even more interesting, FACS low-level face representation can be used to caracterize higher-level facial expressions. For example, the typical happy face is likely to display AU12 (smile), AU6 (cheek raiser) and AU7 (Lid Tightener).

Context, Objectives and Challenges

The objective of this thesis is to design an efficient automatic AU detection system. It is motivated by the potential medical applications that such improvement would enable. In particular, it is part of DYSFER, a project that aims at improving the detection of respiratory distress, also called dyspnea, in a hospital environment.

The initial observation that inspired the DYSFER project is that dyspnea is a source of panic for hospitalized patients. Even worst, it goes unnoticed by usual captors. Indeed, the oxygen saturation rate, which indicates how well the breathing process is doing, does not drop at dyspnea early stages. In those stages, the difficulty in breathing is not life threatening, yet it still represents a fearful experience. It is all the more a problem as hospitalized patients that are prone to suffer from dyspnea are often unable to call for help because of invasive medical support devices. In an attempt to solve that issue, a preliminary study conducted at the Pitié Salpêtrière hospital showed that dyspnea could be detected from the patients' faces. Furthermore, this study roughly caracterizes the typical dyspneic face using FACS. As a consequence, automatic AU detection could be used to detect dyspnea in hospital environment and in turn improve both physical and psychological comfort of patients.

Meanwhile, in the past few years, several AU-annotated datasets have been made publicly available. This permitted the development of deep learning based methods for AU detection. In particular, several methods [23, 60, 64, 63] have shown promising results. Therefore we naturally tackle the AU detection problem using deep learning-based approaches.

From a machine learning point of view, Action Unit Detection can be formulated as a multi-task problem in which each task consists in detecting whether a single AU is present or not. In practice, the main challenge that hinders the development of efficient AU detection is data scarcity. Indeed, the way AU activation modify faces may vary across subject (mainly because of face shape variations). Consequently, proper Action Unit detection learning requires training sets with large subject variability. Unfortunately, harvesting AU labels is an extremly tedious process as it requires frame by frame expert annotations of subtle facial movements. As a consequence, the available datasets only feature few different identities ($\sim$ 50). Furthermore, those datasets are based on videos so that, as AU are rather short events, only few frames feature AU activations. Last but not least, Action Units often involve very subtle movements. Thus, some face images may be challenging even for expert annotators [99]. Therefore, annotated examples may display some noise. In a nutshell, AU detection datasets display too few subjects with too few examples of activation for each AU, and, on top of that, those scarce labels may not be so reliable. For all those reasons, a vanilla deep neural network trained on such dataset is likely to overfit on the training set and consequently display poor generalisation performance.

Research Directions

In order to reduce this overfitting problem, we investigate three different tracks. The first track consists in exploiting known dependencies between Action Units. Indeed for both social and physiological reasons, AU display strong inter-dependencies. Efficient capture of those dependencies could help structure a network prediction and consequently improve its performance.

The second track that we follow in our struggle against overfitting is to guide deep neural networks with expert knowledge based spatial prior. It builds up on the observation that most Action Units are constrained to specific parts of the face. For example, AU1 and AU2 affects the eyebrows while AU12 modify the shape of the mouth. Therefore, providing a deep neural network with a spatial prior that specifies the location of AU relevant zones alleviate the need to learn such zones from the data. Intuitively enough, such spatial guidance of feature extraction may prevent a network from learning to predict AU occurence based on specific details of the faces contained in the training set. Consequently, it may improve generalisation performance.

The last axis that we investigate to fight overfitting is annotation noise mitigation methods. Indeed, AUs are subtle facial movements. Therefore, even expert annotators may miss some AU activations. Critically, as neural networks are highly

expressive learners, they are likely to fit those noisy annotations, which in turn degrades the generalisation performance of the learned model. As a consequence, leveraging noise mitigation methods may improve performance.

Summary of contributions

Multi-Order Network for better AU dependency modelling

In an attempt to model AU inter-dependencies, we leverage multi-task recurrent networks for their capacity to help the learning of a given task with the predictions of previously predicted ones. However, contrary to standard sequence processing which is the textbook case of recurrent neural networks, the set of tasks has no natural order *a priori*. Therefore, predicting tasks sequentially implies enforcing an arbitrary order into the set of tasks. Crucially, it has been shown that recurrent networks [80] are sensitive to the prediction order.

For that purpose, we propose a Multi-Order Network (MONET) that jointly learns multiple tasks along with the order in which they should be predicted. More precisely, we introduce soft orders, the probabilistic extension of prediction orders that allow MONET to smoothly navigate between orders. Additionally, we further refine MONET order selection by introducing two mechanisms: an order warmup and an order dropout that enhance order exploration by encouraging MONET to keep good predictive performance for several orders of prediction. From an experimental point of view, we show (a) that MONET is able to retrieve the correct order in a controlled environment, (b) that MONET outperforms baseline multi-task approaches on a wide range of attribute detection problems with diverse levels of inter-task dependencies, and (c) We demonstrate that MONET extends state-of-the-art performance in action unit detection. All in all, our contributions are as follows:

- We introduce MONET, a multi-task learning method with joint task order optimization and prediction. From an architectural standpoint, MONET uses soft order selection in Birkhoff's polytope as well as task-wise cell sharing to model task order and prediction in an end-to-end manner.

- From a learning standpoint, we introduce order dropout and warm up strategies that work hand in hand with the order selection to encourage modules to keep good predictive performances in several orders of prediction.

- Experimentally, we first validate MONET architecture in controlled environments by a) Verifying MONET capacity to retrieve the correct order on

a toy dataset, and b) Showing that MONET outperforms several multi-task approaches on a wide range of attribute detection problems with diverse levels of inter-task dependency. Finally, we conclude by demonstrating that MONET extends state-of-the-art performance in action unit detection.

This work has been accepted as a journal paper [73]. Furthermore a patent [72] related to this invention has also been submitted.

Guiding Transformers Attention for better Action Unit detection

In order to guide AU detection networks with a spatial prior we inspire from existing work on convolutional networks [29, 59, 60, 3]. Yet, recent work [86, 23] showed that vision transformers [78, 13] display promising performance AU detection. However, those highly expressive architectures are known to be sensitive to overfitting. Consequently, appropriate learning strategies could improve their performance on AU detection.

For that purpose, we propose to adapt attention guiding methods to transformer architectures. For that purpose, we investigate several methods to constrain the multi-head attention mechanism of transformers with AU-relevant priors. More precisely, we implement several guiding strategies that use either landmarks or face parsing priors. Overall, our main finding is that, regardless of the type of prior used, constraining each head of the multi-head attention mechanism to a different prior (eg. a face parsing class per head) improve the predictive performance of a transformer. We validate this claim on both BP4D and DISFA AU detection problems. In particular, we show that (a) constraining each head with a landmark-based prior improve performance on BP4D (b) constraining each head with face parsing-based prior boost performance on both BP4D and DISFA. To summarize, our contributions are:

- We propose a landmark-based prior guidance method that leverages the same kind of constraint as in [3]. It bolsters collaboration between the different heads of the multi-head attention by encouraging each head to attend to different AU-relevant locations. We show that this method provides a boost in performance on BP4D w.r.t a baseline.

- Similarly, we propose a face-parsing based prior guidance that makes use of the token-based mechanism in [14] to constrain the attention of each head to a different semantical part of the face. Again, we demonstrate that this method performs better than the baseline method on DISFA and BP4D.

The work on guiding transformers with expert prior have not yet been submitted. However, we used similar transformer-based architecture to concur to the ABAW3 and ABAW4 challenges which are CVPR 2022 and ECCV 2022 workshops respectively. The summaries of our submissions at those challenges can be found in [74] and [70].

Noise mitigation in imbalanced situations

In order to address the noisy annotation problem of AU detection, we aim at taking advantage from the recent success of label smoothing [69] at mitigating noise [38] by reducing over-confidence. However, vanilla label smoothing reduces over-confidence uniformly in all classes. Therefore, applying it in imbalanced situations may worsen the pre-existing under-confidence in the minority class. For that purpose, we propose a Vanilla Asymetric Label Smoothing (VALS) that separate smoothing coefficients based on whether labels are positive or negative. Such separation provides the flexibility to apply more smoothing to examples from the majority class and consequently avoid worsening the minority class under-confidence problem. We further embrace this philosophy by proposing Robin Hood Label Smoothing (RHLS) that smoothes only the majority class with a per AU smoothing coefficient that depends on AU empirical frequencies (the higher the frequency the larger the smoothing coefficient). By doing so, it introduces a probability to take examples from the rich class and give them to the poor. Consequently, it reduces both the imbalance-based over-confidence issue and the negative impact of noisy majority class examples. In a nutshell, the contribution of this part of our work are the following:

- We introduce VALS and RHLS that adapts label smoothing to imbalanced situations by restraining over-confidence reduction to the majority class. Consequently, it mitigates both imbalance over-confidence issue and the negative impact of majority class noisy examples.
- Experimentally, we show that AU detection performance benefits from the use of VALS and RHLS without any additional computational overhead. More precisely, we demonstrate that applying VALS on a modern multi-task baseline displays promising performance on BP4D and outperforms state-of-the-art performance on DISFA. Furthermore, RHLS frequency-based scaling of AU-wise smoothing coefficient seems to be particularly adapted to the large AU frequency variations in DISFA, as it surpasses both VALS and *a fortiori* state-of-the-art performance.

This work [71] is currently under review at *ICIP2023*.

List of Publications

The work in this thesis lead to the following preprints and publications:

Journal paper:

- G. Tallec, A. Dapogny, K. Bailly. Multi-order networks for action unit detection. In *IEEE Transactions on Affective Computing 2022*.

Conference Submissions:

- G. Tallec, A. Dapogny, K. Bailly. Fighting noise and imbalance in Action Unit detection problems. *arXiv preprint 2303.02994* (submitted to ICIP2023).
- G. Tallec, E. Yvinec, A. Dapogny, K. Bailly. Fighting over-fitting with quantization for learning deep neural networks on noisy labels. *arXiv preprint 2303.11803* (submitted to ICIP2023).

ABAW challenge submissions:

- G. Tallec, E. Yvinec, A. Dapogny, K. Bailly. Multi-label transformer for action unit detection. *arXiv preprint 2203.12531*.
- G. Tallec, J. Bonnard, A. Dapogny, K. Bailly. Multi-Task Transformer with uncertainty modelling for Face-Based Affective Computing. *arXiv preprint 2208.03506*.

Reading Guide

This work articulates around the three main tracks that we investigate to fight overfitting in AU detection. In particular, chapter 2 presents our method for better AU inter-dependencies exploitation, chapter 3 outlines our work on guiding transformers and chapter 4 contains our contributions on noise mitigation for AU detection.

The core of each chapter is roughly structured the same way. First, the problem that we address is introduced. Then, we present existing methods to solve that problem. In particular, this discussion is split in two parts: A first part that focus on methods that are specific to Action Units, and a second part that present more general methods. Subsequently, we present our method along with the experiments implemented to validate it. Finally, we discuss our results and expose future work plans.

Chapter 2

Multi-Order Networks for better AU dependency modelling

2.1 Introduction

Specific combinations of Action Units are widely used to transmit emotional information. For example, AU12 (smile) and AU6 (cheek raiser) are often used together to convey happiness. Similarly, the coocurrence of AU1-2 (eyebrow raisers) and AU25 (open mouth) is an indicator of surprise. As a consequence Action Units display strong inter-dependencies with one another.

In this chapter, we aim at fully exploiting those dependencies. Ideally, such exploitation may help to better structure the prediction of a model, and in turn improve performance. For that purpose, we use a multi-task formulation of the Action Unit Detection problem in which each task is the binary classification problems associated with the detection of a single AU.

The most widely adopted strategy [53, 27] for solving such multi-task problems is to use a common encoder and predict the different tasks using separate regressors in parallel. If this strategy comes with the advantage of parallel task processing and thus low computational costs, it fails to model inter-task dependencies. Therefore, it is not the ideal architecture for modelling the known dependencies between the different Action Units (such as for example, mutual exclusion between AU1 and AU4 as an individual cannot simultaneously frown and raise his.her eyebrows).

In an attempt to provide architectures that better model inter-task dependencies, several works [47, 33] leveraged recurrent networks that predict tasks sequentially. However, contrary to standard sequence processing, which is the textbook case of recurrent networks, the set of tasks to predict has no natural

order *a priori*. Therefore, making task prediction sequential requires enforcing an arbitrary order into the set of tasks. Crucially, recurrent networks performance have been proven sensitive to the order in which elements are predicted [80]. Consequently, in the frame of multi-task learning where predicted elements are tasks, the task prediction *order matters*.

In the light of the latter remarks, we introduce Multi-Order Networks (MONET) for jointly learning to predict the tasks along with the order in which they should be predicted. In particular, MONET leverages a differentiable order selection mechanism to jointly learn task-wise modules along with the order in which they should be chained. Furthermore, we introduce two different order learning regularization methods, a warm-up and an order dropout that encourage MONET order selection mechanism toward efficient exploration of the order space. To summarize, the contributions presented in this chapter are the following:

- We introduce MONET, a multi-task learning method with joint task order optimization and prediction. From an architectural standpoint, MONET uses soft order selection in Birkhoff's polytope as well as task-wise cell sharing to model task order and prediction in an end-to-end manner.

- From a learning standpoint, we introduce order dropout and warm up strategies that work hand in hand with the order selection to encourage modules to keep good predictive performances in several orders of prediction.

- Experimentally, we first validate MONET architecture in controlled environments by a)Verifying MONET capacity to retrieve the correct order on a toy dataset b) Showing that MONET outperforms several multi-task approaches on a wide range of attribute detection problems with diverse levels of inter-task dependency. Finally, we conclude by demonstrating that MONET extends state-of-the-art performance in Action Unit detection.

The rest of this chapter is organized as follows : in section 2.2, we review state-of-the-art approaches for multi-task learning. Section 2.3 presents the general learning framework along with the multi-task baselines, while section 2.4 describes MONET architecture. Section 2.5 displays our experimental results on (a) a toy dataset in which the order is known (b) different subsets of CelebA chosen for the wide diversity of dependency they display (c) AU detection datasets. Finally section 2.6 contains a discussion about current MONET limits and deduces possible extensions.

2.2 Related Works

AU dependency modelling

As AUs are known to display strong inter-dependencies, capturing those dependencies could help better structure the prediction of a neural network and in turn improve its performance. For that purpose multiple approaches adopted graph-based modelling of AU dependencies. The method in [10] uses backpropagation through a probability graphical model (PGM) to learn pairwise dependencies. Similarly, copula CNN were introduced in [81] to model the relationships between ordinal variables associated with the activation intensity of each AU. However, the main drawback of those two approaches is that they require complex learning strategies with multiple steps that learn each component separately before training them all together.

To circumvent this issue, several works used Graph Neural Networks (GNN) to jointly learn AU-specific features along with how they should interact with each other. More precisely, in [28] and [48], a graph of dependencies is defined *a priori* and further used to orchestrate the interactions between AU-specific features in an end-to-end manner. In the same vein, authors in [64] proposed to reproduce the same graph-based interaction orchestration and to refine it with a MCMC sampling algorithm that jointly update the graph to optimize detection performance.

Multi-task Learning

Deep multi-task learning methods is the subset of deep learning methods that aim at learning several tasks within a single architecture. Such architecture have parts that are shared across tasks and parts that are specific to each task. In theory, there are several advantages for sharing parts across tasks. The most obvious is computational cost reduction as sharing operations naturally reduces the total number of operations. A less straightforward advantage of multi-task learning is overfitting reduction. In fact, the idea is that using the same features as input to predict several tasks encourages the network toward producing more general features and in turn improves its generalisation performance.

However in practice, accessing the sweet spot that unlocks those advantages is difficult. In particular, the main obstacle that hinders efficient multi-task learning comes from the intrinsec multi-objective aspect of the optimization it implies. Indeed, optimizing several losses within the same architecture is likely to generate conflicts as the minimization of the different objectives may require to move the same parameters in different and potentially conflicting directions.

In order to address that problem, a first line of research consists in finding appropriate multi-task supervision signals that prevent such conflicts. Naturally enough, as the most widely adopted multi-task loss is simply the sum of all task losses, several works attempted to replace the sum of losses by a weighted sum of losses. Authors in [25] noted that the uniform weighting of each task loss contribution theoretically comes from the assumption that all tasks have the same level of uncertainty. Therefore, they proposed to relax this hypothesis and to learn the level of uncertainty of each task (softmax temperature for classification and variance for regression). Then, they used this learned task uncertainty measure to weight the loss contribution of the task it is associated. Parallely, it has been noticed [9, 58, 88, 32] that scaling the contribution of each task loss in the total loss is equivalent to scaling the influence of each task gradient in the total gradient. Consequently, several works used gradient-based heuristics to find appropriate weighting coefficients. For example GradNorm [9] adaptatively weight task losses so that the resulting task gradients share the same scale. Similarly, PCGrad [88] aim at reducing conflicts between gradients. For that purpose, when the gradients associated with the loss of two different tasks points to opposite directions, it solves the conflicts by projecting one into the normal plane of the other.

Concurrently, another line of research consists in limiting conflicts between losses by carefully choosing which weights are shared across tasks and which weights are task specific. Seminal work [53, 100, 84, 27], made use of a common encoder along with task specific regressors. However, the main weakness of such method comes with deciding how far features should be shared. Indeed, it intuitively depends on task relatedness (the more tasks are related the more they benefit from sharing) which is not obvious to measure *a priori*. To tackle this issue, numerous approaches [56, 46, 17, 31] used adaptative architectures that jointly learn which layers should be shared between tasks, as well as the task prediction itself. Notably, [46] first pretrains single task networks and then, learns an hybrid architecture that uses convex combinations of the single task networks features to solve all tasks at once. More recently, this philosophy of learning to share features between tasks has been used in modular approaches. It consists in learning a set of trainable modules along with how they should be combined for each tasks. For instance, soft layer ordering [43] learns a set of modules along with the best way to combine them to predict each task. In the same vein, a select or skip policy [68] was used to determine which module should be used for each task.

Sharing weights across tasks help to find features that are useful for all tasks. Therefore it implicitly models input-related task conditional dependencies. Though, it doesn't capture inter-task relationships that do not depend on input (e.g the prior that detection of a beard implies high probability that the subject also has

a mustache). In order to model those dependencies, several approaches [56, 47, 2] leveraged recurrent neural networks. Those networks decompose the task joint distribution into a product of conditional distributions using Bayes chain rule. Most importantly, the work in [80] showed that *order matters*, meaning that the order in which the chain rule is unrolled impacts the final joint estimate modelization performance. In the light of this observation it proposes a two-steps method: The first step consists in an exploration phase in which the performance of several orders are tested. At the end of this phase, a single order is fixed once and for all based on the exploration phase performance and predictions are computed using this order.

Our work lies in the continuity of the order optimization paradigm proposed in [80] which aim at improving current Bayes chain rule based joint distribution estimation. However, we stand out from it by drawing inspiration in [43] to: (a) Propose a soft order selection mechanism that navigates through Birkhoff's polytope searching for relevant orders, and (b) Propose a new task-wise modular recurrent architectural design. More precisely, MONET smooth selection contrasts with the once and for all choice of order in [80] by keeping on learning several orders during all the training phase. We take full advantage of this by adding warm up and order dropout mechanisms that encourage modules to display good predictive performances for several orders of prediction.

By optimizing the task order, MONET takes advantage of situations where *order matters*. Furthermore, we believe that learning more than one task order all along the training improves MONET generalization capacity and thus predictive performance in more general multi-task settings where order do not necessarily matter.

2.3 Multi-task learning framework and baselines

2.3.1 Learning framework

In this section we present the framework for multi-task classification that we will use all along this work. For that purpose, we provide ourselves with a T tasks training dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ assumed to be sampled i.i.d from general distribution p. $\mathbf{x}$ represents the input features and $\mathbf{y} \in \mathbb{R}^T$ holds the annotation for each task, such that for $t \in [1, T]$, y^t is the ground truth for task t.

The training general philosophy is to make use of the example data points in $\mathcal{D}$ in order to learn the relationships that links inputs and labels. In more technical

words, the training procedure aims at approximating the general joint distribution p using a parametric family of probability distribution $\mathcal{F} = \{p_\theta, \theta \in \Theta\}$. For that purpose, we aim at minimizing the following optimization problem:

$$\underset{\theta \in \Theta}{\text{minimize}}\, D_{\text{KL}}(p \,||\, p_\theta). \tag{2.1}$$

Classically, the Kullback-Leibler term is often simplified as follows:

$$\begin{aligned} D_{\text{KL}}(p \,||\, p_\theta) &= \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p}[\log \frac{p(\mathbf{x},\mathbf{y})}{p_\theta(\mathbf{x},\mathbf{y})}] \\ &= \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p} \log p(\mathbf{y} \mid \mathbf{x}) - \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p} \log p_\theta(\mathbf{y} \mid \mathbf{x}), \\ &= \text{CE}(p \,||\, p_\theta) - H(p), \end{aligned} \tag{2.2}$$

where $H(p)$ is constant w.r.t θ. Consequently, problem 2.1 is equivalent to:

$$\underset{\theta \in \Theta}{\text{minimize}}\, \text{CE}(p \,||\, p_\theta). \tag{2.3}$$

Yet we do not have access to the general joint distribution p. To circumvent that problem we define the training dataset empirical distribution $\hat{p}$ as follows:

$$\hat{p} = \frac{1}{N} \sum_{i=1}^{N} \delta_{(\mathbf{x}_i, \mathbf{y}_i)}, \tag{2.4}$$

where $\delta_{\mathbf{x}}$ represents the dirac distribution centered in $\mathbf{x}$. This empirical distribution is then used to estimate the cross-entropy:

$$\text{CE}(p \,||\, p_\theta) \approx \text{CE}(\hat{p} \,||\, p_\theta) = \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(\mathbf{y}_i \mid \mathbf{x}_i) = \mathcal{L}(\theta), \tag{2.5}$$

Therefore the training procedure consists in minimizing $\mathcal{L}(\theta)$. This minimization is usually performed using gradient descent. At the end of this procedure, the conditional distribution p_{θ^*} can be used to estimate the distribution of labels for input that are not in the training dataset. The general hope that underpines all this strategy is that distribution p_θ shares regularities with p so that the interpolation that p_θ does between the points in $\mathcal{D}$ is reasonably close to the real values in p.

Strong with this learning framework, we now introduce our multi-task learning methods.

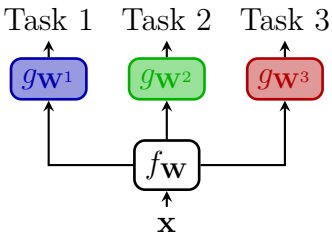


Figure 2.1: VMN computational graphs. Input features are first fed to a shared encoder to compute a representation that is common to all tasks. Then this common representation is given to task specific predictors (represented in different colors) to output the final prediction for each task.

2.3.2 Multi-task baselines

Vanilla Multi-task Networks (VMN)

The most commonly used modelling hypothesis for multi-task learning is the assumption that tasks are independant conditionally to the input. From a formal point of view, this hypothesis implies that:

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} p_\theta(y^t \mid \mathbf{x}) \tag{2.6}$$

In order to take some benefits from the multi-task setup, methods that work under this assumption often use an encoder $f_{\mathbf{W}}$ with parameters $\mathbf{W}$ to encode inputs into a common representation before feeding the result to task-wise regressor $g_{\mathbf{W}^t}$, parametrized by $\mathbf{W}^t$ that outputs prediction p^t for each task:

$$p^t = \text{act}^t(g_{\mathbf{W}^t} \circ f_{\mathbf{W}}(\mathbf{x})), \tag{2.7}$$

where act^t is an activation function chosen based on the nature of task t. Basically, classical choices of activation functions include sigmoid for binary classification, softmax for categorical classification.

We refer to those architecture as Vanilla Multi-task Networks (VMN) and a general computational graph is depicted in 2.1.

From a supervision point of view, the conditional independence assumption

2.6 enables the following development of the cross-entropy based loss in 2.5:

$$\begin{aligned}\mathcal{L}(\theta) &= \frac{1}{N}\sum_{i=1}^{N}\log p_\theta(\mathbf{y}_i \mid \mathbf{x}),\\ &= \frac{1}{N}\sum_{i=1}^{N}\log\prod_{t=1}^{T} p_\theta(y_i^t \mid \mathbf{x}),\\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\log p_\theta(y_i^t \mid \mathbf{x}),\\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\ell(y_i^t, p_i^t),\end{aligned} \tag{2.8}$$

where $\theta = \{\mathbf{W}, (\mathbf{W}^t)_t\}$ regroups all networks parameters.

Similar to the choice of the activation function, the expression of loss function ℓ depends on the nature of task t, for binary classification it is a binary cross-entropy while for categorical classification it becomes a categorical cross-entropy.

The interests of VMN are mainly two fold: First it allows parallel prediction of all the tasks which comes pretty handy from a computational perspective. Second, sharing representation between tasks encourages the encoder toward producing more general feature. This in turn may improve its generalization performance and reduce overfitting.

However, as mentioned in section 2.2, determining the correct size for the shared and the task specific parts is both difficult and important as it significantly affect predictive performance. To investigate this we study two instances of the VMN template: VMN Common (VMNC) in which everything is shared across tasks and VMN Separate (VMNS) in which the regressors for each parts consists of two or three dense layers.

Finally, another line of criticism that calls the use of VMN into question is that its core assumption does not take into account inter-task relationships that are independant of the input image. Typically, such relationships include numerous common knowledge prior about the nature of the annotated labels. For example, if an image is annotated with beard, one does not need to see the image to conclude that it is probably also annotated with mustache. More specifically, AU1 (inner brow raiser) and AU2 (outer brow raiser) are known to be dependant as they both intervene in the expression of surprise. Therefore without seeing the image, one can say that if it is annotated with AU1 then it is probably also annotated with AU2. Therefore, it is fairly natural to believe that the exploitation of such dependencies may improve the predictive performance of deep neural networks.

Multi-task Recurrent Neural Networks (MRNN)

The usual method to model inter-task relationship is to impose an arbitrary order on the set of tasks (for now let's use the order of coordinates) and to use the Bayes Chain Rule to decompose the joint distribution of tasks into a chain of conditional distributions:

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} p_\theta(y^t \mid y^{<t}, \mathbf{x}) \tag{2.9}$$

From an architectural point of view, the methods that use this modelling assumption use the same encoding technique as for VMN and feed the encoded representation $f_{\mathbf{W}}(\mathbf{x})$ as initialization $\mathbf{h}^0$ for a recurrent computation process driven by a recurrent cell $g_{\mathbf{V}}$ with parameters $\mathbf{V}$. The objective of this recurrent process is to produce the prediction for each tasks conditioned by the ground truth of all preceding tasks. For that purpose, at step t, it takes ground truth for task $t-1$ along with previous state $\mathbf{h}^{t-1}$ as input and outputs both prediction p^t and the next state $\mathbf{h}^t$. The recurrent loop that underpins the network computational graph can be summarized as follows:

$$\begin{aligned} \mathbf{h}^0 &= f_{\mathbf{W}}(\mathbf{x}), y^0 = 0, \\ o^t, \mathbf{h}^t &= g_{\mathbf{V}}(\mathbf{h}^{t-1}, y^{t-1}), \end{aligned} \tag{2.10}$$

and the predictions for task t are simply computed based on the output $\mathbf{o}^t$:

$$p^t = \mathrm{act}^t(o^t) \tag{2.11}$$

We refer to those architecture as Multi-task recurrent neural networks (MRNN) and a general computational graph is depicted in 2.2.

When it comes to supervising such network, the chain rule development can be plugged into 2.5 as follows:

$$\begin{aligned} \mathcal{L}(\theta) &= \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(\mathbf{y}_i \mid \mathbf{x}), \\ &= \frac{1}{N} \sum_{i=1}^{N} \log \prod_{t=1}^{T} p_\theta(y_i^t \mid \mathbf{y}_i^{<t}, \mathbf{x}), \\ &= \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log p_\theta(y_i^t \mid y_i^{<t}, \mathbf{x}), \\ &= \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \ell(y_i^t, p_i^t), \end{aligned} \tag{2.12}$$

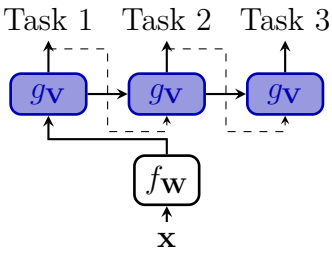


Figure 2.2: MRNN computational graphs. Input features are first fed to a shared encoder to compute a representation that is common to all tasks. Then the representation is given to a recurrent cell (in blue) that recursively output the prediction for each task based on the predictions of previous tasks.

where $\theta = \{\mathbf{W}, \mathbf{V}\}$ regroups all networks parameters.

One of the main differences between MRNN and VMN is that, due to the sequential nature of the training phase, MRNN networks do not directly outputs the distribution of tasks conditionally to the input. Instead, it outputs the distribution of tasks conditioned by previous tasks along with the input image. One of the simplest manner to compute the distribution we want from the distribution that are provided by MRNN is to leverage Monte-Carlo sampling as follows:

$$\begin{aligned} p_\theta(y^t \mid \mathbf{x}) &= \mathbb{E}_{\mathbf{y}^{<t}}[p_\theta(y^t \mid \mathbf{y}^{<t}, \mathbf{x})], \\ &\simeq \frac{1}{L}\sum_{l=1}^{L} p_\theta(y^t \mid \hat{\mathbf{y}}_l^{<t}, \mathbf{x}), \end{aligned} \tag{2.13}$$

where the L trajectories $(\hat{\mathbf{y}}_l)_{l=1}^L$ are computed following algorithm 1.

From a modelling point of view, MRNN is superior to VMN as it is capable of learning inter-task dependencies. Therefore it should intuitively be able to extract more knowledge from training datasets and consequently display better predictive performance than VMN.

However, in practice, the modelling power is not the only factor that affects performance. In particular, the used weight sharing pattern, *i.e.* the choice of which weights are shared across tasks and which weights are not, is known to be of critical importance. Typically, in MRNN, the weights of the predictors are shared across tasks. Similar to what happen with VMN 2.3.2, this sharing pattern may cause task loss competition inside the shared parts and degrade predictive performance.

Algorithm 1 MRNN sampling for inference

Require: Input vector $\mathbf{x}$
1: **for** $l = 1$ **to** L **do**
2: $\quad \hat{y}_l^0 = 0$, $\mathbf{h}_l^{(0)} = f_{\mathbf{W}}(\mathbf{x})$
3: $\quad$ **for** $t = 1$ **to** T **do**
4: $\quad\quad o_l^t, \mathbf{h}_l^t = g_{\mathbf{V}}(\mathbf{h}_l^{t-1}, y_l^{t-1})$
5: $\quad\quad p_l^t = \text{act}^t(o_l^t)$
6: $\quad\quad \hat{y}_l^t \sim \mathcal{B}(p_l^t)$
7: $\quad$ **end for**
8: **end for**
9: **return** L output trajectories $(\hat{\mathbf{y}}_l)$

Furthermore, in the context of recurrent neural networks, it has been identified that the order in which outputs are predicted matters, meaning that it can affect performance. Casted in the multi-task paradigm in which each output of the network is a task, it means that order in which tasks are predicted may impact performance. As a consequence exploring several orders and learning to predict using the best orders could improve recurrent network based multi-task methods performance. For that purpose, we propose Multi-Order Networks (MONET) to jointly learn to predict the tasks along with the order in which they should be predicted.

2.4 Multi-Order Networks (MONET)

An order σ of $[\![1, T]\!]$ is a bijective assignement from $[\![1, T]\!]$ to $[\![1, T]\!]$. In the case of multi-task recurrent networks it maps the network timesteps to tasks. In formal terms, given an order σ, a recurrent network that learns with order σ predicts task $\sigma(i)$ at timestep i. The philosophy of MONET is to try to jointly learns the task along with the order in which they should be predicted.

2.4.1 Jointly learning task order and predictions

The first problem that occurs when attempting to learn the task order is that the space of orders is discrete. This forbids the use of gradient descent with respect to order based variables. To solve that problem we introduce a probabilistic extension of orders that we call soft orders and that we define as T random variables $(s_1, ..., s_T)$ that maps timestep i to task s_i under the following probabilistic

bijective constraints:

$$\begin{aligned} \forall i, j \in [\![1, T]\!], \quad & p(s_i \neq s_j) = 1 \\ \forall i \in [\![1, T]\!], \quad & p(s_i \in [\![1, T]\!]) = 1 \end{aligned} \tag{2.14}$$

Those assumptions simply ensure that sampling from a soft order yields a valid order. Consequently, with this definition, it is straightforward to check that an order σ is also a soft order as the constant random variables $s_i = \sigma(i)$ verify properties 2.14.

To learn soft orders, we represent it by introducing soft order matrices $\Omega \in \mathbb{R}^{T \times T}$ such that $\Omega_{i,j} = p(s_i = j)$ is the probability that task j is treated at timestep i. From 2.14 we can derive the following properties for matrix Ω:

$$\forall i \in [\![1, T]\!], \sum_{j=1}^{T} \Omega_{i,j} = \sum_{j=1}^{T} p(s_i = j) = p(s_i \in [\![1, T]\!]) = 1 \tag{2.15}$$

$$\forall i \in [\![1, T]\!], \sum_{i=1}^{T} \Omega_{i,j} = p(\exists i \in [\![1, T]\!] \mid s_i = j) = 1, \tag{2.16}$$

From those equations we conclude that, for Ω to represent a correct soft order, it must be bistochastic meaning that both lines and columns sum to 1. In particular the matrix of a deterministic order σ, that we denote $\mathbf{M}_\sigma$ verifies:

$$(\mathbf{M}_\sigma)_{i,j} = p(s_i = j) = \begin{cases} 1 & \text{if } j = \sigma(i), \\ 0 & \text{otherwise. } , \end{cases} \tag{2.17}$$

so that there is a single 1 in each row and each columns.

Even with the so-defined soft order matrices, learning the prediction order is still challenging from an optimization perspective. Indeed, using direct gradient descent w.r.t will generate gradient updates that are likely to pull Ω out of the set of bistochastic matrices so that it won't represent a valid soft order anymore.

In a attempt to circumvent this issue, we draw inspiration from the Bhirkov-Von-Newman theorem:

Theorem 1 (Birkhoff-von Neumann's Theorem) *The set of bistochastic matrices of size $T \times T$ is called Bhirkhoff's polytope, the convex hull of all $N = T!$ order matrices $\mathbf{M}_{\sigma_1}, \ldots, \mathbf{M}_{\sigma_N}$. In more formal words, let Ω a bistochastic matrix,*

there exist $\pi_1, \ldots, \pi_N$ *verifying:*

$$\forall i, \in [\![1, T]\!], 0 \leq \pi_i \leq 1, \sum_{i=1}^{N} \pi_i = 1 \tag{2.18}$$

$$\Omega = \sum_{i=1}^{N} \pi_i \mathbf{M}_{\sigma_i} \tag{2.19}$$

Reciprocally, this theorem implies that any convex combination of order matrix is bistochastic. As a consequence, a simple manner to learn a soft order bistochastic matrix Ω is to learn a convex combination $\pi_1, \ldots, \pi_M$ of M order matrices $\mathbf{M}_{\sigma_1}, \ldots, \mathbf{M}_{\sigma_M}$:

$$\Omega = \sum_{i=1}^{M} \pi_i \mathbf{M}_{\sigma_i} \tag{2.20}$$

Then, the only constraint that needs to be maintained to preserve Ω bistochastic nature is that $\pi_1, \ldots, \pi_M$ is a valid convex combination. A simple way to do so is to learn logits $\mathbf{u}_1, \ldots, \mathbf{u}_M$ and to compute π by applying the widely used softmax function on $\mathbf{u}$:

$$\pi = \text{softmax}(\mathbf{u}). \tag{2.21}$$

With that formulation, predicting tasks with soft order **O***mega* is equivalent to predicting using order $\mathbf{M}_{\sigma_i}$ with probability π_i. Therefore, we first predict the tasks using each of M orders. This is done by unrolling the joint distribution using equation 2.9 in orders $\sigma_1, \ldots, \sigma_M$:

$$p_\theta^\sigma(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} p^\sigma(y^{\sigma(t)} \mid \mathbf{y}^{\sigma(<t)}, \mathbf{x}), \tag{2.22}$$

and finally, we compute MONET's final estimate of the joint distribution as a π-based convex combination of each of those distributions:

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \sum_{m=1}^{M} \pi_m p_\theta^{\sigma_m}(y^{\sigma_m(t)} \mid \mathbf{y}^{\sigma_m(<t)}, \mathbf{x}), \tag{2.23}$$

2.4.2 MONET architecture

MONET modelling requires estimating the task joint distribution in M different ways, each way corresponding to an order. To do so, the most natural method is

to use a different recurrent cell to estimate each joint distribution. However, such architectural choice comes with two major drawbacks.

First, similar to what happens with MRNN, sharing the same weights to learn different tasks may cause loss competition in the shared parts and in turn degrade performance. Second, and most important, the number of possible orders for a T tasks problem is $T!$. Consequently, in order to stand a chance to get a correct order in the set of M orders from which MONET learns, M should be as big as possible. Yet, with this sharing pattern, the number of parameters grows linearly with M and quickly forbids the selection of high values of M.

To circumvent both issues, we propose a new recurrent sharing pattern that is adapted to MONET needs. Instead of using a recurrent cell per order, we rather mimic the sharing pattern of VMNS and use a recurrent cell per task. This solves the two latter mentioned limitations. Indeed, with that sharing strategy, a single cell learns a single task. This reduces task loss competition in the shared parts and allow each cell to specialize in the task it is associated to. Furthermore, this setting enable joint learning of an arbitrary number of orders as the number of parameters is no longer linear in M.

From a formal point, we give ourselves a set of T recurrent cells $g_{\mathbf{W}_1}, \dots, g_{\mathbf{W}_T}$ and M orders $\sigma_1, \dots, \sigma_M$. Then for order σ_m, we initialize a similar recurrent process as for MRNN:

$$\mathbf{h}_m^0 = f_{\mathbf{W}}(\mathbf{x}), y_m^0 = 0, \tag{2.24}$$

and at each timestep t, and for each order σ_m, the cell of task $\sigma_m(t)$ $g_{\mathbf{W}^{\sigma_m(t)}}$ is used to predict task and propagate the hidden state based on the previous state $\mathbf{h}_m^{t-1}$ and the ground truth $y^{\sigma_m(t-1)}$ for the task that has been treated at the previous timestep:

$$o_m^t, \mathbf{h}_m^t = g_{\mathbf{W}^{\sigma_m(t)}}(\mathbf{h}_m^{t-1}, y^{\sigma_m(t-1)}). \tag{2.25}$$

Then prediction is computed by simply applying the appropriate activation on top of $\mathbf{o}_m^t$:

$$p_m^t = \text{act}^t(o_m^t). \tag{2.26}$$

A summary of MONET computational graph is depicted in graph 2.3.

When it comes to the supervision of MONET, we follow the same Maximum

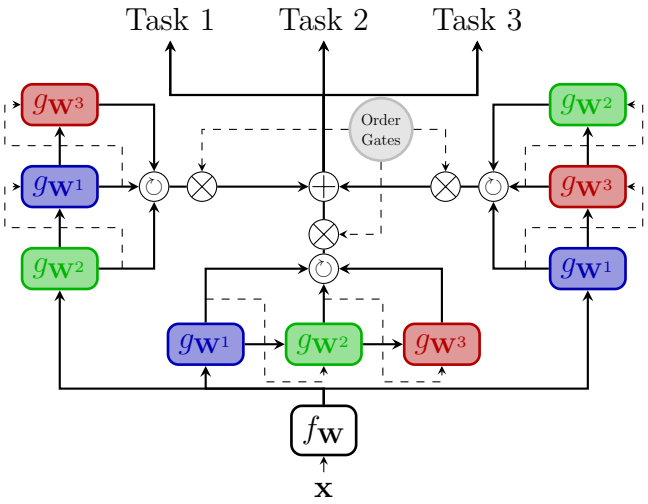


Figure 2.3: MONET computational graphs. MONET uses a recurrent cell per task (represented in blue/green/red). Input features are first fed to a shared encoder to compute a representation that is common to all tasks. Then the representation is given to M recurrent process which estimate the joint distribution of task in M different orders. Order gates uses a convex-combination of those M distributions to learn which orders to choose the best orders and in turn improve performance.

Likelihood Principle as for VMN and MRNN and the final loss is therefore:

$$\begin{aligned}
\mathcal{L}(\theta, \pi) &= -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\mathbf{y}_i \mid \mathbf{x}), \\
&= -\frac{1}{N}\sum_{i=1}^{N} \log \sum_{m=1}^{M} \pi_m p_\theta^{\sigma_m}(\mathbf{y}_i \mid \mathbf{x}), \\
&= -\frac{1}{N}\sum_{i=1}^{N} \log \sum_{m=1}^{M} \pi_m \exp \sum_{t=1}^{T} \log p_\theta^{\sigma_m}(y^{\sigma_m(t)} \mid \mathbf{y}^{\sigma_m(<t)}, \mathbf{x}), \\
&= -\frac{1}{N}\sum_{i=1}^{N} \log \sum_{m=1}^{M} \pi_m \exp \sum_{t=1}^{T} l(p_{m,i}^t, y_i^{\sigma_m(t)}).
\end{aligned} \tag{2.27}$$

where $\theta = \{\mathbf{W}, \mathbf{W}^1, \ldots, \mathbf{W}^T\}$ regroups all networks parameters and π are the order selection coefficients which depend from logits $\mathbf{u}$.

2.4.3 A deeper look into MONET order selection strategy

In this section, we provide theoretical intuitions about MONET order selection. In particular, we confirm that the raw MONET order selection mechanism tends to allocate weights on the order with the lowest loss. For that purpose, we split the loss in 2.27 in both global and order-based element-wise losses:

$$\begin{aligned} \mathcal{L}_i(\theta, \pi) &= -\log \sum_{m=1}^{M} \exp(\log \pi_m - \mathcal{L}_i^{\sigma_m}(\theta)), \\ \mathcal{L}_i^{\sigma}(\theta) &= -\log p_\theta^{\sigma}(\mathbf{y}_i \mid \mathbf{x}_i). \end{aligned} \tag{2.28}$$

MONET order selection is based on the variation of π, which itself is underpinned by gradient updates on order logits $\mathbf{u}$. Consequently, we compute the loss gradient element-wise. More precisely for $i \in [1, N]$, for $m \in [1, M]$:

$$\frac{\partial \mathcal{L}_i(\theta, \pi)}{\partial u_m} = \pi_m(1 - \exp(\mathcal{L}_i(\theta, \pi) - \mathcal{L}_i^{\sigma_m}(\theta)), \tag{2.29}$$

and:

$$\mathcal{L}_i^{\sigma_m}(\theta) \leq \mathcal{L}_i(\theta, \pi) \iff \frac{\partial \mathcal{L}_i(\theta, \pi)}{\partial u_m} \leq 0. \tag{2.30}$$

The latter equivalence implies that, in the case of element by element loss minimization, coefficient π_m increases if the loss associated to its order is inferior to the global loss. In a more realistic scenario, optimization is performed by batch of size B. The gradient of loss $\mathcal{L}$ on a batch is then:

$$\begin{aligned} \frac{\partial \mathcal{L}_B(\theta, \pi)}{\partial u_m} &= \frac{1}{B} \sum_{i=1}^{B} \frac{\partial \mathcal{L}_i(\theta, \pi)}{\partial u_m}, \\ &= \frac{1}{B} \sum_{i=1}^{B} \pi_m(1 - \exp(\mathcal{L}_i(\theta, \pi) - \mathcal{L}_i^{\sigma_m}(\theta))). \end{aligned} \tag{2.31}$$

Then, by noting that $f : x \mapsto 1 - exp(x)$ is a concave function, it follows that:

$$\frac{\partial \mathcal{L}_B(\theta, \pi)}{\partial u_m} \leq \pi_m(1 - \exp(\mathcal{L}_B(\theta, \pi) - \mathcal{L}_B^{\sigma_m}(\theta))) \tag{2.32}$$

and therefore:

$$\mathcal{L}_B^{\sigma_m}(\theta) \leq \mathcal{L}_B(\theta, \pi) \Rightarrow \frac{\partial \mathcal{L}_B(\theta, \pi)}{\partial u_m} \leq 0. \tag{2.33}$$

Consequently, orders whose losses are the lowest on a batch get positive updates on their order selector coefficients. As a consequence, using raw MONET order selection results in selecting the order whose joint estimation best fits the train set. Though using order with the best training loss could seem a fairly intuitive choice, it comes with its bags of optimization problems. Indeed, at the beginning of the training the order losses tend to depend more on the network initialization rather than from their associated order respective performance. Consequently, order selection is a very noisy process during the first epochs and is likely to assign weights to a random order rather than to the orders with the best performance. Such early random selection is a problem because weights allocation is prone to snowballing, i.e to keep allocating more and more weights to a previously selected order. Indeed, if for a train step, the loss $\mathcal{L}^{\sigma_i}$ is lower than the other order losses, then, according to 2.33, MONET will allocate more weights to order σ_i by positively modifying coefficient π_i. At the next train step, this will generate a higher contribution of loss $\mathcal{L}^{\sigma_i}$ to the global loss as π_i weights the contribution of the loss of each order. Consequently, it will encourage further specialization of the task modules into predicting in order i by keeping on lowering $\mathcal{L}^{\sigma_i}$ and so on until $\pi_i = 1$.

In a nutshell, raw MONET order selection risks selecting a random order at the beginning of the training and snowballing on it hence never getting a chance to further explore the space of possible orders.

Warm-up to step away from bad starting points:

To circumvent this issue we draw inspiration from [80] to propose warm-up phase. The principle of the warm-up is simple. It consists in avoiding early random order selection by freezing the optimization of order logits $\mathbf{u}$ during the first n epochs.

By doing so, the network starts by training all orders during n epochs without modifying the contribution of each order losses. Therefore when order logits optimization is triggered at epoch $n+1$, order selection is no longer random but guided by each order performance estimated during the n first epoch. By preventing random order selection, warm-up improve the quality of the selected orders which in turn improve performance. Figure 2.4 shows warm-up learning rate schedule.

Order dropout to avoid order selection snowball:

Warm-up helps improve order selection by preventing early random order weight allocation. However, it doesn't prevent the order selection from snowballing on the first order it selects. Yet, this snowball effect is a problem as it yields increased

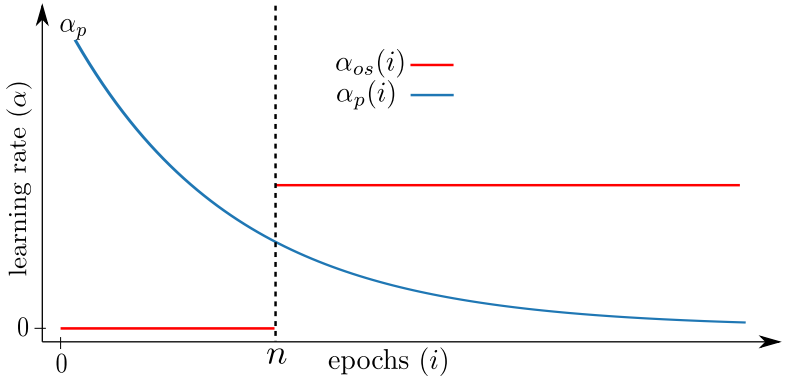


Figure 2.4: Warm up learning rate schedule. α_p represents the learning rate in the predicting part of the network while α_{os} is the order selector learning rate. The order selector stay frozen for the n first epochs so that the predictor estimates the performance of each orders. At the end of this exploration phase, the order selector coefficients are released and uses those estimations to allocate weight to the orders with the best performance.

risks of getting trapped in a suboptimal order. Furthermore, it completely neglects the benefits of training several orders in parallel as, in the end, it only optimizes a single order. In an attempt to avoid this pitfall, we inspire from usual dropout strategy [65] to propose order dropout. At each train step, order dropout selects a random subset of k orders among the set of M orders and zeros-out the contribution of the $M - k$ remaining order selection coefficient:

$$\tilde{\pi}_{m,i} = \frac{t_{m,i} \exp(u_m)}{\sum_{l=1}^{M} t_{m,l} \exp(u_l)}, \tag{2.34}$$

where t_m^i is a randomly sampled binary mask with k ones and $(M - k)$ zeros.

For inference, we use the same method as for usual dropout [65] i.e we multiply each $\exp(u_m)$ by its probability of presence $p(k, M)$:

$$\pi_m = \frac{p(k, M) \exp(u_m)}{\sum_{l=1}^{M} p(k, M) \exp(u_l)} = \frac{\exp(u_m)}{\sum_{l=1}^{M} \exp(u_l)}. \tag{2.35}$$

With such strategy, the order with the lowest loss (which is therefore the order that would be systematically selected in MONET raw order selection) is not always included in the k selected orders. Therefore it does not get systematically reinforced by successive positive gradient updates of its associated order logits. In

that extent, order dropout short-circuits the snowballing effect, forces MONET to spread weight allocation and consequently encourage good predictive performance in several orders.

This strategy has several benefits. First, by short-circuiting the snowball effect, it provides MONET with a possibility to escape from suboptimal order selection. As a consequence, it improves the order space exploration, allow better order selection and improves performance. Second, it reduces computational cost. Indeed, with MONET raw order selection, M recurrent processes have to run in parallel which is costly. Order dropout on the contrary only requires k recurrent processes to run in parallel. As a consequence, using order dropout reduces MONET cost with fixed M or opens the possibility of further M increase which also improves order space exploration. Yet,this order space exploration enhancement comes at a cost. Indeed, similar to what happens with classic dropout [65], the lower the number k of orders trained simultaneously, the more epochs it takes for MONET to reach full performance.

Finally, we believe that, order dropout may help reduce overfitting. Indeed, by randomly selecting k orders for each example it encourages the task-wise cells toward keeping good predictive performance in several orders. Consequently, it prevents MONET from learning features that are too specific to each order which may consequently improve its generalization performance.

2.4.4 Inference with MONET

Algorithm 2 MONET sampling for inference

Require: Input vector $\mathbf{x}$
for $r = 1$ **to** R **do**
  $s_r \sim \mathbf{S}$
  for $l = 1$ **to** L **do**
    $\hat{y}^0_{r,l} = 0, \mathbf{h}^0_{r,l} = f_{\mathbf{W}}(\mathbf{x})$
    for $t = 1$ **to** T **do**
      $o^t_{r,l}, \mathbf{h}^t_{r,l} = g_{\mathbf{W}^{\sigma_r(t)}}(\mathbf{h}^{(t-1)}_{r,l}, \tilde{y}^{(t-1)}_{r,l})$
      $p^t_{r,l} = \mathrm{act}^t(o^t_{r,l})$
      $\hat{y}^t_{r,l} \sim \mathcal{B}(p^t_{r,l})$
    end for
  end for
end for
return LR output trajectories $(\hat{\mathbf{y}}_{r,l})$

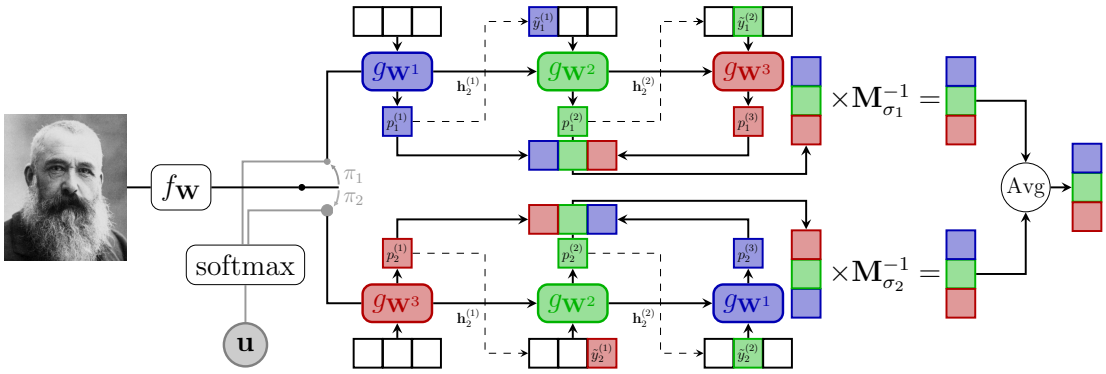


Figure 2.5: MONET inference overview with $T = 3$ tasks and $M = 2$ orders $\sigma_1 = [1, 2, 3]$ and $\sigma_2 = [3, 2, 1]$. All visual elements with the same color are relative to the same task (Blue for task 1, green for task 2 and red for task 3), rounded rectangles represent prediction modules while straight corner rectangles represent their predictions. At inference time, MONET predicts all tasks in L different orders sampled from order selector π. The final prediction is then the average of those L predictions. Consequently, the tasks are predicted in orders that are learned at train time. Hence it better captures inter-task dependencies and yields enhanced multi-task performance.

Similar to what happens with MRNN, MONET training loop does not provide direct access to the probability distribution for each task. Instead, for each order σ_m, MONET provides the chain-rule based successive conditional distribution associated with σ_m. In order to exploit those distributions we proceed as for MRNN. Yet we stand aside from it by leveraging the order selection information stored in convex combination π. More precisely, we begin by sampling R orders from a discrete random variable $\mathbf{S}$ with distribution π. For each of those orders, we compute each task distribution in that order using the same algorithm as for MRNN. Finally, we obtain the final distribution prediction for each task by averaging the so-obtained order-based distribution estimation. From a formal standpoint, the final prediction is obtained as follows:

$$\begin{aligned} p_\theta(y^t \mid \mathbf{x}) &= \mathbb{E}_{\mathbf{S}}[p_\theta(y^t \mid \mathbf{x}, \mathbf{S})], \\ &= \mathbb{E}_{\mathbf{S}}[\mathbb{E}_{\mathbf{y}^{\sigma_{\mathbf{S}}(<\sigma_{\mathbf{S}}^{-1}(t))}}[p_\theta^{\sigma_{\mathbf{S}}}(y^t \mid \mathbf{x}, \mathbf{y}^{\sigma_{\mathbf{S}}(<\sigma_{\mathbf{S}}^{-1}(t))})]], \\ &\simeq \frac{1}{LR} \sum_{r=1}^{R} \sum_{l=1}^{L} p_\theta^{\sigma_{s_r}}(y^t | \hat{\mathbf{y}}_{r,l}^{\sigma_{s_r}(<\sigma_{s_r}^{-1}(t))}, \mathbf{x}), \end{aligned} \tag{2.36}$$

where $(\hat{\mathbf{y}}_{r,l})$ for $r \in [1, R], l \in [1, L]$ are sampled using algorithm 2 and figure 2.5

shows the computational graph of MONET in inference.

2.5 Experiments

In this section, we experimentally validate the efficiency of MONET on both toy and real world Action Unit Detection experiments. In particular, in section 2.5.1 we empirically demonstrate that MONET is able to find the correct order in a controlled toy environment. Those toy experiments are complemented in section 2.5.2 by a careful study of MONET predictive performance on several subset of CelebA attributes that are chosen for the wide range of dependency types they display. The main outcome of those experiments is that no matter the type of dependency displayed MONET is always at least even with the best multi-task baseline (between VMN and MRNN). As MONET shares task-wise sharing pattern with VMN and Bayes Chain Rule based dependency modelling with MRNN, we conclude that MONET is an efficient hybrid between those two methods. Last but not least, in the last section 2.5.3 we show that MONET outperforms state-of-the-art in AU detection on both DISFA and BP4D.

2.5.1 A deeper look into MONET order selection

Toy Dataset

In order to demonstrate MONET's ability to find the correct order, we first need to find a dataset for which the correct order is known. For that purpose we design the following 2D dataset :

$$\begin{aligned} \mathbf{X} &\sim \mathcal{U}([-1;1]^2), \\ \forall t \in [1,T] : Y_T^t &= \mathbf{1}_{\bigcup_{i=1}^{2^t}[b_{2i}^t, b_{2i+1}^t]}(X^{(1)}), \end{aligned} \tag{2.37}$$

where $b_i^t = -1 + \frac{i-1}{2^t}$.

From a more intuitive perspective, this dataset recurrently splits the $[-1,1]^2$ square in a vertical fashion. Task 1 simply splits the square in 2 zones by drawing a vertical line in the middle of it and by considering that all examples left to that line are positive and all examples right to that line are negative. Then, in a recurrent manner, task $n+1$ splits each of the 2^n zones formed by task n by drawing a vertical line in the middle of it and by considering that all examples in the zone that are left to the line are positive and all examples right to the line are negative. With that formulation Task n of the toy dataset is a classification problem with

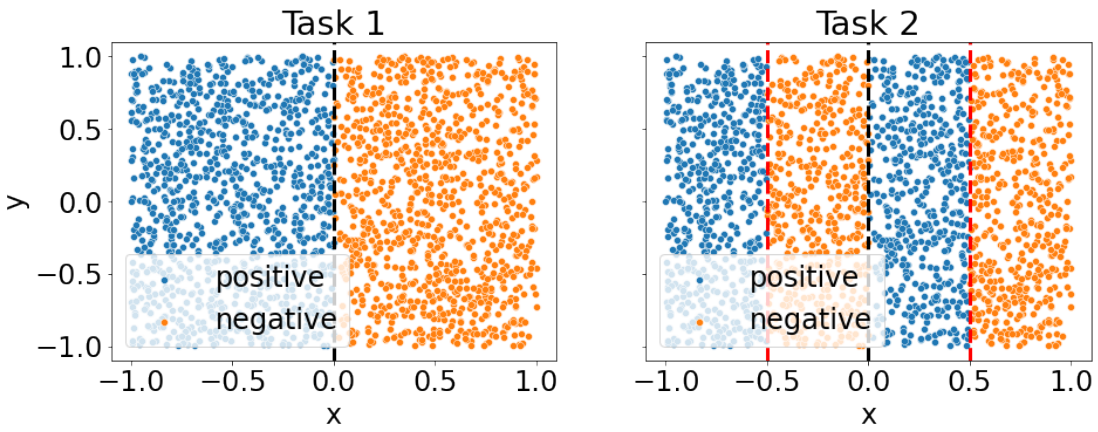


Figure 2.6: Distribution of 2000 examples sampled from our toy dataset with $T = 2$. Given task 1, *i.e.* given the left-right positioning of the sample w.r.t to the black dashed boundary, task 2 is simplified into a single linear boundary learning problem represented by red boundaries.

$2^n - 1$ vertical boundary. Figure 2.6 provides a graphic representation of the toy dataset.

The intuition behind the design of this toy is the following : Task 1 is easy to predict as it consists in a single linear boundary problem. Furthermore, conditionned on the prediction for task 1 (*i.e.* on weither the example is left or right of the vertical line $x = 0$), the prediction for task 2 is reduced to a single linear boundary classification problem. Indeed, suppose for example that the example is at the left side of the figure (task 1 = 1), predicting task 2 simply consists in determining weither the example is on the left side or on the right side of task 1 left zone. In that extent conditionning by task 1 reduced task 2 problem from a 3 vertical boundary classification problem to a single boundary classification problem. Reciprocally, conditionning the prediction of task 1 using the prediction of task 2 doesn't reduce the number of linear boundary in task 1 classification problem. For example positive examples for task 2 can be either positive or negative. As a conclusion, from a learning perspective, it is intuitively better to process task 1 first and to condition the prediction of task 2 using the result for task 1.

More generally, we can reuse the same reasonning and notice that conditionning by task 1, ..., task $n - 1$ locate an example in one of the 2^{n-1} zones formed by task $n-1$ and that predicting task n with that information is simply determining weither the example is left or right of this zone. Reciprocally conditionning task n by one of the following tasks do not reduce the number of linear boundaries in task n classification problem. Consequently, the best order for the toy dataset is

intuitively the identity order.

Implementation details

For all our experiments on the toy dataset, we sample 500, 250 and 250 examples for train, validation and test respectively. As far as the architecture is concerned, we use a shared encoder that consists of four dense layers with 64 units and ReLU activation. Prediction heads for VMNC and VMNS consist in dense layers with 64 units. Both MRNN and MONET employ GRU cells with 64 units and $L = 20$ orders. All networks are trained by applying 500 epochs with Adam [26], batch size 64 with an exponentially decaying base learning rate $5e{-}4$ and $\beta = 0.99$. MONET order selector is trained with Adam with learning rate 0.005. Other MONET related parameters (dropout k, warmup n) are determined by hyperparameter tuning on the validation set.

Order importance in the toy dataset

In order to validate the intuition about the identity order being the best order on the toy dataset, we train single order instance of MONET with each of the $T!$ possible orders. Figure 2.7 shows the performance of those single order training as a function of the frobenius norm between the trained order σ and the identity order $\mathbf{I}_T$ with $T = 1, 2, 3$. For all considered values of T, we first notice that best performance are obtained for $\|\Omega - \mathbf{I}_T\|_F^2 = 0$ which only happens when $\Omega = I_T$ *i.e.* when the selected order is the identity.

Furthermore, we observe that MONET performance decrease as selected order σ moves away from the identity order (in frobenius norm). Therefore, this experiment validates that in the toy setting (a) order matters which was already claimed in [80] (b) that the best order is the identity.

Therefore now that we are provided with a toy dataset on which we know the best order, we can test (and validate) MONET's ability to retrieve the correct order in the toy dataset.

MONET order selection validation

For that purpose we use the toy dataset with $T = 5$, and compare the performance of different versions of MONET order selection in term of both mean accuracy and number of successful order selection over 10 runs. Table 2.1 reports the result of those experiments.

First, the worst performing method, that we called Hard Selection is an implementation of [80] order selection mechanism. It first explores the performance of

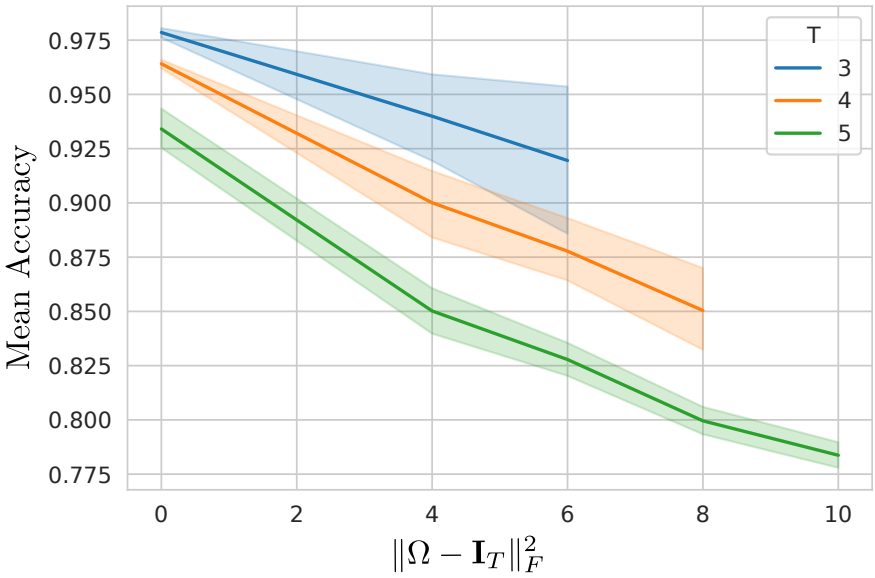


Figure 2.7: Performance of MONET trained with a single order σ ($M = 1, k = 1$) as a fonction of the frobenius distance between σ and the identity order. Accuracy scores are averaged over 10 runs.

Warm up	Dropout	Hard Selection	Mean Accuracy	Finds $\mathbf{I}_T$?
✗	✗	✓	74.2	0/10
✗	✗	✗	89.2	8/10
✓	✗	✗	89.3	9/10
✗	✓	✗	89.8	10/10
✓	✓	✗	**90.1**	10/10

Table 2.1: Comparison of different order selection mechanism for $T = 5$ and $M = 120$. Performance are averaged over 10 runs.

all orders in a phase that is similar to our warm-up and then samples a single order using a distribution in which each order has a probability that is proportional to its performance (measured using training losses of each order). Table 2.1 shows that this order selection mechanism always misses the correct order and therefore gets poor performance. In fact, we believe that such sampling strategy is sensitive to performance measurements noise and is therefore likely to result in selecting a sub-optimal order. By smoothly learning its order selection coefficients all along the training phase, MONET gets a better estimation of each order performance, selects better orders and significantly improve performance.

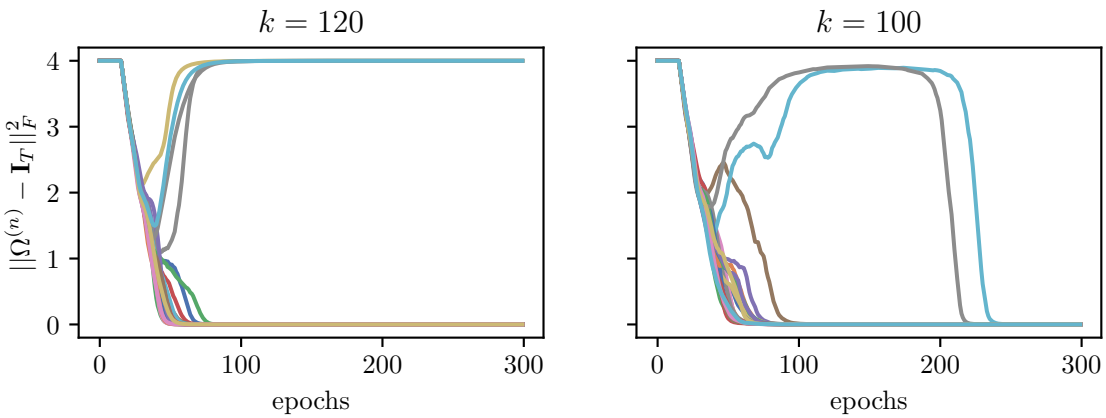


Figure 2.8: Evolution of the Frobenius distance between MONET soft order and the toy correct order for 20 MONET training with different values of k. Each line represent the soft order trajectory w.r.t the identity matrix for a different training of MONET.

Second, table 2.1 shows that MONET order selection refinements are efficient. Warm-up helps MONET to find better orders by providing an exploration phase to estimate each order performance before making a choice. However it doesn't prevent the order selection from snowballing, *i.e.* to keep allocating weights to the first selected order. It explains that adding warm up to MONET only provides a tiny boost in performance.

Dropout on the other hand, short-circuits the snowball effect by forcing the network to train on randomly selected orders: it allows MONET order selection mechanism to smoothly deviate from any previous choice of order. Indeed, figure 2.8, shows that contrary to its standard version ($k = 120, M = 120$), the dropout version of MONET ($k = 100, M = 120$) is able to recover the correct order even if it started with a bad guess. Combined warmup and dropout provide MONET with an initial order guess that is likely to be good, and the ability to move away from this guess if needed. Those two ingredients result in better order selection which, in turn, implies better performance.

Setting number of orders M and order dropout k

In Figure 2.9, we investigate the influence of both M and k. First, it shows that the performance of MONET without order dropout (red plot) are increasing with the number of randomly sampled order M. This is fairly logical. Indeed, the

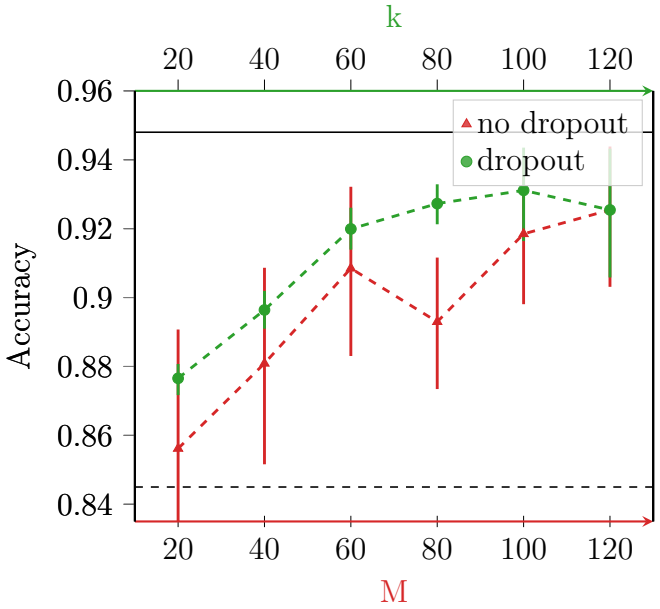


Figure 2.9: MONET performances comparaison for toy dataset with $T = 5$ between: (in red) dropout less version (k=M) for different values of M and (in green) dropout version with M=T! and different values of k. Dashed and full black lines are respectively the random order selection baseline and the oracle mean performances. Means and standard deviations are computed on 10 runs.

higher M, the more likely it is to get a good order in the set of randomly selected orders, the better the performance are.

Second, it shows that on the toy dataset, for equal value of k, it is always better to set M as big as possible. A possible explanation for this is that high values of M increase the probability of getting good orders in the set of MONET trained orders.

Lastly, it demonstrates the exploration/exploitation trade-off aspect of k by showing its influence with constant number of orders $M = 120$ (green plot). For low values of k (from $k = 20$ to $k = 60$), the set of simultaneously trained orders is too small which slows down MONET's ability to converge toward the exploitation of a given order and in turn degrades performance w.r.t to $k = 120, M = 120$. For fairly high values of k (typically 100), dropping orders in training allow to avoid the order selection snowball effect, hence getting better selected orders and better performance than $k = 120, M = 120$ while being less costly from a computational perspective.

Accuracy	Task 1	Task 2	Task 3	Task 4	Task 5	Mean
VMNC	98.7	97.6	92.4	65.2	54.0	81.5
VMNS	98.7	97.5	72.8	51.5	52.9	74.9
MRNN	93.7	93.6	80.1	60.5	51.3	75.9
MONET	99.4	98.3	95.8	90.0	68.2	90.1
Oracle†	99.6	98.6	96.1	92.1	80.6	93.4

Table 2.2: Multi-task baselines comparaison on toy dataset with $T = 5$ tasks. †: oracle predictor with correct order.

Comparison with baseline methods

Finally, Table 2.2 shows relative performance of MONET w.r.t multi-task baselines. MONET displays significantly better performance than all other considered methods, significantly narrowing the gap with the oracle performance.

Conclusion

In this section, we demonstrated in a controlled benchmark where the optimal task chaining order is known that (a) MONET was able to consistently retrieve said order, and (b) that thanks to its joint order selection mechanism and task-specific recurrent cell sharing architecture, backed by the proposed order dropout strategy, MONET was able to consistently outperform other multi-task baselines, getting closer to an oracle predictor using the optimal order. We now consider real-world applications with potentially more complex inter-task dependencies and less clear ordering patterns.

2.5.2 MONET in different dependency situations

In this section, we aim at investigating MONET performance in different dependency settings. For that purpose, we use subsets of the CelebA datasets 40 attributes that we choose for the wide range of dependency setting they display.

CelebA is a widely used database in multi-task learning. It is composed of $\approx 200k$ celebrity images annotated with 40 different facial attributes. For performance evaluation, we measure accuracy score using the classic train ($\approx 160k$ images), valid ($\approx 20k$ images) and test ($\approx 20k$ images) partitions for 5 different subsets of 5 attributes each. Those subsets were chosen to test MONET behaviour in different dependency settings:

- *gender*: with moustache, beard, lipstick, heavy makeup and sex detection. Those attributes display statistical dependencies. For example a beard often implies a moustache.
- *accessory*: with earrings, eyeglasses, necklaces and neckties detection.
- *beauty*: with arched eyebrows, attractiveness, high cheekbones, rosy cheeks, and oval faces detection.
- *haircut*: with baldness, black, blond, brown and gray hair detection. Those attributes are mutually exclusive.
- *miscellaneous*: with 5 o'clock shadow, pointy nose, mouth slighlty open, oval face, and weither the subject is young or not. Those attributes are independant *a priori*.

Both *accessory* and *beauty* were choosen for their lack of clear a priori on the type of dependencies that bind the tasks together.

Implementation Details

For all our CelebA experiments, we make use of an Inception resnet v1 encoder pretrained on VGGFace2 [4] along with dense layers with 64 units as prediction heads for VMNC and VMNS. Both MRNN and MONET use GRU cells with 64 units and $L = 20$ orders. Networks are trained with 30 epochs using AdamW [37] with learning rate/weight decay set to 0.0005 and exponential decay ($\beta = 0.96$). For MONET hyperparameters, we use $M = 5! = 120$ permutations and order dropout $k = 32$.

Comparaison of MONET with other Multi-task Baselines.

Table 2.3 draws a comparison between different multi-task methods for attribute detection on CelebA. On the one hand, there is no clear winner between the two VMN versions: for instance, VMNS performs better on the *gender* and *accessories* subsets while VMNC performs better on *haircut* and *misc*. Those performance discrepancies may result in practical difficulties to find an all-around, well performing architecture, as echoed in [46]. Furthermore, MRNNs gets consistently outperformed by at least one of the VMN methods. In fact, we believe that MRNN recurrent cell sharing across tasks leads to early conflicts between task-associated gradients and prevents it from taking full advantage of its theoretically better inter-task relationship modelling. Additionally, random task order sampling may prevent MRNN from properly learning and hurt its predictive performance.

Gender	H. Makeup	Male	Mustache	No Beard	W Lipstick	Avg.
VMNC	88.0	95.8	96.6	95.4	90.2	93.2
VMNS	90.2	97.0	96.7	95.5	93.9	94.6
MRNN	89.9	96.9	96.7	95.6	93.6	94.5
MONET	90.5	97.5	96.8	95.8	93.9	**94.9**
Accessories	Eyeglasses	W Earrings	W Hat	W Necklace	W Necktie	Avg.
VMNC	99.4	88.5	98.4	86.8	95.8	93.8
VMNS	99.5	89.5	98.6	87.1	96.7	**94.3**
MRNN	99.5	89.6	98.7	87.1	96.6	**94.3**
MONET	99.2	89.9	98.5	87.3	96.8	**94.3**
Haircut	Bald	Black Hair	Blond Hair	Brown Hair	Gray Hair	Avg.
VMNC	98.6	88.3	95.3	88.0	98.0	93.6
VMNS	98.6	86.2	95.2	86.9	97.9	93.0
MRNN	98.6	86.7	95.2	86.2	97.8	92.9
MONET	98.7	88.3	95.4	88.0	98.0	**93.7**
Beauty	A Eyebrows	Attractive	H ChBones	R Cheeks	Oval Face	Avg.
VMNC	82.8	81.1	86.8	94.2	73.7	83.7
VMNS	82.5	81.0	86.5	94.4	74.0	83.7
MRNN	82.8	80.6	86.3	94.6	74.1	83.7
MONET	82.8	81.5	86.9	94.7	74.4	**84.1**
Misc.	5 Shadow	P Nose	M S Open	Oval Face	Young	Avg.
VMNC	93.6	76.6	93.5	74.2	86.4	84.9
VMNS	93.6	76.7	84.8	74.1	86.8	83.2
MRNN	93.6	76.1	93.2	73.6	85.6	84.4
MONET	94.2	76.7	93.6	74.3	87.0	**85.2**

Table 2.3: Comparaison of MONET with multi-task baselines on several attributes subsets of CelebA.

The take home message of those experiments is that finding the best multi-task architecture is difficult to do a priori because the performance depend on a lot of factors (dependency modeling, weight sharing, model expressivity, etc). On the other hand, MONET shows consistently better performances than both VMN as well as MRNN on every subset, due to both its task-wise modular weight sharing strategy and its order selection mechanism that, in turn, allows to correctly model inter-task dependencies. As a consequence, MONET is a good a priori choice for multi-task problems for a wide range of dependency settings.

Visualization of MONET order matrices

Figure 2.10 depicts two soft-order matrices extracted at the end of MONET training on the *gender* subset. First, those two matrices are very similar, showing that MONET order selection mechanism is relatively stable across several networks and order selector initializations. Second, it seems that MONET learns to process easy tasks in priority and uses the result of those easy tasks to condition the prediction on harder ones. For example, it typically learns to predict *beard* (which is more visible and therefore easier to predict) before predicting *mustache* and *lipstick* (which has a very characteristic color) before *heavy makeup* (which exhibit more variability and is fairly subjective). From an intuitive point of view, such learned order is fairly reasonable. Indeed, predicting easy tasks earlier reduces the chances of propagating prediction mistakes along the processing chain and in turn improves performance.

Conclusion

In this section we investigated MONET performance w.r.t other multi-task baselines in several dependency settings based on hand-crafted subsets of the CelebA datasets. The main conclusion of those investigations is that finding the multi-task architecture that is most appropriate to a problem is difficult to do a priori because performance may depend on a lot of factors. However we showed that whatever the considered dependency setting, MONET was always at least even with the best perform multi-task baseline. Therefore MONET is a successful hybrid between VMN and MRNN and a handy default choice for multi-task learning problems.

2.5.3 MONET for Action Units detection

In the following section, we study MONET's behavior in Real World Action Unit detection problems on two Action Unit datasets: BP4D and DISFA. First, we

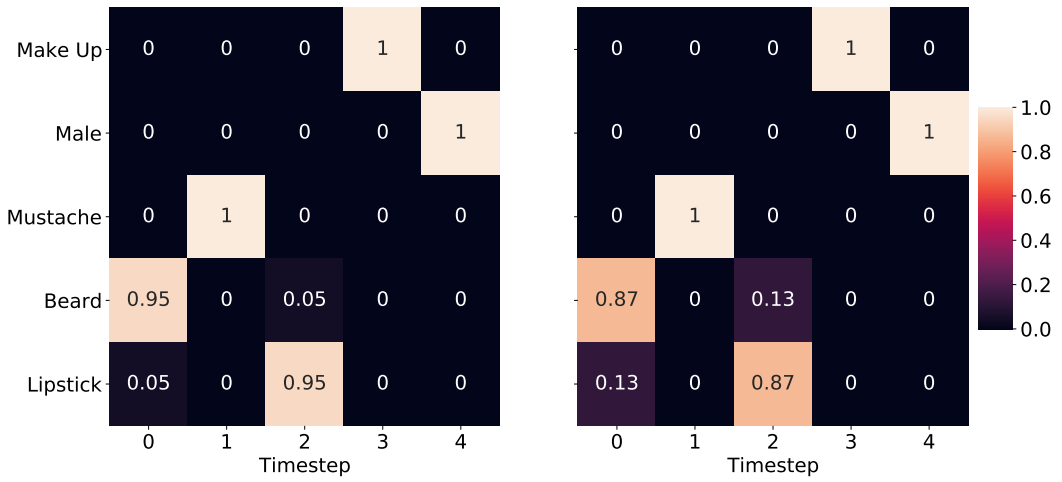


Figure 2.10: Two soft-order matrices extracted at the end of two different MONET training on CelebA *gender* subset.

start by showing that MONET task-wise sharing pattern helps it to better locate the face zones that are important for each AU. Second, similar to what we did for CelebA, we provide intuitions about the orders that MONET finds. Then we demonstrate that MONET achieves state-of-the-art performance on both DISFA and BP4D. Finally, we take a closer look at MONET predictions in order to pinpoint its strengths and weaknesses and guide further research in Action Unit Detection.

Action Unit datasets

BP4D is a dataset for facial action unit detection. It is composed of approximately $140k$ images featuring 41 people (23 female, 18 male) with different ethnicities. Each image is annotated with the presence of 12 AU. For performance evaluation, we follow related work strategy that is to report F1-Score on all 12 AUs using a subject exclusive 3-fold cross-validation with publicly available fold repartition from [59] and [60].

DISFA is another dataset for facial action unit detection. It contains 27 videos for $\approx 100k$ face images. Those images were collected from 27 participants and annotated with 12 AUs. Originally, each AU label is an intensity score ranging from 0 to 5.. In detection, labels with an intensity score higher than 2 are considered

positive [101]. Similarly to BP4D, the performance evaluation protocol consists in measuring the F1-Score for 8 AUs using a subject exclusive 3-fold cross validation.

Implementation Details

Action Units are relatively short events. Therefore video-based datasets such as DISFA and BP4D display a data imbalance problem [60] that may act as a barrier to efficient learning. To circumvent this problem, we use the same biais initialization as in [30] to stabilize gradient updates at the beginning of the training. This simply consists in setting the last bias of the t-th AU (just before the sigmoid) to :

$$b^t = \log \frac{f^t}{1 - f^t} \tag{2.38}$$

where f^t is the frequency of the t-th AU in the training set. The effect of this change is that, at the beginning of the training, the average network prediction is close to $S(b^t) = f^t$ that is the mean prediction we expect on the train dataset.

Then, similar to [60], we apply per AU loss weighting [60] that weight the loss associated to each AU with its inverse occurence rate. This way, AU with low frequencies are not left away in the learning process. For a given order σ_m, the associated loss becomes :

$$\mathcal{L}^{\sigma_m}(\theta) = \sum_{t=1}^{T} w^{\sigma_m(t)} \text{BCE}(y^{\sigma_m(t)}, p_m^t), \tag{2.39}$$

Finally, we further adapt to the AU evaluation protocol that measure F1 score by adding a Dice score contribution ([60, 23]) to the final loss:

$$\text{DICE}^{\sigma_m}(\theta) = \sum_{t=1}^{T} w^{\sigma_m(t)} (1 - \frac{2y^{\sigma_m(t)} p_m^t + \epsilon}{(y^{\sigma_m(t)})^2 + (p_m^t)^2 + \epsilon}), \tag{2.40}$$

For BP4D, we use an Inceptionv3 backbone pretrained on imagenet on top of which we put a MONET instance with $M = 512$ and $k = 128$. The order selection part of MONET is trained using Adam with warmup $n = 5$ epochs and constant learning rate $5e - 3$ while the rest of the network uses AdamW optimizer with learning rate/weight decay set to $1e-4$ and exponential decay $\beta = 0.99$. Batchsize is set to 16, number of epochs is 2 and dice loss coefficient is $\lambda = 0.5$.

For DISFA, we make use of the same encoder as for CelebA and retrain with AdamW learning rate $5e - 4$ with exponential decay $\beta = 0.96$, and batchsize 64. All other parameters stay the same as for BP4D.

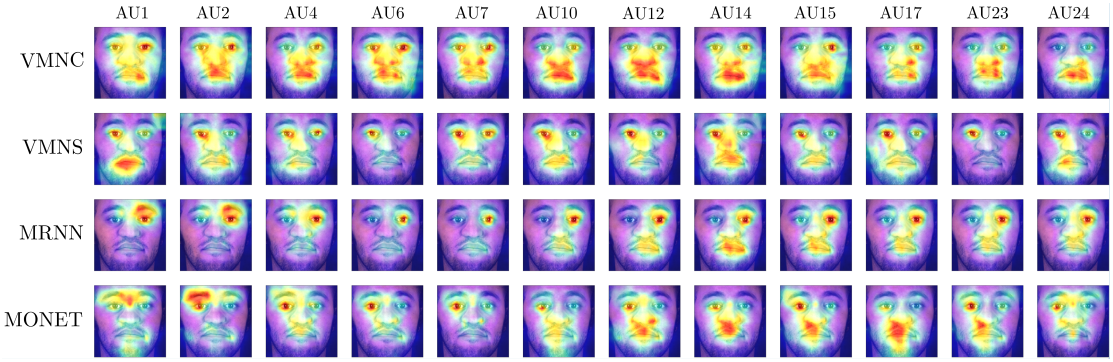


Figure 2.11: AU-wise attribution maps for multi-task baselines as well as MONET. While other architectures may struggle to correctly locate each AU, MONET qualitatively retrieve the correct area for each AU due to parameter sharing and joint task distribution modelling.

Visualisation of MONET attribution maps

In order to investigate the impact of MONET sharing pattern, we compute the attribution maps as the gradient of the prediction for each AU w.r.t to the input image or intermediate features in the convolutional backbone. From a formal point of view, for the t-th action unit and intermediate feature $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$:

$$\mathbf{A} = \frac{1}{C} \sum_{i=1}^{C} (\nabla_{\mathbf{x}} p^t)_c \in \mathbb{R}^{H \times W}. \tag{2.41}$$

Figure 2.11 display AU-wise attribution maps for the three multi-task baselines along with MONET. VMNC and MRNN predicts action units using an architecture that is fully shared across tasks. Therefore, there is no space for task specialization. This result in similar heatmaps across tasks. For example, MRNN heatmaps from AU14 to AU24 are all very close. On the other hand, VMNS which uses a specific regressor by task manage to specialize its attribution for each tasks (heatmaps by tasks are different). However it misses important localizations such as eyebrows zone for AU1, AU2 while MRNN catches them. In fact, we believe that using the predictions of previous tasks along with the input image helps to guide MRNN attribution. MONET gets the best of the two worlds: On one side, its task-wise module enable task-wise attribution specialization. On the other side previous tasks in the learned order helps to guide its attribution.

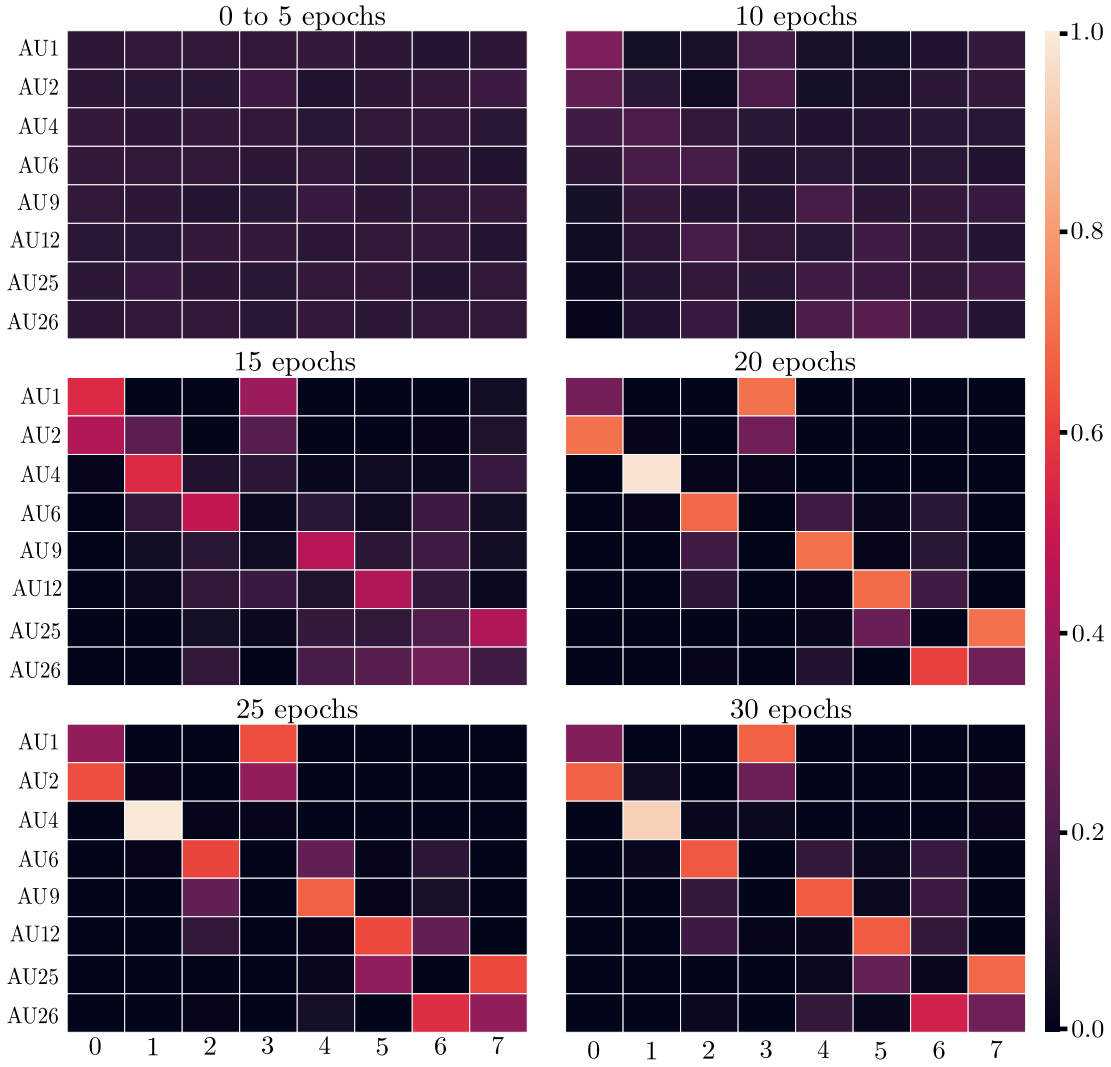


Figure 2.12: Evolution of MONET soft order during 30 epochs of training on DISFA. For action unit j, the coefficient associated with timestep i can be interpreted as the probability that action unit j is processed at timestep i.

Order visualisation in Action Unit Detection

Figure 2.12 shows MONET soft order evolution when training on DISFA. During the warm up phase (first 5 epochs), the soft order is the average of 512 randomly sampled permutation matrices and is therefore close to uniform (all probabilities are close to $\frac{1}{T}$). At the end of the warm up, MONET starts learning the order selection coefficients. In particular, we observe that MONET learns to start by predicting AU1, 2, 4, 6, 9 which correspond to the upper part of the face (eyes and brows related action units) and then processes 12, 25 and 26 which are located in the lower part of the face (mostly the mouth). In fact we believe that tasks that use the same part of the image and more generally the same piece of information benefit from being processed close to each other.

This coarse intuition seems to be fairly verified at a more detailed scale as we also observe smaller blocks of consecutive tasks. For instance, AU25-26 basically focuses on mouth and jaw, and AU4-6 both have their regions of interest located around the eyes. As far as brow movements are concerned, it is rather suprising that AU1 (inner brow raiser) and AU2 (outer brow raiser) are not processed in neighbouring timesteps. A possible explanation is that AU4 is easy to predict (mainly because it is less local as it often comes with nose and corner of the eye wrinkles) and more informative about AU1. Therefore AU1 benefits more from being processed near to AU4 than near to AU2. Also it is important to note that the set of orders on which MONET trains is finite. Consequently, if none of the orders selected has AU1 and AU2 next to each other, the soft order MONET learns can not process those two next to each other.

Comparison with state-of-the-art methods

BP4D : Table 2.4 draws a comparison between MONET and other state of the art deep approaches on the detection of 12 action units on BP4D database. Thanks to its order selection mechanism, MONET outperforms all the methods that explicitly models AU label dependencies such as DSIN or HMP-PS. More interestingly, it outperforms methods that use external information such as landmarks (EAC-NET, JAANET, PT-MT-ATsup-CC-E) or textual description of Action Units (SEV-NET). Finally, MONET performs better than PT-MT-ATsup-CC-E which leverages transformers in its architecture.
DISFA : Table 2.5 compares the performance of MONET with other state of the art approaches on DISFA. Similar to BP4D, MONET displays better performance than classical multi-task methods and all other existing approaches. Therefore, MONET consistently outperform state of the art performance for facial action unit detection.

F1 Score-AU	1	2	4	6	7	10	12	14	15	17	23	24	Avg.
DRML [101]	36.4	41.8	43.0	55.0	67.0	66.3	65.8	54.1	33.2	48.0	31.7	30.0	48.3
EAC-NET [29]	39.0	35.2	48.6	76.1	72.9	81.9	86.2	58.8	37.5	59.1	35.9	35.8	55.9
DSIN [10]	51.7	40.4	56.6	76.1	73.5	79.9	85.4	62.7	37.3	62.9	38.8	41.6	58.9
JAANet [59]	47.2	44.0	54.9	77.5	74.6	84.0	86.9	61.9	43.6	60.3	42.7	41.9	60.0
LP-Net [49]	43.4	38.0	54.2	77.1	76.7	83.8	87.2	63.3	45.3	60.5	48.1	54.2	61.0
CMS [57]	49.1	44.1	50.3	79.2	74.7	80.9	88.3	63.9	44.4	60.3	41.4	51.2	60.6
ARL [61]	45.8	39.8	55.1	75.7	77.2	82.3	86.6	58.8	47.6	62.1	47.7	55.4	61.1
SRERL [28]	46.9	45.3	55.6	77.1	78.4	83.5	87.6	63.9	52.2	63.9	47.1	53.3	62.1
JÂANET [60]	53.8	47.8	58.2	78.5	75.8	82.7	88.2	63.7	43.3	61.8	45.6	49.9	62.4
HMP-PS [64]	53.1	46.1	56.0	76.5	76.9	82.1	86.4	64.8	51.5	63.0	49.9	54.5	63.4
SEV-Net [87]	58.2	50.4	58.3	81.9	73.9	87.8	87.5	61.6	52.6	62.2	44.6	47.6	63.9
FAUwT [23]	51.7	49.3	61.0	77.8	79.5	82.9	86.3	67.6	51.9	63.0	43.7	56.3	64.2
VMNS	51.7	46.6	57.8	77.7	74.2	81.1	88.3	59.3	45.7	60.8	45.0	49.5	61.5
VMNC	48.7	45.2	56.8	77.9	77.8	83.2	87.9	62.9	51.1	59.1	47.4	52.7	62.6
MRNN	47.1	44.7	59.1	77.5	78.3	84.1	85.3	63.9	41.6	62.8	43.3	51.5	61.6
MONET (ours)	54.5	45.0	61.5	75.9	78.0	84.5	87.6	65.1	54.8	60.5	53.0	53.2	**64.5**

Table 2.4: Comparaison of MONET with state-of-the-art deep learning based AU detection methods on BP4D

Closer look at MONET's prediction

Figure 2.13 depicts images from BP4D along with their associated ground truth and MONET predictions. On the one hand, the left part (a, b, c) shows images where MONET predictions are fully accurate. It demonstrates that MONET is able to correctly classify both lowly expressive face (eg: (c, 3) and (c, 4)) and highly expressive ones (eg: (b, 2), (c, 2)).

On the other hand, the right part (d, e, f) focuses on sources of error i.e on edge cases in which MONET predictions are inaccurate. First, annotations seems to be noisy. For example, (e, 4) is annotated with AU1 (the inner brow raiser) whereas the subject is clearly frowning. Similarly, image (d, 3) is annotated with AU17 (chin raiser) which can't occur while the mouth is open.

Second, MONET tends to include the subjects neutral face bias in its predictions. For example subject (f)'s face looks, to the human eye, biased towards AU2 (the outer brow raiser), meaning that his eyebrows are naturally raised when his face is neutral. MONET includes this bias in its prediction and therefore predicts AU2 whereas it shouldn't because face bias are taken into account in FACS annotations.

Table 2.5: Comparaison of MONET with state-of-the-art deep learning based AU detection methods on DISFA

F1 Score-AU	1	2	4	6	9	12	25	26	**Avg.**
DRML [101]	17.3	17.7	37.4	29.0	10.7	37.7	38.5	20.1	26.7
EAC-NET [29]	41.5	26.4	66.4	50.7	8.5	89.3	88.9	15.6	48.5
DSIN [10]	42.4	39.0	68.4	28.6	46.8	70.8	90.4	42.2	53.6
SRERL [28]	45.7	47.8	59.6	47.1	45.6	73.5	84.3	43.6	55.9
JAANet [59]	43.7	46.2	56.0	41.4	44.7	69.6	88.3	58.4	56.0
LP-Net [49]	29.9	24.7	72.7	46.8	49.6	72.9	93.8	65.0	56.9
CMS [57]	40.2	44.3	53.2	57.1	50.3	73.5	81.1	59.7	57.4
ARL [61]	43.9	42.1	63.6	41.8	40.0	76.2	95.2	66.8	58.7
SEV-Net [87]	55.3	53.1	61.5	53.6	38.2	71.6	95.7	41.5	58.8
HMP-PS [64]	38.0	45.9	65.2	50.9	50.8	76.0	93.3	67.6	61.0
FAUwT [23]	46.1	48.6	72.8	56.7	50.0	72.1	90.8	55.4	61.5
JÂANET [60]	62.4	60.7	67.1	41.1	45.1	73.5	90.9	67.4	63.5
VMNS	53.4	51.3	64.8	45.5	36.0	70.1	89.8	62.4	59.2
VMNC	56.8	59.0	64.4	51.4	43.7	75.1	92.5	62.8	63.2
MRNN	47.4	49.7	61.8	46.7	38.8	71.0	91.9	60.9	58.5
MONET (ours)	55.8	60.4	68.1	49.8	48.0	73.7	92.3	63.1	**63.9**

Actually this particular source of error is fairly expected. Indeed, for a given expressive face image, MONET has no explicit information on the associated neutral face nor a fortiori on its bias. Consequently it cannot discriminate between an unbiased neutral face which display a particular AU (AU2 in (f) for example), and a neutral face that is biaised toward this particular AU.

Finally, MONET is not fully accurate on faces that display very specific emotions. Typically (d, 3) seems to be yawning, (e, 4) displays fear/disgust and (f, 4) looks extremely puzzled. An explanation for MONET inaccurate predictions could be that BP4D do not provide enough examples for those specific facial expressions, that may involve low-probability AU combinations. In other words, the existing AU-annotated databases are not large enough to train a fully efficient classifier.

Conclusion

In this section, we explored the MONET behaviour in Action Unit Detection. In particular, we first confirmed that MONET was a successful hybrid between VMN and MRNN. Furthermore, we observed that MONET tend to process task that

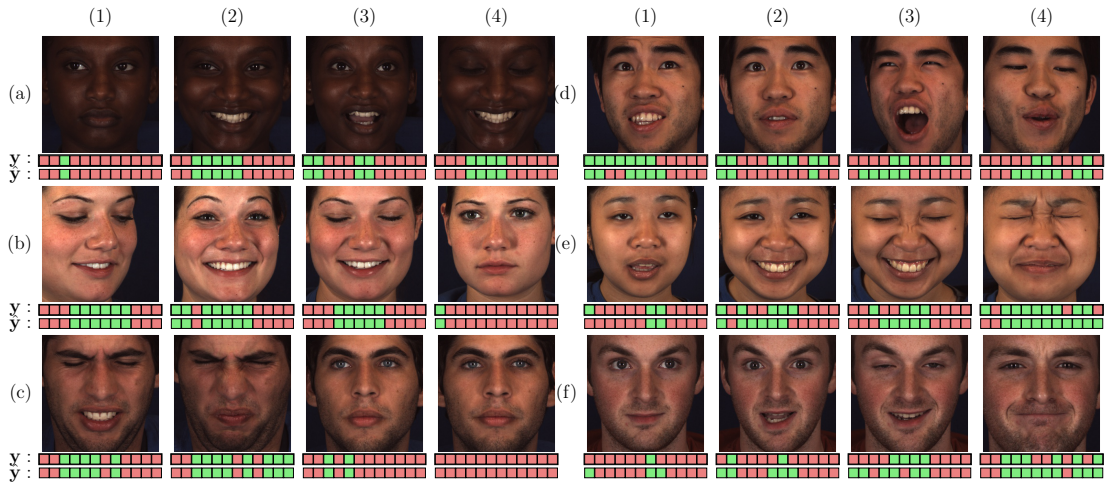


Figure 2.13: Samples of images from BP4D database with the AU groundtruth ($\mathbf{y}$) and MONET predictions ($\hat{\mathbf{y}}$). Each coordinate (represented by a square) represent a single AU (the order is AU1, AU2, AU4, AU6, AU7, AU10, AU12, AU14, AU15 AU17, AU23, AU25). A green square means that the AU is present in the image while a red square denotes its absence.

needs the same piece of information close to one another. Finally, we demonstrated that MONET outperforms state-of-the-art performance on both DISFA and BP4D. Additionally we identified neutral face bias as an important source of error. We consequently believe that finding ways to provide neutral face information to the network could help it remove that bias and in turn improve performance.

Further explorations

In this section we discuss MONET modelling assumption and the attempts we made at relaxing it. In fact, MONET core assumption is that order matters and that the best orders of prediction do not depend on the networks input. However, this assumption might be too restrictive. Indeed section 2.5.2 and 2.5.3, we provided empirical evidence that the order chosen by MONET mostly depended on (a) the difficulty of the tasks at end (the easiest task at the beginning and the hardest at the end) (b) information sharing between tasks (task that shares information should be treated close to one another). Yet, intuitively enough, the order induced by those two factors may not always be the same across all inputs. For example, if on an face image, the subject mouth is hidden then the mouth-associated Action Units will be harder to predict than the forehead associated

ones and the inverse is true if now the forehead is hidden. As a consequence, MONET could benefit from adaptative ordering. A simple way to achieve that is by making the mixture π depend from $\mathbf{x}$ so that :

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \sum_{m=1}^{M} \pi_m(\mathbf{x}) p_\theta^{\sigma_m}(y^{\sigma_m(t)} \mid \mathbf{y}^{\sigma_m(<t)}, \mathbf{x}), \tag{2.42}$$

where typically $\pi(\mathbf{x})$ is computed by applying dense layers and a softmax activation over the encoder output. As this modelling introduces a dependency in $\mathbf{x}$ we simply call this method XMONET.

Similar to what we did for MONET, we designed a toy dataset for testing if this method was able to find the correct orders. For that purpose, we drew inspiration from our original dataset on which the correct order is known, we splitted horizontally in 2 zones of equal size (above and below $\mathbf{y} = 0$) and we permutated the tasks with order σ_1 above and σ_2 below so that the best order is now σ_1 when $y \geq 0$ and σ_2 when $y \leq 0$. Figure 2.14 display the resulting toy.

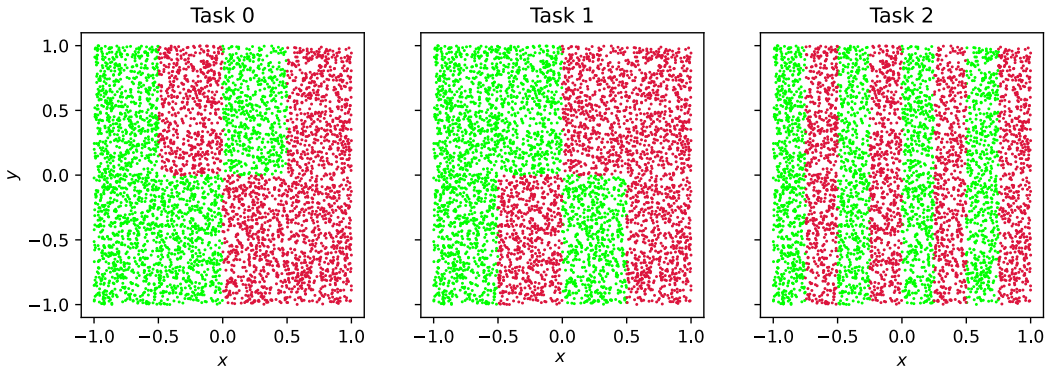


Figure 2.14: Distribution of 5000 examples sampled from our extended toy dataset with $T = 3$. In the extended toy dataset the best order is different based on weither examples are above or below the $y = 0$ line. Typically the best order is $\sigma_1 = [1, 0, 2]$ in the lower part and $\sigma_2 = [0, 1, 2]$ in the lower part.

Figure 2.15 shows the order finding frequency of XMONET for different toy order couples σ_1, σ_2. First, we observe that non-zero coefficients are mostly on the diagonal which correspond to the original toy. However, it seems that XMONET is less efficient than MONET when it comes to retrieving the order of a toy with a single order (only 9 of the 24 coefficients are close to 1 while MONET retrieves the best order everytime (see section 2.1). This is in fact rather intuitive as the single

order toy is MONET architecture canonical use case while XMONET architecture is not biased toward learning a single order and therefore has to learn this bias from the data which explains lesser performance.

Yet it is noticeable that, even in a more adapted scenario with 2 orders to learn, XMONET display variable performance order findings performance. Again, a possible explanation is that XMONET lacks appropriate bias for such scenario. Indeed, nothing in XMONET architecture constrains it to search for 2 possible orders. Therefore it has to learn it from the data. In fact, we believe that XMONET failures may be caused by the weakness of the order supervision signal. Indeed just like for MONET, XMONET's order selection relies on a comparaison of the losses associated with the different orders. Such comparaison may be noisy and difficult when several orders perform similarly which in turn explain XMONET's difficulties at retrieving the exact correct orders.

To summarize, extending order selection to slightly more complicated scenario is a rather difficult task. Indeed, the order losses comparaison seem to be too weak to learn rather easy order classification problems (a single horizontal boundary and constant class up and down of it). Further explorations could include training XMONET with more data to observe weither or not more accurate order loss supervision could strengthen the order supervision signal and allow it to find good orders. For now, the only way to retrieve the exact orders in a dataset is to make correct assumptions about the underlying data generation process. This is of course extremly difficult to do in a realistic scenario.

2.6 Conclusion

2.6.1 Discussion

In this chapter, we introduced MONET, a multi-order network for joint task order and prediction modelling in deep multi-task learning. MONET leverages a differentiable order selection mechanism based on soft order modelling inside Birkhoff's polytope, as well as task-wise recurrent cell sharing for concurrent multi-order prediction learning. Furthermore, we proposed to refine MONET order selection strategy with warmup and order dropout that enhances order space exploration, allows MONET to find better orders and in turn improve performance.

Similar to MRNN, MONET predicts the set of tasks at hand sequentially. Yet, contrary to MRNN where the prediction order is arbitrary and set once and for all at the beginning of training, MONET learns the order in which tasks should be predicted. The toy dataset exemplifies a situation where this sequential modelling is optimal in a given order. In this controlled scenario, we proved that MONET

converges towards the correct order and significantly outperforms its random order selection counterparts along with other existing multi-task baselines.

In general learning problems, we do not know whether task sequential modelling is beneficial compared to parallel prediction, nor, *a fortiori*, if *order matters*. In other words, we have no *a priori* information on whether MONET modelisation is well-adapted or not to a given situation. Yet, we experimentally demonstrate that MONET display competitive performance on a wide variety of task relationship settings: when applied to facial attribute detection (CelebA), MONET performs at least as well as the best multi-task baseline on each 5 attributes subset. Based on this empirical evidence, we argue that MONET is an all-around better multi-task method and could therefore constitute a valid architectural choice for many multi-task learning problems.

Furthermore, we show that, thanks to both its order selection mechanism and its task-wise sharing pattern, MONET successfully exploit the dependencies between the different Action Units and consequently outperforms state-of-the-art performance in AU detection (BP4D and DISFA). To further explore the reasons for those good performance, we first investigated the impact of MONET sharing pattern on its attribution maps, i.e on the zones it uses to predict the different AU. We found out that MONET benefits from the advantages of both VMN and MRNN multi-task baselines. On one side, VMN-like task-wise cell sharing allow successful specialization of each module in the task it has too treat. On the other side, MRNN-like explicit dependency modelling enables better guidance of the network attribution toward the zones that are important to predict AU. Lastly, we highlighted several possible tracks for further improvement of deep learning based Action Unit Detection system. In particular, as several BP4D images seem to have noisy annotations, modelling annotation noise could be an interesting line of research to follow. Furthermore, we pinpointed that neutral face bias could be an important obstacle to proper AU detection. Indeed, without neutral face information, a network cannot distinguish between a neutral face that is biased toward an AU activation or the AU activation itself. Therefore, integrating neutral face information into the learning process could help disentangle this confusion and in turn improve performance.

2.6.2 Limits and future works

To conclude, MONET still suffers from certain limitations. The most important of those limitations is the scalability to a large number of tasks. Indeed, with T, the number of possible orders is $T!$ so that training with all possible orders quickly becomes impossible. In that case, we believe that trying to select relevant

order beforehands could improve order exploration and performance. To design a selection heuristic, a possibility is to inspire from the observation in 2.5.3 that tasks that needs similar information should be treated close to on another (typically AU1 should be close to AU2). For that purpose, we could first partition the set of T tasks in G groups of T_G tasks that share similar information and then sample from the $G!(T_G)^G$ orders that permute tasks order inside groups along with the blocks themselves. However, choosing (a) the criterion for task similarity that guide groups assignement (b) the size of groups along with their number requires further exploration.

In the same vein, it is important to notice that MONET's sharing pattern uses a recurrent cell per task. Therefore, there is a linear dependency between the number of tasks T and the number of parameters. As a consequence, the number of parameters quickly become the limiting factor as the number of tasks increases. Similar to the previous idea, a first intuition to reduce the number of used cells would be to switch from task ordering to groups of task ordering i.e to partition the set of tasks into groups and to learn the ordering of the blocks. More formally, our idea would be to partition the T tasks into n_G groups $\mathbf{G}_1, \ldots, \mathbf{G}_{n_G}$ of tasks. Then, by assuming conditional independence of the task inside the blocks, it is possible to use Bayes Chain Rule to decompose the joint distribution of labels as follows :

$$\begin{aligned} p_\theta(\mathbf{y} \mid \mathbf{x}) &= p_\theta(\mathbf{y}^{\mathbf{G}_1}, \ldots, \mathbf{y}^{\mathbf{G}_{n_G}} \mid \mathbf{x}) \\ &= \prod_{i=1}^{n_G} p_\theta(\mathbf{y}^{\mathbf{G}_i} \mid \mathbf{y}^{\mathbf{G}_{<i}}, \mathbf{x}) \\ &= \prod_{i=1}^{n_G} p_\theta(\mathbf{y}^{\mathbf{G}_i} \mid \mathbf{y}^{\mathbf{G}_{<i}}, \mathbf{x}) \\ &= \prod_{i=1}^{n_G} \prod_{t \in \mathbf{G}_i} p_\theta(\mathbf{y}^t \mid \mathbf{y}^{\mathbf{G}_{<i}}, \mathbf{x}) \end{aligned} \tag{2.43}$$

Or by mimicking MONET task order selection :

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \sum_{m=1}^{M} \pi_m \prod_{i=1}^{n_G} \prod_{t \in \mathbf{G}_{\sigma(i)}} p_\theta^{\sigma_m}(\mathbf{y}^t \mid \mathbf{y}^{\mathbf{G}_{\sigma(<i)}}, \mathbf{x}) \tag{2.44}$$

The incentive that underpines the use of such modelling is that using MONET to find a correct task block order (a) reduces the number of recurrent cells from T to n_G (b) May lead to better order selection if used with a correct block partitioning heuristic (as $n_G! < T!$). However, similar to our previous idea, finding adapted partitioning strategies is non trivial and requires further explorations.

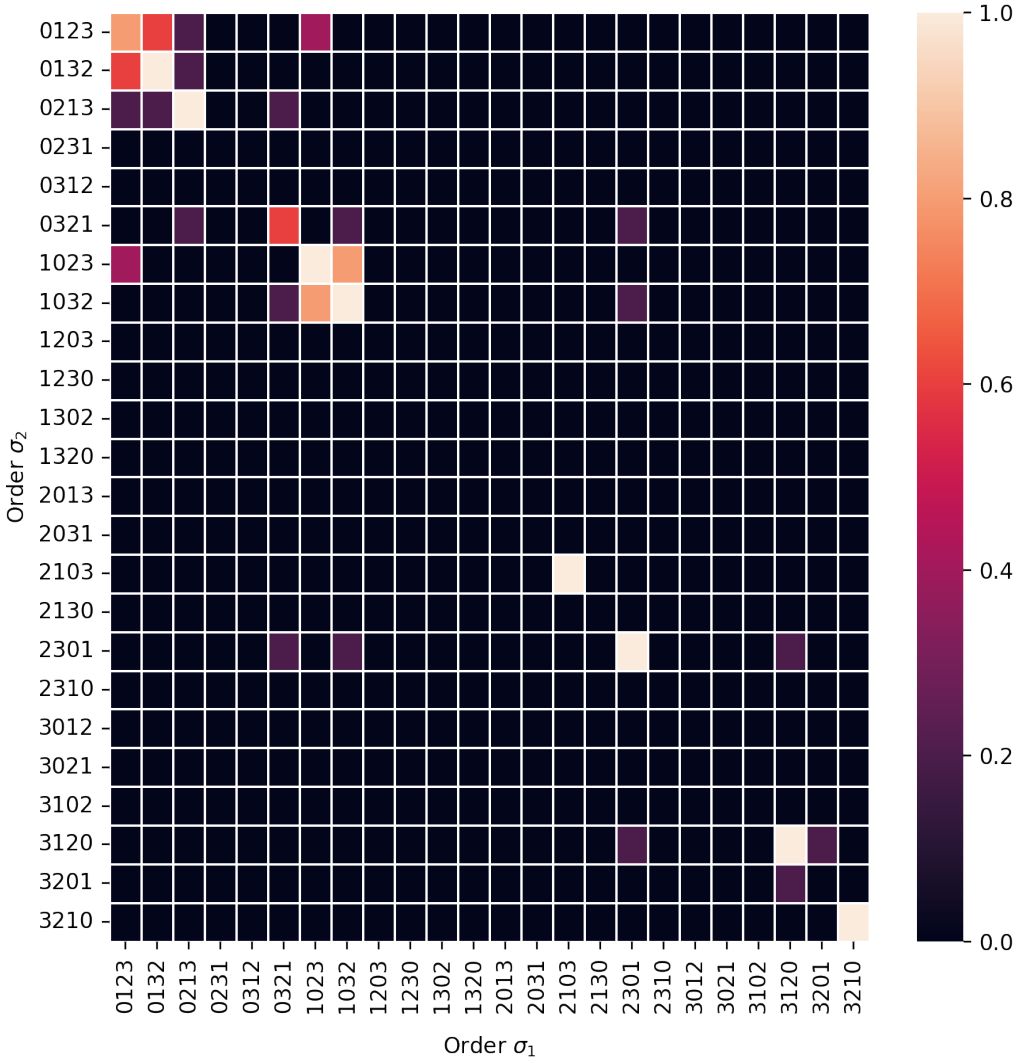


Figure 2.15: Order finding frequency of the XMONET architecture on the horizontally split toy with $T = 4$ and different values for σ_1 and σ_2. Each order is represented under the form $\sigma(1)\dots\sigma(T)$. We take advantage of the toy symetry and only computes the frequency for σ_1 vs σ_2 and report the same value for σ_2 vs σ_1 .

Chapter 3

Guiding transformers attention for better AU detection

3.1 Introduction

An Action Unit consists in the activation of a given facial muscle. Such activation modifies the texture of a fixed local portion of the skin. For different faces, the position of the affected zone and the way the AU activation modifies the skin may vary. For example, AU12 (smile) appearance depends on the size and position of the mouth. Yet, publicly available AU datasets only contains annotations for a very limited number of faces. Therefore, a deep neural network trained on such dataset may learn features that are too specific to the featured faces and consequently generalize poorly.

To tackle this aspect of the AU detection overfitting problem, the most widely adopted approach [101, 29, 59, 60, 3] is to help the learning phase with AU relevant zones computed from low-level facial primitives. Such guidance strategy alleviates the need to learn those zones from the data and considerably simplifies the learning problem. Therefore, it results in features that are less specific to the faces in the dataset which naturally enhances the generalisation capacity of the learned model.

Concurrently, vision transformers [13] have shown promising results in numerous computer vision applications [5, 66, 75]. However, prior works [13, 75] demonstrated that transformers are particularly sensitive to overfitting. Consequently, those architectures are ill-adapted to AU detection low data regimes.

For that purpose, recent works have tried to bridge that gap. In particular, hybrids between convolutional networks and transformers have shown relatively promising performance in affective computing applications [86] and more specifically in AU detection [23].

In this chapter, we therefore aim at designing new learning strategies that tackle transformers overfitting problem for AU detection. For that purpose, we first inspire from [23, 60] to build an hybrid architecture that is adapted to the multi-task AU detection problem. Then, we empirically observe that our architecture attention heads do not assign importance to the correct AU related zones. As a consequence, we inspire from the attention guiding literature [101, 29, 59, 60, 3] to design an approach that constrain transformers multi-head attention module using AU relevant spatial priors. More precisely, we attempt several guiding strategies that use either landmarks or face parsing priors. Overall, our main finding is that, regardless of the type of prior used, constraining each head of the multi-head attention mechanism to a different prior (eg. a face parsing class per head) improve the predictive performance of a transformer. To summarize, our contributions are as follows :

- On the one hand, we propose a landmarks-based prior guidance method in which each headlearns to pay attention to different AU relevant zones. In that extent, such method doesn't require landmarks information at test time. Experimentally, we show that this method provides a boost in performance on BP4D.
- On the other hand, we propose a prior guidance method that makes use of face parsing maps to bias each head toward looking at different semantical parts of the face. Contrary to the landmarks-based method, this method requires face parsing maps in both train and test phases. Again, we demonstrate that such method improves performance on both DISFA and BP4D AU detection problems.

This chapter is organized as follows : in section 3.2, we review state-of-the-art approaches techniques for both prior guided Action Unit detection and transformer-based methods. Then, section 3.3 describes how we adapt the original vision transformer [13] to the AU detection problem. In section 3.5, we present and evaluate the performance of different transformer-based attention guiding strategies. Finally, section 3.6 contains conclusive remarks and discuss future work.

3.2 Related Works

Guiding Action Unit detection with spatial priors

The idea of guiding AU detection with spatial priors have been extensively studied in the literature. Seminal work [101] proposed to split face images in square regions

and to use specific convolutional kernels in each of those regions. The incentive behind this strategy is that using different convolution kernels for different regions of the image allow each kernel to learn more local features and is consequently more adapted to AU detection. However, the performance of such method is conditioned upon face alignment i.e upon whether or not specific face parts are always located in the same rectangular region.

To circumvent this limitation, several approaches [29, 59, 60] used facial key-points to guide the extraction of AU-related features. The work in [49] leveraged landmarks information to normalize face shapes and get AU-discriminative features that are less person-specific. Authors in [29] proposed EAC-Net. EAC-Net is composed of two parts : an enhancing layer (E-Net) and a cropping layer (C-Net). The role of the E-Net is to enhance AU-relevant features. To do so, it simply multiplies convolutional features by attention maps that weight each pixel based on its proximity with AU-relevant centers. The resulting feature maps are both used to predict AU and fed to the C-net that crops small zones around AU-relevant centers and learn to predict each AU based only on those crops.

Following the same inspiration as the E-net, JAANET was proposed in [59, 61] for joint landmark localization and AU detection. More precisely, JAANET first predicts facial keypoints and deduces the position of AU-relevant centers. Then it proceeds similarly as in [29] to enhance AU-relevant information inside the convolutional feature that learn to predict AU. However, all those attention maps-based methods were criticized in [61] for their lack of flexibility. Indeed as AU are correlated, attention maps for a fixed AU should not only enhance features that are relevant for this AU but also the ones relevant to the AUs that are correlated with it.

In the same vein but for Facial Expression Recognition (FER), the work in [3] draw inspiration from the visual explanation literature [1] to guide a network prediction with landmarks. Those methods estimates each image parts influence *a posteriori* based on the gradient of the prediction with respect to the original image or intermediate convolutional features. Those estimations are referred to as *attribution maps*. In order to force a network to privilege pixels that are close to facial key-points, the approach in [3] designed a *Privileged Attribution Loss* (PAL) that encourages the mean of a portion of the attribution map channels to look like a landmarks-based prior. The main incentive of this attribution method is that the constraint it imposes affects the whole network and is therefore stronger i.e more difficult to ignore than the ones of attention maps-based methods. However, it is important to note that this stronger constraint comes with an additional backward pass and is therefore significantly more computationally expensive.

Transformer Architectures

Transformer architectures were originally introduced for machine translation [78] as a fully parallel alternative to recurrent networks. The keystone of those architectures is the multi-head attention module. Multi-head attention is powered by a mechanism that uses queries and key/value pairs. In short, for each query, it outputs a recombination of linearly projected values that is weighted by a learned similarity between the query at hand and the values associated keys. In the original paper [78] in which queries, keys and values are words embedding, it is used in two mains variations. First, self-attention (SA) that takes the same input for queries, keys and values. SA is used on both input and output sentences to produce input and output sentence representations. The purpose of this module is to group words that are semantically similar regardless of their distance in the sentence. Then, the cross-attention parts (CA) learns to predict each word of the translation. For that purpose, it uses previously predicted word representations as queries and input sentence representation as keys and values. The role of this module is to select the parts of the input sentence that are most relevant to the prediction of each translated words.

Compared to recurrent networks, transformer's attention mechanism is able to learn relationships between words regardless of their distance in the sentence. As a consequence, it is more adapted for the processing of long sequences. This explains their success in most natural language processing (NLP) applications [78, 12, 51].

More recently, transformers have been adapted from NLP to computer vision [13] by feeding image patches representation instead of word embeddings to the multi-head attention mechanism. In that context, the self-attention modules is then tasked with grouping patches that holds similar information regardless of their distance in the image. Then, cross-attention learn to pay attention to the patches that are most relevant to the task at hand by using a special learnable token (often refered to as the class token (CLS) [75]) as query and the patch based representation as keys and values.

Contrary to convolutions that capture local patterns, multi-head attention on patches allows to capture links between patch contents no matter how far they are in the image. This comparative advantage explains that vision transformers have recently displayed competitive performance in several computer vision problem such as object recognition [77, 75], object detection [5] and segmentation [66].

However, contrary to what happened in NLP, convolutions based networks remain competitive with the more recent self-attention based architecture [36], especially in low data regimes. In fact, authors in [14] argues that contrary to transformers, convolutions local nature and weight sharing offers useful hard inductive

architectural bias that improves data-efficiency. For example, convolutional features are equivariant by translation (*i.e* meaning that the features extracted from the translation of an image are roughly the same as the translation of the features extracted from the original image). As transformers lacks those inductive architectural bias, they are able to learn more flexible relationships which make them more efficient when sufficient training data is provided. However, in low data-regimes, such flexibility rather hurts transformer performance by making them particularly sensitive to overfitting.

Recently, massive efforts [75, 34, 14, 90, 44, 52, 35, 83, 89, 85] have been made to get the best of the two worlds. In particular, DeiT [75] proposed a transformer-adapted distillation strategy, in which a transformer student learns from the latent representation of a convolutional teacher. In the same spirit, several works attempted to introduce convolutions at different levels of the self-attention modules [14, 90, 52, 89, 83]. Specifically, CvT [83] and CeiT [89] applied convolutions before and after the SA module respectively. However, such approaches have been criticized in [14]. The main point is that introducing convolutions hard inductive bias may cripple transformer architecture with convolution limitations. To circumvent this problem, authors in [14] proposed ConViT. ConVit first emulates convolution using a well-chosen query/key comparaison. Then, it provides the transformer's SA module with a gating mechanism that allows them to softly learn weither attention-based convolution or classical attention is more beneficial for each layer.

Those efforts permitted to significantly reduce transformers data hunger [14]. However, on Imagenet-1k [11], which is a rather small compared with typical affective computing datasets, convolutional networks remain competitive with their transformer counterpart [76]. Therefore, there is still a lot to be done to adapt transformers to affective computing and typically AU detection lower data regimes.

In an attempt to bridge that gap, several works tried to apply transformers to affective computing tasks. In particular, the method in [23] inspires from formerly mentioned works on mixing convolutions with attention. More precisely, it uses a convolutional backbone to extract specific features for each AU and a self-attention module to refine those features. In the same vein but for FER, TransFER was introduced in [86]. Similar to [23], TransFER uses an architecture with both convolution and transformer blocks. Additionally, it takes a step further toward fighting transformer blocks overfitting by designing an adapted structured dropout strategy. More precisely, it proposes to randomly zero-out attention heads in both CA and SA module to ensure that each head learn to pay attention to face parts that are useful to the task at hand.

Our work lies in the line of research of [23] and [86] and is motivated by the increasing popularity of attention guiding approaches in the affective computing

field [3, 59, 60]. In fact we share the same objective as [86] which is to help attention heads focus on relevant face parts. For that purpose, we inspire from both transformer [14, 6] and non transformer [3, 59, 61] litterature to investigate prior-based guiding strategies for transformers.

3.3 Building Vision Transformers

In this section we inspire from vision transformer original paper [13] and [75] to build a vanilla vision transformer architectures.

3.3.1 Architecture Building blocks

Transformers General Attention modules (GA) [78] are constituted of Multi Head Attention Blocks (MHA) and Multi Layer Perceptron Blocks (MLP).

Multi Head Attention Block

In a Multi Head Attention block with feature size d and N_H heads, input queries $\mathbf{Q} \in \mathbb{R}^{N_Q \times d}$, and key/value pairs ($\mathbf{K} \in \mathbb{R}^{N_K \times d}$, $\mathbf{V} \in \mathbb{R}^{N_K \times d}$) are projected and split in N_H different heads. Hence, for head h:

$$\tilde{\mathbf{Q}}^{(h)} = \mathbf{Q}\mathbf{W}_Q^{(h)} \in \mathbb{R}^{N_Q \times d_H}, \tag{3.1}$$

$$\tilde{\mathbf{K}}^{(h)} = \mathbf{K}\mathbf{W}_K^{(h)} \in \mathbb{R}^{N_K \times d_H}, \tag{3.2}$$

$$\tilde{\mathbf{V}}^{(h)} = \mathbf{V}\mathbf{W}_V^{(h)} \in \mathbb{R}^{N_K \times d_H}, \tag{3.3}$$

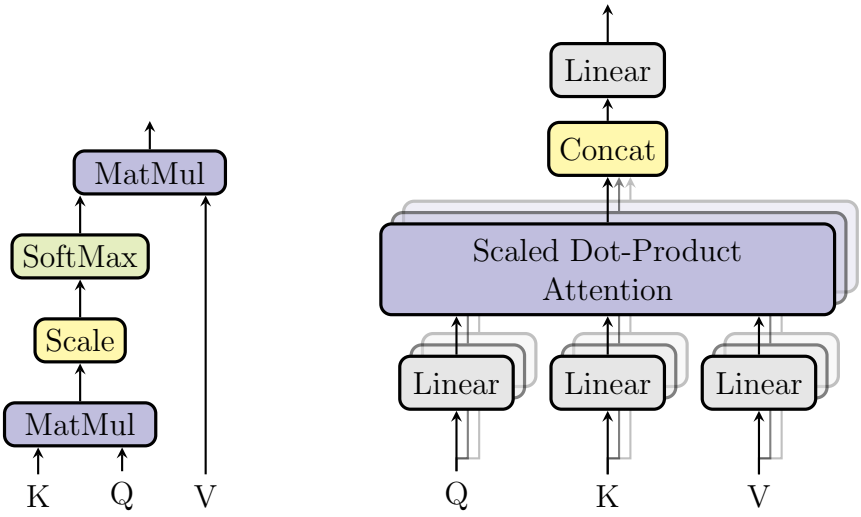
where $d_H = d/N_H$ and $\mathbf{W}_Q^{(h)}, \mathbf{W}_K^{(h)}, \mathbf{W}_V^{(h)}$ are the encoding matrices of head h for queries, keys and values respectively.

Then, each head computes an attention matrix $\mathbf{A}^{(h)}$ that measure similarities between queries and keys:

$$\mathbf{A}^{(h)} = \frac{\tilde{\mathbf{Q}}^{(h)}(\tilde{\mathbf{K}}^{(h)})^T}{\sqrt{d_H}} \in \mathbb{R}^{N_Q \times N_K}, \tag{3.4}$$

and, for each query, values are recombined in a convex fashion based on the similarity of their associated keys to the query:

$$\mathbf{O}^{(h)} = \text{softmax}(\mathbf{A}^{(h)})\tilde{\mathbf{V}}^{(h)} \in \mathbb{R}^{N_Q \times d_H}, \tag{3.5}$$



(a) Scaled Dot-Product Attention (b) Multi-Head Attention

Figure 3.1: Computational graph of the multi-head attention module from [78]

Finally, the different heads are concatenated and reprojected. The final output of the block is consequently:

$$\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}; \theta_{\text{MHA}}) = \text{concat}(\mathbf{O}^{(h)})\mathbf{W}_O \in \mathbb{R}^{N_Q \times d}, \tag{3.6}$$

where θ_{MHA} regroups all trainable linear projections i.e $\mathbf{W}_O$ along with $\mathbf{W}_Q^{(h)}, \mathbf{W}_K^{(h)}, \mathbf{W}_V^{(h)}$ for $h \in [\![1, N_H]\!]$. Figure 3.1 provides a detailed graphical representation of the multi-head attention module computational graph.

Multi Layer Perceptron Block

The MLP Block is a single hidden layer multi layer perceptron which projects all N_Q inputs $\mathbf{Q}$ (typically the output of the MHA block) into a higher dimensional space where a gelu non linearity is applied before shrinking the features back into their original dimension:

$$\text{MLP}(\mathbf{Q}; \theta_{\text{MLP}}) = \text{GELU}(\mathbf{Q}\mathbf{W}_u)\mathbf{W}_d \in \mathbb{R}^{N_Q \times d}, \tag{3.7}$$

where θ_{MLP} regroups linear projections $\mathbf{W}_d$ and $\mathbf{W}_u$.

General Attention module

Given input queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$, the general attention module computational graph, represented in figure 3.2, consists in a succession of MHA and

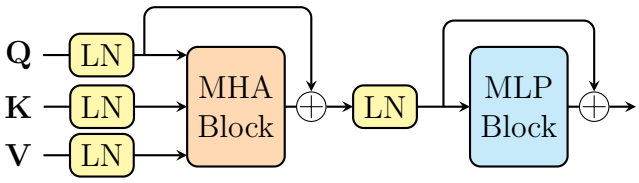


Figure 3.2: Computational graph of the General Attention module

MLP interspersed with layer normalizations LN and residual connections. More precisely, the inputs are first normalized:

$$\bar{\mathbf{Q}} = \mathrm{LN}(\mathbf{Q}), \bar{\mathbf{K}} = \mathrm{LN}(\mathbf{K}), \bar{\mathbf{V}} = \mathrm{LN}(\mathbf{V}), \tag{3.8}$$

then, the resulting features are updated by adding the output of the MHA block in a residual fashion and subsequently renormalized:

$$\mathbf{Q}_{\mathrm{MHA}} = \bar{\mathbf{Q}} + \mathrm{MHA}(\bar{\mathbf{Q}}, \bar{\mathbf{K}}, \bar{\mathbf{V}}; \theta_{\mathrm{MHA}}), \tag{3.9}$$

$$\bar{\mathbf{Q}}_{\mathrm{MHA}} = \mathrm{LN}(\mathbf{Q}_{\mathrm{MHA}}). \tag{3.10}$$

and finally re-updated in the same residual way by adding the output of the MLP block:

$$\mathrm{GA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}; \theta_{\mathrm{GA}}) = \bar{\mathbf{Q}}_{\mathrm{MHA}} + \mathrm{MLP}(\bar{\mathbf{Q}}_{\mathrm{MHA}}; \theta_{\mathrm{MLP}}), \tag{3.11}$$

where θ_{GA} regroups all learnable parameters. In particular, we separate two specific cases of general attention modules that are self attention modules (SA) in which query, keys and values are the same:

$$\mathrm{SA}(\mathbf{Q}; \theta_{\mathrm{SA}}) = \mathrm{GA}(\mathbf{Q}, \mathbf{Q}, \mathbf{Q}; \theta_{\mathrm{SA}}), \tag{3.12}$$

and cross attention modules (CA) in which only keys and values are the same:

$$\mathrm{CA}(\mathbf{Q}, \mathbf{K}; \theta_{\mathrm{CA}}) = \mathrm{GA}(\mathbf{Q}, \mathbf{K}, \mathbf{K}; \theta_{\mathrm{CA}}). \tag{3.13}$$

3.3.2 Vision Transformer Architecture

Vision transformers [13, 76, 75, 77] use representation based on regularly distributed patches of size $P \times P$. Therefore the first step of a vision transformer architecture is a patch-embedding step that convert image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$ into a patch-based representation by passing it into convolutional layer C with kernel size and stride equal to P and d filters:

$$\mathbf{P} = C(\mathbf{x}; \theta_C) \in \mathbb{R}^{h \times w \times d}, \tag{3.14}$$

where $h = H/P, w = W/P$. Then, a positional encoding $\mathbf{PE} \in \mathbb{R}^{h \times w \times d}$ is added to the patch-based representation. The result is fed to N_{SA} self-attention layers that learn to group patches with similar semantical content:

$$\mathbf{P}_0 = \mathbf{P} + \mathbf{PE}, \tag{3.15}$$

$$\mathbf{P}_{n+1} = \text{SA}(\mathbf{P}_n; \theta_{\text{SA}}^{(n)}), n \in [\![0, N_{\text{SA}} - 1]\!]. \tag{3.16}$$

The role of positional encoding is to provide each patch with an indicator of its position in the image. Basically, it allows self-attention to combine patches from different parts of the image without loosing spatial information. In practice, it is often implemented [78, 13] by adding the following contribution to patch (i, j) :

$$\mathbf{PE}_{i,j} = \text{concat}(\mathbf{PE}_i, \mathbf{PE}_j), \tag{3.17}$$

where:

$$\mathbf{PE}_k = \begin{bmatrix} \cos(w_1 k) \\ \sin(w_1 k) \\ \vdots \\ \cos(w_{\frac{d}{4}} k) \\ \sin(w_{\frac{d}{4}} k) \end{bmatrix}, w_k = \frac{1}{10000^{4k/d}}. \tag{3.18}$$

Once provided with the self-attention-refined patch-based representation $\mathbf{P}_{N_{\text{SA}}}$, the vision transformer architecture introduces a special learnable class token $\text{CLS} \in \mathbb{R}^d$ that is used to query the patch-based representation using N_{CA} cross-attention layers:

$$\text{CLS}_0 = \text{CLS} \tag{3.19}$$

$$\text{CLS}_{n+1} = \text{CA}(\text{CLS}_n, \mathbf{P}_{N_{\text{SA}}}; \theta_{\text{CA}}^{(n)}), n \in [\![0, N_{\text{CA}} - 1]\!]. \tag{3.20}$$

Throughout the cross-attention layers, the special token progressively extracts and stores relevant information for the task at hand. Consequently, $\text{CLS}_{N_{\text{CA}}}$ is fed to a dense layer D with appropriate activation act and the output is used as prediction for the task at hand:

$$\mathbf{p} = \text{act} \circ D(\text{CLS}_{N_{\text{CA}}}; \theta_D). \tag{3.21}$$

Figure 3.3 summarizes the full vision transformer architecture. In the next section, we adapt this transformer architecture to the AU detection problem.

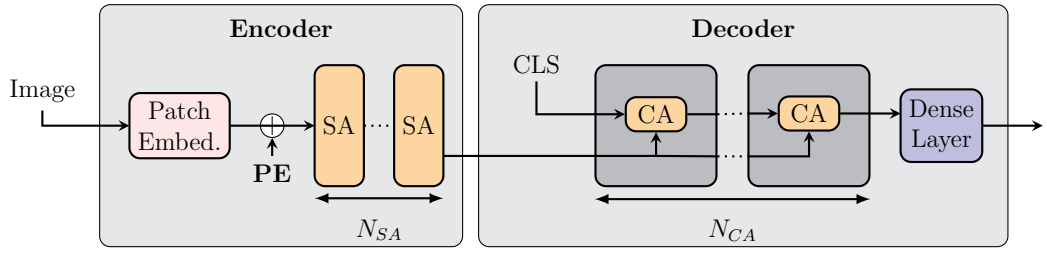


Figure 3.3: Overview of the vision transformer architecture

3.4 Adapting Vision Transformers to AU detection

3.4.1 Convolutional encoder to fight overfitting

Affective computing and more specifically Action Unit detection problems are known to suffer from overfitting due to the lack of annotated data. To circumvent this issue, prior work leverage convolutional encoder pretrained on larger scale face-based databases (typically VGGFACE2 [4]). However, to the best of our knowledge, no such pretrained transformer backbone is publicly available yet. For that purpose, we follow the strategy in [86, 23] to build an hybrid encoder with self-attention layers applied on top of a pretrained convolutional encoder.

From a formal point of view, our baseline architecture simply replace vision transformer patch-embedding by a convolutional encoding step. For that purpose, it uses a convolutional encoder f_{θ_e} with parameters θ_e in which the last spatial aggregation layer is replaced by a 1×1 2D convolution followed by layer normalization on the last axis. The role of this layernorm is to ensure that the resulting feature have the same scale as the positional encoding in order to the position information from being ignored. As a result for a given image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$, equation 3.14 is replaced by:

$$\mathbf{P} = \mathrm{LN}[f_{\theta_e}(\mathbf{x})] \in \mathbb{R}^{h \times w \times d}, \tag{3.22}$$

where $h \times w$ is the resolution of the convolutional representation.

3.4.2 Multi-task decoder

To adapt the vision transformer architecture to a multi-task binary classification framework, we propose two different multi-task decoders: The Common Multi-Task Transformer (MTTC) that learns to attend to zones that are relevant to all

tasks at once while the Separate Multi-Task Transformer (MTTS) learns to attend to zones that are relevant to each specific task.

Common Multi-Task Transformer (MTTC)

In order to learn an attention that is common to all the tasks, MTTC proceeds in the same way as the vanilla vision transformer architecture. More precisely, similar to equation 3.20, it passes a single learnable token $\mathbf{T} \in \mathbb{R}^d$, that we refer to as task token, through N_{CA} cross-attention layers with $\mathbf{P}_{N_{\text{SA}}}$ as keys and queries:

$$\mathbf{T}_0 = \mathbf{T} \tag{3.23}$$

$$\mathbf{T}_{n+1} = \text{CA}(\mathbf{T}_n, \mathbf{P}_{N_{\text{SA}}}; \theta^{(n)}_{\text{CA}}) \in \mathbb{R}^d, n \in [0, N_{\text{CA}} - 1]. \tag{3.24}$$

Then, the resulting embedding is passed through a dense layer D parametrized by θ_D with T units, and a sigmoidal activation σ is applied to output the final prediction for each AU:

$$\mathbf{p} = \sigma \circ D(\mathbf{T}_{N_{\text{CA}}}; \theta_D) \in \mathbb{R}^T. \tag{3.25}$$

Separate Multi-Task Transformer (MTTS)

On the otherside, MTTS learns task-wise attention, i.e tasks may look at different locations based on the most informative patches of the image. For that purpose, it passes T task tokens $\mathbf{T} \in \mathbb{R}^{T \times d}$, that we refer to as task tokens, through N_{CA} cross-attention layers with $\mathbf{P}_{N_{\text{SA}}}$ as keys and queries:

$$\mathbf{T}_0 = \mathbf{T} \tag{3.26}$$

$$\mathbf{T}_{n+1} = \text{CA}(\mathbf{T}_n, \mathbf{P}_{N_{\text{SA}}}; \theta^{(n)}_{\text{CA}}) \in \mathbb{R}^{T \times d}, n \in [0, N_{\text{CA}} - 1]. \tag{3.27}$$

Finally, the t-th tokens $\mathbf{T}^t_{N_{\text{CA}}}$ is fed into its own dense layer D parametrized by θ^t_D with a single unit, and a sigmoidal activation σ is applied to output prediction p^t for the t^{th} AU:

$$p^t = \sigma \circ D(\mathbf{T}_{N_{\text{CA}}}; \theta^t_D), \tag{3.28}$$

For both decoders all parameters θ are optimized using classic maximum-likelihood based losses. Figure 3.4 displays the full computational graph of our baseline transformers.

In the next section, we experimentally explore baseline transformer architectural choices on BP4D.

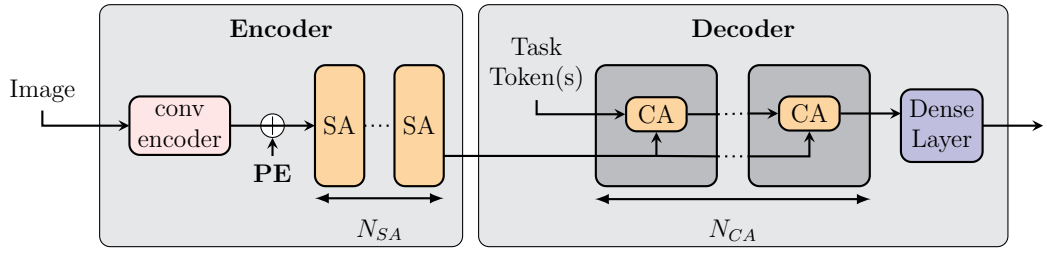


Figure 3.4: Overview of the AU-adapted hybrid architecture. When a single task token is provided this architecture correspond to the MTTC. When several task tokens are provided it corresponds to MTTS

3.4.3 Preliminary Experiments

Implementation details

In all the following experiments, we use a ViT-b16 pretrained on Imagenet-22k object recognition task for the encoder transformer. For the convolutional encoder we use a Resnet50 pretrained on either Imagenet-1k or VGGFACE2 face identification task with $N_{\text{SA}} = 1$ self-attention layers on top of it. We push our study further by cutting the Resnet50 at different points (directly after the different main convolutional blocks) to compare patch-based convolutional representation at different resolution (28×28, 14×14, 7×7). As far as self-attention is concerned we provide result with ($N_{\text{SA}} = 1$) and without ($N_{\text{SA}} = 0$) it.

For the comparaison between ViT-b16 and Resnet50 we set the model size to $d = 768, n_h = 12$ and use $N_{\text{CA}} = 2$ cross-attention layers [77].

As far as optimization is concerned, we use Adam[37] with exponential learning rate decay $\beta = 0.75$ for 2 epochs. In the convolutional part we use initial learning rate $\lambda_c = 5e{-}5$. In the transformer part we scale the initial learning rate w.r.t the number of queries q (P in self attention and T in cross-attention), the model size d and the batchsize $B = 32$. Thus the learning rate becomes $\lambda_t = \lambda_t^{(0)} \frac{Bq}{S\sqrt{d}}$ where $S = 2.5 \times 1e5$ is a scale derived from the experiments in [78] and $\lambda_t^{(0)} = 1e - 2$.

Architectural choices

Table 3.1 shows the AU detection results for different versions of our hybrid baseline architecture. First we notice that the ViT-b16 encoder is outperformed by all the convolutional encoders and *a fortiori* by versions that use self-attention on top of the convolutional network. This validates the choice of a convolutional encoder with a self-attention layer on top of it as backbone. Interestingly enough,

the pretraining doesn't seem to be responsible for the latter performance gap as Resnet50 is still better than ViT-b16 even when pretrained on Imagenet. In fact, those results resonate with litterature findings [75] that point out that the lack of inductive bias (eg: no translation equivariance) in transformer architecture make them particularly sensitive to overfitting and hinders their performance in low data regimes.

Table 3.1: Comparaison of the multi-task architecture with different backbones and different decoders

Encoder	patch resolution	pretraining	N_{SA}	Decoder	Mean F1Score
ViT-b16	14×14	ImageNet	0	MTTC	59.1 ± 0.9
Resnet50	28×28	ImageNet	0	MTTC	59.0 ± 0.7
Resnet50	14×14	ImageNet	0	MTTC	61.2 ± 0.9
Resnet50	7×7	ImageNet	0	MTTC	61.7 ± 0.5
Resnet50	28×28	ImageNet	1	MTTC	59.9 ± 0.8
Resnet50	14×14	ImageNet	1	MTTC	62.3 ± 0.6
Resnet50	7×7	ImageNet	1	MTTC	61.5 ± 0.5
Resnet50	28×28	VGGFACE	1	MTTC	60.8 ± 0.9
Resnet50	14×14	VGGFACE	1	MTTC	62.3 ± 0.5
Resnet50	7×7	VGGFACE	1	MTTC	62.0 ± 0.5
Resnet50	28×28	VGGFACE	1	MTTS	60.9 ± 0.3
Resnet50	14×14	VGGFACE	1	MTTS	$\mathbf{62.6 \pm 0.8}$
Resnet50	7×7	VGGFACE	1	MTTS	61.4 ± 0.3

To further discuss pretraining concerns, we observe that the VGGFACE2 pretrained versions of the convolutional encoders is on par or slightly above their Imagenet counterparts for all tested resolutions. However, the observed gap is not as significant as expected. Indeed, VGGFACE2 is composed of 3.3M faces with more than 9000 identities. Therefore using VGGFACE2 pretraining should initialize the training with more face-adapted features than Imagenet pretraining. A possible explanation for those disappointing performances could come from the nature of VGGFACE2 task which is face identification. As a consequence the features extracted from a network pretrained on VGGFACE2 are mostly identity-related (typically shape of the face, color of the eyes) and may be invariant to facial expression (because facial expression may vary for the same identity) making them ill-adapted for the AU detection task.

Furthermore, Table 3.1 provides a comparaison between the two multi-task token sharing strategies. We notice that MTTS consistently outperforms MTTC

and reaches the performance of VMNC that are reported in Table 2.4. A possible explanation for the performance gap between MTTC and MTTS is that, by using cross-attention between image patches and each task-relative tokens, MTTS decoder is able to efficiently locate and use relevant information for each task.

For all the upcoming examples we use a Resnet50 with patch resolution convolutional backbone on top of which we build a MTTS multi-task decoder.

Attention Visualisation

To further investigate, the attention mechanism of MTTS, Figure 3.5 displays the attention matrices cross-attention layer for different heads and different AU. More precisely, for given head h and task t, we first resize cross-attention matrix in 3.4 to the image size. This process outputs a matrix $\tilde{\mathbf{A}}^{(h)} \in \mathbb{R}^{T\times H\times W}$. Then we overlay the t-th coordinate of $\tilde{\mathbf{A}}^{(h)}$ on the input image. The resulting image highlights the locations that are attended by head h for task t.

Even though MTTS performs relatively well compared to convolutional baselines results reported in 2.4, we observe that the attention matrices are not in adequacy with what we expect. The most striking example is the attention heads that concern the mouth related AU (12 to 24) as none of the displayed attention heads is really focused on the mouth.

We believe that the lack of training data is responsible for such incoherent attention matrices. Our belief is endorsed by previous works on transformers [75] which describe transformers as data-hungry architecture that requires millions of training examples to achieve competitive performance. Consequently, in an attempt to reduce transformers data-hunger we design two different methods to help transformer training by guiding attention matrices using either landmarks or face parsing.

3.5 Guiding Transformers Attention

3.5.1 Landmarks-based Cross-Attention guidance for Transformers

In this section, each face image $\mathbf{x} \in \mathbb{R}^{H\times W\times 3}$ is provided with L 2D face keypoints $\mathbf{l} \in \mathbb{R}^{L\times 2}$. Furthermore, for each of the T tasks, we suppose that we have access to a landmark-based expert spatial prior that locates C centers of interest $\mathbf{c} = f_{\text{prior}}(\mathbf{l}) \in \mathbb{R}^{T\times C\times 2}$. Table 3.2 provides explicit description on how f_{prior} maps landmarks to AU-wise centers of interests. Figure 3.6 display example of the

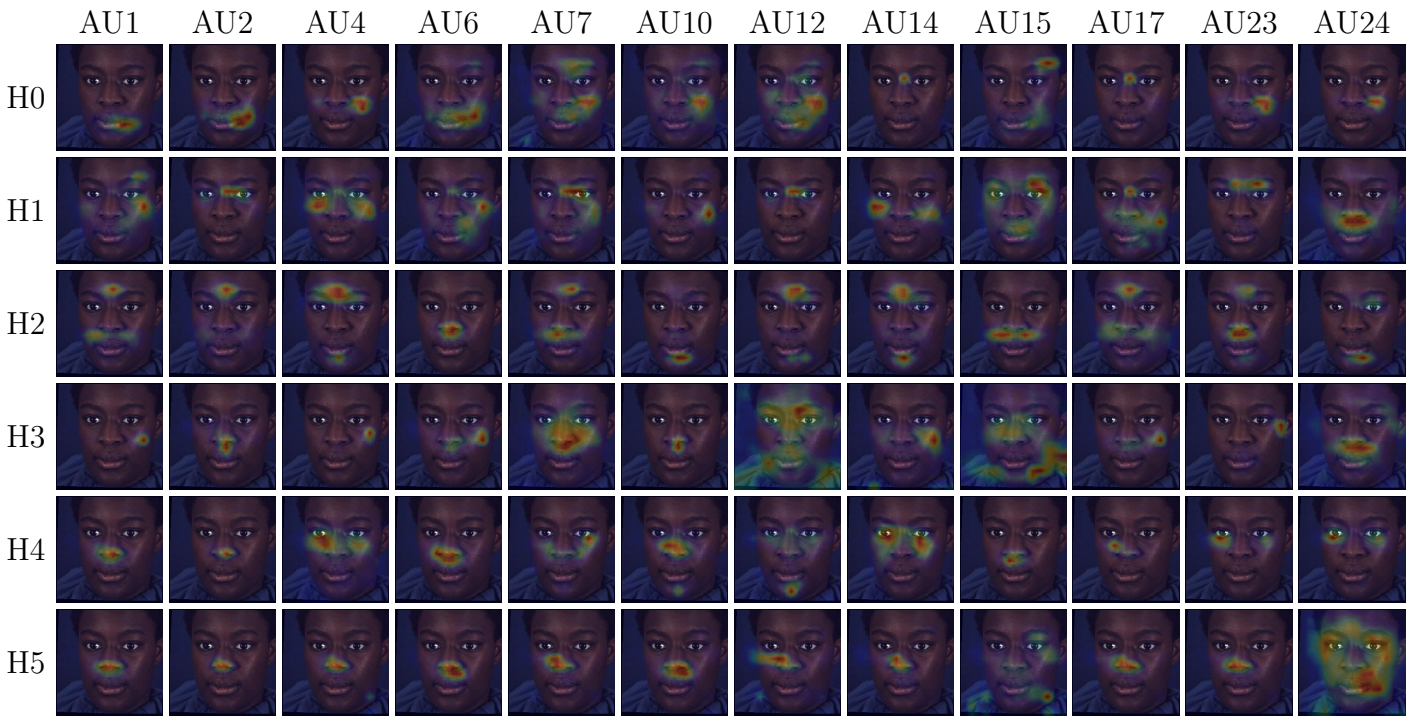


Figure 3.5: Visualisation of the 6 first CA layer attention heads overlayed on the input image.

centers of interests. For instance, the two centers of interest for AU12 (smile) lies in the two mouth corners.

Computing attention priors from landmarks

For a given attention head and a given task, the cross-attention matrices of a transformer can be interpreted as a probability distribution over a discrete $h \times w$ rectangular grid that partitions the input image. From an intuitive perspective those probabilities represent a ranking of the learned importance of each part of the image.

In order to constrain cross-attention matrices, we first need to convert our expert-based landmarks locations into something homogenous with those images. For that purpose the first necessary step is to rescale our center of interests to the same scale as the attention images. For a given center k that is expertly known as relevant to task t, we therefore perform a rescaling operation:

$$\tilde{\mathbf{c}}_k^t = \mathbf{c}_k^t \odot \begin{bmatrix} \frac{h}{H} & \frac{w}{W} \end{bmatrix}, \tag{3.29}$$

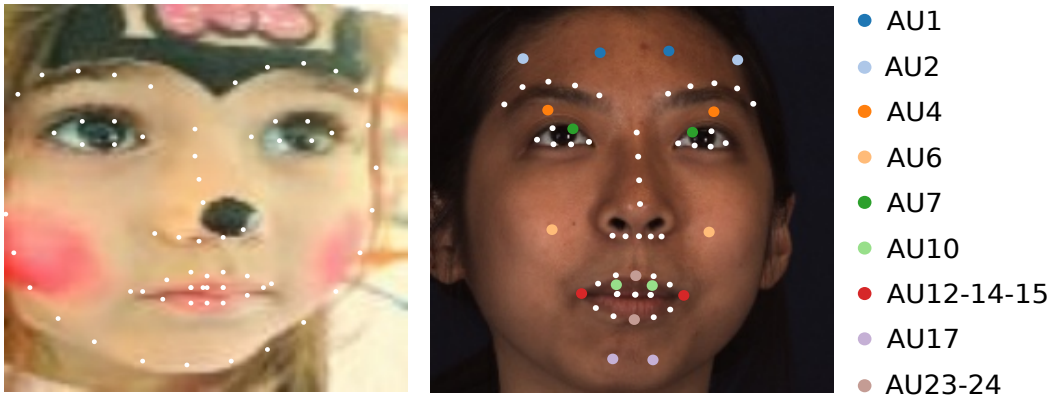


Figure 3.6: Example of images with associated landmarks with $L = 68$ on RAFDB (left) and both landmarks with $L = 49$ and AU-wise centers of interest used in [29, 59, 60] on BP4D (right).

AU	Description	Location
1	Inner brow raiser	1/2 scale above inner brow
2	Outer brow raiser	1/3 scale above outer brow
4	Brow lowerer	1/3 scale below brow center
6	Cheek raiser	1 scale below eye bottom
7	Lid tightener	Eye
9	Nose wrinkler	1/2 scale above nose bottom
10	Upper lip raiser	Upper lip center
12	Lip corner puller	Lip corner
14	Dimpler	Lip corner
15	Lip corner Depressor	Lip corner
17	Chin raiser	1/2 scale below lip
23	Lip tightener	Lip center
24	Lip pressor	Lip center
25	Lips part	Lip center
26	Jaw drop	1/2 scale below lip

Table 3.2: AU centers of interest from [60], 'scale' refers to the distance between the two inner eye corners

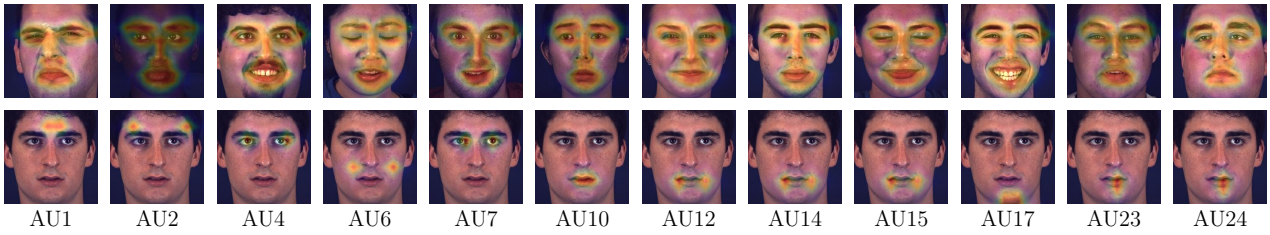


Figure 3.7: Examples of the two considered prior. In the top row, the *mono-prior* is displayed for several input images. In the bottom row the *multi-prior* priors are displayed for each AU on the same image.

Then, each of those centers is associated to a spatial distribution that is a dirac centered on its position. A gaussian kernel with standard deviation $\sigma = 1$ is then applied to each of those spatial distributions, and the resulting center images associated with the same tasks are averaged together and renormalized. In a nutshell, the probability of grid point (i, j) in the resulting heatmap for task t is given by:

$$\mathbf{h}^t_{i,j} = \frac{\sum_{k=1}^{C} \exp(-\|\tilde{\mathbf{c}}^t_k - \begin{bmatrix} i & j \end{bmatrix}\|_2^2/\sigma^2)}{\sum_{i,j}\sum_{k=1}^{C} \exp(-\|\tilde{\mathbf{c}}^t_k - \begin{bmatrix} i & j \end{bmatrix}\|_2^2/\sigma^2)}. \tag{3.30}$$

In this work we investigate two types of landmarks based priors: first, the mono-prior as in [3] that consists in considering that all landmarks as centers of interest. We refer to it as the *mono-prior* method because it constrain all tasks with the mono-prior. Second a prior that is AU specific and use the centers of interest in Table 3.2. We refer to it as the *multi-prior* because it defines 10 different priors. Figure 3.7 provides heatmap examples for each of those priors.

Mean of Heads Guidance

Now that we have appropriate ground truth heatmaps, we first inspire from the approach in [3] to constrain our transformers attention. In particular, the work in [3] aims at guiding a convolutional network with prior knowledge about the importance of each pixel. For that purpose, it first estimates pixel importance maps refered to as attribution maps that simply consists in gradients of the prediction w.r.t each pixel of the image (similar to equation 2.29). Then, it forces those attribution maps to resemble ground truth heatmaps. More precisely, it proposes a priviledged attribution loss that encourage the mean of a fraction of the attribution maps channels to resemble the prior heatmaps. The inspiration

behind such mean of channels strategy is to provide a constraint that is relatively loose so that the proportion of channels that are not affected by the constraint are free to look out of the ground truth heatmap scope.

The main drawback of such method is that it requires two backward passes at train time: a first backward pass to estimate the attribution maps that are necessary to compute the privileged attribution loss, and a second backward pass for supervising the network with both the loss of the task at hand and the privileged attribution. As a consequence, this method yields significant additional computational cost.

On the contrary, in transformer architectures, some information about patch importance is readily available in the forward pass under the form of the cross-attention matrices $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N_H)} \in \mathbb{R}^{T \times h \times w}$. Therefore, we propose to extend the work in [3] to transformer achitectures by using cross-attention matrices instead of attribution maps. More formally, for a given fraction f of constrained heads, we introduce an attention-based counterpart to the priviledged attribution loss (PAL) in [3] that we naturally refer to as Priviledged Attention Loss (PAtteL) and that we define as follows:

$$\mathcal{L}_{PAtteL}(\theta) = -\sum_{t,i,j} \mathbf{h}^t_{i,j} \log \bar{\mathbf{A}}^t_{i,j}, \tag{3.31}$$

where $\bar{\mathbf{A}}$ is the mean of a fraction $f \in [0, 1]$ of the attention matrices so that $n_{fh} = \lfloor f \times N_H \rfloor$ heads are constrained:

$$\bar{\mathbf{A}}^t_{i,j} = \sum_{k=1}^{n_{fh}} \frac{1}{n_{fh}} \text{softmax}(\mathbf{A}^{(k)}) \in \mathbb{R}^{T \times h \times w}. \tag{3.32}$$

Similarly to what is done in [3], $\mathcal{L}_{PAtteL}(\theta)$ is added to the total loss and provide the n_{fh} first heads with expert prior based guidance while leaving the $n_h - n_{fh}$ remaining heads free.

One-to-one head-to-prior guidance

If our privileged attention loss provides a fraction of attention heads with the freedom to look out of the scope of the prior heatmaps, it still constrains the attention of several heads to look at roughly the same image parts. In particular, this may prevent complementarity between heads and consequently degrade performance. Critically, the head-based attention visualisation in [6] tends to confirm this claim as it shows that each head of a unsupervisedly trained ViT attends semantically different zones of the image.

In the light of those remarks we propose to investigate a head-to-prior strategy that assigns priors to heads in a one-to-one fashion. For that purpose, we slightly modify the way we use the ground truth priors. While previously, each task was associated with its prior, we now consider the set of ground truth heatmaps priors independently of the tasks it is associated to. To clarify, we assume that we dispose of P priors $\mathbf{h}_1, \dots, \mathbf{h}_P$ that may be relevant to all the task at hand. Then, we define a matrix of assignement $\mathbf{M} \in \{0,1\}^{n_h \times P}$ that is general to all self-attention layers and all tasks s.t $\mathbf{M}_{h,p} = 1$ means that head h is constrained with prior $\mathbf{h}_p$. Then, we define our head-to-prior loss $\mathcal{L}^t_{h2p}$ for a given task t as follows:

$$\mathcal{L}^t_{h2p}(\theta) = \frac{1}{n_h} \sum_{h=1}^{n_h} \sum_{p=1}^{P} \frac{\mathbf{M}_{h,p}}{\sum_{p=1}^{P} \mathbf{M}_{h,p}} \ell(\mathbf{h}_p, \mathbf{A}^t_h) \tag{3.33}$$

where ℓ is the classical spatial categorical crossentropy and the final additional loss term is given by:

$$\mathcal{L}_{h2p}(\theta) = \sum_{t=1}^{T} \mathcal{L}^t_{h2p}(\theta) \tag{3.34}$$

Contrary to $\mathcal{L}_{PAtteL}$, $\mathcal{L}_{h2p}$ constrains each head to a different prior encourage heads to attend different locations and consequently enhance collaboration between heads.

In the upcoming section, we investigate the performance of the two previously defined landmarks-based guidance strategies on both RAFDB and BP4D.

3.5.2 Experiments

Implementation Details

RAFDB is a Facial Expression Recognition (FER) dataset in which $15k$ images ($10k$ in train, $2k$ in valid and $3k$ in test) are annotated with a categorical label that correspond to one of the 7 Ekman's basic emotion. We use it as a sanity check to verify that the *mono-prior* prior constraint provide a performance boost in a situation where landmark information is known to help improve performance [3]. For that purpose we consider FER as a single task categorical classification problem, therefore we simply use the MTTS architecture with $T = 1$ and a dense layer with output size 7 and a softmax activation on top of it. As far as optimization is concerned we keep the same parameters as for preliminary experiments on BP4D except for the learning rate exponential decay that we set to $\beta = 0.9$.

For **BP4D**, as we are provided with AU-wise priors we can compare the *mono-prior* and *multi-prior* privileged attention method with the *head-to-prior* constraint. In particular for the *head-to-prior* constraint, we set the assignement matrix $\mathbf{M}$ to a diagonal matrix with 1 for the 10 first diagonal elements and 0 elsewhere. Therefore, each of the 10 first heads are assigned to one of the 10 different AU-wise prior and the last 2 heads are left unconstrained.

Method Validation on RAFDB

Table 3.3: Comparaison between our Transformer-based PAtteL and PAL [3] for different proportion f of constrained heads and channels respectively.

Method	Prop(f)	Mean Valid Acc.	Mean Eval Acc.	Max Eval Acc.
MTTS + PAttel	0.0	86.2 ± 0.2	88.8 ± 0.1	88.9
MTTS + PAttel	0.25	$\mathbf{86.7 \pm 0.4}$	$\mathbf{89.0 \pm 0.4}$	**89.6**
MTTS + PAttel	0.5	86.6 ± 0.4	89.0 ± 0.3	89.4
MTTS + PAttel	0.75	86.4 ± 0.6	88.9 ± 0.3	89.0
MTTS + PAttel	1.0	86.5 ± 0.2	88.8 ± 0.3	89.2
Resnet50 + PAL [3]	0.0	-	88.6	-
Resnet50 + PAL [3]	0.5	-	**89.6**	-

Table 3.3 shows both valid and test accuracy result of the *mono-prior* method for different proportions of constrained heads f on RAFDB. Similar to the claims in [3], it shows that increasing the proportion of constrained heads f from 0 to 0.5 provides a performance boost because it guides the model attention toward salient facial areas. However, this boost tend to disappear for higher values of f as constraining too many heads prevent the network from exploring other features. In particular, that may be necessary in case of noisy landmarks annotations.

Furthermore, it is noticeable that the performance boost provided by the gradient-based method seems superior to the one based one transformers (at least in average). A possible explanation for this is that the gradient-based method offers a better estimation of the influence of each pixel on the final prediction. Indeed, attention may be explicit in transformers, yet the relationship between the features extracted by attention mechanism and the final prediction are unclear as the extracted features are passed through many MLP modules and dense layers. For example, one could imagine an extreme situation in which the final dense kernel of the attention module (Figure 3.1) discards all the features from the constrained heads (by setting its $f \times d$ constrained heads associated columns

to 0 for example), hence completely negating the guidance influence. As a consequence, gradient-based methods offer a stronger constraint which explains its better performance. Nonetheless, it remains that the transformer-based method provides a performance increment which in a sense validates its capacity to exploit landmark-based priors for facial expression recognition. Therefore, we now apply our transformer-based method on BP4D for AU detection using the validated best head proportion $f = 0.25$.

Evaluation on BP4D

Table 3.4 reports the performance of MTTS with both the *mono-prior* and the *multi-prior* strategy. We compare our method both with and without self-attention layer. Unfortunately, we observe that, despite the fact that (a) our method provides an increment by guiding the network attention toward landmarks on RAFDB (b) Prior works have shown that facial landmarks information could help increase AU detection performance [61, 60, 29], both the *mono-prior* and the *multi-prior* strategy fail to provide a performance boost in AU detection. On the contrary, it is noticeable that the *head-to-prior* method without self-attention is the only one that outperforms the baseline. This endorses our previously-stated intuition that encouraging each head to look at different priors provides better synergy between heads than encouraging the mean of heads to look at a single prior. Furthermore the superiority of the version without self-attention shows that the patch-based combination performed in those layers may hurt spatial coherence. This brings food for thought on how to transmit spatial information across self-attention layers.

Table 3.4: Comparaison of different landmarks based guidance method on BP4D

Method	Constraint	N_{SA}	Mean F1Score
Baseline	✗	1	62.6 ± 0.8
PAtteL	mono-prior	0	61.9 ± 0.6
PAtteL	mono-prior	1	61.8 ± 0.9
PAtteL	multi-prior	0	61.9 ± 0.6
PAtteL	multi-prior	1	61.8 ± 0.9
H2P	multi-prior	0	61.6 ± 1.2
H2P	multi-prior	1	62.1 ± 0.2
H2P	multi-prior	0	$\mathbf{62.8 \pm 0.9}$
H2P	multi-prior	1	62.3 ± 0.7

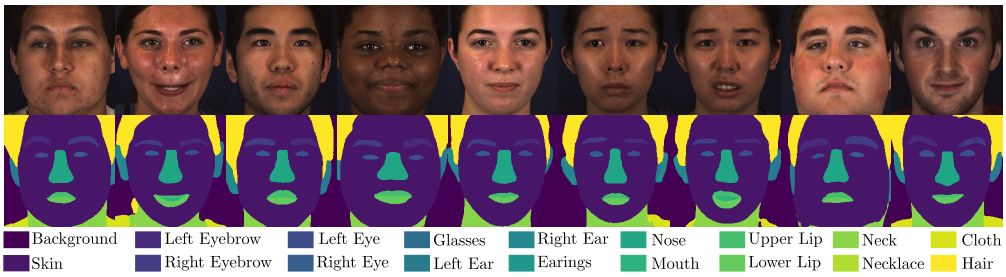


Figure 3.8: Example of images with their associated face parsing maps for BP4D

In the next section we build upon the promising performance of the *head-to-prior* method that were obtained by following an intuition based on [6]. In particular, we further inspire on the attention visualisation in [6]. Indeed, it shows that the different heads not only look at different parts of the image, but also learns to look at different semantical parts of the image. Therefore, we draw inspiration from this observation to design a cross-attention prior similar to [14] that uses a face parsing prior to help cross-attention heads focus on different semantical parts of the face.

3.5.3 Face Parsing based attention guidance

Similar to the work in [14], we design a soft mechanism for guiding the cross-attention heads with face parsing based prior. For that purpose we predict segmentation maps on BP4D faces using the off-the-shelf *keras face toolbox*. It classifies each image pixel in 18 classes. Figure 3.8 display examples of the obtained face parsing maps. To further adapt those face parsing maps to the needs of our method, we manually group face parsing elements that are not informative about facial expressions (Typically background, glasses, ears, etc) into a single class resulting in a total of $F = 10$ face parsing classes.

Computing face parsing tokens

The aim of our method is to provide the CA module with a mechanism to query patches based on the face parsing classes that are present in it. Then, it will use this mechanism to ensure that each head of the CA module queries different face parsing classes.

To reach that goal, the first step is to encode patches based on the face parsing classes they contain. For that purpose, we begin by associating face parsing classes to learnable tokens $\theta_{\mathbf{S}} \in \mathbb{R}^{F \times d}$. Following this, $\theta_{\mathbf{S}}^{(f)}$ is the token for the f-th face

parsing class. Then, for each patch (i, j), we aggregate the face parsing tokens in a patch-based fashion to get a compact representation that describes the face parsing classes present in each patch:

$$\mathbf{S}_{i,j} = \frac{\theta_{\mathbf{S}}^T \mathbf{f}_{i,j}}{\|\mathbf{f}_{i,j}\|_2} \in \mathbb{R}^d, \tag{3.35}$$

where $\mathbf{f}_{i,j} \in \mathbb{R}^F$ contains the frequency of each class in the patch (i, j) of the face parsing map $\mathbf{s}$. As a consequence, segmentation tokens $\mathbf{S}_{i,j}$ holds the face parsing information for patch (i, j).

Soft Face Parsing-based prior

Now that we are provided with a representation of the face-parsing information held in each patch, the next step is to guide each head of the CA module toward paying attention to different face parsing classes. For that purpose, we give ourselves the same kind of assignment matrix $\mathbf{M} \in \{0, 1\}^{N_H \times F}$ as in the *head-to-prior* method in 3.5.1. In other words, $\mathbf{M}_{h,f} = 1$ if head h is constrained to look at face parsing class f. To simplify, we constrain $\mathbf{M}$ to have up to a single 1 in each row. Therefore each head is constrained to look at up to 1 face-parsing class. We then use this matrix to assign a guiding token to each head of the CA module:

$$\theta_{\mathbf{S}_h} = \mathbf{M}\theta_{\mathbf{S}} \in \mathbb{R}^{n_h \times d}. \tag{3.36}$$

Then, to constrain each head to look at its associated face parsing token, we inspire from the method in [14] and add the face parsing prior before applying the softmax activation to the attention matrix. Concretely the attention for head h that compute the relevance of patch (i, j):

$$\mathbf{A}_{i,j}^{(h)} = \frac{\tilde{\mathbf{Q}}^{(h)} \tilde{\mathbf{K}}_{i,j}^{(h)} + \lambda_{\mathbf{S}} \mathbf{S}_{i,j}^T \theta_{\mathbf{S}_h}^{(h)}}{\sqrt{(\lambda_{\mathbf{S}}^2 + 1)d/H}} \tag{3.37}$$

where $\lambda_{\mathbf{S}}$ controls the importance of the face parsing prior. With such formulation, the additional term $\mathbf{S}_{i,j}^T \theta_{\mathbf{S}_h}^h$ mainly depends on the frequence of the $\theta_{\mathbf{S}_h}^h$ token in the patch (i, j) so that each head look at the class that is assigned by $\mathbf{M}$ to token $\theta_{\mathbf{S}_h}^h$.

The main difference between soft face parsing-based method and the H2P method in section 3.5.1 lies in the way the information is encoded. In fact, a face-parsing version of H2P is easily obtained by considering the binary classification heatmaps associated with each face parsing class as prior. However, to constrain attention heads with those priors, we would need to resize them from the image

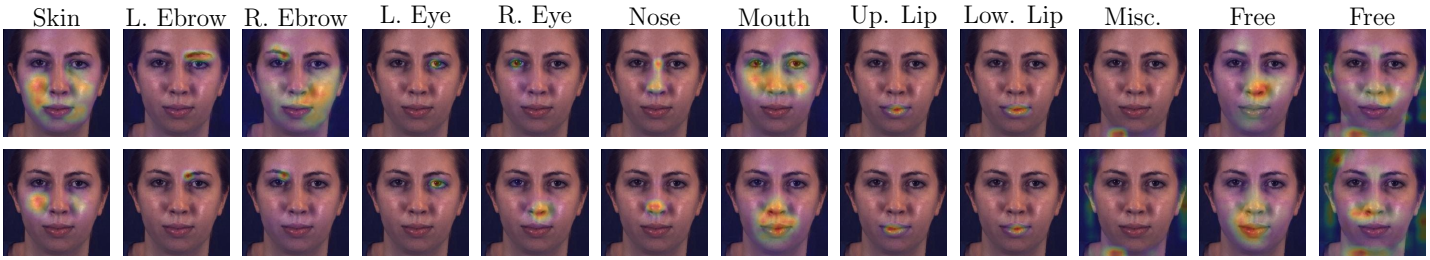


Figure 3.9: Visualisation of face parsing constrained cross-attention heads for AU1 and AU24 with $\lambda_{\mathbf{S}} = 1.0$

original size $H \times W$ to the attention heads $h \times w$. Depending on the used resizing algorithm, this method may loose information about small face parts which are important for facial expression analysis (eg. Eyebrows).

On the other side, the aggregation mechanism proposed in equation 3.35 keeps the information about the proportion of each face parsing class in a patch. Therefore it allows the CA module to query even the smallest parts of the face.

In the next section, we investigate the influence of the face parsing cross-attention prior. In particular, we show that it provides a performance boost on BP4D and DISFA which further validates the hypothesis that multi-head attention guidance is more efficient when using different priors for different heads.

3.5.4 Experiments

Implementation Details

In the following experiments, we use the same baseline network as in 3.5.2. For the head to face-parsing class matrix $\mathbf{M}$ we set the 10 first diagonal elements to 1 so that each of the 10 first heads are constrained with one of the 10 available face parsing classes. The two last elements of the diagonal are set to 0 so that the 2 last heads are left unconstrained.

Cross-Attention Visualisation

Figure 3.9 displays examples of cross-attention matrices for each head of the first cross-attention layer for AU1 and AU24. Those visualisations empirically validate the capacity of our face parsing-based constraint to guide each head toward a different face parsing class. The only head matrices that does not seem to be properly guided are the one associated with the mouth class. Indeed, the associated heads do not look at the mouth at all. In fact, the reason for such result is simply

that the mouth class of our face parser refers to the mouth interior which is not displayed on the image. Therefore, none of the patch-based segmentation token $\mathbf{S}_{i,j}$ holds a mouth interior token component. As a consequence, the influence of the guiding term is almost zero (as learnable tokens are initialized as standardized independant gaussian vectors), which explains that those heads are left unsupervised. Finally, it is noticeable that our soft constraint is relatively loose as it lets the attention matrix of each task (which are constrained the same way) attend to significanttly different zones.

Comparative study on BP4D

Method	$\lambda_{\mathbf{S}}$	N_{SA}	Mean F1Score
Baseline	0.0	1	62.6 ± 0.8
Landmarks H2P	✗	0	62.8 ± 0.9
Landmarks H2P	✗	1	62.3 ± 0.7
Face Parsing Prior	0.01	0	62.7 ± 0.9
Face Parsing Prior	0.01	1	62.5 ± 0.3
Face Parsing Prior	0.1	0	$\underline{62.9 \pm 0.7}$
Face Parsing Prior	0.1	1	62.4 ± 0.5
Face Parsing Prior	1.0	0	62.1 ± 0.3
Face Parsing Prior	1.0	1	61.5 ± 0.4
SOTA results [73]	-	-	**64.5**

Table 3.5: Comparaison between Landmarks and Face Parsing Guided Methods on BP4D

Table 3.5 display a comparison between the performance of our face-parsing method on BP4D for different values of $\lambda_{\mathbf{S}}$ and the ones of the previously defined landmarks based *head to prior*. For low values of $\lambda_{\mathbf{S}}$ (0.01 and 0.1) the face parsing prior helps the different heads focus on the different parts of the face. This, in turn, improves collaboration between the different heads and consequently provide a boost of performance compared to the baseline. For higher values of $\lambda_{\mathbf{S}}$, the applied constraint is too strong and deprive the different heads from the capacity to attend elsewhere when needed, hence degrading the predictive performance.

Furthermore, similarly to what we observed in the previous section, we observe that removing the self-attention attention layer that is directly after the convolutional backbone improves performance. This further endorses our belief that the self-attention layer looses the spatial coherence of the patch representation. From

this point of view, using a spatial prior after this dilution is less efficient than doing it directly on top of the convolutional network.

Most importantly, the face parsing prior is on par (even slightly above) with the landmarks based prior. First, it further validates the intuition that each head should look at different priors as our two methods that do so outperform baseline results. Second this is rather promising as the two prior guiding methods are not mutually exclusive. As a matter of fact, we believe that exploring combination of those two priors could be an interesting line of improvement for our transformer based method. In particular, it could be interesting to constrain the first cross-attention layers with the lower level face parsing prior and the upper layers with the higher level landmarks based prior.

Comparative study on DISFA

Method	$\lambda_{\mathbf{S}}$	N_{SA}	Mean F1Score
Baseline	0.0	0	63.0 ± 1.6
Baseline	0.0	1	61.2 ± 1.3
Face Parsing Prior	0.01	0	63.0 ± 1.3
Face Parsing Prior	0.01	1	63.0 ± 0.7
Face Parsing Prior	0.1	0	$\mathbf{64.2 \pm 0.8}$
Face Parsing Prior	0.1	1	63.7 ± 0.7
Face Parsing Prior	1.0	0	62.2 ± 1.1
Face Parsing Prior	1.0	1	62.4 ± 1.4
SOTA results [73]	-	-	<u>63.9</u>

Table 3.6: Comparaison between Landmarks and Face Parsing Guided Methods on DISFA

Table 3.6 reports the performance of our method on DISFA. Interestingly enough, the observed results empirically confirms most of the claims that we made based on the BP4D results. Indeed, we observe almost exactly the same performance variation based on the value of $\lambda_{\mathbf{S}}$: $\lambda_{\mathbf{S}} = 0.1$ provides with the best performance boost and values above it degrades performance.

Furthermore the claim about the self-attention layers hurting spatial coherence is also be verified on DISFA as it degrade performance in almost every setting. Hence, we believe that learning to transmit spatially salient information through self-attention layers could be an interesting line of research to explore.

Finally, from both Table 3.5 and 3.6 we observe that our face parsing method outperforms state-of-the-art results on DISFA but not on BP4D. A possible ex-

planation for this gap is that DISFA dataset display lower face variability than BP4D. Therefore the face parsing prior may transmit information that are more useful for DISFA than for BP4D. However, the increment observed on BP4D is still promising and encourage us toward exploring more sophisticated attention head constraints.

3.6 Conclusion and future works

3.6.1 Discussion

In this chapter, we investigated transformer-based attention guidance strategies. In particular, we started from the empirical observation that the attention maps for our baseline multi-task Action Unit detection transformer didn't assign importance to the correct AU related zones. To circumvent this issue, we tried to adapt the recent convolutional based mean of channel attribution constraint [3] to transformer architectures. This method provides a slight increment on the original use-case presented in [3]. Yet the obtained boost did not meet our expectations that were based on the increment reported in [3]. Worst, we show that such constraint have a negative effect when applied to our AU detection use-case.

To justify those disappointing performances, we argued that encouraging several heads to look at the same zones reduces collaboration between the different attention heads. Strong with this observation, we drew inspiration from [6] and came up with a constraint that encourages each head to attend to different AU relevant locations. Contrary to the previous method, such approach favors independence between the attention learned by each head and consequently bolsters collaboration between heads. We empirically validate this intuition by showing that our method outperforms our transformer baseline.

Then we tried to further mimic the visualisation obtained in [6]. More precisely, we notice that the ViT attention heads in [6] tend to focus on different semantical parts of the image. To do so we proceeded similarly as in [14] to design a method that leverages a token-based mechanism to softly introduce a face parsing prior. This prior helps each head focus on a different part of the face. Again, we empirically validated our face parsing-based method increment on BP4D. We further pushed the validation by showing that the provided increment is consistent on DISFA. Interestingly enough, the face-parsing based method is slightly better than the landmarks based approach. This further validates that constraining each head to different semantical element is a promising way to guide multi-head attention.

As a conclusion, even if our results did not completely bridge the gap to reach state-of-the-art in AU detection, it is worth noticing that they still provide consistent performance increment. Therefore they constitute a step toward better guidance of the attention mechanism in transformers.

3.6.2 Future Work

In future work, we would like to investigate the use of self-attention layers on tasks tokens to better model task interactions. More precisely, we would like to follow the work in [22] that refines the decoder attention module by adding a self-attention layer on tokens just before the cross-attention layer that use tokens as queries and patches as keys and values. Actually, we already attempted to use such mechanism in [73]. However we observed that adding those self-attention layers did not bring any significant performance improvement.

A possible explanation is that, by applying self-attention layers over task tokens, we loose the information about which token identifies which task. Indeed, without those self-attention layers, each task token learns to pay attention to the parts of the image that are relevant to its task. In that situation, there is an association between a task and the token that is used to predict it. For that purpose the name task token is fully justified.

On the contrary, when passed through a self-attention layers, the tokens are combined together and different combinations of tokens are used to select parts of the image that are relevant for each task. In that scenario a single token is no longer associated to a specific task as it is used to select information relevant to several tasks. In that extent, the name task tokens is no longer relevant.

In fact, such architecture could be thought as an ensemble of learners (one per token) that interacts through self-attention to solve all the tasks at once. In the light of this latter observation, there is no a priori reasons to keep the number of tokens equal to the number of tasks. Therefore a first line of research could consist in finding the optimal number of tokens for solving a multi-task problem.

A second line of research could be to explore the exact opposite direction. This means trying to keep the task-to-token association while still using the self-attention layer to model dependency between tasks. For that purpose, we could inspire from cross-stitch networks [46] that first pretrains independent network to predict different tasks and then learns convex combinations of each network features to exploit dependencies between tasks.

We believe that this method could be adapted to transformer architectures. In particular a possible adaptation would be to first pretrain the transformer with short-circuited self-attention layers and to free self-attention layers only after the

pretraining convergence. This way, the pretraining step would enable each task token to learn to select parts that are relevant to each task. In that sense, and because transformer architectures are highly expressive, we can argue that the transformer would learn independant networks that predicts each task. Then freeing the self-attention layers would allow the transformer to learn convex combinations of the features of each of those independant predictors. The advantage of this method compared to [46] is that the learned combination of the predictors feature could depend on the image. This flexibility could improve performance when provided with sufficient training data.

Chapter 4

Noise mitigation in imbalanced situations for AU detection

4.1 Introduction

As mentionned in section 1, Action Unit annotation is an extremly tedious process. Indeed, it requires specialists to annotate videos frame by frame. Furthermore, most AU involve subtle face movements. Those movements may be difficult to detect even for expertly trained annotators. As a consequence, the publicly available AU datasets display annotation noise. Examples of noisy annotation are displayed in Figure 4.1. Fitting a deep neural network on those examples may naturally degrade generalisation performance. In this chapter, we aim at avoiding this annotation noise pitfall.

For that purpose, we aim at taking advantage from the recent successes of label smoothing [69] for noise mitigation [38] and over-confidence reduction. However, vanilla label smoothing reduces over-confidence uniformly for all classes. Therefore, applying it in imbalanced situations may worsen the pre-existing under-confidence in the minority class. For that purpose, we propose Vanilla Asymetric Label Smoothing (VALS) that consists in separating smoothing coefficients based on whether labels are positive and negative. Such separation provides the flexibility to apply more smoothing to the majority class examples and consequently avoid worsening the minority class under-confidence problem. We further embrace such philosophy by proposing Robin Hood Label Smoothing (RHLS). RHLS consists in smoothing only the majority class. Furthermore, it scales the smoothing coefficient of each AU based on its empirical frequency (the higher the frequency the larger the smoothing coefficient). By doing so, RHLS introduces a probability to take examples from the rich (over-represented) class to give them to the

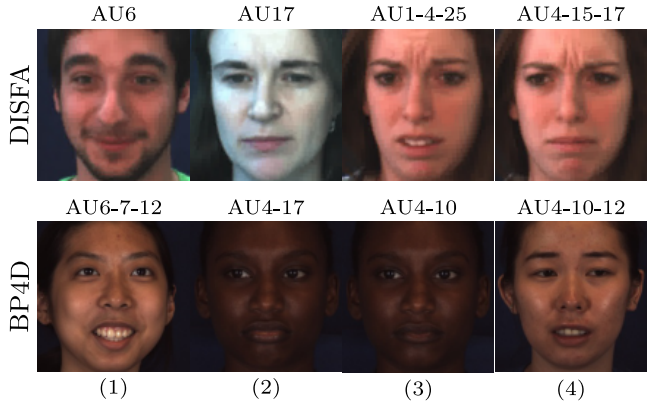


Figure 4.1: Noisy examples from DISFA and BP4D. For example, the first face from DISFA is annotated with neither smile (AU12) or eyebrow raise (AU1-2). Fitting on those examples may prevent the network from properly understanding which zones are involved in each AU and degrade performance.

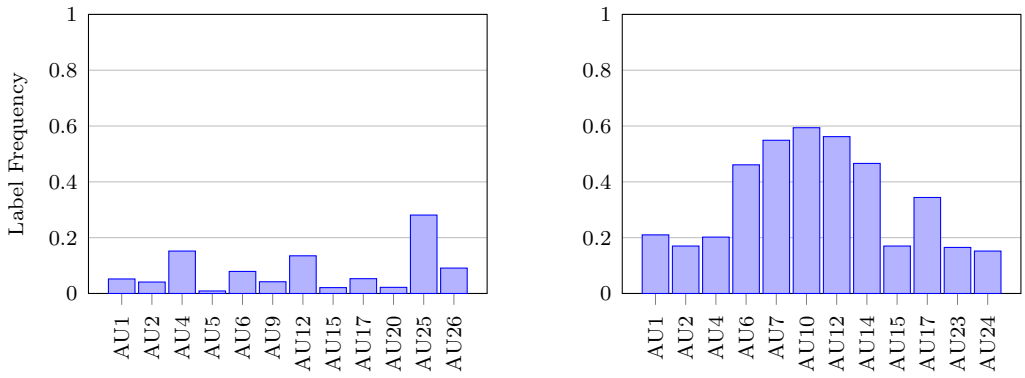
poor one (under-represented). Consequently, it reduces both the imbalance-based over-confidence issue and the negative impact of noisy majority class examples. To summarize, our contributions are as follows :

- We introduce VALS and RHLS that adapts label smoothing to imbalanced situations. To do so, it applies smoothing to the majority class only. Consequently, it mitigates both imbalance over-confidence issues and the negative impact of majority class noisy examples.
- Experimentally, we show that the performance of an AU detection system benefits from the use of VALS and RHLS without any additional computational overhead. More precisely, we demonstrate that applying VALS on a modern multi-task baseline display promising performance on BP4D, and outperforms state-of-the-art performance on DISFA. Furthermore, RHLS seems to be particularly beneficial on DISFA as it surpasses both VALS and *a fortiori* the state-of-the-art performance.

4.2 Related Works

4.2.1 Dealing with imbalance in Action Unit Detection

Action Unit detection datasets are imbalanced meaning that most annotated frames feature neutral faces. Indeed, Figure 4.2 shows that, in the most widely



(a) DISFA (b) BP4D

Figure 4.2: Action Unit frequencies for BP4D and DISFA. BP4D is slightly imbalanced toward negativeness compared to DISFA where most AU are represented in less than 1/10 frame. Training an AU detector in such imbalanced situations may push toward only negative examples. Similarly, the large variations in AU frequencies on DISFA may result in variations of the contribution of each AU to the multi-task loss. In particular, least frequent AU may have lower loss contributions and consequently get ignored during training.

used AU datasets, several AU have low empirical frequencies. Training in such imbalanced setting is likely to push a model toward predicting only the majority class. Furthermore, we also observe that imbalance levels vary across AU. Therefore, if the network predicts only negatives, this imbalance variations will result in low loss contribution for the least frequent AU and consequently prevent the network from learning them.

In order to circumvent those different issues, several approaches attempted to adapt multi-task binary cross-entropy loss to imbalanced situations. The work in [28] replaces the binary cross-entropy associated with each AU by a balanced cross-entropy. For each AU, the balanced cross-entropy rescales the loss contribution of positive and negative examples using the empirical frequency of the concerned AU. In the same spirit of scaling loss terms with frequencies, the methods in [61, 60] weighted the loss contribution of each AU based on its empirical frequency. Contrary to the balancing loss in [28] that aim at solving AU imbalance on each AU independently, this method is more focused on compensating imbalance variations and ensuring that none of the considered Action Unit gets ignored in training.

Furthermore, numerous approaches [61, 60, 23] have used additional loss terms that further penalizes false negatives. In particular, the methods in [61, 60] both used a frequency weighted Dice coefficient loss [45] that is based on the F1-score

metric which is known to be adapted to imbalanced situations. In the same vein, authors in [23] used additional Tversky loss [28] term which is simply an extension of the Dice coefficient. Those dice coefficient-based losses may be appealing because of their rebalancing properties [28], however the gradients derived from those losses often display more variance than the ones of classical binary cross-entropy which may result in training instabilities.

Finally, authors in [63] proposed to fight imbalance by weighting each example based on a measure of the model uncertainty. Indeed, they argue that, for a given example, uncertain model prediction highlights the lack of similar training examples. To circumvent this issue, they assign higher weights to the loss contribution of uncertain examples. However, we believe that such method is risky when used on noisy datasets, which is the case of Action Unit datasets [99]. Indeed, the model is likely to be uncertain on noisy examples. As a consequence, assigning higher weights to those examples will encourage the model toward fitting the corrupted supervision signal and may in turn degrade performance.

Imbalance is an extensively studied problem of the AU litterature. On the contrary, methods that aim at reducing the impact of noisy AU labels attracted less attention. For that purpose, we now dig into more general noise mitigation methods.

4.2.2 Noise mitigation methods

The high expressive power of deep neural network allows them to fit arbitrary/random data [98]. Therefore, in the presence of noise, such expressive power may come as a burden. Indeed fitting noisy annotations is likely to hurt the generalisation performance. To tackle this problem, several approaches attempted to design noise-aware learning strategy. In particular, those strategies can be roughly classified into two groups: the sample selection approaches [41, 24, 20] and the loss correction strategies [50, 18, 67, 54, 7, 55].

Sample Selection

Sample selection consists in selecting which examples should be used during training. In a noisy label setting those examples are naturally the clean labels. Yet unfortunately, without full dataset verification, distinguishing clean labels from corrupted ones is a very challenging task. In order to tackle that challenge, several methods used meta learning. For example, authors in [41] propose to decouple "when to update" from "how to update". More concretely, they propose to train two different networks in parallel and to update those networks using only the ex-

amples on which they disagree. This strategy favors the two networks agreement over ground truth supervision. Therefore, if the label of an example is corrupted but the two networks consistently predict the true label, this method will ignore the corrupted supervision signal.

In the same vein, Co-teaching [20] also trains two networks, each one selecting low loss examples to feed to the other network. Finally, several approaches complemented the sample selection step with label refurbishment [62] that aim at curing noisy examples. In particular, the strategy presented in [62] first select examples whose predicted class is consistent but wrong (w.r.t the annotation) over several epochs. Then it considers that the consistently predicted class is the true class and consequently reannotate the selected examples with it. Similarly, the work in [82] proposes a self-cure network that relabels the most uncertain examples using the class with highest probability in the network prediction.

Loss Correction

Loss correction takes its name from the fact that, in the presence of noisy data, minimizing the classification loss (eg. categorical cross-entropy) between the network prediction and some corrupted label pushes the network toward wrong prediction. In that sense, the loss needs to be corrected to adapt to noisy settings. To do so, early works [54] proposed to correct the loss with a term that encourage a network prediction toward consistency across epochs. The incentive behind this approach is to prevent supervision with noisy labels from modifying predictions on examples that are already properly classified.

Concurrently, several methods [50, 18, 67] attempted to correct the classification loss by modelling the corruption noise. The most natural approach to do so is by estimating the corruption matrix, *i.e.* the matrix that holds the probability of observing corrupted label j knowing that the clean label is i. Both approaches in [18] and [67] followed this idea. In particular, authors in [67], aim at training a base network to predict clean labels. For that purpose, they propose to multiply the base network prediction by a learned square noise matrix that is tasked with estimating the corruption matrix. In fact, the idea is that, by supervising the final output with the corrupted labels, the noise matrix may learn the corruption matrix which will push the base network toward predicting the clean labels.

In the saim vein, the work in [50] proposed two approaches to model label noise: a forward procedure and a backward procedure. On the one hand, the forward procedure is similar to what is done in [67, 18] as it simply multiplies a base network prediction by an estimate of the corruption matrix before supervising it with the classical classification loss. On the other hand, the backward procedure

slighly differs from it. Indeed, instead of using the corruption matrix in the forward pass, it introduces it in the loss computation. More precisely, it first uses the prediction of the base network along with the corrupted labels to compute of a vector of loss associated to each class. Then, it multiplies this vector of loss by the inverse of the corruption matrix. Theoretically, if the corruption matrix is correctly estimated, this approach is equivalent to supervising the base network with the clean labels.

The main challenge of all these noise modelling strategies is to properly estimate the corruption matrix. In particular, the work in [67] showed that, in the forward procedure, getting a learned matrix to properly estimate the corruption matrix is difficult. Indeed many degenerate configurations are as optimal as the one where the noise matrix is a perfect estimate of the corruption matrix. For example, if the base network is expressive enough it may directly learn to predict the noisy labels while the noise matrix learns the identity matrix. In an attempt to avoid this pitfall, penalization of the noise matrix trace have been proposed in [67] to prevent the noise matrix from learning the identity. With a similar goal, the method in [18] proposed to first train the base network without the noise adaptation layer and to subsequently initialize the noise matrix with the resulting base network confusion matrix. Different from those two strategies, the approach in [50] estimates the corruption matrix by relying on clean examples for each class. More precisely, it proposes to first supervise a network with corrupted labels and then feed a clean example of each class to this network. If the network perfectly fits the corrupted labels distribution, the predicted probability of class j for a clean example of class i is the probability that clean label i gets corrupted to class j. Repeating this for all i consequently provides an estimate of the corruption matrix. A minor drawback for this method is that it requires to identify clean examples for each class. A more important line of criticism is that in practice, the trained network do not perfectly fit the noisy label distribution which degrades the quality of the corruption matrix estimation.

Other loss correction approaches include leveraging small noise-free validation set to learn to reweight [55] or clean [79] noisy examples, reweighting the examples loss contribution using active learning [7], curriculum learning [24] or uncertainty estimation [39, 82], and automatic noisy label re-annotation [79, 62].

Recently, the work in [38] showed that label smoothing [69] was competitive with loss correction approaches for noise reduction. Concretely, label smoothing modifies the classification loss by adding a probability α to replace ground truth annotation by an uniform distribution over all classes. Authors in [38] argues that such loss modification is close to the backward procedure in [50]. However, they also show that contrary to the backward procedure, label smoothing does not

aim at recovering the clean labels from the corrupted ones. In other words, label smoothing is not a noise modelling method. It is rather a model regularization method [69] that mitigates the influence of noisy examples by preventing the model from assigning them too much confidence.

The method we propose directly draw inspiration from the recent success of label smoothing at mitigating noise [38]. Yet, in preliminary experiments, we observe that vanilla label smoothing actually degrades AU detection performance. To explain this drop, we hypothesize that label smoothing may aggravate pre-existing AU imbalance-based under-confidence. To address that problem, we propose Vanilla Asymetric Label Smoothing (VALS) that consists in introducing separate smoothing coefficients based on whether the ground truth label is positive or negative. The incentive behind this separation is that it allows to limit the impact of confidence reduction on the minority class. Furthermore, we also propose Robin Hood Label Smoothing (RHLS) that further embrace this philosophy. Concretely, RHLS only smoothes the majority class and scales the smoothing coefficient associated with each AU based on its empirical frequency.

4.3 Label smoothing strategies for AU detection

4.3.1 Vanilla Label Smoothing

As mentionned in 4.2.1 AU detection involves subtle changes in skin texture that are difficult to detect, even for expertly trained annotators. As a consequence, the main available annotated datasets display label noise.

Prior work [38] showed that label smoothing could help mitigate the influence of annotation noise by reducing model confidence [69] and consequently preventing the network from over-fitting on noisy examples. Concretely, label smoothing introduces a probability α to replace the ground truth label by an uniform distribution over all classes.

Single Task Label Smoothing (LS)

As a consequence, in a single binary classification task the dataset empirical joint distribution $\hat{p}$ of input $\mathbf{x}$ and label y is modified as follows:

$$\mathrm{LS}_\alpha(\hat{p})(\mathbf{x}, y) = (1 - \alpha)\hat{p}(\mathbf{x}, y) + \alpha u(\mathbf{x}, y) \tag{4.1}$$

where u refers to the distribution that assign uniformly random labels to all sampled images:

$$u(\mathbf{x}, y) = u(y \mid \mathbf{x})\hat{p}(\mathbf{x}) = \frac{\hat{p}(\mathbf{x})}{2} \tag{4.2}$$

Therefore, the cross-entropy-based loss between the model estimated distribution p_θ and the empirical distribution in equation 2.5 becomes:

$$\mathrm{CE}(\mathrm{LS}_\alpha(\hat{p}) \,||\, p_\theta) = (1-\alpha)\mathrm{CE}(\hat{p} \,||\, p_\theta) + \alpha\mathrm{CE}(u \,||\, p_\theta). \tag{4.3}$$

If we denote p_i, the network prediction for example i, this term can be further developped as:

$$\begin{aligned}
\mathrm{CE}(\mathrm{LS}_\alpha(\hat{p}) \,||\, p_\theta) &= \frac{1}{N}\sum_{i=1}^{N}(1-\alpha)[y_i \log p_i + (1-y_i)\log(1-p_i)] + \frac{\alpha}{2}[\log p_i + \log(1-p_i)] \\
&= \frac{1}{N}\sum_{i=1}^{N}\tilde{y}_i \log p_i + (1-\tilde{y}_i)\log(1-p_i), \\
&= \frac{1}{N}\sum_{i=1}^{N}\ell(\tilde{y}_i, p_i)
\end{aligned} \tag{4.4}$$

where $\tilde{y}_i = (1-\alpha)y_i + \frac{\alpha}{2}$ and ℓ is the classical binary cross-entropy loss. This justifies the common label smoothing usage [69] that simply consists in replacing label y_i by $\tilde{y}_i$.

Multi-Task Vanilla Label Smoothing (VLS)

In a multi-task binary classification setting, the most widely adopted practice is to suppose that (a) the model predictions are independent conditionally to $\mathbf{x}$ (similar to the assumption in section 2.3.2) and (b) the classification task t is corrupted with probability α, *i.e* the marginal empirical distribution $\hat{p}^t$ with respect to task t is modified as follows:

$$\mathrm{LS}_\alpha(\hat{p}^t)(\mathbf{x}, y^t) = (1-\alpha)\hat{p}^t(\mathbf{x}, y^t) + \alpha\frac{\hat{p}(\mathbf{x})}{2}. \tag{4.5}$$

Therefore, the cross-entropy loss becomes:

$$\begin{aligned} \mathrm{CE}[\mathrm{VLS}_\alpha(\hat{p})|p_\theta] &= \sum_{t=1}^{T} \mathrm{CE}[\mathrm{LS}_\alpha(\hat{p}^t)|p_\theta^t] \\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \ell(\tilde{y}_i^t, p_i^t), \end{aligned} \tag{4.6}$$

where $\tilde{y}_i^t = (1-\alpha)y_i^t + \frac{\alpha}{2}$.

However, in section 4.4.2, we experimentally show that applying VLS ($\alpha = 0.1$) to the multi-task AU detection binary classification problem slightly degrades AU detection performance. We hypothesize that the imbalance in AU datasets is responsible for such performance drop. Indeed, as shown by figure 4.2, in the most popular AU datasets, several AU display low empirical frequencies. In particular, such imbalance has been shown to push model toward under-confident predictions for the minority class [30]. Therefore, by reducing confidence of both positive and negative examples, label smoothing may worsen the pre-existing confidence problem on the minority class, explaining the observed performance gap.

4.3.2 Vanilla Asymetric Label Smoothing

In order to address the aforementioned problem we extend label smoothing to Vanilla Asymetric Label Smoothing (VALS) that relaxes label smoothing by introducing separate smoothing coefficients for each ground truth class. The general intuition is that such relaxation provides ALS with the flexibility to target the smoothing on the high frequency classes, leaving the other ones untouched. For that purpose VALS introduces two separate probabilities α_+ and α_- that correspond to the probability for any task to be corrupted if its label y^t is positive or negative respectively. As a consequence, the marginal empirical distribution of task t becomes:

$$\mathrm{VALS}_{\alpha_+,\alpha_-}(\hat{p}^t)(\mathbf{x}, y^t) = y^t \mathrm{LS}_{\alpha_+}(\hat{p}^t)(\mathbf{x}, y^t) + (1-y^t)\mathrm{LS}_{\alpha_-}(\hat{p}^t)(\mathbf{x}, y^t) \tag{4.7}$$

Similarly as before, we can therefore derive the cross-entropy loss as follows:

$$\mathrm{CE}(\mathrm{VALS}_{\alpha_+,\alpha_-}(\hat{p}^t) \mid p_\theta) = \sum_{t=1}^{T} \mathrm{CE}(\mathrm{VALS}_{\alpha_+,\alpha_-}(\hat{p}^t) \mid p_\theta). \tag{4.8}$$

Furthermore, the right term of this equation can be developped as follows for $t \in [1, T]$:

$$\begin{aligned} \mathrm{CE}(\mathrm{VALS}_{\alpha_+,\alpha_-}(\hat{p}^t) \mid p_\theta^t) &= \frac{1}{N}\sum_{i=1}^{N} \tilde{y}_i^t \log p_i^t + (1-\tilde{y}_i^t)\log(1-p_i^t). \\ &= \frac{1}{N}\sum_{i=1}^{N} \ell(\tilde{y}_i^t, p_i^t). \end{aligned} \tag{4.9}$$

where:

$$\tilde{y}_i^t = y_i^t(1-\alpha_+) + (1-y_i^t)\alpha_- \tag{4.10}$$

Therefore, similarly to what happened in the vanilla version of multi-task label smoothing, the implementation of the seemingly complex asymetric label smoothing simply consists in replacing y_i^t by $\tilde{y}_i^t$.

4.3.3 Robin Hood Label Smoothing (RHLS)

VALS applies the same two smoothing coefficients to all AU. Therefore, it addresses AU imbalance from a global perspective. However, figure 4.2 shows that different AU may display different imbalance level. To further adapt to those variations, we propose the Robin Hood Label Smoothing (RHLS). RHLS smoothes only the majority class with a coefficient that is based on its frequency. As a consequence, it introduces a probability to take examples from the majority (the rich) class to give it to the minority class (the poor) which justify its name. Formally, it introduces task-wise smoothing probabilities $(\alpha_+^t)_{t=1}^T$ and $(\alpha_-^t)_{t=1}^T$ for positive and negative examples of each task respectively. Then it simply tweaks the smoothing intensities α_+^t, α_-^t using the empirical frequency f^t of task t empirical frequency to ensure that only the majority class get smoothed:

$$\alpha_-^t = \beta \max(0, \frac{1-2f^t}{1-f^t}), \alpha_+^t = \beta \max(0, \frac{2f^t-1}{f^t}), \tag{4.11}$$

where $\beta \in [0, 1]$ quantifies the amount of noise introduced in the majority class from 0 (No noise is applied) to 1 (noise is applied so that the resulting dataset is balanced). Similarly to VALS, RHLS modifies the label of example i for task t as follows:

$$\tilde{y}_i^t = y_i^t(1-\alpha_+^t) + (1-y_i^t)\alpha_-^t \tag{4.12}$$

In the next section, we perform a comparative study between VALS and RHLS showing their superiority over vanilla label smoothing in imbalanced situations, and discussing the situations in which each of them is more adapted.

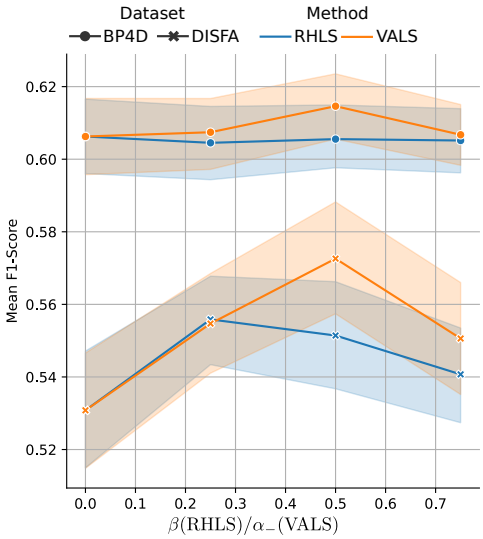


Figure 4.3: Label Smoothing methods validation on Action Unit Detection with 68% confidence interval.

4.4 Experiments

4.4.1 Implementation Details

For all our experiments, we use the same baseline architecture as in chapter 3.5, *i.e.* a Resnet50 convolutional encoder with a transformer based decoder. To validate the hyper parameters of our methods on BP4D (resp. DISFA) we perform 5 folds (resp. 6 folds) cross-validation on each of the 3 usual cross-evaluation folds. The validation score for each set of hyper-parameters therefore consists in a mean F1-score over 15 runs (resp. 18). For the grid of hyper parameters, we consider $\beta = [0.25, 0.5, 0.75]$ for RHLS. For VALS, as most AU are imbalanced toward negativeness, we consider $\alpha_+ = 0$ and $\alpha_- = [0.25, 0.5, 0.75]$.

4.4.2 Ablation Study

In this section we validate the hyper parameters of our smoothing methods and compare their performance to baseline methods for noise mitigation (label smoothing [69]) and imbalance (frequency weighted cross-entropy [59, 61, 23]) on BP4D and DISFA.

Figure 4.3 shows the evolution of the validation scores for RHLS on both DISFA and BP4D. On BP4D, RHLS does not seem to strikingly impact validation scores.

Method/Dataset	DISFA	BP4D
Baseline	63.0 ± 1.9	62.0 ± 1.0
Vanilla Label Smoothing($\alpha = 0.1$)	62.0 ± 1.8	61.9 ± 0.7
Frequency Weighted BCE	61.7 ± 2.1	62.6 ± 0.8
VALS	64.6 ± 0.7	**63.2 ± 0.3**
RHLS	**65.8 ± 1.4**	62.2 ± 1.1

Table 4.1: Comparaison between RHLS/VALS and prior label smoothing methods for noise and imbalance mitigation on DISFA and BP4D.

However on DISFA, it is relatively clear that low values of β significantly boosts the model predictive performance by reducing overconfidence in the majority class and consequently lower the negative influence of both imbalance and noisy majority class examples. However, passed a certain threshold for β, RHLS introduces too much noise to the majority class. This hurts the learning process and results in performance drops. For evaluation, we select $\beta = 0.25$ on both DISFA and BP4D.

Similarly, Figure 4.3 also shows the evolution of the validation scores for VALS on both DISFA and BP4D. Compared to RHLS, VALS seems to boost predictive performance on both AU detection datasets: for low values of α_-, VALS slightly reduces confidence in negative examples which alleviate the noise on those examples and consequently improve performance. Yet, for higher values of α_-, VALS introduces too much false positives which in turn degrades performance. Finally for evaluation, we select $\alpha_- = 0.5$ as it displays optimal validation scores and outperforms RHLS on both DISFA and BP4D.

Table 4.1 compares the performance of the proposed label smoothing strategies with other existing imbalance and noise modelling methods. Frequency weighted BCE displays varying performance on BP4D and DISFA. More precisely, it improves performance on BP4D while causing a significant drop on DISFA. This may be caused by the difference of scales across AU frequencies in DISFA (see figure 4.2, eg: $f_{\text{AU1}} \sim 1e-2$, while $f_{\text{AU25}} \sim 1e-1$). In fact, weighting AU loss contribution using $w_i = \frac{1/f_i}{\sum_{j=1}^{T} \frac{1}{f_j}}$ may encourage the learning of extremely low frequency AU at the expense of all the others. On the other side, on BP4D, frequencies are closer to one another so that weights have roughly the same scale, which explain that frequency weighted binary crossentropy slightly improves performance.

Contrary to what we expected from the validation scores in figure 4.3, we observe in table 4.1 that VALS only outperforms RHLS on BP4D. This shows that correctly validating parameters is difficult on BP4D and DISFA. Indeed, even with a costly cross-validation strategy with a fairly large number of folds,

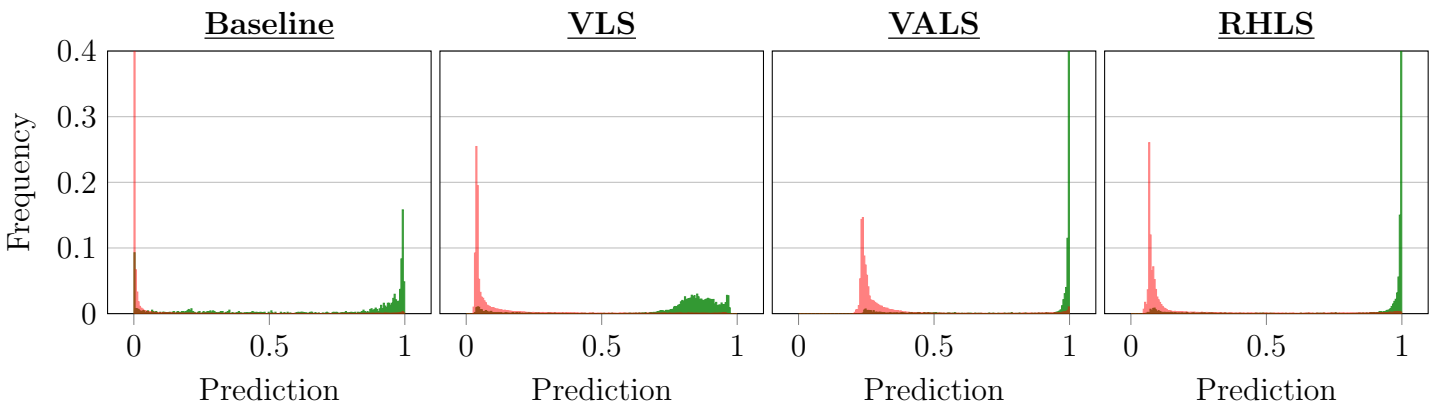


Figure 4.4: Histogram of network predictions for AU4 with different smoothing methods. Positive ground truth examples are represent in green while negatives are in red.

the tendencies in validation are not fully coherent with the observed evaluation scores.

To further explain the difference of performance between RHLS and VALS on DISFA, we argue that RHLS is more adapted to DISFA large AU frequency variations. Indeed, VALS apply the same smoothing coefficient to all negative examples and is consequently unable to adapt to the different scales of imbalance that are displayed on DISFA. On the contrary, by adapting the smoothing coefficient to empirical AU frequencies, RHLS is able to apply a more adapted smoothing coefficient to each AU and consequently enhances performance.

Conversely, on BP4D, we observe the inverse tendency, *i.e* that VALS performs better than RHLS. In fact, a possible explanation is that VALS assumption is more adapted than the one of RHLS to BP4D. Typically, there might be roughly the same proportion of noisy negative examples for each AU regardless of their empirical frequency. This could indeed explain that applying the same smoothing to all negative examples display better performance than scaling smoothing coefficient using frequencies.

Finally, figure 4.4 shows the histogram of predictions for different smoothing methods on DISFA for AU1, 2, 4, 6. We observe that baseline results display majority class overconfidence. Indeed, for all the considered AU, we observe that many positive examples are predicted negative with probability 0. This corresponds to the peak of the green histograms in 0. For all displayed AU, label smoothing mitigates that problem. However, it worsens the imbalance-based minority class low confidence problem. As a matter of fact, the vanilla label smooth-

Mean F1 Score	BP4D	DISFA
JAANet [59]	60.0	56.0
ARL [61]	61.1	58.7
JÂANET [60]	62.4	63.5
UGN-B [63]	63.3	60.0
FAUwT [23]	64.2	61.5
MONET [73]	**64.5**	63.9
Baseline	61.9	63.1
VALS	63.2	64.6
RHLS	62.2	**65.8**

Table 4.2: Comparison of VALS and RHLS with state-of-the-art deep learning based AU detection methods

ing green histograms are significantly more spread around 1 than the baseline histograms. This in turn reduce overall performance (see Table 4.1). By smoothing only the majority class, both VALS and RHLS mitigates majority class confidence without any influence on the minority class and consequently obtains the performance boost reported in Table 4.1.

4.4.3 Comparaison with state-of-the-art methods

In this section we compare RHLS and VALS with state-of-the-art AU detection methods.

Table 4.2 provides results for VALS and RHLS on BP4D. Interestingly, applying VALS over a modern multi-task baseline is competitive with several recent methods [61, 60], including other noise modelling strategies [63]. However, it gets outperformed by the most recent ones that either involve more complex landmark-guided transformer architecture [23] or refined AU dependency modelling [73, 64]. Nonetheless, the increment VALS provides to the baseline shows that it is a simple yet efficient way to improve performance without any additional computational overhead. In that extent, attempting to plug it on top of more complex methods could be a promising track toward better overall AU detection performance.

On DISFA, Table 4.2 shows that both VALS and RHLS allows the baseline architecture to surpass state-of-the-art performance. To explain those excellent results, it is worth noticing that most state-of-the-art methods [59, 61, 73, 23] use frequency-weighted losses. Therefore the superiority of RHLS and VALS against the frequency-weighted loss on DISFA (see Table 4.1) may justify the significant increment that we observe.

Beyond that, it is also worth noticing that the simple and free label modifications of VALS and RHLS pushes a simple baseline above more complex methods with spatial prior guidance [60] or explicit AU dependency modelling [73]. On the one hand, it highlights that imbalance reduction and noise modelling are as critical as input feature extraction or dependency modelling for AU detection performance. On the other hand, it offers potential improvement perspectives as integrating our label smoothing extensions with more complex methods could further improve performance.

4.5 Conclusion and future works

4.5.1 Discussion

In this chapter, we investigated the impact of label smoothing to fight against AU datasets imbalance and noise problems. In particular, we showed that vanilla label smoothing is ill-adapted to imbalanced situations as it may worsen pre-existing under-confidence problems and degrade performance. To alleviate this issue, we proposed Vanilla Asymetric Label smoothing (VALS) that applies different smoothing based on weither a label is positive or negative. Furthermore, we introduced Robin Hood Label Smoothing (RHLS) that constrains label smoothing and scales the smoothing coefficient based on each AU empirical frequency.

Experimentally, we showed that those two label smoothing extensions displayed promising performance on both BP4D and DISFA. Indeed, on BP4D, VALS provide a simple and free increment over a modern multi-task baseline results and outperforms the widely adopted frequency weighted loss. Furthermore on DISFA, the increment provided by RHLS allow the multi-task baseline to outperform state-of-the-art results (and *a fortiori* the frequency-weighted loss). Interestingly, those results indicate using RHLS or VALS provides better results than the widely used frequency-weighted binary cross-entropy, while keeping similar computational cost. As a consequence, we believe that adopting our label smoothing variants instead of the frequency weighted binary-crossentropy could pave the way toward improving state-of-the-art results in AU detection.

4.5.2 Future Works

In future works we would like to further dig into the intricate relationship between imbalance and noise in AU detection. Indeed, the current experimental results are not fully informative on whether the observed increments come from the imbalance problem reduction or from the noise modelling part of our label

smoothing extensions. For that purpose, an idea would simply be to inspire from frequency-weighted binary crossentropy to design a loss that, instead of setting positive examples to random with probability α simply ignores them with the same probability. Such loss would only tackle the imbalance problem. Therefore, comparing our label smoothing extensions with it will allow us to measure the relative importance of rebalancing and noise mitigation.

Another track that we would like to explore consists in structuring the noise that label smoothing introduces. Indeed, in this chapter we worked under the assumption that the label smoothing noise is applied to each task independently. However, such assumption may virtually introduce unrealistic examples that break AU dependencies. For example, if AU1 (inner brow raiser) and AU4 (brow lowerer) are simultaneously affected by the label smoothing noise, this might result in activation of both AU which is impossible in practice. As a consequence, we believe that taking AU dependencies into account in the way labels are smoothed could improve performance.

Finally, we would like to further explore rebalancing strategies for AU datasets. Indeed, with the current fold distribution [61], AU labels are not balanced between folds. As a consequence, part of a model performance shows how well the bias of the model manages to get close to the bias of the test set. This concern is rather crucial as it deflects the performance measure from its primary purpose that is to quantify how well a model manages to predict the correct AU of a given face. The reason for this imbalance is that multi-label problems are extremly difficult to properly balance because of the dependency between the different labels.

A first, very naive approach would simply be to cast the T binary labels into a single categorical classification label with 2^T classes as follows :

$$y_{\text{cat}} = \sum_{t=1}^{T} y^t 2^t \tag{4.13}$$

Then one could create balanced fold by providing each fold with the same number of examples from each class. However, the main obstacles coming in the way of using such natural rebalancing trick is (a) that according to the litterature, folds have to display mutually exclusive identity and (b) that the discrete distribution of categorical labels resulting from 4.13 may display several empty or under-populated classes (c) that those empty and under-populated classes may significantly vary across identities. The latter point significantly complicates the rebalancing process, and even makes it impossible for example if a set of AU activations is only present for a single identity (or for less identity than there are balanced folds to create).

To try to circumvent such issue, the idea we would like to pursue consists in partitioning Action Units in n_B blocks, deriving a categorical label by block and balancing each of the n_B resulting categorical problem independently. The incentive behind that is that the block-wise categorical labels will have less bins than the full categorical problems and are therefore likely to have less empty bins which in turn make the rebalancing task easier. However, finding the correct partition (intuitively the partition in which blockwise labels are as independant as impossible) is non-trivial to do and still requires further exploration.

Chapter 5

Conclusion

5.1 Discussion

The aim of this thesis was to design efficient deep learning-based methods for Action Unit detection. For that purpose, we identified data scarcity and label noise as the two main challenges that hinders the development of AU detection.

Indeed, the publicly available AU datasets only contain annotations for few different faces. Furthermore, they are imbalanced, meaning that, for each face, only few frames are annotated with AU activation. Even worst, as AUs are subtle face movements those annotations may be wrong. Altogether, a deep neural network trained on those datasets is likely to display poor generalisation performance.

In order to address this overfitting problem, we adopted three different strategies. First, we attempted to capture the Action Unit inter-dependencies in order to better structure our network predictions. For that purpose, we proposed MONET, a network that uses a soft order selection mechanism to jointly learn multiple tasks along with the order in which they should be predicted. We further refined the order selection mechanism with two strategies that enhance the exploration of the order space, namely order warmup and order dropout. Experimentally, we first showed that MONET was able to retrieve the best order in a controlled environment. Then, we provided evidence that MONET was more efficient than baseline multi-task methods on various CelebA subsets. Last but not least, we demonstrated than MONET significantly outperforms state-of-the-art performance in AU detection on both BP4D and DISFA.

Secondly, we tried to exploit the local aspect of Action Units. More precisely, we proposed to tackle transformers' well known overfitting problems by adapting recent attention guiding method to those architectures. For that purpose, we followed previous work on AU detection with transformers and designed an archi-

tecture that is an hybrid between transformers and convolutional networks. Then, we investigated multiple strategies to constrain its multi-head attention mechanism. The main conclusion of our investigations is that the different heads of a transformer benefit from having to pay attention to different priors. Indeed, we showed that forcing different heads to pay attention to different landmarks-based priors improves predictive performance on BP4D. We further validated that claim by showing that guiding different heads with different face parsing priors provide increments on both DISFA and BP4D.

The last line of research that we explored is the modelling of Action Units annotation noise. To do so, we have drawn inspiration from the recent success of label smoothing [69] at mitigating noise [39]. However, vanilla label smoothing reduces the confidence of a network on both negatively and positively annotated examples. Therefore, we showed that applying it as it is to the AU detection problem worsen pre-existing under-confidence in the minority class, thus degrading performance. To circumvent that problem, we proposed a Vanilla Asymetric Label Smoothing (VALS) that apply separate smoothing to positive and negative examples. This method enables us to smooth the majority class examples only, consequently avoiding worsening the minority class under-confidence problem. We further pushed this idea by proposing a Robin Hood Label Smoothing (RHLS) that smoothes only the majority class with smoothing coefficients that scales based on AU empirical frequencies (the higher the frequency the larger the smoothing coefficient). We validate the ability of those two methods to address both imbalance and noise by (a) showing that VALS improve performance of a modern multi-task baseline on BP4D and outperform state-of-the-art on DISFA. Furthermore, we showed that RHLS frequency-based scaling is particularly adapted to DISFA large AU frequencies variations. Indeed RHLS outperforms both VALS and *a fortiori* state-of-the-art result on DISFA.

5.2 Future Work

5.2.1 DYSFER project

As mentionned in the introduction 1, the original purpose of our AU detectors was to help identify dispnea in hospital environment. However, the COVID-19 pandemic significantly slowed down the data acquisition process. Indeed, it only started in the last months of 2022. Meanwhile, we were provided with a dataset that featured patients doing apnea exercises. On this dataset, our AU detectors displayed promising results. Yet, it is hardly representative of its performance on real bedridden patients.

A development project has been launched, and an intern will be recruited to study the performance of our AU detection methods on real-life data. This study will be complemented with further investigations to optimize our methods performance for more efficient deployment in hospital environment. In particular, we would like to design transfer learning methods to adapt our methods to hospitalized patient faces that may not be properly aligned and/or obstructed with invasive medical devices (eg. an artificial respirator).

5.2.2 Better pretraining for AU detection

Furthermore, we would like to elaborate on the observation we made about our baseline transformer pretraining in the preliminary experiments 3.4.3. Indeed, we showed that, contrary to what we expected, additional pretraining on VGGFACE2 [4] (which is a 3M face images dataset) didn't provide a significative improvement w.r.t the ImageNet pretraining. We believe that such disappointing performance comes from the fact that VGGFACE2 pretraining task is identity recognition. The problem with identity recognition is that it is ideally invariant to facial expressions. In other words, the perfect identity classifier should output the same prediction regardless of a person's facial expression. As a consequence, the features extracted from a pretrained VGGFACE backbone are likely to discard all facial expressions information which may explain the poor performance improvement observed for AU detection. This claim is endorsed by recent AU detection performance improvement [40] obtained by using unsupervised learning on wide face databases.

Therefore, we believe that investigating new unsupervised learning methods could be a promising line of research. More precisely, the work in [19, 6] that extracts knowledge in an unsupervised manner by learning invariance to well-chosen data-augmentations seem both interesting and accessible. Indeed, it seems to suffer less from reduced batchsize than previous works [8, 21]. However, it is known to require adapted augmentation schemes [19] that have not been yet investigated in the case of facial expression detection. It will consequently require further explorations.

5.2.3 Quantization for AU detection

Finally, in this thesis, we mostly investigated AU-specific methods to fight overfitting. However, we recently worked in collaboration with Edouard Yvinec to perform a more general study of overfitting reduction methods for AU detection [71]. For that purpose, we first tested the performance of classical methods such

as dropout, weight decay and architecture shrinking. Additionally, we drew inspiration from recent compression methods [93, 92, 94, 91, 95, 96, 97] and proposed to use quantization as a new way to fight overfitting. The incentive is that learning with discretized weights and activations limits a model representation power, prevents it from fitting the training data too closely and consequently reduces overfitting risks.

Experimentally, we validated this claim by showing that quantization outperforms over regularization techniques on BP4D. This result is all the more interesting as, beside limiting overfitting, quantization also reduces computational costs in inference. Therefore, quantized models are more efficient in both training and inference phases.

In the light of those results, we believe that further exploration on model quantization could constitute a promising line of research toward better overfitting reduction methods and consequently better AU detection. More precisely, we would like to further adapt quantization techniques to the multi-task aspect of AU detection. For example, a naive approach would be to quantize the different tasks prediction branches to account for the different overfitting levels of each task. Intuitively, the quicker a task overfits, the more quantized the branch that predicts it should be. However such method requires to quantify how much each task is overfitting which is difficult to do in practice. It therefore needs further exploration.

Bibliography

[1] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017. 61

[2] Hakan Bilen and Andrea Vedaldi. Integrated perception with recurrent multi-task neural networks. In *Advances in neural information processing systems*, pages 235–243, 2016. 19

[3] Jules Bonnard, Arnaud Dapogny, Ferdinand Dhombres, and Kevin Bailly. Privileged attribution constrained deep networks for facial expression recognition. In *ICPR*. IEEE, 2022. 11, 59, 60, 61, 64, 75, 76, 77, 78, 85

[4] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE, 2018. 42, 68, 109

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020. 59, 62

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021. 64, 76, 80, 85, 109

[7] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance

samples. *Advances in Neural Information Processing Systems*, 30, 2017. 92, 94

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 109

[9] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018. 18

[10] Ciprian Corneanu, Meysam Madadi, and Sergio Escalera. Deep structure inference network for facial action unit recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 298–313, 2018. 17, 50, 51

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 63

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 62

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 11, 59, 60, 62, 64, 66, 67

[14] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pages 2286–2296. PMLR, 2021. 11, 62, 63, 64, 80, 81, 85

[15] Paul Ekman et al. Basic emotions. *Handbook of cognition and emotion*, 98(45-60):16, 1999. 7

[16] Paul Ekman and Wallace V Friesen. Facial action coding system. *Environmental Psychology & Nonverbal Behavior*, 1978. 7

[17] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3205–3214, 2019. 18

[18] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *International conference on learning representations*, 2017. 92, 93, 94

[19] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. 109

[20] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems*, 31, 2018. 92, 93

[21] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 109

[22] Ronghang Hu and Amanpreet Singh. Unit: Multimodal multitask learning with a unified transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1439–1449, 2021. 86

[23] Geethu Miriam Jacob and Bjorn Stenger. Facial action unit detection with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7680–7689, 2021. 8, 11, 46, 50, 51, 59, 60, 63, 68, 91, 92, 99, 102

[24] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International conference on machine learning*, pages 2304–2313. PMLR, 2018. 92, 94

[25] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018. 18

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 37

[27] Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6129–6138, 2017. 15, 18

[28] Guanbin Li, Xin Zhu, Yirui Zeng, Qing Wang, and Liang Lin. Semantic relationships guided representation learning for facial action unit recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8594–8601, 2019. 17, 50, 51, 91, 92

[29] Wei Li, Farnaz Abtahi, Zhigang Zhu, and Lijun Yin. Eac-net: Deep nets with enhancing and cropping for facial action unit detection. *IEEE transactions on pattern analysis and machine intelligence*, 40(11):2583–2596, 2018. 11, 50, 51, 59, 60, 61, 74, 79

[30] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 46, 97

[31] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 18

[32] Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *International Conference on Learning Representations*, 2021. 18

[33] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2873–2879. IJCAI/AAAI Press, 2016. 15

[34] Yahui Liu, Enver Sangineto, Wei Bi, Nicu Sebe, Bruno Lepri, and Marco Nadai. Efficient training of visual transformers with small datasets. *Advances in Neural Information Processing Systems*, 34:23818–23830, 2021. 63

[35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 63

[36] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 62

[37] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 42, 70

[38] Michal Lukasik, Srinadh Bhojanapalli, Aditya Menon, and Sanjiv Kumar. Does label smoothing mitigate label noise? In *ICML*. PMLR, 2020. 12, 89, 94, 95

[39] Tohar Lukov, Na Zhao, Gim Hee Lee, and Ser-Nam Lim. Teaching with soft label smoothing for mitigating noisy labels in facial expressions. In *ECCV*, pages 648–665. Springer, 2022. 94, 108

[40] Bowen Ma, Rudong An, Wei Zhang, Yu Ding, Zeng Zhao, Rongsheng Zhang, Tangjie Lv, Changjie Fan, and Zhipeng Hu. Facial action unit detection and intensity estimation from self-supervised representation, 2022. 109

[41] Eran Malach and Shai Shalev-Shwartz. Decoupling" when to update" from" how to update". *Advances in neural information processing systems*, 30, 2017. 92

[42] Brais Martinez, Michel F Valstar, Bihan Jiang, and Maja Pantic. Automatic analysis of facial actions: A survey. *IEEE transactions on affective computing*, 10(3):325–347, 2017. 7

[43] Elliot Meyerson and Risto Miikkulainen. Beyond shared hierarchies: Deep multitask learning through soft layer ordering. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 18, 19

[44] Kevin Miao, Akash Gokul, Raghav Singh, Suzanne Petryk, Joseph Gonzalez, Kurt Keutzer, Trevor Darrell, and Colorado Reed. Prior knowledge-guided attention in self-supervised vision transformers, 2022. 63

[45] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. Ieee, 2016. 91

[46] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016. 18, 42, 86, 87

[47] Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J Kim, and Johannes Fürnkranz. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *Advances in neural information processing systems*, pages 5413–5423, 2017. 15, 19

[48] Xuesong Niu, Hu Han, Shiguang Shan, and Xilin Chen. Multi-label co-regularization for semi-supervised facial action unit recognition. *Advances in neural information processing systems*, 32, 2019. 17

[49] Xuesong Niu, Hu Han, Songfan Yang, Yan Huang, and Shiguang Shan. Local relationship learning with person-specific shape regularization for facial action unit detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11917–11926, 2019. 50, 51, 61

[50] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1944–1952, 2017. 92, 93, 94

[51] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 62

[52] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 63

[53] Rajeev Ranjan, Vishal M Patel, and Rama Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE transactions on pattern analysis and machine intelligence*, 41(1):121–135, 2017. 15, 18

[54] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014. 92, 93

[55] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018. 92, 94

[56] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017. 18, 19

[57] Nishant Sankaran, Deen Dayal Mohan, Srirangaraj Setlur, Venugopal Govindaraju, and Dennis Fedorishin. Representation learning through cross-modality supervision. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, pages 1–8. IEEE, 2019. 50, 51

[58] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pages 527–538, 2018. 18

[59] Zhiwen Shao, Zhilei Liu, Jianfei Cai, and Lizhuang Ma. Deep adaptive attention for joint facial action unit detection and face alignment. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 705–720, 2018. 11, 45, 50, 51, 59, 60, 61, 64, 74, 99, 102

[60] Zhiwen Shao, Zhilei Liu, Jianfei Cai, and Lizhuang Ma. Jaa-net: Joint facial action unit detection and face alignment via adaptive attention. *International Journal of Computer Vision*, 129(2):321–340, 2021. 8, 11, 45, 46, 50, 51, 59, 60, 61, 64, 74, 79, 91, 102, 103

[61] Zhiwen Shao, Zhilei Liu, Jianfei Cai, Yunsheng Wu, and Lizhuang Ma. Facial action unit detection using attention and relation learning. *IEEE Transactions on Affective Computing*, 2019. 50, 51, 61, 64, 79, 91, 99, 102, 104

[62] Hwanjun Song, Minseok Kim, and Jae-Gil Lee. Selfie: Refurbishing unclean samples for robust deep learning. In *International Conference on Machine Learning*, pages 5907–5915. PMLR, 2019. 93, 94

[63] Tengfei Song, Lisha Chen, Wenming Zheng, and Qiang Ji. Uncertain graph neural networks for facial action unit detection. In *AAAI*, 2021. 8, 92, 102

[64] Tengfei Song, Zijun Cui, Wenming Zheng, and Qiang Ji. Hybrid message passing with performance-driven structures for facial action unit detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6267–6276, 2021. 8, 17, 50, 51, 102

[65] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 32, 33

[66] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021. 59, 62

[67] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014. 92, 93, 94

[68] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33, 2020. 18

[69] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 12, 89, 94, 95, 96, 99, 108

[70] Gauthier Tallec, Jules Bonnard, Arnaud Dapogny, and Kévin Bailly. Multi-task transformer with uncertainty modelling for face based affective computing. *arXiv preprint arXiv:2208.03506*, 2022. 12

[71] Gauthier Tallec, Arnaud Dapogny, and Kevin Bailly. Fighting noise and imbalance in action unit detection problems. *arXiv preprint arXiv:2303.02994*, 2023. 12, 109

[72] Gauthier Tallec, Arnaud Dapogny, and Kévin Bailly. Multi-order networks, 2023. Patent EP21306638.4. 11

[73] Gauthier Tallec, Edouard Yvinec, Arnaud Dapogny, and Kevin Bailly. Multi-label transformer for action unit detection. *arXiv preprint arXiv:2203.12531*, 2022. 11, 83, 84, 86, 102, 103

[74] Gauthier Tallec, Edouard Yvinec, Arnaud Dapogny, and Kevin Bailly. Multi-label transformer for action unit detection. *arXiv preprint arXiv:2203.12531*, 2022. 12

[75] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers; distillation through attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 18–24 Jul 2021. 59, 62, 63, 64, 66, 71, 72

[76] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit, 2022. 63, 66

[77] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021. 62, 66, 70

[78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017. 11, 62, 64, 65, 67, 70

[79] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*, 2017. 94

[80] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. 10, 16, 19, 31, 37

[81] Robert Walecki, Vladimir Pavlovic, Björn Schuller, Maja Pantic, et al. Deep structured learning for facial action unit intensity estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3405–3414, 2017. 17

[82] Kai Wang, Xiaojiang Peng, Jianfei Yang, Shijian Lu, and Yu Qiao. Suppressing uncertainties for large-scale facial expression recognition. In *CVPR*, 2020. 93, 94

[83] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22–31, 2021. 63

[84] Zhizheng Wu, Cassia Valentini-Botinhao, Oliver Watts, and Simon King. Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4460–4464. IEEE, 2015. 18

[85] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9981–9990, 2021. 63

[86] Fanglei Xue, Qiangchang Wang, and Guodong Guo. Transfer: Learning relation-aware facial expression representations with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3601–3610, 2021. 11, 59, 63, 64, 68

[87] Huiyuan Yang, Lijun Yin, Yi Zhou, and Jiuxiang Gu. Exploiting semantic embedding and visual feature for facial action unit detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10482–10491, 2021. 50, 51

[88] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020. 18

[89] Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 579–588, 2021. 63

[90] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 558–567, 2021. 63

[91] Edouard Yvinec, Arnaud Dapgony, Matthieu Cord, and Kevin Bailly. Rex: Data-free residual quantization error expansion. *arXiv preprint arXiv:2203.14645*, 2022. 110

[92] Edouard Yvinec, Arnaud Dapogny, and Kevin Bailly. To fold or not to fold: a necessary and sufficient condition on batch-normalization layers folding. *arXiv preprint arXiv:2203.14646*, 2022. 110

[93] Edouard Yvinec, Arnaud Dapogny, Matthieu Cord, and Kevin Bailly. Red: Looking for redundancies for data-freestructured compression of deep neural networks. *Advances in Neural Information Processing Systems*, 34:20863–20873, 2021. 110

[94] Edouard Yvinec, Arnaud Dapogny, Matthieu Cord, and Kevin Bailly. Red++: Data-free pruning of deep neural networks via input splitting and output merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3664–3676, 2022. 110

[95] Edouard Yvinec, Arnaud Dapogny, Matthieu Cord, and Kevin Bailly. Singe: Sparsity via integrated gradients estimation of neuron relevance. *arXiv preprint arXiv:2207.04089*, 2022. 110

[96] Edouard Yvinec, Arnaud Dapogny, Matthieu Cord, and Kevin Bailly. Powerquant: Automorphism search for non-uniform quantization. *arXiv preprint arXiv:2301.09858*, 2023. 110

[97] Edouard Yvinec, Arnaud Dapogny, Matthieu Cord, and Kevin Bailly. Spiq: Data-free per-channel static input quantization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3869–3878, 2023. 110

[98] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017. 92

[99] Xing Zhang, Lijun Yin, Jeffrey F Cohn, Shaun Canavan, Michael Reale, Andy Horowitz, Peng Liu, and Jeffrey M Girard. Bp4d-spontaneous: a high-resolution spontaneous 3d dynamic facial expression database. *IVC*, 2014. 9, 92

[100] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *European conference on computer vision*, pages 94–108. Springer, 2014. 18

[101] Kaili Zhao, Wen-Sheng Chu, and Honggang Zhang. Deep region and multi-label learning for facial action unit detection. In *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition, pages 3391–3399, 2016. 46, 50, 51, 59, 60