Adaptive deep ensemble methods for face analysis in the wild

Estèphe Arnaud

Sorbonne Université

Ecole Doctorale 391:
Sciences Mécaniques, Acoustique, Electronique et Robotique

Adaptive Deep Ensemble Methods For Face Analysis In The Wild

Thèse de doctorat

Estèphe ARNAUD

Date de soutenance : 12/03/2021

Composition du jury :

Directeur de thèse :	Kévin BAILLY	Sorbonne Université, ISIR
Co-encadrant :	Arnaud DAPOGNY	Datakalab
Rapporteurs :	Mohamed DAOUDI Laurent HEUTTE	Université de Lille, CRIStAL Université de Rouen, LITIS
Examinateurs :	Matthieu CORD David PICARD Catherine SOLADIE	Sorbonne Université, LIP6 ENPC, LIGM CentraleSupélec, IETR

Contents

Chapter 1

Introduction

The face contains a lot of semantic information about a person, such as his gender, age, or ethnicity, and can even give a glimpse of his personality or his cognitive state. Several works in psychology [121][15][158] have also highlighted the importance of the face in human communication. For example, it was shown in [121] that facial expression contributes to 55% in the impact of a spoken message, while textual content and speech signals (e.g. word accentuation, punctuation) contribute only to 7% and 38% respectively. During social interaction, humans thus mainly analyse their respective faces to adjust their behaviour accordingly.

For several decades, automatic face analysis has become a very active research topic in computer vision, providing the opportunity (1) to save human resources, (2) to process data rapidly, continuously and on a large scale, and (3) to obtain objective face descriptors. A broad range of applications take advantage of this technology, such as security (*e.g.* to improve identification via biometrics authentification or to deploy surveillance systems on a large scale), robotics (*e.g.* to recognize, interpret, process, and simulate human affects allowing human-machine interaction), healthcare (*e.g.* to improve facial communication or to measure the intensity of apparent suffering), entertainment (*e.g.* to generate avatars for video games or to measure emotional engagement in front of an advertisement).

In order to automate face analysis tasks, most approaches use machine learning techniques for their capacity to automatically extract knowledge from training data and to learn task-specific face patterns. However, current face analysis systems are still struggling to adapt to the immense variety of morphological traits, head poses, or objects that can occlude the face.

Thus, the scientific objective of this thesis is to improve their robustness in unconstrained environments that can involve many variations in the face appearance.

To this end, the choice of the model architecture and the learning algorithm is therefore crucial to ensure the robustness. With an ever-increasing amount of data and computational resources, deep neural networks (DNN) based methods have often led to significant advances for many machine learning problems, including face analysis. However, DNN generalize poorly on out-of-distribution data, thus limiting their capacity to adapt to the most extreme variations not observed in the training data. On the other hand, ensemble methods are a promising approach to address this issue. Indeed, instead of a single strong predictor, they use several base predictors, whose combination leads to reduce the variance in prediction errors and improve the overall robustness. Throughout this thesis, we then propose to merge DNN-based methods and ensemble methods to develop an accurate and robust face analysis system. It takes place in the frame of the FacIL project (supported by the French National Agency - ANR). FacIL is an acronym standing for "Face Interpretation with deep and ensemble Learning".

1.1 Face analysis pipeline

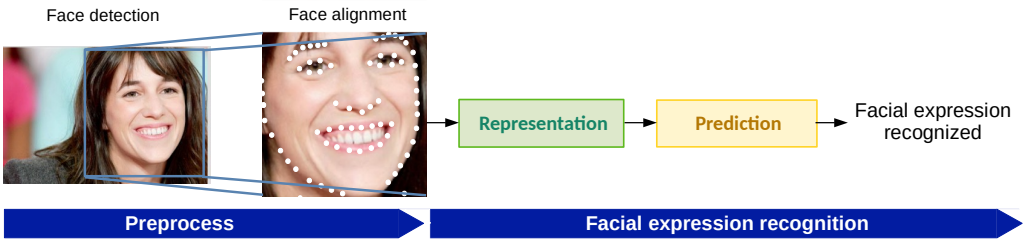


Figure 1.1: A classical face analysis pipeline.

As illustrated in Figure 1.1, a face analysis pipeline contains several elements. Given an image displaying somebody, a preprocess is first performed by detecting the face in order to keep only the global face appearance. In addition, face alignment can be conducted to refine the face localization and to obtain information about the face geometry as well as shape-indexed local appearances that can help to better learn task-specific face patterns. Then, the face appearance and shape information are used to extract a face representation upon which a predictor can be trained from training data. In particular, in this thesis, we perform facial expression recognition.

1.1.1 Preprocess

Face detection. Face detection is a specific case of object detection, focusing on the presence and location of one or more human faces in an image. As illustrated in Figure 1.1 (left), the objective is to output the smaller rectangular bounding box encompassing a face. The image is then cropped from the detected face bounding box to remove non-face related information as much as possible and keep only the face appearance.

Face alignment. Face alignment consists in locating facial landmarks (eyebrows, eyes, nose, mouth, jaw) that form the face shape. This is an active research topic, whose modeling techniques may have similarities with those used for human pose estimation [112]. It allows to identify the geometric structure of the face and to determine the shape of face components (e.g. smile, open/closed eyes). On the other hand, the shape-indexed local appearances allows to detect highly-discriminative local texture patterns that

cannot be identified by the face shape alone (e.g. dimples in the mouth corner). In addition to providing knowledge about the face structure, face alignment also enables to better crop the face image by using the smaller bounding box containing all the facial landmarks.

1.1.2 Facial expression recognition

From the texture and geometric information of the face, it is then possible to model a multitude of higher-level face analysis tasks, such as face recognition [83] or face reconstruction [14]. In this thesis, we address in particular the issue of facial expression recognition, which can be used to estimate the cognitive state about a person. Three main approaches allow to recognize facial expression (FE): (1) a categorical model, (2) a dimensional model, or (3) the facial action coding system.

Categorical model. A first approach is to classify FEs into several categories. Ekman et al. [47] have retained six prototypical FEs that have been validated through an inter-cultural consensus: happiness, surprise, sadness, anger, fear, and disgust. To this list can be added the neutral FE. This categorical approach is currently the most popular to model FEs due to the simplicity of image annotation. Indeed, a small number of categories makes it easier to obtain an inter-cultural consensus, thus reducing the risk of annotation inconsistencies between different human labellers. In addition, it is the simplest strategy for annotating very large datasets [123]. Finally, it can be noticed that the FE probabilities can provide additional nuances, by describing a FE as a mixture of several basic FEs, as illustrated in Figure 1.2 (left).

Dimensional model. Facial expression can also be described by two dimensions [139]: (1) the valence, which measures its attractiveness from pleasant (positive valence) to unpleasant (negative valence), and (2) the arousal, which measures its intensity from excited (positive arousal) to calm (negative arousal). For example, a scream associated with fear has negative valence and positive arousal. Two other dimensions can be added [51]: the control and expectancy, which respectively measures the feeling of dominance and unpredictability. However, most dimensional approaches use only valence and arousal. Figure 1.2 (middle) illustrates different possible projections of

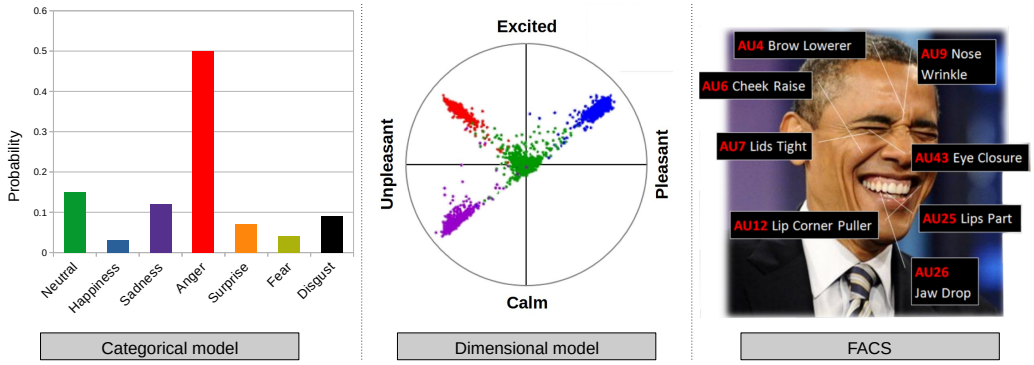


Figure 1.2: Three main approaches to recognize facial expression (FE). **Categorical model**: classification of the 7 basic FEs. **Dimensional model**: possible 2d-projections of basic FEs in terms of valence (from unpleasant to pleasant) and arousal (from calm to excited). Excerpt from [85]. **Facial action coding system**: each FE can be described by a combination of action units, i.e. activation of one of the 44 facial muscles. Excerpt from [118].

categorical FEs on the two-dimensional valence/arousal space. Contrary to the categorical approach, it is difficult to establish a consensus to annotate each face image. It requires expert human labellers to ensure annotation quality, thus limiting the availability of annotated data.

Facial action coding system. As the facial expression is linked with the contraction of particular facial muscles, other approaches have instead implemented a coding system mapping each FE towards a combination of contracted facial muscles. The most used coding system is the Facial Action Coding System (FACS) proposed by Ekman et al. [48]. The FACS manual divides the face into 44 facial muscles. The contraction of a specific facial muscle is called an action unit (AU). Anatomical studies have then led to describe each basic FE by a specific combination of AUs, which can be used for example to better capture micro-expressions [98]. Figure 1.2 (right) illustrates the most common AUs from upper and lower face parts. For example, happiness is typically associated with the smile described by lip corner puller (AU12) and with cheek raise (AU6). Unfortunately, few annotated data are available to learn AU detection. Indeed, this requires human experts in FACS labeling, whose specific skills take time to assimilate.

1.2 Factors of variation in the data

The face analysis tasks presented above are well modeled in lab-controlled conditions: each face is frontal and well exhibited, allowing to capture a maximum of semantic information during training and to obtain a good accuracy during testing under similar conditions. However, in real conditions, the model performance can be strongly degraded due to factors *exogenous* to the given task, acting as noise. Indeed, a great variety of head poses or objects (e.g. glasses, mask, hat, and so on) can affect the face visibility and corrupt the input features fed to the model. In addition, identity-related information (e.g. morphological traits, age, ethnicity) can also be an important source of variations affecting the face appearance, leading to deteriorate the model performance. Generally speaking, three factors of variation are particularly challenging today to obtain a robust face analysis system: identity, head pose, and partial occlusions. The objective of this thesis is to improve the robustness of face analysis systems to each of these factors.

1.2.1 Identity variations

Figure 1.3: Identity variations. Images from VGGFace2 [23].

A face can be identified by several attributes, such as gender, age, ethnicity, and specific morphological traits. There is an immense variability of such identity-related information, which is then one of the main sources of variation in face images. Given a face analysis task, it is then difficult to

learn a model that can adapt to all possible variations in identity. For instance in facial expression recognition (FER), a large intra-class variance is often observed and can be attributed to the strong variations in identity. As illustrated in Figure 1.3, we can see that for each FE class (top to bottom: anger, fear, happiness), identity-related information has a strong impact in the face appearance, which can lead to corrupt the input features fed to the FER model and degrade its performance.

1.2.2 Head pose variations

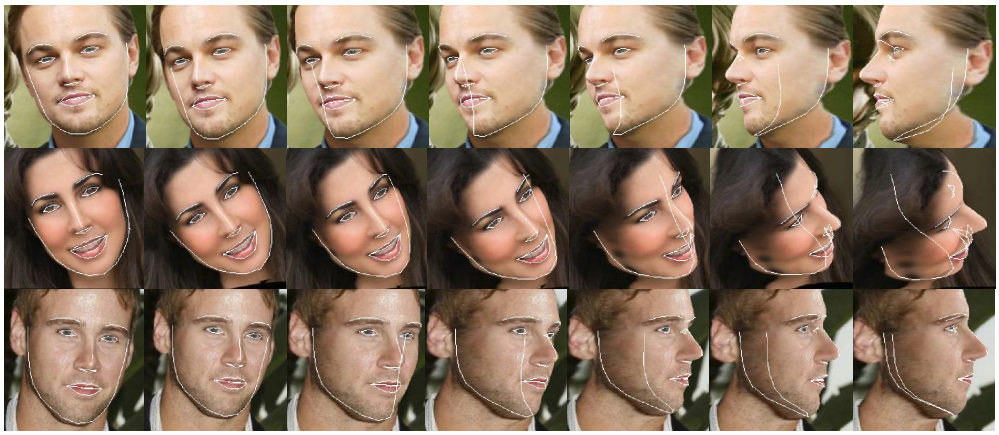


Figure 1.4: Head pose variations. Images from 300W-LP [199].

The face appearance can also be greatly influenced by head pose with report to the camera. The head pose is defined by three Euler angles around each of the 3D-axes. These angles are called yaw (rotation around the y-axis), pitch (rotation around the x-axis) and roll (rotation around the z-axis). Figure 1.4 shows the impact of yaw intensities on the face appearance of three different subjects. We can then observe that head pose variations rapidly modify the face appearance, which can degrade at the same time the performance of a face analysis model. In particular, the most extreme orientations lead to hide face parts that can be highly discriminative depending on the task. For example in face alignment, a profile face (i.e. associated with extreme yaw) hides almost half of the face, as illustrated in Figure 1.4 (right). The model should then be able to estimate the facial landmarks coordinates in this non-visible part without having access to their surrounding texture.

1.2.3 Partial occlusions

Figure 1.5: Partial occlusion variations. Images from COFW [20].

Partial occlusion is another important factor of variations in the face images. Indeed, a great variability of face parts can be hidden by an infinity of possible objects with various textures and orientations. Figure 1.5 illustrates the diversity of possible objects that can occlude a face, such as a hat, hair, glasses, mask, bubbles, corn, and so on. In addition, as we have seen above, certain face parts can also be self-occluded by extreme head pose. Any model cannot capture all possible occlusions. Its architecture and/or its learning algorithm should then be able to learn to adapt to this great diversity of appearances, otherwise its performance is severely affected. The robustness to occlusions, in addition to identity and head pose variations, thus remains a challenge for current face analysis systems.

1.3 Contributions

In order to increase the overall robustness in unconstrained environments, we propose to merge three approaches commonly used in the literature: deep neural networks-based methods, ensemble methods, and adaptive methods that explicitly integrate a specific factor of variation into the model to better adapt it to the most extreme variations related to this factor.

Given a target task (e.g. face alignment or facial expression recognition), and a model layer (e.g. representation or prediction layer), we use an ensemble of base networks, whose decisions are adaptively weighted by a so-called gating variable, identified as an important source of variations exogenous to the task (e.g. head pose). Thus, these base networks are each specialized on a region of the gating space (e.g. left-oriented faces), so that the model can select the most relevant base networks depending on the input, even under the most extreme conditions.

To sum it up, the main contributions of our approach are three-folds:

- From an architectural standpoint, we propose a new adaptive deep ensemble architecture using (1) an efficient gating structure, allowing to jointly learn the base networks and a hierarchical partition of the gating space upon which specialize them, and (2) a suitable exogenous gating variable to better adapt the base networks to the most extreme conditions.

- From a learning standpoint, we propose a new training loss encouraging to remove the exogenous information from the endogenous representation deciphering the target task, further improving the overall learning algorithm and the robustness of base networks to exogenous variations.

- From an experimental standpoint, we propose generic methods that can be applied to multiple layers (e.g. prediction, representation layer) and multiple face analysis tasks (e.g. face alignment, facial expression recognition), and even more generally to any predictive task where an important source of exogenous variations can be identified (e.g. digit recognition with rotation variations, shape recognition with scale variations). We experimentally validate our approach on synthetic and realistic datasets. In particular, we show that our face analysis system is particularly robust to large variations in head pose, identity, and partial occlusions.

These contributions have led to multiple publications in international journals and conferences:

Preprint

- E. Arnaud, A. Dapogny, and K. Bailly, "THIN: THrowable Information Networks and application for facial expression recognition in the wild", under review for *IEEE Transactions on Image Processing*, 2020.

Journal paper

- E. Arnaud, A. Dapogny, and K. Bailly, "Tree-gated deep mixture-of-experts for pose-robust face alignment," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 2019.

Conference papers

- E. Arnaud, A. Dapogny, and K. Bailly, "Tree-gated deep regressor ensemble for face alignment in the wild", in *IEEE International Conference on Automatic Face and Gesture Recognition*, 2019.

- S. Bernheim, E. Arnaud, A. Dapogny and K. Bailly, "MoDuL: Deep Modal and Dual Landmark-wise gated network for facial expression recognition", in *IEEE International Conference on Automatic Face and Gesture Recognition*, 2020.

Seminar

- E. Arnaud, A. Dapogny, and K. Bailly, "Tree-gated deep regressor ensemble for face alignment in the wild", *Reconnaissance des Formes, Image, Apprentissage et Perception*, 2020.

In addition, we have released the open-source Python code framework (running on top of TensorFlow) in [4], allowing to perform end-to-end facial expression recognition and face alignment in unconstrained environments.

1.4 Outline

This thesis is organized as follows. In Chapter 2, we review the main methods used for face alignment and facial expression recognition, then discuss those improving their robustness. In Chapter 3, we introduce deep ensemble methods that allow both to be accurate due to DNN-based methods that have a strong representational capacity while being robust to large variations due to ensemble methods that reduce the variance of prediction errors. We also discuss adaptive methods that allow to emphasize on the most informative parts of the model depending on the input image. These methods then allow us to generically define several adaptive deep ensemble methods, which are first evaluated on synthetic datasets, then on real-world datasets in Chapter 4, thus validating the robustness of our face analysis framework to large variations in head pose, identity, and partial occlusions. Finally, we conclude and discuss future works in Chapter 5.

Chapter 2

Related work

In this chapter, we present the methods used in the literature to model face alignment [79][179] and facial expression recognition [142][99]. These methods differ mainly by (1) the model architectures, which can use handcrafted- or learning-based approaches to extract a global or local representation of the face, and (2) the learning algorithms. All these items are discussed in the following sections.

For face alignment (in Section 2.1) and facial expression recognition (in Section 2.2) respectively, we first review the main methods with an emphasis on deep learning approaches that have led to significant advance in these fields. We then present the methods explicitly integrating a specific factor of variation in their model architecture or learning algorithm, in order to increase the robustness to variations related to this factor. In Section 2.3, we conclude and discuss the different advantages of these methods, so that they can be merged to take advantage of their respective benefits and provide a robust model.

2.1 Face alignment

From a 2D face image I, face alignment aims to locate P facial landmarks that form the face shape $\mathbf{y} \in \mathbb{R}^{2\times P}$ (the p-th column of this matrix corresponds to the 2D-coordinates of the p-th facial landmark). It typically starts from an initial shape $\hat{\mathbf{y}}^{(0)}$, then iteratively updates the landmark coordinates until convergence.

Two main types of models can be used for face alignment:

- Appearance model, which processes the pixel intensities, either by using a single holistic model for the global face appearance, or by using P local models for the shape-indexed local appearances.
- Shape model, which processes the landmark-wise distances to ensure that these landmarks form a feasible face shape. It allows to integrate shape constraints in the face alignment model.

In this section, we first present parametric methods that use *explicitly* shape constraints to align face appearance and shape models. We then present regression-based methods that directly learn the mapping from face appearance to landmark coordinates, thus using *implicitly* the shape constraints. Finally, we detail the methods integrating a specific factor of variation in their model architecture of learning algorithm.

2.1.1 Parametric methods

Parametric face alignment methods use deformable appearance and shape models, whose the control parameters are fitted for each test image. These methods mainly differ according to the approach used to model the face appearance: either a holistic approach with Active Appearance Models as representative, or a local approach with Constrained Local Models.

2.1.1.1 Active Appearance Models

Holistic appearance model. Holistic face appearance is defined by the shape-free texture, *i.e.* the pixel intensities of the warped image onto a given shape. Each training face image is warped onto the mean shape to generate the face appearance $\mathbf{g}$. Face appearance can then be modeled by applying Principal Component Analysis (PCA) to learn a set of basis appearances

forming the column matrix $\mathbf{P}_g$ that captures the appearance variations from a mean appearance $\bar{\mathbf{g}}$. Any face appearance $\mathbf{g}$ can then be generated as follows:

$$\mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g \mathbf{b}_g \tag{2.1}$$

where $\bar{\mathbf{g}}$ is the mean appearance, $\mathbf{P}_g$ the matrix containing the main modes of appearance variations, and $\mathbf{b}_g$ the appearance parameter vector controlling the deformation of the generated appearance.

Shape model. Face shapes are commonly first normalized by removing rigid transformations (translation, scaling, rotation) with Procrustes analysis [57]. Second, they are modeled by applying PCA to learn a set of basis shapes forming the column matrix $\mathbf{P}_s$ that contains the main modes of face shape variations from the mean shape $\bar{\mathbf{s}}$. Any normalized face shape $\mathbf{s}$ can then be generated as follows:

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{P}_s \mathbf{b}_s \tag{2.2}$$

where $\bar{\mathbf{s}}$ is the mean shape, $\mathbf{P}_s$ the matrix containing the main modes of face shape variations, and $\mathbf{b}_s$ the shape parameter vector controlling the deformation of the generated shape.

Aligning appearance and shape models. Active Appearance Models (AAM) [27] estimate landmark coordinates by aligning the appearance and shape models to a test image. Given shape parameters (i.e. including PCA, translation, scaling and rotation parameters) and appearance parameters, the alignment is evaluated by comparing the generated appearance and the test image warped by the generated shape.

This usually involves minimizing the sum of pixel squares of their appearance difference, called residual image. This optimization problem is solved by a search algorithm that iteratively updates model parameter estimations. Figure 2.1 illustrates iterative updates from an AAM search, by showing the generated appearance within the generated shape.

Two types of search algorithms can then be used to minimize the residual image: (1) analytic methods by using the gradients of residual image with report to the parameters in a Gauss-Newton fashion [119][10], and (2) regression-based methods by regressing the mapping between the residual image and the remaining parameter updates [65][141].

Unfortunately, AAM generalize poorly on unseen data by their limited capacity to capture all possible variations in the PCA matrices. Nevertheless,

Figure 2.1: The iterative process of the convergence of the AAM. Fore clarity, the facial landmarks are linked. Excerpt from [96].

advances on AAM have recently been proposed to improve their robustness: (1) by using in-the-wild training data [160], (2) by representing face appearance otherwise than the raw pixels, such as HOG [3], SIFT [162], SURF [26], or Haar-like features [9], (3) by fitting with advanced strategies [161]. However, the robustness to certain factors of variation such as partial occlusion remains difficult to achieve.

2.1.1.2 Constrained Local Models

Rather than modeling the face appearance in a holistic fashion, Constrained Local Models (CLM) learn P independent local appearance models for each facial landmark, whose the decisions are regularized by a global face shape model.

Given a face image I, CLM can be formulated as searching the shape parameter $\mathbf{b}$ that maximizes the product of the likelihood $\mathbb{P}(\mathbf{b})$ of the generated shape $\hat{\mathbf{y}}$ and the P independent local appearance likelihoods of each facial landmark:

$$\mathbf{b}^* = \arg\max_{\mathbf{b}} \mathbb{P}(\mathbf{b}) \prod_{p=1}^{P} \mathbb{P}(\hat{\mathbf{y}}_{.,p} = \mathbf{y}_{.,p} | \hat{\mathbf{y}}_{.,p}, I) \tag{2.3}$$

where $\mathbb{P}(\hat{\mathbf{y}}_{.,p} = \mathbf{y}_{.,p} | \hat{\mathbf{y}}_{.,p}, I)$ is the probability that the p-th true facial landmark is located at position $\hat{\mathbf{y}}_{.,p}$ given the local appearance provided by the $\hat{\mathbf{y}}_{.,p}$-centered image patch (illustrated in Figure 2.2).

Thus, the local appearances modeled by $\mathbb{P}(\hat{\mathbf{y}}_{.,p} = \mathbf{y}_{.,p} | \hat{\mathbf{y}}_{.,p}, I)$ allow to estimate the landmark coordinates forming a face shape, whose likelihood is measured by $\mathbb{P}(\mathbf{b})$ thus acting as a regularizer of the local appearance models.

The local appearance models can then be categorized into two groups:

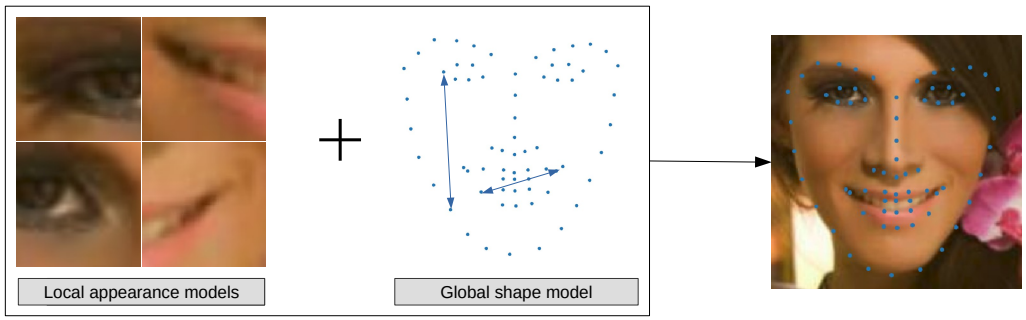


Figure 2.2: Constrained Local Models [179]. The local appearance models are independent from each other to estimate landmark coordinates. The global shape model ensures the consistency of the generated face shape.

- Classifier-based models [8][30]: for each facial landmark p, a binary classifier is trained to predict whether the $\hat{\mathbf{y}}_{.,p}$-centered patch is aligned with the true $\mathbf{y}_{.,p}$-centered patch.
- Regression-based models [105][31]: for each facial landmark p, a regressor is trained to predict the remaining displacement $\Delta\hat{\mathbf{y}}_{.,p} = \mathbf{y}_{.,p} - \hat{\mathbf{y}}_{.,p}$ towards the ground truth, depending on the current $\hat{\mathbf{y}}_{.,p}$-centered patch.

After *independently* estimating the coordinates of each landmark, the global face shape should be consistent. CLM then use a parametric face shape model to regularize and refine the local appearance models. Most CLM model the global face as a Gaussian multivariate, whose the covariance matrix is used to measure the likelihood of landmark-wise distances, and thus penalizing the infeasible face shapes.

The optimization problem in Equation 2.3 is then solved by a search algorithm that alternatively estimates the landmark coordinates that best fit the local appearance models independently, then refines them with the global shape model. This alternative optimization is repeated until convergence.

Although CLM is often better than AAM due to the local approach, which can for example alleviate the influence of partial occlusion, their robustness to large variations is nevertheless limited by their low capacity to capture all possible variations.

2.1.2 Landmark coordinates regression methods

Rather than using parametric methods to handle or regularize the face shape variations, regression-based methods aim to learn directly the mapping function from face appearance to landmark coordinates. We can then divide these methods into two categories:

- Direct regression: given a face image I, the objective is to learn a regression mapping function R between appearance features $\phi(I)$ and landmark coordinates $\mathbf{y}$:

$$R : \phi(I) \rightarrow \mathbf{y} \in \mathbb{R}^{2\times P} \tag{2.4}$$

- Cascaded regression: given a face image I and an initial face shape $\mathbf{y}^{(0)}$ (typically the mean shape), the objective is to sequentially learn a cascade of regression mapping function R_t between shape-indexed local appearance features $\phi(\mathbf{y}^{(t)}, I)$ and remaining displacements $\Delta\mathbf{y}^{(t)}$ in the shape space:

$$R_t : \phi(\mathbf{y}^{(t-1)}, I) \rightarrow \Delta\mathbf{y}^{(t)} \in \mathbb{R}^{2\times P} \tag{2.5}$$

Different regressor architectures and appearance features can be used for these methods. In particular, it is possible to learn them jointly by employing deep learning techniques.

2.1.2.1 Direct regression

Direct regression methods consist in learning directly (one single stage) the mapping from image appearance to landmark coordinates, without initialization. There are then two types of direct regression methods: local (by using a set of local patches), and global (by using the holistic appearance image).

Local direct regression methods randomly sample patches in the face image and learn the displacements towards the true landmark coordinates by using the patch-wise local appearance features. Unlike CLM that predict landmark-wise displacements independently, direct regression learns all displacements simultaneously. Random forests [181][34] (for regression) are often used to handle local patches, where each tree uses the appearance features related to a specific patch. The main drawback to these local methods is the *randomly* sampling of the patches. Indeed, highly informative parts of

the face can be under-represented and poorly captured by the patches. In addition, it can lead to a high sensitivity to occlusions.

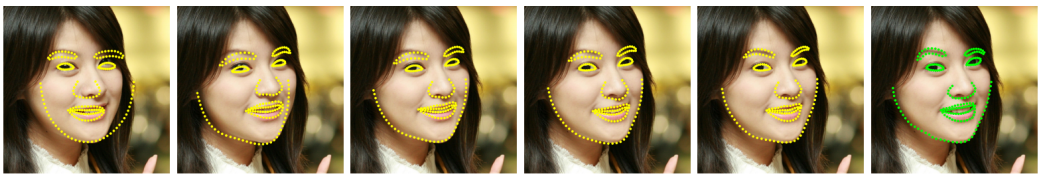
Global direct regression methods use instead the holistic image appearance to better capture all the facial parts that can be informative. This approach is the most direct but can rapidly lead to overfitting if the amount of training data available is not large enough, or underfitting if the representational capacity is not strong enough. In the last decade, new datasets annotated in landmark coordinates have been published [89][140][20][199], containing a lot of in-the-wild data. Combined with the availability of powerful computational resources, it then became possible to take advantage of the strong representational capacity of large deep neural networks (DNN), which can encompass many possible variations in the data. DNN-based methods [92][187][191] have thus allowed to significantly improve the localization performance compared to previous approaches, especially in unconstrained environments.

2.1.2.2 Cascaded regression

As direct regression in one-stage is very challenging, other approaches have instead divided the regression task into several tasks. They use sequentially several regressors, each predicting the remaining displacements between the landmark coordinates estimated by the previous regressors and the target coordinates. This is called cascaded regression.

Starting from a face image I and an initial guess $\hat{\mathbf{y}}^{(0)}$ (typically the mean shape), a first regressor R_1 is trained to predict the displacements $\Delta\mathbf{y}^{(1)} = \mathbf{y} - \hat{\mathbf{y}}^{(0)}$ from initial landmark coordinates towards the target. The input of R_1 is usually the shape-indexed local appearance features $\phi(I, \hat{\mathbf{y}}^{(0)})$ of the initial guess. At the end of this stage, a first landmark coordinates estimation is given by $\hat{\mathbf{y}}^{(1)} = \hat{\mathbf{y}}^{(0)} + R_1(\phi(I, \hat{\mathbf{y}}^{(0)}))$. Then a second regressor R_2 is trained to predict the remaining displacements $\Delta\mathbf{y}^{(2)} = \mathbf{y} - \hat{\mathbf{y}}^{(1)}$ from the estimation $\hat{\mathbf{y}}^{(1)}$, and so on. Figure 2.3 illustrates the cascaded regression procedure for face alignment, whose the pseudo-code is summarized in Algorithm 1.

Thus, at each cascade stage, the predictions are refined in order to improve the displacements predicted at the previous stages. Training is then performed in a coarse-to-fine fashion: the first stages of the cascade capture large deformations (*e.g.* translation, scaling), while the last stages focus on more subtle deformations to better align locally with the face (*e.g.* the contours of the mouth, eyes or jaw).



(a) $T = 0$ (b) $T = 1$ (c) $T = 2$ (d) $T = 3$ (e) $T = 10$ (f) Ground truth

Figure 2.3: Cascaded regression. From the initial mean shape, cascaded regressors are trained to sequentially update the landmark coordinates towards the ground truth. Excerpt from [84].

Algorithm 1 Cascaded regression

Input

Face image: I

True landmark coordinates: $\mathbf{y}$

Initial landmark coordinates estimation $\hat{\mathbf{y}}^{(0)}$ (*e.g.* mean shape)

Procedure

1: **for** $t = 1, ..., T$ **do**

2: Train a regressor R_t from shape-indexed features $\phi(I, \hat{\mathbf{y}}^{(t-1)})$ to the remaining displacements $\Delta\mathbf{y}^{(t)} = \mathbf{y} - \hat{\mathbf{y}}^{(t-1)}$

3: Update landmark coordinates estimation:

$$\hat{\mathbf{y}}^{(t)} = \hat{\mathbf{y}}^{(t-1)} + R_t(\phi(I, \hat{\mathbf{y}}^{(t-1)})) \tag{2.6}$$

Output

Landmark coordinates estimation: $\hat{\mathbf{y}} = \hat{\mathbf{y}}^{(T)}$

Cascaded regression methods depend on the model architecture of the regressor R_t and how the features are extracted through ϕ. The early work was proposed by Cao et al. [24], with a fern for the regressor and pixel intensity differences for shape-indexed local appearance features. Xiong et al. [180] use instead a cascade of linear regressors based on SIFT descriptors applied to each patch. However, these regressors are rapidly limited in their ability to capture many possible variations. The strong representational capacity of deep neural networks have then allowed to address this issue.

Sun et al. [153] were the first to use convolutional layers for each feature extractor ϕ_t at each stage t in the cascade. Note that the feature extractors

are indexed by t, and are thus different from one stage to another in the cascade, contrary to the previous methods. At the first stage, they use a deep CNN with four convolution layers to estimate the coordinates of 5 facial landmarks. Then, for each landmark, they refine locally the previous estimation using a shallower CNN. In the same vein, Zhou et al. [196] localize 68 facial landmarks. But instead of refining the landmark coordinates independently, they divide the face by components (eyebrows, eyes, mouth and nose), then share network parameters on each of these component to refine locally the coordinates of landmark subsets.

On the other hand, other approaches train the cascaded regressors in an end-to-end fashion [35][159]. For instance, Trigeorgis et al. [159] propose to mimic cascaded behaviour by a recurrent neural network. By sharing convolutional layers in all cascade stages, the feature extraction can be improved. In addition, learning simultaneously all cascaded regressors allows the landmark displacements to follow a more optimized trajectory, leading to increase the overall accuracy.

2.1.3 Methods explicitly integrating a factor of variation

Although the strong representational capacity of DNN-based methods have significantly improved the robustness in unconstrained environments by encompassing more possible variations, recent approaches also integrate a specific factor of variations to better adapt the model architecture and/or the learning algorithm, leading to achieve a better robustness to large variations in this factor. In what follows, we review such methods according to the factor of variation: facial expression, partial occlusion and head pose.

Facial expression. Face appearance and shape can be strongly affected by various facial expressions (FEs). Compared to neutral expression, each FE involves localized and complex changes in each facial component, depending on the FE intensity. Some approaches have been developed to be explicitly robust to FE variations. For instance, the authors in [146][178][103] use the multi-task learning framework to jointly locate facial landmarks and recognize facial action units. By sharing a common representation, what is learned for one task can help to better learn the other task, leading to improve their respective generalization capacities and accuracies. In particular, using AU recognition as co-task allow to better adapt the landmark detector to FE variations. Indeed, AU describes the facial expression, and learning to recognize them leads to identify certain discriminative face parts that can help both to better locate facial landmarks and to alleviate the influence of FE. It is also possible to add more tasks (*e.g.* facial attributes classification [192][135]) to further take advantage of the multi-task learning techniques and learn a more robust representation. However, these techniques require additional data annotated with auxiliary attributes.

Partial occlusion. Regarding partial occlusion, several difficulties have to be overcome to alleviate its influence: predicting the visible part of the face so as to use the associated local appearance to predict all the landmark coordinates, and being flexible enough to detect an immense variety of occlusions. Several strategies have thus been proposed in the literature [20][55][177][186][190]. These approaches generally consist in explicitly predicting the occluded part of the face. Mainly, these methods differ in three ways.

First, these methods differ in the way they predict occlusions. A first approach is to manually predefine several regions in the face image, then train an occlusion detection model predicting which region is occluded. In [20], they divide the face image into nine uniform parts, whereas [186] divide the image into facial components from the current landmark coordinates. Another way is to estimate the occlusion probability of each landmark [55][177]. It allows to emphasize on the local appearances of the visible landmarks.

Second, these methods differ in how are used the information related to the occluded regions. For example, in [186], the corresponding features of a region detected as occluded are discarded. Conversely, Zhang et al. [190] use denoising auto-encoders to recover the appearance of the occluded region and use this information to perform alignment, as shown in Figure 2.4.

Figure 2.4: Examples of face images denoised by auto-encoders in [190] to better locate landmarks (occluded or not). The first row shows the origin occluded faces. The second raw shows the denoised images. The third row shows the occlusion location estimation.

Finally, these methods differ on the type of training data that is used to learn the occlusion detector. For example, [55] use non-occluded face images augmented with synthetically generated occlusions, while [20] use real occluded data with manually annotated landmark visibility ground truth. In both cases, the data used are either non-realistic or tedious and time consuming to collect.

Head pose. Regarding head pose, a first approach to alleviate its influence on the localization performance is to train pose-dependent models, by having a specific model for a given range of pose (*e.g.* frontal, left and right profile). For instance, Cootes et al. [28] propose multi-view AAM by training one specific AAM for each pose range. At inference time, the landmark coordinates are estimated by the AAM model having the smallest fitting error among the three. Zhu et al. [198] extend multi-view methods with more complex model architectures and use more pose ranges.

Rather than directly predicting 2D landmark coordinates, other approaches predict 3D face shape to better handle the self-occluded landmarks. For instance, Jourabloo et al. [81] use a 3D deformable face shape model. They train a cascaded CNN models to iteratively update the 3D shape parameters from the face appearance. At each iteration of the cascade, a 3D-to-2D projection model is then used to predict the 2D landmark coordinates. However, it requires 3D scans of the face which are difficult to collect to be able to train the models with a lot of data.

Finally, other approaches instead explicitly use the head pose information into the model [92][200]. For instance, Kumar et al. [92] propose to integrate head pose estimation in the architecture to weight through gates several heatmaps, each estimating landmark coordinates. Thus, the model learns to extract pose-specific heatmaps, and uses gates to emphasize on the most relevant heatmaps, leading to improve the robustness to large pose variations. To the best of our knowledge, no face alignment model proposes to use it upstream in the network, which could allow to better condition the representation, upon which the regressor better adapt to locate facial landmarks.

Whether head pose or partial occlusion, we have therefore reviewed methods that explicitly integrate them into the model architecture to better cope with their variations. As we will see in Chapter 3 and validate in Chapter 4, our method also allows to integrate these factor of variations to several model layers in order to better locate facial landmarks and to increase the robustness to the most extreme variations.

2.2 Facial expression recognition

In this section, we present the main methods of facial expression recognition (FER). We have seen in Section 1.1.2 that the categorical approach classifying facial expression (FE) into 7 categories (*i.e.* happiness, surprise, sadness, anger, fear, disgust, and neutral) due to the annotation simplicity. In what follows, we then limit the review to categorical FER methods.

First, we briefly present in Section 2.2.1 the early work that used handcrafted representations on which to learn a FE classifier. Second, we review in Section 2.2.2 the DNN-based methods that have significantly increased the performance of FER models, especially in unconstrained environments. Finally, we review in Section 2.2.3 the methods explicitly integrating a specific factor of variation.

2.2.1 Handcrafted features based methods

As described in [142], the early works used handcrafted features upon which to learn a classifier, usually relying on SVM [78][144][148][195][164]. These models differ mainly according to the image descriptor: either by using only the face shape to extract geometric features, or by using the pixel intensities to extract appearance features. There are a multitude of possible descriptors: histogram, Gabor, Bag-of-Words, NMF, or part-based representations. Figure 2.5 illustrates these different descriptors given a face image.

Face shape. A first approach is to directly use the raw facial landmark coordinates [130]. Some typical patterns of facial expressions (e.g. smile) can then be detected by comparing distances or angles between different landmarks. Ignoring the pixel intensities allows then to be illumination-invariant. On the other hand, when the neutral face is available for each subject, the face deformation given by differential landmark coordinates can be used to recognize FE, as shown in Figure 2.5 - (a). However, the neutral face is often not available, and the accuracy of the FER model strongly depends on the robustness of the face alignment model. Finally, using the face shape alone may not be sufficient to represent FEs in some case. In particular, certain appearance patterns (e.g. dimples) not detectable by face shape may provide highly-discriminative information to recognize FE. It is therefore wiser to combine appearance and shape features to maximize the FER accuracy.

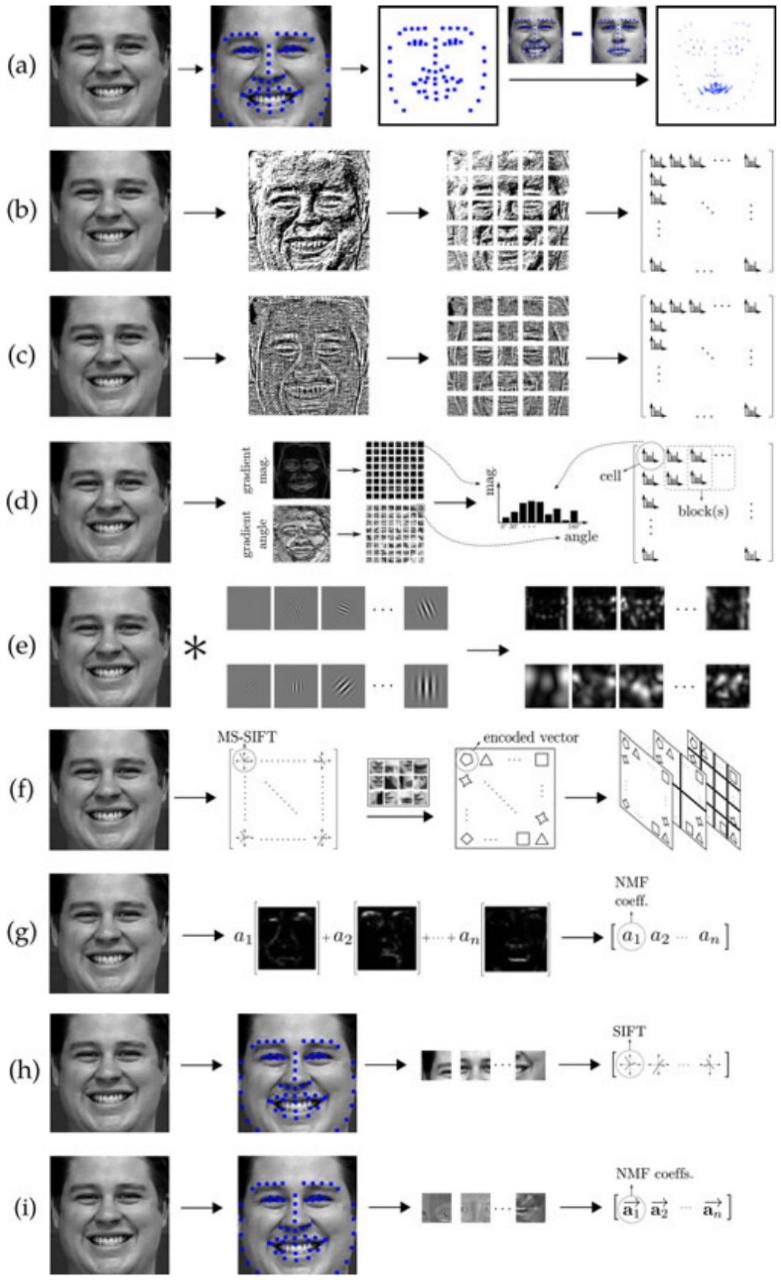


Figure 2.5: Handcrafted features extraction [142]. (a) Face shape; (b) LBP histograms; (c) LPQ histograms; (d) HoG; (e) Gabor-based representation; (f) Bag-of-Words; (g) NMF; (h) part-based SIFT; (i) part-based NMF.

Histogram-based methods. Histogram methods [40][145][128] divide the face image into several small uniform regions to extract local features on each of these regions. A pooling over regions allows to obtain local histograms. The overall features then correspond to the concatenation of these local histograms which are each normalized. These methods have been very popular for their implementation simplicity and low runtime, while allowing to efficiently extract discriminative low-level appearance features under variable illumination conditions. However, they are very sensitive to identity variations. The most popular histogram methods are Local Binary Patterns (LBP) [2] and Local Phase Quantization (LPQ) [126], illustrated in Figure 2.5 - (b)(c) respectively. In both cases, these methods describe local appearance variation around each pixel with an integer in $[0, 255]$. The histograms then give the distribution of each integer for each region of the face image. Other popular methods such as Histogram of Gradients (HoG) [33] or Scale-Invariant Feature Transform (SIFT) [110], illustrated in Figure 2.5 - (d), locally describe the direction of the edges by histograms giving the distributions of the image gradient magnitudes.

Gabor-based methods. Another approach to extract handcrafted appearance features is to use Gabor-based methods [113] by convolving the face image with a set of Gabor filters of various scales and orientations, as illustrated in Figure 2.5 - (e). These convolution operations generate high dimensional features, especially if the number of filters is large, thus requiring dimension reduction techniques (e.g. PCA). Similar to the above, Gabor-based methods are quite robust to illumination variations, but are very sensitive to identity variations.

Bag-of-Words. From local features (e.g. extracted by SIFT descriptors) describing local appearance variations, Bag-of-Words (BoW) [148] aims to learn a set of features called visual words (e.g. centroids resulting from K-means clustering), and then measure the similarity between the extracted local features and each of the visual words. The face image is then translated into a set of visual words that represent the facial expression. Figure 2.5 - (f) illustrates this method by representing each visual word by a symbol (e.g. circle, triangle, square) for each region of the image. In particular, visual words can be pooled hierarchically to extract features at various scales, as performed in [148] with spatial pyramid matching.

NMF-based methods. Non-negative matrix factorization (NMF) aims to factorize a matrix $\mathbf{X}$ into two matrix $\mathbf{B}$ and $\mathbf{A}$, such that the three matrices have no negative elements for interpretability. For FER, $\mathbf{X}$ is the matrix containing all the training face images: the i-th column corresponds to the pixel intensities of the i-th face image. Thus, each face image is written as a linear combination of basis images $\mathbf{B}$ such that: $\mathbf{X}_{.,i} = \mathbf{B}\mathbf{A}_{.,i}$. The features correspond to the weights $\mathbf{A}_{.,i} = (a_1, ..., a_n)$ used to reconstruct the face image from the base images. Figure 2.5 - (g) illustrates the image basis learned by NMF with associated weights for the given face image. NMF methods aim at minimizing the distance between $\mathbf{X}$ and its reconstruction $\mathbf{BA}$, sometimes adding sparsity constraints for more interpretability, so as to more emphasize some basis image to represent facial expressions. The robustness of these methods to variations in illumination or identity depends on the variability of the face images in the training data. With sufficient variability, NMF-based methods tend to be more robust to identity variations than previous methods by learning identity-free basis images. However, the linearity of the model limits performance.

Part-based methods. Part-based methods extract local appearance features from patches centered around each facial landmark. Figure 2.5 - (h)(i) illustrates part-based methods using SIFT and NMF descriptors respectively. The local descriptors are then independent of each other, leading to ignore the spatial relationships of the whole face and decrease the sensitivity to head pose variations. These methods are robust to illumination, but remain sensitive to identity variations. Furthermore, although part-based methods are less sensitive to head pose variations than previous methods, the local appearance is still affected by large poses, rapidly limiting the robustness to extreme conditions.

Handcrafted features based methods adapt well under lab-controlled conditions with little variation, but their performance is strongly deteriorated in unconstrained environments due to their limited representational capacity, thus preventing to model complex variations. As we will see in the following, DNN-based methods allow to address this limitation.

2.2.2 DNN-based methods

With an ever-increasing amount of labeled (in-the-wild) data and computational resources, DNN-based methods have become the mainstream approach to model facial expressions. It has allowed to significantly improve the accuracy of the FER models compared to the previous methods.

In what follows, we present the main DNN-based methods for FER: either (1) by finetuning classical pretrained networks for FER, (2) by improving the model architecture or (3) by improving the learning algorithm.

2.2.2.1 Finetuning a pretrained DNN

Rather than directly training deep FER models from scratch from relatively small datasets (several thousands of data), a first approach is to use classical networks (e.g. AlexNet, VGG, ResNet) for their strong representational capacity over large domains (usually pretrained on ImageNet containing several millions of data and several thousands of labels) and finetune them specifically for FER.

It is also possible to perform a multistage finetuning, by successively finetuning a pretrained network for more and more specialized tasks (e.g. object recognition (OR) $\rightarrow$ face recognition (FR) $\rightarrow$ facial expression recognition (FER)). In [88], Knyazev et al. have shown that using a FR network, pretrained on a large FR dataset, improves the learning of FER. Indeed, since the domain gap between FER and FR is narrow, finetuning robust FR representation leads to learn robust FE representation with few additional data annotated in FE. Another approach is to successively finetune a pretrained network on more and more specialized datasets by using transfer learning techniques. In [125], Ng et al. use a network pretrained on a first FER dataset (FER2013) and then finetune it on the target dataset (EmotiW), leading to increase the overall accuracy.

The drawback of finetuning strategy is that the final model is likely to use non-discriminative information for FER. For example, finetuning a FR network may lead to remain sensitive to identity variations because the FE representation can stay close to the identity representation. To address this issue, Ding et al. [43] propose to finetune only the representation layer (i.e. convolutional layers) and to jointly learn the prediction layer (i.e. fully-connected layers) from scratch, leading to improve the discriminative power of the FE representation.

2.2.2.2 Improving the model architecture

Rather than simply finetuning a classical network for FER, another approach is to improve the model architecture, either by adding auxiliary layers or by using deep ensemble methods.

Auxiliary layers. Several approaches add auxiliary layers to classical DNN in order to be specifically more effective for FER. For instance, Acharya et al. [1] have introduced covariance pooling to integrate second-order statistics into the architecture in order to better capture face deformations induced by FE, while Hasani et al. [60] have enhanced the ResNet architecture with more complex skip-connections. On the other hand, Zeng et al. [188] propose to learn an auxiliary network to model the latent truth from the inconsistent annotations between databases in order to improve cross-database accuracy. These methods have significantly improved FER accuracy, especially on in-the-wild datasets containing large variations.

Deep ensemble methods. In order to increase the robustness to large variations, another way to improve the model architecture is to combine DNN-based methods and ensemble methods. By using an ensemble of base networks instead of a single network, it allows to diversify the possible predictions, allowing to decrease the variance of errors, thus increasing the overall robustness. For example. Bargal et al. [11] propose to use the features extracted from several pretrained networks (VGG13, VGG16 and ResNet). Then, these features are concatenated to train a single FE classifier. Wen et al. [172] propose instead to jointly learn several convolutional networks, each initialized in a different way to ensure diversity. Fan et al. [49] propose in addition to specialize each of them on a specific face part (e.g. eyes, nose, mouth) by using the corresponding local appearances.

Rather than simply averaging or performing a majority vote on the ensemble decisions, it may be wiser to use a weighted averaging so as to emphasize on the most relevant base networks. For example, Fan et al. [49] sets the weights with an *a priori* on the discriminativeness of each facial region: 4/7 for the base networks using the eyes appearance, 2/7 for those using the mouth appearance, and 1/7 for those using the nose appearance. Kim et al. [86] propose instead to weight the base network predictions depending on their accuracies on a validation set. These deep ensemble methods have

particularly improved the robustness to the most extreme variations in the data.

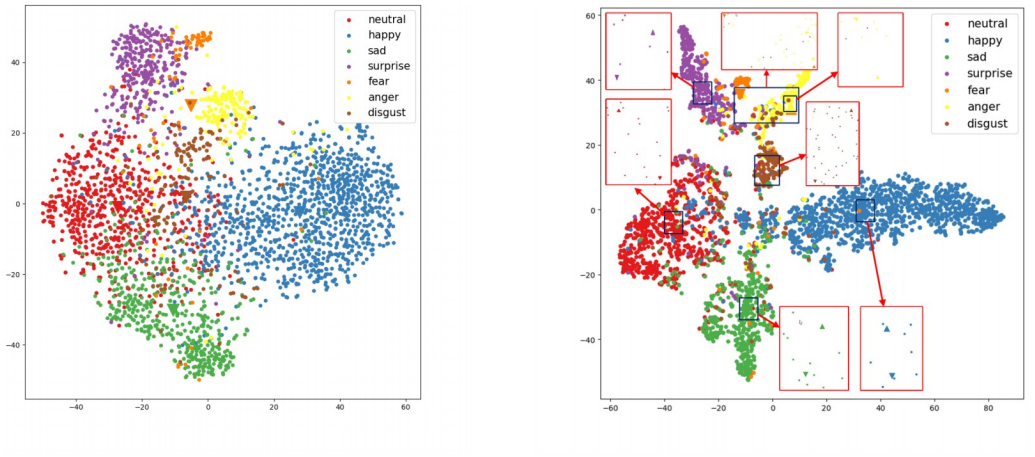
2.2.2.3 Improving the learning algorithm

The learning algorithm can also be improved to better learn FE representation, either by designing specific loss functions or by using the multi-task framework.

Specific loss functions. A large intra-class variance is often observed for FER in-the-wild. To address this issue, some approaches have been inspired by the center loss [173] that penalizes the distance between the features and the centroid of the corresponding class. For instance, Cai et al. propose the island loss [22] to penalize the pairwise-distance between different FE class centroids. To both reduce intra- and increase inter-class variances, Li et al. propose the separate loss [101] that maximizes intra-class similarity while minimizing the similarity between different FE classes. Figure 2.6 then illustrates the feature separation induced by this training strategy. Other methods follow advances in metric learning by using sample pairs. For instance, Li et al. propose the locality-preserving loss [100] to penalize the pairwise-distance between features of the same class, in order to compress local features space of each class. All these methods have thus allowed to improve the discriminative power of the FE representation.

Multi-task learning. Another approach to improve the FE representation is to learn it jointly with other correlated tasks in the multi-task learning framework, so as to regularize the training and boost the generalization capacities of each task, leading to improve the overall robustness. Typically, the model uses a backbone, which extracts a representation common to all tasks, then the higher layers are specific to each one.

Devries et al. [39] thus show that jointly learning facial landmark localization and FER is beneficial for each of these tasks. Indeed, representing facial expressions leads to identify certain discriminative face parts and can then help to better locate facial landmarks, and conversely, locating facial landmarks allows to better take into account the spatial relationships and can help to better discriminate facial expressions. Rather than using facial landmarks localization as co-task for FER, Pons et al. [132] propose to jointly learn to recognize basic FE and to detect facial action units. Knowing



(*a*) softmax loss (*b*) softmax loss and separate loss

Figure 2.6: Visualization of the intra-class features learned with/without separate loss [101]. This specific loss function allows to reduce intra-class variance while increasing inter-class variance, thus improving the discriminative power of the FE representation.

the categorical FE helps to recognize the combination of activated muscles that characterizes the expression, and conversely, knowing the combination of AUs helps to categorize the facial expression.

The disadvantage of these methods is that not all annotations are necessarily available for all tasks simultaneously. To address this issue, some approaches only update task-specific networks whose task-labels are available. For instance, Ranjan et al. [135] use several datasets, each containing different labels for different tasks (face detection, face alignment, head pose estimation, gender recognition, smile detection, age estimation and face recognition), in order to alternately train the different task-specific networks. It thus allows to take advantage of a lot of data, leading to learn robust features for each task.

2.2.3 Methods explicitly integrating a factor of variation

In the above, we have presented DNN-based methods that increase the robustness to large variations by improving model architectures or using more efficient learning algorithms. In the same vein as what has been developed for face alignment in Section 2.1.3, a greater robustness can be achieved by explicitly integrating in the method (model architecture or learning algorithm) an information related to a specific factor of variation, in order to better adapt the system to the most extreme variations in this factor. In what follows, we review recent methods using this strategy according to the factor of variation: partial occlusion, head pose, and identity.

Partial occlusion. Regarding partial occlusion, several approaches have been proposed in the literature, generally consisting in explicitly predicting the occluded part of the face. Similar to what has been done for face alignment, these methods differ in three ways.

First, these methods differ in the way they predict occlusions. For example, Huang et al. [72] divide the face into three parts (eyes, nose, mouth) and a sparse representation based occlusion detector allows to only use the visible and FE-discriminative face part. In contrast, Dapogny ey al. [36] divide the face into much finer parts and an auto-encoder is used to estimate the occlusion probability of each part, measured by the reconstruction error, in order to emphasize on the visible face part. On the other hand, it is also possible to jointly learn the FE classifier and an occlusion detector, as proposed by Li et al. [102] with their Patch-Gated CNN illustrated in Figure 2.7. They use gates to weight shape-indexed local appearance features, and can automatically learn the "unobstructed-ness" of each corresponding patch by assigning the highest weights to the unoccluded FE-discriminative face parts, as illustrated in Figure 2.7 (top).

Second, these methods differ in how are used the information related to the occluded face parts. For example, in [72], the features associated with occluded parts are discarded, while in [136], they use a generative model to recover the appearance of the occluded face parts to better classify FE.

Finally, these methods differ on the type on training data that is used to learn an occlusion-robust FE classifier, either non-occluded face images augmented with synthetically generated occlusions [36][29][91], or real occluded data [102].

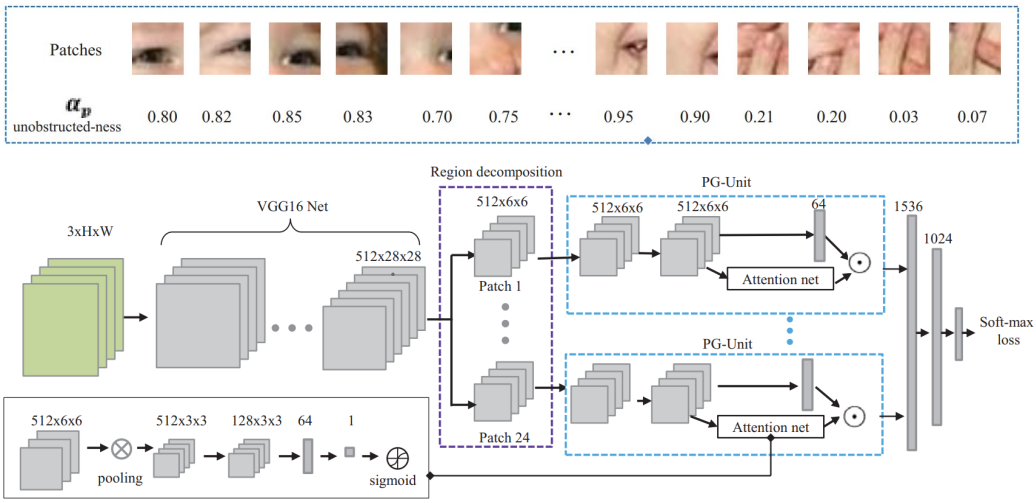


Figure 2.7: Patch-Gated CNN for Occlusion-aware FER [102]. The unoccluded face parts (left patches) are associated with high weights, and the occluded ones (right patches) are associated with low weights.

Head pose. Regarding head pose, a first approach to alleviate its influence on the FER model is to train one single FE classifier per pose cluster. At test time, head pose is first estimated, and is then used to weight the classifier predictions so as to select or emphasize on the most relevant ones. Following this strategy, Moore et al. [124] predefine pose clusters bins of 15° in yaw orientation, and train for each of these clusters a SVM to classify the facial expression from LBP features. During testing, a single SVM is then selected depending on the estimated pose. It is also possible to jointly learn the pose-specific FE classifiers and a pose estimation network, as proposed by Liu et al. [108]. At inference time, the estimated pose-cluster distribution provide then the weights of each FE classifier prediction, leading to emphasize on the most relevant FE classifiers. However, as we have discussed for face alignment, we argue that using head pose estimation upstream in the model could allow to better alleviate its influence on the representation, and thus better adapt the FE classifier to the most extreme pose variations.

A second approach consists in frontalizing the face to remove the influence of head pose on the face appearance while preserving the (pose-invariant) expressive information upon which train a FE classifier. For example, Vieriu et al. [167] propose to project 3D scans of face onto a 2D representation of

the frontalized face, but it requires high-resolution 3D images, which are very expensive to collect. Recently, Generative Adversarial Networks (GAN) [56] are used to generate very realistic frontalized faces to learn pose-invariant FER models. In [95], the generator frontalizes the face while preserving the expressive information and the discriminator distinguishes the real and fake (frontalized) face. The expressive information thus extracted is then directly feeded to a FE classifier. As this information is isolated from the pose component, the model is less sensitive to large pose variations. In the other hand, a GAN-based model can also be used to generate different FEs with multiple poses to enlarge and enrich the trainset for FER, as proposed in [189]. However, the training of GANs is difficult to tune, instable, and computationally expensive.

Identity. Learning FER models that are robust to identity variations requires having enough training data from different subjects in order to reflect the immense variability of identity-related information, such as morphological traits, age, gender, and ethnicity. Such data have recently been collected via webscrapping techniques [100][123][193], and are mainly process using DNN-based methods [122][107][106][182][183] to cope identity variations. In particular, these methods usually employ metric learning or adversarial learning techniques.

Regarding metric learning, Meng et al. [122] propose to design a contrastive loss, which consists in jointly learning similarity metrics related to FEs and identity from pairs of samples annotated with either information. Instead of using only pairs of samples, Liu et al. [107] propose to use a triplet loss, by jointly decreasing the distance between an anchor and a positive sample (i.e. same FE class and different identities) and increasing the distance between the anchor and a negative sample (i.e. different FE classes and same identity). In a later work [106], the same authors fix instead the neutral expression for the anchor to better capture the expressive information. Since neutral faces are not necessarily available, they propose to use a GAN to provide synthetic and realistic neutral faces.

Generating the corresponding neutral face while preserving identity-related information is equivalent to performing a de-expression process that is invariant to identity, thus enabling to extract FE features robust to identity variations. Following this strategy, Yang et al. have proposed De-expression Residue Learning [182]: during the de-expression process, the expressive in-

formation is still embedded in the intermediate layers of the generator, which then filters-out this information by learning the residue (containing the expressive component) that remains in the intermediate layers to generate the corresponding neutral face. As a FE can be regarded as the combination of a neutral face image and an expressive component, the identity-invariant expressive information thus extracted is directly fed into a FE classifier.

Another GAN-based method for identity-invariant FER is to generate each prototypic FE from any input face images while preserving the identity-related information. For instance, Yang et al. [183] propose an Identity-Adaptive generation model with two parts. First, several GANs are trained to generate images of the same subject with different FEs. Second, for each sample, the generated expressive face images are each fed into a pretrained CNN for FER, to determine which one is closest to the original image by comparing their respective features in this CNN, thus classifying the FE. However, the training data (from CK+ [111], Oulu-CASIA [194], BU-3DFE [185] and BU-4DFE [184] databases) have been collected in lab-controled conditions and are annotated in identity so as to have several expressive face images per subject. Unfortunately, the most recent in-the-wild FER datasets containing a lot of spontaneous FEs [100][123] are not annotated in identity, preventing to supervise the training of this approach. In addition, as said above, the training of GANs is difficult to tune, instable, and computationally expensive.

2.3 Outline

The main approaches proposed above allow us to retain several methods favouring the robustness both for face alignment and facial expression recognition: (1) DNN-based methods, (2) adaptive methods, and (3) ensemble methods.

DNN-based methods. Deep learning has become the mainstream approach in computer vision, including face analysis. With an increasing amount of in-the-wild labeled data, these techniques allow to jointly learn powerful predictors and complex high level representations that can encompass a lot of variations. It gives the opportunity to define a generic method that can be applied to multiple layers into the model (representation or prediction layer) for multiple face analysis tasks. We have also seen that designing specific model architecture or learning algorithm for face analysis tasks could lead to better performance than simply finetuning a pretrained model.

Adaptive methods. To better adapt to the most extreme variations of a given factor, most approaches explicitly integrate it into the model architecture or the learning algorithm. Regarding partial occlusion, an occlusion detector is often used, as seen for face alignment [20][55][177][186][190], but it is also possible to use gates to adaptively emphasize on the most visible and discriminative face parts, as seen in occlusion-aware FER [102]. Regarding head pose, gates can also be used in the model architecture to emphasize on the most relevant model parts depending on head pose estimation, as seen both for face alignment [92] and FER [108]. Regarding identity, most approaches use specific learning algorithms (e.g. metric learning [122][107]) to alleviate sensitivity to inter-subject variations, or GAN-based methods [183][182] that can generate prototypic facial expressions. But these methods require auxiliary annotations or are computationally expensive and difficult to tune. To the best of our knowledge, no identity-robust FER models explicitly integrate an identity representation in their architecture (e.g. by using the features extracted by a deep face recognition network) to adaptively emphasize on the most relevant model parts. For all these reasons, we propose to develop an adaptive DNN-based method that integrate into the model architecture an embedding related to an important source of variations.

Ensemble methods. Instead of using a single strong predictor that adapts to all possible variations, some approaches use instead an ensemble of base predictors, each specialized on predefined clusters of this factor. For instance, regarding head pose, multi-view models are proposed for face alignment [34] or FER [108] by specializing one single predictor per predefined pose cluster. These models thus use ensemble methods, which have already proven their efficiency both for face alignment and FER, e.g. by using random forest [137][84][34] or deep neural forest [37]. We then propose to develop an adaptive deep ensemble method, which use gates to emphasize on the most relevant base predictors depending on the input image. In addition, instead of predefining clusters upon which specialize each base predictor (e.g. predefined pose clusters for multi-view FER [108]), we propose to learn the clusters jointly with the whole system by using differentiable gates. The resulting adaptive deep ensemble method is detailed in the next chapter.

Chapter 3

Adaptive deep ensemble methods

In this chapter, we present our generic framework that can be applied for multiple layers of the model (e.g. prediction, representation layer) and multiple predictive tasks (e.g. face alignment, facial expression recognition). In Section 3.1, we first introduce ensemble methods and the benefits of using several base predictors rather than a single strong predictor. In Section 3.2, we detail the different modules to build an accurate and diverse ensemble, by varying training data, features or architectures used by each base predictor. In particular, modeling the base predictors with deep neural networks allows to merge deep and ensemble learning to take full advantage of their respective benefits and ultimately obtain a complex, accurate, and robust system that can be learned in an end-to-end fashion. In Section 3.3, we focus on adaptive networks allowing to emphasize on the most informative part of the system by the use of gates. In Section 3.4, we present the Mixture-of-Experts architecture that combines the above approaches: an adaptive deep ensemble model whose predictions are weighted by a gating variable that specializes each base predictor on a region of the input space. We finally detail in Section 3.5 our generic method extending the Mixture-of-Experts. In order to be more robust to large variations, we propose (1) a hierarchical gating structure to improve the specialization of the base predictors, (2) an exogenous gating variable, identified as an important source of variation in the data, to better adapt the model to the most extreme variations of this exogenous variable, and (3) a new training loss to remove undesirable exogenous-related information from the discriminative variable deciphering the task.

3.1 Ensemble methods: an introduction

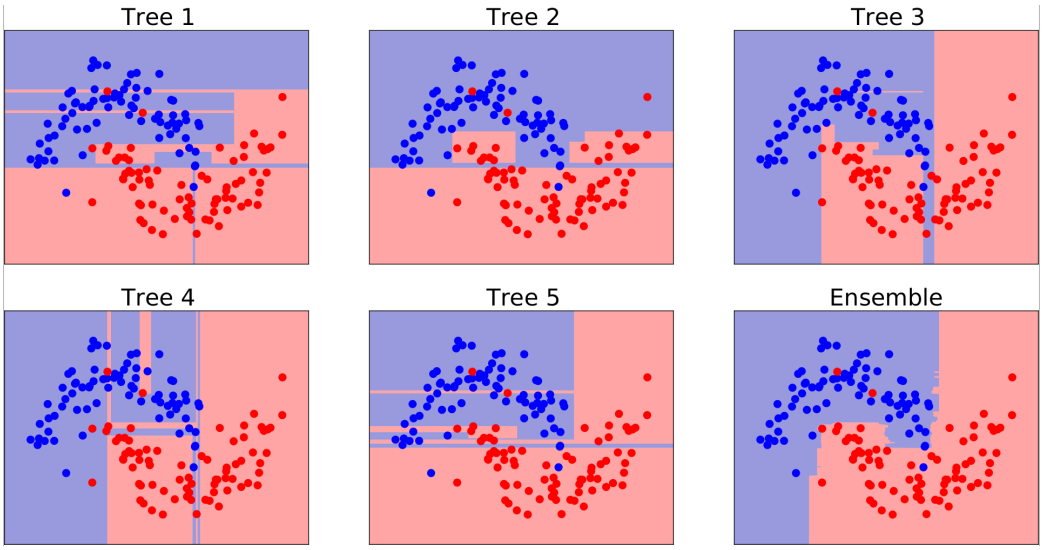


Figure 3.1: An ensemble of decision tree classifiers. The ensemble is more robust than any individual decision tree.

The main idea of ensemble methods can be summarized as follows: it is often better to combine several individual opinions than to choose the opinion of a single individual. Applied to machine learning, ensemble methods consists in combining accurate and diverse base predictors specialized on different data or feature subsets, instead of using a single strong predictor. In the literature, the notion of base predictor can be found as *weak learner* [52], or *expert* [76]. The search space of possible predictors is then further explored, leading to an accuracy higher than the one obtained with any of the base predictors. Recently, Fernandez-Delgado et al. [50] compared 179 classifiers from 17 families on 121 datasets (from UCI and other real-world problems) and showed that ensemble methods led to the best accuracy in most cases.

Consider a simple binary classification task. Figure 3.1 shows the decision borders of an ensemble of 10 decision tree classifiers trained on MOON [131], a toy dataset showing two interleaving half circles in the 2d plan. The trees share the same features (i.e. point coordinates), but each use different training data to diversify their training settings and to learn different deci-

sion borders in the input space. It can be seen that these borders are very irregular, illustrating an overfitting: each tree has perfectly memorized its training data but can't generalize on unseen ones. In particular, we can see that the two outliers located in the middle of the blue points push the trees to segment the space so as to contain them. The borders do not reflect then the data structure, and many unseen neighbors would be misclassified. By combining the trees for each data, the decision borders are then much more regular and allows the ensemble to be more robust to unseen data.

In [42], Dietterich identified three main reasons explaining the success of ensemble methods. First, the generalization capacity is improved by smoothing the decision borders, as illustrated above. Second, the decision borders of the ensemble cannot be learned by each base predictor. In the example above, each d-depth tree cannot segment the input space in the same way as the ensemble. The decision borders of the ensemble could only be approximated by a d'-depth tree with $d' >> d$. Thus, an ensemble method increases the search space of possible predictors. Third, specially for predictors trained with local search algorithms (e.g. gradient-based methods for neural network), an ensemble method decreases the risk of achieving a local optima. Indeed, many of these predictors can get stuck in local optima even if the trainset is large enough, thus limiting the exploration of the search space of possible predictors.

Ensemble methods are then a promising approach to improve the generalization capacity and therefore the robustness to large variations. In order to be able to take advantage of these methods, two conditions must be verified. First, the accuracy of each predictor of the ensemble should be better than those obtained in a fully random fashion, because if the majority of the predictors are wrong, then the ensemble is wrong. Second, the ensemble predictions should be as diverse and decorrelated as possible. Indeed, if they are strongly correlated and some predictors are wrong, then the others are also wrong and no gains can be obtained from the ensemble. Therefore, sufficient diversity between predictions is needed so that the predictors are able to cooperate, i.e. some predictors take over if others are wrong.

3.2 Modules to build an ensemble method

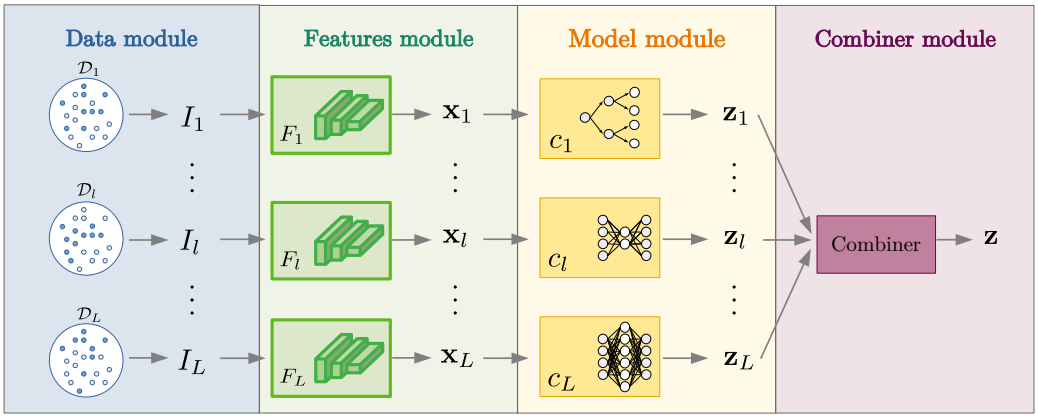


Figure 3.2: The use of one or more of the following modules allows to build an ensemble method [94]. **Data module**: different trainsets can be used for each base predictor, by sampling the original trainset. **Feature module**: different features can be used for each base predictor, by selecting them from a common feature set or by using different feature extractors. **Model module**: different architectures can be used for each base predictor. **Combiner module**: the predictions are either fused to generate a final decision, or a single one is selected depending on the input image.

Consider a dataset $\mathcal{D}$ where each sample has the form $(I_i, \mathbf{y}_i)$, with $I_i \in \mathcal{I}$ the pixel intensities of the i-th image and $\mathbf{y}_i \in \mathcal{Y}$ a categorical variable (i.e. $\mathcal{Y} = \{1, ..., K\}$) for classification or a continuous variable (i.e. $\mathcal{Y} = \mathbb{R}$) for regression. An ensemble method approximates the unknown true mapping between the input space $\mathcal{I}$ and the output space $\mathcal{Y}$ by using L base predictors, noted $c_1, c_2, ..., c_L$ respectively, whose predictions are combined to obtain a final decision. In order to achieve the accuracy and diversity properties, several modules can be used to build the ensemble method: data, features, model or combiner module.

In this section, we present each of these modules, illustrated in Figure 3.2. In the data module (Section 3.2.1), each base predictor is trained with its own trainset by randomly sampling the original trainset. In the feature module (Section 3.2.2), different features are used for each base predictor by selecting features from a common feature set or by using different feature extractors. In the model module (Section 3.2.3), different architectures are

used for each base predictor. In the combiner module (Section 3.2.4), the ensemble predictions are combined to generate a final decision: either by fusing the predictions (*e.g.* averaging or voting), or by selecting the prediction of a single base predictor depending on the region where the input is located.

3.2.1 Data module

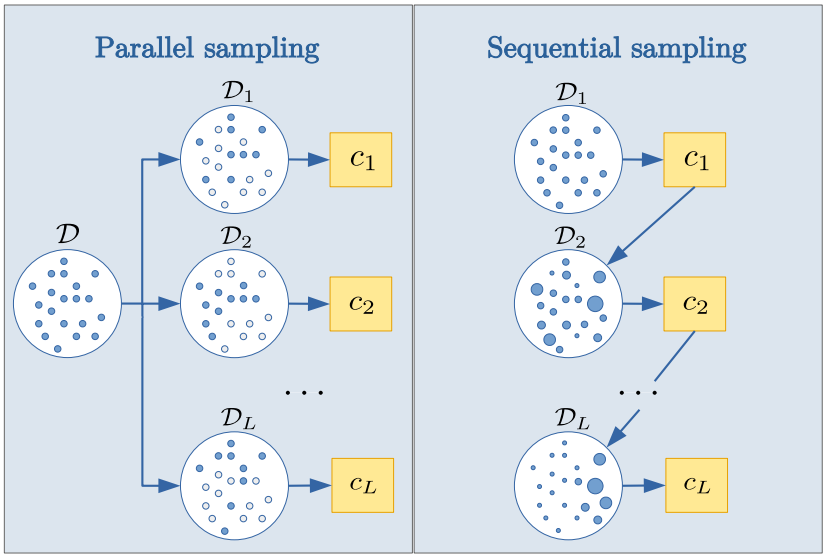


Figure 3.3: Two approaches to diversify the trainsets used by each base predictor. **Parallel sampling**: a single sampling distribution is used to create L new trainsets for each base predictor. **Sequential sampling**: the original trainset is uniformely sampled to train the first base predictor. Then, at step l, a new sampling distribution is used to train the l-th base predictor, by emphasizing on the hardest data misclassified by the previous trained base predictors. Each circle corresponds to a data. A blue one means that it is used in the training. Its size is proportional to its weight in the sample distribution. Excerpt from [133].

In order to diversify and decorrelate the ensemble predictions, it is possible to vary the trainsets used by each of them. This is the functionality of the data module: training an ensemble of L models by sampling the original trainset $\mathcal{D}$ to generate L new trainsets. Each base predictor c_l is then

trained with its own trainset $\mathcal{D}_l$. Sampling distributions can be parallel or sequential, as illustrated in Figure 3.3.

Parallel sampling. Parallel sampling allows to train the ensemble in parallel by using a single distribution to create L new trainsets for each base predictor. The base predictors can then be trained independently of each other from different training data, leading them to make their decisions as decorrelated as possible. Moreover, parallel sampling may improve runtime by using multi-core computing processors or parallel computers.

The most famous ensemble method using parallel sampling is bagging [17], standing for Bootstrap AGGregatING. Bagging uses bootstrap technique [156] to generate a new trainset $\mathcal{D}_l$ for each base predictor, by sampling the original trainset $\mathcal{D}$ uniformly from a bootstrap distribution. Once trained, the models are usually combined by voting (for classification) or by averaging (for regression) their decisions [149][150].

Sequential sampling. Contrary to parallel sampling which generates each trainset $\mathcal{D}_l$ independently of each other, sequential sampling adapts each new trainset depending on the accuracies of the previous trained base predictors, so as to emphasize on the hardest data misclassified. More specifically, at step l, the base predictor c_l is trained by sampling the original trainset $\mathcal{D}$ with a new distribution $D^{(l)}$ giving more weight to the examples that are the most difficult to predict with the earlier combination of base predictors build at step $(l-1)$. The ensemble is thus trained sequentially, so that each base predictor is mainly trained on the regions of the input space that have been the most difficult to predict for the previously trained base predictors.

Boosting methods, with AdaBoost [52] as representative, are the most famous approaches using sequential sampling. Boosting has been studied a lot, with a solid theoretical background, and largely applied in several applications. However, boosting has been observed to be very sensitive to noise [41]. In order to improve the robustness, several approaches use other loss functions to implicitly reduce the weights of the outliers (e.g. LogitBoost [53], GentleBoost [53]), or regularization techniques to explicitly bound the weights of each training data (e.g. MadaBoost [44], FilterBoost [16]), or more robust model architectures (e.g. XGBoost [25]).

3.2.2 Feature module

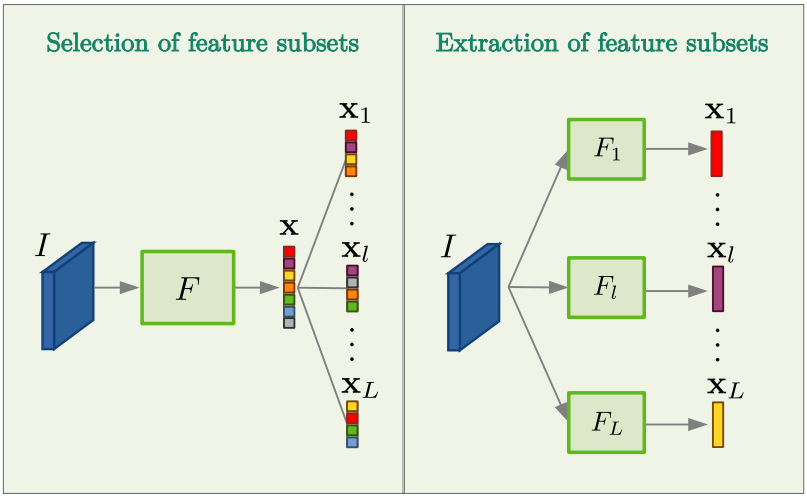


Figure 3.4: Two approaches to diversify the features used by each base predictor. **Selection of feature subsets**: one single feature extractor F outputs a feature set $\mathbf{x}$, and diverse feature subsets of $\mathbf{x}$ can be selected for each base predictor. **Extraction of feature subsets**: diverse feature subsets can be outputted by different feature extractors.

Generally speaking, given an input image I, a feature extractor $F : I \mapsto \mathbf{x}$ is used to output a feature set $\mathbf{x}$ upon which a predictor can be trained. Regarding ensemble methods, an approach to diversify and decorrelate the ensemble predictions can be to use different features for each base predictor. As illustrated in Figure 3.4, two approaches then allow to generate diverse feature sets: (1) by selecting different subsets of $\mathbf{x}$, or (2) by using different feature extractors instead of a single one.

Selection of feature subsets. Given a p-dimensional feature set $\mathbf{x}$, a first method to diversify the features is to select a q-dimensional ($q < p$) feature subset $\mathbf{x}_l$ of $\mathbf{x}$ for each base predictor. These feature subsets can be selected either (1) in parallel, or (2) sequentially. For (1), the features are selected independently of the other subsets. For instance, the random subspace method [63] randomly select features in $\mathbf{x}$ for each base predictor. For (2), the features in a subset $\mathbf{x}_l$ are selected depending on the previously

built subsets $\{\mathbf{x}_k, k < l\}$. For instance, Grabner et al. [58] propose a boosting procedure for feature selection, which selects the l-th feature subset so as to optimally refine the ensemble accuracy of the previously trained base predictors $\{c_k, k < l\}$.

Given a subset $\mathbf{x}_l$, several feature selection methods can be used. First, the selection can be done in a fully-random fashion, as in the random subspace method [63]. Second, it can be done in a non-random fashion [59], using either filter methods to select the features in $\mathbf{x}$ most correlated with the outcome variable $\mathbf{y}$, or wrapper methods to select the features in conjunction with the training of the associated base predictor (e.g. search for the feature subset $\mathbf{x}_l$ that maximizes the accuracy of the base predictor c_l).

Extraction of feature subsets. Another method to diversify the features is to use different feature extractors for each base predictor.

First, the feature extractors can differ in nature (e.g. the first base predictor uses HoG features, the second one uses SIFT features, and so on). For instance, Senechal et al. [143] use both geometric and appearance features into a multi-kernel SVM framework, while Oliveira et al. [127] use both handcrafted and learned features by combining HoG-based SVM and convolutional networks.

Second, the feature extractors can differ depending on the region of the image they take as input. Regarding the architecture, they can be based on a diverse fixed local region, as in [49] where each feature extractor is specialized on the local appearances of a specific face part (e.g. eyes, nose, mouth). Regarding the learning algorithm, diverse data augmentations, e.g. by randomly cropping the image for each feature extractor [45], can be performed to improve the overall robustness of the ensemble.

Finally, it is also possible to learn different feature extractors by using several DNNs (e.g. convolutional networks, auto-encoders) with diverse architectural and training hyperparameters. For instance regarding convolutional networks, it is possible to diversify the number of convolution layers and the size of the filters [172] or to use different parameter initializations [73].

3.2.3 Model module

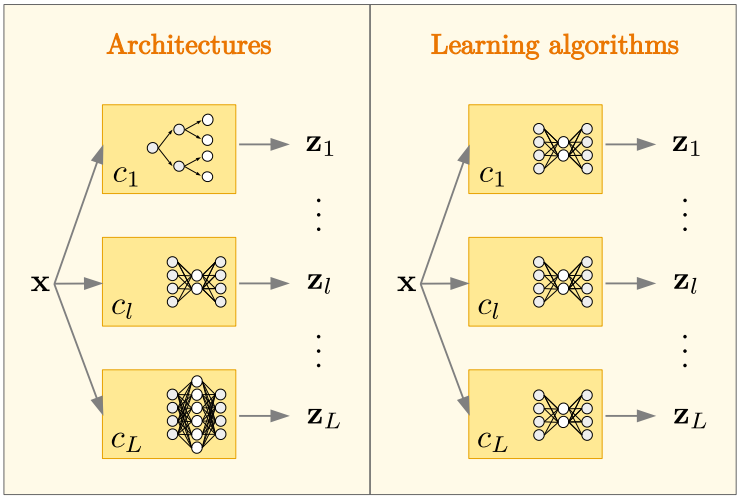


Figure 3.5: Examples of methods to diversify the base predictor models. **Architectures**: the base predictors use different architectures. **Learning algorithms**: the base predictors use different training hyperparameters for the same architecture (e.g. different parameter initializations).

As illustrated in Figure 3.5, a third approach to diversify and decorrelate the ensemble predictions is to use (1) different architectures or (2) different learning algorithms for each base predictor. In what follows, each base predictor is a deep neural network, for its strong representational power, its flexibility, and its ability to be learned in an end-to-end fashion. For (1), one can either vary architecture hyperparameters or use a specific architecture that behave as a deep ensemble method. For (2), one can either vary training hyperparameters or use specific learning and regularization techniques (e.g. snapshot ensemble, dropout) that behave as deep ensemble methods.

Varying hyperparameters. Consider a deep neural network c_θ with θ the trainable parameters. The dimension of θ is mostly very large (e.g. several millions of parameters), leading the training of c_θ to be extremely sensitive to its hyperparameters $\mathcal{H}$: either the architecture ones (e.g. depth, width, cardinality) or the training ones (e.g. θ initialization, type of optimizer, learning rate, batch size, regularization parameters).

Modifying only one of these hyperparameters often lead to a drastically different result. To address this issue, a first method is to train c_θ several times with different hyperparameters, e.g. by initializing θ in L different ways. This method is thus equivalent to learn different networks, where each network c_{θ_l} of the ensemble is a version of c_θ, whose its parameters θ_l are trained with its own hyperparameters $\mathcal{H}_l$.

Thus, it first allows to take the maximum advantages of ensemble methods described in Section 3.1: the search space of possible networks being particularly large and complex, different hyperparameters generate diverse local optimas, leading to learn simply an ensemble model with a high representational power, while improving the generalization capacity. Second, the networks can be trained independently and limit the need to tune the hyperparameters, which is the main problem for deep learning practitioners. However, the main drawbacks are that it's heavy to load in memory, and time-consuming to process at inference time.

Specific architecture. Some deep neural networks have an architecture that makes them behave like a deep ensemble, e.g. the residual networks (ResNet) [61]. ResNet uses skip-connections that bypass each layer by adding the input and output layers. By propagating the gradients through the skip-connections, the early layers receive more signals from the last layers, facilitating representation learning and allowing to train very deep networks. Thus, the input both traverses and bypasses each layer, and can flow from any layer directly to any subsequent layers. Veit et al. [166] then rewrite residual networks containing N layers as a collection of 2^N paths. They have experimentally shown that removing any layers has a negligible impact on performance. In particular, by removing more and more layers, they show that the overall performance reduces smoothly and the collection of paths in residual networks behave like ensembles.

Snapshot ensemble. Regarding the learning algorithm, another approach to diversify the ensemble predictions is snapshot ensemble method [69]. It saves the L best versions of a network c_θ during a single training run, depending on its performance on a validation set. A version is called snapshot or checkpoint model. This method is thus equivalent to sequentially learn L different networks, where the l-th network c_{θ_l} is a snapshot of c_θ, whose its parameters θ_l are initialized by the trained ones θ_{l-1} of the $(l-1)$-th network

and are then finetuned with the same hyperparameters as before. In order to avoid to get stuck in a local optima reached by $c_{\theta_{l-1}}$ and to explore more regions in the search space of the different possible versions of c_θ, it is also possible to update the hyperparameters, such as restarting the learning rate [109].

Regularization. Several regularization techniques can also be used to stabilize the training of c_θ, some of which behave as an ensemble method. Dropout [151] is a well-known regularization technique for training deep neural networks that can simulate having a large number of different network architectures. It's consists in randomly dropping a proportion of units in c_θ for each training iteration, thus temporarily stopping the parameters updates of the dropped units. Thus, it forces the neighboring units to generalize more and prevents co-adaptation between units. The network is then less sensitive to the specific parameters of each unit.

Dropping a proportion of units in c_θ is equivalent to sample a sub-network from it. For a network containing N units, there are 2^N possible sub-networks whose parameters are shared. Therefore, training c_θ with dropout can be regarded as training $L = 2^N$ networks with parameters sharing, and where each network is trained very rarely [62][151].

At test time, all the units of c_θ are used, but the parameters are rescaled: if a unit has a probability p of being retained during training, their specific parameters are multiplied by p during testing. Instead of storing an exponential number of trained networks that share parameters, training with dropout then simulates it with only one network.

On the other hand, rather than dropping out some units in each layer, other approaches propose to drop out some connection parameters (DropConnect [168]) to better prevent co-adaptation between the output units, or entire feature maps (SpatialDropout [157]) to better decorrelate them, or entire layers (deep networks with stochastic depth [70]) to better train the earlier layers by shorting the network during training.

3.2.4 Combiner module

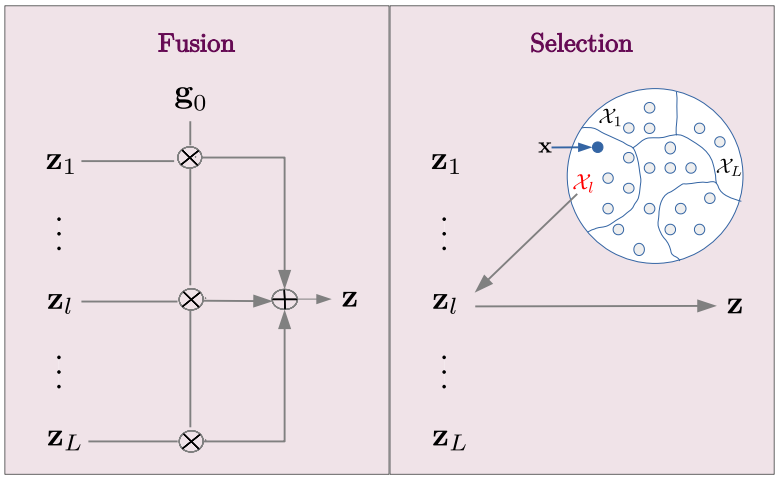


Figure 3.6: Examples of combination methods to produce a final decision from the ensemble predictions. **Fusion**: the ensemble predictions are averaged by weighting coefficients $\mathbf{g}_0$ that do not depend on $\mathbf{x}$. **Selection**: the final decision is the one that is generated by the most specialized base predictor depending on the location of $\mathbf{x}$ in the input space.

Let's note $\mathbf{Z} \in \mathcal{Z}^L$ the decision profile of $\mathbf{x}$, containing all the predictions $\mathbf{z}_l \in \mathcal{Z}$. For regression, each base predictor outputs a scalar, i.e. $\mathcal{Z} = \mathbb{R}$. For classification with K labels, each base predictor outputs either a binary decision vector, i.e. $\mathcal{Z} = \{0,1\}^K$, or a continuous one, i.e. $\mathcal{Z} = [0,1]^K$, giving the posterior probabilities of each class.

From an input representation $\mathbf{x}$ and the corresponding decision profile $\mathbf{Z}$, a combiner ζ generates a final decision $\mathbf{z}$. As illustrated in Figure 3.6, there are two types of combination methods :

- Fusion: all the ensemble predictions are used in the combiner, and the combination doesn't depend on the input $\mathbf{x}$.

$$\mathbf{z} = \zeta(\mathbf{Z})$$

- Selection: the final decision is the one that is generated by the most specialized base predictor depending on the location of $\mathbf{x}$ in the input space.

$$\mathbf{z} = \zeta(\mathbf{Z}, \mathcal{X}_{r(\mathbf{x})}) = \mathbf{Z}_{r(\mathbf{x})}$$

where $r : \mathcal{X} \rightarrow \{1, ..., L\}$ maps an input $\mathbf{x}$ towards a predefined region $\mathcal{X}_{r(\mathbf{x})}$ of the input space containing $\mathbf{x}$ and acting as the region of specialisation of a single base predictor.

3.2.4.1 Fusion

Averaging. For regression, the most popular combination method is to simply average the ensemble predictions, as follows:

$$\hat{y} = \frac{1}{L}\Sigma_{l=1}^{L}\hat{y}_l \tag{3.1}$$

where $\hat{y}_l$ is the l-th prediction.

Let's note y the ground truth, and μ, σ, Σ the averaged bias, variance, covariance respectively of the base predictors:

$$\begin{aligned}
\mu &= \frac{1}{L}\Sigma_{l=1}^{L}\mathbb{E}(\hat{y}_l - y) \\
\sigma &= \frac{1}{L}\Sigma_{l=1}^{L}\mathbb{E}((\hat{y}_l - \mathbb{E}(\hat{y}_l))^2) \\
\Sigma &= \frac{1}{L(L-1)}\Sigma_{l_1=1}^{L}\Sigma_{l_2=1, l_2\neq l_1}^{L}\mathbb{E}((\hat{y}_{l_1} - \mathbb{E}(\hat{y}_{l_1}))(\hat{y}_{l_2} - \mathbb{E}(\hat{y}_{l_2})))
\end{aligned} \tag{3.2}$$

By the bias-variance decomposition [163], it can be proved that:

$$\mathbb{E}((\hat{y} - y)^2) = \mu^2 + \frac{1}{L}\sigma + (1 - \frac{1}{L})\Sigma \tag{3.3}$$

If the base predictors are unbiased and decorrelated, then the error variance of the ensemble is L times smaller than a single predictor on average. Even if these assumptions rarely hold true (the base predictors are often correlated because they are trained for the same task), this combination method is the most popular for its simplicity and efficiency. It can be noticed that the median can be used instead of the average to reduce the sensitivity to extreme values.

Voting. For classification, the most popular combination methods are voting. Let's assume that the decision profile has binary values, i.e. $\mathbf{Z} \in \{0, 1\}^{K \times L}$ where $\mathbf{Z}_{k,l} = 1$ if the l-th predictor assigns the k-th label to $\mathbf{x}$ and $\mathbf{Z}_{k,l} = 0$ otherwise. The main voting methods are the following:

- Plurality voting. The predicted class is the one that takes the largest number of votes:

$$k^* = \underset{k \in \{1,...,K\}}{\arg\max} \Sigma_{l=1}^{L} \mathbf{Z}_{k,l} \tag{3.4}$$

- Majority voting. The predicted class is the one that takes more than half of the votes, with a reject option:

$$k^* = \begin{cases} j & \text{if } \Sigma_{l=1}^{L} \mathbf{Z}_{j,l} > \frac{1}{2} \Sigma_{k=1}^{K} \Sigma_{l=1}^{L} \mathbf{Z}_{k,l} \\ \text{reject} & \text{otherwise} \end{cases} \tag{3.5}$$

In the case of binary classification ($K = 2$), plurality and majority voting are the same method.

Stacking. In the above, the combiner uses a predefined rule to fuse the ensemble predictions. But it is possible to train a learner to perform the role of a combiner. Following this strategy, stacking methods [174][18] train a so-called *meta*-learner (*e.g.* a simple linear estimator, a decision tree or a deep neural network) to combine the ensemble predictions. The original trainset $\mathcal{D}$ is firstly used to train the base predictors, then a new trainset $\mathcal{D}'$ is generated: the inputs of $\mathcal{D}'$ are the decision profiles on $\mathcal{D}$ and the outputs are still the same as in $\mathcal{D}$. This method then allows to recognize patterns in the decision profiles to better fusing the base predictors. However, the training of the meta-learner is separated from the training of the base predictors, losing the interest of learning in an end-to-end fashion with deep neural networks.

3.2.4.2 Selection

Instead of fusing the ensemble predictions, another combination method is to select the decision of a single predictor of the ensemble depending on the input subspace where $\mathbf{x}$ is located. The selected predictor is then called the *expert* of the region to which $\mathbf{x}$ belongs. In what follows, the decision profile $\mathbf{Z}$ is assumed to have continuous values.

Cluster and selection. A first approach is to predefine regions $\mathcal{X}_1, ..., \mathcal{X}_L$ in the input space $\mathcal{X}$ to obtain specialized predictors on each of these regions [93]. For example, these regions can be build by using the K-means algorithm [115] on the trainset $\mathcal{D}$.

Let's note $\mathcal{D}_l$ the dataset containing the training data in the l-th region:

$$\mathcal{D}_l = \{(\mathbf{x}, y) \in \mathcal{D} : \mathbf{x} \in \mathcal{X}_l\} \tag{3.6}$$

Two approaches allow then to specialize the predictors:

- Each predictor c_{θ_l} is trained by using the dataset $\mathcal{D}_l$, and is thus specialized in $\mathcal{X}_l$. If a new data $\mathbf{x}$ is located in $\mathcal{X}_l$, then c_{θ_l} is selected.
- Each predictor c_{θ_l} is trained by using the entire trainset $\mathcal{D}$. Then, for each region $\mathcal{X}_l$, we select the predictor that has the best performance on $\mathcal{D}_l$. This predictor is then the most specialized for $\mathcal{X}_l$, and is selected if a new data is in $\mathcal{X}_l$.

Dynamic classifier selection. Rather than predefining regions of specialization on which a single predictor is selected, a second approach is to use the neighborhood of a given input $\mathbf{x}$ to adaptively select the most accurate predictor on this neighborhood [32].

Given an input $\mathbf{x}$, the selected predictor is the one that maximizes the average accuracy of the n-nearest samples from $\mathbf{x}$ in the trainset or a validation set. However, the ensemble predictions must be evaluated for each sample in the neighborhood, thus leading to be very expensive computationally.

3.2.4.3 Discussion

By using all the ensemble predictions at once to generate a final decision, fusion methods reduce the variance of prediction errors, which is an advantage for robustness to large variations. However, these methods don't explicitly adapt to the region in the input space where $\mathbf{x}$ is located. Conversely, selection methods allow to use only the most relevant predictor of the ensemble to decipher $\mathbf{x}$. However, this is not necessarily robust to large variations. Indeed, let's assume $\mathbf{x}$ belongs to a region $\mathcal{X}_i$ and the i-th predictor is the most relevant to decipher it. If noise is added and pushes $\mathbf{x}$ in a neighboring region $\mathcal{X}_j$, the j-th predictor will be used (which is not necessarily suitable for deciphering $\mathbf{x}$) instead of the i-th one.

To address this issue, it is possible to merge these methods, by weighting all the ensemble predictions whose weights depend on the region of the input space where $\mathbf{x}$ is located. The Mixture-of-Experts architecture [76] is a such instance, which we will detail in Section 3.4.

3.2.5 Ensemble method instances

An ensemble method then combines several of the modules presented so far, taking advantage of their respective benefits. The combination of several methods from different modules allows to further diversify the ensemble than using each of them individually.

Random forest. A first instance of such a method is to combine the data module and the feature module. Following this strategy, Breiman et al. [19] proposed random forest (RF), which is one of the most popular ensemble methods. For the data module, RF uses bagging to diversify the trainsets of each predictor. For the feature module, RF uses random subspace method to diversify the features of each predictor. For the model module, RF uses a decision tree for the architecture of each predictor. The ensemble is called a forest.

Given an input representation $\mathbf{x}_l \in \mathcal{X}_l$ of a sample from a trainset $\mathcal{D}_l$ and a decision tree with split nodes $\mathcal{S}$ and terminal nodes $\mathcal{T}$, the prediction $\mathbf{z}_l$ of the l-th predictor can then written as:

$$\mathbf{z}_l = \sum_{t \in \mathcal{T}} \left(\prod_{s \in \mathcal{S}} d_s(\mathbf{x}_l)^{\mathbb{1}_{t \swarrow s}} (1 - d_s(\mathbf{x}_l))^{\mathbb{1}_{t \searrow s}} \right) \pi_t \tag{3.7}$$

where π_t is the prediction of the t-th terminal node of the decision tree, $d_s(\mathbf{x}_l) = \mathbb{1}_{\mathbf{w}_s . \mathbf{x}_l - b_s > 0}$ the binary routing function with the trainable parameters $(\mathbf{w}_s, b_s)$ of the s-th split node, and $t \swarrow s$ is true if the t-th terminal node belongs to the left subtree of the s-th split node. Thus, the l-th decision tree splits hierarchically the input space $\mathcal{X}_l$, where each subspace is as homogeneous as possible (e.g. by using Gini impurity criteria).

The forest predictions are finally combined by averaging (for regression), or voting (for classification).

It is also possible to use boosting instead of bagging, as proposed by Bernard et al. [12] with the dynamic random forest. The forest is then built sequentially so as to train each new decision tree with the hardest data, *i.e.* generating the most errors with previously built trees. This allows to take advantage of the diversity of features used by each tree, the diversity of training subsets used by each tree and specifically built to boost the ensemble accuracy.

Neural forest. Recently, Kontschieder et al. [90] proposed neural forest (NF). Similarly to RF, NF is a set of trees using each different features and different training data, and whose ensemble predictions are combined by averaging or voting. However, the routing of each tree is probabilistic and modeled by a differentiable function jointly trainable with a gradient-based optimization algorithm, contrary to a decision tree whose routing is deterministic and not differentiable. It allows to learn more complex splits in the input space, leading to increase the accuracy. Each tree is then called neural tree.

At split node s and given an input $\mathbf{x}$, the probability to reach the left child is modeled by:

$$d_s(\mathbf{x}) = \sigma(\mathbf{w}_s.\mathbf{x} - b_s) \tag{3.8}$$

where σ is the sigmoid function, replacing the binary routing in decision tree.

The predictions of each neural tree are then computed in the same way as in Equation 3.7, thus corresponding the expectation of the predictions π_t of each terminal node.

For classification, π_t corresponds to the posterior probabilities on $\mathcal{Y}$ estimated by the terminal node t, initialized by an uniform distribution: $\pi_t = (|\mathcal{Y}|^{-1}, ..., |\mathcal{Y}|^{-1})$. The matrix π containing all the predictions of the terminal nodes is then trained with an offline learning approach: at the end of each epoch, π is updated to solve a convex optimization problem and a global solution can be easily determined. On the other hand, the routing parameters are trained by varying mini-batches for each neural tree in order to diversify the training subsets, similarly to bagging. For regression, terminal nodes can also be fixed at the beginning of the training, so as to train only the routing parameters, as proposed in [37].

3.2.6 Conclusion

We have presented the four modules that allow to build an ensemble method by diversifying the trainsets, the features or the architectures of each predictor. Different ways to combine their predictions have been also reviewed, as well as ensemble methods methods combining several modules.

In particular, modeling the predictors and the combiner with deep neural networks allows to learn them jointly, thus improving the two fundamental properties of ensemble methods described in 3.1: (1) a better accuracy for each base predictor, and (2) a better cooperation between them.

In addition, as described in 3.2.4, using a combiner that weights the ensemble predictions adaptively to the input also reduces the variance of prediction errors (like fusion methods) while emphasizing on the most relevant predictors deciphering the input (like selection methods). This adaptation mechanism is therefore a promising approach to better adapt the model to the most extreme variations.

In what follows, we review adaptive networks, i.e. deep neural networks incorporating an adaptation mechanism.

3.3 Adaptive networks

The strong representational capacity of deep neural networks allows to model many possible variations in the data, under the condition that the network is deep enough and the layers are large enough. All network layers are jointly trained to encompass these variations. The network has thus to learn the same transformation functions for all the samples. Given a sample, some layers can then extract irrelevant features to decipher the task.

To alleviate this constraint, some approaches aim instead to adapt the transformations according to the sample. Generally speaking, it consists in using adaptive gates to emphasize or select the most informative parts of the network depending on the sample. The gating function can be regarded as an attention mechanism. Its differentiability makes it possible to learn it jointly with the entire network, and to obtain an adaptive network that better decipher the input, especially under the most extreme conditions.

It is then possible to apply gates at different levels of the model architecture: (1) either in an intra-layer fashion to select the feature maps and the spatial regions containing the discriminative informations, or (2) in an inter-layer fashion to select the most specific layers of the network.

3.3.1 Intra-layer gates

Given an input image and a convolutional layer of the model architecture, let's note $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ the feature maps extracted, i.e. 3D visual features along the 1D-channel and 2D-spatial axes. In order to improve its representation power, gates can be applied to each of these dimensions, so as to emphasize on the most informative parts of $\mathbf{x}$. Gating the channel axis allows to emphasize on the most informative feature maps. Gating the spatial axes allows to emphasize on the most informative spatial regions of each feature map.

Channel gates. Each feature map can be regarded as a visual feature detector (e.g. detecting a dimple). Channel gates focuses then on "what" are the feature detectors of $\mathbf{x}$ useful to decipher the task. To emphasize on some feature maps rather than others, Hu et al. [66] introduced Squeeze-and-Excitation block, in which gates are used to adaptively weight each feature map. A *squeeze* function $G_{sq}^{(ch)} : \mathbf{x} \in \mathbb{R}^{H \times W \times C} \mapsto \mathbf{c} \in \mathbb{R}^{C}$ first extracts a channel descriptor $\mathbf{c}$ by squeezing the global spatial information of $\mathbf{x}$ on

each of its channels with an average-pooling along the spatial axes. Then, an *excitation* function $G_{ex}^{(ch)} : \mathbf{c} \in \mathbb{R}^C \mapsto \mathbf{g}^{(ch)} \in [0,1]^C$ uses the channel descriptor to weight each feature map with channel-gates. The c-th feature map of the channel-weighted features $\mathbf{x}'$ can then be written as:

$$\mathbf{x}'_{.,.,c} = \mathbf{g}_c^{(ch)} . \mathbf{x}_{.,.,c} \tag{3.9}$$

where $\mathbf{g}^{(ch)} = G_{ex}^{(ch)} \circ G_{sq}^{(ch)}(\mathbf{x})$.

The excitation function $G_{ex}^{(ch)}$ is modeled with a 2-FC network, thus capturing the inter-channel dependencies. The most informative feature maps are then the most weighted ones, and are then emphasized by feeding $\mathbf{x}'$ into the subsequent layers.

Spatial gates. Spatial gates focuses on "where" is the discriminative part of $\mathbf{x}$, which is then complementary with channel attention. Woo et al. [175] thus propose to sequentially use channel and spatial gates, as illustrated in Figure 3.7.

First, a channel attention module outputs channel-gates to weight each feature map of $\mathbf{x}$ in the same way as SE block. However, these channel-gates are computed from two different channel descriptors (i.e. average-pooling and max-pooling along the spatial axes respectively). They experimentally validated that using these two descriptors significantly improves the representation power rather than using each independently.

Second, given the channel-weighted features $\mathbf{x}'$, a spatial attention module outputs spatial-gates to weight each local unit of $\mathbf{x}'$. In particular, a squeeze function $G_{sq}^{(sp)} : \mathbf{x}' \in \mathbb{R}^{H \times W \times C} \mapsto \mathbf{s} \in \mathbb{R}^{H \times W \times 2}$ first extracts a spatial descriptor by concatenating the global information with an average-pooling and a max-pooling along the channel axis of $\mathbf{x}'$. Then, an excitation function $G_{ex}^{(sp)} : \mathbf{s} \in \mathbb{R}^{H \times W \times 2} \mapsto \mathbf{g}^{(sp)} \in [0,1]^{H \times W}$ extracts spatial-gates $\mathbf{g}^{(sp)}$ to weight each local unit of all feature maps. The c-th feature map of the refined features $\mathbf{x}''$ can then be written as:

$$\mathbf{x}''_{.,.,c} = \mathbf{g}^{(sp)} \otimes \mathbf{x}'_{.,.,c} \tag{3.10}$$

where $\otimes$ is the element-wise operator, and $\mathbf{g}^{(sp)} = G_{ex}^{(sp)} \circ G_{sq}^{(sp)}(\mathbf{x}')$.

The excitation function $G_{ex}^{(sp)}$ is modeled with a shallow convolutional network, thus capturing the inter-spatial dependencies. The most informative spatial regions are then the most weighted ones.

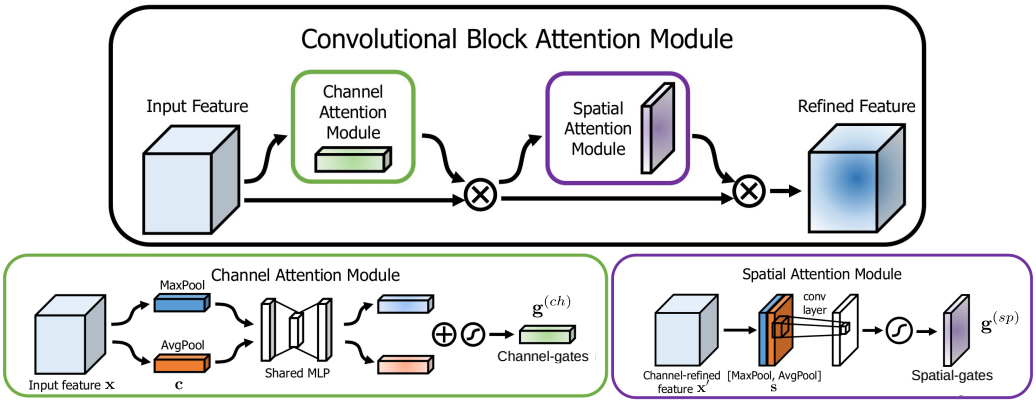


Figure 3.7: Channel and spatial attention emphasize on the most informative parts of a convolutional layer [175]. Channel attention module extracts channel-gates to weight each feature map. Spatial attention module extracts spatial-gates to weight each local unit of all feature maps.

Woo et al. have experimentally shown that using sequentially channel and spatial attention improves the representational power rather than using each independently. In particular, spatial attention helps to remove the noise generated by certain factors of variation in the feature maps, leading to better capture the visual structure that allows to decipher the task. It is also possible to directly learn a 3D-mask that combines both channel and spatial gates, as proposed in [169]. However, Woo et al. have shown that separating channel-gates and spatial-gates limits the number of trainable parameters while improving performance.

Therefore, a first approach to design an adaptive network is to use intra-layer gates for each representation layer: either on the channel axis to emphasize on the most suited feature maps depending on the input, or on the spatial axes to emphasize on the most discriminative regions while removing the noise related to the factor of variations. Recent work [68][134] relies on this approach to better adaptively select the most informative parts of the representation layers.

3.3.2 Inter-layer gates

A second approach to design an adaptive network is to use inter-layer gates to emphasize or select the most sample-specific layers. It allows to specialize each layer on data subsets, preventing a redundancy of information between layers and improving the overall performance.

Given an input image $\mathbf{x}_0$, traditional feed-forwark networks extract intermediate representations $\{\mathbf{x}_1, ..., \mathbf{x}_L\}$ by sequentially applying transformations through layers, which can then be written recursively as follows:

$$\mathbf{x}_l = F_l(\mathbf{x}_{l-1}) \tag{3.11}$$

where F_l is the transformation of the l-th layer extracting the intermediate representation $\mathbf{x}_l$ from the input layer $\mathbf{x}_{l-1}$.

To use an inter-layer gating mechanism, Srivastava et al. [152] introduced Highway Network, in which each input layer $\mathbf{x}_{l-1}$ is fed into a gating function G_l taking values in $[0, 1]$ to modulate the signals of $\mathbf{x}_{l-1}$ and $F_l(\mathbf{x}_{l-1})$ in the output layer, as follows:

$$\mathbf{x}_l = G_l(\mathbf{x}_{l-1}).F_l(\mathbf{x}_{l-1}) + (1 - G_l(\mathbf{x}_{l-1})).\mathbf{x}_{l-1} \tag{3.12}$$

If $G_l(\mathbf{x}_{l-1}) = 1$, the l-th layer is suited to handle the input image and $F_l(\mathbf{x}_{l-1})$ is fully fed into the $(l+1)$-th layer. If $G_l(\mathbf{x}_{l-1}) = 0$, the l-th layer is not suited to handle the input image and the transformation F_l is bypassed. The input layer $\mathbf{x}_{l-1}$ is then directly fed to the $(l+1)$-th layer. By modeling the gating function as $G_l(\mathbf{x}_{l-1}) = \sigma(\mathbf{W}_{G_l}.\mathbf{x}_{l-1} + \mathbf{b}_{G_l})$ where $\mathbf{W}_{G_l}$ and $\mathbf{b}_{G_l}$ are the trainable parameters, its differentiability allows to learn it jointly with the entire network. Thus, the inter-layer gating functions $\{G_1, ..., G_L\}$ allow to filter the information through the entire network depending on the input image $\mathbf{x}_0$.

Recently, Veit et al. [165] have applied this approach to ResNet architectures. As a remember, ResNet uses skip-connections that bypass each layer by adding the input and output layers, as follows:

$$\mathbf{x}_l = F_l(\mathbf{x}_{l-1}) + \mathbf{x}_{l-1} \tag{3.13}$$

Veit et al. then propose Adaptive Inference Graph (AIG): for each ResNet layer l, a binary gate $G_l(\mathbf{x}_{l-1}) \in \{0, 1\}$ selects or skips the transformation F_l depending on the input layer, as follows:

$$\mathbf{x}_l = G_l(\mathbf{x}_{l-1}).F_l(\mathbf{x}_{l-1}) + \mathbf{x}_{l-1} \tag{3.14}$$

To incorporate binary gates, Veit et al. use differentiable approximations techniques for discrete nodes in neural networks [116][77]. As above, if $G_l(\mathbf{x}_{l-1}) = 1$, then the l-th layer is suited to handle the input image. If $G_l(\mathbf{x}_{l-1}) = 0$, then the l-th layer is not suited and is bypassed by directly feeding $\mathbf{x}_{l-1}$ into the $(l+1)$-th layer. In contrast to Highway Networks which executes all layers at inference time, the binary gates in AIG allow to execute only the selected layers that are specific to the input image, leading to save computations.

In the same vein, other approaches [74][155] have designed an adaptive network with inter-layer gates, but whose layers are hierarchically distributed in a tree-structured network. These approaches progressively transform the input image at each split node of the tree to both route the network and extract intermediate representations deciphering the task. Traversing the root-to-leaf path in the tree thus allows to select only the transformations specific to the input image. In addition, the hierarchical structure of the tree saves computations better than a flat structure (like AIG).

In all these approaches, it has been empirically validated that using inter-layer gates allows to learn distinct paths for different categories: for example, some layers are specialized for man-made object categories, while others are specialized for animal categories. Inter-layer gates thus allows to learn specialized layers on data subsets. Moreover, Veit et al. have validated the robustness of the gates to additional noise in the data. It can then be assumed that the factors identified as source of important variations in the data don't affect the inference in the graph and don't prevent to select the more specialized layers depending on the input image.

3.3.3 Conclusion

We have presented different adaptive networks that use gates to select or emphasize on the most suited parts of the network given an input image. Intra-layer gates emphasize on the most informative channels and spatial regions of feature maps. Inter-layer gates emphasize on the most specific intermediate representations of the input image.

In these approaches, the gates *sequentially* select the sample-specific transformations to extract the representation. In what follows, we present Mixture-of-Experts, an adaptive deep ensemble model whose gates are used to emphasize on the most sample-specific predictors of the deep ensemble, which are then distributed *in parallel* by sharing the same input representation.

3.4 Mixture-of-Experts

The gating mechanism can also be applied to deep ensemble methods, allowing each base network to be specialized in regions of the input space. The Mixture-of-Experts (MoE) architecture is an instance of a such adaptive deep ensemble method. In what follows, we first present the original MoE architecture. Second, we review several extensions, consisting in stacking several MoE layers which outputs each an intermediate representation.

3.4.1 The MoE architecture

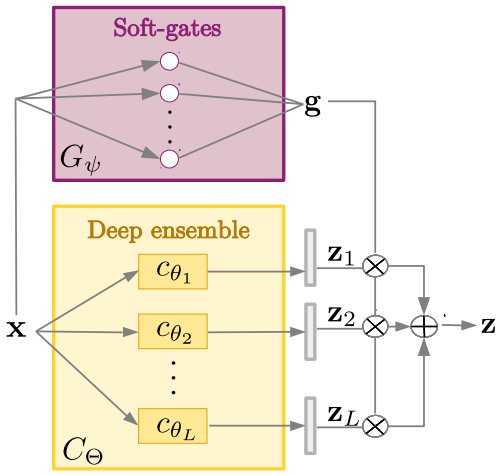


Figure 3.8: The Mixture-of-Experts architecture. A deep neural network ensemble outputs L decisions, then are adaptively weighted by gates given the input $\mathbf{x}$ to generate the final decision. Gates are outputted by a L-dimensional softmax layer.

The MoE architecture was introduced by Jacobs et al. [76][75]. They propose to use the divide-and-conquer principle in which the target task is divided between multiple base predictors, called *experts*, whose outputs are adaptively weighted by gates in order to emphasize on the most relevant experts depending on the input.

In the same vein as selection in the combiner module (described in 3.2.4.2), the objective of MoE is to learn subspaces $\mathcal{X}_1, ..., \mathcal{X}_L$ forming a soft-partition

of the input space $\mathcal{X}$, on which to specialize the experts on each of these regions. But instead of selecting a single expert to generate the final decision, the combiner uses gates, which output a mixture of coefficients allowing to weight each expert prediction. The highest coefficients are then attributed to the most relevant experts.

As illustrated in Figure 3.8, each expert $c_{\theta_l} : \mathbf{x} \in \mathcal{X} \mapsto \mathbf{z}_l \in \mathcal{Z}$ is a neural network with parameters θ_l based on the input representation $\mathbf{x}$ and generating a decision $\mathbf{z}_l$ (e.g. logits vector for classification). It should be noticed that all the experts share the same input.

The gates $\mathbf{g}$ are outputted by a gating function $G_\psi : \mathbf{x} \in \mathcal{X} \mapsto \mathbf{g} \in [0,1]^L$ with parameters ψ and such that $\Sigma_{l=1}^{L} \mathbf{g}_l = 1$. The l-th gate $\mathbf{g}_l$ can be interpreted as the probability that the expert c_{θ_l} is the most relevant expert to evaluate the input $\mathbf{x}$. The gating function G_ψ is modeled by L-dimensional softmax layer. The weight $\mathbf{g}_l$ of the expert c_{θ_l} is then written as:

$$\mathbf{g}_l = \frac{e^{\mathbf{w}_l.\mathbf{x}+b_l}}{\Sigma_{l=1}^{L} e^{\mathbf{w}_l.\mathbf{x}+b_l}} \tag{3.15}$$

where $\psi = \{\mathbf{w}_l, b_l\}_{l=1,...L}$ are the trainable parameters. From a geometric view, the vector $\mathbf{w}_l$ define a hyperplan in the input space $\mathcal{X}$, thus delimiting the region of specialization of the l-th expert.

The final decision $\mathbf{z}$ can then be calculated in several ways:

- Winner-takes-all: the final decision is that of the expert with the most weight in the mixture.

  $$\mathbf{z} = \mathbf{z}_{l^*} \tag{3.16}$$

  where $l^* = \arg\max_{l=1}^{L} \mathbf{g}_l$

- Stochastic selection: the final decision is randomly selected among the experts. Each expert decision $\mathbf{z}_l$ has a probability $\mathbf{g}_l$ to be chosen.

  $$\mathbf{z} = \mathbf{z}_U \tag{3.17}$$

  where $U \sim MultiNomial(\mathbf{g}_1, ..., \mathbf{g}_L)$

- Weighting: the final decision is the average of the expert decisions weighted by the gates. This corresponds to the expectation of stochastic selection.

  $$\mathbf{z} = \Sigma_{l=1}^{L} \mathbf{g}_l \mathbf{z}_l \tag{3.18}$$

In the original paper, the training is carried out in an alternative way by the EM algorithm [38]: given the ensemble of expert networks, train the gating network; then for the pretrained gating network, train the expert networks according to the mixture specified by the gates; and so on. However, MoE can also be seen as a single neural network with a complex structure. It is then possible to train it with the gradient-based optimization techniques usually used for deep neural networks.

3.4.2 Stacked MoE

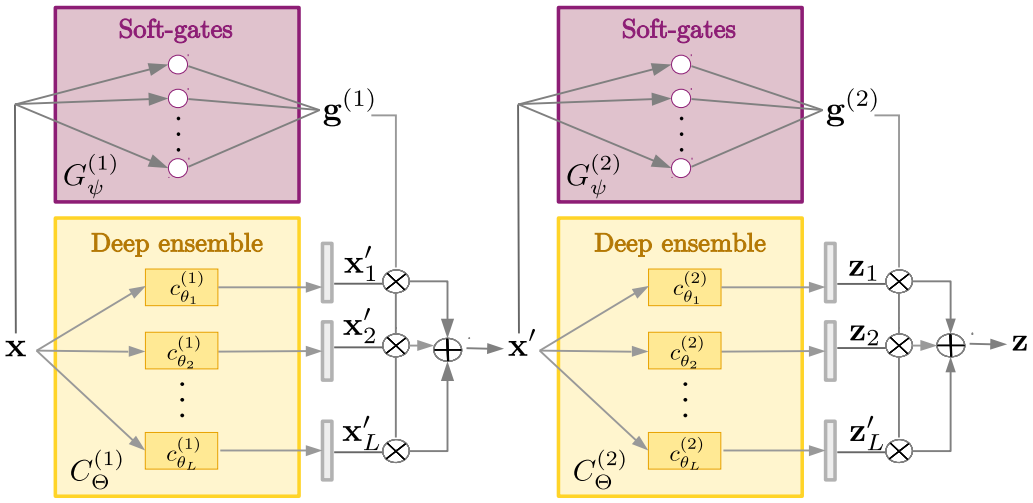


Figure 3.9: Stacked MoE with two layers [46].

In the above, MoE is considered as a layer modeling the mapping between a high-level representation and the ground truth. However, it is possible to stack several MoE layers, where each MoE layer generates an intermediate representation.

Eigen et al. [46] were the first to propose such an approach, by stacking two MoE layers, as illustrated in Figure 3.9. The model can then represent an exponential number of effective experts through all the different combinations of experts at each layer. Indeed, different inputs generate different paths in the network, exponentially multiplying the number of possible combinations with the depth of the network.

Shazeer et al. [147] introduce sparsity in stacked MoE by keeping the top-k values in the logits vector $(\mathbf{w}_1.\mathbf{x} + b_1, ..., \mathbf{w}_L.\mathbf{x} + b_L)$ of the Equation 3.15 and set the other values to $-\infty$ so that the corresponding gate is equal to 0. Sparse gates allow to save computation by processing the data $\mathbf{x}$ only for the k most relevant experts, thus providing the ability to use a large number of experts ($L > 1000$ experts per layer in their experiments) and to increase the capacity of representation.

Very recently, Wang et al. [171] propose to extend the MoE architecture to convolutional networks, with another gating variable than the input $\mathbf{x}$ used by the experts. In particular, they use a single embedding which acts as gating variable for all model layers. This embedding is extracted by an auxiliary CNN based on the image input and jointly learned with the entire system. It allows to learn how to improve the specialization of the experts in a more suited gating space. However, it is very challenging to learn from scratch such an embedding. They propose then to add an intermediate supervision loss to encourage this embedding to predict the outcome variable $\mathbf{y}$ and to maximize its semantic level, a method often used to increase the discriminative power of the lower layers of a DNN [154][67]. However, in order to be robust to an exogenous variable identified as an important source of variations in the data, we argue that it is more relevant to use a gating variable that predicts this exogenous variable rather than the outcome variable. As we will see in Section 3.5, using this exogenous variable as a gating variable also provides the ability to design learning algorithms removing this exogenous information in the expert networks, and leading to increase the robustness to the most extreme variations in this exogenous variable.

3.4.3 Conclusion

Instead of using a single strong network that has to adapt to all possible variations in the input space $\mathcal{X}$, MoE allows to jointly learn a soft-partition of $\mathcal{X}$ and the base networks specialized in each region. The gates then allow to emphasize on the most specialized base networks according to the region where the input is located, leading to improve the overall robustness. Moreover, stacking several MoE layers allows to learn intermediate representations by using an intra-layer adaptive network which selects the most specific transformations according to the region where the input is located. Finally, it is possible to use another gating variable aiming to better specialize the experts in a more suited gating space.

3.5 Adaptive deep ensemble methods

So far, we have presented deep ensemble methods allowing to jointly learn complex representation and prediction functions (by using deep learning techniques), while being robust to large variations in unconstrained environments (by using ensemble methods). Adaptive networks have also been reviewed to emphasize on certain parts of the network conditioned on the input, leading to a better specialization of the model to large variations. Several components can then be used to develop the intended robust face analysis system: deep ensemble and adaptive components. A method combining these components is Mixture-of-Experts (MoE) which uses an ensemble of base networks for the deep ensemble component, and gates for the adaptive one. The ensemble accuracy can be improved by a better cooperation between the base networks. This can be handled either by (1) the gating structure, (2) the gating variable conditioning the task, or (3) the learning algorithm.

For (1), we propose tree-structured gates (described in Section 3.5.1) in order to learn a hierarchical clustering of the base networks, leading to a more efficient selection and therefore specialization of them. For (2), we propose to embed an exogenous variable (described in Section 3.5.2), identified as an important source of variation, in order to better condition the target task and to better specialize the base networks. Thus, we separate exogenous and endogenous representations: the first captures the exogenous variable while the second deciphers the target task. For (3), we propose to disentangle these representations (described in Section 3.5.3) in order to encourage the removal of exogenous information from the endogenous representation by using a new training loss.

To sum it up, the main contributions of our approach are thus three-folds:

- From an architectural standpoint, we propose a new adaptive deep architecture using (1) a hierarchical gating structure to learn more efficiently input space clusters, and (2) a gating variable exogenous to the target task to better condition it and learn base networks more robust to these exogenous variations.

- From a learning standpoint, we propose a new training loss encouraging to remove the exogenous information from the endogenous representation, further improving the overall learning algorithm and the robustness of base networks to exogenous variations.

- From an experimental standpoint, we propose generic methods that can be applied to multiple layers (e.g. prediction, representation layer) and multiple predictive tasks (e.g. face alignment, facial expression recognition). We experimentally validate our approach on synthetic and realistic datasets. In particular, we show that our method significantly improves the robustness to large variations.

3.5.1 Tree-gated MoE

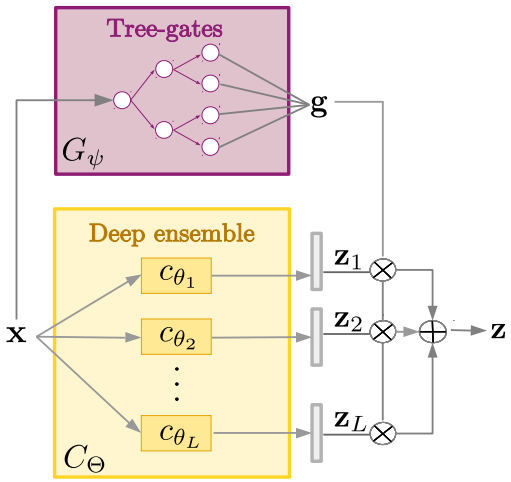


Figure 3.10: Tree-gated MoE. A deep neural network ensemble outputs L decisions, then are weighted by tree-gates depending on the input $\mathbf{x}$. Tree-gates correspond to the leaf probabilities of a $\log_2(L)$-deep neural tree.

In the MoE architecture (described in section 3.4), the gates correspond to a L-dimensional vector $\mathbf{g} = (\mathbf{g}_1, ..., \mathbf{g}_L)$ such that $\Sigma_{l=1}^{L} \mathbf{g}_l = 1$, where $\mathbf{g}_l$ can be interpreted as the probability that the l-th base network is the most relevant to evaluate the input $\mathbf{x}$. These gates, that we called soft-gates, are outputted by a L-dimensional softmax layer based on the input $\mathbf{x}$.

The objective is thus to learn a soft-partition of the input space:

$$\mathcal{X} = \bigcup_{l=1}^{L} \mathcal{X}_l$$

with $\mathcal{X}_l$ the region where the l-th base network is specialized. However, an overlap between the L regions can occur, thus limiting the diversity of

specialization of each base network. To address this issue, we propose to use a neural tree (described in section 3.2.5), as illustrated in Figure 3.10. Indeed, the tree structure allows to learn a hierarchical partition of $\mathcal{X}$ (i.e. $\bigcap_{l=1}^{L} \mathcal{X}_l = \emptyset$). In addition, the differentiability of a neural tree allows to learn the partition jointly with the whole system.

The region $\mathcal{X}_l$ where the l-th base network is specialized then corresponds to the l-th leaf of the neural tree. We then define tree-gates, defined as the concatenation of the $L = 2^{\mathcal{D}}$ leaf probabilities of a single neural tree of depth $\mathcal{D}$. Thus, if an input $\mathbf{x}$ is in the $\mathcal{X}_l$ region, the specialized base network c_{θ_l} has the most weight in the final decision, and others from the most remote regions have the least weights. Moreover, tree-gates allows a hierarchical pooling of the base networks over large regions of the input space $\mathcal{X}$, as follows:

$$\mathbf{z}^n = d_n(\mathbf{x}).\mathbf{z}^n_+ + (1 - d_n(\mathbf{x})).\mathbf{z}^n_- \tag{3.19}$$

where $d_n(\mathbf{x})$ the probability to reach the left subtree at split node n given $\mathbf{x}$ and $\mathbf{z}^n$, $\mathbf{z}^n_+$, $\mathbf{z}^n_-$ are the decisions respectively outputted by the n-th split node, the left and right subtrees. Tree-gates allows then to learn a hierarchical clustering of base networks and can be used to interpret the decisions at different levels of the tree.

The core idea of a hierarchical gating structure to weight the base network predictions has been introduced in [80]. However, the learning algorithm used (EM algorithm) constrains the tree-gated MoE architecture to be applied only to the prediction layer. Similar to the soft-gated MoE architecture which has been extended to other layers of a deep architecture (described in section 3.4.2), we propose to extend tree-gated MoE to other layers of the model, whose learning algorithm uses the latest gradient-based optimization techniques for end-to-end training.

Figure 3.11 illustrates the differences between soft-gates and tree-gates on MOON [131]. At each epoch of the training, we can see the region $\mathcal{X}_l$ where the l-th base predictor c_{θ_l} is specialized (yellow surface, i.e. when the corresponding gate $\mathbf{g}_l$ is close to 1), as well as its prediction $\mathbf{z}_l$ in the input space $\mathcal{X}$ (red/blue surfaces).

By using tree-gates, we can observe at the first epoch that c_{θ_2} and c_{θ_4} first partition the space horizontally by comparing $\mathbf{g}_2$ and $\mathbf{g}_4$. At the second epoch, c_{θ_4} then cooperates with c_{θ_3} by subdividing the half-high region. Finally, c_{θ_2} specializes to the right of the half-low region and lets c_{θ_3} separating the red/blue points to the left of this region.

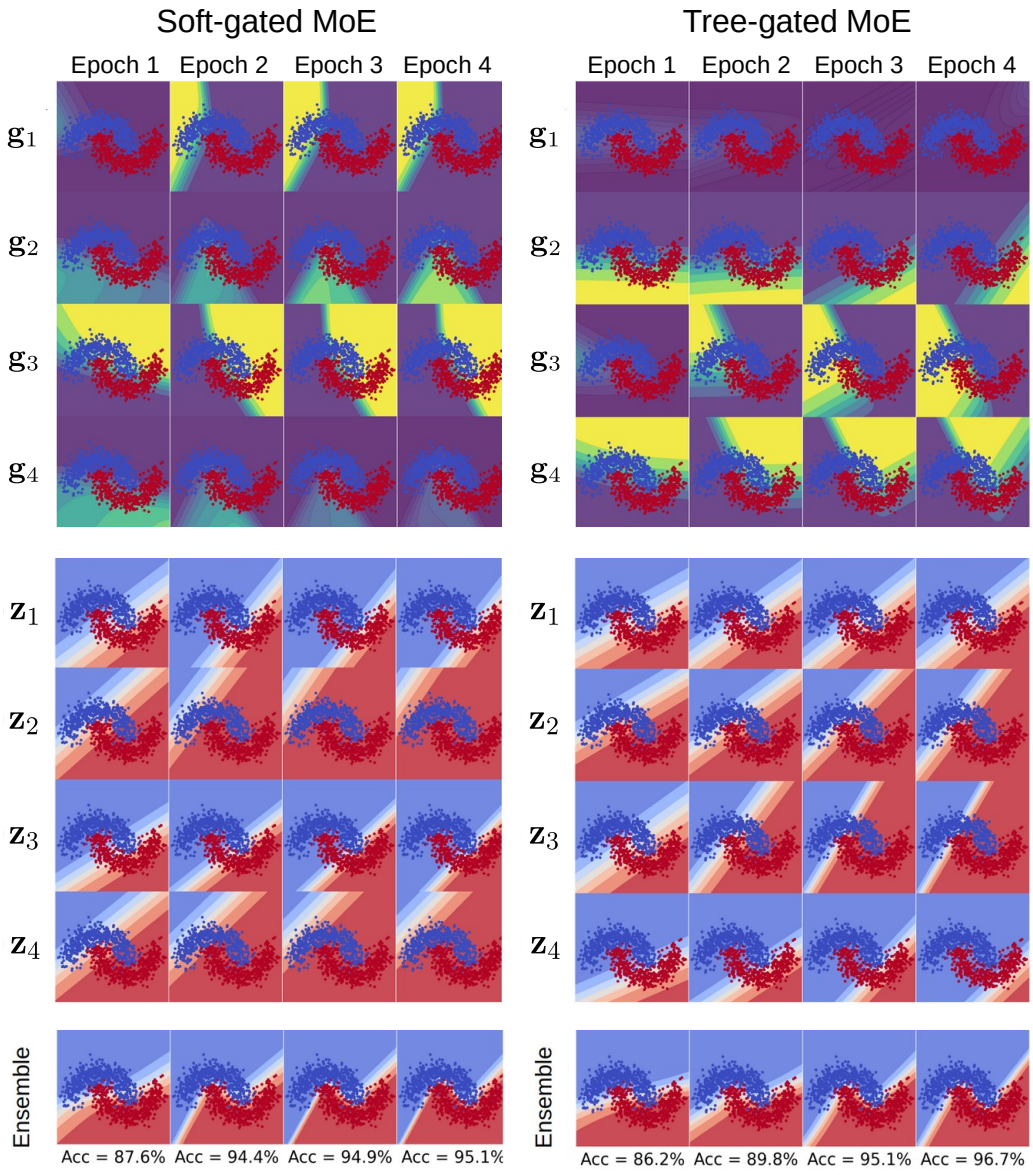


Figure 3.11: Visualization of MoE during training depending on the gating structure. Left: soft-gates. Right: tree-gates. The first four lines show the regions of specialization of each base network. The next four lines show the predictions of each base network on the input space. The last line shows the ensemble predictions and the accuracy obtained at the end of each epoch.

By using soft-gates, we can see in the first epoch that c_{θ_2} and c_{θ_4} share the same region of specialization. Although this region tends to be attributed to c_{θ_2} on the latest epochs, soft-gates struggle to avoid overlap between regions of specialization, unlike tree-gates which are sparser by using only three base networks. Due to this overlap, soft-gates limits then the specialization of the base networks, leading to a lower overall accuracy (95.1% with soft-gates instead of 96.7% with tree-gates).

Finally, it can be noted that the single base network separating red/blue points in their region of specialization (i.e. c_{θ_3} in both cases at the last epoch) has a finer decision borders if tree-gates are used. This means that tree-gates have allowed to better specialize this base network by increasing his prediction confidence (i.e. by decreasing the entropy of the class probability distribution).

Therefore, tree-gates have the following advantages compared to soft-gates: first, tree-gates allow to better distribute the base networks in the input space by avoiding overlaps between their regions of specialization. Second, tree-gates are sparser, allowing to restrict the number of base networks. Third, tree-gates allow to better specialize the base networks with more confidence in their decisions.

The hierarchical gating structure is thus a first approach to improve the cooperation between the base networks and ensemble accuracy. As we discuss in the following, this can be strengthened by the use of a more suitable gating variable.

3.5.2 Exogenous tree-gated MoE

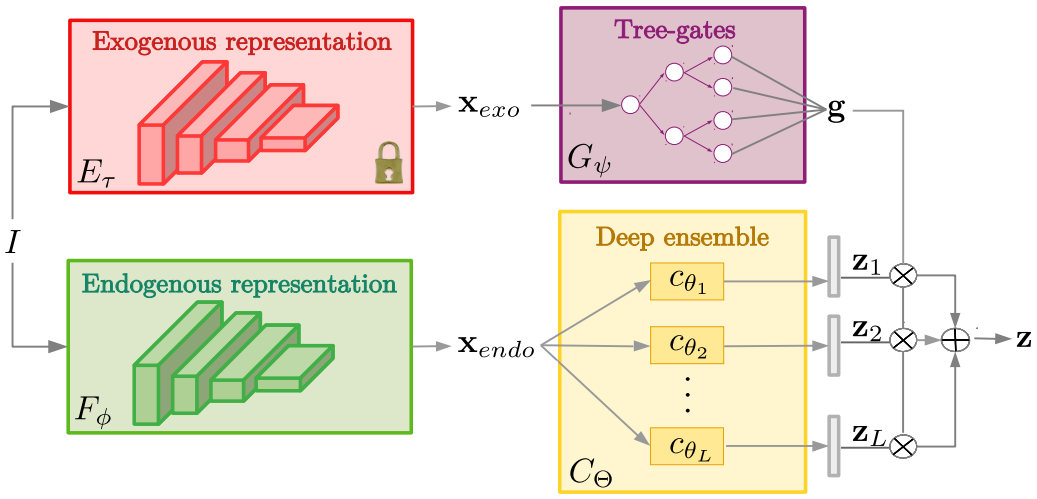


Figure 3.12: Exogenous tree-gated MoE. The exogenous representation captures an exogenous variable, which is (1) identified as an important source of variations in the data and (2) modeled with sufficient accuracy. It acts as a better gating variable than the endogenous representation deciphering the task.

By jointly learning a partition of the input space and the base networks specialized in each region, tree-gated MoE allows the model to better adapt to unconstrained environments by emphasizing on the most specialized base networks depending on the input. The input is thus used twice: as a discriminative variable to decipher the task through the base networks, and as a gating variable to select the most relevant base networks depending the region to which it belongs.

However, some exogenous variations can corrupt the input $\mathbf{x}$ and push it into a wrong region $\mathcal{X}_l$, leading to the use of an inappropriate base network. Figure 3.13 shows two predictive tasks where an exogenous variable can be identified as an important source of variation. In Figure 3.13 - left, the rotation can greatly influence the appearance to the digit. For the most extreme rotations (close to $\pm 90°$), digit features can thus vary greatly from one region to another, leading to an inefficient use of the base networks. In the same vein, in Figure 3.13 - right, the scale can influence the shape recognition by strongly varying the shape features.

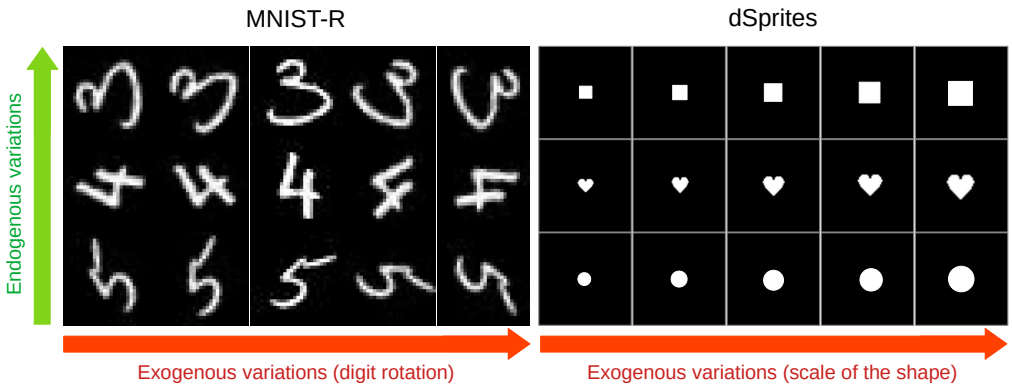


Figure 3.13: Examples of predictive task where an exogenous variable can be identified. **Left**: digit recognition with rotation as the exogenous variable. **Right**: shape recognition with the scale. The endogenous variable deciphering the task is sensitive to variations in the exogenous variable.

To address this issue, we propose to distinguish the discriminative and the gating variables to specialize the base networks in a more adapted gating space, as illustrated in Figure 3.12. In particular, if an exogenous variable can be (1) identified as an important source of variation in the data, and (2) modeled with sufficient accuracy, we argue that its representation is an ideal gating variable. We then call it **exogenous** representation. In this way, each base network is specialized in a region of the exogenous representation space, allowing the model to better adapt to the most extreme exogenous variations. Moreover, the discriminative variable has less constraint to capture exogenous variations (already handled by gates), leading to improve the discriminative power of its representation. We then call it **endogenous** representation.

To model the exogenous variable, we train a deep neural network $e_\sigma(E_\tau(\mathbf{x}))$ where e_σ is the exogenous predictor with parameters σ and using the exogenous representation $\mathbf{x}_{exo}$ outputted by the network E_τ with parameters τ and based on the input $\mathbf{x}$. If the input $\mathbf{x}$ is the image, the exogenous model is firstly trained, then the parameters τ are kept frozen so that the exogenous representation $\mathbf{x}_{exo}$ can then be integrated within another deep neural network. In addition, it allows to train the model on a separate trainset from the one used for the target task, thus removing the constraint of having the ground truth annotations for both the exogenous variable and the labels of

Table 3.1: Accuracies of the exogenous classifiers. For Rotation and Scale, the range is $[-90°, 90°]$ and $[0.5, 1]$, respectively.

Exogenous variable	Accuracy
Rotation (MNIST-R)	3.020
Scale (dSprites)	0.0291

the target task.

We have experimentally validated this approach for two predictive tasks where an exogenous variable verifies conditions (1) and (2).

Digit classification under rotation on the MNIST-Rotated or MNIST-R database. From the well-known digit handwritten digit MNIST database [97], we have augmented each image by a random rotation from $-90°$ to $90°$ (18 rotation classes with bins of $10°$). In this case, the rotation can then be identified as an exogenous variable to the digit recognition. We use $60k$ samples for training and $10k$ for testing both the digit classifier and the rotation classifier.

Shape recognition on the dSprites database [120] is commonly used for learning disentangled representations. Each sample is a grayscale image containing a shape (heart, ellipse, square) generated by 5 independent factors (scale, color, rotation, x and y positions). We have annotated the scales with 10 different classes, by bins of 0.05 from 0.5 to 1. We have then defined the scale as the exogenous variable so as to select $60k$ samples for training and $10k$ for testing both the shape classifier and the scale classifier.

Implementation details. For both MNIST-R and dSprites, the exogenous network e_τ consists in 2 convolutional layers $16@3 \times 3$ with max-pooling and ReLU activation, and the classification layer c_σ consists in 2 fully-connected layers containing a hidden layer of 256 units (also with ReLU activation). With this configuration, the exogenous classifier can predict the exogenous variable with a high accuracy, as reported by Table 3.1. Conditions (1) and (2) are thus verified: rotation (resp. scale) is an exogenous variable to digit (resp. shape) recognition greatly affecting the digit (resp. shape) appearance, and the rotation (resp. scale) classifier is accurate enough to integrate it into the digit- (resp. scale-) classifier. Then, the endogenous network F_ϕ also consists in 2 convolutional layers $16@3 \times 3$. The deep ensemble C_Θ consists in $L = 8$ base networks, where each base classifier c_{θ_l} contains a hidden layer of 32 units. For tree-gates, the neural tree G_ψ is thus of depth

Table 3.2: Comparison of tree-gated MoE methods in term of average accuracy (%) on MNIST-R and dSprites databases. †: oracle classifier (i.e. exogenous tree-gated MoE conditionned by the ground truth exogenous variable).

Method	MNIST-R	dSprites
Tree-gated MoE	97.31	95.91
Exogenous tree-gated MoE	**98.07**	**98.1**
Oracle†	98.06	98.43

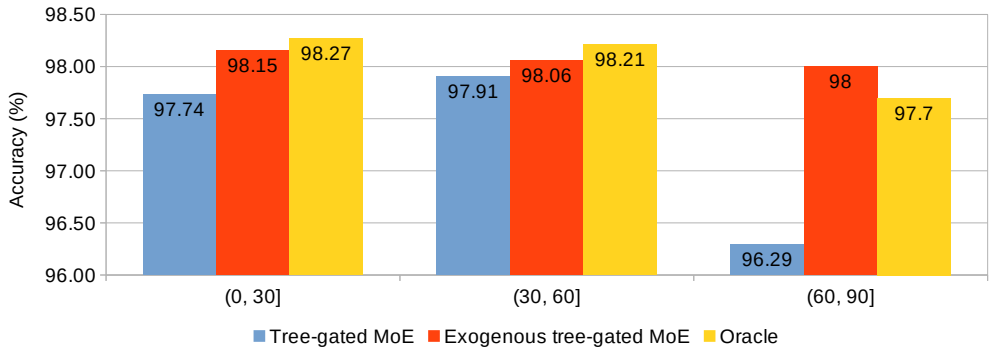


Figure 3.14: Comparison of tree-gated MoE methods in term of average accuracy (%) by absolute rotation variation on MNIST-R.

$\log_2(8) = 3$. Training is done by optimizing the standard cross-entropy over the parameters for endogenous representation ϕ, classification Θ and gating ψ layers. These parameters are trained jointly in an end-to-end manner by applying ADAM optimizer [87] with a learning rate of $1e^{-3}$ and batch size 32.

Evaluation. From a quantitative standpoint, we compare the accuracy differences according to the gating variable used on both MNIST-R and dSprites in Table 3.2. The exogenous variable thus appears as a better gating variable than the endogenous one, by improving the average accuracy by +2.3% on dSprites, and +0.8% on MNIST-R. Figure 3.14 also compares the accuracies according to the intensity of variation of the exogenous variable in the case of MNIST-R with three ranges of rotation variation. Using rotation as gating variable allows then to improve the overall robustness, especially for the most extreme rotation variations where the accuracy gap is strongly increased. Thus, by specializing the base networks in the exogenous repre-

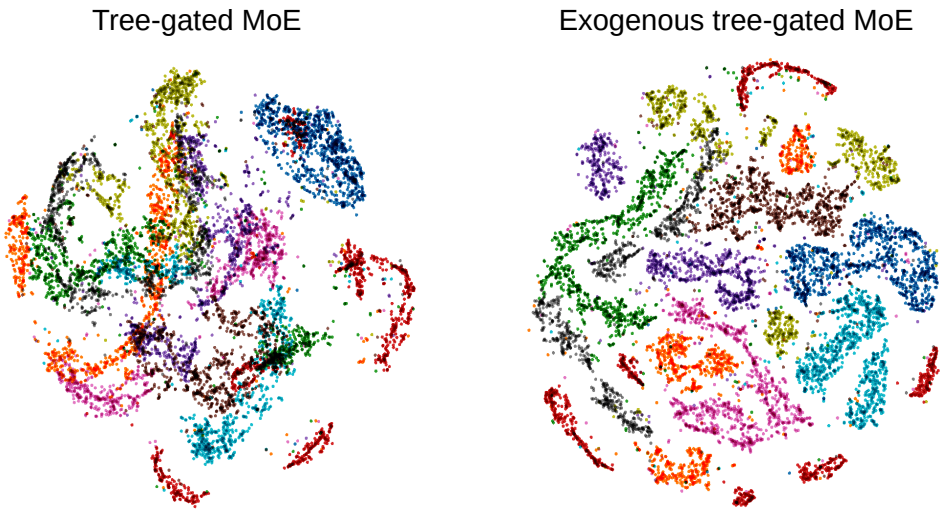


Figure 3.15: Visualization of the endogenous representation on MNIST-R.

sentation space instead of the endogenous one, they are much more adapted to the extreme variations in the exogenous variable. It can also be observed that the performance of exogenous tree-gated MoE is close to the oracle classifier (i.e. an exogenous tree-gated MoE with the ground truth exogenous variable) which can be considered as a ceiling in this experiment.

From a qualitative standpoint, we use t-SNE [114] to visualize the class distribution of MNIST-R in the 2d-projection of the endogenous representation (Figure 3.15). We can then observe that using exogenous representation as gating variable allows a clearer distribution of the different digit classes in the features space. This highlights that the endogenous representation has less the constraint of capturing exogenous variations (which are already handled by gates), leading to improve its discriminative power and the overall performance.

Therefore, defining the gating variable as an exogenous variable identified as an important source of variation has the following advantages: first, it allows to better specialize the base networks on a more suitable gating space. Second, it allows to better adapt them to the most extreme variations in the exogenous variable. Last but not least, it improves the discriminative power of the endogenous representation. As we discuss in the following, this can be strengthened by a more suitable learning algorithm aiming to remove undesirable exogenous-related information from the endogenous representation.

3.5.3 THrowable Information Networks

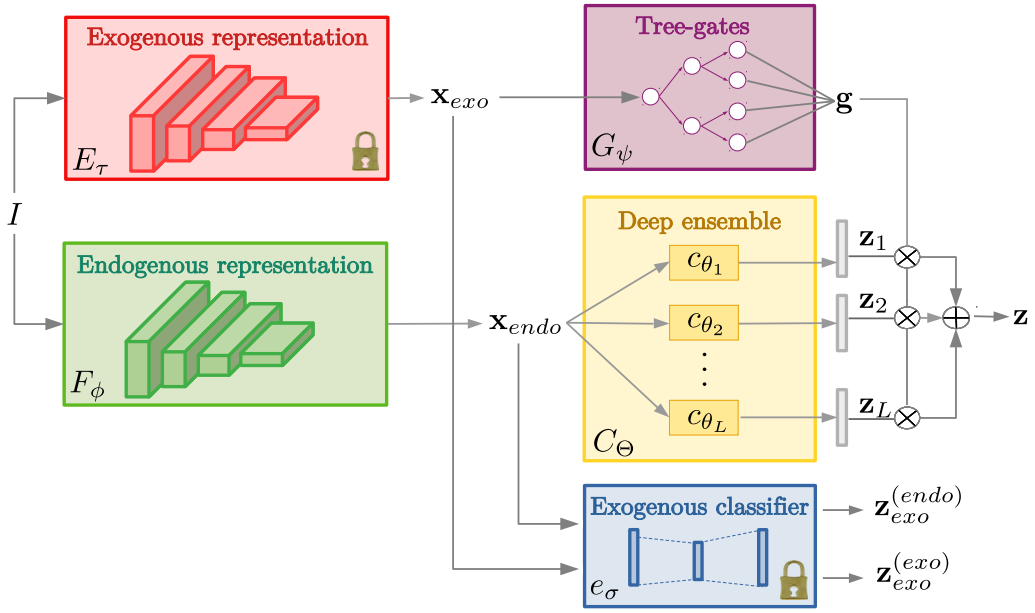


Figure 3.16: THIN. The exogenous variable is used twice in a throwable fashion: (1) as a gating variable to condition the task, and (2) as an anchor to remove exogenous-related information from the endogenous representation.

Although the exogenous variable appears to be a better gating variable to specialize the base networks, exogenous-related information may be retained in the endogenous variable. A strong variation of the exogenous variable can thus lead to a strong variation of the endogenous variable. Ideally, the prediction of each base network should be invariant to the variations of the exogenous variable. We then aim to encourage the removal of exogenous information from the endogenous representation $\mathbf{x}_{endo}$, by throwing from it any discriminatory power predicting the exogenous variable.

Suppose that the exogenous network e_τ and the endogenous network F_ϕ have the same architecture (i.e. $E = F$ but $\tau \neq \phi$). It is then possible to feed $\mathbf{x}_{endo}$ into the exogenous classifier e_σ to generate a prediction $\mathbf{z}_{exo}^{(endo)}$ of the exogenous variable and compare it with the prediction $\mathbf{z}_{exo}^{(exo)}$ obtained with the exogenous representation $\mathbf{x}_{exo}$. If $\mathbf{x}_{endo}$ contains discriminative information predicting the exogenous variable, then $\mathbf{z}_{exo}^{(endo)}$ and $\mathbf{z}_{exo}^{(exo)}$ are similar. This similarity can be measured by their angular distance, defined as follows:

$$\mathcal{L}_{sim} =\mid cos(\mathbf{z}_{exo}^{(exo)}, \mathbf{z}_{exo}^{(endo)}) \mid= \frac{\mid \mathbf{z}_{exo}^{(exo)}.\mathbf{z}_{exo}^{(endo)} \mid}{||\mathbf{z}_{exo}^{(exo)}||_2.||\mathbf{z}_{exo}^{(endo)}||_2} \tag{3.20}$$

In order to remove exogenous-related information from the endogenous representation, $\mathbf{z}_{exo}^{(endo)}$ must be as unsimilar as possible to $\mathbf{z}_{exo}^{(exo)}$. We then propose a new training loss encouraging to *semantically* orthogonalize the exogenous representation $\mathbf{x}_{exo}$ and the endogenous representation $\mathbf{x}_{endo}$ by penalizing their similarity in the the exogenous space modeled by the classifier c_σ, as illustrated in Figure 3.16.

The parameters ϕ of the endogenous representation layer are thus trained to minimize both the loss $\mathcal{L}_{target}$ of the target task in order to learn discriminative features, and the dispelling loss $\mathcal{L}_{sim}$ in order to remove undesirable exogenous-related information. The final loss can then be written as:

$$\mathcal{L}(\Theta, \phi, \psi) = \mathcal{L}_{target}(\Theta, \phi, \psi) + \lambda \mathcal{L}_{sim}(\phi) \tag{3.21}$$

where Θ, ϕ, ψ are the parameters for the deep ensemble, the endogenous representation and the gating layers respectively, and $\lambda > 0$ the penalization coefficient of the dispelling loss whose setting depends on the experiment.

The exogenous variable is thus used twice in a throwable fashion: (1) as a gating variable to condition the task, and (2) as an anchor to remove exogenous-related information from the endogenous representation. We call then this method THIN, standing for THrowable Information Networks.

We have experimentally validated this approach both on MNIST-R and dSprites with the same training settings as before. From a quantitative standpoint, Figure 3.17 shows the accuracies of several THIN models depending on the penalization coefficient λ used in the dispelling loss $\mathcal{L}_{sim}$ during the training. If λ is too strong ($\lambda > 0.01$ in this case), the training is too involved in removing rotation-related information from the digit features rather than improving its discriminative power for the digit classification, thus leading to decrease the overall performance. However, if we set $\lambda = 0.005$ or $\lambda = 0.01$, it is possible to balance these two objectives, leading to increase the overall accuracy by 0.2%.

From a qualitative standpoint, we visualize the class distribution on MNIST-R in the 2d-projection of the endogenous representation (Figure 3.18). We can then observe that encouraging the removal of exogenous information in the endogenous representation allows a slightly better distribution

Table 3.3: Comparison of exogenous tree-gated MoE methods in term of average accuracy (%) on MNIST-R and dSprites databases.

Method	MNIST-R	dSprites
Exogenous tree-gated MoE	98.07	98.1
THIN	**98.26**	**98.5**

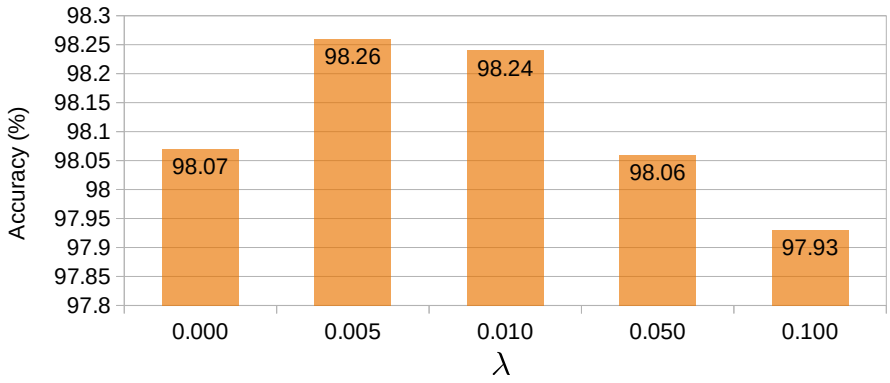


Figure 3.17: Ablation study for the penalization coefficient λ in term of accuracy (%) on MNIST-R.

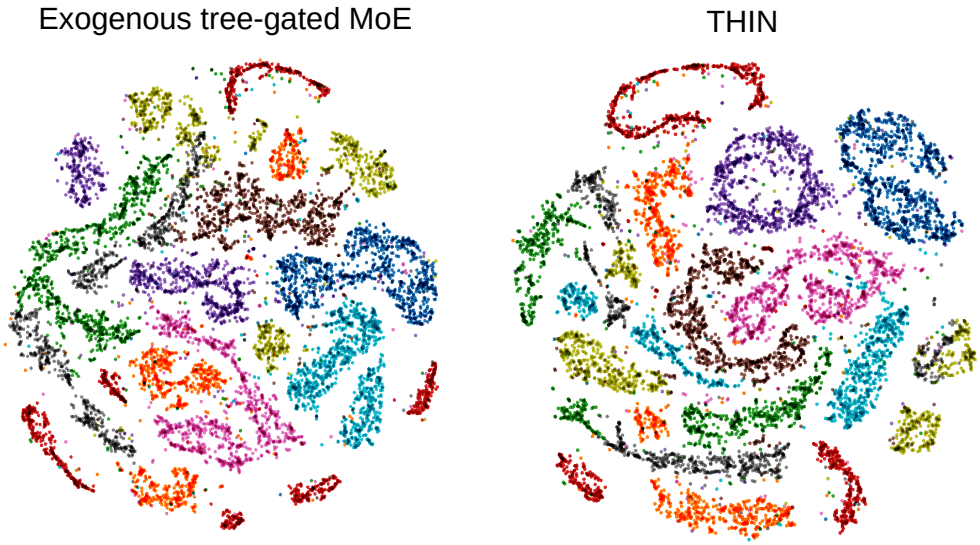


Figure 3.18: Visualization of the endogenous representation on MNIST-R.

of the different digit classes in the features space, indicating a greater discriminative power of the endogenous representation.

Therefore, our training loss has the following advantage: first, it removes undesirable exogenous-related information from the endogenous representation, limiting the endogenous representation variations induced by exogenous variations. Second, it improves the discriminative power of the endogenous representation.

3.5.4 Conclusion

In order to improve the robustness to large variations, we have presented a new adaptive deep ensemble method inspired by the Mixture-of-Experts: an ensemble of base networks whose decisions are adaptively weighted by a gating variable that specializes each base network on a region of the gating space.

First, we propose a hierarchical gating structure in order to learn a hierarchical partition of the gating space to improve the specialization of the base networks both in their distribution in the gating space and in their prediction confidence. Second, instead of using the endogenous representation of the task as a gating variable (i.e. like MoE), we propose to use the representation of an exogenous variable, identified as an important source of variation and modeled with sufficient accuracy. Thus, the base networks are specialized in order to be able to adapt to the most extreme variations of this exogenous variable. Finally, we propose a new training loss to remove undesirable exogenous-related information from the endogenous representation. Thus, it reduces the sensitivity of the endogenous variable to variations in the exogenous variable and improves its discriminative power.

In this chapter, we have illustrated and experimentally validated our approach in the prediction layer for predictive tasks that are relatively easy to model. As we will see in the next chapter, our method can be applied to other layers in the model (e.g. representation layer [5]), and to other tasks much more complex to model (e.g. face alignment [6][5], facial expression recognition [7]), thus proving the relevance of our approach in real in-the-wild conditions.

Chapter 4

Applications

In order to develop a face analysis system robust in unconstrained environments containing large variations, our framework presented in Section 3.5 can be used to model each task of the system (i.e. face alignment, facial expression recognition) and can be applied to each of the layers constituting the models (i.e. representation, prediction). As illustrated in Figure 4.1, several variables can be identified as important source of exogenous variations. For face alignment (left), head pose variations affect the face appearance regardless of the face shape, while being exogenous to it. For facial expression recognition (right), identity variations affect the face appearance regardless of the facial expression, while being exogenous to it. We then propose to explicitly use these exogenous variables to better adapt the system to their most extreme variations.

In this chapter, we experimentally validate the genericity of our approach through quantitative and qualitative evaluations on the most recent and challenging databases. In Section 4.1, we apply our framework to face alignment [6][5]. We first show that adaptive deep ensemble methods significantly increase the overall robustness of the face alignment model. In particular, using head pose estimation as gating variable in the representation layer further allows to extract specific features for each pose cluster, leading the model to better adapt to the most extreme variations in pose. In Section 4.2, our framework is applied to facial expression recognition [7]. In particular, we argue that identity is an ideal gating variable to increase the robustness of the model. Finally, we validate that a better representation of facial expression can be learned by using our new dispelling loss to remove identity-related information from it, leading to increase the overall robustness.

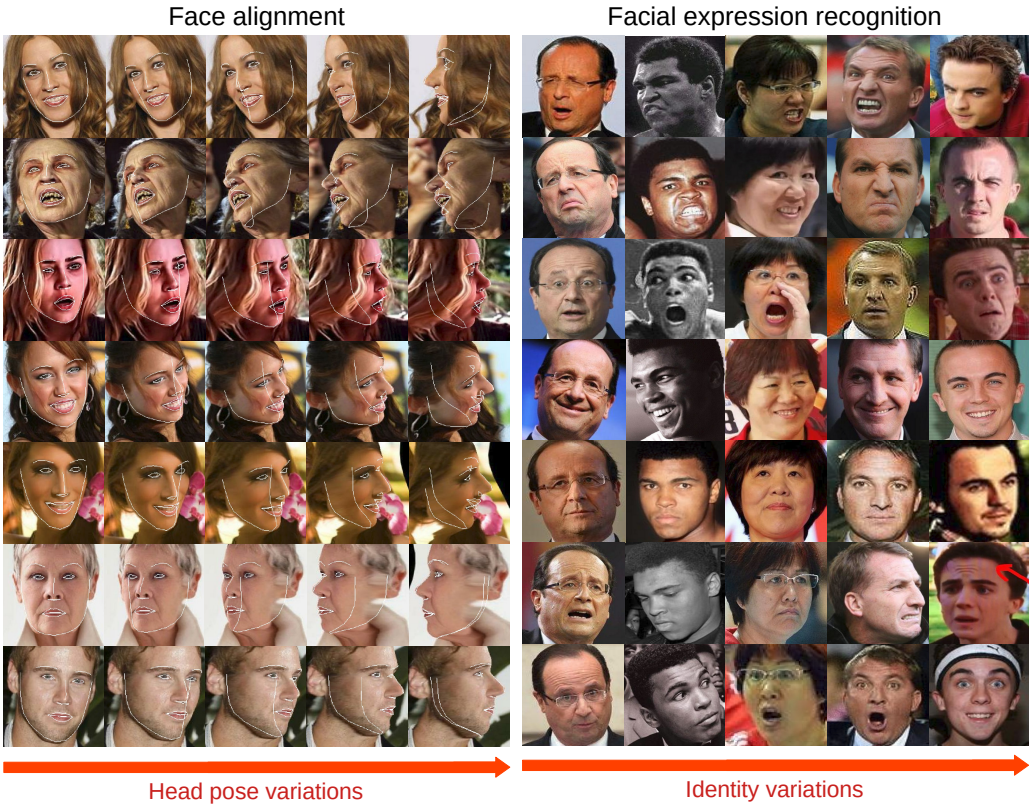


Figure 4.1: Exogenous variations in several face analysis tasks. **Face alignment**: Head pose variations greatly affect the face appearance regardless of the face shape, while being exogenous to it. Images from 300W-LP [199]. **Facial expression recognition**: Identity variations greatly affect the face appearance regardless of the facial expression, while being exogenous to it. Images from VGGFace2 [23].

4.1 Face alignment

As presented in Section 2.1, face alignment aims to locate P facial landmarks that form the face shape $\mathbf{y} \in \mathbb{R}^{2\times P}$ (the p-th column of this matrix corresponds to the 2D-coordinates of the p-th facial landmark). This is a regression problem that we address in a cascade fashion, as is often done in literature: from an image $\mathcal{I}$ and an initial shape $\mathbf{y}^{(0)}$ (typically the mean shape), the objective is to sequentially learn a cascade of regression mapping functions $R_t : (\mathcal{I}, \mathbf{y}^{(t-1)}) \rightarrow \Delta\mathbf{y}^{(t)}$ between the shape-indexed local appearances and remaining displacements for each cascade stage t. We then propose to apply our approach to model each regression mapping function of the cascade: the datasets used are presented in Section 4.1.1 and the methodology as well as the experimental validations are described in Section 4.1.2. In addition, as shown in Figure 4.2 (middle), head pose can be identified as an important source of variation in face alignment datasets. Head pose estimation is then an ideal gating variable to condition landmark localization and thus better adapt the model to variations at large poses, as detailed in Section 4.1.3.

4.1.1 Datasets

Several databases can be used to train and test the face alignment models. As illustrated in Figure 4.2, they can be divided into three categories: those containing moderate variations, head pose variations, or partial occlusions.

Moderate variations. The 300W database [140] is the most commonly used to train and test face alignment models. It contains three datasets with moderate variations in head pose, occlusions, illuminations or facial expressions: **LFPW** (811 images for training, 224 images for testing), **HELEN** (2000 images for training, 330 images for testing), **AFW** (337 images for training). This represents a total of 3148 images for training and 554 for testing. Each image of 300W is annotated with the 2D-coordinates of 68 facial landmarks.

Head pose variations. The 300W database contains a fourth dataset: **I-BUG** (135 images for testing), acting as the challenging subset of 300W, with mainly head pose variations and some occlusions. However, training the models by using the 300W database (which contains moderate variations) greatly limits the generalization capability on large head pose range.

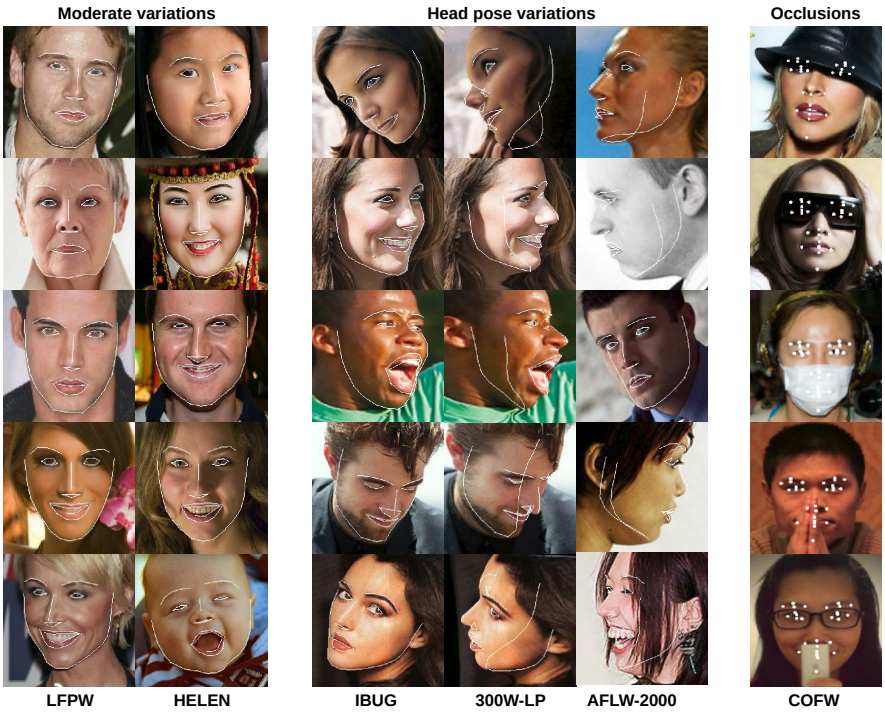


Figure 4.2: Face alignment datasets. For clarity, the 29 facial landmarks of COFW are displayed as dots and the 68 ones of the other datasets are linked.

To address this issue, Zhu et al. [199] produced the **300W-LP** database: for each image in 300W, they generated head pose variations, ranging from $-90°$ to $+90°$ along the yaw axis. It allows to generate a total of 61225 images, leading to learn pose-robust face alignment models. For real large head pose variations, the **AFLW2000-3D** dataset contains the first 2000 images of the Annotated Facial Landmarks in-the-Wild database [89]. This dataset is divided into three head pose ranges, depending on absolute degree yaw: 1306 samples in the $[0, 30]$ range, 462 examples in the $[30, 60]$ range and 232 examples in the $[60, 90]$ range. Each image of 300W-LP and AFLW2000-3D are annotated with the *real* 2D-coordinates of 68 facial landmarks: the self-occluded landmarks (due to extreme head pose) are not projected on the visible part of the face contrary to 300W, as illustrated by the annotation differences between the 3-th and 4-th columns in Figure 4.2. As done in the literature [200], all images from 300W-LP are used for training, and those of AFLW2000-3D for testing.

Occlusions. The **COFW** or Caltech Occluded Faces in the Wild dataset [20] is commonly used to learn face alignment models robust to partial occlusions. It contains only occluded face images with moderate variations in head pose, as illustrated in Figure 4.1.1 (right). It contains 500 images for training and 507 images for testing. Each image is annotated with the 2D-coordinates of 29 facial landmarks. In what follows, we use only the COFW testet to test the robustness of the models trained on 300W and predicting 68 landmarks. As done in [55], we perform a linear mapping between the 68 predicted landmarks to the 29 landmarks.

4.1.2 Tree-gated MoE in the prediction layer

In order to model the regression mapping functions of each cascade stage, we propose to apply our framework described in Section 3.5. A first instance is to apply the tree-gated MoE (Section 3.5.1) in the prediction layer. We then define several models that incrementally add the architectural components of the model (i.e. deep ensemble, adaptive gates), as well as implementation details to ensure reproducibility of the results (i.e. evaluation metric, training hyperparameters). Second, we compare the architectures with each other and with the state-of-the-art approaches, and conduct qualitative evaluations.

4.1.2.1 Methodology

Architectures. The architectures presented below are defined for each cascade stage, whose the different layers are illustrated in Figure 4.3: patches extraction, representation and regression layers.

Given a 150x150-grayscale face image I and initial facial landmark coordinates $\mathbf{y}^{(0)} \in \mathbb{R}^{2\times P}$, we first extract 32x32-patches centered around each landmark to get P local appearances. The pixels are centered so that they take values in $[-1, 1]$.

Second, we feed each of them into a single convolutional network F_ϕ to extract features for each landmark. In particular, it contains 5 strided convolutional layers (20@5x5 $\rightarrow$ 40@5x5 $\rightarrow$ 80@3x3 $\rightarrow$ 160@3x3 $\rightarrow$ 30@1x1) allowing to output 30 features per landmark. By concatenating these shape-indexed local appearance features, we then obtain a vector $\mathbf{x}$ of size $30 \times P$ (2040 with $P = 68$ landmarks) representing the whole face.

Third, several architectures can be defined in the regression layer, as illustrated in Figure 4.3 (bottom):

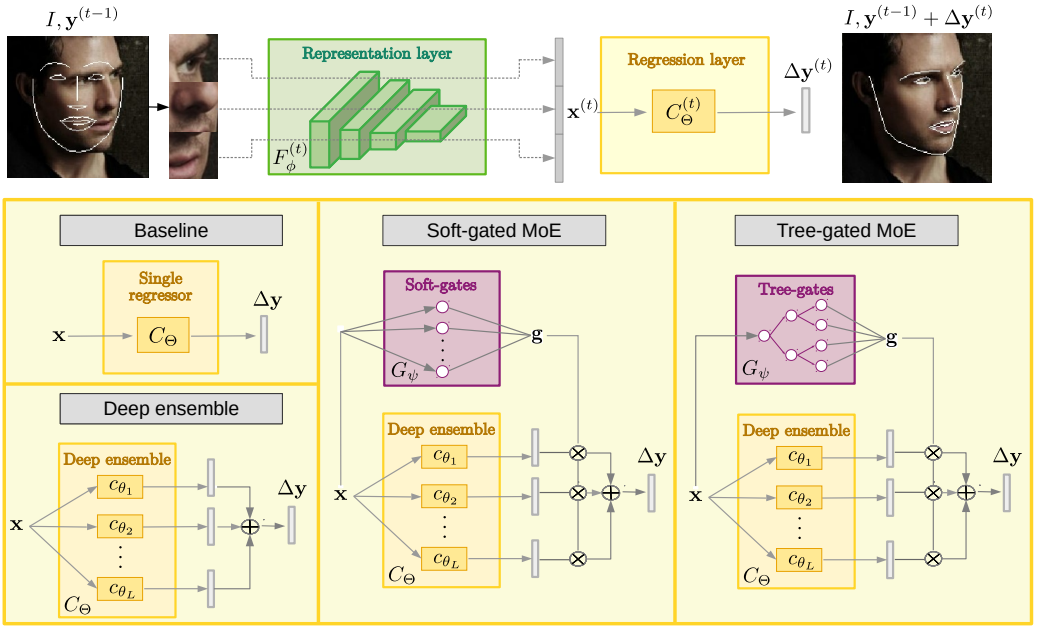


Figure 4.3: Architectures overview for a given cascade stage. **Top**: Several layers transform sequentially the image I from the input shape $\mathbf{y}^{(t-1)}$: patches extraction, representation and regression layers. **Bottom**: The different proposed architectures in the regression layer. For clarity, we omit the t-indexing.

- **Baseline**: a single strong 2-FC regressor C_Θ containing 8192 hidden units.
- **Deep ensemble:** a deep ensemble of $L = 64$ base 2-FC regressors, each containing 128 hidden units. The predictions are equally weighted: $\mathbf{g} = (\frac{1}{L}, \ldots, \frac{1}{L})$.
- **Soft-gated MoE:** a deep ensemble of $L = 64$ base 2-FC regressors, each containing 128 hidden units. The predictions are weighted by soft-gates outputting by a 64-dimensional softmax layer G_ψ.
- **Tree-gated MoE:** a deep ensemble of $L = 64$ base 2-FC regressors, each containing 128 hidden units. The predictions are weighted by tree-gates outputting by a 6-depth neural tree G_ψ ($2^6 = 64$ terminal nodes).

With this configuration, each architecture has roughly the same total number of parameters (18 million parameters total for each cascade stage), allowing a fair comparison between them.

Evaluation metric. Given a dataset containing N samples in the form $(I_i, \mathbf{y}_i)$ where I_i and $\mathbf{y}_i$ are the i-th image and the true landmark coordinates respectively, the common metric used in the literature to evaluate a prediction $\hat{\mathbf{y}}_i$ is the normalized mean error (NME). It corresponds to the average landmark-wise distance between the ground truth and the prediction, normalized by the inter-pupil distance:

$$NME = \frac{1}{N} \sum_{i=1}^{N} \frac{||\hat{\mathbf{y}}_i - \mathbf{y}_i||_2}{||\mathbf{g}_{i,l} - \mathbf{g}_{i,r}||_2} \tag{4.1}$$

where $\mathbf{g}_{i,l}$ and $\mathbf{g}_{i,r}$ are respectively the left and right pupil centers of the i-th face image.

Learning. As often done in the literature, we train 4 cascade stages. For each sample, we augment the initial landmark coordinates (i.e. mean shape) by a random translation $t \sim \mathcal{N}(0, 10)$ and a random scaling $s \sim \mathcal{N}(1, 0.1)$. Training is done by optimizing the NME over the parameters for representation ϕ, regression Θ and gating ψ layers. These parameters are optimized jointly in an end-to-end fashion by using ADAM optimizer [87] with a learning rate of $1e^{-3}$ and batch size 32.

4.1.2.2 Evaluations

Table 4.1: Comparison of different architectures in term of NME (%).

Method	LFPW + HELEN	I-BUG	COFW
Baseline	4.22	8.89	5.9
Deep ensemble	4.06	8.95	5.87
Soft-gated MoE	<u>4.01</u>	<u>8.8</u>	<u>5.84</u>
Tree-gated MoE	**4.01**	**8.38**	**5.76**

Architecture comparison. Table 4.1 shows the accuracy of each architecture obtained according to the category of variations: moderate variations (LFPW + HELEN), head pose variations (I-BUG) and occlusions

(COFW). As these architectures were trained on 300W annotated with the 2D-coordinates landmarks of the visible parts of the face, the 300W-LP and AFLW2000-3D testsets are not used in this experiment. We can then observe that using a deep ensemble of base regressors slightly improves performance (4.22 $\rightarrow$ 4.06 on moderate variations, 5.9 $\rightarrow$ 5.87 on occlusions). The robustness to head pose variations and occlusions is mostly achieved by using gates. Indeed, soft-gates increase the accuracy on all types of variations (4.22 $\rightarrow$ 4.01 on moderate variations, 8.95 $\rightarrow$ 8.8 on head pose variations, and 5.9 $\rightarrow$ 5.84 on occlusions). The hierarchical structure of the tree-gates further improves the robustness of the soft-gates to large variations in pose and occlusion (8.8 $\rightarrow$ 8.38 on head pose variations, 5.84 $\rightarrow$ 5.76 on occlusions). Thus, Tree-gated MoE is a first approach to improve the robustness of a simple regressor in an unconstrained environment containing large variations, such as partial occlusions or head pose variations.

Table 4.2: Comparison with state-of-the-art approaches in term of NME (%).

Method	LFPW + HELEN	I-BUG	COFW
RCPR [20]	6.18	17.3	8.50
SDM [180]	5.57	15.4	7.70
PIFA [82]	5.43	9.98	-
LBF [137]	4.87	11.98	13.7
TCDCN [192]	4.80	8.60	-
CSP-dGNF [37]	4.76	12.00	-
ERCLM [13]	4.58	8.90	-
RCN$^+$ [64]	<u>4.20</u>	**7.78**	-
DRDA [190]	-	10.79	6.46
SFLD [176]	-	-	<u>6.40</u>
Tree-gated MoE	**4.01**	<u>8.38</u>	**5.76**

Comparison with state-of-the-art approaches. Table 4.2 shows a comparison between Tree-gated MoE and other recent state-of-the-art methods for each category of variations. We can then observe that our approach outperforms these methods, except on head pose variations where RCN$^+$ [64] outperforms Tree-gated MoE. This method uses indeed additional data to regularize the training of the model, thus explaining the greater robustness

observed on head pose variations. Finally, our method achieves a new state-of-the-art on the COFW testset, illustrating the relevance of our approach to occlusion robustness.

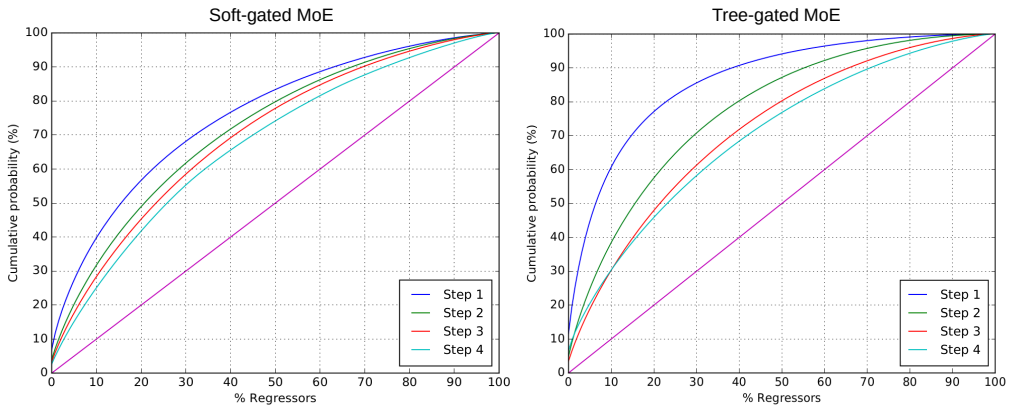


Figure 4.4: Cumulative top-weighted base regressor distribution by using either soft-gates (left) or tree-gates (right).

Qualitative evaluation. In addition to the quantitative evaluation presented above, we conducted other experiments to qualitatively evaluate our approach. Figure 4.4 shows the average cumulative sum of the gate values of the base regressors, sorted in descending order, depending on whether the training was performed with soft-gates (left) or tree-gates (right). This allows to have a measure of the sparsity of the gates, giving an estimate of the average number of base regressors involved in the final prediction. For instance, at the first cascade stage, soft-gates allow 10% of the base regressors to explain 40% of the final prediction, while tree-gates allow 10% of the base regressors to explain 60% of the final prediction. We can then observe that the gates are on average more sparse on the first cascade stages than on the last ones. Indeed, the first cascade stages tend to capture coarse landmark displacements (e.g. translation, scaling), thus requiring fewer base predictors to model these displacements. However, the last cascade stages capture finer and more subtle landmark displacement, requiring more base regressors. This is more prominent with tree-gates. Indeed, its structure allows to hierarchically pool the base predictors (as seen in Equation 3.19), leading to a better cooperation between them and an overall more efficient learning.

In addition, the runtime of Tree-gated MoE is low with $4,72$ ms per image on a NVIDIA GTX 1080 GPU, i.e. 211 fps. It can be improved by selecting a small proportion of the base regressors, so as not to have to compute the predictions of the unselected ones, as done in [147]. However, the selected ones must be sufficiently accurate. Figure 4.4 shows that we can obtain approximately the same level of accuracy with few base regressors, especially on the first cascade stages and by using tree-gates rather than soft-gates. This is confirmed by Figure 4.5, which shows the face shapes generated by the top-weighted base regressor (i.e. maximum value of the gate) depending on whether the training was performed with tree-gates or soft-gates. We can then see that tree-gates allow to learn much more accurate base regressors, whose predictions generate plausible face shapes that are close to the ground truth. This reinforces the fact that tree-gates allow an overall more efficient learning of the deep ensemble.

4.1.2.3 Conclusion

The experiments described above allow us to validate our approach in the prediction layer for face alignment. In particular, it has been shown that (1) using several base predictors leads to increase the robustness to head pose variations and occlusions compared to a single strong predictor, (2) using an adaptive mixture allows to better specialize the base regressors, and (3) using tree-gates improves the overall robustness while reducing the number of accurate base predictors.

In what follows, we extend our approach to the representation layer in order to better adapt feature extraction to the most extreme variations and thus obtain a robust representation to better regress facial landmark localization.

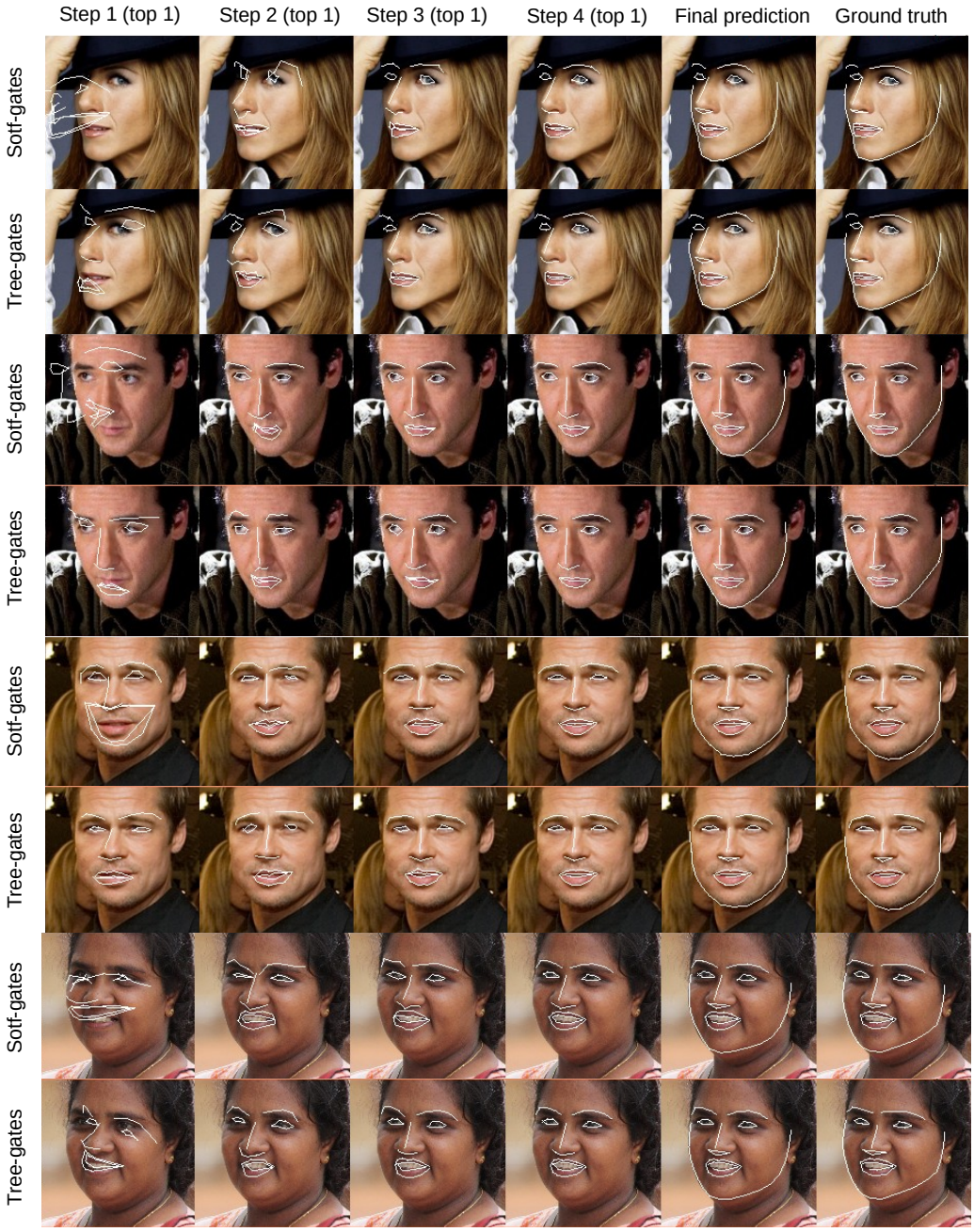


Figure 4.5: Visualisations of the predictions generated by the top-weighted (maximum value of either soft-gates or tree-gates) base regressor, for each cascade stage. Images from 300W testset.

4.1.3 Exogenous tree-gated MoE in the representation layer

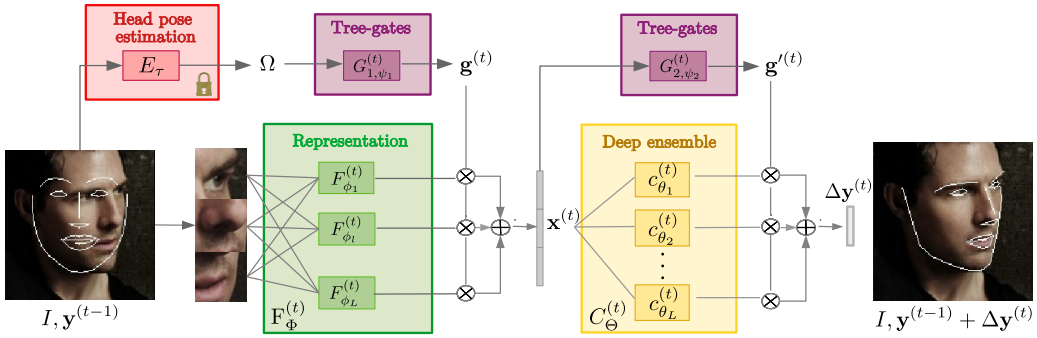


Figure 4.6: Exogenous tree-gated MoE for face alignment, with head pose estimation as gating variable in the representation layer.

The robustness of the last face alignment model can be further improved if the representation layer is better adapted to large variations. Thus, we propose to use an adaptive deep ensemble model for the representation layer, as an instance of the exogenous tree-gated MoE described in Section 3.5.2. Instead of using the raw pixel intensities as gating variable, which are high-dimensional with a low semantic level, we propose to use head pose which is (1) an important source of exogenous variations in face alignment datasets (e.g. 300W-LP, AFLW2000-3D) while being exogenous to landmark localization, and (2) can be modeled with high accuracy. Head pose estimation is therefore an ideal gating variable to condition feature extraction. Figure 4.6 illustrates the resulting model, called **Pose tree-gated MoE** and presented in the following.

4.1.3.1 Methodology

Architecture. To extend our adaptive deep ensemble method in the representation layer, we use $L = 8$ base convolutional networks (instead of the single strong one used in Tree-gated MoE), whose outputs are combined depending on head pose estimation.

Similar as before, all the patches are used for each base CNN to extract facial features. Each base CNN then contains as many layers to keep extracting 30 features per landmark, but uses fewer convolutional kernels in

each layer so as to keep the same number of trainable parameters: (7@5x5 → 14@5x5 → 28@3x3 → 56@3x3 → 30@1x1).

The facial features of the CNN ensemble are then adaptively weighted by head pose estimation (outputted by the model described below) through 3-depth tree-gates (i.e. $2^3 = 8$ terminal nodes) in order to specialize each base CNN on a hierarchical partition of the head pose space. The convolutional kernels learned by each base CNN are thus specialized on a region of the head pose space, leading the model to better adapt feature extraction, especially for large poses. At the top of the representation, the regression layer is then the same as in Tree-gated MoE. Thus, head pose estimation allows to extract a more suitable representation upon which the regressor ensemble better adapt to locate facial landmarks.

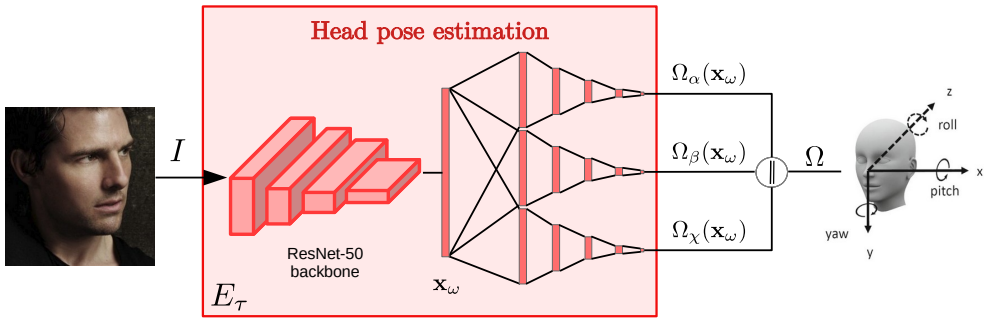


Figure 4.7: Head pose estimation.

Head pose model. To model head pose, we adopted the approach [138] which uses a pretrained ResNet-50 backbone to extract a high-level representation $\mathbf{x}_\omega$ (2048 units) from the face image I, upon which to regress head pose. In [138], the regressor is a deep network that uses several fully-connected layers predicting the three head pose angles at once. We have obtained greater accuracy by using instead three separate regressors, i.e. one for each component pose, in a multi-task learning fashion as illustrated in Figure 4.7.

Head pose estimation $\Omega \in [-\pi, \pi]^3$ is then the concatenation of the predictions outputted by the pitch regressor Ω_α with parameters α, the yaw regressor Ω_β with parameters β, and the roll regressor Ω_χ with parameters χ, as follows:

$$\Omega = \Omega_\alpha(\mathbf{x}_\omega)||\Omega_\beta(\mathbf{x}_\omega)||\Omega_\chi(\mathbf{x}_\omega) \tag{4.2}$$

where || is the concatenation operator.

The parameters for pitch, yaw and roll can thus be trained jointly. Lastly, we also finetune the ResNet-50 backbone parameters to further improve the representation and the regression accuracy.

Table 4.3: Mean absolute error between the prediction and the ground truth for each pose angle on 300W-LP testset. The range is $[-90°, 90°]$.

Head pose angle	Mean absolute error
Pitch	5.42
Yaw	7.22
Roll	5.09

Table 4.3 shows the accuracies obtained on the 300W-LP testsets. With a mean absolute error of 5.91°, we validate the strong accuracy of our head pose model. Thus, head pose verifies the properties described in Section 3.5.2 to be considered as an ideal gating variable, which is indeed (1) identified as an important source of variations in the data while being exogenous to landmark localization, and (2) modeled with high accuracy.

Evaluation metric. In order to evaluate the robustness of this approach to the most extreme head pose variations, we will use 3D-face alignment datasets, annotated with the 2D-coordinates of the true landmarks, (i.e. self-occluded or not) instead of their projections on the visible part of the face (i.e. as in 2D-face alignment datasets). For 3D-face alignment, the common evaluation metric used in the literature [200] is still a normalized mean error (i.e. average landmark-by-landmark distance between the ground truth $\mathbf{y}_i$ and the prediction $\hat{\mathbf{y}}_i$ of the i-th sample), but the normalization coefficient is the size of the face bounding box encompassing all the landmarks:

$$NME = \frac{1}{N} \sum_{i=1}^{N} \frac{||\hat{\mathbf{y}}_i - \mathbf{y}_i||_2}{\sqrt{h_i \times w_i}} \tag{4.3}$$

where h_i, w_i the height and width of the face bounding box, respectively.

Learning. Training is done with the same hyperparameters as for Tree-gated MoE. However, the training loss used is either the 2D-NME for 2D-face alignment datasets (i.e. 300W and COFW), or the 3D-NME for 3D-face alignment datasets (i.e. 300W-LP and AFLW2000-3D).

4.1.3.2 Evaluations

Table 4.4: Comparison of different architectures in term of 2D-NME (%).

Method	LFPW + HELEN	I-BUG	COFW
Baseline	4.22	8.89	5.9
Tree-gated MoE	**4.01**	<u>8.38</u>	<u>5.76</u>
Pose tree-gated MoE	<u>4.02</u>	**7.5**	**5.58**

Architecture comparison. Table 4.4 shows the accuracy of tree-gated deep ensemble models on 300W and COFW. Without drastically degrading the performance for moderate variations (4.01 $\rightarrow$ 4.02 on average on LFPW and HELEN testets), Pose tree-gated MoE allows to significantly improve the accuracy for head pose variations (8.38 $\rightarrow$ 7.5 on I-BUG). Indeed, by conditioning feature extraction by head pose estimation, the representation is better adapted and more robust to head pose variations, leading to a better accuracy. This better generalization capability is also observed for occluded faces (5.76 $\rightarrow$ 5.58 on COFW). By modeling both representation and regression layers with our deep ensemble method, robustness and accuracy are thus greatly improved in these unconstrained environments containing large variations.

Table 4.5: Comparison with state-of-the-art approaches on AFLW2000-3D in term of NME (%) for several yaw ranges.

Method	$[0, 30]$	$[30, 60]$	$[60, 90]$	Mean
LBF [137]	8.15	9.49	12.91	10.19
ESR [24]	4.60	6.70	12.67	7.99
CFSS [197]	4.77	6.71	11.79	7.76
RCPR [20]	4.26	5.96	13.18	7.80
MDM [159]	4.85	5.92	8.47	6.41
SDM [180]	3.67	4.94	9.76	6.12
3DDFA [200]	2.84	**3.57**	4.96	**3.79**
Tree-gated MoE	<u>2.84</u>	4.01	<u>4.93</u>	3.92
Pose tree-gated MoE	**2.78**	<u>3.97</u>	**4.76**	<u>3.84</u>

Comparison with state-of-the-art approaches. Table 4.5 shows a comparison between our tree-gated deep ensemble methods and other recent state-of-the-art methods on AFLW2000-3D for several yaw ranges. First, we can observe that Pose tree-gated MoE improves the average accuracy of Tree-gated MoE ($3.92 \rightarrow 3.84$). In particular, the performance gap is maximal on the $[60, 90]$ yaw range ($4.93 \rightarrow 4.76$). Specializing several base networks on regions of the head pose space thus allows to extract a more robust representation to the most extreme head pose variations, leading to increase the overall accuracy.

Second, Pose tree-gated MoE outperforms most of the other methods evaluated on AFLW2000-3D, except 3DDFA [200] which remains the state-of-the-art method on the $[30, 60]$ yaw range by leveraging a parametric face model. Our model allows to reach similar performances without the need to compute the parameters of a morphable model by directly predicting the landmark coordinates. Last but not least, our method allows to reach a new state-of-the-art in the [60.90] yaw range. Thus, our method is particularly efficient in the most extreme conditions, by using a more suitable representation upon which the regressor ensemble locate the facial landmarks.

Qualitative evaluation. In addition to quantitative evaluations, we also introspected the model to study the learned hierarchical partition of the head pose space. Figure 4.8 (bottom) shows the face images of AFLW2000-3D in the head pose 3D-space with yaw (red axis), pitch (green axis), and roll (blue axis). Each face image is colored according to the top-weighted base CNN in the first cascade stage. For more visibility, each face image is projected on the unit sphere. Figure 4.8 (top) shows the colors associated with each base CNN, so as to visualize how the tree splits the head pose space. By comparing the purple/blue images and orange/brown ones, we can then observe that the first level of the tree thus splits the head pose space according to the yaw angle (red axis). Indeed, the model has been trained on 300W-LP where each face image is augmented in yaw, which is therefore the dominant factor of head pose variations. By focusing only on the face images with positive yaw (in right figure), we can then see that the second level of the tree splits mostly according to the roll angle (blue axis), which is the second factor of head pose variation. Thus, the partition learned by Pose tree-gated MoE has well captured hierarchically the importance of the main factors of head pose variations, i.e. first yaw then roll, in the representation layer.

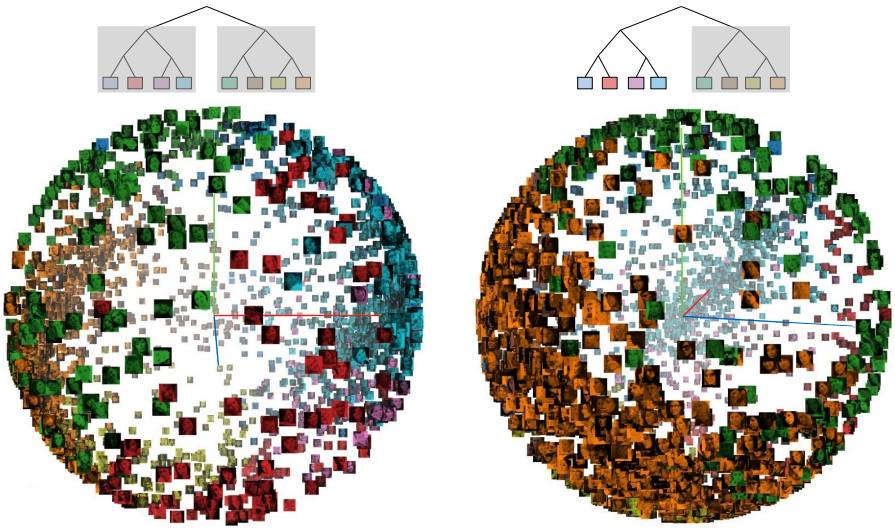


Figure 4.8: Hierarchical partition of the head pose space (learned by Pose tree-gated MoE on AFLW2000-3D). **Top**: Colors of each region of the partition. Each face image is colored according to the top-weighted base CNN. **Bottom**: Partition of the head pose space: yaw (red axis), pitch (green axis), roll (blue axis). **Left**: The first level of the tree splits the head pose space on the yaw axis. **Right**: For faces with positive yaw, the second level of the tree splits mostly on the roll axis.

This robust representation thus provides greater predictive power for the base regressors in the prediction layer. Figure 4.9 shows the face shapes generated by the top-weighted base regressor for each cascade stage. Compared to Tree-gated MoE in Figure 4.9, we can see that Pose tree-gated MoE has allowed to learn a strong representation to (1) generate a very likely face shape with only one single base predictor, and (2) even under extreme head pose conditions (up to 90° in absolute yaw). The final predictions (5^{th} and 11^{th} columns) are then very close to the ground truth (6^{th} and 12^{th} columns), whether to estimate head pose or to locate facial landmarks.

4.1.3.3 Conclusion

We have evaluated our framework in the representation layer, by using an ensemble of base convolutional networks whose the outputs are adaptively

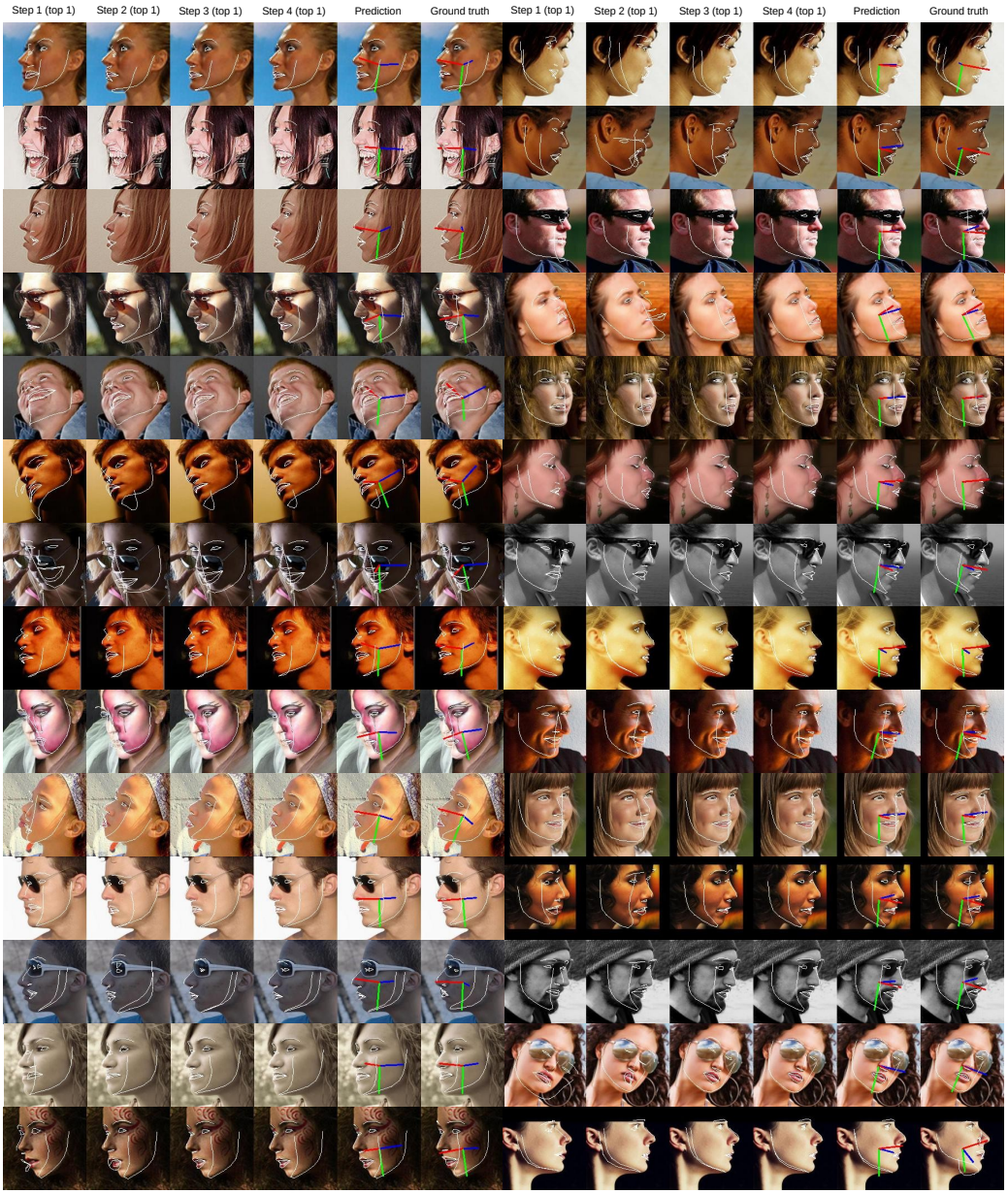


Figure 4.9: Visualisations of the predictions generated by the top-weighted base regressor, for each cascade stage. Head pose estimation is also displayed, as well as the ground truth. Images from AFLW2000-3D.

combined by a gating variable. For face alignment, head pose estimation is an ideal gating variable because it is a high-level semantic information that greatly affects the face appearance while being exogenous to landmark localization. Conditioning feature extraction by head pose estimation then increases the robustness of the model to the most extreme pose variations. In addition, the tree-gates have captured hierarchically the importance of each head pose component. The extracted features are then more suitable and robust to large poses, thus providing a strong representation upon which the regressor ensemble better adapt to locate facial landmarks.

Adaptive deep ensemble methods can therefore significantly improve the face alignment accuracy, especially under the most extreme conditions, and even more when applied to several model layers (i.e. representation and prediction layers). In what follows, we apply our framework to facial expression recognition.

4.2 Facial expression recognition

As seen in Section 1.1.2, facial expression (FE) can be described by using (1) a categorical model, (2) a dimensional model, or (3) the facial action coding system. The most recent and challenging datasets presented in Section 4.2.1 have large amounts of in-the-wild data annotated in categorical FE among the 7 basic FEs (happiness, surprise, sadness, anger, fear, disgust, and neutral) to establish cross-cultural consensus. We then propose in Section 4.2.2 to apply our adaptive deep ensemble methods to classify the 7 basic FEs under wild conditions. In particular, as shown in Figure 4.10, identity-related information (encompassing specific morphological traits, gender, age or ethnicity) greatly affect the face appearance while being exogenous to FER. By modeling it accurately enough, identity can then be considered as an ideal gating variable and thus better adapt the FE classifier to identity variations. Finally, we detail in Section 4.2.3 how to further improve the robustness of the FE representation to identity variations by using our new dispelling loss (described in Section 3.5.3).

4.2.1 Datasets

Several databases can be used to train and test the FER models. In particular, we have chosen the three most recent and challenging databases: RAF-DB, AffectNet and ExpW. Figure 4.10 shows examples of face images for each of these databases and for each facial expression. We can then observe that identity-related information accounts for a large part of the intra-class variability and therefore can be identified as an important source of variation in these data.

The **Real-world Affect database or RAF-DB** [100] is one of the most used FER databases with a good trade-off between data quantity and annotation quality. It contains $30k$ face images, annotated in categorical FE: either the 7 basic FEs, or compound FEs. As used mostly in literature, we use only the face images annotated with the 7 basic FEs, accounting for a total of 12271 samples for training and 3068 samples for testing. Identity-related information is an important source of variability in RAF-DB: by gender (52% female, 43% male, 5% unsure), age (from 0 to 70 years old), ethnicity (77% Caucasian, 8% African-American, and 15% Asian), and by the great variability in morphological traits as illustrated in Figure 4.10 (left). RAF-DB was annotated by 315 different human coders, and each image face

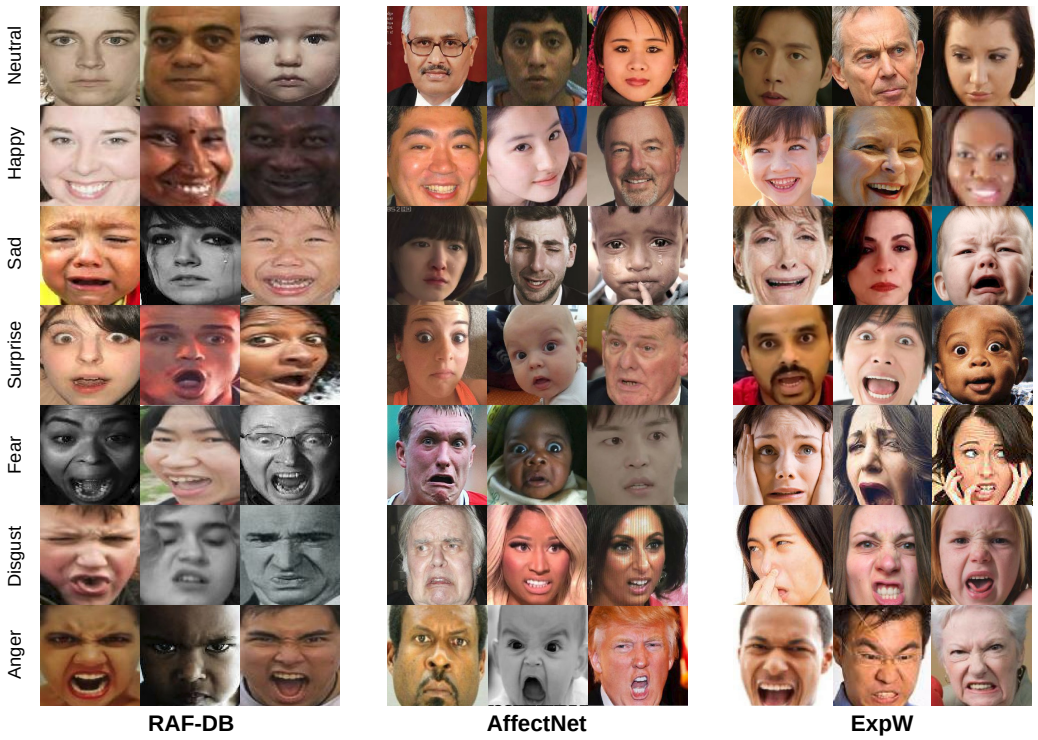


Figure 4.10: Facial expression recognition databases. Identity can be identified as an important source of variations in these data, while being exogenous to the FER task.

was annotated at least 40 times before obtaining the final annotation *via* crowdsourcing methods, thus validating the quality of the annotations.

The **AffectNet** database [123] is the largest FER database. It contains $400k$ face images, manually annotated by 12 different human coders in both categorical FE (the 7 basic FEs) and two-dimensional FE (valence/arousal intensities). These images have been collected by querying three search engines (Google, Bing, Yahoo) using 1250 expression related keywords in six different languages. Thus, webscrapping techniques have made it possible to collect a very large amount of data, with a lot of variability, especially in terms of identity, as illustrated in Figure 4.10 (middle). We use only the face images annotated with the 7 basic FEs, representing a total of $280k$ samples for training and $3,5k$ samples for testing. As used mostly in literature, we train the models by sampling uniformely the FE classes for each mini-batch.

The **Expression in-the-Wild or ExpW** [193] is the most recent FER database. It contains 91793 face images, manually annotated with the 7 basic FEs. These images were collected by querying a list of emotion-related keywords using the Google search image API. A face detector was then used to obtain face regions and measure face confidence in order to filter non-face images or cartoon-like images. As done in the literature, we use the images with a face confidence greater than 60, accounting for a total of 26701 samples for training and 6673 samples for testing. Similar to AffectNet, webscrapping techniques used to collect ExpW has led to obtain many identity variations, as illustrated in Figure 4.10 (right).

4.2.2 Exogenous tree-gated MoE in the prediction layer

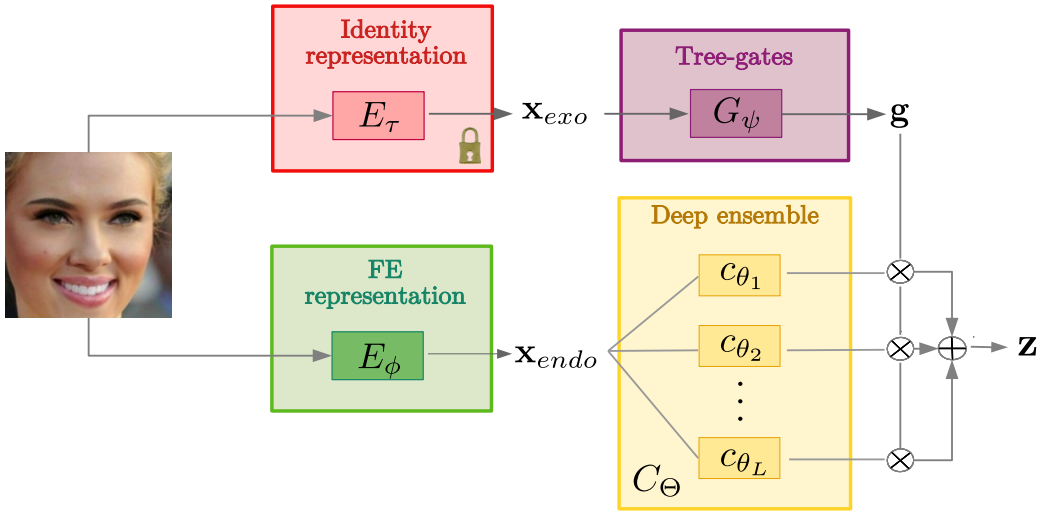


Figure 4.11: Exogenous tree-gated MoE for FER, with identity representation as gating variable in the prediction layer.

In order to classify the 7 basic FEs in unconstrained environments, we propose to use an adaptive deep ensemble method for the prediction layer, as an instance of the exogenous tree-gated MoE described in Section 3.5.2. As we have seen above, identity-related information can be identified as an important source of variations in the data, while being exogenous to the FER task. An identity representation, extracted by an accurate deep face recognition system, can thus be considered as an ideal gating variable to

better condition FER. Figure 4.11 illustrates the resulting model, presented in the following.

4.2.2.1 Methodology

Architectures. In the same vein as done in Section 4.1.2.1 for face alignment, we then define several models that incrementally add the architectural components of our method to recognize FE (i.e. deep ensemble, adaptive gates, identity as the exogenous gating variable).

For the representation layer, we use the VGG16 convolutional layers E_ϕ with parameters ϕ. It thus allows to extract a FE representation $\mathbf{x}_{endo}$ containing 512 feature maps of size 7x7. The parameters ϕ are pretrained either on ImageNet (denoted as VGG16) or for face recognition (denoted as VGGFace) and are finetuned for FER.

For the classification layer, several different architectures can be defined, as illustrated in Figure 4.12, whose the parameters are trained from scratch:

- **Baseline**: a single strong 3-FC classifier C_Θ containing two hidden layers with 4096 units each, and an output layer to classify the 7 facial expressions.

- **Deep ensemble:** a deep ensemble of $L = 32$ base 3-FC classifiers, each containing two hidden layers with 512 units each. The predictions are equally weighted: $\mathbf{g}_0 = (\frac{1}{L}, \ldots, \frac{1}{L})$.

- **Tree-gated MoE:** a deep ensemble of $L = 32$ base 3-FC classifiers, each containing two hidden layers with 512 units each. The predictions are weighted by tree-gates outputted by a 5-depth neural tree G_ψ and based on the (endogenous) FE representation $\mathbf{x}_{endo}$.

- **Identity tree-gated MoE:** a deep ensemble of $L = 32$ base 3-FC classifiers, each containing two hidden layers with 512 units each. The predictions are weighted by tree-gates outputted by a 5-depth neural tree G_ψ and based on the (exogenous) identity representation $\mathbf{x}_{exo}$.

With this configuration, each architecture has roughly the same total number of parameters (100 millions parameters total), allowing a fair comparison between the models.

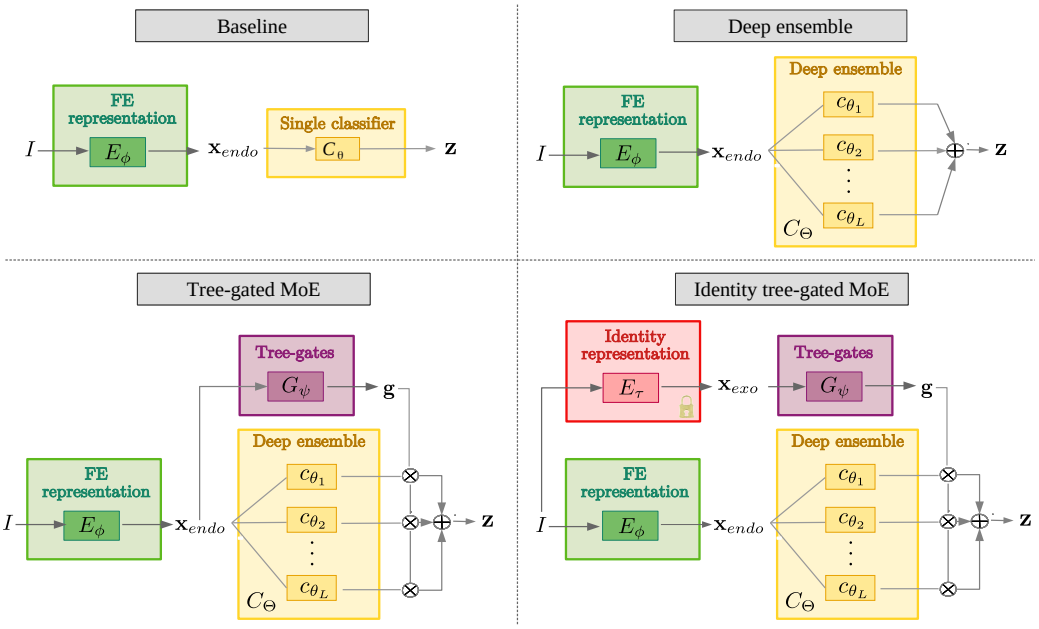


Figure 4.12: Architectures overview. **Baseline**: a single strong classifier. **Deep ensemble**: an ensemble of 32 base classifiers. **Tree-gated MoE**: the base classifiers are specialized in the FE representation space. **Identity tree-gated MoE**: the base classifiers are specialized in the identity representation space.

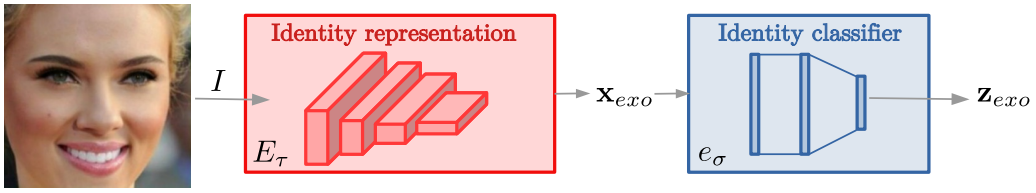


Figure 4.13: Identity model.

Identity model. To model identity, we use a VGG16 deep network trained for face recognition [129], as illustrated in Figure 4.13. In particular, we use the trained parameters available in [117]. The representation layer E_τ with parameters τ consists in 13 convolution layers. It allows to extract an identity representation $\mathbf{x}_{exo}$ containing 512 feature maps of size 7x7. The classification layer c_σ with parameters σ then consists in 3-FC layers containing two hidden layers with 4096 units each and an output layer to classify 2622 unique

identities (celebrity names) from the logits $\mathbf{z}_{exo}$. It was trained with 1000 face images per subject. This deep face recognition network allows to reach 98.95% accuracy on Labeled Faces in the Wild (LFW) [71], a benchmark dataset to evaluate face recognition models. Thus, it verifies the properties described in Section 3.5.2 to be considered as an ideal gating variable, which is indeed (1) identified as an important source of variations while being to the FER task, and (2) modeled with high accuracy. We then propose to condition a deep FE classifier ensemble by the identity representation $\mathbf{x}_{exo}$, which is robust and informative enough to characterize a great variability of identities.

Evaluation metric. Given a dataset containing N samples in the form (I_i, y_i) where I_i and y_i are respectively the i-th image and the class label, two metrics are used in the literature to evaluate a label prediction $\hat{y}_i$.

A first evaluation metric is the overall accuracy Acc corresponds to the proportion of samples that are correctly classified:

$$Acc = \frac{1}{N}\Sigma_{i=1}^{N}\mathbb{1}_{y_i=\hat{y}_i} \tag{4.4}$$

To evaluate the accuracy independently of the class distribution, a second evaluation metric is the confusion matrix M, giving the proportion of samples with label c that are classified with label p:

$$M_{c,p} = \frac{1}{N}\Sigma_{i=1}^{N}\mathbb{1}_{y_i=c}\mathbb{1}_{\hat{y}_i=p} \tag{4.5}$$

Some approaches in the literature use then the average accuracy per class, corresponding to the average of the M-diagonal values.

Learning. Preprocessing is done by first resizing the face image to 224x224, then by augmenting it with random rotation $\theta \sim Uniform([-10°, +10°])$, random horizontal flip, and random brightness, saturation, hue and contrast variations, as it is traditionally the case in the literature. Training is done by optimizing the standard cross-entropy over the parameters for FE representation ϕ, classification θ and gating ψ layers. These parameters are jointly trained in an end-to-end fashion by using ADAM optimizer [87] with a learning rate of $1e^{-5}$ and batch size 16. The parameters τ and σ of the identity model are kept frozen. For each experiment, we generate a validation set by randomly sampling 256 data from the trainset. We then select the best model in terms of accuracy on this validation set after 100k iterations.

4.2.2.2 Evaluations

Table 4.6: Comparison of different architectures in term of overall accuracy (%).

Method	RAF-DB	AffectNet	ExpW
Baseline (VGG16)	82.99	61.31	70.96
Baseline (VGGFace)	84.06	61.66	71.57
Deep ensemble	85.59	63.00	75.05
Tree-gated MoE	<u>86.38</u>	<u>63.34</u>	<u>75.17</u>
Identity tree-gated MoE	**87.29**	**63.71**	**75.74**

Architecture comparison. Table 4.4 shows the accuracy of each model on RAF-DB, AffectNet and ExpW databases. First, using a VGG16 pretrained for face recognition leads to a better accuracy than using a VGG16 pretrained on ImageNet (+1.29% on RAF-DB, +0.57% on AffectNet,+0.86% on ExpW). Indeed, the domain gap is narrower between FER and face recognition compared to object recognition.

Second, using several base predictors instead of a single strong predictor significantly increases the accuracy (+1.82% on RAF-DB, +2.17% on AffectNet, +4.86% on ExpW). It confirms that the robustness to large intra-class variations (as is the case in these FER datasets) can be improved by deep ensemble methods. This is also verified by the inter-class/intra-class ratio, which increases from 0.27/0.24/0.20 on RAF-DB/AffectNet/ExpW respectively for the baseline to 0.44/0.34/0.40 for the deep ensemble.

Third, using tree-gates to weight the predictions of the deep ensemble further improves the accuracy (+0.92% on RAF-DB, +0.54% on AffectNet,+0.16% on ExpW). It confirms that the gates allows to better adapt the deep ensemble to the most extreme variations by specializing each base predictor on specific regions of the gating space (here, the FE representation) so as to emphasize on the most relevant ones depending on the image input.

Finally, using identity representation as gating variable instead of the FE representation further increases the accuracy (+1.05% on RAF-DB, +0.58% on AffectNet,+0.76% on ExpW), Indeed, it allows to better specialize the base predictors and improve the overall robustness. It confirms that an exogenous variable identified as an important source of variation in the data is an ideal variable gating for combining base predictors.

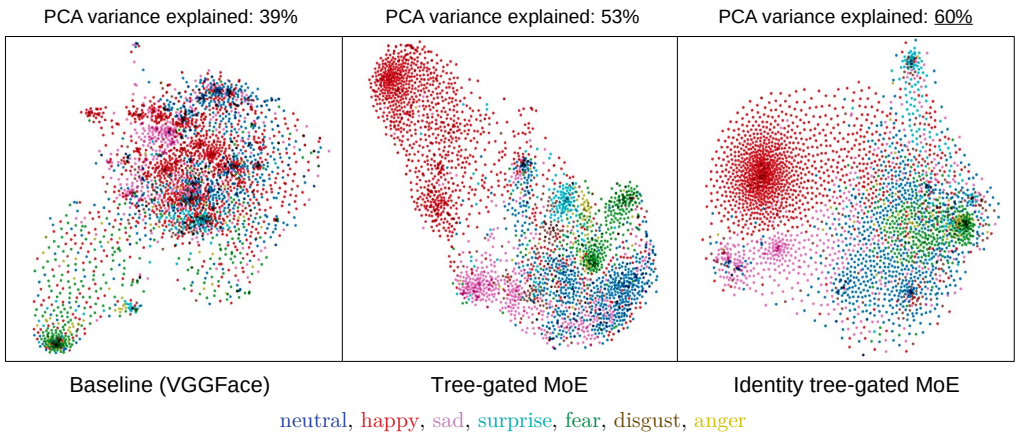


Figure 4.14: Visualization of the FE classes distribution of RAF-DB in the FE representation space. At the top of each figure, the percentage of variance explained on the first three axes of the PCA.

FE representation comparison. From a qualitative standpoint, Figure 4.14 shows the FE classes distribution of RAF-DB in the 2D-projection of the FE representation space, by using the t-SNE algorithm [114] for dimension reduction. We can then observe that our adaptive deep ensemble methods allow to learn a clearer distribution of the different FE classes in the representation space, illustrating their better robustness. In particular, using identity representation as gating variable further improves the distribution of the FE classes (e.g. sad, surprise). This highlights that the FE representation has less the constraint of capturing identity variations (which are already handled by gates), leading to improve its discriminative power and the overall performance.

4.2.2.3 Conclusion

The experiments described above allow us to validate our approach in the prediction layer for FER. Quantitative and qualitative evaluations have validated that (1) using several base predictors increases the robustness to identity variations compared to a single strong predictor, (2) using tree-gates allows to better specialize the base predictors and to better distribute the FE classes in the FE representation space, and (3) using identity as gating variable is better suited for the robustness in these FER datasets.

In what follows, we show how the dispelling loss introduced in Section 3.5.3 allows to learn a better FE representation by removing undesirable identity-related information from it, leading to increase the overall performance and to achieve a new state-of-the-art of these challenging FER databases.

4.2.3 THrowable Information Networks

Although we have showed that identity features is a better gating variable to specialize the base predictors, identity-related information may be retained in the FE representation. A strong identity variation without modifying the facial expression can thus strongly vary the FE representation fooling the base predictors. Ideally, each base predictor should be invariant to identity variations. We then aim to encourage the removal of identity-related information from the FE representation.

4.2.3.1 Methodology

In the identity tree-gated MoE architecture, FE and identity representation layers have the same structure. Thus, the FE representation $\mathbf{x}_{endo}$ can be feed into the identity classifier e_σ to generate a prediction $\mathbf{z}_{exo}^{(endo)}$ (logits vector), as illustrated in Figure 4.15. This identity prediction can then be compared with the prediction $\mathbf{z}_{exo}^{(exo)}$ obtained with the identity representation $\mathbf{x}_{exo}$. If $\mathbf{x}_{endo}$ contains discriminative information predicting the true identity, then $\mathbf{z}_{exo}^{(endo)}$ and $\mathbf{z}_{exo}^{(exo)}$ are similar. To remove identity-related information in the FE representation, we then propose to penalize the similarity between FE and identity representations during training, by using our dispelling loss $\mathcal{L}_{sim}$ described in Section 3.5.3:

$$\mathcal{L}_{sim}(\phi) = \mid cos(\mathbf{z}_{exo}^{(exo)}, \mathbf{z}_{exo}^{(endo)}) \mid = \frac{\mid \mathbf{z}_{exo}^{(exo)} . \mathbf{z}_{exo}^{(endo)} \mid}{||\mathbf{z}_{exo}^{(exo)}||_2 . ||\mathbf{z}_{exo}^{(endo)}||_2} \tag{4.6}$$

The parameters ϕ of the FE representation layer are thus trained to minimize both the cross-entropy $\mathcal{L}_{target}$ in order to learn discriminative features, and to minimize its similarity with the identity representation through our new dispelling loss $\mathcal{L}_{sim}$ in order to remove undesirable identity-related information from it. The final loss can then be written as:

$$\mathcal{L}(\Theta, \phi, \psi) = \mathcal{L}_{target}(\Theta, \phi, \psi) + \lambda \mathcal{L}_{sim}(\phi) \tag{4.7}$$

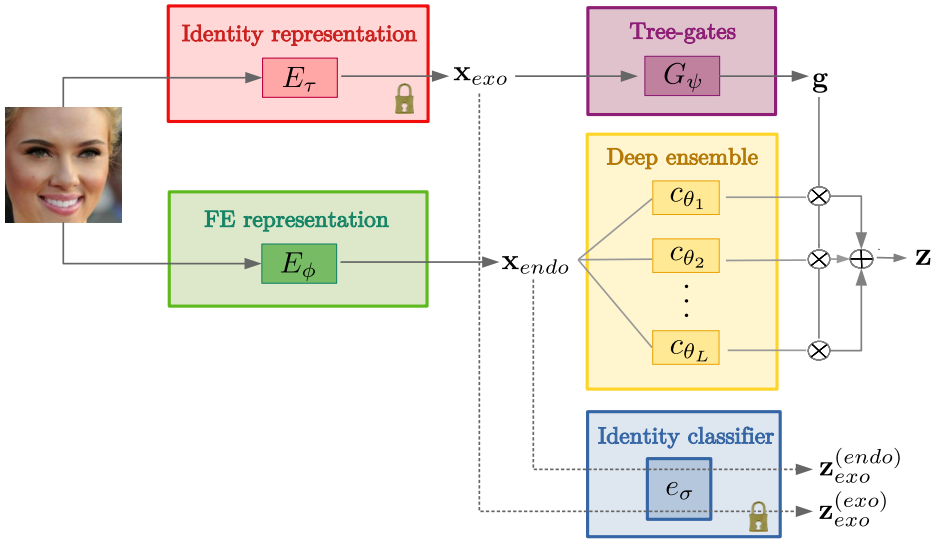


Figure 4.15: THIN overview. To remove identity-related information in the FE representation, we penalize the similarity between FE and identity representations in the identity classification space during training.

where Θ, ϕ, ψ are the parameters for the deep ensemble, the endogenous representation and the gating layers respectively, and $\lambda > 0$ the penalization coefficient of the dispelling loss whose setting is discussed in the following.

4.2.3.2 Evaluations

Table 4.7: Comparison of different architectures in term of accuracy (%).

Method	RAF-DB	AffectNet	ExpW
Identity tree-gated MoE	87.29	63.71	75.74
THIN	**87.81**	**63.97**	**76.08**

Architecture comparison. Table 4.7 shows the gain in accuracy that can be obtained on the three FER databases, by adding the dispelling loss in the training of Identity tree-gated MoE (+0.60% on RAF-DB, +0.41% on AffectNet, +0.45% on ExpW). Thus, encouraging the removal of identity-related information in the FE representation improves the discriminative power of the representation upon which the base predictors are based, leading

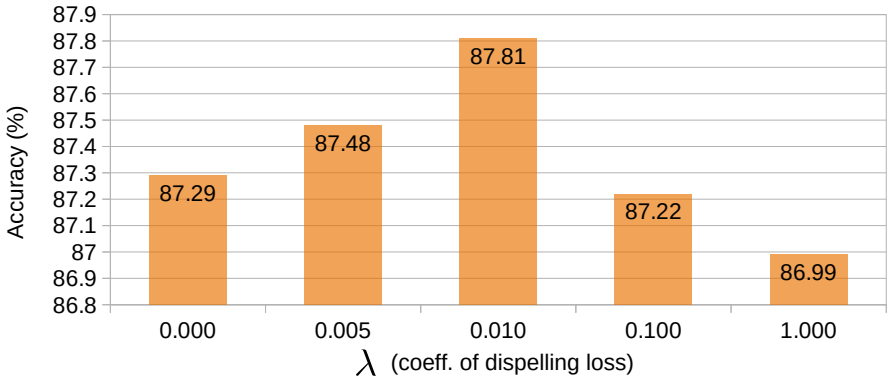


Figure 4.16: Ablation study for the coefficient of dispelling loss in term of accuracy (%) on RAF-DB.

to a better robustness to identity variations.

However, similarly to the toy experiments showcased in Section 3.5.3, the penalization coefficient λ of the dispelling loss must be carefully calibrated. Figure 4.16 shows the accuracies of several THIN models depending on the penalization coefficient λ used in the dispelling loss $\mathcal{L}_{sim}$ during the training. If λ is too strong ($\lambda > 0.1$ on RAF-DB), the training is too involved in removing rotation-related information from the FE representation rather than improving its discriminative power for FER, thus leading to decrease the overall performance. However, if we set $\lambda = 0.005$ or $\lambda = 0.01$, it is possible to balance these two objectives. It allows then to remove undesirable identity-related information from the FE representation while learning to recognize FE, leading to improve the robustness to identity variations and to increase the overall accuracy.

Comparison with state-of-the-art approaches. Table 4.8 shows a comparison between THIN and other recent state-of-the-art approaches on RAF-DB, AffectNet and ExpW. We can then observe that THIN allows to reach a new state-of-the-art on the three FER databases which are the largest, most recent and most challenging ones (87.00 $\rightarrow$ 87.81 on RAF-DB, 63.54 $\rightarrow$ 63.97 on AffectNet, 71.90 $\rightarrow$ 76.08 on ExpW). Our method has therefore effectively leveraged identity information, as an important source of variation in these data, to improve both prediction (by using it as gating variable to better specialize the base predictors) and representation (by using it in the dispelling

Table 4.8: Comparison with state-of-the-art approaches in term of overall accuracy (%).

Method	RAF-DB	AffectNet	ExpW
PG-CNN [102]	83.27	55.33	-
Separate loss [101]	86.38	58.89	-
IPA2LT [188]	86.77	57.31	-
RAN [170]	86.9	59.5	-
Covariance pooling [1]	<u>87.00</u>	-	-
SNA [54]	-	62.7	-
BReG-Net [60]	-	<u>63.54</u>	-
PAT-VGG [21]	86.28	-	71.5
EAFR [104]	82.69	-	<u>71.90</u>
THIN	**87.81**	**63.97**	**76.08**

loss to remove undesirable identity-related information) layers. Moreover, our method is at most close to the consensus of annotations between human coders. Indeed, on AffectNet, the annotators agree for 65.3% of the testset. It can then be considered as a ceiling. THIN thus generalizes well on AffectNet, i.e. the largest FER database containing the most identity variations.

We also display in Figure 4.18 the confusion matrices for THIN on each testset. First, we can see that happy and neutral expressions are generally well classified, and that fear and disgust are the least well classified. Second, predictions tend to be misclassified to neutral, especially on ExpW. One strategy to address this issue may be to balance the mini-batches in terms of FE classes, as is done for AffectNet, so as to better balance the predictions. Third, as is often observed in literature, surprise and fear are most often confused. Indeed, these FEs use roughly the same action units (open eyes and mouth) making it more difficult to differentiate them, even for humans.

In addition, it is important to note that the quality of annotations, in particular for the databases whose images have been webscrapped (i.e. AffectNet and ExpW), can be low on a significant amount of samples that have not been re-annotated. Figure 4.17 shows examples of well or badly classified samples. We can then see that some face images, whose action units associated to each FE are well activated (e.g. smile and squinted eyes for the happy expression), have been well classified by our model despite wrong annotations thus penalizing the overall accuracy.

Figure 4.17: Examples of good (green) and bad (red) predictions on RAF-DB, AffectNet and ExpW. For each sample, we display the true FE class (in black) and the prediction (in green or red). Misclassifications can be due to wrong annotations, especially on AffectNet and ExpW.

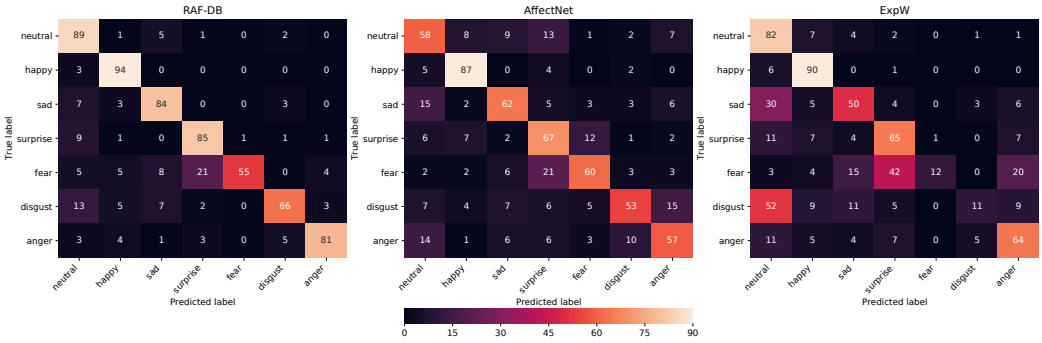


Figure 4.18: Confusion matrix (%).

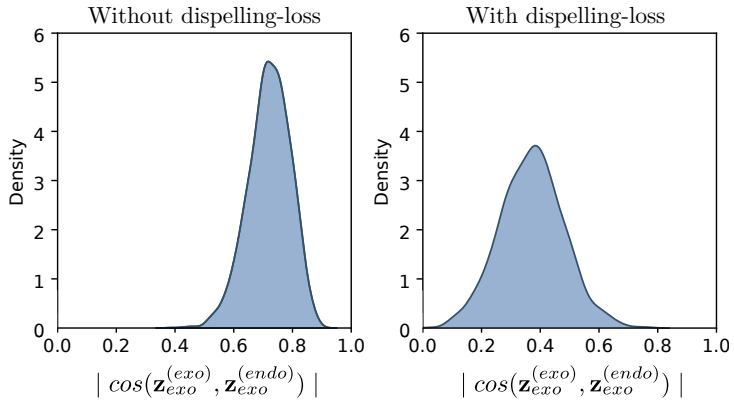


Figure 4.19: Distributions of absolute cosine distances $\mathcal{L}_{sim}(\phi)$ on the RAF-DB testset. Left: $\lambda = 0$ (no dispelling loss). Right: $\lambda = 0.01$. Our dispelling loss allows to effectively orthogonalize the FE and identity representations.

Dispelling identity in the FE representation. Several qualitative evaluations validate that our dispelling loss allows to remove undesirable identity-related information in the FE representation, leading to improve the overall accuracy, as we have seen in the above.

Figure 4.19 shows the distribution of similarities between FE and identity representations on the RAF-DB testset (i.e. cosine distances between their respective identity logits vectors), depending on whether our dispelling loss is used during training (i.e. $\lambda = 0.01$) or not (i.e. $\lambda = 0$). We can then observe that cosine distances tend to average more towards 0 if our dispelling loss is used, showing that THIN has indeed semantically orthogonalized FE and identity representations, thus making them less similar.

To evaluate the sensitivity of our models to identity variations, we study how its FE representation varies between two face images depending on whether it is from the same person or not, so as to see if the FE representation is invariant to identity. From the LFW database (which can contain several images of the same person), we then generated positive pairs (i.e. same identity) and negative pairs (i.e. different identities). Figure 4.20 shows the distance distributions of FE representations for positive (red curve) and negative (green curve) pairs.

In Figure 4.20 (left), we show these distributions in the identity representation space, corresponding to the case where the identities are separated as much as possible. Above the figure is displayed the Intersection-over-Union

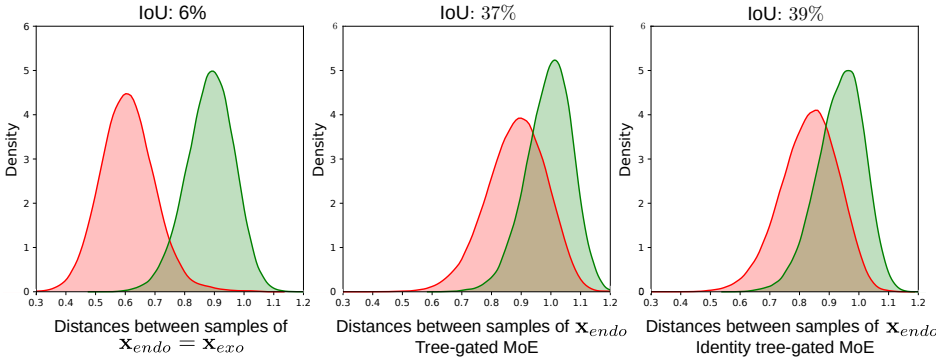


Figure 4.20: Distance distributions in the FE representation space between sample pairs of LFW [71]. Red: positive sample pairs (i.e. same identity). Green: negative sample pairs (i.e. different identity).

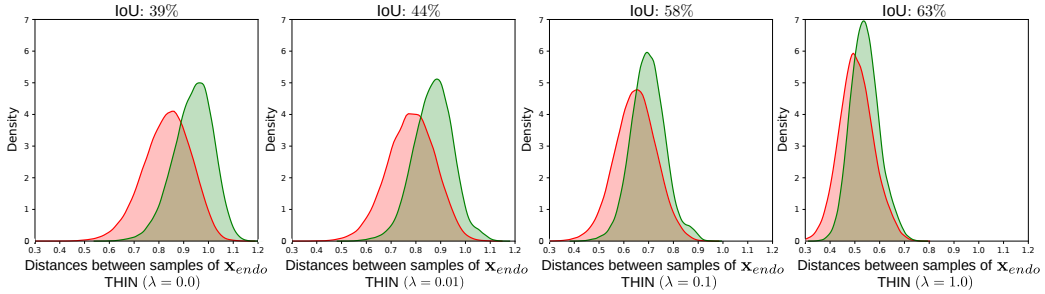


Figure 4.21: Distance distributions in the FE representation space between sample pairs of LFW for THIN models. Red: positive sample pairs (i.e. same identity). Green: negative sample pairs (i.e. different identity).

(IoU) measuring the overlap between the distributions. Thus, in this case, the distributions do not overlap much, indicating that it is easy to distinguish positive and negative pairs, which is natural because the identity representation is used to recognize a specific person.

In Figure 4.20 (middle), we show these distributions in the FE representation space of the tree-gated MoE. We can then see that the FE representation is much more invariant to identity than in the previous case. Indeed, the distributions overlap much more (IoU increases from 6% to 37%). As the FE representation is initialized by the identity representation at the beginning of the training, it means that it is optimal to remove some identity-related information for FE representation learning. It reinforces the idea that FER and face recognition are semantically orthogonal tasks.

In Figure 4.20 (right), we can then observe that the identity tree-gated MoE increases very slightly the overlap of the distributions (IoU increases from 37% to 39%). While tree-gated MoE must retain some identity-related information in the FE representation so that the base predictors can adapt the classification, the identity tree-gated MoE has less this constraint because the gates already contain identity information, leading to learn a FE representation more invariant to identity variations.

Figure 4.21 shows the distance distributions in the FE representation space for THIN models. We can then observe that the stronger the coefficient λ is, the higher the IoU increases. This means that our dispelling loss decreases well the sensitivity of the FE representation to identity variations. However, as we have seen in the quantitative evaluations, one must be careful not to choose too strong coefficients λ that focus the training on identity-dispelling and limit the FE representation learning.

FE representation comparison. Figure 4.22 shows the FE classes distribution of RAF-DB in the 2D-projection of the FE representation space. We can then observe that THIN allows to learn a clearer distribution of the different FE classes in the representation space (e.g. fear, surprise), leading to improve its discriminative power. This better distribution is further confirmed the 2D-projection quality of THIN: the percentage of variance explained on the first three axes of the PCA increase from 60% to 66%. It illustrates that our dispelling loss allows to learn a more robust FE representation to the identity variations, facilitating the base predictors to recognize FE.

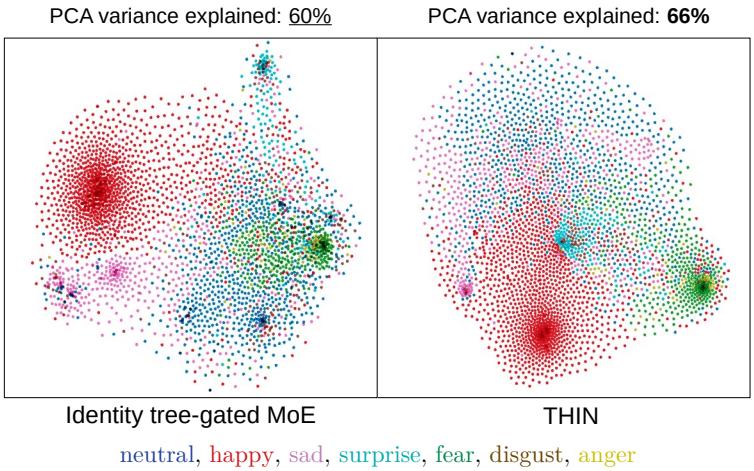


Figure 4.22: Visualization of the FE classes distribution of RAF-DB in the FE representation space depending on whether our dispelling loss is used or not. At the top of each figure, the percentage of variance explained on the first three axes of the PCA.

4.2.3.3 Conclusion

The experiments described above allow us to validate our dispelling loss to learn a better FE representation, leading to increase the overall accuracy and to achieve a new state-of-the-art on the most recent and challenging FER datasets, containing large identity variations. Thus, our method leverages the identity information twice: first as gating variable to best specialize the base predictors, second to remove undesirable identity-related information in the FE representation so as to make it more invariant and robust to identity variations, leading to these high performances.

4.3 Outline

In this chapter, we have experimentally validated the genericity of our framework through several machine learning problems in unconstrained environments (i.e. regression for face alignment and classification for facial expression recognition) and several model layers (i.e. prediction and representation layers).

First, we have shown that using several base networks increases the robustness to large variations compared to a single strong network. Second, using an adaptive combiner allows to better specialize the base networks. Similarly to the toy experiments showcased in Section 3.5, we have validated that a gating variable identified as an important source of variations while being exogenous to the task is better suited for the robustness of the base networks to large variations in this exogenous variable. In addition, the hierarchical structure of the tree-gates increases the overall robustness while reducing the number of accurate base networks, thus reinforcing their respective specializations. Finally, we have shown that our dispelling loss allows to learn a better endogenous representation by removing undesirable exogenous information from it, leading to increase the robustness and invariance to the exogenous variations.

Our framework has allowed to reach the top of state-of-the-art on today's most recent and challenging databases. Adaptive deep ensemble methods can therefore significantly improve the overall accuracy, especially under the most extreme conditions, and even more when applied sequentially to all model layers.

Chapter 5

Conclusion and Future works

5.1 Conclusion

Throughout this thesis, we have proposed a framework (model architecture and learning algorithm) aiming to increase the robustness to large variations in the data. In particular, it has allowed to develop a robust face analysis system both to localize facial landmarks and recognize facial expressions while dealing with a great variety of head poses, morphological traits, or objects that can occlude the face.

As seen in Chapter 2, three main approaches aim to increase the robustness of face analysis systems:

- DNNs-based methods, by jointly learning predictor and complex representation, which can encompass many possible variations.
- Ensemble methods, by using several base predictors whose the diversity of decisions allows to decrease the variance of prediction errors.
- Adaptive methods, by explicitly integrating a specific factor of variation into the model to extract more suitable features and to better adapt the model to the most extreme variations related to this factor.

We have merged these approaches to take advantage of each of their respective benefits and proposed adaptive deep ensemble methods. Given a target task (e.g. face alignment) and a model layer (e.g. a representation or prediction layer), we use an ensemble of base networks whose decisions are adaptively weighted by a gating variable (e.g. head pose estimation)

that specializes each of these base networks on a region of the gating space (e.g. left-oriented faces). The ensemble accuracy can then be improved by a better cooperation between the base networks, which is handled either by (1) the gating structure, (2) the gating variable conditioning the task, or (3) the learning algorithm.

For (1), we propose a hierarchical gating structure in order to learn a hierarchical partition of the gating space to improve the specialization of the base networks both in their distribution in the gating space and in their prediction confidence. For (2), we propose to integrate an exogenous variable, identified as an important source of variation and modeled with sufficient accuracy, in order to better condition the target task and to better specialize the base networks. Thus, the base networks are specialized in order to be able to adapt to the most extreme variations of this exogenous variable. For (3), we propose a new training loss to remove undesirable exogenous-related information from the endogenous representation deciphering the task. Thus, it reduces the sensitivity of the endogenous variable to variations in the exogenous variable and improves its discriminative power.

Through several experiments, we have validated our approach for different predictive tasks in various domains (both on synthetic and realistic datasets), and by applying it to different model layers (prediction and representation layers). In particular, our framework has allowed to reach the top of state-of-the-art on today's most recent and challenging databases. Thus, adaptive deep ensemble methods significantly improve the overall accuracy for several face analysis tasks, such as face alignment or facial expression recognition, especially under the most extreme conditions, and even more when applied sequentially to all model layers.

To sum it up, the main contributions of our approach are three-folds:

- From an architectural standpoint, we propose a new adaptive deep ensemble architecture using (1) a hierarchical gating structure to learn more efficiently input space clusters, and (2) a gating variable exogenous to the target task to better condition it and learn base networks more robust to these exogenous variations.

- From a learning standpoint, we propose a new training loss encouraging to remove the exogenous information from the endogenous representation, further improving the overall learning algorithm and the robustness of base networks to exogenous variations.

- From an experimental standpoint, we propose generic methods that can be applied to multiple layers (e.g. prediction, representation layer) and multiple predictive tasks (e.g. face alignment, facial expression recognition, digit recognition, shape recognition). We experimentally validate our approach on synthetic and realistic datasets. In particular, we show that our method significantly improves the robustness to the most extreme variations.

These contributions led to multiple publications in international conferences and journals [6][5][7], as well as a Python code framework that have been released open-source [4] for performing face analysis in the wild (facial expression recognition, face alignment, head pose estimation, face recognition).

5.2 Future works

Our framework provides several interesting directions from an architectural, learning or experimental standpoint.

5.2.1 Architectural standpoint

Gating variable. A first architectural direction is to learn a more suitable gating space in which to specialize the base networks. Rather than adapting the model according to an exogenous variable related to a *single* factor of variations, we could use one that encapsulates *all* the factors. To this end, we could first use unsupervised disentanglement methods to learn representations related to each factor of variations in the data. In particular, these methods use an auto-encoder, whose encoder extracts several embeddings each capturing a specific factor of variation. Second, we could retrieve these embeddings and condition the target task by a linear combination of them, whose weighting coefficients are jointly trained with the entire system. Thus, it allows to learn the importance of each of the factor of variations in the data and capture more exogenous variations through the gating variable, leading to better adapt the model to all exogenous variations.

Gating structure. A second architectural direction is to learn a better clustering of the gating space. Rather than fixing the structure of the neural

tree (i.e. binary tree with arbitrary depth), we could progressively grow the tree jointly with the learning of the entire system. To this end, we could use the growth procedure of the adaptive neural tree proposed in [155]. It would not require to arbitrarily set the depth of the tree (i.e. and therefore the number of clusters in the gating space), leading to further exploration of the space of possible trees that may be unbalanced (i.e. and thus learning more efficient clustering).

5.2.2 Learning standpoint

Dispelling loss. In order to improve the learning algorithm, a first direction is to use a more efficient dispelling loss to better remove undesirable information related to the exogenous variable in the endogenous representation. Rather than encouraging to orthogonalize the exogenous and endogenous variables in the exogenous prediction space, we could employ an adversarial technique by training the parameters of the endogenous representation layer so as to both minimize the loss of the target task and maximize the loss used to model the exogenous variable.

Multi-task learning. A second direction to improve the learning algorithm is to use multi-task learning techniques. Consider an adaptive deep ensemble where $\mathbf{x}_{endo}$ is the endogenous representation and $\mathbf{x}_{exo}$ is the exogenous one. Since the exogeneity relation is symmetric (i.e. if $\mathbf{x}_{endo}$ is exogenous to $\mathbf{x}_{exo}$, then $\mathbf{x}_{exo}$ is exogenous to $\mathbf{x}_{endo}$), then $\mathbf{x}_{endo}$ is a suitable gating variable to condition the predictor of the exogenous variable. We could then jointly learn the $\mathbf{x}_{exo}$-conditioned predictor of the endogenous variable and the $\mathbf{x}_{endo}$-conditioned predictor of the exogenous variable. Following the multi-task framework, what is learned for one predictor can help the other one to better learn representation. We can also perform alternative learning: first train a first predictor by fixing the second one, then train the second one by fixing the first one, and so on, to iteratively refine them. Thus, by using the endogenous representation two times (i.e. to decipher the target task and condition the exogenous task), its generalization capacity can be improved, leading to facilitate the predictive power of the base predictors.

5.2.3 Experimental standpoint

Applications to real use-cases. It would be interesting to deploy our face analysis system on practical applications, such as (1) measuring the likelihood of a facial expression mimed by a child with autism in order to help him better transmit his emotions, or (2) measuring the level of respiratory suffering of a person under respiratory assistance in order to detect the dyspnea disease. Indeed, for (1), our system is robust to identity variations, allowing it to adapt accurately to children's face appearance. For (2), it would be more appropriate to describe FE by using a dimensional model (i.e. valence/arousal) to detect and measure the intensity of the unpleasant feeling, or the FACS manual to detect certain facial muscles that may reflect a respiratory gene. In particular, we could use transfer learning techniques from our FE classifier pretrained on large datasets (i.e. AffectNet) and whose representations robust to identity variations and occlusions can greatly help to regress valence/arousal or to detect action units (which are highly correlated tasks) with few additional data.

Applications to other machine learning problems. The contributions of our framework have been experimentally validated for two predictive tasks in real conditions that contained large variations. Although we showed the genericity of our approach in other domains from synthetic datasets, the predictive tasks (i.e. digit recognition, shape recognition) were relatively easy to model. It would then be interesting to investigate other real domains drastically different from those related to the faces. For computer vision, our framework could be applied to other domains in recognition such as gesture recognition with the body orientation as the exogenous gating variable, or for other categories of tasks such as image segmentation with the nature of landscape as gating variable. For natural language processing, we could apply our framework to various tasks such as question answering with topic features as gating variable. Last but not least, our approach could be applied to other types of machine learning, such as unsupervised learning by training auto-encoders, each specialized in a region of a suitable gating space.

Bibliography

[1] D. Acharya, Z. Huang, D. Pani Paudel, and L. Van Gool, "Covariance pooling for facial expression recognition," in *CVPR workshops*, 2018.

[2] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *Transactions on pattern analysis and machine intelligence*, 2006.

[3] E. Antonakos, J. Alabort-i-Medina, G. Tzimiropoulos, and S. Zafeiriou, "Hog active appearance models," in *ICIP*, 2014.

[4] E. Arnaud, *Adaptive deep ensemble methods for face analysis in the wild*, https://github.com/estephe-arnaud/ademfa, 2020.

[5] E. Arnaud, A. Dapogny, and K. Bailly, "Tree-gated deep mixture-of-experts for pose-robust face alignment," *Transactions on biometrics, behavior, and identity science*, 2019.

[6] ——, "Tree-gated deep regressor ensemble for face alignment in the wild," in *FG*, 2019.

[7] ——, "Thin: Throwable information networks and application for facial expression recognition in the wild," *arXiv preprint arXiv:2010.07614*, 2020.

[8] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic, "Robust discriminative response map fitting with constrained local models," in *CVPR*, 2013.

[9] K. Bailly, M. Milgram, P. Phothisane, and E. Bigorgne, "Learning global cost function for face alignment," in *ICPR*, 2012.

[10] S. Baker, "Lucas-kanade 20 years on: A unifying framework: Part 3," Citeseer, Tech. Rep., 2003.

[11] S. A. Bargal, E. Barsoum, C. C. Ferrer, and C. Zhang, "Emotion recognition in the wild from videos using images," in *ICMI*, 2016.

[12] S. Bernard, S. Adam, and L. Heutte, "Dynamic random forests," *Pattern recognition letters*, 2012.

[13] V. N. Boddeti, M.-C. Roh, J. Shin, T. Oguri, and T. Kanade, "Face alignment robust to pose, expressions and occlusions," *Transactions on pattern analysis and machine intelligence*, 2017.

[14] O. Bouafif, B. Khomutenko, and M. Daoudi, "Hybrid approach for 3d head reconstruction: Using neural networks and visual geometry," in *ICPR*, 2020.

[15] E. A. Boyle, A. H. Anderson, and A. Newlands, "The effects of visibility on dialogue and performance in a cooperative problem solving task," *Language and speech*, 1994.

[16] J. K. Bradley and R. E. Schapire, "Filterboost: Regression and classification on large datasets," in *NIPS*, 2008.

[17] L. Breiman, "Bagging predictors," *Machine learning*, 1996.

[18] ——, "Stacked regressions," *Machine learning*, 1996.

[19] ——, "Random forests," *Machine learning*, 2001.

[20] X. P. Burgos-Artizzu, P. Perona, and P. Dollár, "Robust face landmark estimation under occlusion," in *ICCV*, 2013.

[21] J. Cai, "Improving person-independent facial expression recognition using deep learning," PhD thesis, 2019.

[22] J. Cai, Z. Meng, A. S. Khan, Z. Li, J. O'Reilly, and Y. Tong, "Island loss for learning discriminative features in facial expression recognition," in *FG*, 2018.

[23] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *FG*, 2018.

[24] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," *CVPR*, 2012.

[25] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *ACM SIGKDD*, 2016.

[26] Z. H. Choudhury and K. Mehata, "Robust facial marks detection method using aam and surf," *International journal engineering research and applications*, 2012.

[27] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *Transactions on pattern analysis and machine intelligence*, 2001.

[28] T. F. Cootes, K. Walker, and C. J. Taylor, "View-based active appearance models," in *FG*, 2000.

[29] S. F. Cotter, "Sparse representation for accurate classification of corrupted and occluded facial expressions," in *ICASSP*, 2010.

[30] D. Cristinacce and T. F. Cootes, "Feature detection and tracking with constrained local models," in *BMVC*, 2006.

[31] ——, "Boosted regression active shape models," in *BMVC*, 2007.

[32] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti, "Dynamic classifier selection: Recent advances and perspectives," *Information fusion*, 2018.

[33] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[34] M. Dantone, J. Gall, G. Fanelli, and L. Van Gool, "Real-time facial feature detection using conditional regression forests," in *CVPR*, 2012.

[35] A. Dapogny, K. Bailly, and M. Cord, "Decafa: Deep convolutional cascade for face alignment in the wild," in *ICCV*, 2019.

[36] A. Dapogny, K. Bailly, and S. Dubuisson, "Confidence-weighted local expression predictions for occlusion handling in expression recognition and action unit detection," *International journal of computer vision*, 2018.

[37] A. Dapogny, K. Bailly, and S. Dubuisson, "Face alignment with cascaded semi-parametric deep greedy neural forests," *Pattern recognition letters*, 2017.

[38] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society*, 1977.

[39] T. Devries, K. Biswaranjan, and G. W. Taylor, "Multi-task learning of facial landmarks and expression," in *CRV*, 2014.

[40] A. Dhall, A. Asthana, R. Goecke, and T. Gedeon, "Emotion recognition using phog and lpq features," in *FG*, 2011.

[41] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning*, 2000.

[42] ——, "Ensemble methods in machine learning," in *MCS workshops*, 2000.

[43] H. Ding, S. K. Zhou, and R. Chellappa, "Facenet2expnet: Regularizing a deep face recognition net for expression recognition," in *FG*, 2017.

[44] C. Domingo, O. Watanabe, *et al.*, "Madaboost: A modification of adaboost," in *COLT*, 2000.

[45] N. Dvornik, C. Schmid, and J. Mairal, "Diversity with cooperation: Ensemble methods for few-shot classification," in *CVPR*, 2019.

[46] D. Eigen, M. Ranzato, and I. Sutskever, "Learning factored representations in a deep mixture of experts," in *ICLR workshops*, 2014.

[47] P. Ekman and W. V. Friesen, "Constants across cultures in the face and emotion," *Journal of personality and social psychology*, 1971.

[48] ——, *Manual for the facial action coding system*. Consulting psychologists press, 1978.

[49] Y. Fan, J. C. Lam, and V. O. Li, "Multi-region ensemble convolutional neural network for facial expression recognition," in *ICANN*, 2018.

[50] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *Journal of machine learning research*, 2014.

[51] J. R. Fontaine, K. R. Scherer, E. B. Roesch, and P. C. Ellsworth, "The world of emotions is not two-dimensional," *Psychological science*, 2007.

[52] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *ICML*, 1996.

[53] J. Friedman, T. Hastie, R. Tibshirani, *et al.*, "Additive logistic regression: A statistical view of boosting," *The annals of statistics*, 2000.

[54] Y. Fu, X. Wu, X. Li, Z. Pan, and D. Luo, "Semantic neighborhood-aware deep facial expression recognition," *Transactions on image processing*, 2020.

[55] G. Ghiasi and C. C. Fowlkes, "Occlusion coherence: Localizing occluded faces with a hierarchical deformable part model," in *CVPR*, 2014.

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

[57] J. C. Gower, "Generalized procrustes analysis," *Psychometrika*, 1975.

[58] H. Grabner and H. Bischof, "On-line boosting and vision," in *CVPR*, 2006.

[59] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, 2003.

[60] B. Hasani, P. S. Negi, and M. H. Mahoor, "Bounded residual gradient networks (breg-net) for facial affect computing," in *FG*, 2019.

[61] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

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

[63] T. K. Ho, "The random subspace method for constructing decision forests," *Transactions on pattern analysis and machine intelligence*, 1998.

[64] S. Honari, P. Molchanov, S. Tyree, P. Vincent, C. Pal, and J. Kautz, "Improving landmark localization with semi-supervised learning," in *CVPR*, 2018.

[65] X. Hou, S. Z. Li, H. Zhang, and Q. Cheng, "Direct appearance models," in *CVPR*, 2001.

[66] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018.

[67] P. Hu, D. Cai, S. Wang, A. Yao, and Y. Chen, "Learning supervised scoring ensemble for emotion recognition in the wild," in *ICMI*, 2017.

[68] W. Hua, Y. Zhou, C. M. De Sa, Z. Zhang, and G. E. Suh, "Channel gating neural networks," in *NIPS*, 2019.

[69] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get M for free," in *ICLR*, 2017.

[70] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *ECCV*, 2016.

[71] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, 2007.

[72] X. Huang, G. Zhao, W. Zheng, and M. Pietikäinen, "Towards a dynamic expression recognition system under facial occlusion," *Pattern recognition letters*, 2012.

[73] E. P. Ijjina and C. K. Mohan, "Hybrid deep neural network model for human action recognition," *Applied soft computing*, 2016.

[74] Y. Ioannou, D. Robertson, D. Zikic, P. Kontschieder, J. Shotton, M. Brown, and A. Criminisi, "Decision forests, convolutional networks and the models in-between," Microsoft Research, Tech. Rep. MSR-TR-2015-58, 2015.

[75] R. A. Jacobs, "Methods for combining experts' probability assessments," *Neural computation*, 1995.

[76] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, 1991.

[77] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *NIPS workshops*, 2016.

[78] L. A. Jeni, J. M. Girard, J. F. Cohn, and F. De La Torre, "Continuous au intensity estimation using localized, sparse facial feature space," in *FG*, 2013.

[79] X. Jin and X. Tan, "Face alignment in-the-wild: A survey," *Computer vision and image understanding*, 2017.

[80] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural computation*, 1994.

[81] A. Jourabloo and X. Liu, "Large-pose face alignment via cnn-based dense 3d model fitting," in *CVPR*, 2016.

[82] A. Jourabloo, M. Ye, X. Liu, and L. Ren, "Pose-invariant face alignment with a single cnn," in *ICCV*, 2017.

[83] B.-N. Kang, Y. Kim, B. Jun, and D. Kim, "Hierarchical feature-pair relation networks for face recognition," in *CVPR workshops*, 2019.

[84] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *CVPR*, 2014.

[85] E. Kim, H. Song, and J. W. Shin, "Affective latent representation of acoustic and lexical features for emotion recognition," *Sensors*, 2020.

[86] B.-K. Kim, H. Lee, J. Roh, and S.-Y. Lee, "Hierarchical committee of deep cnns with exponentially-weighted decision fusion for static facial expression recognition," in *ICMI*, 2015.

[87] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[88] B. Knyazev, R. Shvetsov, N. Efremova, and A. Kuharenko, "Convolutional neural networks pretrained on large face recognition datasets for emotion classification from video," in *ICMI*, 2017.

[89] M. Koestinger, P. Wohlhart, P. M. Roth, and H. Bischof, "Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization," in *ICCV workshops*, 2011.

[90] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo, "Deep neural decision forests," in *ICCV*, 2015.

[91] I. Kotsia, I. Buciu, and I. Pitas, "An analysis of facial expression recognition under partial facial image occlusion," *Image and vision computing*, 2008.

[92] A. Kumar and R. Chellappa, "Disentangling 3d pose in a dendritic cnn for unconstrained 2d face alignment," in *CVPR*, 2018.

[93] L. I. Kuncheva, "Clustering-and-selection model for classifier combination," in *KES*, 2000.

[94] ——, *Combining pattern classifiers: methods and algorithms*. 2014.

[95] Y.-H. Lai and S.-H. Lai, "Emotion-preserving representation learning via generative adversarial network for multi-view facial expression recognition," in *FG*, 2018.

[96] S. Le Gallou, "Détection robuste des éléments faciaux par modèles actifs d'apparence," PhD thesis, 2007.

[97] Y. LeCun, C. Cortes, and C. J. Burges, *The mnist database of handwritten digits*, http://yann.lecun.com/exdb/mnist, 1998.

[98] J. Li, C. Soladie, and R. Seguier, "Ltp-ml: Micro-expression detection by recognition of local temporal pattern of facial movements," in *FG*, 2018.

[99] S. Li and W. Deng, "Deep facial expression recognition: A survey," *Transactions on affective computing*, 2020.

[100] S. Li, W. Deng, and J. Du, "Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild," in *CVPR*, 2017.

[101] Y. Li, Y. Lu, J. Li, and G. Lu, "Separate loss for basic and compound facial expression recognition in the wild," in *ACML*, 2019.

[102] Y. Li, J. Zeng, S. Shan, and X. Chen, "Patch-gated cnn for occlusion-aware facial expression recognition," in *ICPR*, 2018.

[103] Y. Li, S. Wang, Y. Zhao, and Q. Ji, "Simultaneous facial feature tracking and facial expression recognition," *Transactions on image processing*, 2013.

[104] Z. Lian, Y. Li, J.-H. Tao, J. Huang, and M.-Y. Niu, "Expression analysis based on face regions in read-world conditions," *International journal of automation and computing*, 2020.

[105] C. Lindner, P. A. Bromiley, M. C. Ionita, and T. F. Cootes, "Robust and accurate shape model matching using random forest regression-voting," *Transactions on pattern analysis and machine intelligence*, 2014.

[106] X. Liu, B. V. Kumar, P. Jia, and J. You, "Hard negative generation for identity-disentangled facial expression recognition," *Pattern recognition*, 2019.

[107] X. Liu, B. V. Kumar, J. You, and P. Jia, "Adaptive deep metric learning for identity-aware facial expression recognition," in *CVPR workshops*, 2017.

[108] Y. Liu, J. Zeng, S. Shan, and Z. Zheng, "Multi-channel pose-aware convolution neural networks for multi-view facial expression recognition," in *FG*, 2018.

[109] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," in *ICLR*, 2017.

[110] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, 2004.

[111] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression," in *CVPR workshops*, 2010.

[112] D. C. Luvizon, H. Tabia, and D. Picard, "Human pose regression by combining indirect part detection and contextual information," *Computers & Graphics*, 2019.

[113] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *FG*, 1998.

[114] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, 2008.

[115] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Berkeley symposium on mathematical statistics and probability*, 1967.

[116] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *ICLR*, 2017.

[117] R. C. Malli, *Oxford vggface implementation using keras functional framework v2+*, https://github.com/rcmalli/keras-vggface, 2017.

[118] B. Martinez, M. F. Valstar, B. Jiang, and M. Pantic, "Automatic analysis of facial actions: A survey," *Transactions on affective computing*, 2017.

[119] I. Matthews and S. Baker, "Active appearance models revisited," *International journal of computer vision*, 2004.

[120] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner, *Dsprites: Disentanglement testing sprites dataset*, https://github.com/deepmind/dsprites-dataset, 2017.

[121] A. Mehrabian and S. R. Ferris, "Inference of attitudes from nonverbal communication in two channels," *Journal of consulting psychology*, 1967.

[122] Z. Meng, P. Liu, J. Cai, S. Han, and Y. Tong, "Identity-aware convolutional neural network for facial expression recognition," in *FG*, 2017.

[123] A. Mollahosseini, B. Hasani, and M. H. Mahoor, "Affectnet: A database for facial expression, valence, and arousal computing in the wild," *Transactions on affective computing*, 2017.

[124] S. Moore and R. Bowden, "Local binary patterns for multi-view facial expression recognition," *Computer vision and image understanding*, 2011.

[125] H.-W. Ng, V. D. Nguyen, V. Vonikakis, and S. Winkler, "Deep learning for emotion recognition on small datasets using transfer learning," in *ICMI*, 2015.

[126] V. Ojansivu and J. Heikkilä, "Blur insensitive texture classification using local phase quantization," in *ICISP*, 2008.

[127] L. Oliveira, U. Nunes, and P. Peixoto, "On exploration of classifier ensemble synergism in pedestrian detection," *Transactions on intelligent transportation systems*, 2009.

[128] C. Orrite, A. Gañán, and G. Rogez, "Hog-based decision tree for facial expression classification," in *IbPRIA*, 2009.

[129] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *BMVC*, 2015.

[130] R. A. Patil, V. Sahula, and A. S. Mandal, "Facial expression recognition in image sequences using active shape model and svm," in *UKSim*, 2011.

[131] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of machine learning research*, 2011.

[132] G. Pons and D. Masip, "Multi-task, multi-label and multi-domain learning with residual convolutional networks for emotion recognition," *arXiv preprint arXiv:1802.06664*, 2018.

[133] A. Porras Garrido, *What is the difference between bagging and boosting?* `https://quantdare.com/what-is-the-difference-between-bagging-and-boosting`, 2010.

[134] S. Qiao, L.-C. Chen, and A. Yuille, "Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution," *arXiv preprint arXiv:2006.02334*, 2020.

[135] R. Ranjan, S. Sankaranarayanan, C. D. Castillo, and R. Chellappa, "An all-in-one convolutional neural network for face analysis," in *FG*, 2017.

[136] M. Ranzato, J. Susskind, V. Mnih, and G. Hinton, "On deep generative models with applications to recognition," in *CVPR*, 2011.

[137] S. Ren, X. Cao, Y. Wei, and J. Sun, "Face alignment at 3000 fps via regressing local binary features," in *CVPR*, 2014.

[138] N. Ruiz, E. Chong, and J. M. Rehg, "Fine-grained head pose estimation without keypoints," in *CVPR workshops*, 2018.

[139] J. A. Russell, "A circumplex model of affect," *Journal of personality and social psychology*, 1980.

[140] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. P. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: Database and results," *Image and vision computing*, 2016.

[141] J. Saragih and R. Goecke, "A nonlinear discriminative approach to aam fitting," in *ICCV*, 2007.

[142] E. Sariyanidi, H. Gunes, and A. Cavallaro, "Automatic analysis of facial affect: A survey of registration, representation, and recognition," *Transactions on pattern analysis and machine intelligence*, 2014.

[143] T. Senechal, V. Rapp, H. Salam, R. Seguier, K. Bailly, and L. Prevost, "Combining aam coefficients with lgbp histograms in the multi-kernel svm framework to detect facial action units," in *FG*, 2011.

[144] ——, "Facial action recognition combining heterogeneous features via multikernel learning," *Transactions on systems, man, and cybernetics*, 2012.

[145] C. Shan, S. Gong, and P. W. McOwan, "Facial expression recognition based on local binary patterns: A comprehensive study," *Image and vision computing*, 2009.

[146] Z. Shao, Z. Liu, J. Cai, and L. Ma, "Deep adaptive attention for joint facial action unit detection and face alignment," in *ECCV*, 2018.

[147] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *ICLR*, 2017.

[148] K. Sikka, T. Wu, J. Susskind, and M. Bartlett, "Exploring bag of words architectures in the facial expression domain," in *ECCV*, 2012.

[149] M. Skurichina and R. P. Duin, "Bagging for linear classifiers," *Pattern recognition*, 1998.

[150] ——, "Bagging and the random subspace method for redundant feature spaces," in *MCS workshops*, 2001.

[151] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of machine learning research*, 2014.

[152] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," in *ICML workshops*, 2015.

[153] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *CVPR*, 2013.

[154] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[155] R. Tanno, K. Arulkumaran, D. Alexander, A. Criminisi, and A. Nori, "Adaptive neural trees," in *ICML*, 2019.

[156] R. J. Tibshirani and B. Efron, "An introduction to the bootstrap," *Monographs on statistics and applied probability*, 1993.

[157] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *CVPR*, 2015.

[158] F. Torre and J. F. Cohn, "Facial expression analysis," *Visual analysis of humans*, 2011.

[159] G. Trigeorgis, P. Snape, M. A. Nicolaou, E. Antonakos, and S. Zafeiriou, "Mnemonic descent method: A recurrent process applied for end-to-end face alignment," in *CVPR*, 2016.

[160] G. Tzimiropoulos, J. Alabort-i-Medina, S. P. Zafeiriou, and M. Pantic, "Active orientation models for face alignment in-the-wild," *Transactions on information forensics and security*, 2014.

[161] G. Tzimiropoulos and M. Pantic, "Optimization problems for fast aam fitting in-the-wild," in *ICCV*, 2013.

[162] ——, "Fast algorithms for fitting active appearance models to unconstrained images," *International journal of computer vision*, 2017.

[163] N. Ueda and R. Nakano, "Generalization error of ensemble estimators," in *ICNN*, 1996.

[164] M. F. Valstar, B. Jiang, M. Mehu, M. Pantic, and K. Scherer, "The first facial expression recognition and analysis challenge," in *FG*, 2011.

[165] A. Veit and S. Belongie, "Convolutional networks with adaptive inference graphs," in *ECCV*, 2018.

[166] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *NIPS*, 2016.

[167] R.-L. Vieriu, S. Tulyakov, S. Semeniuta, E. Sangineto, and N. Sebe, "Facial expression recognition under a wide range of head poses," in *FG*, 2015.

[168] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *ICML*, 2013.

[169] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *CVPR*, 2017.

[170] K. Wang, X. Peng, J. Yang, D. Meng, and Y. Qiao, "Region attention networks for pose and occlusion robust facial expression recognition," *Transactions on image processing*, 2020.

[171] X. Wang, F. Yu, L. Dunlap, Y.-A. Ma, R. Wang, A. Mirhoseini, T. Darrell, and J. E. Gonzalez, "Deep mixture of experts via shallow embedding," in *UAI*, 2020.

[172] G. Wen, Z. Hou, H. Li, D. Li, L. Jiang, and E. Xun, "Ensemble of deep neural networks with probability-based fusion for facial expression recognition," *Cognitive computation*, 2017.

[173] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *ECCV*, 2016.

[174] D. H. Wolpert, "Stacked generalization," *Neural networks*, 1992.

[175] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *ECCV*, 2018.

[176] Y. Wu, C. Gou, and Q. Ji, "Simultaneous facial landmark detection, pose and deformation estimation under facial occlusion," in *CVPR*, 2017.

[177] Y. Wu and Q. Ji, "Robust facial landmark detection under significant head poses and occlusion," in *ICCV*, 2015.

[178] ——, "Constrained joint cascade regression framework for simultaneous facial action unit recognition and facial landmark detection," in *CVPR*, 2016.

[179] ——, "Facial landmark detection: A literature survey," *International journal of computer vision*, 2019.

[180] X. Xiong and F. De la Torre, "Supervised descent method and its applications to face alignment," in *CVPR*, 2013.

[181] H. Yang and I. Patras, "Privileged information-based conditional regression forest for facial feature detection," in *FG workshops*, 2013.

[182] H. Yang, U. Ciftci, and L. Yin, "Facial expression recognition by de-expression residue learning," in *CVPR*, 2018.

[183] H. Yang, Z. Zhang, and L. Yin, "Identity-adaptive facial expression recognition through expression regeneration using conditional generative adversarial networks," in *FG*, 2018.

[184] L. Yin, X. Chen, Y. Sun, T. Worm, M. Reale, and A. High-resolution, "A high-resolution 3d dynamic facial expression database," in *FG*, 2008.

[185] L. Yin, X. Wei, Y. Sun, J. Wang, and M. J. Rosato, "A 3d facial expression database for facial behavior research," in *FG*, 2006.

[186] X. Yu, Z. Lin, J. Brandt, and D. N. Metaxas, "Consensus of regression for occlusion-robust facial feature localization," in *ECCV*, 2014.

[187] L. Yue, X. Miao, P. Wang, B. Zhang, X. Zhen, and X. Cao, "Attentional alignment networks.," in *BMVC*, 2018.

[188] J. Zeng, S. Shan, and X. Chen, "Facial expression recognition with inconsistently annotated datasets," in *ECCV*, 2018.

[189] F. Zhang, T. Zhang, Q. Mao, and C. Xu, "Joint pose and expression modeling for facial expression recognition," in *CVPR*, 2018.

[190] J. Zhang, M. Kan, S. Shan, and X. Chen, "Occlusion-free face alignment: Deep regression networks coupled with de-corrupt autoencoders," in *CVPR*, 2016.

[191] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Facial landmark detection by deep multi-task learning," in *ECCV*, 2014.

[192] ——, "Learning deep representation for face alignment with auxiliary attributes," *Transactions on pattern analysis and machine intelligence*, 2015.

[193] ——, "From facial expression recognition to interpersonal relation prediction," *International journal of computer vision*, 2018.

[194] G. Zhao, X. Huang, M. Taini, S. Z. Li, and M. PietikäInen, "Facial expression recognition from near-infrared videos," *Image and vision computing*, 2011.

[195] L. Zhong, Q. Liu, P. Yang, B. Liu, J. Huang, and D. N. Metaxas, "Learning active facial patches for expression analysis," in *CVPR*, 2012.

[196] E. Zhou, H. Fan, Z. Cao, Y. Jiang, and Q. Yin, "Extensive facial landmark localization with coarse-to-fine convolutional network cascade," in *ICCV workshops*, 2013.

[197] S. Zhu, C. Li, C. Change Loy, and X. Tang, "Face alignment by coarse-to-fine shape searching," in *CVPR*, 2015.

[198] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *CVPR*, 2012.

[199] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li, "Face alignment across large poses: A 3d solution," in *CVPR*, 2016.

[200] X. Zhu, X. Liu, Z. Lei, and S. Z. Li, "Face alignment in full pose range: A 3d total solution," *Transactions on pattern analysis and machine intelligence*, 2017.