Face Identification With Metric Learning

Nguyen Minh Cuong[1, *], Doan Ngoc Cuong[1, *], Bui Khanh Linh[1, *], Nguyen Minh Tuan[1,*], Tran Le My Linh[1, *], Nguyen Thanh Long[1, *]

Abstract

Face Recognition, a pivotal technology at the confluence of computer vision and artificial intelligence, has revolutionized security, identity verification, and human-computer interaction. Offering seamless authentication, it identifies and verifies individuals based on facial features, involving tasks from detection and alignment to feature extraction and classification. This paper addresses the core challenges in Face Recognition, focusing on pivotal tasks: face detection and verification. We scrutinize models—YOLOv5s, ADNet, MTCNN, SSD, VGG16, InceptionV2, and MobileNetV3—each chosen strategically for specific challenges. Our goal is to pinpoint the most effective model or combination for each task, contributing valuable insights to the dynamic realm of Face Recognition.

Keywords

face recognition; facial features; fetection alignment; feature extraction

[1] *The School of Information and Communication Technology - Hanoi University of Science and Technology*

*__Corresponding author__:
cuong.nm210140@sis.hust.edu.vn,
cuong.dn210141@sis.hust.edu.vn,
linh.bk214910@sis.hust.edu.vn,
tuan.nm214940@sis.hust.edu.vn
linh.tlm210535@sis.hust.edu.vn,
long.nt214912@sis.hust.edu.vn

Contents

1. Introduction

In the era of technological advancements, Face Recognition has emerged as a transformative force, reshaping the landscape of security, identity verification, and human-computer interaction. This computational process, rooted in computer vision and artificial intelligence, seeks to identify and verify individuals based on their facial features, providing a seamless and non-intrusive means of authentication. The Face Recognition Problem encompasses a complex journey, requiring intricate steps to unravel the nuances inherent in human faces.

Recent years have witnessed remarkable strides in computer vision, machine learning, and deep neural networks, catapulting Face Recognition to the forefront of biometric technologies. Beyond its promise in enhancing security protocols, this technology extends its applications to diverse fields,

including access control, surveillance, and personalized user experiences.

The fundamental challenge in Face Recognition lies in orchestrating a series of intricate tasks. From the initial detection of facial landmarks and alignment to the extraction of discriminative features and subsequent classification, each step plays a crucial role in achieving accurate and reliable results. The fusion of computer vision algorithms and artificial intelligence methodologies has paved the way for robust Face Recognition systems capable of handling diverse scenarios, facial expressions, and environmental variations.

In the pursuit of understanding and advancing Face Recognition technology, our focus turns to the critical tasks of face detection and verification. This endeavor involves the exploration and evaluation of state-of-the-art models, such as YOLOv5s, ADNet, MTCNN, SSD300_VGG16, VGG16, InceptionV2, and MobileNetV3. Each model is strategically selected to address specific challenges in face detection and verification, with the ultimate goal of identifying the most effective solution or combination of models for each task. Through this comprehensive exploration, we aim to contribute valuable insights to the evolving field of Face Recognition, fostering advancements in accuracy, efficiency, and applicability across a spectrum of real-world scenarios.

1.1 Organization

The subsequent sections of this paper will discuss the existing methods and models for predicting cryptocurrency prices, describe the parameter configuration and error setup, address the feature selection and preprocessing of explanatory variables, evaluate the performance of the proposed model, highlight the limitations of this research, and conclude with future directions for further studies.

2. Data processing

2.1 Face Detection

When training face detection models, we used WilderFace dataset which contains 32,203 images and label 393,703 faces with a high degree of variability in scale, pose and occlusion as depicted in the sample image. WIDER FACE dataset is organized based on 61 event classes. For face alignment task,we trained model with LFW dataset. It contains 5,590 LFW images and 7,876 other images downloaded from the web. The training set and validation set are defined in trainImageList.txt and testImageList.txt, respectively. Each line of these text files starts with the image name, followed by the boundary positions of the face bounding box retured by our face detector, then followed by the positions of the five facial points.

2.2 Face Verification

For the training procedure, we use a subset of MS-Celeb-1M (MS1M) dataset. This data set contains about 10,000,000 images of roughly 100,000 people. Because of limited hardware, we only use a subset of randomly picked 10,000 labels (appox. 100,000 images) for our training. The variation used on this part is cropped into the face and aligned using RetinaFace. Each image has the shape of (3, 112, 112), that is it is a 3 channels RGB image with 112 pixels both horizontally and vertically.

3. Preliminaries

3.1 Face Detection

3.1.1 Neural Network Architectures

Face Detection, as the initial phase in Face Recognition, demands the use of sophisticated neural network models, with Convolutional Neural Networks (CNNs) being the most prominent. These networks are designed to effectively recognize and localize human faces within digital images, even under varying environmental and conditional challenges. The architecture of CNNs, particularly their convolutional layers, is adept at extracting and learning spatial hierarchies of features from images, making them ideal for detecting facial features in diverse settings.

3.1.2 Image Processing and Bounding Box Prediction

The face detection process begins by segmenting the input image into multiple smaller grids. Each grid cell independently predicts the likelihood of a face's presence by generating potential bounding boxes. These bounding boxes, typically rectangular, are defined by coordinates that outline the estimated location and size of faces in the image. Alongside these coordinates, each cell computes an objectness score, assessing the probability that the bounding box indeed contains a face.

3.1.3 Augmentation

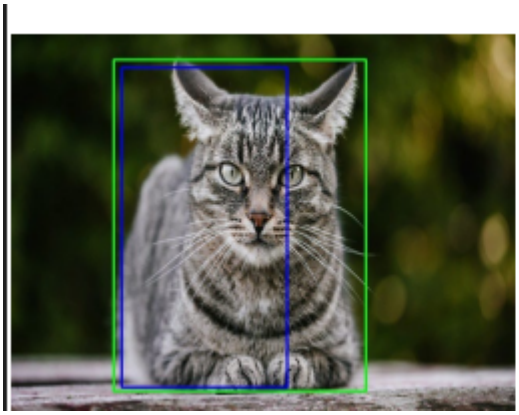
The key factor in achieving effective face recognition is training the model on rich and diverse datasets, including many facial images, varying in size, pose, expression sensitivity and lighting conditions. Such training ensures the model can generalize well and detect faces accurately in the diversity of real-life situations. Data augmentation techniques, such as cropping, rotating, scaling, adjusting lighting, blurring, color contrast, adding noise, etc. can be used to perform this expansion of diversity.

3.1.4 Performance Evaluation Metrics

The performance of a face detection model is evaluated using several key metrics: accuracy, precision, recall, and Intersection Over Union (IOU). Accuracy provides an overall measure of the model's correct detections. Precision focuses on the proportion of true positive detections among all positive identifications made by the model, while recall measures how many actual faces are correctly identified. IOU offers a geometric assessment of the overlap between predicted bounding boxes and actual face locations, indicating the model's accuracy in localizing faces.

3.2 Face Verification

In Face Verification, the primary objective is to determine whether two images depict the same person. While a sim-



$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$

Figure 1. IOU Metric - Face Detection

plistic approach involves pixel-wise image comparison, this method proves inefficient due to variations in lighting, facial orientation, and other factors. To address these challenges, an encoding function denoted as $f(\text{img})$, is introduced. This encoding facilitates element-wise comparisons, resulting in more accurate judgments regarding the similarity of two images.

3.2.1 Face Encoding

To enhance Face Verification, we utilize advanced neural network architectures to process RGB face images, generating 128-dimensional embedding vectors containing distinctive features of those faces.

Calculating the distance between two embedding vectors and applying a threshold makes it possible to ascertain whether two pictures represent the same person. The effectiveness of an encoding is measured based on the following criteria.

- The embeddings of two images of the same person should be notably similar.
- The embeddings of two images of different individuals should exhibit significant dissimilarity.

3.2.2 The Triplet Loss

For an image x, denoted as $f(x)$ where f is the function computed by the neural network, training involves triplets of images (A, P, N):

- A represents the "Anchor" image – a picture of a person.
- P is the "Positive" image – a picture of the same person as the Anchor image.
- N is the "Negative" image – a picture of a different person than the Anchor image.

These triplets are selected from the training dataset and are represented as $(A^{(i)}, P^{(i)}, N^{(i)})$ for the i-th training example. The objective is to ensure that the encoding of an image $A^{(i)}$ of an individual is closer to the Positive image $P^{(i)}$ than to the Negative image $N^{(i)}$ by at least a margin α:

$$\|f(A^{(i)}) - f(P^{(i)})\|_2^2 + \alpha \leq \|f(A^{(i)}) - f(N^{(i)})\|_2^2$$

This constraint guides the training process to embed similar faces closer in the feature space while pushing dissimilar faces farther apart.

3.2.3 Online Triplet Loss

In the original implementation of triplet loss , as known as offline triplet loss, triplets are pre-selected and stored in a dataset before the training starts. Cocretely, we would create batches of these triplets of size B, which means we will have compute $3B$ embeddings (anchor, positive, negative) to get the B triplets, compute the loss of these B triplets and then backpropagate into the network. Within online triplet loss, triplets are selected dynamically during each training iteration. For each training batch, an anchor sample is chosen, and then a positive sample and a negative sample are selected based on the current state of the model. Given a batch of B examples (for instance B images of faces), we compute the B embeddings and we then can find a maximum of B^3 triplets. In fact, many of these triplets are not valid (i.e. we can not ensure have enough positives and negatives).

The use of the sampler method in the DataLoader helps ensure a specific number of triplets in each batch. According to the PyTorch Metric Learning library, the MPerClassSampler method allows the specification of the number of samples per class in a batch. For instance, with 4 samples per class and 16 classes per batch, a total of 64 samples per batch can be achieved. This approach enhances stabilization and maximizes the number of triplets during training.

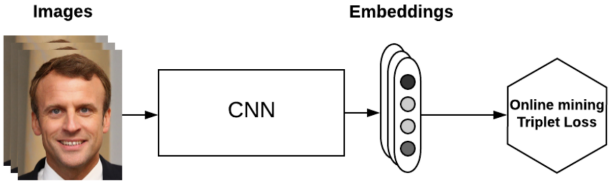


Figure 2. Implementation of online triplet.

Method: For each iteration, we have to computed a batch of B embeddings from a batch of B inputs. Now our mission is to generate triplets from these embeddings. Whenever we have three elements a,p,n which a and p have the same label but are distinct and n has a different label, so a,p,n is a valid trplet. Then we will build the 3D tensor of shape (B,B,B) (B is batch size) where the element at index (i, j, k) represents the loss for triplet(i, j, k), lastly we will set to 0 if the invalid one and take the average over all the valid triplets.

Additionally, to enhance the efficiency of loss function and challenging examples for the network to learn from, we will select a subset triplet among all of them. Of course, we

will not choose random, paper In Defense of the Triplet Loss for Person Re-Identification introduced 2 strong strategies:

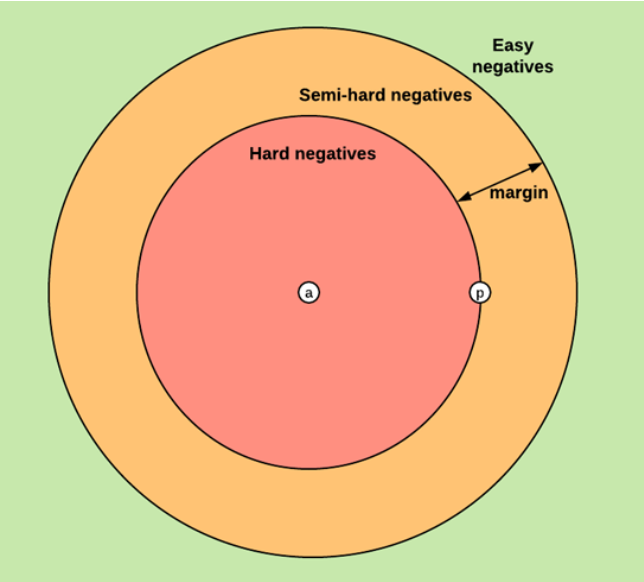


Figure 3. Selection strategies.

- • Hard triplet: for each triplet, negative is closer to anchor than the positve, for example: d(a,p)>d(a,n)
- • Semi-hard triplet: for each triplet, the negative is farther from the anchor than the positive but still provides a challenge for the network. For example: d(a,p) < d(a,n) < d(a,p) + margin

3.2.4 Additive Angular Margin Loss

Additive Angular Margin Loss [1] aims to further improve the discriminative power of the face recognition model and to stabilize the training process by adding a penalty m to the angle between the current feature and the target weight:

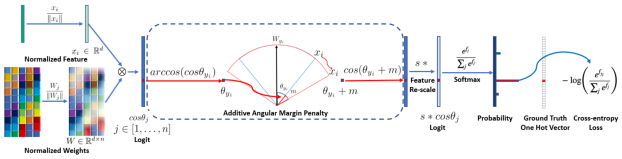
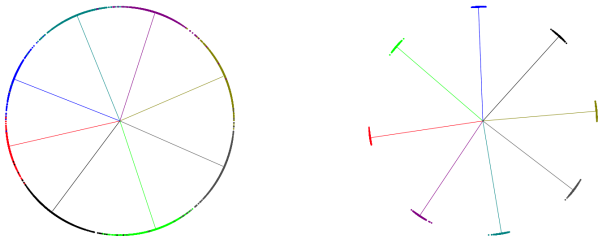


Figure 4. Training a DCNN for face recognition supervised by the ArcFace loss [1]. Based on the feature x_i and weight W normalization, we get the $cos\theta_j$ (logit) for each class as WT j xi. We calculate the $arccos(\theta_{y_i})$ and get the angle between the feature xi and the ground truth weight W_{y_i}. In fact, W_j provides a kind of centre for each class. Then, we add an angular margin penalty m on the target (ground truth) angle θ_{y_i}. After that, we calculate $cos(\theta_{y_i} + m)$ and multiply all logits by the feature scale s. The logits then go through the softmax function and contribute to the cross-entropy loss.

As a result, the full loss function is:

$$L_2 = -\frac{1}{N}\sum_{i=1}^{N}\frac{e^{s(cos(\theta_{y_i}+m))}}{e^{s(cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{N} e^{s\,cos\theta_j}}$$

By doing so, the proposed ArcFace loss enforce more evident gap between nearest classes comparing to the popular softmax loss' roughly separable feature embedding but noticeably ambiguous decision boundaries.



(a) Softmax (b) ArcFace

Figure 5. Toy examples under the softmax and ArcFace loss on 8 identities with 2D features [1]. Dots indicate samples and lines refer to the centre direction of each identity. Based on the feature normalisation, all face features are pushed to the arc space with a fixed radius. The geodesic distance gap between closest classes becomes evident as the additive angular margin penalty is incorporated.

3.3 Model Integration

Face Recognition Problem requires at least the following 3 steps:

Face Detection and Face Alignment This step focuses on determining the position of the face in an image or video frame. A rectangle is created around the facial region for precise localization. Additionally, this process includes aligning the facial angle to ensure accuracy and uniformity in identifying facial features.

Face Extraction (Face Embedding) After determining the face's position, this step concentrates on extracting facial features. These features are represented as a vector in a high-dimensional space, typically a 128-dimensional vector. This numeric representation of the face is used to differentiate between individuals.

Face Classification (Face Verification) The final decision-making step is to classify the face based on the generated feature vector. In this task, the model compares the distance between the feature vector of the new face and those known beforehand in the dataset. A smaller distance indicates similarity between the two faces, allowing the model to determine whether the person in the new image is someone known or unknown from the dataset.

4. Modelling

4.1 Face Detection

4.1.1 YOLOv5 with YOLOv5s

In this section, we focus on deploying the YOLOv5s model, specifically tuned for the task of face detection. YOLO (You

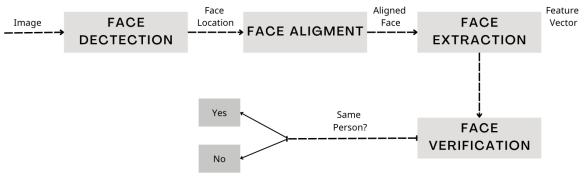


Figure 6. Processing flow of the Face Recognition problem.

Only Look Once) is a state-of-the-art, real-time object detection system, and YOLOv8 is the latest in this series. The 's' in YOLOv5s stands for 'small', indicating that this model is the smallest and fastest variant, making it ideal for applications that require real-time processing with limited computational resources.

Model Architecture The YOLOv5 architectural model is designed for efficiency and speed. It uses the CSPDarknet53 backbone, a modification of the Darknet configuration that was used in previous versions. New configuration of SPPF and CSP-PAN is used for the Neck part. YOLOv5 uses the output of YOLOv3 for the Head section.

1. **Backbone:** YOLOv5 employs a CSP-Darknet53 backbone, which is an enhanced version of the Darknet53 architecture used in YOLOv3. The CSP (Cross Stage Partial Network) strategy is used here to enable the flow of information to deep layers and reduce redundant gradients.

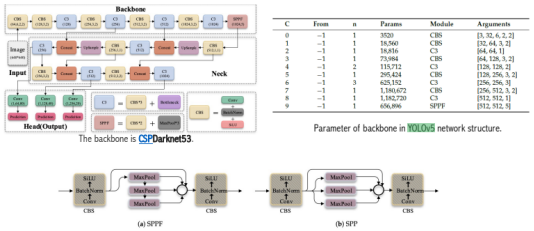


Figure 7. Backbone CSPDarknet53 of Yolov5

2. **Neck:** The neck of the model is designed to construct feature pyramids that aid in detecting objects of various sizes and scales. YOLOv5 incorporates a modified Path Aggregation Network (PANet) with the CSP strategy and a variant of the Spatial Pyramid Pooling (SPP) block called SPPF for faster processing.

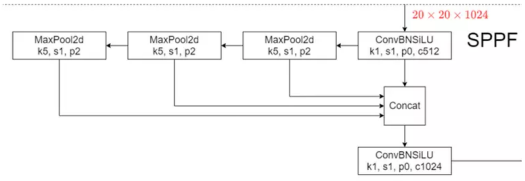


Figure 8. SPPF module architecture

3. **Head:** The head of YOLOv5 is identical to that used in YOLOv3 and YOLOv4. It performs the final prediction, generating bounding box coordinates, objectness scores, and class predictions.

Input Preprocessing For face detection, the input images are first resized to a standard dimension of 640x640 pixels. YOLO labels.txt format (x_center, y_center, width, height), normalized by image dimensions.

Detection Mechanism YOLOv5 divides the input image into a grid, and each grid cell predicts bounding boxes, objectness scores indicating the likelihood of an object's presence, and class probabilities. For face detection, the model is trained to identify and localize faces within these bounding boxes.

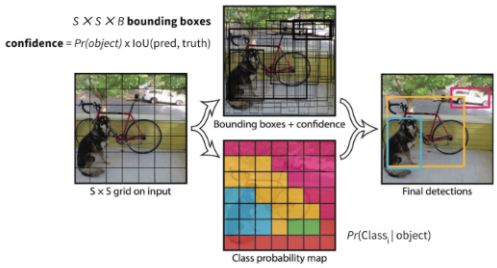


Figure 9. Detection Mechanism of Yolov5

Model Performance YOLOv5s demonstrates high accuracy and speed in face detection. Experimentally, we have only trained Yolov5 on the WIDER set with 12880 training images and 3226 validation images on 1 epoch then the detection ability of the model with actual images has been quite good. After that, we continue to train the model for additional 100 epochs. The Yolov5s model's very small size, only about 14-15MB makes it suitable for deployment in mobile applications and embedded systems.

4.1.2 ADNet with STAR Loss

ADNet: Leveraging Error-Bias Towards Normal Direction in Face Alignment STAR Loss: Reducing Semantic Ambiguity in Facial Landmark Detection

ADNet with ADL and AAM In our research on face alignment and facial feature detection, we research the ADNet network developed by Yangyu Huang, Hao Yang, Chong Li, Jongyoo Kim, and Fangyun Wei from Microsoft Research Asia. This network solves the problem of skew in the error distribution of facial feature points. ADNet uses Anisotropic Direction Loss (ADL) applies strong constraint in the normal direction of the facial boundaries and Anisotropic Attention Module (AAM) focuses on local point and edge regions providing loose constraint in the tangential direction for coordinate and heatmap regression. ADL applies strong constraint in the normal direction of the facial boundaries, while AAM . These two methods support each other in learning facial structure and texture details. ADNet has shown excellent results on datasets such as 300W, WFLW and COFW.

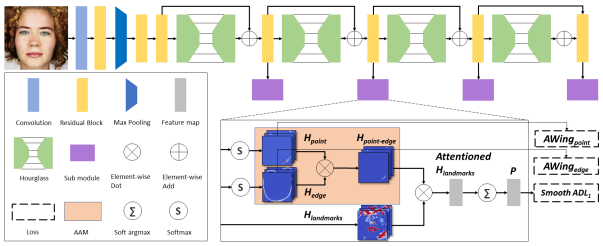


Figure 10. The Framework of ADNet

STAR Loss: Reducing Semantic Ambiguity in Facial Landmark Detection STAR Loss (Self-adaptive Ambiguity Reduction) or self-adaptive anisotropic direction loss, reduces ambiguity in facial point detection by measuring the anisotropy of the predicted distribution, conceptualized by Zhenglin Chu, Huaxia Li, Hong Liu, Nanyang Wang, Gang Yu and Rongrong Ji. This method uses PCA for predictive analysis, determining the direction and magnitude of semantic ambiguity. STAR Loss reduces prediction error towards ambiguity, thereby improving model accuracy and stability, especially effective on the COFW, 300W and WFLW datasets.

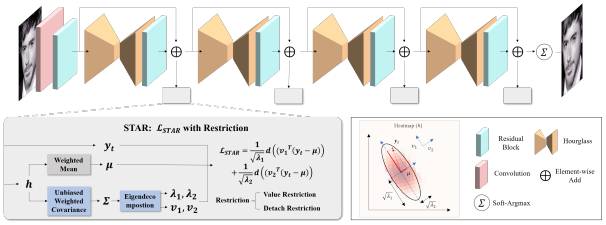


Figure 11. Self-adapTive Ambiguity Reduction (STAR) loss

Implementation Details We retrained ADNet from scratch using the WFLF (Wider Facial Landmarks in the Wild) dataset, featuring 10,000 images (7,500 for training, 2,500 for testing) with 98 landmarks each. Conducted on Google Colab's T4 GPU, each training epoch took around one hour. Remarkably, the average STAR Loss dropped from 3 to below 0.5 in just the initial epoch. After that, we have evaluated the model with a mini set of actual images right after the initial epoch, and the results were very satisfactory with quite standard landmarks. In the following epochs, the average STAR loss continues to decrease. Due to limitations of the T4 GPU colab, we train many times, each time training 3 epochs in 3 hours. NME is the main index we use for the Evaluation process.

4.1.3 MTCNN

A popular deep learning model used for face detection and facial feature alignment in images, MTCNN consists of three stages:

Stage 1 (Proposal Network or P-Net) This stage generates candidate bounding boxes where faces might be located. It uses a fully convolutional network to produce initial bounding box proposals. Non-maximum suppression (NMS) is employed to merge highly overlapped candidates.

Stage 2 (Refinement Network or R-Net) All candidates from the previous stage are fed into another CNN, refining the bounding box proposals generated by the P-Net. This stage rejects false positives and generates more accurate bounding boxes around faces.

Stage 3 (Output Network or O-Net) This stage further refines bounding boxes and performs facial landmark localization (eyes, nose, mouth).

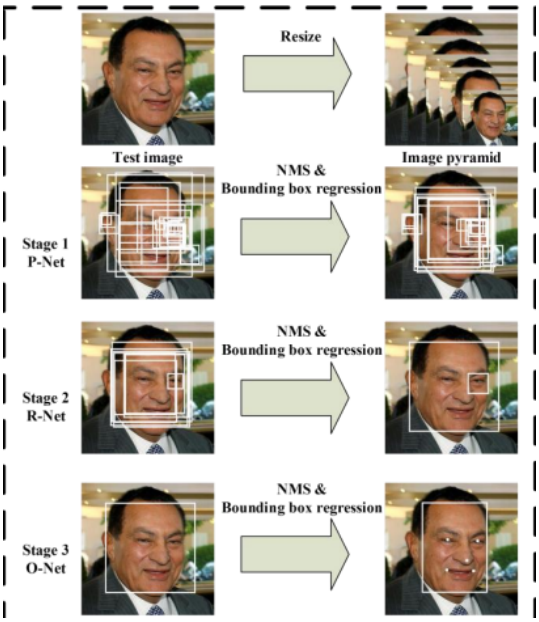


Figure 12. Pipeline of MTCNN model that includes three-stage multi-task deep convolutional networks.

Model Architecture **R-Net:** Similar to P-Net but more complex with additional layers to refine bounding box proposals. **O-Net:** More sophisticated, performing both bounding box regression and facial landmark localization.

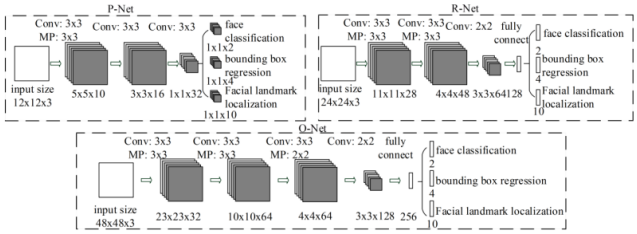


Figure 13. The architectures of P-Net, R-Net, and O-Net.

Training: MTCNN is trained in a cascaded manner, with each stage trained separately but depending on the outputs of the previous stage. Training data typically includes labeled face images with bounding box coordinates and facial landmark annotations. The model is trained using a combination of loss functions, including bounding box regression loss, classification loss, and landmark localization loss.

1) Face Classification: The learning objective is formulated as a two-class classification problem. For each sample x_i, the cross-entropy loss is used:

$$L_i^{\text{det}} = -(y_i^{\text{det}} \log(p_i) + (1 - y_i^{\text{det}})(1 - \log(p_i))) \quad (1)$$

where p_i is the probability produced by the network indicating whether the sample is a face. The notation $y_i^{\text{det}} \in \{0, 1\}$ denotes the ground-truth label.

2) Bounding Box Regression: For each candidate window, the offset between it and the nearest ground truth (bounding box's left top, height, and width) is predicted. The learning objective is formulated as a regression problem using Euclidean loss for each sample x_i:

$$L_i^{\text{box}} = \|\hat{y}_i^{\text{box}} - y_i^{\text{box}}\|^2 \quad (2)$$

where $\hat{y}_i^{\text{box}}$ is the regression target obtained from the network, and $y_i^{\text{box}} \in \mathbb{R}^4$ contains coordinates for left top, height, and width.

3) Facial Landmark Localization: Similar to bounding box regression, facial landmark detection is formulated as a regression problem:

$$L_i^{\text{landmark}} = \|\hat{y}_i^{\text{landmark}} - y_i^{\text{landmark}}\|^2$$

where $\hat{y}_i^{\text{landmark}}$ is the ground-truth landmark coordinate, and $y_i^{\text{plandmark}}$ is the regression target obtained from the network.

Performance: MTCNN is known for its high accuracy in face detection and facial feature alignment tasks. It achieves good performance in detecting faces under various conditions like different poses, scales, and lighting conditions. The model is computationally efficient compared to some other deep learning-based face detection approaches.

4.2 Face Verification

4.2.1 Model with ArcFace Loss

For this loss function, we deploy a model consists of a MobileNetV3 [2] backbone and an embedding head of 128-dimension vector. The MobileNetV3 model that is used is the small, pretrained using ImageNet version from TorchVision library. Its classifier is replaced with the aforementioned embedding head consisting a linear layer following a batch norm and a drop out layer for better convergence. During training process, we also randomly create the classify weight matrix W which is discarded when inferencing.

The input data for this model is first augmented using Random Erasing, Random Crop and Random Horizontal Flip to simulate different real world variation of the same face, then resized to 224 x 224 before feed to the model.

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d, 3x3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	✓	RE	2
$56^2 \times 16$	bneck, 3x3	72	24	-	RE	2
$28^2 \times 24$	bneck, 3x3	88	24	-	RE	1
$28^2 \times 24$	bneck, 5x5	96	40	✓	HS	2
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	120	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	144	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	288	96	✓	HS	2
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	conv2d, 1x1	-	576	✓	HS	1
$7^2 \times 576$	pool, 7x7	-	-	-	-	1
$1^2 \times 576$	conv2d 1x1, NBN	-	1024	-	HS	1
$1^2 \times 1024$	conv2d 1x1, NBN	-	k	-	-	1

Figure 14. MobileNetV3-Small architecture [2].

4.2.2 Model with Triplet Loss

For this loss function, we deploy a model consisting of an InceptionV2 [3] and an embedding head of a 128-dimensional vector. The paper "Rethinking the Inception Architecture for Computer Vision" proposed several upgrades to factorizing convolutions and aggressive dimension reductions inside neural networks, resulting in networks with relatively low computational costs while maintaining high quality [3].

General Design Principles Avoid representational bottlenecks, especially early in the network. Downscale the input image and feature maps gently. The more different filters you have, the more different feature maps you will have, leading to faster learning. Spatial aggregation (dimension reduction) can be done over lower-dimensional embeddings without much or any loss in representational power.

Factorizing Convolutions

- They factorize a 5x5 convolution into two stacked 3x3 convolutions, resulting in a (9+9)/25 computation load reduction with a relative gain of 28% by this factorization [3].
- Factorizing n x n convolutions into a combination of 1 x n and n x 1 convolutions, termed asymmetric convolution, proves to be a cost-effective alternative.

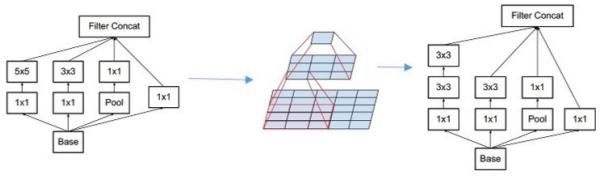


Figure 15. Factorizing a 5x5 convolution into two stacked 3x3 convolutions [3].

Grid Size Reduction Traditionally, convolutional networks use pooling before convolution operations to reduce the grid size of the feature maps, but this can introduce a representational bottleneck. The authors propose increasing the number of filters to remove the bottleneck, achieved by the inception module.

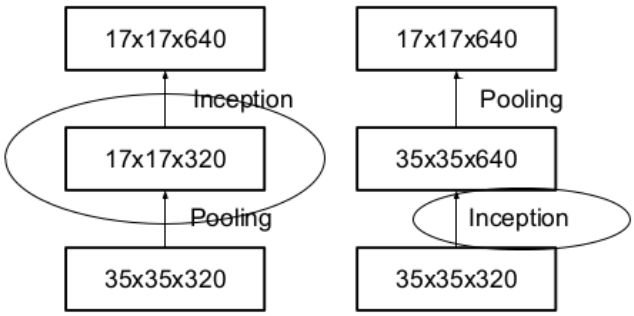


Figure 16. Bottleneck Feature Learning Module.

In the left picture, we are introducing a representational bottleneck by first reducing the grid size and then expanding the filter bank which is the other way around in the right picture.

However, the right side is more expensive so they proposed another solution that reduces the computational cost while eliminating the bottleneck (by using 2 parallel stride 2 pooling/convolution blocks).

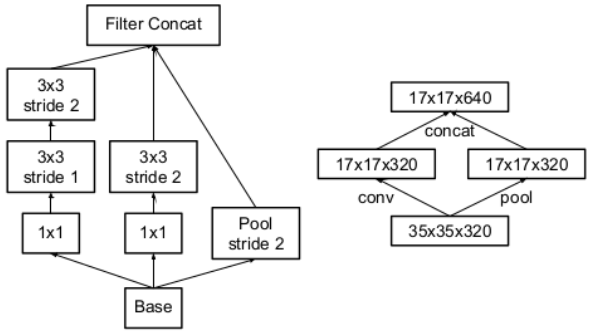


Figure 17. The diagram on the right represents the same solution but from the perspective of grid sizes rather than the operations.

Inception-V2 The model takes an input image of size 112x112 pixels with 3 color channels (RGB). Utilizes the nn.Upsample layer to resize the input image to 112x112 pixels. Uses the Linear layer to predict the output class (classification) with 128 classes. Adds an additional fully connected layer to create a 128-sized embedding from the last layer, which can be used in applications such as unsupervised learning. Implements dropout to reduce overfitting.

The input data for this model is first augmented using Random Rotation and Random Horizontal Flip to simulate different real-world variations of the same face, then resized to 112 x 112 before being fed to the model.

Type	Patch Size/Stride	Input Size
ConvBlock	3x3/2	112x112x3
ConvBlock	3x3/1	56x56x32
MaxPool2d	3x3/2	28x28x64
ConvBlock	3x3/1	28x28x64
ConvBlock	3x3/2	14x14x80
ConvBlock	3x3/1	14x14x192
ConvBlock	3x3/1	14x14x288
InceptionF5	as in figure 5	14x14x288
InceptionF6	as in figure 6	14x14x768
InceptionF7	as in figure 7	7x7x1280
InceptionF7	as in figure 7	7x7x2048
AdaptiveAvgPool2d	1x1	1x1x2048
Linear (Logits)	-	128

Table 1. InceptionV2 Model Architecture

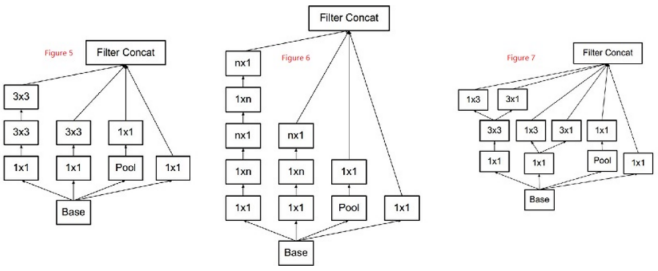


Figure 18. Inception Block in figure 5, 6, 7.

4.2.3 Model with Online Triplet Loss

For the implementation of a model utilizing the Online Triplet Loss function, we deploy the VGG16 architecture. The VGG16 model is renowned for its simplicity and effectiveness in various computer vision tasks. Originally introduced in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition," the VGG16 architecture has proven to be a strong candidate for feature extraction due to its straightforward design principles. We keep the advantage of VGG16 and make some modifications to suit with our problem.

General Design Principles First, model used a tiny 3×3 receptive field with a 1-pixel stride—for comparison, AlexNet used an 11×11 receptive field with a 4-pixel stride. The 3×3 filters combine to provide the function of a larger receptive field. The advantage of employing multiple smaller layers, as opposed to a single large layer, is that enhances the decision functions and allows the network to converge quickly. Second, VGG16 use a small convolutional filter, which decreases the probability of overfitting during training. The choice of a 3×3 filter is considered optimal because a smaller size is unable to capture left-right and up-down information.

VGG16 Architecture The VGG16 architecture is known for its simplicity and uniform structure, consisting of 16 weight layers, including 13 convolutional layers and 3 fully connected

layers.

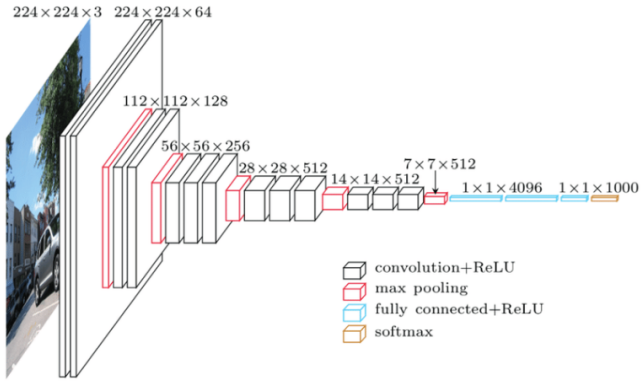


Figure 19. Figure architectural of VGG16

Here is a quick outline of the network architecture:

Table 2. VGG16 Architecture

Layer	Output Shape	Param #
Conv2d-1	256x256x64	1,792
Conv2d-2	256x256x64	36,928
MaxPool2d-3	128x128x64	0
Conv2d-4	128x128x64	73,856
Conv2d-5	128x128x128	147,584
MaxPool2d-6	64x64x128	0
Conv2d-7	64x64x256	295,168
Conv2d-8	64x64x256	590,080
Conv2d-9	64x64x256	590,080
MaxPool2d-10	32x32x256	0
Conv2d-11	32x32x512	1,180,160
Conv2d-12	32x32x512	2,359,808
Conv2d-13	32x32x512	2,359,808
MaxPool2d-14	16x16x512	0
Conv2d-15	16x16x512	2,359,808
Conv2d-16	16x16x512	2,359,808
Conv2d-17	16x16x512	2,359,808
MaxPool2d-18	8x8x512	0
AdaptiveAvgPool2d-19	7x7x512	0
Linear-20	4096	102,764,544
Linear-21	4096	16,781,312
Linear-22	256	1,048,832

- **Input** —The image first is transformed to tensor and resized to (256x256) before fed to the model, this is differed from original input of VGG16 because the original VGG16 was design for ImageNet competition where consisted of images with a resolution of size 224x224 pixels. There are several reasons for larger resize, one is a higher resolution will provide more information to the model to handle these variations effectively. Furthermore, faces in images contain fine details, and using a higher resolution can help capture these details.
- **Convolutional layers** — The convolutional filters of network use the smallest possible receptive field of 3×3. Network also uses a 1×1 convolution filter as the input's linear transformation.
- **ReLu activation** — Rectified Linear Unit (ReLU) activation functions are used after each convolutional layer.ReLU is linear function return matching output for positve input and set to 0 for negative input. VGG has a set convolution stride of 1 pixel to preserve the spatial resolution after convolution.
- **Hidden layers** — Rectified Linear Unit (ReLU) activation functions are used after each convolutional layer. ReLU is linear function return matching output for positve input and set to 0 for negative input. VGG has a set convolution stride of 1 pixel to preserve the spatial resolution after convolution.
- **Pooling layers** – A pooling layer follows several convolutional layers— a pooling layer is used to reduce the dimensionality and the parameter count of the feature maps generated through each convolutional step. Pooling is crucial step, especially considering the substantial increase in the number of filters, progressing from 64 to 128, 256, and ultimately reaching 512 in the final layers.
- **Fully connected layers** — Network includes three fully connected layers. The first two layers each have 4096 channels, and the third layer has 256 channels, which is the dimension of face embedded vector (in original VGG16 network, the output layer has 1000 channels)

5. Result

5.1 Evaluation Methods

5.1.1 Face Detection

To evaluate our model's proficiency in face detection, we utilized a set of 1000 images from the CELEBA dataset, which was not included in our training data to ensure unbiased testing. Our assessment focuses on three primary metrics: precision, recall, and mean Intersection over Union (IOU), with a threshold set at 0.5. These metrics are crucial as precision measures the accuracy of the detected faces against the total detected objects, recall evaluates the model's ability to detect all actual faces, and mean IOU assesses the accuracy of the bounding box placement. Additionally, we use Normalized Mean Error (NME) metrics to evaluate model performance in face alignment, NME focuses on the precision of key facial landmarks against ground truth annotations to ensure accurate and reliable alignment.

5.1.2 Face Verification

We evaluated our model using the standardized Labeled Face in the Wild dataset. Concretely, we follow the dataset's instruction to use a 10-fold cross validation using pre-splitted

set. We also report the model Validation Rate (True Positive Rate, TPR) with given False Acceptance Rate (False Positive Rate, FAR) of [0.1, 0.01, 0.001] respectively. These two metrics specify how often the model is able to verify an user given a threshold satisfying a pre-defined FAR.

Furthermore, we also collected 60 images of 12 different people and report the TPR given the aforementioned FAR.

5.2 Performance

5.2.1 Face Detection

Evaluation on 1000 images of CELEBA Dataset

5.2.2 Face Verification

Evaluation on Labeled Face in the Wild

model	*Embed*	*Accuracy*	*TPR @ FAR=10%*	*TPR @ FAR=1%*	*TP*
InceptionNetV2 + Triplet	128	86.87	81.13	78.12	
VGG16 + Triplet + Mining	256	77.35	60.00	59.03	
MobileNetV3 + ArcFace	128	95.93	95.76	92.86	

As can be seen, all models have exceptional results: All models achieve at least 77.35% accuracy with a minimum of 59.03% true positive rate given a false positive rate of just 0.1%. Though mining helps improve convergence speed, it doesn't give the model a better result overall. This can be the result of a not-fully converged model and the improved InceptionNetV2 backbone compared to the old VGG16 backbone.

Overall, ArcFace Loss with MobileNetV3 gives the best result with 95.93% accuracy and 86.43% true positive rate given a false positive rate of 0.1%. Without going with the smallest model, the result could be slightly better.

6. Conclusion

In conclusion, this project has

7. Future Work

References

[1] J. Deng, J. Guo, J. Yang, N. Xue, I. Kotsia, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition, 2022.

[2] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam. Searching for mobilenetv3, 2019.

[3] Z. Wojna. Rethinking the inception architecture for computer vision.