Big Data Research Project

Using VAE and AE encoder as feature extractor for face verification

Dilbar ISAKOVA
Sihem BOUTEBAL
Melissa MEDJAHED

Github Link

Advisor:	Nacera SEGHOUANI	-	CentraleSupélec	nacera.seghouani@centralesupelec.fr
Advisor:	Alexandre MONTGIEUX	-	CentraleSupélec	alexandre.fournier-montgieux@centralesupelec.fr

Contents

Chapter 1
Introduction

1.1 Introduction

In the dynamic landscape of biometric identification technologies, face verification has emerged as a cornerstone for non-intrusive, user-friendly, and swift authentication solutions. This technology has become indispensable in a multitude of critical applications, such as personal device security, surveillance systems, access control mechanisms, and personalized interactions with smart devices. Its appeal lies in its ability to offer high accuracy without requiring physical contact, making it particularly suitable for scenarios demanding seamless and hygienic interactions [48].

The transformative role of deep learning in face verification cannot be overstated. Advanced techniques have enabled systems to process and interpret complex, high-dimensional data derived from human faces with unprecedented precision. While **Convolutional Neural Networks (CNNs)** have been the predominant approach, alternative methods such as **Autoencoders (AEs) and Variational Autoencoders (VAEs)** have been explored for their ability to learn meaningful latent representations in an unsupervised manner. These models perform dimensionality reduction and feature extraction, making them candidates for face verification tasks [27] [32].

Despite their potential, AEs and VAEs have not gained widespread adoption in face verification, primarily due to theirinferior performance compared to CNN-based approaches. Existing literature does not provide extensive evaluations of their effectiveness across multiple benchmark datasets, making the current body of research on the topic relatively weak. This gap in the literature presents an opportunity to investigate whether AEs and VAEs can be optimized to improve verification accuracy and generalization [9] [31].

AEs and VAEs operate by encoding an input image into a compact latent space representation and subsequently reconstructing it. While AEs employ a deterministic encoding-decoding process, VAEs introduce a probabilistic framework that models the data distribution in the latent space. This approach allows VAEs to better capture variations in pose, expression, and illumination—critical challenges in face verification. However, the probabilistic nature of VAEs can also introduce inconsistencies in feature extraction, making their practical application more challenging [48] [9].

The objective of this project is to investigate the application of AEs and VAEs as feature extractors for face verification systems and explore methods to enhance their *accuracy, reliability, and adaptability.* Specifically, our study focuses on:

- **Refining the training process of VAEs** to improve the consistency of latent space representations for face verification.

- **Evaluating the robustness of AE/VAE-based approaches** across multiple datasets and operational conditions.
- **Comparing the performance of AE/VAE-based feature extraction with CNN-based methods** (ResNet18 trained with ArcFace loss) to assess their viability for face verification tasks.

By addressing these challenges, this project aims to contribute to a better understanding of generative models in face verification and explore strategies to improve their performance relative to more conventional CNN-based approaches. [31] [30].

This paper is organized as follows: **Chapter 2** presents a comprehensive review of prior research and existing approaches in face verification. **Chapter 3** outlines the theoretical foundations of Autoencoders and Variational Autoencoders, detailing their advantages and limitations. **Chapter 4** describes the methodology employed in this study, followed by **Chapter 5**, which presents an in-depth analysis of the experimental results. Finally, **Chapter 6** synthesizes the key insights, discusses the challenges encountered, and outlines potential directions for future research.

Chapter 2

Related Work

2.1 Existing Approaches to Face Verification

Various methods have been developed to address the face verification task [13], ranging from traditional holistic approaches to modern deep learning-based models. These approaches can be broadly categorized into holistic methods, feature-based methods, and unsupervised methods.

2.1.1 Holistic Methods

Holistic, or appearance-based methods, aim to represent faces using a global approach that considers the entire facial region. Techniques like ***Eigenfaces***[23] and ***Fisherfaces*** [36] build low-dimensional representations of facial features based on the statistical distribution of pixel intensities. Despite their pioneering contributions to face verification, these methods are limited by their sensitivity to variations in facial appearance, such as changes in lighting, expression, and pose. Consequently, holistic approaches often fail to perform robustly in uncontrolled environments. To overcome these challenges, local hand-crafted methods were introduced, focusing on extracting geometric features from specific facial regions (e.g., eyes, nose, and mouth) to enhance robustness against environmental changes. However, these methods are constrained by their reliance on manually designed features and the difficulty in determining the optimal size of the feature extraction codebook. Additionally, they often struggle to model the non-linear variations present in real-world facial data, resulting in suboptimal performance.

2.1.2 Feature-Based Methods

Feature-based methods address the non-linearity problem by introducing more sophisticated models that can learn complex representations. Examples include ***Hidden Markov Models (HMMs)*** [11] : These probabilistic models capture sequential information in images and have been used to model the spatial structure of facial features. HMMs provide a more dynamic analysis of facial data compared to holistic methods.

Convolutional Neural Networks (CNNs):[39] CNNs revolutionized face verification by learning hierarchical representations of facial features through deep architectures. They are capable of capturing detailed, non-linear characteristics of images through successive convolutional and pooling layers[43],[1]. However, to effectively capture the intricate details present in facial data, CNNs often require architectures with many deep layers. This leads to the ***vanishing gradient problem*** [20], where gradients diminish as they are backpropagated through the network, resulting in slow or stalled learning.

To address this, ***ResNet (Residual Networks)***[17] were introduced in 2015. This innovative architecture employs *skip connections*[40],[2] that allow gradients to flow directly through the network, effectively mitigating the vanishing gradient issue. The introduction of ResNets made it feasible to train networks with hundreds of layers and achieve impressive results in tasks such as ImageNet classification [44],[6], where a 152-layer ResNet achieved a top-5 error rate of 3.57%, winning 1st place at ILSVRC 2015.

Additionally, after further research on related works, we have learned about ***Siamese Networks*** [29] which is significant development in face verification. Siamese networks leverage a twin-network architecture to learn a similarity function between two face images. The contrastive loss function or triplet loss is commonly used to optimize these models, enhancing their effectiveness in one-shot learning scenarios.

Also, we have to mention about ***Vision Transformers (ViTs)*** [8]. Recently, ViTs have emerged as an alternative to CNNs for face verification. These models utilize self-attention mechanisms to capture global dependencies in an image, achieving competitive performance in large-scale face verification benchmarks.

2.1.3 Unsupervised Methods

Unsupervised methods leverage models that learn representations without the need for labeled data. ***Autoencoders***[4],[7],[10] have become popular in this context due to their ability to learn compressed, meaningful representations of input data. They consist of an encoder-decoder structure, where the encoder compresses input features into a latent space, and the decoder reconstructs the original input from this latent space.

Variational Autoencoders (VAEs) [27]: VAEs extend traditional autoencoders by incorporating probabilistic constraints, enabling the model to learn meaningful latent distributions. This makes them particularly useful for generative face verification tasks.

Additionally to the above explorations, we have to briefly mention about ***Generative Adversarial Networks (GANs)*** [14] and ***Self-Supervised Learning***[5, 32]. GANs, particularly architectures like StyleGAN and StarGAN, have demonstrated strong capabilities in generating highly realistic facial images. Their application in face verification includes data augmentation, domain adaptation, and robustness enhancement.[14]

Self-Supervised Learning [5, 32]: Recent advancements have introduced self-supervised learning techniques, where models learn representations from unlabeled data using contrastive learning objectives. Examples include SimCLR and MoCo, which have demonstrated promising results in face verification without extensive labeled datasets.

2.1.4 Application to This Work

The objective of this work is to harness the power of ***Variational Autoencoders (VAEs)***, a variant of autoencoders, that imposes a probabilistic framework to capture latent distributions of facial features. VAEs have been widely explored in generative modeling tasks, including image synthesis and feature learning, but their application to face verification remains relatively underexplored. By learning latent encodings, VAEs can extract critical facial representations that generalize across variations in pose, expression, and lighting, making them a promising alternative for face verification.

Early works on Autoencoders (AEs) for face verification primarily focused on their ability to perform ***unsupervised feature extraction***. Traditional autoencoders have been utilized in [19, 3], where they were employed to learn compressed representations of facial images. These studies demonstrated that AEs could achieve comparable performance to shallow feature-based approaches, but struggled when compared to deep CNN-based architectures. More recently, researchers have explored ***denoising autoencoders (DAEs)***, which introduce robustness to noise and occlusions [45]. However, a key limitation remains: AEs do not inherently enforce structured latent spaces, leading to feature inconsistencies across different facial images of the same individual.

To address this limitation, Variational Autoencoders (VAEs) were introduced as an enhancement to traditional AEs. VAEs impose a probabilistic constraint on the latent space, ensuring that embeddings follow a continuous distribution. This property makes VAEs more effective in capturing intra-class variations, such as facial expressions, but also introduces challenges in feature stability. Studies such as [26, 42] have explored the use of VAEs for face verification by learning structured latent representations. However, most of these works have relied on ***traditional reconstruction loss functions***, which do not explicitly optimize for verification performance.

Recent approaches have attempted to bridge this gap by incorporating *metric learning* into the training process of VAEs. Works such as [41, 18] have demonstrated that combining VAEs with ***contrastive loss or triplet loss*** can improve the discriminative power of learned embeddings. However, these methods still fall short of **CNN-based approaches**, as they do not fully exploit the angular margin properties that have proven effective in modern face verification.

To enhance the effectiveness of VAEs for face verification, this work proposes the integration of the ***ArcFace Loss*** [12]. ArcFace introduces an additive angular margin penalty to the softmax loss, significantly improving the discriminative power of deep embeddings. Unlike conventional verification methods that rely on Euclidean distance, ArcFace optimizes the embeddings in an angular space, making intra-class distances smaller while increasing inter-class separability. This approach has become state-of-the-art for face recognition, outperforming traditional softmax-based methods.

The results obtained in this study will be compared to a baseline model using ***ResNet***, which has demonstrated strong performance in face verification tasks due to its ability to learn deep, robust representations through skip connections. By evaluating the effectiveness of **VAE-based face verification with ArcFace loss**, this study aims to contribute new insights into the feasibility of generative models as feature extractors for face verification tasks.

Chapter 3

Background

3.1 Concepts and Definitions

3.1.1 Face Recognition and Face Verification

Face recognition [13] and face verification [49] play a major role in AI-based applications of today. The objective in face recognition is to determine who people are from among a wider set of people. However, face verification confirms if two pictures depict the same person, a method often employed in security and authentication. Nevertheless, such cases can sometimes result in a fall in performance due to the presence of less favorable conditions, for instance, bad lighting, angles, and changes in facial expressions.

3.1.2 Autoencoders (AEs)

Autoencoders [4], [7] are a type of neural network designed to learn compact representations of data by encoding information into a smaller space and then reconstructing it. They consist of two main parts: the encoder, which compresses the data, and the decoder, which attempts to reconstruct the original data from this compressed form. Variants like sparse autoencoders, denoising autoencoders, and variational autoencoders (VAEs) allow for different levels of control in data compression. In face verification, autoencoders are valuable because they can learn from data without needing labels, helping to reduce the need for large labeled datasets (self-supervised learning).

3.1.3 Variational Autoencoders (VAEs)

VAEs add a probabilistic layer to traditional autoencoders, allowing them to learn distributions rather than fixed representations. This makes them especially good at handling variations in data. In face verification, VAEs can generalize better by capturing the natural variability in faces, such as differences in pose or lighting.

3.2 Existing Algorithms and Techniques

Over the years, face recognition technology has evolved from simple statistical models to highly advanced deep learning algorithms. Here's a quick look at some major techniques.

3.2.1 Feature-Based Models

Feature-based models analyze specific facial regions rather than the entire face, making them more robust to changes in pose, lighting, and occlusions. Two important approaches here are Hidden Markov Models (HMMs) [11] and Convolutional Neural Networks (CNNs)[43]. However, in this project we monsly focus on CNN, since it is the architecture we will be comparing with AE and VAE.

Convolutional Neural Network (CNN): CNNs are especially powerful for face recognition as they learn complex patterns by applying multiple layers of convolutional filters. These filters can capture deep features like the unique structure of a person's face, adapting to lighting, expression, and pose variations, making them highly effective for real-world applications.

Residual Networks (ResNets) Residual Networks, or *ResNets*, are a deep learning architecture introduced by *Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun* in their 2015 paper *"Deep Residual Learning for Image Recognition"*[17] . ResNets are designed to address some of the fundamental challenges encountered in training very deep neural networks, particularly the *vanishing gradient problem*[20]. This architecture enables the training of much deeper networks without suffering from degradation in performance, which typically occurs as depth increases.

3.2.1.1 The Vanishing Gradient Problem

As neural networks become deeper, it becomes increasingly challenging to propagate gradients back through the layers during training. This issue, known as the *vanishing gradient problem* as shown in the figure 3.1, causes gradients to become extremely small as they pass backward through each layer, resulting in slow learning or even a complete halt in gradient-based learning. The vanishing gradient problem can lead to *degradation*, where deeper networks start to perform worse than shallower ones because they fail to learn effectively.

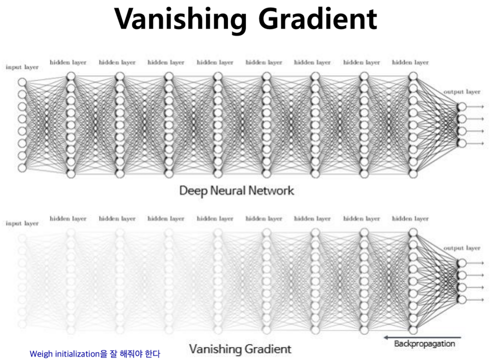


Figure 3.1: Vanishing gradient illustration in a neural network

3.2.1.2 Skip Connections in ResNets

To overcome this problem, ResNets introduce a novel concept known as *skip connections* or *shortcut connections*. Rather than having each layer learn a direct mapping, ResNets reframe the task by letting each layer learn a *residual mapping*. Given an input x, a residual layer learns the function $F(x) := H(x) - x$, where $H(x)$ is the desired underlying mapping. Thus, the output of a residual block is $F(x) + x$, effectively allowing the model to "skip" the learning of an identity mapping when it is beneficial. The figure 3.2 illustrates a residual block architecture.

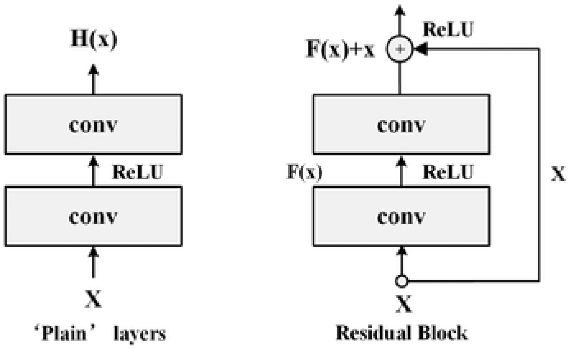


Figure 3.2: A basic residual block with a skip connection

This reformulation allows gradients to flow directly through the network via the skip connections, alleviating the vanishing gradient problem. Consequently, this approach makes it feasible to train networks with hundreds of layers effectively.

3.2.1.3 ResNet Architectures

ResNets come in various architectures, denoted by the number of layers. Common versions include:

- **ResNet-18** and **ResNet-34**: These are shallower ResNet versions, often used in applications where computational efficiency is critical.

- **ResNet-50**, **ResNet-101**, and **ResNet-152**: These deeper versions, designed for large-scale image recognition tasks, can capture more complex patterns and details.

The figure 3.3 shows an exemple of a Resnet18 architecture.

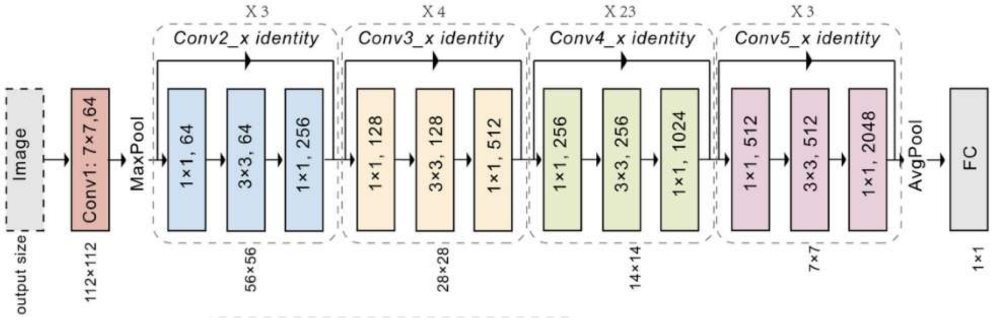


Figure 3.3: An example of a Resnet18 architecture

For instance, ResNet-152, which has 152 layers, achieved a top-5 error rate of 3.57% on the ImageNet dataset, winning 1st place in the ILSVRC 2015 classification competition. The ResNet-50 architecture, another popular choice, uses a "bottleneck" block design to balance depth and computational cost. These architectures have shown remarkable performance not only in image classification but also in tasks like object detection and segmentation.

3.2.2 Autoencoders

Autoencoders provide a way to learn meaningful features for face verification without relying on labeled data. They compress facial data into compact embeddings, which can then be used to compare faces effectively. This capability allows autoencoders to capture complex relationships in facial features, often providing better insights than traditional methods.

An autoencoder typically includes an encoder-decoder structure: the encoder compresses the input into a smaller latent space, while the decoder reconstructs the original data. The model is trained to minimize reconstruction loss, forcing it to learn meaningful, compressed features.

There are various types of autoencoders, including denoising autoencoders (DAEs) and variational autoencoders (VAEs), each serving unique purposes in feature extraction.

Denoising Autoencoders: Denoising autoencoders (DAEs) increase the robustness of feature extraction by training on slightly corrupted images and learning to reconstruct clean images. This method is especially useful for real-world applications, where images can be noisy or incomplete.

Variational Autoencoders (VAEs): VAEs improve upon regular autoencoders by generating probabilistic, rather than deterministic, embeddings, capturing variations in data naturally (the encoder maps the input to a distribution over the latent space). They combine reconstruction loss with a regularization term to create generalizable embeddings.

VAEs are particularly useful in generating synthetic faces for data augmentation, which helps improve verification accuracy in cases with limited labeled data.

Applications in Face Verification: Autoencoders can serve as a foundation for supervised learning in face verification. By pre-training an autoencoder on large, unlabeled datasets, the learned embeddings can later be fine-tuned using labeled data within a neural network. Studies on popular datasets like LFW and YTF have shown that embeddings from autoencoders, combined with simple similarity metrics, achieve results comparable to fully supervised methods.

3.2.3 ArcFace Loss

ArcFace [12], also known as Additive Angular Margin Loss, enhances the separation of classes in face recognition by introducing a margin in angular space. Unlike softmax loss, which doesn't specifically enforce class separability, ArcFace makes intra-class distances smaller and inter-class distances larger.

3.2.3.1 Mathematical Formulation of ArcFace

ArcFace builds on softmax loss by mapping each feature vector to a hypersphere and adding an angular margin between feature vectors and class centers. The standard softmax formula is:

$$L = -\log \frac{e^{W_{y_i}^T x_i}}{\sum_{j=1}^{N} e^{W_j^T x_i}},$$

where W_j represents the weight vector for class j, and x_i is the embedding of sample i. ArcFace adds an angular margin m to the angle between x_i and W_{y_i}, enhancing class separation:

$$L_{\text{ArcFace}} = -\log \frac{e^{s \cdot \cos(\theta_{y_i} + m)}}{e^{s \cdot \cos(\theta_{y_i} + m)} + \sum_{j \neq y_i}^{N} e^{s \cdot \cos \theta_j}},$$

where: θ_{y_i} is the angle between x_i and W_{y_i}, m is the angular margin and s is a scaling factor for stable convergence.

3.2.3.2 Advantages of ArcFace

1. **Improved Class Distinction:** By introducing an angular margin, ArcFace optimizes the distance between classes on a hypersphere, which is effective for varied face conditions.
2. **Stable Training:** The angular margin aids convergence, making it more reliable than traditional methods.
3. **Enhanced Separability:** ArcFace enforces both intra-class compactness and inter-class separability, crucial for high-performance face verification.

3.2.3.3 Applications of ArcFace

ArcFace has achieved top performance on benchmarks like LFW and MegaFace. It excels in challenging scenarios with varying lighting, poses, and expressions by maintaining consistent decision boundaries across diverse facial data.

3.2.4 Benchmark Datasets for Face Verification

Evaluating face verification models requires extensive benchmarking across datasets that capture real-world variations, such as pose, age, illumination, and occlusion. Various datasets have been developed over the years, each focusing on different challenges that affect the performance of biometric authentication systems[37, 22, 35]. Further we have collected and discussed about several benchmark datasets for face verification.

3.2.4.1 Labeled Faces in the Wild (LFW)

Labeled Faces in the Wild (LFW) [21] is one of the most widely used benchmarks for face verification. It contains over 13,000 images of faces collected from the web, with varying poses, lighting conditions, and expressions. LFW is particularly challenging because it features unconstrained face images, making it an effective benchmark for evaluating the generalization capability of verification models. However, due to its limited diversity and relatively small size, it may not fully represent more complex verification scenarios.

3.2.4.2 YouTube Faces (YTF)

The YouTube Faces (YTF) dataset [46] extends the LFW benchmark by including video sequences rather than static images. It consists of 3,425 videos of 1,595 different people, providing a unique opportunity to evaluate models under varying motion and resolution conditions. YTF is commonly used to assess the robustness of verification models in video-based authentication scenarios.

3.2.4.3 MS-Celeb-1M

MS-Celeb-1M [16] was introduced to scale face recognition and verification tasks to millions of identities. Although the dataset originally contained noisy labels, later versions were curated to provide more reliable identity annotations. MS-Celeb-1M is useful for training deep learning models that require large-scale identity distributions. However, due to ethical concerns regarding data collection, access to this dataset has been restricted.

3.2.4.4 CASIA-WebFace

CASIA-WebFace [47] is a large-scale dataset containing over 490,000 images of 10,575 individuals. It is widely used for training face verification models, particularly in learning deep representations. CASIA-WebFace is useful for evaluating how well models perform on large identity spaces and how they generalize to new, unseen individuals.

3.2.4.5 MegaFace

MegaFace [25] is a challenging dataset designed to evaluate large-scale face verification performance. It contains over 1 million images from more than 690,000 individuals. The dataset is structured to test verification performance at scale, specifically analyzing how well models handle low inter-class variability and high intra-class variance. MegaFace is particularly useful for assessing models that need to distinguish identities in real-world, large-scale applications.

3.2.4.6 AgeDB

AgeDB [34] is the first publicly available dataset designed for age-invariant face verification. It contains 16,488 images of 568 identities with annotated age information. This dataset is essential for evaluating how well models generalize across age progression, an important factor in long-term biometric authentication.

3.2.4.7 IJB-A, IJB-B, and IJB-C

The IARPA Janus Benchmark (IJB) series [28]—including IJB-A, IJB-B, and IJB-C—are among the most rigorous datasets for face verification. Unlike LFW, these datasets feature images with large pose variations, motion blur, and occlusions. They are widely used to evaluate face verification under real-world conditions where traditional methods often fail.

3.2.4.8 CelebA

CelebA [33] is a large-scale dataset that includes over 200,000 celebrity face images with 40 attribute annotations per image. It is commonly used to test how well models handle occlusions, expressions, and accessories such as glasses, hats, and beards. CelebA's attribute annotations allow researchers to assess how different facial characteristics impact verification performance.

3.2.5 Pose, Illumination, and Expression (PIE) Dataset

The PIE dataset [15] contains images of 68 individuals captured under 13 different poses, 43 different illumination conditions, and 4 different expressions. It is valuable for testing models on pose, lighting, and expression variations, three of the most challenging factors in real-world face verification.

3.2.6 Implications for Model Evaluation

Each of these datasets presents unique challenges that face verification models must overcome. While datasets such as LFW and MegaFace provide benchmark performance comparisons, datasets like IJB-A and PIE are essential for testing robustness to extreme variations. This study will incorporate CASIA-WebFace for training and LFW, MegaFace, and AgeDB for evaluation, ensuring that the proposed approach is assessed across diverse real-world conditions.

Table 3.1: Summary of Benchmark Face Verification Datasets

Dataset	**No. of Images**	**Identities**	**Key Challenge**
LFW [21]	13,233	5,749	Unconstrained faces, varied conditions
YTF [46]	3,425 videos	1,595	Video-based verification
MS-Celeb-1M [16]	10M+	1M+	Large-scale face identification
CASIA-WebFace [47]	490,000+	10,575	Training deep face models
MegaFace [25]	1M+	690,000+	Large-scale identity classification
AgeDB [34]	16,488	568	Age variation
IJB-C [28]	23,124	3,531	Pose and occlusion variations
CelebA [33]	200,000+	10,177	Attribute-based analysis
PIE [15]	41,368	68	Pose, illumination, and expression

The table 3.1 summarizes the key datasets used for face verification evaluation, highlighting the primary challenge each dataset addresses. By integrating these datasets into model evaluation, we can provide a comprehensive assessment of verification performance across multiple real-world conditions. This approach ensures that models are not only optimized for high accuracy but also for robustness under variations in pose, age, illumination, and occlusion.

Chapter 4

Our Methodology and Approach

In this project, we wanted to explore the efficacy of VAEs in the context of face verification, a critical component of modern facial recognition systems. Face verification (Figure 4.1) specifically entails the process of determining whether two given facial images correspond to the same individual.[48] This task plays a pivotal role in various applications, such as security systems, identity verification, and personalized services. Unlike conventional approaches that often rely on engineered features or deep convolutional architectures, our methodology utilizes the representational power of VAEs to encode facial data into compact and meaningful latent spaces.[24]

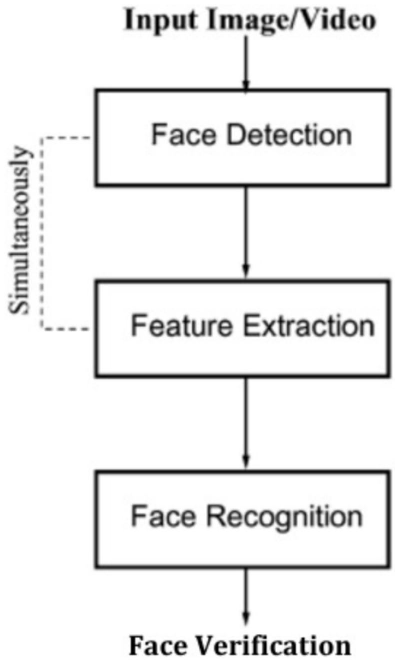


Figure 4.1: Configuration of face recognition system

Our work is motivated by the unique potential of VAEs to capture underlying patterns in data through probabilistic modeling. Building upon prior research in this area,[38] we were seeking to enhance the performance of VAEs for face verification by integrating the Adaface pipeline, which is a robust framework designed to refine embeddings for face-related tasks.[30] Additionally, we employed diverse benchmarking datasets to ensure a comprehensive evaluation of our model, assessing its generalizability across varied contexts and conditions.

Our research methodology can be divided into two primary stages. ***In the first stage***, we focused on face recognition, where the model is trained using various loss functions to optimize the embedding space effectively. This step emphasizes learning discriminative representations that encode the critical features of facial images. ***The second stage*** involves the implementation of an evaluation pipeline. Here, we utilized the embeddings generated during the first stage to perform face verification tasks. By systematically assessing the similarity or dissimilarity between embeddings, this stage provides a detailed analysis of the model's performance. The overall process is visualized in the accompanying Figure 4.2 below, which illustrates the transition from feature learning to verification.

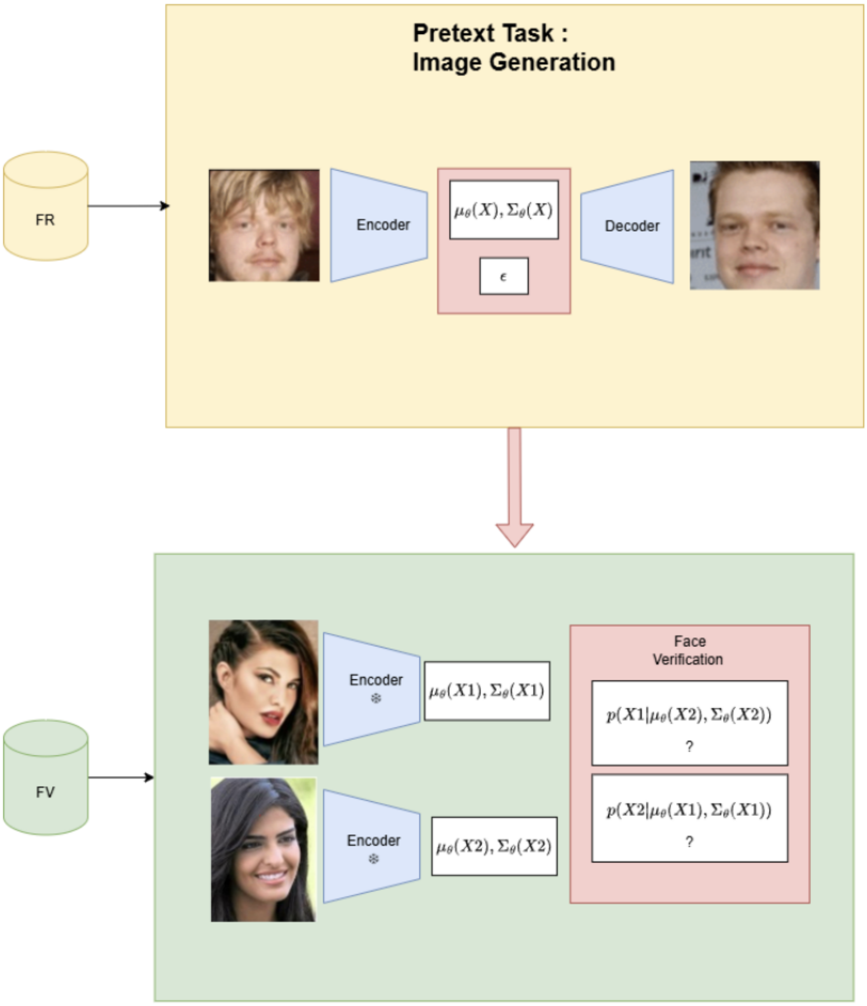

Pretext Task :
Image Generation
FR
Encoder
$\mu_\theta(X), \Sigma_\theta(X)$
ϵ
Decoder
FV
Encoder
$\mu_\theta(X1), \Sigma_\theta(X1)$
Encoder
$\mu_\theta(X2), \Sigma_\theta(X2)$
Face Verification
$p(X1|\mu_\theta(X2), \Sigma_\theta(X2))$
?
$p(X2|\mu_\theta(X1), \Sigma_\theta(X1))$
?

Figure 4.2: Two primary stages of the project

Through this two-step approach, we aimed to provide deeper insights into the capabilities and limitations of VAEs in face verification. Furthermore, this exploration contributes to the broader understanding of generative models as feature extractors, offering potential pathways for future advancements in the field.

4.1 Implementation of the Evaluation Pipeline

The first step in our research involved implementing an evaluation pipeline designed to assess the effectiveness of results obtained from both the baseline model and the VAE. This pipeline plays a crucial role in evaluating the performance of a face verification system by comparing pairs of images and determining their similarity based on predefined metrics.

4.1.1 Underlying Principle

The evaluation pipeline operates on the principle of comparing the embeddings of two input images. Embeddings are high-dimensional representations generated by the model for the purpose of capturing facial features. To evaluate the similarity between two images, a similarity metric is employed, and in this case, the Euclidean distance.

- **Small Distance:** A small distance between two embeddings implies that the images likely belong to the same individual.
- **Large Distance:** Conversely, a larger distance indicates that the images are of two different individuals.

To automate the decision-making process, we established a threshold distance. The threshold serves as a critical determinant:

- If the computed distance exceeds the threshold, the two images are classified as belonging to different individuals.
- If the distance is below the threshold, the images are classified as belonging to the same individual.

4.1.2 Threshold Determination

To accurately determine the threshold, we adopted a cross-validation approach using two benchmark datasets:

1. **LFW (Labeled Faces in the Wild) Dataset:** A standard dataset widely used for face verification tasks.
2. **LFW-C (Labeled Faces in the Wild-Curated) Dataset:** A curated version of the LFW dataset designed for robust evaluation.

The following methodology was employed:

1. **Data Preparation:** A dataset was prepared consisting of pairs of embeddings along with their corresponding labels:
    - *Label 0:* Indicates the embeddings belong to different individuals.
    - *Label 1:* Indicates the embeddings belong to the same individual.

2. **Distance Computation:** For each pair of embeddings, the Euclidean distance was computed to measure the similarity between the images.

3. **Data Visualization:** The distributions of computed distances were plotted separately for the two classes (0 and 1).

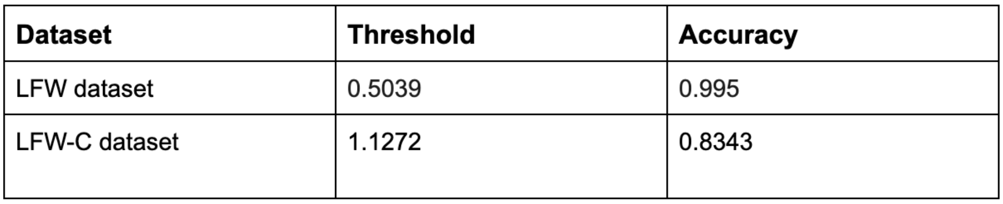
4. **Threshold Selection:** The intersection point of the two distributions was identified. This value was chosen as the optimal threshold for classification, as it minimizes errors in distinguishing between the two classes.

5. **Cross-Validation:** A 10-fold cross-validation procedure was conducted to validate the robustness of the threshold and assess the overall performance. Accuracies were computed across all folds to ensure the reliability of the evaluation process.

4.1.3 Rationale for Methodology

The use of Euclidean distance as a similarity metric was motivated by its simplicity and effectiveness in measuring embedding space distances. The application of cross-validation ensures that the results are not biased by specific splits of the dataset, thereby enhancing the generalizability of the findings. The inclusion of both the LFW and LFW-C datasets further reinforces the robustness of the evaluation pipeline, as the two datasets introduce complementary challenges for the verification task.

Below in the table there are the results we have obtained:



Dataset	Threshold	Accuracy
LFW dataset	0.5039	0.995
LFW-C dataset	1.1272	0.8343

Figure 4.3: Results of the Threshold and Accuracy Values for Face Recognition on LFW and LFW-C Datasets

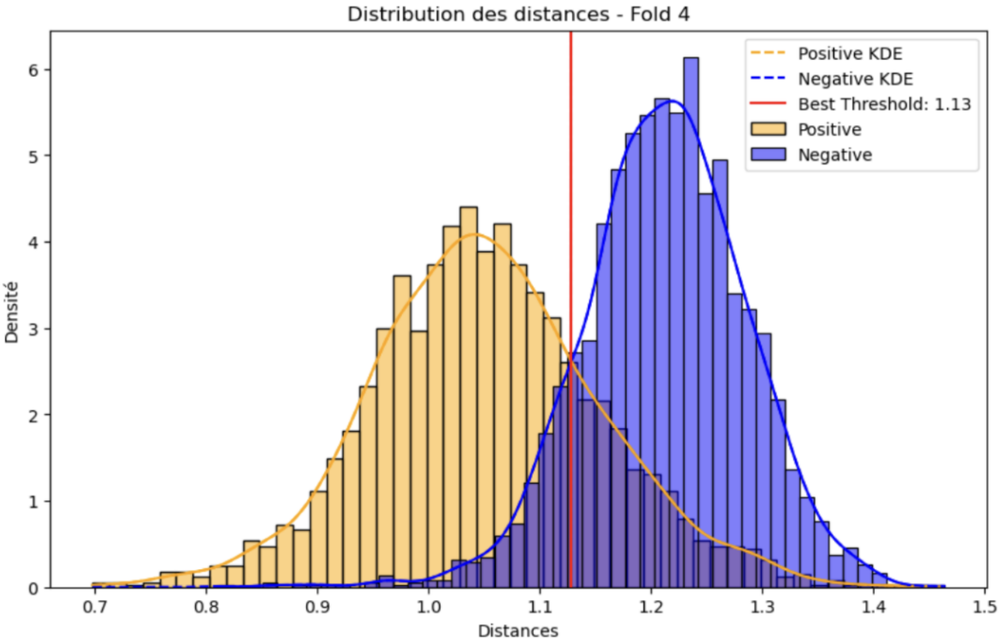


Figure 4.4: Distance Distribution and Optimal Threshold for Positive and Negative Pairs at k=4 of the cross validation

This figure presents the distribution of Euclidean distances between pairs of embeddings for Fold 4, derived from cross-validation in face verification tasks. The yellow histogram and curve represent the positive pairs (label 1), where the embeddings belong to the same individual. The blue histogram and curve correspond to the negative pairs (label 0), where the embeddings belong to different individuals. The red vertical line indicates the optimal threshold value (1.13), determined at the intersection of the two distributions. This threshold minimizes misclassification errors by effectively separating the two classes. The visualization is part of a systematic threshold determination process applied to both LFW and LFW-C datasets, ensuring robust and generalized performance through cross-validation.

4.2 Performing a face recognition task using ResNet18 Model

4.2.1 Resnet18 Model Architecture

ResNet18 is a convolutional neural network that uses residual blocks to overcome the vanishing gradient problem commonly encountered in deep neural networks. Unlike plain layers, residual blocks introduce shortcut connections that allow information to skip layers, improving gradient flow and enabling the training of very deep networks. ResNet18 specifically consists of 18 layers, structured with a combination of convolutional, pooling, and fully connected layers, making it effective for tasks like image classification while maintaining computational efficiency.

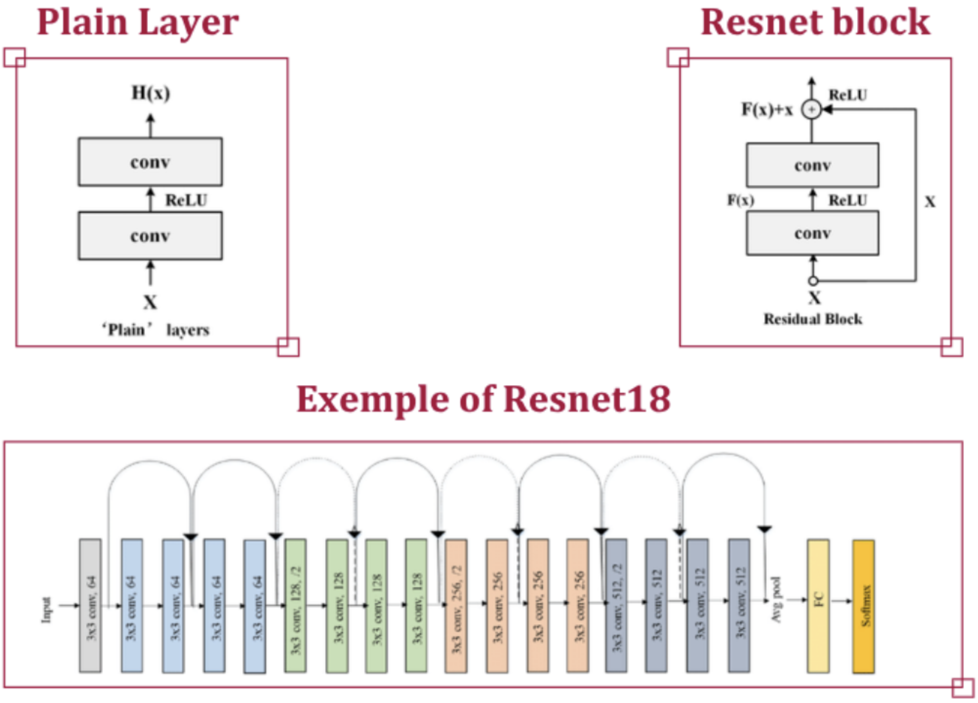


Figure 4.5: Resnet18 Model Architecture

4.2.2 Using Resnet18 for Face Recognition Task

To use ResNet18 for the face verification (FV) task, the process begins by first training the model for a face recognition task. In this phase, ResNet18 is employed to classify input images based on their identities. The model architecture includes ResNet layers for feature extraction, followed by fully connected layers and a softmax function to output a probability distribution over the possible identities. The goal of this phase is to train the network to learn meaningful features that can represent the identity of an individual effectively. The weights of the ResNet layers are optimized during this process to capture discriminative facial features. Once the face recognition task is complete, the next step is to use the trained model for face verification. At this stage, the final classification layer, including the fully connected, is removed, leaving only the ResNet backbone. This backbone is used to extract embeddings for input images. These embeddings are high-dimensional vectors that represent the essential features of the images. To determine whether two images belong to the same individual, the embeddings of the two images are compared using the evaluation pipeline explained before. This pipeline involves computing a similarity metric and setting a threshold to classify the pairs. Through this approach, we assess whether the images represent the same person or different individuals. This methodology allows ResNet18 to first learn essential facial representations during the face recognition task and then repurpose these representations for face verification by comparing the embeddings generated for image pairs.

1. **Training Resnet18 on Casia Dataset using Cross Entropy Loss:**

The first step of this work involved training a ResNet18 model on the CASIA dataset, a well-known dataset used for face recognition tasks. The CASIA-WebFace dataset consists of over 10,000 individuals and more than 490,000 facial images, which makes it a diverse and comprehensive dataset for training models to recognize and classify facial features. Its wide variety of images, spanning different poses, lighting conditions and facial expressions, allows models to learn generalizable facial embeddings. For training, we utilized the ***CrossEntropyLoss*** function, a standard choice for multi-class classification tasks like face recognition. ***CrossEntropyLoss*** measures the difference between the predicted probability distribution (output of the softmax layer) and the actual class labels. The training was conducted using the Adam optimizer with a learning rate of 10-4. The model was trained over 30 epochs, with its performance evaluated periodically using a validation dataset. During training, both the training loss and validation loss were tracked to monitor the model's convergence and generalization performance. The best-performing model, based on the validation loss, was saved for subsequent use in the face verification task.

Here's the pseudo-algorithm for the training loop code:

Algorithm 1 Train ResNet18 on CASIA Dataset

```
Initialize Adam optimizer, CrossEntropyLoss, and epoch count (30 epochs).
for each epoch do
    Training Phase:
    Set the model to training mode.
    for each batch in the training data do
        Compute logits.
        Calculate loss.
        Backpropagate and update weights.
        Record batch training loss.
    Validation Phase:
    Set the model to evaluation mode.
    for each batch in the validation data do
        Compute logits.
        Calculate loss.
        Evaluate accuracy.
        Record batch validation loss and accuracy.
    Metrics and Saving:
    Calculate average training loss, validation loss, and accuracy for the epoch.
    Save the model if validation loss improves.
Save Statistics:
Save training and validation metrics for analysis.
```

2. **Training Resnet18 on Casia Dataset using ArcFace Loss:**

   What is ArcFace Loss?

   ArcFace loss or Additive Angular Margin Loss, is a loss function designed to improve the discriminative power of face recognition models. It enhances the intra-class compactness (making features of the same class closer together) and the inter-class separability (pushing features of different classes farther apart) in the learned feature space. Unlike traditional softmax loss, which focuses on separability but lacks strong geometric constraints, ArcFace directly optimizes the angular margin on a hypersphere. This geometric interpretation ensures that face embeddings are well-clustered and distinct. ArcFace introduces a margin into the angle between the feature vector (embedding) and the target class center. This margin, added in the angular space, ensures a better separation between classes. The key idea is that the embedding features are distributed on a hypersphere with a fixed radius, and an additive angular margin improves the discriminative capacity by enforcing a consistent geodesic distance between classes. It works by projecting feature vectors and class weights onto a hypersphere, ensuring they lie on a uniform surface. It calculates the angle between an embedding and its class center using the cosine function and adds an angular margin to create stricter decision boundaries. The softmax function is then used to compute probabilities, incorporating this margin to make embeddings more structured within classes and better distinguished across classes. This approach focuses on optimizing angular distances, providing a systematic way to structure embeddings on the hypersphere while being computationally efficient and straightforward to implement in deep learning frameworks.

The loss is defined as:

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

Where:

- θ_{y_i}: Angle between the embedding and the true class center.
- s: Scale factor controlling the hypersphere's radius.
- m: Additive angular margin.
- θ_j: Angle between the embedding and other class centers.

We used the ArcFace class provided in the InsightFace repository, which implements the ArcFace loss as described in the original paper. The class handles the addition of the angular margin to logits and scales them to improve the discriminative ability of the embeddings. Here are the important steps involved in the ArcFace implementation:

1. Normalize the embeddings and class weights to ensure they lie on the hypersphere.
2. Compute the cosine similarity (logits) between the embeddings and class weights.
3. Adjust the target class logits by adding an angular margin to the angle.
4. Scale all logits by a predefined scale factor to enhance optimization.
5. Return the modified logits for further processing.

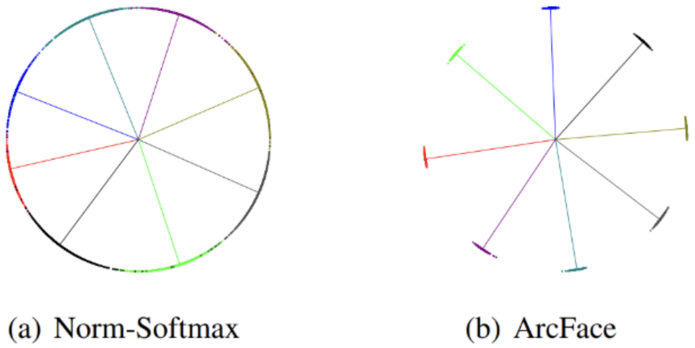

(a) Norm-Softmax (b) ArcFace

Figure 4.6: Comparison of Feature Distributions: Norm-Softmax vs. ArcFace

4.3 Extracting embedding after training and using Evaluation Pipeline for FV task

After training the two models, one using CrossEntropy loss and the other using ArcFace loss, we used the backbones of these pretrained models to extract embeddings for each image in the dataset. These backbones representing the feature extraction components of the models, generated high-dimensional feature vectors that encapsulate the essential characteristics of the input images. Once the embeddings were obtained, we fed them into the previously implemented evaluation pipeline. This pipeline computes the similarity between pairs of embeddings using the Euclidean distance, and applies the threshold to classify whether the images belong to the same individual or not. By using the same evaluation pipeline for both models, we ensured a fair comparison of their performance in the face verification task.

Algorithm 2 Embedding Extractor

```
Input: Backbone model backbone, Input tensor x.
Output: Flattened embedding x.
Initialize the EmbeddingExtractor class with the backbone.
Define forward function for the model.
Forward Pass:
Pass input x through the following layers of the backbone sequentially:
    1. Convolutional layer (conv1).
    2. Batch normalization (bn1).
    3. ReLU activation function (relu).
    4. Max pooling layer (maxpool).
Pass x through deeper layers of the backbone:
    1. Layer block 1 (layer1).
    2. Layer block 2 (layer2).
    3. Layer block 3 (layer3).
    4. Layer block 4 (layer4).
Apply global average pooling (avgpool) to x.
Flatten the pooled tensor x along dimension 1.
Return the flattened tensor x.
Create an instance of EmbeddingExtractor with the pre-trained model as backbone.
Move the model to the specified device.
```

Implementation of the VAE

The third stage of the project involved implementing a VAE from scratch, a deep generative model that learns a compressed representation of the input data and can reconstruct the original data from its latent space.

4.3.1 CASIA Dataset

For training the VAE, we used the CASIA dataset, the same dataset utilized for training the ResNet baseline. The CASIA dataset contains diverse and rich facial image data, which makes it suitable for evaluating the model's ability to learn meaningful latent representations.

4.3.2 Architectural Components

The VAE comprises two primary components, see **Figure 4.7** below:

4.3.2.1 Encoder

The encoder is responsible for mapping input images to a latent representation in a lower-dimensional space. Its architecture consists of a series of convolutional layers designed to extract hierarchical features while progressively reducing the spatial dimensions of the input image. The key details of the encoder are as follows:

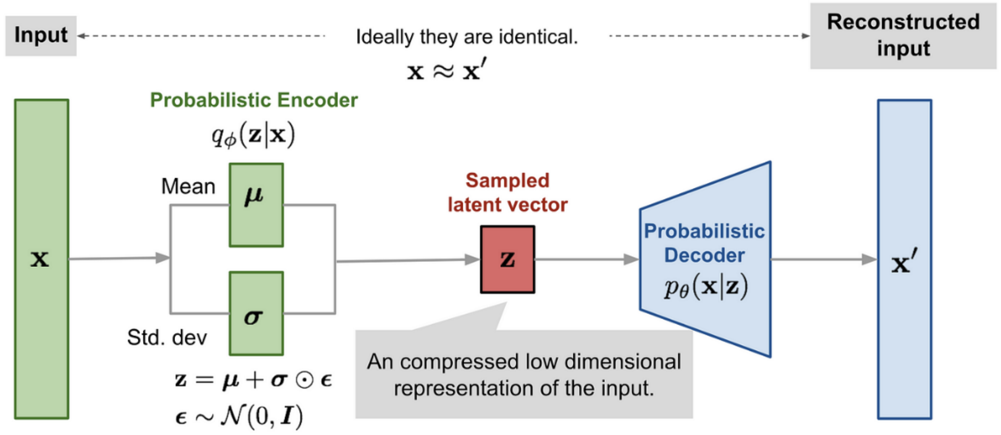

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
x
Mean
$\boldsymbol{\mu}$
$\boldsymbol{\sigma}$
Std. dev
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
Sampled latent vector
z
An compressed low dimensional representation of the input.
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$

Figure 4.7: VAE Architecture

- Each convolutional layer is followed by batch normalization to stabilize learning and LeakyReLU activation to introduce nonlinearity.
- The convolutional layers use a kernel size of 3 × 3, a stride of 2, and padding of 1, effectively halving the spatial dimensions with each layer.
- The encoder accepts 3-channel RGB images and generates a flattened feature map of size 512 × 4 × 4 at the final layer.

4.3.2.2 Latent Space

The output of the encoder is projected into a latent space by two distinct layers:

- **Mean Layer**: Computes the mean (μ) of the latent variables.
- **Log Variance Layer**: Computes the logarithm of the variance ($\log \sigma^2$), which is used for the reparameterization trick.

4.3.2.3 Decoder

The decoder is responsible for reconstructing the input image from the latent space representation. It uses a sequence of transpose convolutional layers to progressively upsample the latent representation back to the original image dimensions. The specifics of the decoder are as follows:

- The architecture mirrors the encoder, with convolutional transpose layers replacing the convolutional layers.
- Each layer is followed by batch normalization and LeakyReLU activation to preserve stability during reconstruction.
- The final output layer employs a 3 × 3 transpose convolution followed by a Tanh activation function to normalize the pixel values between -1 and 1.

4.3.3 Reparameterization Trick

One of the key innovations in VAEs is the reparameterization trick, which enables gradients to flow through the stochastic latent space during backpropagation. This is achieved by sampling from a standard normal distribution and transforming it using the computed mean and standard deviation:

$$z = \mu + \sigma \cdot \varepsilon, \quad \text{where} \, \varepsilon \sim \mathcal{N}(0, I).$$

This process allows the model to generate latent representations while preserving differentiability.

4.3.4 Forward Pass

Once the components of the VAE were implemented, the forward pass of the model was defined to process the data as follows:

1. The encoder processes the input image to generate the mean (μ) and log variance ($\log \sigma^2$) of the latent space.

2. The reparameterization trick is applied to sample from the latent space using the computed mean and variance.

3. The decoder processes the sampled latent representation to reconstruct the original input image.

4. The model outputs the reconstructed image along with the mean and log variance, which are used for loss computation.

4.3.5 Conclusion

The implementation of the VAE architecture allows for the creation of compressed, meaningful representations of input images while enabling high-quality reconstructions. This foundational architecture will be further utilized for downstream tasks and evaluations.

4.4 Loss Function Description

The Variational Autoencoder (VAE) employs a composite loss function that is critical for optimizing its dual objectives: reconstructing input data and learning a meaningful latent space representation. This loss function comprises two primary components: Reconstruction Loss and the Kullback-Leibler Divergence (KLD).

4.4.1 Reconstruction Loss

Type: Mean Squared Error (MSE)

Objective: The Reconstruction Loss quantifies the fidelity of the reconstructed data by comparing it to the original input data. Specifically, it measures the pixel-wise squared differences between the original images (x) and the reconstructed images ($\hat{x}$). This term ensures that the VAE effectively captures the input data's salient features and minimizes reconstruction error, thereby encouraging the model to generate outputs that closely resemble the input data.

Calculation: This component is implemented using PyTorch's `nn.functional.mse_loss` function with the argument `reduction='sum'`. Summing the loss over all elements within a batch strengthens the gradient signal and stabilizes optimization, particularly for small batch sizes. Mathematically, the Reconstruction Loss is represented as:

$$\text{Reconstruction Loss} = \sum_{i=1}^{n} (x_i - \hat{x}_i)^2$$

where x_i denotes the original data, and $\hat{x}_i$ represents the reconstructed output.

4.4.2 Kullback-Leibler Divergence (KLD)

Objective: The KLD term acts as a regularizer, encouraging the learned latent variable distribution $q(z|x)$ to approximate a prior distribution, typically the standard normal distribution $\mathcal{N}(0, I)$. This regularization mitigates overfitting and stabilizes the training process by ensuring that the latent space is well-structured and generalizable.

Calculation: The KLD term is derived as follows:

$$\text{KLD} = -0.5 \times \sum \left(1 + \log(\sigma^2) - \mu^2 - \sigma^2\right)$$

Here, μ and σ represent the mean and standard deviation of the latent variable distribution, respectively. The KLD term penalizes deviations from the prior distribution, driving the model toward a more structured and interpretable latent space.

4.4.3 Total Loss Function

The total loss function combines the Reconstruction Loss and the KLD term. This balance ensures that the VAE optimizes both reconstruction quality and latent space regularization. The total loss is expressed as:

$$\text{Total Loss} = \text{Reconstruction Loss} + \text{KLD}$$

By jointly optimizing these terms, the VAE learns to generate realistic reconstructions while maintaining a robust and meaningful latent space representation.

4.5 Parameters Used

The training configuration for the Variational Autoencoder (VAE) on the CASIA dataset involved carefully chosen hyperparameters and architectural considerations. These parameters were optimized to ensure a balance between computational efficiency, model performance, and training stability.

4.5.1 Image Dimension

The input images from the CASIA dataset were preprocessed to have dimensions of 3 × 112 × 112, where:

- **3:** Represents the three color channels (Red, Green, Blue) in the RGB format.
- **112 x 112 pixels:** Specifies the spatial resolution of each image. This resolution was chosen to capture sufficient detail while remaining computationally manageable.

4.5.2 Batch Size

A batch size of **32** was utilized during training. This value provides an optimal trade-off between computational efficiency and the stability of the training process:

- **Efficiency:** Enables effective utilization of GPU memory, allowing multiple data points to be processed simultaneously.
- **Gradient Approximation:** Ensures that the computed gradients are meaningful and representative of the dataset.
- **Generalization:** Allows for diverse representations of the dataset in each batch, preventing overfitting and improving model generalization.

4.5.3 Learning Rate

The initial learning rate was set to 10^{-5}. This small value was selected to ensure stable and gradual updates to the model parameters, preventing overshooting in the optimization process.

Weight Decay

A weight decay of 10^{-5} was applied as a regularization technique. This parameter penalizes large weights in the model, reducing the likelihood of overfitting by discouraging the model from fitting noise in the training data.

4.5.4 Optimizer

The **Adam optimizer** was employed for training. Adam is widely regarded for its adaptive learning rates and efficient handling of noisy or sparse gradients. By averaging recent gradients in each dimension, it helps stabilize training and accelerates convergence.

4.5.5 Learning Rate Scheduler

To further enhance training, a **StepLR scheduler** was utilized to dynamically adjust the learning rate across epochs. The scheduler was initialized with the following parameters:

- **Step Size:** 10 epochs
- **Gamma:** 0.1

This configuration reduces the learning rate to 10% of its current value every 10 epochs, promoting better convergence during later stages of training.

4.6 VAE Training Procedure

The training of the Variational Autoencoder (VAE) was conducted over **30 epochs** using the following structured approach:

Algorithm 3 Train a VAE on CASIA Dataset

```
Input: Model model, Optimizer optimizer, Number of epochs epochs, Device device.
Output: Trained model with minimized loss.
Set the model to training mode (model.train()).
for each epoch from 1 to epochs do
  Initialize overall_loss to 0.
  for each batch (x, _) in casia_dataloader_train do
    Transfer input data x to the specified device.
    Reset gradients in the optimizer (optimizer.zero_grad()).
    Perform forward pass to compute:
        Reconstructed data x_hat, Mean mean, and Log variance log_var.
    Compute the loss using loss_function(x, x_hat, mean, log_var).
    Accumulate the loss into overall_loss.
    Perform backward pass to compute gradients (loss.backward()).
    Update model parameters using the optimizer (optimizer.step()).
  Adjust the learning rate using the scheduler (scheduler.step()).
  Print the epoch loss and average loss:
     loss: overall_loss
     Average Loss: overall_loss / len(casia_dataloader_train.dataset).
```

4.6.1 Initialization

The VAE model was initialized in training mode, enabling specific layers (e.g., dropout and batch normalization) to adapt their behavior to the training phase.

4.6.2 Epoch Iteration

The training loop iterated over the predefined number of epochs. Each epoch aimed to minimize the loss function and improve the VAE's performance in reconstructing images.

4.6.3 Batch Processing

For each batch in the training dataset, the following steps were performed:

1. **Data Loading and Preprocessing:**

   - Input images were loaded and normalized.
   - The images were transferred to the designated computing device (typically a GPU) for efficient processing.

2. **Forward Pass:**

   - The VAE computed the reconstructed images ($\hat{x}$), mean (μ), and log variance ($\log(\sigma^2)$) for the latent variables in the current batch.

3. **Loss Calculation:**

   - The combined loss (Reconstruction Loss + Kullback-Leibler Divergence) was calculated for the batch.

4. **Backpropagation:**

   - The loss was backpropagated through the network to compute gradients with respect to the model parameters.

5. **Optimizer Step:**

   - The optimizer updated the model's parameters based on the computed gradients.

6. **Scheduler Step:**

   - After all batches in an epoch were processed, the learning rate scheduler adjusted the learning rate according to its policy, ensuring smooth convergence in subsequent epochs.

4.6.4 Logging

At the end of each epoch, the following metrics were computed and logged:

- **Total Loss:** The overall loss for the entire epoch.
- **Average Loss per Sample:** The mean loss across all samples in the epoch.

These metrics provided valuable insights into the model's progress and guided adjustments to the training process if necessary.

4.6.5 Repetition

The above steps were repeated for the full 30 epochs. This systematic approach gradually reduced the loss, leading to a well-trained VAE capable of reconstructing images and generating meaningful latent representations.

Chapter 5

Experiments and Evaluation

For accessing the code repository, please visit: **GitHub link**

5.1 Datasets used for experiments

5.1.1 LFW (Labeled Faces in the Wild) Dataset

The Labeled Faces in the Wild (LFW) dataset is a widely used benchmark dataset for face verification and recognition tasks. It contains 13,233 facial images of 5,749 unique individuals, collected from online sources. The dataset is specifically designed for face verification, where the goal is to determine whether two facial images belong to the same person. Each image in LFW is labeled with the corresponding individual's name, and the dataset provides a standardized evaluation protocol with predefined pairs of images for testing. The dataset emphasizes real-world variability, including differences in lighting, resolution, and occlusions, making it a gold standard for evaluating the performance of face recognition systems.

5.1.2 CASIA-WebFace Dataset

The CASIA-WebFace dataset is a large-scale facial recognition dataset designed for training deep learning models. It contains approximately 10,575 unique identities and over 494,414 facial images, collected from the web. The dataset includes a wide variety of facial images with different poses, lighting conditions, expressions, and backgrounds, making it a challenging and diverse resource for developing robust face recognition systems. CASIA-WebFace is primarily used for tasks such as face identification and verification, providing a benchmark for training feature extractors like ResNet-based models. Its broad variability ensures that models trained on this dataset generalize well to real-world scenarios.

5.2 Performing Face Recognition Task

5.2.1 Training ResNet18 on CASIA Dataset using Cross Entropy Loss

The training was conducted using the Adam optimizer with a learning rate of 10^{-4}. The model was trained over 30 epochs, with its performance evaluated periodically using a validation dataset. During training, both the training loss and validation loss were tracked to monitor the model's convergence and generalization performance. The best-performing model, based on the validation loss, was saved for subsequent use in the face verification task.

Algorithm 4 Train ResNet18 on CASIA Dataset

```
Initialize: Adam optimizer, CrossEntropyLoss, epoch count (30 epochs).
for each epoch do
    Training Phase:
    Set the model to training mode.
    for each batch in the training data do
        Compute logits.
        Calculate loss.
        Backpropagate and update weights.
        Record batch training loss.
    Validation Phase:
    Set the model to evaluation mode.
    for each batch in the validation data do
        Compute logits.
        Calculate loss.
        Evaluate accuracy.
        Record batch validation loss and accuracy.
    Metrics and Saving:
    Calculate average training loss, validation loss, and accuracy for the epoch.
    Save the model if validation loss improves.
Save Statistics:
Save training and validation metrics for analysis.
```

Here are the first results we have achieved using the ResNet18 model. The accuracy reaches around 0.67 using the cross EntropyLoss.

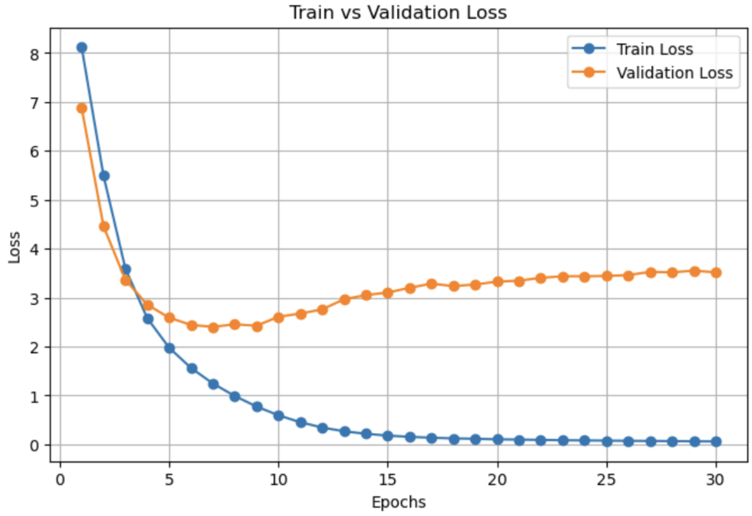


Figure 5.1: Training vs. Validation Loss for ResNet18 Model on CASIA Dataset

Training Resnet18 on Casia Dataset using Arcface loss: Initially, we used the same

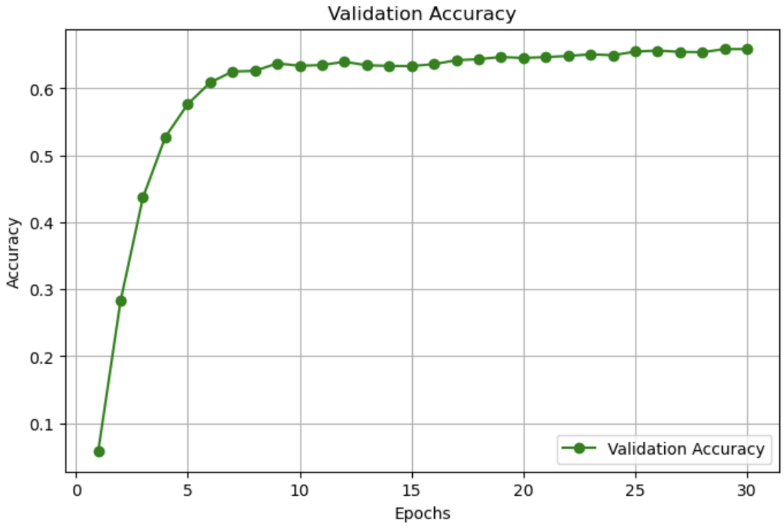


Figure 5.2: Validation Accuracy Progression Over 30 Epochs for ResNet18 Model on CASIA Dataset

parameters as in the previous training setup, including the optimizer, learning rate, and number of epochs. However, the training did not perform well under these settings. To address this, we updated the training parameters and introduced the OneCycleLR scheduler to improve optimization and trained for 100 epochs. The ***OneCycleLR scheduler*** is a learning rate policy designed to maximize training efficiency by starting with a low learning rate, gradually increasing it to a peak, and then decreasing it toward the end of training.

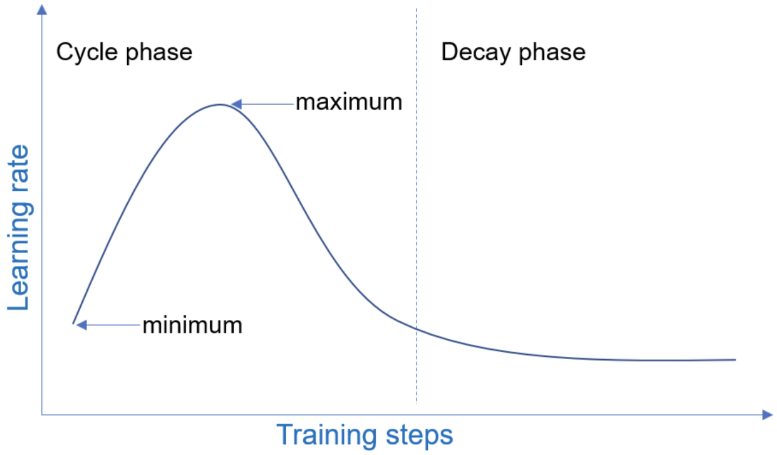


Figure 5.3: Learning Rate Schedule: Cycle and Decay Phases

With this update, the training began to perform better, showing improved accuracy and convergence. However, as training progressed, we observed signs of overfitting, as evidenced by the divergence between the training and validation losses in the results shown in the figures below. This indicated that while the model learned well on the training data, it struggled to generalize to unseen validation data. Thus, further adjustments would be required to mitigate this issue and achieve better performance.

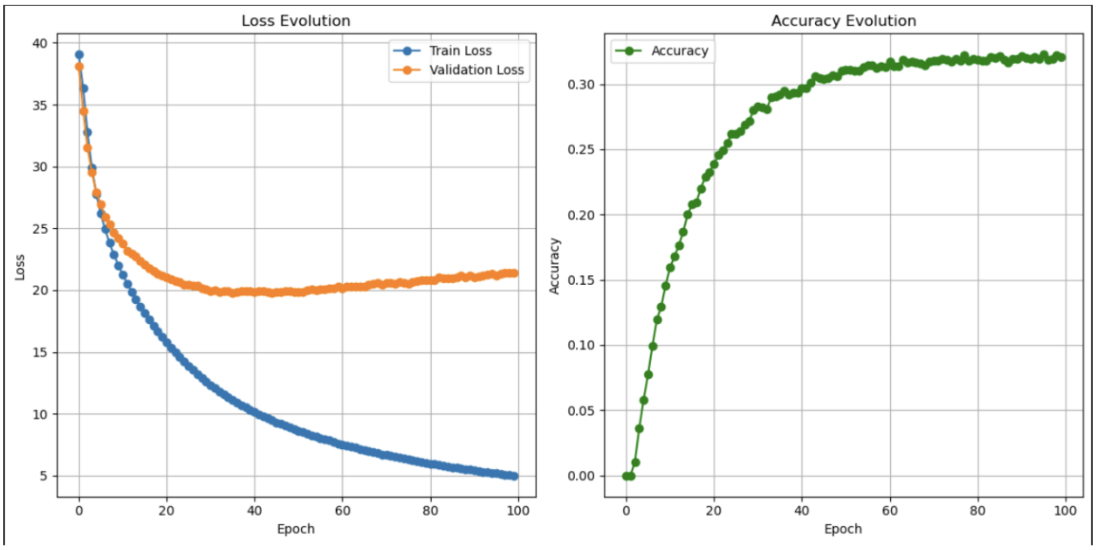


Figure 5.4: Loss and Accuracy Evolution During Training Over 100 Epochs

5.2.2 Adjustments

To address the overfitting problem observed during training, we applied data augmentation techniques to enhance the diversity of the training data and improve the model's generalization. Data augmentation involves applying various transformations to the input images, creating new variations that help the model learn more robust and invariant features. Specifically, we used two main augmentation methods: padding and flipping. Padding involves adding extra pixels around the edges of an image, effectively enlarging the image's canvas. This technique helps the model become more robust to variations in object positioning and prevents it from overfitting to tightly cropped inputs. Flipping involves horizontally flipping the images to create mirrored versions. This augmentation simulates real-world scenarios where a subject's orientation may vary, such as looking left or right. By introducing flipped images, the model learns to recognize faces regardless of their horizontal orientation. By incorporating these modifications, we effectively addressed the overfitting problem. These changes enriched the training dataset with diverse variations, enabling the model to generalize better to unseen data and reducing the gap between training and validation performance. As a result, the model achieved improved stability during training.

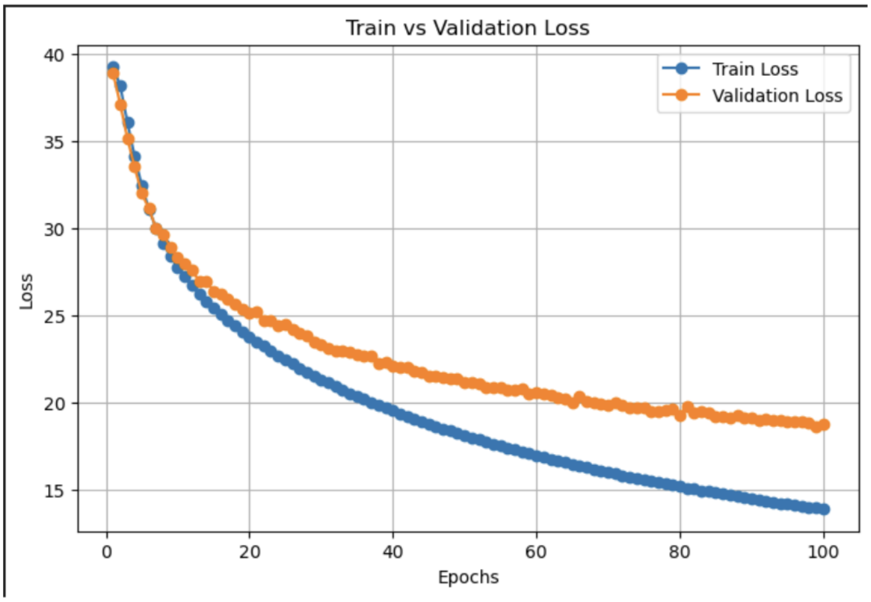


Figure 5.5: Train vs. Validation Loss Over 100 Epochs

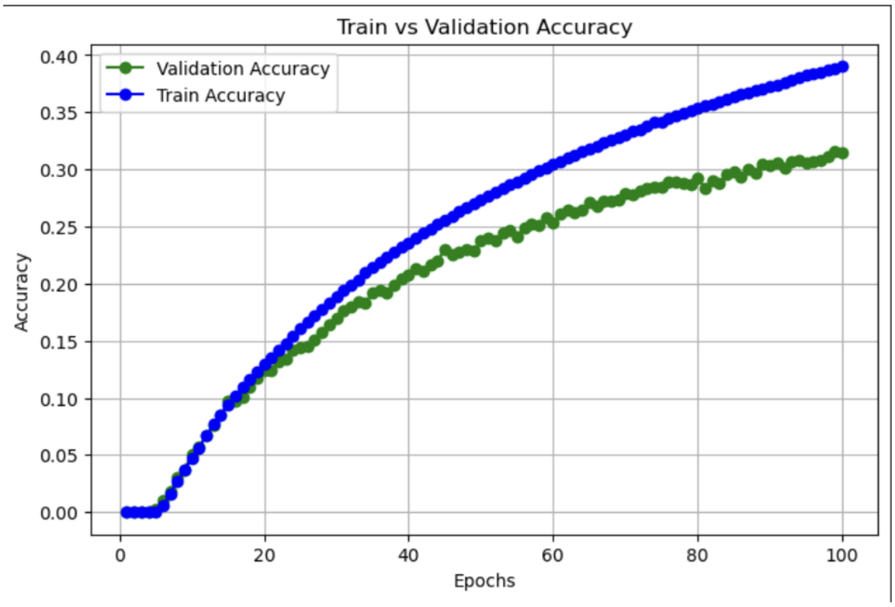


Figure 5.6: Train vs. Validation Accuracy Over 100 Epochs

The results show that while the training and validation accuracies have improved, the overall accuracy is still not optimal. We are currently working on testing other parameter changes and enhancements to further improve the model's performance and achieve better results.

5.3 Performing Face Verification task using the evaluation Pipeline

In this part, we used the LFW (Labeled Faces in the Wild) dataset for the face verification task. The ResNet18 model, pretrained separately with CrossEntropy loss and ArcFace loss, was employed to extract embeddings for the images in the LFW dataset. These embeddings were then fed into the previously implemented evaluation pipeline. The pipeline computed similarity scores between pairs of embeddings and classified whether the images belonged to the same individual based on a predefined threshold. This approach allowed us to systematically evaluate the performance of the two models on the face verification task.

Here are the results using Resnet18 pre-trained using CrossEntropyLoss:

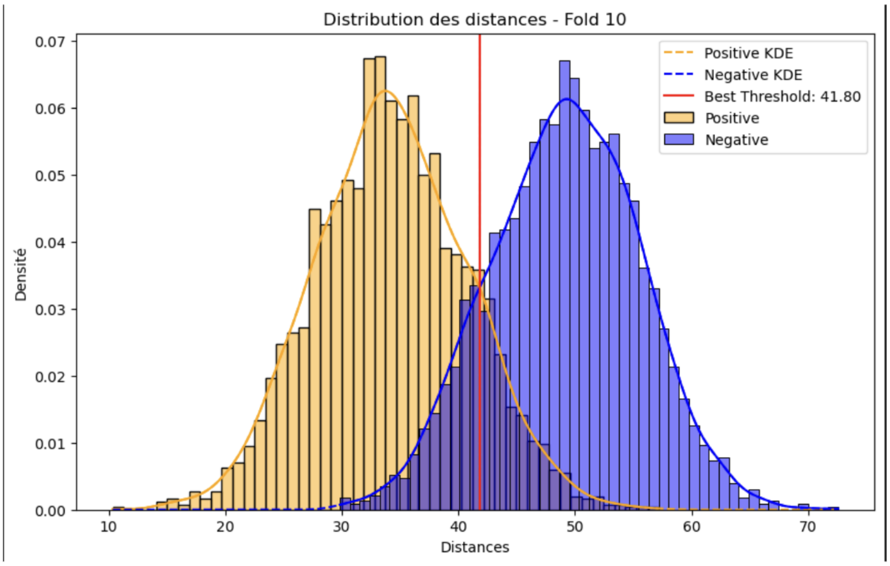


Figure 5.7: The result on the 10th fold (ResNet with CrosssEnropyLoss)

Here are the results using ArcFaceLoss:

Since the ArcFace loss is designed to improve feature representation, face verification (FV) should theoretically perform better using it. However, because we have not yet achieved optimal accuracy in the face recognition (FR) task using this loss, the embeddings extracted from the pretrained model are not yet optimal for the face verification task.

5.4 Results, Discussion, and Next Steps

5.4.1 Obtained Results

The Variational Autoencoder (VAE) training on the CASIA dataset demonstrated measurable improvements over the course of 30 epochs. The following results were recorded at the end of the training:

Fold	Seuil Optimal	Mean Accuracy
Fold 1	41.9267	
Fold 2	41.7398	
Fold 3	41.9267	
Fold 4	41.7398	
Fold 5	41.8425	
Fold 6	41.8644	
Fold 7	41.8021	
Fold 8	41.8978	
Fold 9	41.6152	
Fold 10	41.8021	
Mean	**41.8157**	**0.8665**

Table 5.1: Summary of Seuil Optimal and Mean Accuracy across folds

Fold	Seuil Optimal	Mean Accuracy
Fold 1	0.6801	
Fold 2	0.6801	
Fold 3	0.6801	
Fold 4	0.6821	
Fold 5	0.6841	
Fold 6	0.6860	
Fold 7	0.6764	
Fold 8	0.6801	
Fold 9	0.6801	
Fold 10	0.6843	
Mean	**0.6814**	**0.6148**

Table 5.2: Summary of Seuil Optimal and Mean Accuracy across folds

- **Epoch 30, Average Loss:** 1237.49

This value represents the combined loss, which is the sum of the Reconstruction Loss (Mean Squared Error) and the Kullback-Leibler Divergence (KLD). The observed loss indicates the model's ability to reconstruct input images while maintaining a regularized latent space.

5.4.2 Discussion

The training results show that the Variational Autoencoder progressively improved its performance across the 30 epochs. The average loss decreased significantly from an initial value of **2329.23** to **1237.49**, reflecting the following observations:

- **Initial Rapid Improvement:** The steep decline in loss during the early epochs indicates that the model effectively learned to capture key features of the input

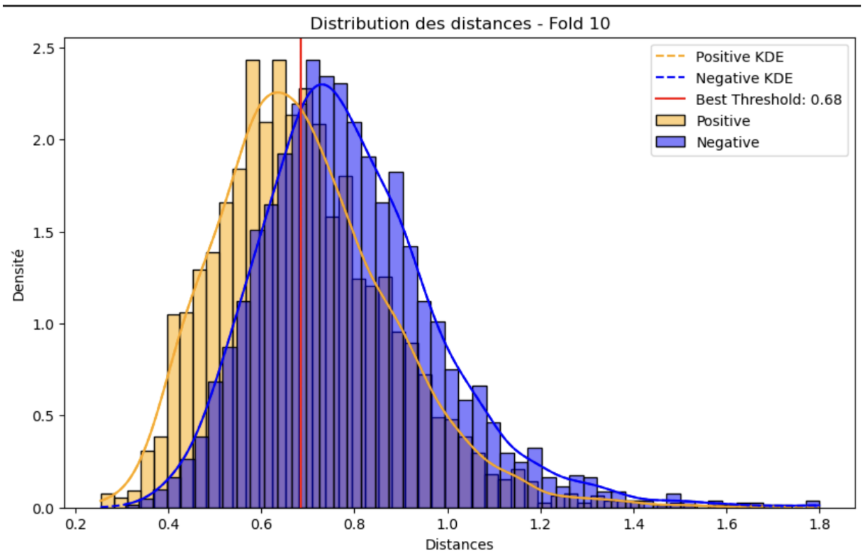


Figure 5.8: The results on the 10th fold (ResNet with ArcFaceLoss)

images. This phase highlights the effectiveness of the encoder-decoder architecture in extracting salient representations and reconstructing images.

- **Stabilization in Later Epochs:** As the training progressed, the rate of loss reduction slowed, suggesting that the model approached a point of convergence. This stabilization is typical in deep learning models and implies that the VAE had largely optimized its ability to reconstruct images and regularize the latent space.

Overall, the results affirm the VAE's ability to balance reconstruction quality and latent space regularization, achieving meaningful representations of the CASIA dataset.

5.4.3 Next Steps

To further refine the model's performance and achieve improved results, several strategies will be explored:

1. **Image Resizing:**
   - The input images will be resized to smaller dimensions (e.g., halved) to facilitate better compression through the encoder. This approach is expected to improve the efficiency of the latent space and enhance reconstruction quality by simplifying the input data.
2. **Loss Component Visualization:**
   - The Reconstruction Loss and KLD components will be visualized separately to identify which term contributes more significantly to the total loss. This analysis will guide adjustments to the weighting of these components in the loss function, ensuring an optimal balance between reconstruction fidelity and latent space regularization.

3. **Latent Space Quality Assessment:**

   - The reconstructed images will be examined to evaluate the effect of latent space size on image quality. If the reconstructed images appear excessively blurry, adjustments will be made to the latent space size to achieve clearer outputs.

4. **Parameter Tuning:**

   - Hyperparameter exploration will be conducted to identify the optimal configuration for the VAE. Parameters such as learning rate, batch size, weight decay, and latent space dimensions will be systematically adjusted to achieve better results.

By implementing these steps, we aim to further improve the VAE's reconstruction quality, enhance the latent space representation, and achieve a more robust performance on the CASIA dataset.

Appendix A

Appendix

A.1 Progress of Week 1 : October 9, 2024

Tasks completed:

1. Read the related works and understand the topic/concepts and main goals.

   Link to the papers

2. Implement the evaluation pipeline using embeddings of pairs of images obtained from training a DNN model on the LFW dataset.

   Link to the data

3. Determine the optimal threshold to distinguish between pairs belonging to the same subject (person) and those belonging to different subjects. Link to the script

Questions:

- What is the criteria to choose the optimal threshold?
- Why did we implement the evaluation pipeline as the first step?
- Can we get the best accuracy with the autoencoder?
- Why distribution of data to get the threshold is more optimal then the models we used? (p.s we trained Logistic Regression) We did get the accuracy of 0.9917 for LR and without using the model we got accuracy 0.9966

Next tasks:

Train the ResNet18 on the CASIA dataset

A.2 Progress of week 2: October 16, 2024

Tasks completed:

1. We trained Resnet18 on CASIA dataset using Cross Entropy Loss.
   Link to the script: Link to the script

2. As an option we tried to train Resnet18 on CIFAR initially to observe performance of the model (ResNet18 was pre-trained on ImageNet which is a dataset of various objects, similar to CIFAR)

3. We tried to create a CNN model from scratch to perform a face verification task using CASIA dataset Link to the script: Link to the script

Questions:

- Why do we use the ResNet18 and what's the relationship between it and the auto-encoder we will build?

- Why are we going to use the ResNet18 for a classification task and then use the embeddings for a Face Verification task, instead of directly using the ResNet model for the face verification task?

- Can we use the crossEntropy loss first and then try the Arcface loss to compare the two corresponding accuracies?

Next tasks:

- Complete the training of Resnet18 using the Arcface loss and document the runtime of each epoch.

- Try to train the Resnet18 with LFW data and apply the evaluation pipeline 3. Start to work on the Step1 presentation which is on November 6th.

A.3 Progress of week 4: October 30, 2024

Tasks completed:

1. Training Resnet18 on Casia dataset using cross entropy loss with Face identification task and presenting results (updating training settings to speed up training time)
Link to the kaggle

2. establish the presentation plan for the first step

3. Adapting the training of Resnet18 to use ArcFace Loss.
Link to the kaggle

Questions:

- How to use Arcface loss to train ResNet18?

- Trying to improve the accuracy of ResNet18 with Arcface loss

Next tasks:

- Presentation of the first step
- Building VAE from scratch

A.4 Progress of week 5-8: November 27, 2024

Tasks Completed:

1. Collaborated with the supervisor to identify and resolve issues with the `Arcface` class.
2. Reviewed Arcface training results and discussed necessary adjustments to training parameters.

Questions:

- Should we expect better results using **Arcface Loss** compared to **CrossEntropy Loss**?
- Is it essential to train the model from scratch using **Arcface Loss**?
- What is the optimal number of epochs for training?

Next Steps:

- Extract embeddings from the **ResNet18** pretrained model.
- Feed these embeddings into the evaluation pipeline for further analysis.

A.5 Progress of week 9: December 4, 2024

Tasks Completed:

1. Discussed the **ResNet** training results using **ArcFace Loss** with the supervisor.
2. Described the evaluation pipeline results using embeddings from:
   - ResNet with **Cross-Entropy Loss**.
   - ResNet with **ArcFace Loss**.
3. Initiated a discussion on the implementation of the **Variational Autoencoder (VAE)** from scratch.

Questions:

- What characteristics of convolutional layers should be considered when designing the encoder and decoder architecture for the VAE?

Next Steps:

- Focus on enhancing the training process of the **ResNet** model with **ArcFace Loss**.
- Begin developing the **Variational Autoencoder (VAE)** implementation from scratch.

A.6 Progress of the week 9: December 11, 2024

Tasks Completed

1. Addressed overfitting issues during the training of **ResNet** with **ArcFace Loss**.
2. Conducted initial refinement of the implemented **Variational Autoencoder (VAE)** model and discussed the results.

Questions:

- What is the impact of the scheduler and optimizer on training performance?
- Which data augmentation methods can be applied to improve the model's generalization?

Next Steps:

- Continue refining the training process of the **ResNet** model with **ArcFace Loss** by experimenting with various training parameters, including:
  - Scheduler.
  - Optimizer.
  - Number of epochs.
  - Revisiting the **ArcFace** class, if necessary.
- Proceed with the implementation of **Variational Autoencoders (VAEs)** by:
  - Gradually adjusting key settings.
  - Deepening the understanding of each parameter and its impact.

Bibliography

[1]

[2] Muhammed Enes Atik and Zaide Duran. Deep learning-based 3d face recognition using derived features from point cloud. Istanbul Technical University, Department of Geomatics Engineering, 4.

[3] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. Proceedings of ICML Workshop on Unsupervised and Transfer Learning, 27:37–50, 2012.

[4] Irfan Dwiki Bhaswara. Exploration of autoencoder as feature extractor for face recognition system. Faculty of EEMCS, University of Twente.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In International Conference on Machine Learning (ICML), 2020.

[6] Alhadi B. Pinkie A. Devvi S., Radifa H. P. Deep learning in image classification using residual network (resnet) variants for detection of colorectal cancer. Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Indonesia, Kampus UI Depok, 16424, Depok, Indonesia.

[7] Raja G. Dor B., Noam K. Autoencoders.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[9] Vincent Dumoulin and Francesco Visin. A guide to convolutional autoencoders. Journal of Machine Learning Research, 2018.

[10] Eyael S. E. Enoch S., Abraham W. Autoencoder based face verification system.

[11] ARA V. et al. Face detetction and recognition using hidden markov models.

[12] Jiankang Deng et al. Arcface: Additive angular margin loss for deep face recognition. JOURNAL OF LATEX CLASS FILES, 14.

[13] W. Zhao et al. Face recognition: A literature survey.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In Advances in Neural Information Processing Systems (NeurIPS), 2014.

[15] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. Multipie. Image and Vision Computing, 28(5):807–813, 2010.

[16] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In European Conference on Computer Vision (ECCV), 2016.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. Microsoft Research.

[18] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. arXiv preprint arXiv:1703.07737, 2017.

[19] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507, 2006.

[20] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Indonesia, Kampus UI Depok, 16424, Depok, Indonesia, 6.

[21] Gary B. Huang and Erik Learned-Miller. Labeled faces in the wild: Updates and new reporting procedures. University of Massachusetts, Amherst Technical Report UM-CS-2014-003.

[22] Anil K. Jain, Brendan F. Klare, and Unsang Park. Face recognition: Some challenges in forensics. IEEE International Conference on Automatic Face Gesture Recognition, pages 726–733, 2011.

[23] Dubravka Jevtić and Marijeta Slavković. Face recognition using eigenface approach*. SERBIAN JOURNAL OF ELECTRICAL ENGINEERING, 9:83:1–83:10, 2012.

[24] A. K. Kalusivalingam, R. Singh, and M. Bose. Enhancing anomaly detection in histopathological images using convolutional neural networks and variational autoencoders. European Advanced AI Journal, 2024.

[25] Ira Kemelmacher-Shlizerman, Steven M. Seitz, Daniel Miller, and Evan Brossard. The megaface benchmark: 1 million faces for recognition at scale. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[26] Diederik Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[27] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In International Conference on Learning Representations, 2014.

[28] Brendan Klare, Anil Jain, Mark Burge, Robert Bruegge, and Jeanie Paul. Face recognition performance: Role of demographic information. IEEE Transactions on Information Forensics and Security (TIFS), 7(6):1789–1801, 2012.

[29] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In Proceedings of the 32nd International Conference on Machine Learning (ICML) Workshop, 2015.

[30] Erik Learned-Miller et al. Labeled Faces in the Wild: A Survey. Advances in Face Detection and Facial Image Analysis, 2016.

[31] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In European Conference on Computer Vision, 2016.

[32] M. Liu et al. Deep learning for face recognition: A critical review. Journal of Computer Science and Technology, 2020.

[33] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015.

[34] Stylianos Moschoglou, Athanasios Papaioannou, Christos Sagonas, Jiankang Deng, Irene Kotsia, and Stefanos Zafeiriou. Agedb: The first manually collected, in-the-wild age database. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2018.

[35] Khoa Nguyen, Li Bai, and Anil Kumar. A comparative evaluation of face recognition algorithms on challenging datasets. Computer Vision and Image Understanding (CVIU), 182:1–14, 2019.

[36] Joao P. Hespanha Peter N. Belhumeur and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. EEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, 19.

[37] P. Jonathon Phillips, William T. Scruggs, Alice J. O'Toole, Patrick J. Flynn, Kevin W. Bowyer, Carl Schott, and Matthew Sharpe. Frvt 2006 and ice 2006 large-scale experimental results. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 32(5):831–846, 2010.

[38] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes, pages 448–453, 1995.

[39] Richard K. G. Kaori T. Rikiya Y., Mizuho N. Convolutional neural networks: an overview and application in radiology.

[40] Kevin L. Sasha T., Diogo A. Resnet in resnet: Generalizing residual architectures. Microsoft Research.

[41] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[42] Kihyuk Sohn. Improved representation learning with variational autoencoders using adversarial training. In Advances in Neural Information Processing Systems (NeurIPS), 2015.

[43] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Hybrid deep learning for face verification. Department of Information Engineering, The Chinese University of Hong Kong.

[44] et al. Syifa A. H. A deep learning review of resnet architecture for lung disease identification in cxr image. Department of Statistics, Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran, Bandung 45363, Indonesia.

[45] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. Proceedings of the 25th International Conference on Machine Learning (ICML), pages 1096–1103, 2008.

[46] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.

[47] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Learning face representation from scratch. arXiv preprint arXiv:1411.7923, 2014.

[48] K. Zhang et al. Autoencoder-based methods for face recognition in various conditions. Symposium on Applied Computing, 2019.

[49] Xiaogang Wang; Cha Zhang; Zhengyou Zhang. Boosted multi-task learning for face verification with applications to web image and video search.