RAPPORT DE TER: ONE-SHOT LEARNING POUR LA RECONNAISSANCE FACIALE

Lounas CHIKI

Sosthene MEDRID

Thanina SALHI

Akil MOUHOUB

Département MIASHS, UFR 6 Informatique, Mathématique et Statistique

Université de Montpellier Paul Valéry

Soumis comme contribution au Module TER

Année universitaire 2025–2026

Remerciements

Nous tenons à exprimer nos sincères remerciements à toutes les personnes qui ont contribué, à la réalisation de ce projet TER.

Nous remercions tout particulièrement notre encadrant, M. PASQUET, maître de conférences pour son accompagnement, ses conseils avisés et sa disponibilité tout au long du projet. Son expertise et ses orientations ont été déterminantes dans les choix méthodologiques effectués.

Nous tenons aussi à remercier l'ensemble de notre promotion M1 MIASHS pour leur participation et leur consentement à les prendre en photo pour le projet.

Résumé

Dans le cadre de ce projet TER, nous nous sommes intéressés à la problématique de la reconnaissance faciale en contexte one-shot learning, c'est-à-dire dans une situation où une seule image de référence est disponible par personne.
Notre travail s'est articulé autour de trois axes principaux : la détection des visages, l'extraction de caractéristiques (apprentissage des visages) et leur reconnaissance. Ce rapport détaille ainsi la démarche adoptée, les outils utilisés (YOLOv5, SIFT, ProtoNet), les choix techniques effectués, ainsi que les résultats obtenus. Il met également en perspective les limites et propose des pistes d'amélioration pour des futurs travaux.

Table des matières

Remerciements **2**

Résumé **3**

I Introduction **8**

II Problématique **9**

III Etat de l'art **10**
III.1 Détection des Visages 10
III.2 Reconnaissance Faciale : Cadre Général 10
III.3 Le Paradigme du Méta Learning 10
a) Le Concept de Méta-Apprentissage : Apprendre à Apprendre 11
b) Structure d'un Épisode (Tâche) : Support Set et Query Set 11
c) L'Apprentissage Métrique (*Metric Learning*) 11
d) Fonctionnement des Réseaux Prototypiques (ProtoNet) 12
e) Autres Approches de Méta-Apprentissage 12
III.4 Positionnement de Notre Travail 13

IV YOLO **14**
IV.1 Présentation Générale 14
IV.2 Principe de fonctionnement 14
a) Architecture 14
b) Métriques de mesure 15
IV.3 Fine-tuning 17
a) Pré-requis / Outils 17
b) Comparaison des métriques : avant et après fine-tuning 18
IV.4 Cas d'étude : Promotion M1 MIASHS Université paul Valéry 19

V SIFT **21**
V.1 1. Présentation générale de la méthode 21
V.2 2. Outils et environnement de travail 21
V.3 3. Données utilisées 21
V.4 4. Détection des visages 21
V.5 5. Prétraitement et affichage 22
V.6 6. Extraction des caractéristiques SIFT 23
V.7 7. Matching des descripteurs 23
V.8 8. Filtrage géométrique par RANSAC 24
V.9 9. Calcul du score par identité et classement Top-3 24
V.10 10. Décision connu / inconnu et évaluation 26
V.11 11. Conclusion 26

VI Approche moderne : ProtoNet **27**
VI.1 Principe du Méta-Apprentissage et de la Métrique 27
VI.2 Architecture de l'Encodeur 27
a) **CNN** 27
b) **ResNet-18 pré-entraîné** 28

VI.3 Fonction de perte, Optimiseur et Scheduler . . . 29
VI.4 Stratégie d'Entraînement . . . 29
VI.5 Pipeline d'Application et Évaluation . . . 31
a) Pré-traitement et Détection Faciale . . . 31
b) Phase de Référence One-Shot (Enrollment) . . . 31
c) Phase de Reconnaissance et Décision . . . 31

VII Résultats et Visualition de Données **32**
VII.1 Configuration du méta-entraînement . . . 32
VII.2 Performance du Méta-train et du Méta-test sur CelebA . . . 32
a) Méta-train . . . 32
b) Évaluation en méta-test . . . 34
VII.3 Analyse de l'Espace d'Embedding (ACP et t-SNE) . . . 34
a) Visualisation par ACP . . . 35
b) Visualisation par t-SNE . . . 35
VII.4 Évaluation Finale sur le Cas d'Usage Réel . . . 36
a) Top-K Accuracy . . . 36
b) Résultats Biométriques et Seuil Optimal . . . 37

VIII Difficultés rencontrées et perspectives **39**
VIII.1 Résultats obtenus avec YOLO, ProtoNet et SIFT . . . 39
VIII.2 Difficultés techniques rencontrées . . . 39
VIII.3 Perspectives et travaux futurs . . . 39

IX Conclusion **41**

X Bibliographie **42**

Table des figures

1 N-way K-shot learning problem description 12
2 Meta learning algorithms 13
3 Détection de visages YOLO et Ground truth. 19
4 Exemple de détection de visage à l'aide du modèle YOLOv5-face. 22
5 Affichage du visage de Lounes après prétraitement. 23
6 Points clés détectés sur le visage de Lounes à l'aide de l'algorithme SIFT. 24
7 Correspondances obtenues entre une image de test et une image de référence à l'aide du matching SIFT. 25
8 Structure générale d'un bloc convolutif. Chaque bloc est composé d'une convolution, d'une normalisation, d'une activation et d'une opération de pooling. 28
9 Architecture générale de ResNet-18 utilisée comme encodeur. 29
10 Illustration du fonctionnement d'un Prototypical Network. 30
11 Méta-train obtenu sur 20 classes (N-way) et 1-Shot. 33
12 Méta-test sur 4 scénarios différents. 34
13 Projection 2D des embeddings (128D) via ACP. 35
14 Projection 2D des embeddings (128D) via t-SNE. 36
15 Courbe ROC du système ProtoNet. 37

Liste des tableaux

1 Comparaison des métriques de performance du modèle YOLOv5m avant et après fine-tuning 18
2 Métriques sur les photos en conditions réelles de la promotions 19
3 ProtoNet Top-K accuracy sur photo de la promotion M1 MIASHS 37
4 Mesure - Courbe ROC 38

I Introduction

Au cours de la dernière décennie, la reconnaissance faciale a connu un fort développement et s'est intégrée dans de nombreux usages du quotidien. Notamment grâce aux avancées de l'apprentissage profond et des réseaux de neurones convolutifs. Toutefois, ces approches reposent souvent sur de grandes quantités de données annotées, ce qui limite leur applicabilité dans certains contextes réels.

Dans le cadre de ce projet TER, notre commanditaire est un enseignant souhaitant automatiser la prise de présence en classe. L'idée est de pouvoir capturer une photo de groupe et d'identifier automatiquement les étudiants présents afin de simplifier et fiabiliser le processus d'appel. Ce cas d'usage constitue une application concrète de la reconnaissance faciale dans un environnement réel.

II Problématique

Dans le scénario considéré, le défi est la disponibilité très limitée des photos par étudiant. En pratique, une seule image de référence est disponible pour chaque individu, ce qui rend les approches classiques de reconnaissance faciale difficiles à appliquer.
De plus, les images de groupe peuvent présenter plusieurs problèmes supplémentaires, tels que la variation des poses, des expressions faciales, des conditions d'éclairage, ainsi que la présence éventuelle d'occlusions. Ces facteurs compliquent à la fois la détection des visages et leur reconnaissance.
La problématique centrale de ce projet est donc la suivante :

L'étude d'une pipeline de reconnaissance faciale capable d'identifier les étudiants à partir d'une photo de groupe, en ne disposant que d'une seule image de référence par personne Ce contexte s'inscrit naturellement dans le cadre du **one-shot learning**, qui vise à apprendre à reconnaître des identités à partir d'un seul exemple.

L'objectif principal de ce projet est de concevoir et d'évaluer une chaîne complète de reconnaissance faciale adaptée au cadre du one-shot learning. Concrètement, ce travail vise à :

- mettre en œuvre un module de détection faciale robuste sur des images de groupe ;
- comparer une approche classique basée sur des descripteurs locaux (SIFT) à une approche moderne de méta-apprentissage (Prototypical Networks) ;
- analyser les performances, les limites et difficultés de chaque méthode .

III Etat de l'art

III.1 Détection des Visages

La détection des visages constitue une étape préalable indispensable à tout système de reconnaissance faciale. Elle vise à localiser automatiquement les régions contenant des visages dans une image, en particulier dans des scènes complexes telles que les photos de groupe. La qualité de cette étape conditionne directement les performances de la reconnaissance, car une détection imprécise peut entraîner l'extraction de caractéristiques non pertinentes.
Les méthodes historiques de détection faciale reposaient sur des approches basées sur des caractéristiques manuelles, comme l'algorithme de Viola-Jones. Bien que rapides, ces méthodes présentent des limitations face aux variations importantes de pose, d'éclairage ou d'échelle.
Les approches modernes s'appuient sur des modèles d'apprentissage profond capables de détecter les visages de manière plus robuste. Parmi elles, les architectures de type *You Only Look Once* (YOLO) se distinguent par leur capacité à effectuer une détection rapide et précise, même en présence de plusieurs visages dans une même image. Leur formulation en détection d'objets permet de traiter efficacement les images de groupe et de répondre aux contraintes de temps réel, ce qui en fait des candidats pertinents pour des applications pratiques.

III.2 Reconnaissance Faciale : Cadre Général

La reconnaissance faciale est une tâche fondamentale de la vision par ordinateur visant à identifier ou vérifier l'identité d'un individu à partir d'une image ou d'une séquence vidéo. Elle repose généralement sur deux étapes principales : l'extraction de caractéristiques discriminantes à partir du visage, puis la comparaison de ces représentations afin de déterminer une correspondance entre individus.
Les premières approches de reconnaissance faciale se fondaient sur des méthodes classiques d'extraction de caractéristiques, telles que les descripteurs locaux ou globaux, combinées à des mesures de similarité. Plus récemment, les avancées de l'apprentissage profond ont permis de concevoir des modèles capables d'apprendre automatiquement des représentations de haut niveau, conduisant à des performances très élevées dans des conditions contrôlées.
Cependant, ces méthodes modernes reposent souvent sur l'entraînement de modèles complexes nécessitant de grandes quantités de données annotées par identité. Cette contrainte limite leur applicabilité dans des scénarios réels où peu d'exemples sont disponibles, comme c'est le cas dans les systèmes d'identification ponctuelle ou personnalisée. Ce constat a motivé l'émergence d'approches alternatives capables de fonctionner efficacement avec un nombre réduit d'images par individu.

III.3 Le Paradigme du Méta Learning

Le *Few-Shot Learning* représente une rupture avec l'apprentissage supervisé classique. Alors que les modèles profonds standards nécessitent des milliers d'exemples par classe pour converger, le Few-Shot Learning vise à apprendre à partir d'un nombre infime d'échantillons, typiquement entre 1 et 5 images par individu. Cette contrainte reflète mieux les scénarios réels où la collecte massive de données pour chaque nouvelle personne n'est pas praticable.

a) Le Concept de Méta-Apprentissage : Apprendre à Apprendre

L'idée centrale du *Few-Shot Learning* est le *Learning to Learn*. Au lieu d'apprendre à reconnaître des visages spécifiques pendant l'entraînement, le modèle apprend une stratégie de comparaison robuste et généralisable. Cette capacité d'adaptation rapide est acquise en exposant le modèle à de nombreuses tâches d'apprentissage différentes. On divise le processus en deux phases distinctes :

- **Phase de Méta-Entraînement (*Meta-Training*)**
  Le modèle est exposé à une multitude de petites tâches de classification appelées *épisodes*. Chaque épisode simule une situation de *few-shot* en présentant un petit nombre d'exemples par classe. Le modèle apprend à extraire des caractéristiques discriminantes qui généralisent bien au-delà des classes d'entraînement spécifiques.

- **Phase de Méta-Test (*Meta-Testing*)**
  Le modèle est confronté à de nouvelles classes qu'il n'a jamais vues durant l'entraînement. Sa capacité à les identifier correctement avec seulement quelques exemples mesure la qualité du méta-apprentissage. C'est ici que se révèle la force de l'approche : le modèle utilise sa connaissance générale acquise pour s'adapter rapidement à ces nouvelles identités.

b) Structure d'un Épisode (Tâche) : Support Set et Query Set

Chaque étape d'apprentissage est structurée sous forme d'un épisode (une tâche) afin de simuler authentiquement la contrainte du monde réel. Un épisode est défini par deux ensembles complémentaires :

- **Le Support Set (S)**
  C'est la base de connaissances limitée dont dispose le modèle. Dans une configuration *N-way K-shot*, le support contient N classes avec K exemples chacune. Par exemple, dans un scénario *5-way 1-shot*, le support contient 5 personnes différentes avec une seule image de référence par personne.

- **Le Query Set (Q)**
  Ce sont les images de test de l'épisode. Le modèle doit prédire la classe de ces images en se basant uniquement sur les informations du *Support Set*, mimant ainsi une situation d'identification réelle où l'on compare un visage inconnu à une petite galerie de références.

Cette structure épisodique force le modèle à développer des capacités de généralisation plutôt que de mémoriser des identités spécifiques.

c) L'Apprentissage Métrique (*Metric Learning*)

Le *Few-Shot Learning* s'appuie majoritairement sur l'apprentissage métrique. L'objectif est d'entraîner un encodeur f_ϕ (dans notre cas, un ResNet-18) pour projeter les images dans un espace de faible dimension appelé *espace d'embedding*. La fonction de perte force le réseau à respecter deux contraintes géométriques complémentaires :

1. **Réduire la distance entre les images d'une même personne (cohésion intra-classe)**
   Les différentes photos d'un individu doivent former un cluster compact.

2. **Augmenter la distance entre les images de personnes différentes (séparation inter-classe)**
   Les clusters de personnes distinctes doivent être bien séparés.

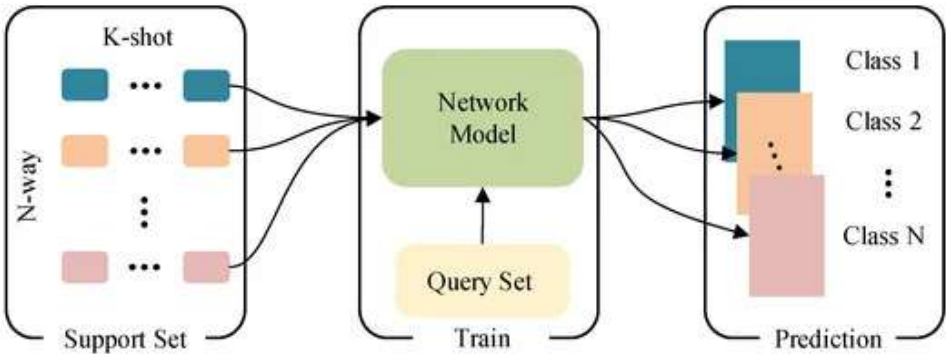


Figure 1: N-way K-shot learning problem description

d) Fonctionnement des Réseaux Prototypiques (ProtoNet)

L'algorithme **ProtoNet**, que nous avons privilégié dans ce projet, simplifie la classification en calculant des prototypes. Il repose sur l'hypothèse qu'il existe dans l'espace d'embedding une représentation prototypique de chaque classe, autour de laquelle gravitent tous les exemples de cette classe.

d).1 Calcul du Prototype

Pour chaque classe n dans le *support set*, le prototype c_n est défini comme le vecteur moyen (centroïde) des embeddings des K images de cette classe :

$$c_n = \frac{1}{K} \sum_{x_i \in S_n} f_\phi(x_i)$$

Cette moyenne capture l'essence de la classe en agrégeant l'information des exemples disponibles, rendant le prototype plus robuste que n'importe quel exemple individuel.

d).2 Classification par Distance

Pour une image de requête x_q, le modèle calcule la distance euclidienne d par rapport à chaque prototype c_n. La probabilité que la requête appartienne à la classe n est obtenue par une fonction *softmax* appliquée aux distances négatives :

$$p(y = n \mid x_q) = \frac{\exp(-d(f_\phi(x_q), c_n))}{\sum_{n'} \exp(-d(f_\phi(x_q), c_{n'}))}$$

où :

$$d(a, b) = \|a - b\|^2$$

représente le carré de la distance euclidienne.

e) Autres Approches de Méta-Apprentissage

Bien que nous ayons retenu **ProtoNet** pour sa simplicité et ses performances robustes, d'autres approches notables existent, se distinguant principalement par leur stratégie d'optimisation :

- **MAML (*Model-Agnostic Meta-Learning*)**
  - *Caractéristique globale* : recherche d'une initialisation optimale des paramètres du modèle.
  - *Principe* : apprend une initialisation θ permettant au modèle de s'adapter à une nouvelle tâche en seulement quelques étapes de gradient.
  - *Méthode* : utilise une double boucle d'optimisation pour ajuster les paramètres initiaux θ de manière à ce qu'ils soient proches des solutions optimales de nombreuses tâches.
- **Reptile**
  - *Caractéristique globale* : approximation simplifiée de MAML recherchant également une bonne initialisation.
  - *Principe* : met à jour l'initialisation du modèle en la déplaçant directement vers les paramètres finaux θ' trouvés après quelques étapes de gradient sur une tâche spécifique.
  - *Méthode* : plus simple et souvent plus rapide à implémenter que MAML.

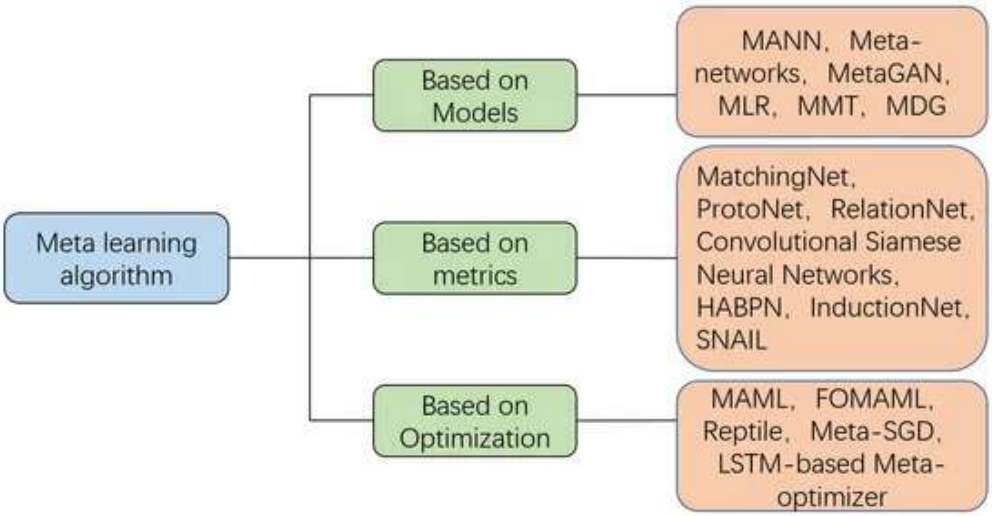


Figure 2: Meta learning algorithms

III.4 Positionnement de Notre Travail

Dans ce contexte, notre travail s'inscrit à l'intersection de la détection faciale et de la reconnaissance en situation de données limitées. Nous nous intéressons à un scénario applicatif concret dans lequel une seule image de référence est disponible par individu, ce qui rend les approches de reconnaissance faciale difficiles à appliquer.

Afin de répondre à cette problématique, nous avons étudié et comparé différentes familles de méthodes. D'une part, nous avons exploré une approche classique basée sur des descripteurs locaux, représentée par l'algorithme SIFT, afin d'analyser les limites des méthodes traditionnelles dans un cadre one-shot. D'autre part, nous nous sommes orientés vers des approches modernes de méta-apprentissage, en particulier les réseaux prototypiques (ProtoNet), conçus pour généraliser à partir d'un nombre très restreint d'exemples.

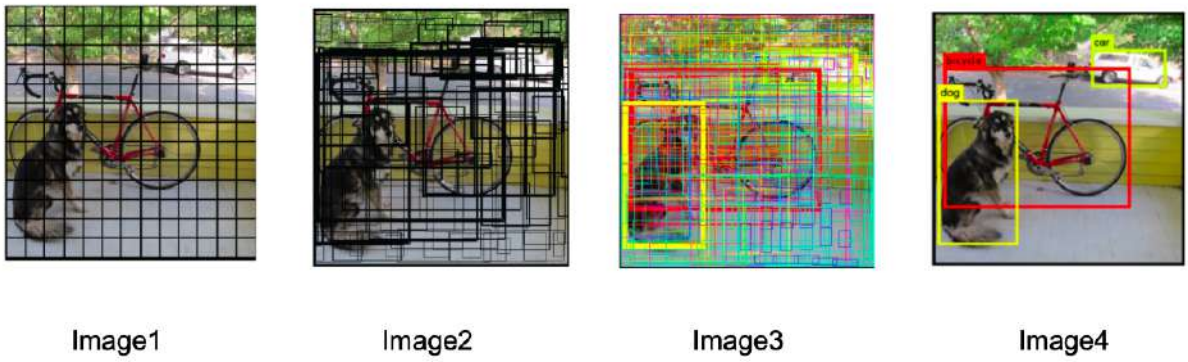
La détection des visages repose sur l'utilisation d'un modèle YOLOv5, choisi pour sa robustesse et son efficacité dans le traitement d'images de groupe. Ce positionnement permet ainsi de comparer des approches de nature différente au sein d'une chaîne de reconnaissance faciale complète, tout en mettant en évidence les enjeux et les défis propres au one-shot learning dans des conditions réelles d'utilisation.

IV YOLO

IV.1 Présentation Générale

Ce travail s'appuie sur des travaux fondateurs en vision par ordinateur (Redmon et al. 2016).
You Only Look Once (YOLO) est une famille d'algorithme de détection d'objets en temps réel introduit en 2015 par Joseph Redmon, Santosh Divvala, Ross Girshick et Ali Farhadi dans leur article de recherche You Only Look Once : Détection unifiée d'objets en temps réel.

Les auteurs présentent le problème de la détection d'objets comme une tâche de régression plutôt que de classification en séparant spatialement les boîtes de délimitation et en associant des probabilités à chaque image détectée à l'aide d'un seul réseau neuronal convolutionnel (CNN) (Keita 2024).



L'image est divisée en une grille spatiale (Image 1). Le backbone convolutif extrait des caractéristiques visuelles à partir de l'image d'entrée, qui sont ensuite utilisées pour générer de nombreuses boîtes candidates (Images 2 et 3). Après application de la suppression non maximale, seules les prédictions les plus pertinentes sont conservées (Image 4).

IV.2 Principe de fonctionnement

YOLO adopte une approche dite *single-stage*, dans laquelle la localisation et la classification des objets sont effectuées simultanément au cours d'un unique passage du réseau neuronal.

a) Architecture

a).1 Entrée et backbone convolutif

Soit une image d'entrée $I \in \mathbb{R}^{H \times W \times C}$, redimensionnée à une taille fixe (par exemple 640×640).
L'image est d'abord traitée par un **backbone convolutif**[1], dont le rôle est d'extraire des caractéristiques hiérarchiques.

Les premières couches du backbone détectent des motifs de bas niveau (bords, textures), tandis que les couches plus profondes capturent des structures sémantiques plus complexes.
Ces représentations intermédiaires servent de base à la détection finale.

[1] c'est la partie du modèle qui regarde l'image et en extrait l'information utile

a).2 Tête de détection et sorties du modèle

À la sortie du backbone, YOLO utilise une **tête de détection** qui prédit, pour chaque cellule d'une grille spatiale, un ensemble de boîtes englobantes (*bounding boxes*).

L'image est divisée en une grille de taille $S \times S$.
Pour chaque cellule, le modèle prédit B boîtes englobantes et leurs informations associées.

La sortie pour une cellule peut être formalisée comme :

$$\mathbf{y} \in \mathbb{R}^{B \times (5+C)}$$

où : - B est le nombre de boîtes par cellule, - C est le nombre de classes (80 pour COCO), - les 5 paramètres correspondent à : - x, y : coordonnées du centre de la boîte, - w, h : largeur et hauteur, - s : score de confiance (*objectness score*).

a).3 Paramétrisation des bounding boxes

Les coordonnées prédites sont relatives à la cellule de la grille.
Dans les versions modernes de YOLO, les prédictions sont exprimées relativement à des **anchor boxes** prédéfinies.

Les coordonnées finales sont obtenues par les transformations suivantes :

$$\begin{aligned} x &= \sigma(t_x) + c_x \\ y &= \sigma(t_y) + c_y \\ w &= p_w \cdot e^{t_w} \\ h &= p_h \cdot e^{t_h} \end{aligned}$$

où :

- (c_x, c_y) est la position de la cellule,
- (p_w, p_h) sont les dimensions de l'anchor box,
- t_x, t_y, t_w, t_h sont les sorties du réseau,
- σ est la fonction sigmoïde.

Cette formulation garantit des prédictions stables et positives pour les dimensions des boîtes.

a).4 Scores de classe et prédiction finale

Pour chaque boîte englobante, YOLO prédit : - un score de présence d'objet $P(\text{object})$, - une distribution de probabilité sur les classes :

$$P(class_i \mid object)$$

Le score final associé à une classe est alors :

$$Score_i = P(\text{object}) \times P(class_i \mid object)$$

b) Métriques de mesure

L'évaluation d'un modèle YOLO repose sur des métriques permettant de mesurer à la fois la qualité de la localisation et celle de la classification.

b).1 Intersection over Union (IoU)

L'Intersection over Union est une métrique mesurant le recouvrement entre une boîte prédite B_{pred} et une boîte de vérité terrain B_{gt}.

$$IoU = \frac{\text{Aire}(B_{pred} \cap B_{gt})}{\text{Aire}(B_{pred} \cup B_{gt})}$$

Une prédiction est considérée correcte si :

$$IoU \geq \tau$$

où τ est généralement fixé à 0.5.

Pertinence :
L'IoU permet de quantifier précisément la qualité de la localisation spatiale, indépendamment de la classe prédite.

b).2 Précision et rappel

La précision et le rappel évaluent respectivement la fiabilité et l'exhaustivité du modèle.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

où :

- TP : vrais positifs,
- FP : faux positifs,
- FN : faux négatifs.

Pertinence :
- une forte précision indique peu de fausses détections, - un fort rappel indique une bonne couverture des objets présents.

b).3 Average Precision (AP)

L'Average Precision est définie comme l'aire sous la courbe précision–rappel pour une classe donnée :

$$AP = \int_0^1 P(R)\, dR$$

Elle résume la performance du modèle sur l'ensemble des seuils de confiance.

b).4 Mean Average Precision (mAP)

c'est la métrique principale utilisée en détection d'objets :

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$$

où N est le nombre de classes.

Pertinence :
La mAP fournit une mesure globale des performances du modèle, intégrant à la fois la localisation, la classification et la gestion des faux positifs.

IV.3 Fine-tuning

Les versions modernes de YOLO sont généralement pré-entraînées sur le jeu de données **COCO**[2], qui contient **80 classes d'objets** couramment rencontrées (personne, voiture, animal, objet du quotidien, etc.) Le fine-tuning consiste à adapter un modèle pré-entraîné à une tâche spécifique à l'aide d'un nouveau jeu de données.
Dans notre cas, nous avons utilisé un modèle YOLO pré-entraîné afin de l'adapter à la détection de visages, une tâche différente de celle du jeu de données COCO.

a) Pré-requis / Outils

a).1 Jeu de données

Le fine-tuning a été réalisé à partir du jeu de données **WIDER FACE**, un benchmark de référence pour la détection de visages en conditions non contrôlées.
Ce jeu de données se caractérise par une grande variabilité des visages en termes de pose, d'illumination, d'occlusion et d'échelle.

La répartition des données utilisée est la suivante : - **Ensemble d'entraînement** : environ 13 000 images contenant près de 170 000 visages annotés, - **Ensemble de validation** : environ 3 200 images, - **Ensemble de test** : Images test de la promotion.

Les annotations fournissent, pour chaque visage, une boîte englobante définie par ses coordonnées spatiales.

Le jeu de données est disponible à l'adresse suivante :
Dataset WIDERFACE

a).2 Environnement matériel

Les expériences ont été réalisées sur un ordinateur personnel de type **MacBook équipé d'une puce Apple M2**, utilisé avec un système de refroidissement externe (ventilateur) afin de limiter la surchauffe lors de l'entraînement prolongé du modèle.

Bien que ce matériel ne soit pas optimisé pour l'entraînement intensif de réseaux profonds, il s'est révélé suffisant pour réaliser un fine-tuning sur un nombre limité d'époques.

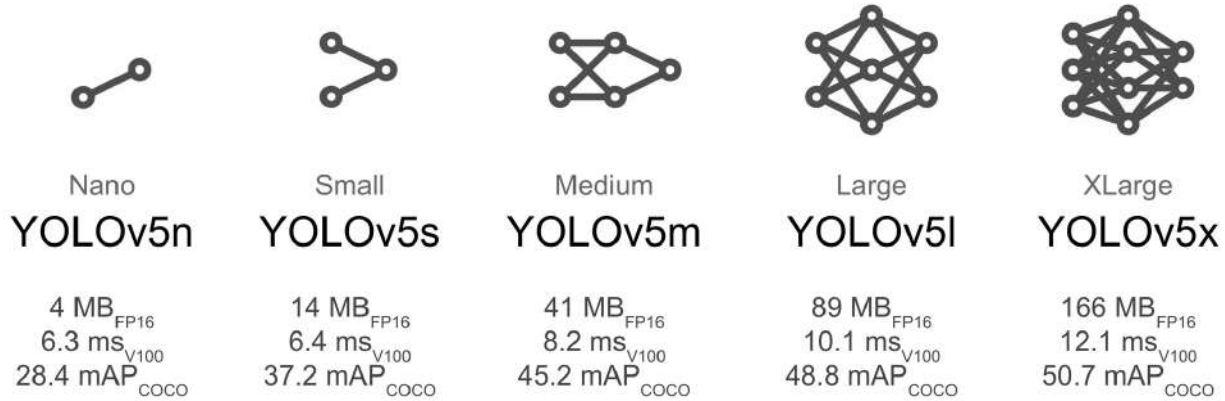
[2]est un ensemble de données de détection, de segmentation et de sous-titrage d'objets à grande échelle.

a).3 Environnement logiciel

Le modèle utilisé est **YOLOv5m**, une version intermédiaire offrant un compromis entre précision et coût computationnel.

Les étapes principales ont été les suivantes : 1. clonage du dépôt officiel Ultralytics, 2. installation des dépendances nécessaires, 3. chargement des poids pré-entraînés, 4. adaptation du format des annotations au format YOLO, 5. lancement du fine-tuning.

Le dépôt officiel est accessible à l'adresse suivante:"Dépot officiel Ultralytics YOLOv5"



La taille du modèle, exprimée en mégaoctets (MB), correspond à la taille du fichier des poids entraînés. Elle reflète directement le nombre de paramètres du réseau et constitue un indicateur du coût mémoire du modèle. Les variantes de YOLOv5 offrent ainsi différents compromis entre précision et complexité, allant de modèles légers adaptés aux systèmes embarqués à des modèles plus volumineux offrant de meilleures performances de détection.

b) Comparaison des métriques : avant et après fine-tuning

Le Tableau ci-dessous présente les performances du modèle YOLOv5m avant le fine-tuning (epoch 0) et après 37 époques d'entraînement sur le jeu de données WIDER FACE.

Table 1: Comparaison des métriques de performance du modèle YOLOv5m avant et après fine-tuning

Époque	Precision	Recall	mAP@0.5	mAP@0.5:0.95
Epoch 0	0.787	0.517	0.576	0.26
Epoch 37 (fine-tuné)	0.86	0.647	0.718	0.387

IV.4 Cas d'étude : Promotion M1 MIASHS Université paul Valéry

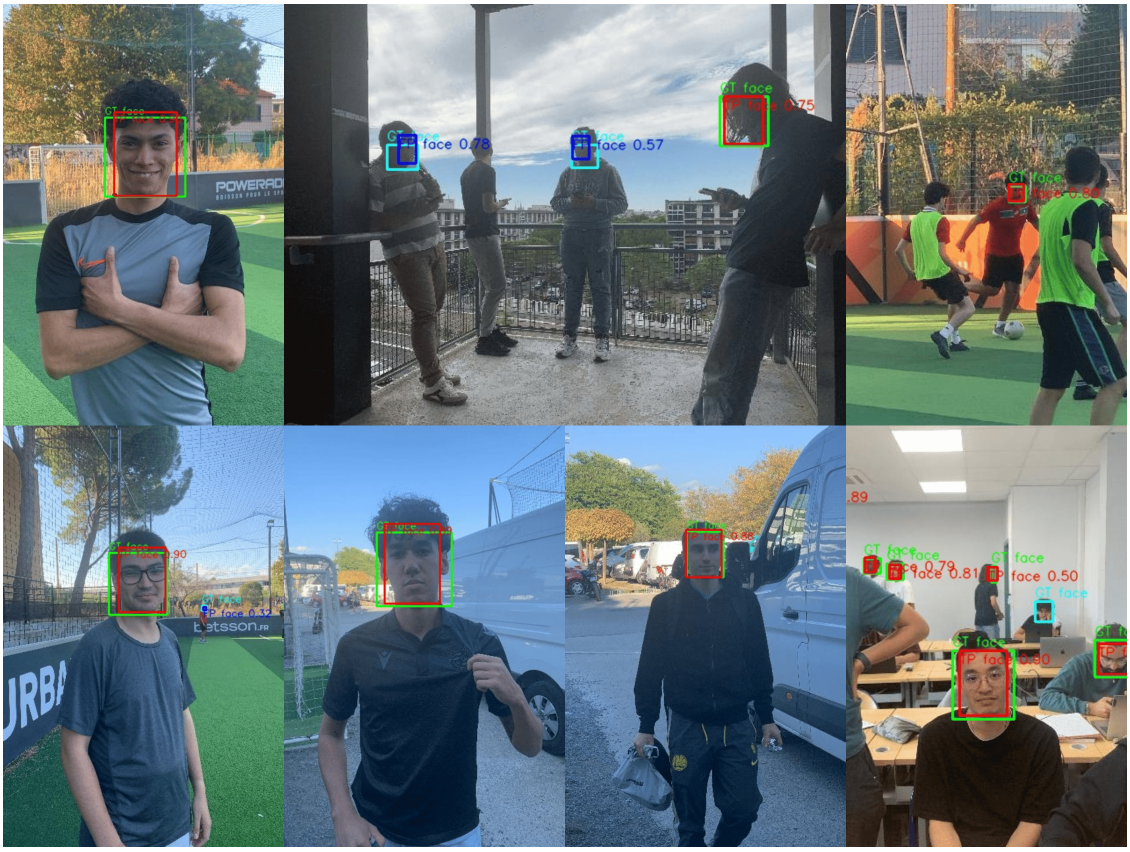


Figure 3: Détection de visages YOLO et Ground truth.

Table 2: Métriques sur les photos en conditions réelles de la promotions

Métrique	Valeur
Total Ground Truth (GT)	79
Total True Positives (TP)	71
Total False Positives (FP)	4
Total False Negatives (FN)	8
Précision (Precision)	0.947
Rappel (Recall)	0.899
Score F1	0.922
mAP@50 (approx.)	0.897
IoU moyen	0.692

La Figure ci-dessous illustre des exemples de détection de visages obtenus avec le modèle YOLO et leur comparaison avec les annotations de vérité terrain (ground truth).

Dans cette visualisation, deux types de boîtes englobantes sont affichés :
- Boîtes Ground Truth (GT): Les boîtes de couleur verte correspondent aux annotations manuelles

de référence (ground truth). Elles représentent la position réelle des visages dans les images et servent de base pour l'évaluation des performances du modèle.

- Boîtes de prédiction YOLO: Les boîtes de couleur rouge ou bleue (cyan) correspondent aux visages détectés automatiquement par le modèle YOLO. Chaque boîte de prédiction est accompagnée :
- du label "face", indiquant la classe détectée,
- d'un score de confiance (par exemple 0.78, 0.57, 0.90), qui représente la probabilité estimée par le modèle que la détection corresponde effectivement à un visage.

Lorsque la boîte de prédiction recouvre correctement la boîte de ground truth avec un chevauchement suffisant (IoU élevé), la détection est considérée comme un vrai positif (True Positive). À l'inverse : - une boîte prédite sans correspondance avec une annotation GT est comptée comme un faux positif (False Positive) ;

- un visage présent dans la ground truth mais non détecté par le modèle correspond à un faux négatif (False Negative).

On observe sur la figure que le modèle parvient à détecter correctement la majorité des visages, y compris dans des scènes complexes comportant :
- plusieurs personnes,
- des variations de pose,
- des conditions d'éclairage variables,
- des visages partiellement visibles.

Cependant, certaines prédictions présentent des scores de confiance plus faibles ou un décalage par rapport à la ground truth, ce qui explique la présence de faux positifs et de faux négatifs observés lors de l'évaluation quantitative.

V SIFT

V.1 1. Présentation générale de la méthode

L'objectif de cette méthodologie est de développer un système de reconnaissance faciale en s'appuyant sur des méthodes classiques de vision par ordinateur. La méthode proposée repose uniquement sur l'extraction et la comparaison directe de caractéristiques visuelles.(OpenAI 2023)

Le système est implémenté en Python et suit un enchaînement d'étapes bien définies :

- Détection des visages
- Prétraitement des images
- Extraction des descripteurs SIFT
- Comparaison des visages
- Filtrage des correspondances
- Évaluation des performances

Chaque étape correspond à des fonctions spécifiques présentes dans le script.

V.2 2. Outils et environnement de travail

Le développement du système est réalisé en Python à l'aide de plusieurs bibliothèques spécialisées :

- **OpenCV** : traitement d'images, extraction des points clés SIFT et matching des descripteurs
- **Ultralytics** : détection des visages via le modèle *YOLOv5m*
- **NumPy** et **Matplotlib** : manipulation des données et visualisation des résultats
- **scikit-learn** : calcul de l'accuracy pour l'évaluation des performances

V.3 3. Données utilisées

Les données sont organisées en deux ensembles distincts : un ensemble de références et un ensemble de tests.
L'expérience utilise **15 images de référence** et **3 images de test**.

Une image de test peut contenir plusieurs visages, chacun étant traité individuellement.

V.4 4. Détection des visages

La première étape du pipeline consiste à détecter automatiquement les visages présents dans les images.
La fonction `detect_faces_yolo` est utilisée pour réaliser cette tâche grâce au modèle **YOLOv5-face**.

Pour chaque image analysée, le modèle fournit des boîtes englobantes définies par les coordonnées :

$$(x_1, y_1, x_2, y_2)$$

Ces boîtes sont vérifiées afin de rester dans les dimensions de l'image. Chaque visage détecté est ensuite croppé puis redimensionné à une taille fixe :

$$\text{IMG_SIZE} = (256, 256)$$

Cette étape permet d'obtenir des images de visages homogènes avant l'extraction des caractéristiques.
Les modèles basés sur YOLO sont largement utilisés pour la détection d'objets et de visages [2].

Figure 4: Exemple de détection de visage à l'aide du modèle YOLOv5-face.

V.5 5. Prétraitement et affichage

Avant l'extraction des descripteurs, les images passent par une étape de prétraitement.
Les images sont d'abord converties du format **BGR** au format **RGB** à l'aide de la fonction `bgr2rgb` afin de garantir un affichage correct.

Les fonctions `show_img` et `show_gray` sont utilisées pour afficher les images en couleur et en niveaux de gris.
La conversion en niveaux de gris est effectuée selon la formule :

$$I_g(x, y) = 0.299R + 0.587G + 0.114B$$

Cette étape est nécessaire car l'algorithme **SIFT** fonctionne uniquement sur les niveaux d'intensité et ne prend pas en compte les couleurs [3].

Figure 5: Affichage du visage de Lounes après prétraitement.

V.6 6. Extraction des caractéristiques SIFT

La fonction `compute_sift` est utilisée pour extraire les caractéristiques des visages détectés en appliquant l'algorithme **SIFT**.

Pour chaque visage, on détecte les points clés et on calcule les descripteurs.
Chaque descripteur est un vecteur de **128 dimensions** décrivant l'information locale autour d'un point clé à partir des gradients.

Ces descripteurs sont calculés aussi bien pour les images de référence que pour les images de test et permettent la comparaison entre les visages [3].

V.7 7. Matching des descripteurs

La comparaison entre un visage de test et un visage de référence est réalisée à l'aide de la fonction `ratio_good_matches`.

Figure 6: Points clés détectés sur le visage de Lounes à l'aide de l'algorithme SIFT.

Les descripteurs sont comparés à l'aide d'un **Brute Force Matcher**, qui calcule la distance entre eux.
Pour chaque descripteur, les deux correspondances les plus proches sont recherchées.

Le **test du ratio de Lowe** est ensuite appliqué avec un seuil de **0,75** afin de conserver uniquement les correspondances les plus fiables et d'éliminer les mauvaises comparaisons [3].

V.8 8. Filtrage géométrique par RANSAC

Après le test du ratio, les correspondances sont filtrées à l'aide de l'algorithme **RANSAC** via la fonction `ransac_inliers_and_warp`.

Cette méthode permet de conserver uniquement les correspondances géométriquement cohérentes entre le visage de test et le visage de référence, et d'éliminer les correspondances incorrectes.

Une correspondance est acceptée si son erreur est inférieure au seuil fixé à **5.0**.
Le nombre de correspondances finales est ensuite utilisé pour estimer la similarité entre les visages [4].

V.9 9. Calcul du score par identité et classement Top-3

Pour chaque visage de test, le système le compare à toutes les images de référence.
Les fonctions `best_for_identity` et `top3_for_test_face` permettent de conserver, pour chaque identité, le meilleur score obtenu.

Les identités sont ensuite classées selon ces scores afin d'obtenir les **trois correspondances les plus probables (Top-3)**.
Les résultats sont affichés avec la fonction `show_top3_matches`.

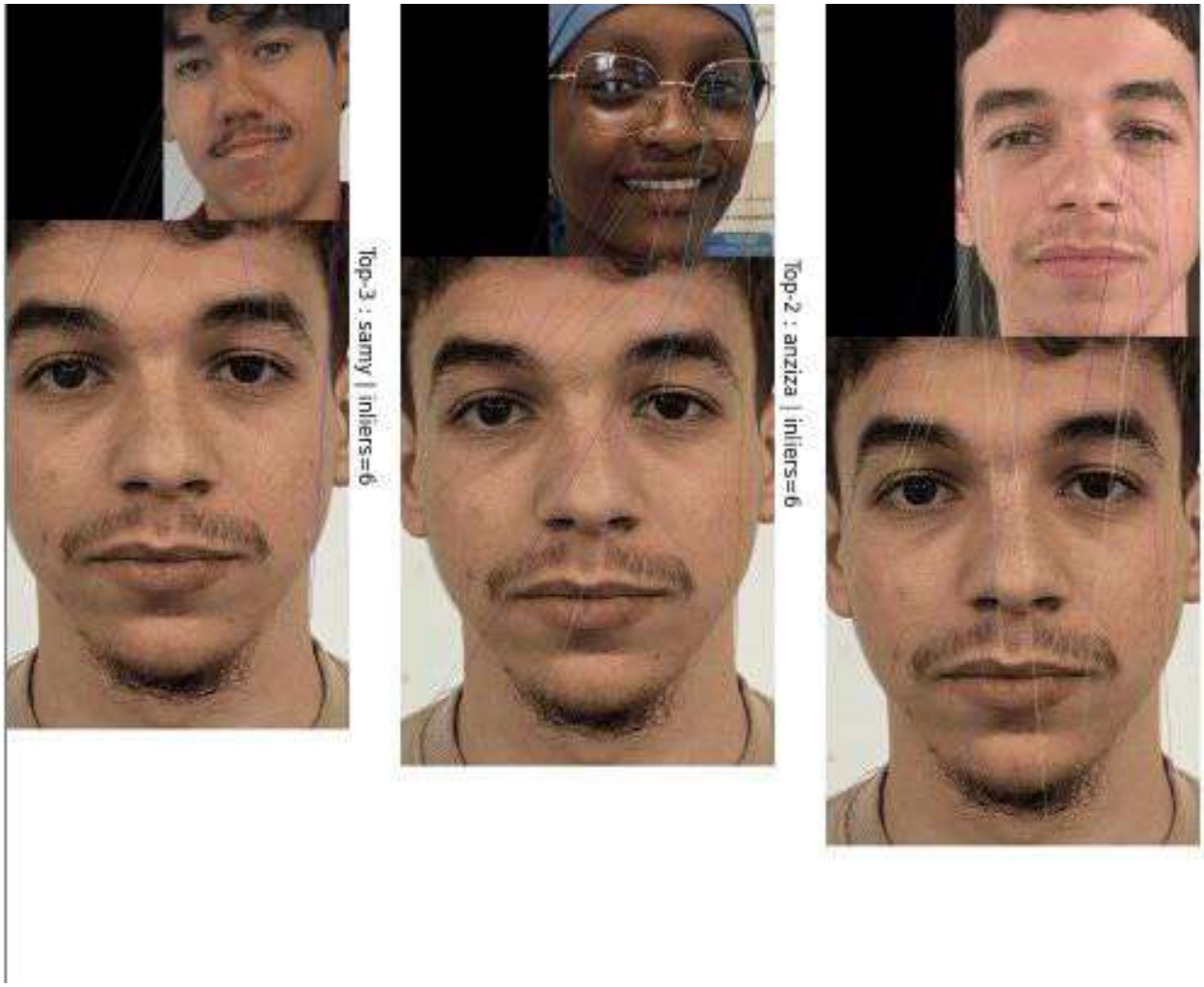


Figure 7: Correspondances obtenues entre une image de test et une image de référence à l'aide du matching SIFT.

V.10 10. Décision connu / inconnu et évaluation

La décision finale repose sur le meilleur score obtenu après le matching et le filtrage RANSAC.
Un seuil de **8 correspondances** est fixé.

- Si ce seuil est atteint, le visage est reconnu

- Sinon, il est considéré comme **inconnu**

Les performances du système sont évaluées à l'aide de l'**accuracy**, définie comme le rapport entre le nombre de prédictions correctes et le nombre total de tests.

Sur les données testées, l'accuracy obtenue est de **0,33**.
Ce résultat montre que le système peut reconnaître certains visages, mais que ses performances restent limitées en raison du faible nombre d'images, de leur qualité et de la difficulté à bien différencier les visages.

V.11 11. Conclusion

La méthodologie présentée repose uniquement sur des méthodes classiques de vision par ordinateur et met en place un système complet de reconnaissance faciale.

Les différentes étapes du traitement permettent une compréhension claire du fonctionnement du système.
Les résultats montrent toutefois que l'utilisation de **SIFT** reste limitée pour la reconnaissance faciale lorsque le nombre d'images est réduit.

VI Approche moderne : ProtoNet

VI.1 Principe du Méta-Apprentissage et de la Métrique

L'approche des *Prototypical Networks* s'inscrit dans le cadre du **méta-apprentissage** (*meta-learning*).
L'objectif n'est pas de classer directement les données, mais d'apprendre une **fonction d'encodage** capable de projeter les images dans un espace latent où la classification se fait par proximité métrique.

VI.2 Architecture de l'Encodeur

L'encodeur f_θ est chargé de produire des vecteurs de caractéristiques discriminants.
Deux architectures ont été explorées.

a) CNN

Le réseau de neurones convolutif (CNN) utilisé comme encodeur est composé de **quatre blocs convolutifs**.
Chaque bloc regroupe plusieurs couches (*layers*) jouant un rôle spécifique dans le traitement de l'image, permettant d'extraire progressivement des représentations de plus en plus abstraites.

Les trois premiers blocs convolutifs sont structurés comme suit :

- une **convolution 2D** avec des noyaux de taille 3×3, permettant l'extraction de caractéristiques locales (contours, textures) ;
- une **normalisation par groupes (Group Normalization)** afin de stabiliser l'entraînement et de réduire la dépendance à la taille des mini-batchs, ce qui est particulièrement adapté au *few-shot learning* ;
- une **fonction d'activation ReLU** introduisant de la non-linéarité ;
- une opération de **sous-échantillonnage MaxPool2d** avec un facteur 2, réduisant la dimension spatiale tout en conservant les activations les plus discriminantes.

Le quatrième bloc convolutif diffère légèrement :

- une convolution 3×3 suivie d'une normalisation par groupes ;
- une couche **Adaptive Average Pooling (AdaptiveAvgPool2d)** produisant une carte de caractéristiques de taille 1×1, indépendamment de la taille de l'image d'entrée.

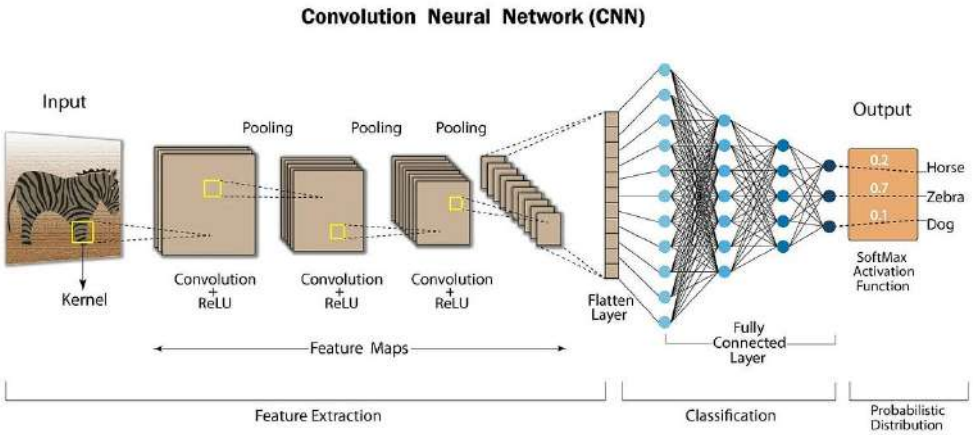


Figure 8: Structure générale d'un bloc convolutif. Chaque bloc est composé d'une convolution, d'une normalisation, d'une activation et d'une opération de pooling.

b) ResNet-18 pré-entraîné

Le réseau **ResNet-18** utilisé comme encodeur repose sur une architecture pré-entraînée sur ImageNet.
Il est constitué de plusieurs **blocs résiduels** successifs permettant l'extraction de représentations de plus en plus abstraites.

Chaque bloc résiduel comprend typiquement :

- une ou plusieurs **convolutions 2D** avec des noyaux 3×3 ;
- une **Batch Normalization** stabilisant l'entraînement et accélérant la convergence ;
- une **fonction d'activation ReLU** ;
- une **connexion résiduelle (skip connection)** facilitant la propagation du gradient dans les réseaux profonds.

Après les blocs résiduels, un **Global Average Pooling** transforme les cartes de caractéristiques en un vecteur de dimension 512.
Ce vecteur est ensuite projeté dans un espace d'embedding de dimension fixe `feature_dim` à l'aide d'une couche linéaire suivie d'une **BatchNorm1d**.
Les embeddings sont finalement **normalisés sur la sphère unité**, ce qui facilite le calcul des distances dans le cadre de ProtoNet.

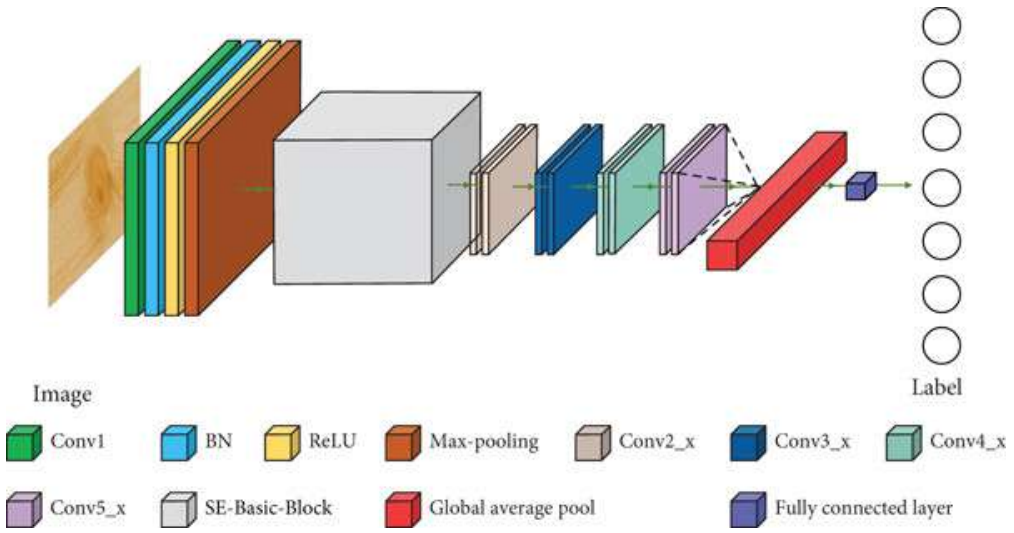


Figure 9: Architecture générale de ResNet-18 utilisée comme encodeur.

VI.3 Fonction de perte, Optimiseur et Scheduler

L'entraînement de ProtoNet dépend à la fois de la fonction de perte, de l'optimiseur et de la stratégie d'ajustement du taux d'apprentissage.

Fonction de perte

La fonction de perte utilisée est la **cross-entropy**, appliquée aux distances entre les embeddings des images du *query set* et les prototypes calculés à partir du *support set*.
Cette formulation encourage le rapprochement des embeddings appartenant à une même classe et l'éloignement de ceux issus de classes différentes, ce qui est particulièrement adapté au cadre *one-shot*.

Optimiseur

L'optimisation des poids est réalisée à l'aide de **AdamW**, qui combine l'algorithme Adam avec une régularisation par décroissance des poids (*weight decay*) afin d'améliorer la généralisation.

Learning rate et scheduler

Le taux d'apprentissage initial est fixé à `lr = 0.001` et est ajusté via un **CosineAnnealingLR**, permettant une diminution progressive du learning rate suivant une courbe cosinus et favorisant une convergence plus stable en fin d'entraînement.

VI.4 Stratégie d'Entraînement

L'entraînement du ProtoNet est effectué sur le dataset **CelebA**, scindé en ensembles *train* et *test* selon un ratio 80/20, garantissant que les identités du test set ne sont pas vues durant l'entraînement.

Chaque image est :

- redimensionnée à 128×128 ;
- convertie en tenseur ;

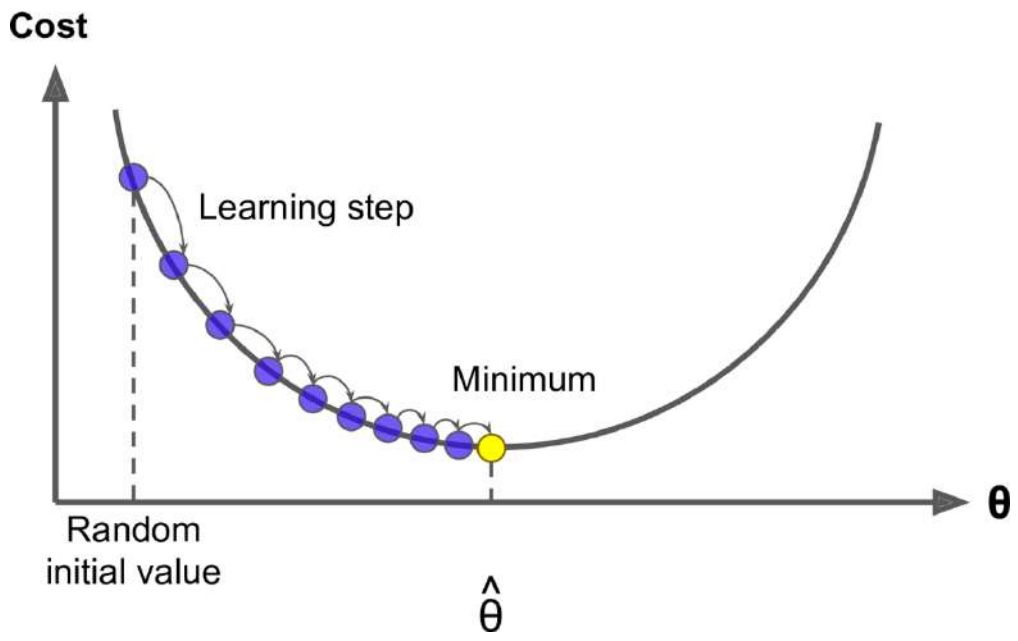


Figure 10: Illustration du fonctionnement d'un Prototypical Network.

- normalisée avec
  `mean = [0.485, 0.456, 0.406]`,
  `std = [0.229, 0.224, 0.225]`.

Une détection faciale automatique via **YOLOv5m-face** est appliquée afin de recadrer chaque visage avant normalisation.

L'entraînement suit une stratégie **one-shot** avec les paramètres suivants :

- $n_{\text{way}} = 20$
- $k_{\text{shot}} = 1$
- $q_{\text{query}} = 10$
- `tasks_per_epoch = 50`
- `num_epochs = 50`

Pour chaque mini-tâche :

1. n_{way} classes sont sélectionnées aléatoirement.
2. Pour chaque classe, k_{shot} images forment le support set et q_{query} le query set.
3. Les embeddings du support set servent au calcul des **prototypes**.
4. Les embeddings du query set sont utilisés pour calculer la **cross-entropy** sur les distances aux prototypes et mettre à jour les poids du modèle.

L'évaluation finale est réalisée sur les scénarios suivants :

- 5-way 1-shot
- 5-way 5-shot
- 10-way 1-shot

- 10-way 5-shot

VI.5 Pipeline d'Application et Évaluation

Cette section décrit le flux opérationnel complet du système, depuis la détection des visages jusqu'à la décision finale.

a) Pré-traitement et Détection Faciale

- **Outil de détection** : modèle YOLO fine-tuné, robuste aux occlusions et aux visages de petite taille.
- **Cropping et normalisation** : chaque visage détecté est découpé puis redimensionné à 128×128 avant traitement par l'encodeur.

b) Phase de Référence One-Shot (Enrollment)

- **Génération des prototypes** : chaque photo individuelle est encodée via f_θ pour produire un embedding de 128 dimensions.
- **Stockage de la galerie** : un seul vecteur est conservé par étudiant comme prototype de référence.

c) Phase de Reconnaissance et Décision

- **Embedding de requête** : chaque visage extrait d'une photo de groupe est encodé par f_θ.
- **Calcul des distances** : distances euclidiennes entre l'embedding de requête et les prototypes.
- **Décision** : l'identité associée à la distance minimale est sélectionnée.

VII Résultats et Visualition de Données

Cette section présente et analyse les performances obtenues par l'approche basée sur les *Prototypical Networks (ProtoNet).*
Les résultats sont étudiés à travers trois phases d'évaluation complémentaires, permettant de valider à la fois :

- les **capacités de généralisation** du modèle,
- la **qualité de l'espace d'embedding** appris,
- l'**efficacité du système** dans un scénario d'utilisation réel.

VII.1 Configuration du méta-entraînement

Le méta-entraînement est réalisé sur le dataset **CelebA** selon une stratégie épisodique, conformément au cadre des réseaux prototypiques.

Chaque épisode correspond à une tâche **20-way 1-shot**, avec **10 images de requête par classe (10-query)**.
L'entraînement est effectué sur **50 epochs**, chacune comprenant **50 tâches** échantillonnées aléatoirement à partir du jeu d'entraînement.

VII.2 Performance du Méta-train et du Méta-test sur CelebA

Les résultats présentés dans cette section visent à évaluer la capacité du modèle ProtoNet à apprendre une métrique discriminante lors du *méta-train*, puis à généraliser à des identités jamais vues lors du *méta-test*.

a) Méta-train

a).1 Analyse des performances en méta-train

La figure ci-dessus illustre l'évolution des principales métriques suivies durant la phase de méta-train.
On observe une diminution progressive de la fonction de perte, indiquant une convergence stable du modèle.

Parallèlement, la précision d'entraînement augmente de manière régulière pour atteindre environ **60 %**, bien au-dessus du niveau de performance aléatoire (**5 %** pour une tâche 20-way).

La courbe du taux d'apprentissage montre l'effet du scheduler *CosineAnnealingLR*, avec une décroissance progressive du learning rate au fil des epochs.
Cette stratégie contribue à stabiliser l'apprentissage et à éviter des oscillations importantes de la fonction de perte en fin d'entraînement.

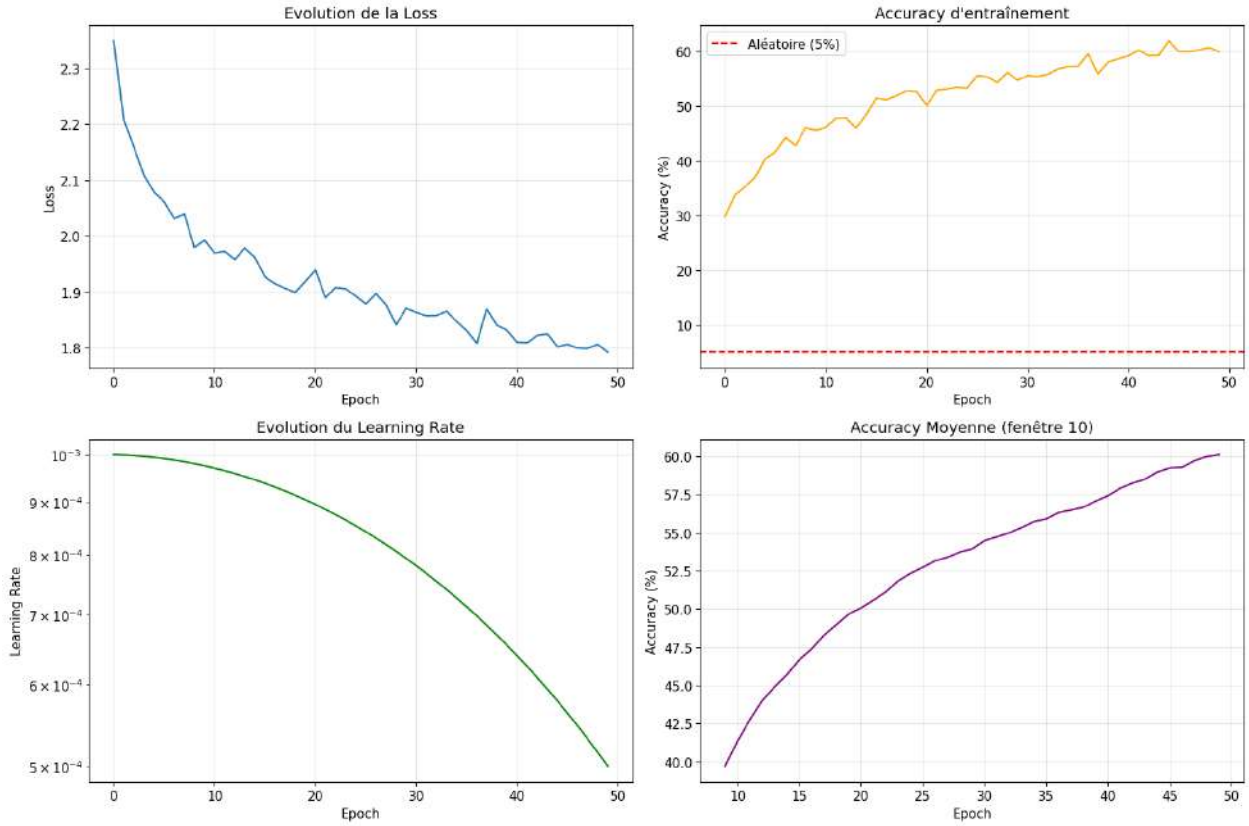


Figure 11: Méta-train obtenu sur 20 classes (N-way) et 1-Shot.

b) Évaluation en méta-test

L'évaluation en *méta-test* est réalisée à l'aide de la fonction `evaluate`, qui mesure la capacité du modèle à reconnaître des identités jamais vues durant l'entraînement (split test).

Pour chaque épisode : - une tâche *few-shot* est échantillonnée à partir du jeu de test, - les prototypes sont calculés à partir du support set, - les images du query set sont classées selon leur distance aux prototypes.

	N-WAY	K-SHOT	Q-QUERY	ACCURACY (%)
5-WAY 1-SHOT	5	1	10	90 %
5-WAY 5-SHOT	5	5	10	86 %
10-WAY 1-SHOT	10	1	10	63 %
10-WAY 5-SHOT	10	5	10	87 %

Figure 12: Méta-test sur 4 scénarios différents.

Les résultats montrent que la performance dépend fortement de la configuration *few-shot* :

- L'augmentation du nombre d'exemples par classe (*k-shot*) améliore significativement la précision.
  - En **10-way**, la précision passe de **63 % (1-shot)** à **87 % (5-shot)**.
- L'augmentation du nombre de classes (*n-way*) rend la tâche plus complexe, entraînant une baisse de performance en 1-shot.

Ces observations sont cohérentes avec les principes fondamentaux du *few-shot learning*.

VII.3 Analyse de l'Espace d'Embedding (ACP et t-SNE)

Au-delà des performances quantitatives, l'analyse qualitative de l'espace d'embedding est essentielle. Le principe des ProtoNets repose sur l'apprentissage d'une représentation où :

- les échantillons d'une même classe sont regroupés,
- les classes différentes sont bien séparées.

Deux méthodes de réduction de dimension sont utilisées : **ACP** et **t-SNE**.

a) Visualisation par ACP

L'Analyse en Composantes Principales (ACP) permet de projeter les embeddings 128D dans un espace bidimensionnel tout en conservant un maximum de variance.

Les deux premières composantes expliquent **59.2 %** de la variance totale.

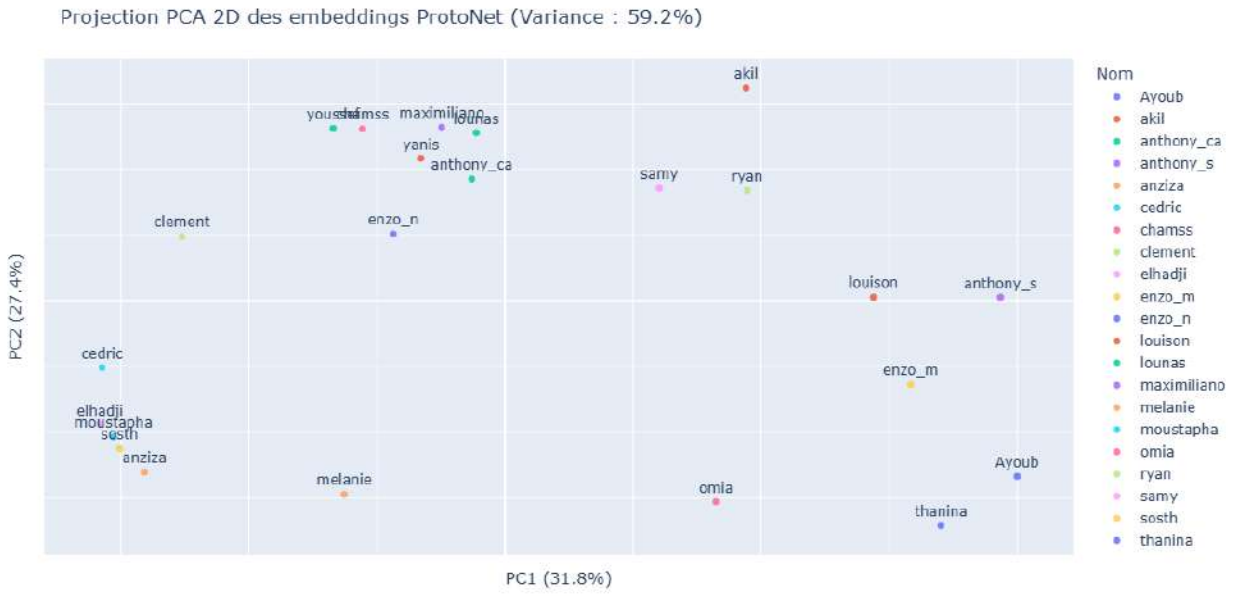


Figure 13: Projection 2D des embeddings (128D) via ACP.

Observations principales :

- **Variance expliquée :** PC1 et PC2 capturent 59.2 % de l'information.
- **Outliers :**
  - *Akil* est fortement isolé (PC2),
  - *Thanina* et *Ayoub* apparaissent également éloignés.
- **Sous-groupes :**
  - Un cluster distinct regroupe *Elhadji*, *Moustapha*, *Sosthene* et *Anziza*.
- **Limitation :**
  - L'ACP est linéaire et ne capture pas les structures non linéaires restantes (~40 % de variance).

b) Visualisation par t-SNE

La méthode **t-SNE**, non linéaire, est utilisée pour préserver la structure locale des données.

Analyse :

- **Clusters compacts et bien séparés :**
  Chaque identité forme un cluster distinct, confirmant l'efficacité de la fonction de perte prototypique.
- **Faible chevauchement :**
  Les embeddings sont globalement bien séparés, indiquant une forte capacité discriminante du modèle.

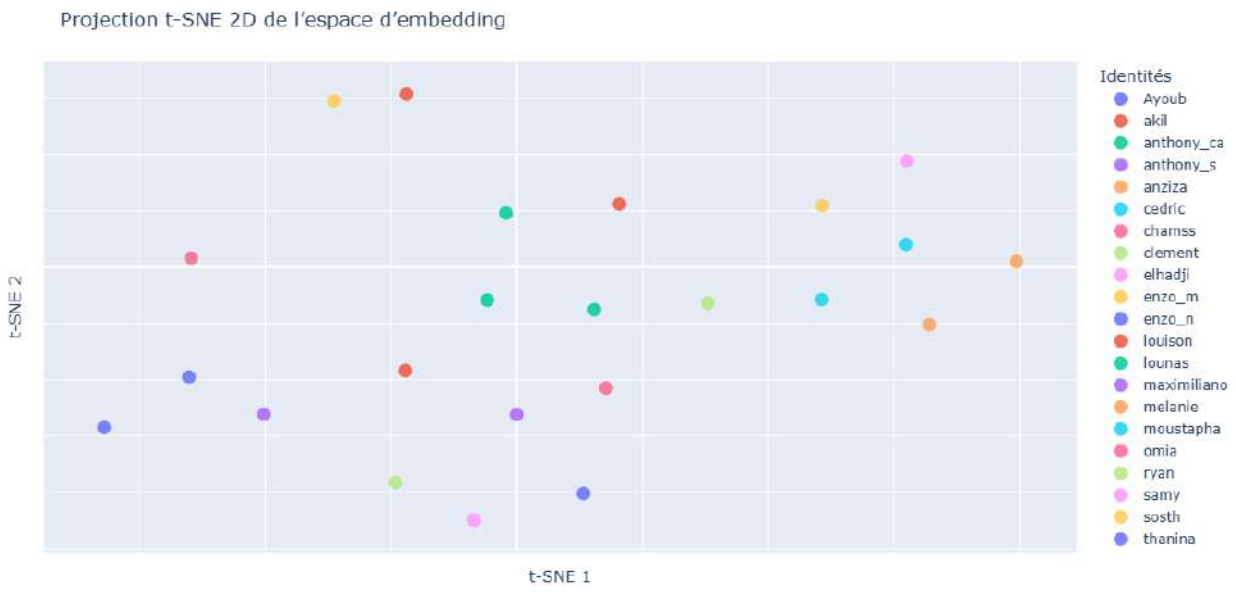


Figure 14: Projection 2D des embeddings (128D) via t-SNE.

VII.4 Évaluation Finale sur le Cas d'Usage Réel

Le modèle ProtoNet est évalué sur **5 photos de groupe** dont une si dessous.

a) Top-K Accuracy

Initialement, **23 visages** sont détectés par YOLO.
Un filtrage est appliqué :

1. Suppression des individus inconnus (hors base).
2. Suppression des visages connus avec une distance minimale supérieure au seuil $\tau = 1.0$.

Après filtrage, **18 visages connus** sont retenus.

Table 3: ProtoNet Top-K accuracy sur photo de la promotion M1 MIASHS

Métrique	Calcul	Résultat
Top-1 Accuracy	3 / 18	**16.67 %**
Top-5 Accuracy	14 / 18	**77.78 %**

Interprétation :

- Le taux **Top-5 = 77.78 %** valide la qualité du clustering.
- Dans la majorité des cas, la bonne identité figure parmi les prototypes les plus proches.

b) Résultats Biométriques et Seuil Optimal

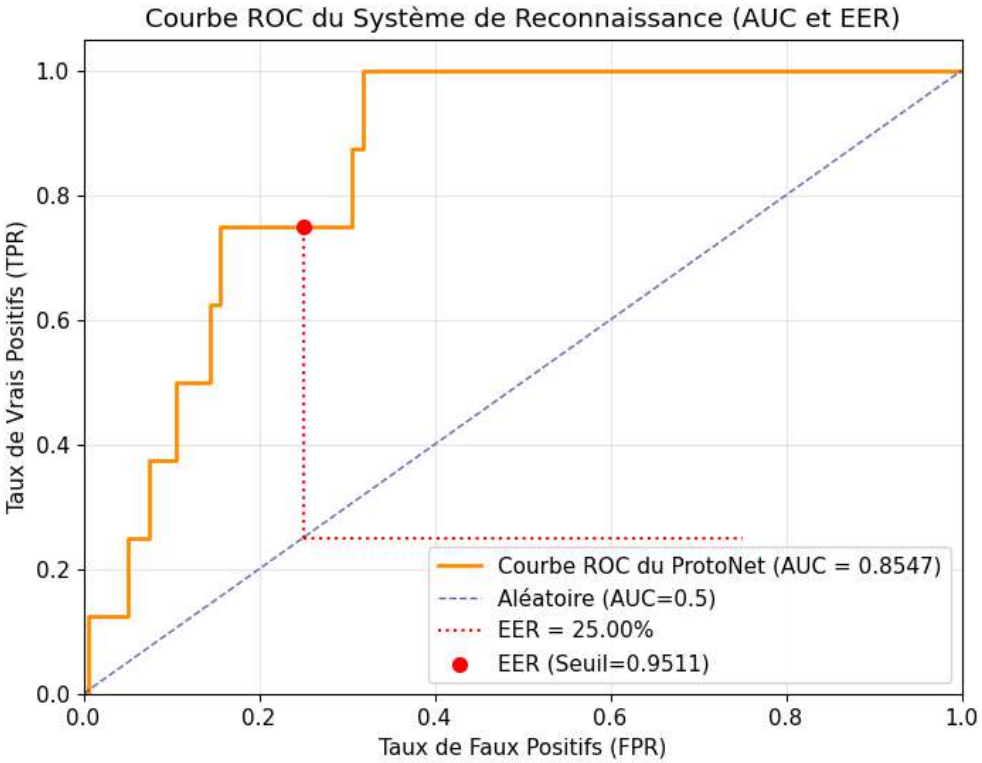


Figure 15: Courbe ROC du système ProtoNet.

Une évaluation biométrique est menée en comparant **9 images tests** à **21 prototypes**, générant **168 paires de distances** pour tracer la courbe ROC.

Table 4: Mesure - Courbe ROC

Métrique	Valeur
AUC	**0.8547**
EER	**25.00 %**
Seuil optimal (τ)	**0.9511**

Analyse :

- **AUC = 0.8547**
  → Bonne séparation globale des classes.
- **EER = 25 %**
  → Performance acceptable pour un usage non critique, mais révélant les limites du *few-shot learning* en conditions réelles (variations de pose, éclairage, peu de données).

VIII Difficultés rencontrées et perspectives

VIII.1 Résultats obtenus avec YOLO, ProtoNet et SIFT

Les expérimentations menées dans ce travail ont permis d'évaluer plusieurs approches pour la détection et la reconnaissance faciale, notamment l'utilisation du modèle **YOLO** pour la détection des visages, l'approche **ProtoNet** pour la classification, ainsi que l'algorithme **SIFT** pour l'extraction et la comparaison de caractéristiques locales.

Les résultats obtenus avec **YOLO** montrent de bonnes performances en détection de visages, avec une précision et un rappel élevés, même dans des scènes complexes comportant plusieurs personnes, des variations d'éclairage et des poses différentes. Le modèle s'avère robuste pour la localisation des visages et constitue une base fiable pour les étapes suivantes du pipeline.

L'approche **ProtoNet**, basée sur l'apprentissage par prototypes, permet une reconnaissance efficace avec un nombre réduit d'exemples par classe. Elle est particulièrement adaptée aux scénarios de *few-shot learning*, mais reste sensible à la qualité des représentations extraites et à la diversité des données d'entraînement.

En revanche, les résultats obtenus avec **SIFT** montrent certaines limites pour la reconnaissance faciale. Bien que cet algorithme soit robuste aux changements d'échelle et de rotation, il reste peu discriminant pour différencier des visages similaires, surtout lorsque le nombre d'images disponibles par identité est faible. Cela se traduit par une précision globale limitée dans les expériences réalisées.

VIII.2 Difficultés techniques rencontrées

Plusieurs difficultés ont été rencontrées au cours de la réalisation de ce projet.

Tout d'abord, l'utilisation de **Google Colab** a présenté certaines contraintes, notamment : - des limitations sur la durée des sessions, - des interruptions fréquentes, - des restrictions liées à l'accès aux ressources GPU.

Ces contraintes ont rendu difficile la continuité des entraînements et des expérimentations, en particulier pour les phases de *fine-tuning*.

Par ailleurs, le **fine-tuning du modèle sur un MacBook** a constitué un défi majeur. L'absence de GPU NVIDIA compatible CUDA a considérablement ralenti les temps de calcul. Cette phase de *fine-tuning* a ainsi duré environ **deux semaines**, ce qui a limité le nombre d'expérimentations possibles et l'optimisation approfondie des hyperparamètres.

VIII.3 Perspectives et travaux futurs

Plusieurs perspectives peuvent être envisagées afin d'améliorer les performances du système proposé.

Tout d'abord, l'utilisation de modèles de reconnaissance faciale plus récents et spécialisés pourrait permettre d'obtenir de meilleurs résultats que les méthodes classiques telles que **SIFT**.

Une piste particulièrement prometteuse consiste à explorer **RetinaFace**, notamment les implémentations optimisées disponibles dans l'écosystème **Apple**. RetinaFace est un détecteur de visages performant, capable de gérer efficacement les visages de petite taille, les poses extrêmes et les occultations partielles. Son intégration pourrait améliorer significativement la qualité de la détection et, par conséquent, les performances globales du système.

D'autres axes d'amélioration incluent :
- l'augmentation de la taille et de la diversité du jeu de données,
- l'utilisation de descripteurs profonds pour la reconnaissance faciale,
- l'entraînement et l'évaluation sur des infrastructures disposant de ressources GPU plus performantes.

IX Conclusion

Ce projet nous a permis de mettre en place une chaîne complète de reconnaissance faciale en partant de zéro, sous la contrainte majeure du **One-Shot Learning**. L'objectif était d'évaluer si un modèle tel que **ProtoNet**, associé à **YOLO**, pouvait être réellement exploitable pour identifier un groupe d'étudiants dans des conditions réelles.

Les tests menés montrent que la solution proposée est prometteuse mais perfectible. L'analyse de la **Top-5 Accuracy** met en évidence que le modèle parvient à bien représenter les visages, puisque l'étudiant recherché apparaît presque systématiquement parmi les premières propositions.

De plus, l'analyse biométrique à l'aide de la **courbe ROC** et de l'**AUC** confirme que l'espace de calcul des distances est cohérent et que les embeddings générés sont discriminants.

Cependant, le passage à une identification stricte en **Top-1** demeure plus difficile. Cette limitation s'explique principalement par la proximité visuelle entre certains étudiants ainsi que par les variations d'éclairage dans les photos de groupe, qui peuvent déplacer les embeddings loin de leurs prototypes respectifs.

X Bibliographie

Keita, Zoumana. 2024. *YOLO*. dataCAmp. https://www.datacamp.com/fr/blog/yolo-object-detection-explained.

OpenAI. 2023. « ChatGPT: Large Language Model ».

Redmon, Joseph, Santosh Divvala, Ross Girshick, et Ali Farhadi. 2016. « You Only Look Once: Unified, Real-Time Object Detection ». *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-88.