Exploring Unsupervised Learning Approaches to Facial Recognition using Autoencoders

Muhammad Zain Ul Abiddin
Computer Science
FAST National University
Karachi, Pakistan
20K-1731

Arhum Hashmi
Computer Science
FAST National University
Karachi, Pakistan
20K-1892

Syed Muhammad Bilal
Computer Science
FAST National University
Karachi, Pakistan
20K-0209

***Abstract*—This research paper investigates the use of autoencoders in the domain of facial recognition and examines the limitations of different autoencoders, such as convolutional autoencoders and variational autoencoders, in generating embeddings based on facial features. In order to address these limitations and improve efficiency, we propose experimenting with autoencoders and unsupervised clustering algorithms to maximize scalability and accuracy. The primary objective of this research is to build upon existing approaches for extracting facial embeddings to evaluate the performance of this unsupervised clustering method for downstream tasks such as Google Photos facial clustering. The novelty is based on a complete unsupervised approach for learning and attempt to maximize performance based on this criteria. Code available at https://github.com/M1keZulu/AutoencodersNet.**

***Index Terms*—facial clustering, unlabeled data, supervised learning, unsupervised learning, K-Means, Hierarchical Agglomerative Clustering, DBSCAN, stacked autoencoders, variational autoencoder, generative models, neural networks, efficiency, computational cost, dimension reduction, clustering, autoencoders, latent space, semi-supervised learning, reconstruction loss**

I. Introduction

Facial recognition has seen remarkable growth in various domains such as security surveillance, identification and social media platforms. A crucial component of facial recognition systems is the generation of facial embeddings. These embeddings are the single most essential part of any system. If generated correctly, they can vastly improve the efficiency of any system. These facial embeddings are represented as high dimensional vectors representing distinctive facial features.

Traditionally, supervised approaches are based on the generation of these facial embeddings, yielding high accuracy. However, this approach demands a considerate amount of labeled data and costly annotations, posing challenges in data acquisition. To address these limitations, experts have delved into unsupervised and semi-supervised approaches that rely on either unlabeled or partially labeled data, making them more attractive for practical use.

In this research paper, we dive further into the exploration of unsupervised and semi-supervised learning methods for generating facial embeddings. We aim to investigate their effectiveness, capabilities and scalability to traditional supervised approaches. Through empirical evaluations and experimentations, we aim to provide insights into potential benefits of adopting alternative methodologies. Ultimately, by shifting the paradigm completely from solely relying on supervised learning, we seek to pave the way for more efficient and robust facial recognition systems, ultimately enhancing the user experience.

A. Literature Review

Autoencoders are deep learning algorithms that convert a high-dimensional data to a lower dimension. They consist of two main parts: an encoder and a decoder [1]. The encoder takes an input image and compresses it into a low-dimensional representation, while the decoder takes the low-dimensional representation and reconstructs the input image minimizing reconstruction loss. By using a bottleneck layer in the middle of the network, autoencoders are able to learn compact representations of input data. Autoencoders can therefore, in the process, learn to capture the key features of an input image, which can be used as facial embeddings for facial recognition tasks.

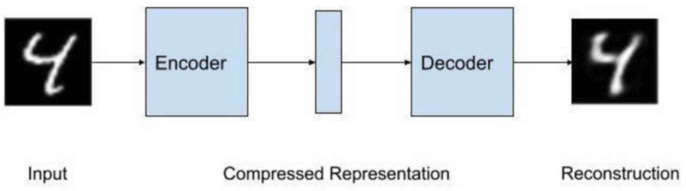

Fig. 1. A simple autoencoder [8].

Explosive growth in annotated facial datasets saw the exponential growth of facial recognition networks but has caused a rise in annotation costs for such datasets to keep up with demands [2]. With unlabeled data being more freely accessible and being largely unused, work could be done into making unlabeled facial clustering more efficient.

A Pakistani team of researchers [3] worked on using Stacked autoencoders to detect human emotions. Their main goal was to reduce the size of the output vector that resulted from the previous layers in the network. Detecting facial emotions through a neural network is quite challenging as it may result in such features being extracted that may not be relevant. To counteract that, features for extraction must be carefully selected and then clustered/classified. They proposed using

Stacked AEs to combat this problem. In Stacked autoencoders, the output of the first layer is immediately passed on as the input for the next layer. This results in a smaller and more efficient representation of an output vector at each layer reducing computational cost exponentially and increasing efficiency by extracting only key features. The team concluded, “The experiment proves that nonlinear dimension reduction using autoencoders is more effective than linear dimension reduction techniques for FER.” [3]

A Dutch team from the University of Amsterdam [4] also proposed a novel estimator of the variational lower bound. They argued that existing generative models have limitations, such as difficulty in modeling high-dimensional data. The paper proposes a new generative model, the Variational Autoencoder (VAE), which addresses these limitations. ”Variational inference” is a way to estimate the probability distribution of something that is difficult to calculate directly. This is often the case with what’s called the ”posterior distribution,” which is a type of probability distribution that represents the uncertainty of something after new information has been taken into account. The VAE combines neural networks and variational inference to learn a generative model. The encoder network outputs the mean and variance of a Gaussian distribution in the latent space. The VAE outperforms existing generative models on several benchmark datasets. The learned latent space of the VAE captures meaningful information about the data. But, the VAE can be used for semi-supervised learning by leveraging the labeled data to improve the learned representations. Yet, it provides a promising approach for unsupervised learning and in the domain of our subject, generating facial embeddings.

To determine the optimal approach for extracting features and achieve our research goal, we referred to I. D. Bhaswara’s evaluation of various autoencoders [5] including CNN, VAE, WAE, and other variants. In their approach, they performed preprocessing on the input image before feeding it to the neural network. This preprocessing was based on detecting faces within the CASIA dataset and then aligning the image according to eyes for pose regularization. The VAE model was found to have the lowest quality of reconstructed and recognized images and mixed-up latent variables. In contrast, the Resnet-WAE performs better due to its deeper network and rich features. The shallow layer models have a trade-off between reconstruction and recognition performance, and increasing latent variables does not guarantee better recognition. The generative autoencoder model based on the discriminator network helps in better generalization over latent variables and clustering data distribution identity.

B. *Problem Statement*

Facial recognition has been traditionally done with supervised learning methodologies which require a considerable amount of labeled data. In this research, we address the exploration of alternate unsupervised learning methodologies for facial recognition. While facial recognition systems have gained popularity in various fields, it is still extremely difficult to obtain all the necessary and costly data. Unsupervised learning approaches, particularly those that rely on autoencoders have the potential to overcome these challenges as they do not require labeled datasets and unlabeled data is largely unused and available.

The goal of this research is to investigate the effectiveness, capabilities, and scalability of unsupervised learning methods, specifically autoencoders, for generating facial embeddings. By generating facial embeddings, which are low-dimensional representations of facial features, we aim to enhance the performance of clustering algorithms in basic common-use applications such as Google Photos. The research aims to optimize the accuracy and speed of facial recognition systems by developing innovative techniques that overcome the limitations of existing solutions.

To achieve this goal, the research will experiment with various autoencoders, including convolutional autoencoders and variational autoencoders, and explore their performance in combination with unsupervised clustering algorithms such as K-Means and Hierarchical Agglomerative Clustering. The research will involve rigorous experimentation, evaluation, and improvements to identify the optimal approach for extracting facial features and enhancing the clustering process. The expected outcomes of this research are the development of more scalable, efficient, and robust facial recognition systems that can be applied in domains such as security, identity verification, and more. The research findings will contribute to the advancement of facial recognition technology and pave the way for the adoption of unsupervised learning approaches in this field, reducing the dependence on labeled data and improving the user experience. Overall, this research aims to address the limitations of existing facial recognition solutions by exploring unsupervised learning approaches using autoencoders, and contribute to the development of more efficient and accurate facial recognition systems with broader practical applications.

II. Autoencoders and Loss

A. ***Mean Squared Error***

Mean Squared Error is a popular loss function used in machine learning and regression problems to measure the average squared difference between the predicted and actual values. It is defined as the average of the squared differences between the predicted values and the actual values of a dataset. In an autoencoder, the MSE loss is used to measure the quality of the reconstruction produced by the decoder. The autoencoder aims to learn a compressed representation of the input data in the encoder, and then use the decoder to reconstruct the original input data from the compressed representation.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

B. ***Reconstruction Loss***

Reconstruction loss is a common loss function used in various machine learning models, especially in unsupervised learning tasks such as autoencoders [4]. The goal of the reconstruction loss is to minimize the difference between the input data and the output data generated by the model. This loss function is also known as mean squared error (MSE) or L2 loss.

Given an input data point x, and its corresponding output data point $\hat{x}$ generated by the model, the reconstruction loss can be defined as follows:

$$\mathcal{L}_{\text{rec}}(x,\hat{x}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{2}$$

where n is the number of dimensions in the input data. The reconstruction loss measures the average of the squared differences between the input and output data points.

C. *Simple Convolutional Autoencoder*

A Convolutional Neural Network Autoencoder (CNN AE) [7] is a deep learning architecture that is commonly used for unsupervised feature learning and dimensionality reduction tasks. It consists of two main components: an encoder and a decoder. The encoder is responsible for transforming the input data into a low-dimensional representation, while the decoder is responsible for reconstructing the original data from the low-dimensional representation. The network is trained using backpropagation to minimize the reconstruction error between the original input data and the reconstructed data output by the decoder. The CNN architecture is particularly suited for image data due to its ability to capture spatial relationships and hierarchical representations of visual features. In addition, the use of convolutional layers in the encoder and decoder allows for the efficient processing of high-dimensional input data while preserving spatial information. Overall, CNN autoencoders have been shown to be effective for a wide range of applications, including image denoising, anomaly detection, and feature extraction.

D. *Variational Autoencoder*

A Variational Autoencoder (VAE) [4] is a type of generative model that extends the traditional autoencoder architecture by adding probabilistic modeling to the latent representation. In a VAE, the encoder generates a mean (μ) and a standard deviation (σ) for the latent variables, which are sampled from a normal distribution. The decoder then maps these sampled variables back to the input space, producing a reconstruction. The VAE is trained by minimizing the sum of two loss terms: the reconstruction loss and the Kullback-Leibler (KL) divergence between the latent distribution and the prior distribution. The KL divergence encourages the learned distribution to be similar to the prior distribution, while the reconstruction loss encourages the decoder to generate outputs that closely match the input data. The objective function for a VAE can be expressed as follows:

$$\mathcal{L}(\theta,\phi;x) = -\mathbb{E}z \sim q\phi(z|x)[\log p_\theta(x|z)] + KL(q_\phi(z|x)||p(z)) \tag{3}$$

where $P(x|z)$ is the likelihood of the data given the latent variable, $q_\phi(z|x)$ is the approximate posterior distribution over the latent variables, $p(z)$ is the prior distribution over the latent variables, and KL divergence is used to measure the difference between $q_\phi(z|x)$ and $p(z)$. By minimizing the ELBO, the VAE learns to generate high-quality reconstructions of the input data while also learning a smooth latent representation that can be used for generative tasks, such as generating new samples from the learned distribution.

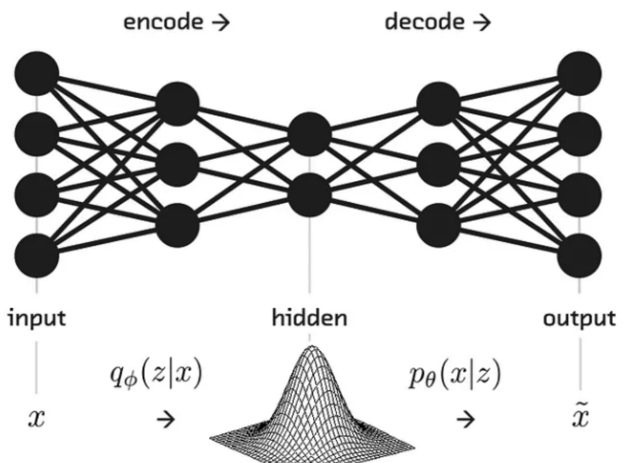

Fig. 2. VAE autoencoder [9].

E. *Adversarial Autoencoder*

An Adversarial Autoencoder (AAE) [6] is a type of generative model that combines the encoder-decoder architecture of an autoencoder with the adversarial training of a generative adversarial network (GAN). In an AAE, the encoder maps the input data to a latent space, and the decoder maps the latent representation back to the input space. Additionally, a discriminator network is trained to distinguish between the true data distribution and the distribution of latent representations produced by the encoder. The AAE is trained by optimizing the following objective function:

$$\min_G \max_D V(D,G,X) = \mathbb{E}x[\log(D(x))] + \mathbb{E}z[\log(1 - D(G(z)))] \tag{4}$$

where G is the generator network (consisting of the encoder and decoder), X is the input data, z is the latent code sampled from a prior distribution, and D is the discriminator network. The generator aims to produce a latent code that can fool the discriminator into thinking it is real data, while the discriminator aims to distinguish between the real data and the generated data. By jointly training the encoder, decoder, and discriminator networks, the AAE learns a smooth latent representation that captures the underlying structure of the data, and can be used for generative tasks, such as generating new samples from the learned distribution.

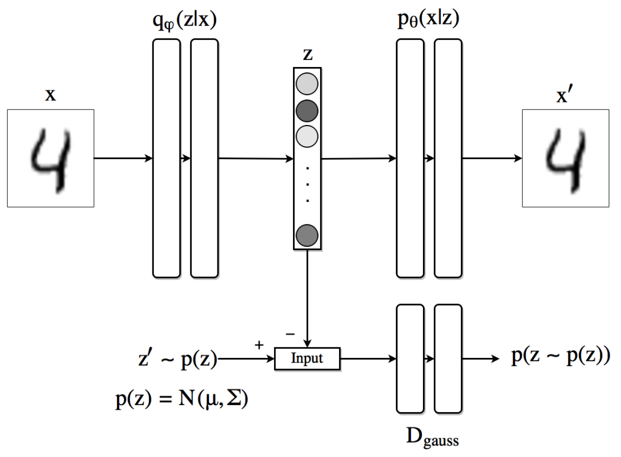

Fig. 3. AAE autoencoder [10].

III. Methodology

The CelebA and CASIA Webface datasets were used as the primary data source. We have tested Convolutional Autoencoders, Variational Autoencoders and Adversarial Autoencoders to generate facial embeddings, while unsupervised clustering algorithms such as K-Means, Spectral Clustering, Gaussian Mixture Models, and Affinity Propagation were used for clustering.

A. Pre-processing

We have applied several pre-processing steps to our datasets for better extraction of facial embeddings and clustering.

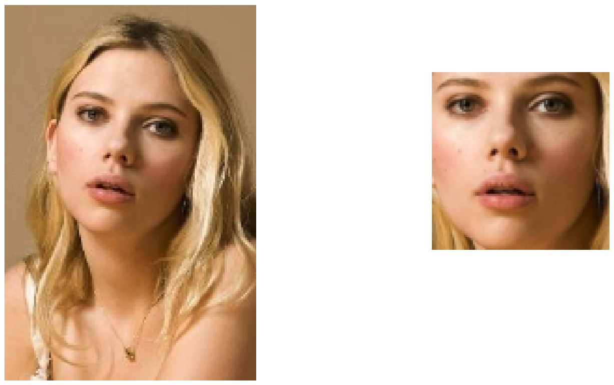

Fig. 4. Example of realignment according to eye level and size 64x64.

- Image Realignment: To ensure consistent orientation and alignment of images for better and efficient analysis, a pre-processing step is performed to align the facial images. This helps minimize the variations due to different head alignment and poses.
- Normalizing Image Pixel Values: In order to ensure consistency and comparability across images, the pixel values of the image are normalized. This process involves scaling the pixel to a specific range.
- Facial Image Extraction through Cropping: To focus solely on the facial region and eliminate background noise or irrelevant features, a cropping technique is applied to extract only the facial area from the original images. This step helps in isolating the facial features and enhancing the accuracy of subsequent analysis.
- Data-Splitting: In order to evaluate the performance of the methodologies and avoid overfitting, the dataset is divided into training and testing subsets using a data-splitting ratio of 80:20. The training set is used for model training and parameter optimization, while the testing set is reserved for evaluating the performance and generalization capability of the trained models.

B. Process

1) Architecture: The encoder part of the network takes an input image of size 64x64 with 3 color channels (RGB), and applies a series of convolutional layers with increasing number of filters to extract high-level features from the input image. The first convolutional layer has 128 filters with a kernel size of 4x4 and a stride of 2, which reduces the spatial dimensions of the feature map by half while increasing the number of filters. Batch normalization is applied after each convolutional layer to improve the stability and convergence of the network, and ReLU activation function is used to introduce non-linearity in the feature map.

Layer (type)	Output Shape	Params
InputLayer	(None, 64, 64, 3)	0
Conv2D	(None, 32, 32, 128)	6144
BatchNormalization	(None, 32, 32, 128)	512
Activation	(None, 32, 32, 128)	0
Conv2D	(None, 16, 16, 256)	524288
BatchNormalization	(None, 16, 16, 256)	1024
Activation	(None, 16, 16, 256)	0
Conv2D	(None, 8, 8, 512)	2097152
BatchNormalization	(None, 8, 8, 512)	2048
Activation	(None, 8, 8, 512)	0
Conv2D	(None, 4, 4, 1024)	8388608
BatchNormalization	(None, 4, 4, 1024)	4096
Activation	(None, 4, 4, 1024)	0
Flatten	(None, 16384)	0
Dense	(None, zdim)	1638400
Total params		12,863,072
Trainable params		12,859,136
Non-trainable params		3,936

TABLE I
Encoder Architecture.

The output of the last convolutional layer is flattened into a 1D vector and fed into a fully connected dense layer with zdim units (in this case, zdim is set to 128), which represents the compressed latent space representation of the input image.

The decoder part of the network takes the latent space representation as input and applies a series of transpose convolutional layers with decreasing number of filters to reconstruct the output image from the compressed representation. The first layer of the decoder is a fully connected dense layer that maps the compressed representation to a feature map of size 8x8

Layer (type)	Output Shape	Params
InputLayer	(None, zdim)	0
Dense	(None, 65536)	16779264
Reshape	(None, 8, 8, 1024)	0
Conv2DTranspose	(None, 16, 16, 512)	8388608
BatchNormalization	(None, 16, 16, 512)	2048
Activation	(None, 16, 16, 512)	0
Conv2DTranspose	(None, 32, 32, 256)	2097152
BatchNormalization	(None, 32, 32, 256)	1024
Activation	(None, 32, 32, 256)	0
Conv2DTranspose	(None, 64, 64, 128)	524288
BatchNormalization	(None, 64, 64, 128)	512
Activation	(None, 64, 64, 128)	0
Conv2DTranspose	(None, 64, 64, 3)	387
Total params		27,690,051
Trainable params		27,687,491
Non-trainable params		2,560

TABLE II
DECODER ARCHITECTURE.

Layer (type)	Output Shape	Params
InputLayer	(None, zdim)	0
Dense	(None, 512)	262656
Dense	(None, 512)	262656
Dense	(None, 512)	262656
Dense	(None, 512)	262656
Dense	(None, 1)	513
Total params		1,051,137
Trainable params		1,051,137
Non-trainable params		0

TABLE III
DISCRIMINATOR ARCHITECTURE.

with 1024 filters. The feature map is then reshaped into a 3D tensor and passed through a series of transpose convolutional layers with decreasing number of filters and increasing spatial dimensions until the final output image of size 64x64 with 3 color channels is produced. The last convolutional layer uses a kernel size of 1x1 and stride of 1, and does not apply any activation function to ensure the output values are within the valid color range.

The model is trained using the mean squared error (MSE) loss function, and the optimizer used is the Adam optimizer. In VAE, reconstruction loss is used as an alternative. The model is trained for one epoch on a training set with 455595 images, using a batch size of 512. A validation set is also provided for evaluation during training. Checkpoint and early stop callbacks are used to save the model weights and stop training early if the validation loss does not improve. Standard learning rate of 0.001 was used while training all autoencoders,

Similar architecture is used for the variational autoencoder with no batch normalization due to gradient explosion while the same network was used for adversarial autoencoder with an additional discriminator network.

2) Evaluation: We evaluated the performance of the clustering using two methods: t-SNE plots and ROC curves. Each model was trained for 5 iterations due to a lack of resources to support a higher number. Reconstruction results showed that simple CNN autoencoder performed the best reconstruction while VAE and AAE were a little behind with AAE performing the worst and taking the most resources to train because of its Generative nature.

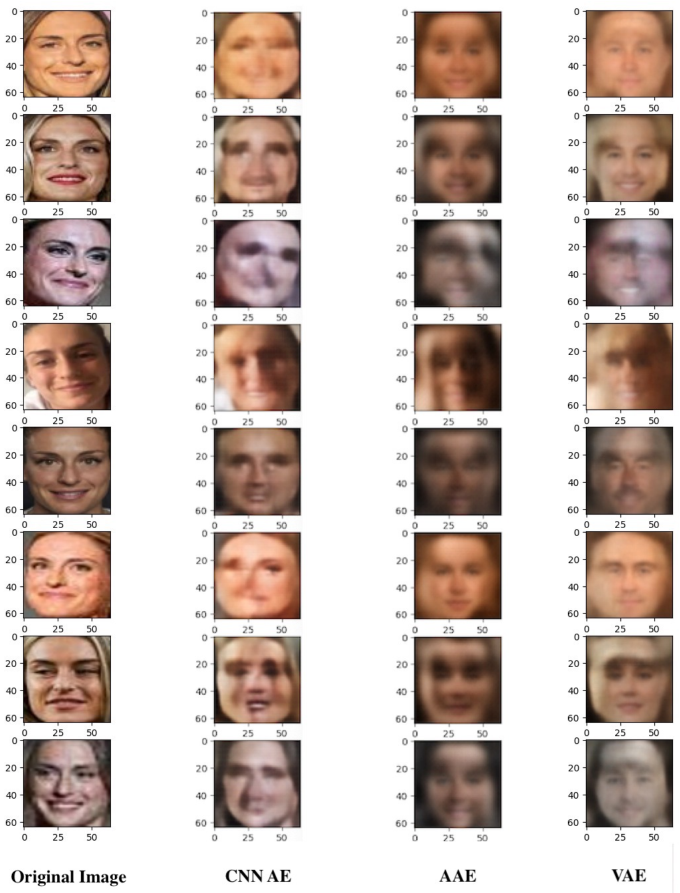

Fig. 5. Reconstruction by different autoencoders with zdim=128.

t-SNE (t-Distributed Stochastic Neighbor Embedding) is a machine learning technique used for visualizing high-dimensional data in a lower-dimensional space, typically 2D or 3D. It is particularly useful for visualizing complex datasets such as facial embeddings generated by autoencoders.

ROC (Receiver Operating Characteristic) curve is a commonly used evaluation metric for binary classification problems. It is a graphical representation of the trade-off between the true positive rate (TPR) and the false positive rate (FPR) for different threshold values. In the context of clustering algorithms, ROC curves are used to evaluate the performance of the algorithm in assigning data points to correct clusters. Since clustering is an unsupervised learning task, there are no labels available to compute metrics like precision, recall or accuracy. Therefore, ROC curves can be used to provide a measure of the clustering performance.

To demonstrate the practical application and determine the effectiveness of the autoencoder and clustering methods, we created an application that allows users to cluster their own facial photos. The application takes a user's photo as input and generates an embedding using the pre-trained autoencoder. The embedding is then assigned to a cluster using the K-means algorithm. We used the Python Flask framework to create the web application, which can be accessed through a web browser.

IV. Findings and Discussion

We computed the true positive rate (TPR) and false positive rate (FPR) for each cluster, and plotted these values on the ROC curve. The area under the ROC curve (AUC) was then used as a measure of the overall performance of the clustering algorithm. Simple distance metric was used. We also generated facial embeddings using three different autoencoder models - CNN, VAE, and AAE - for a dataset of 10 images of 5 different people, resulting in a total of 50 embeddings. We then applied t-SNE to these embeddings to generate a 2-dimensional plot that visualizes the similarity between the embeddings.

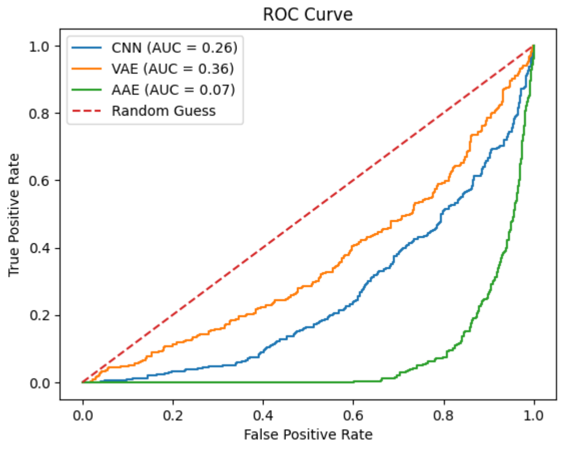

Fig. 6. ROC zdim=128.

Our results showed that the CNN autoencoder produced facial embeddings that were well-separated and easily distinguishable from each other, resulting in ROC curves with high AUC values for all clustering algorithms. The VAE model also showed good performance, seemingly much better performance than CNN autoencoder, with AUC values that were slightly higher than the CNN for most clustering algorithms. The AAE model showed the poorest performance of the three, with AUC values that were lower than the other two models for all clustering algorithms.

The ROC curve for VAE has a higher AUC value than that of CNN for most clustering algorithms. This suggests that VAE performed better in terms of separating positive and negative samples.

There could be several reasons why this is the case. For example, VAE may have captured more relevant information about the data distribution than CNN, which could have improved its clustering performance. Additionally, the way VAE models the latent space may have resulted in a more separable embedding space, even though the clusters are not as well-defined as in CNN. It is also possible that the difference in performance between the two models could be specific to the dataset used in the study.

Similarly, I.D. Bhaswara reported in his paper [5] that adversarial autoencoders performed better than CAE and VAE because of their generative nature. We were not able to replicate the results and the reason for that was probably training with less iterations since Generative networks usually require high number of epochs to give better results.

Therefore, it is important to note that the performance of different models can depend on various factors, including the nature of the data and the specific clustering algorithm used. It is recommended to test multiple models and algorithms to determine the best approach for a given task.

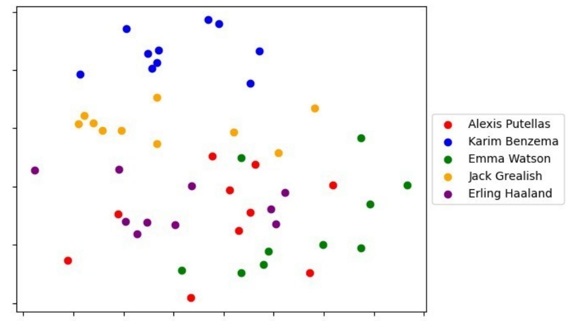

Fig. 7. CNN AE t-SNE zdim=128.

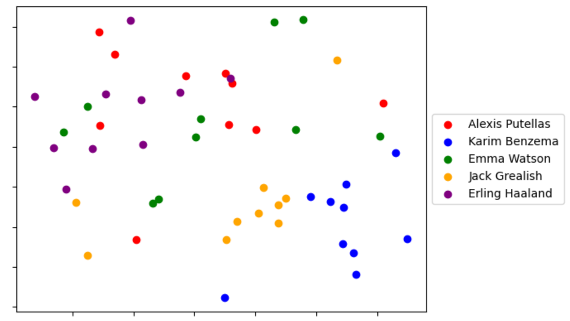

Fig. 8. VAE t-SNE zdim=128.

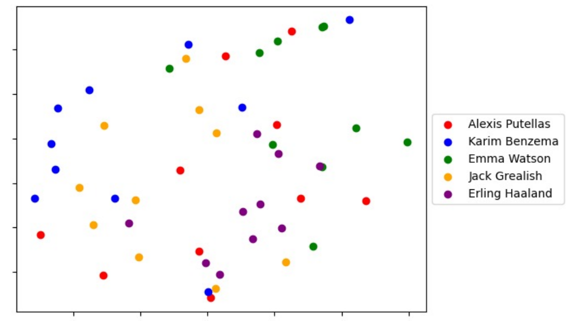

Fig. 9. AAE t-SNE zdim=128.

Our t-SNE plots confirmed that the CNN autoencoder produced facial embeddings that were well-separated and easily distinguishable from each other, resulting in clearly defined clusters. The VAE model also produced reasonably well-defined clusters, although with more overlap than the CNN. The AAE model produced clusters that were more difficult to distinguish and had more overlap than the other two models.

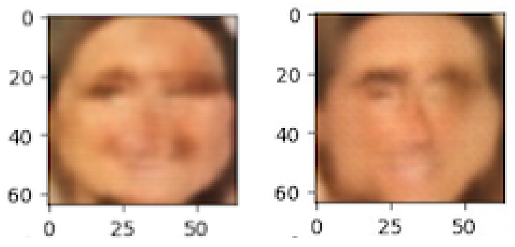

Fig. 10. Reconstruction difference between 128 and 64 latent dimension size of CNN Autoencoder.

Increasing the latent dimension size had a positive effect since there were larger number of features captured but it is not a general observation since sometimes a larger dimension can cause irrelevant features to be captured which causes a decrease in clustering performance. The reconstruction difference between different dimensions can be observed in figure 10.

In addition to evaluating the performance of different autoencoder models for facial embedding generation and clustering, we also tested several clustering algorithms to determine their effectiveness in this context. Specifically, we used K-Means, affinity propagation, spectral clustering, and Gaussian mixture model (GMM) to cluster the facial embeddings generated by the autoencoder models.

Our results showed that the performance of the clustering algorithms varied depending on the autoencoder model used. Interestingly, we found that affinity propagation was particularly well-suited for downstream tasks, such as clustering user images because it does not require the number of clusters to be determined beforehand. This is in contrast to other clustering algorithms, such as K-means and GMM, which require the use of the elbow method to determine the appropriate number of clusters. The elbow method involves plotting the within-cluster sum of squares as a function of the number of clusters and selecting the number of clusters at the ”elbow” of the curve, where the rate of improvement starts to level off. The equation for elbow method is:

$$J(k) = \sum_{i=1}^{n} \|x_i - \mu_{k_i}\|^2 = \sum_{k=1}^{K} \sum_{i=1}^{n} w_{i,k} \|x_i - \mu_k\|^2 \tag{5}$$

The findings suggest that the choice of clustering algorithm can have a significant impact on the performance of facial embedding generation and clustering. Affinity propagation may be a particularly useful algorithm for downstream tasks, such as clustering user photos, where the number of clusters may not be known in advance. A good clustering algorithm might cover up for the poor performance by models.

V. Conclusion

Based on the results of the evaluation, it can be concluded that the performance of the three autoencoder models, namely CNN, VAE, and AAE, in generating facial embeddings and clustering can be ranked in the order CNN similar to VAE but AAE performing poorly. This was determined through the use of two evaluation metrics, ROC and t-SNE.

The CNN autoencoder exhibited the best performance in both metrics, indicating that it was able to produce facial embeddings that were well-separated and easily distinguishable from each other. The VAE model also showed good performance, although not quite as good as the CNN, with clusters that were still reasonably well-defined. The AAE model, on the other hand, showed the poorest performance of the three, with clusters that were more difficult to distinguish and had more overlap.

VI. Recommendations

These findings suggest that, when using autoencoders for facial embedding generation and clustering, the choice of model can have a significant impact on performance. In particular, the use of a CNN model may be more effective than VAE or AAE models for this task. However, it should be noted that these results, including the choice if clustering algorithm, may not be generalizable to other datasets or applications, and further research is needed to confirm these findings and explore other factors that may influence performance.

VII. Limitations and Future Work

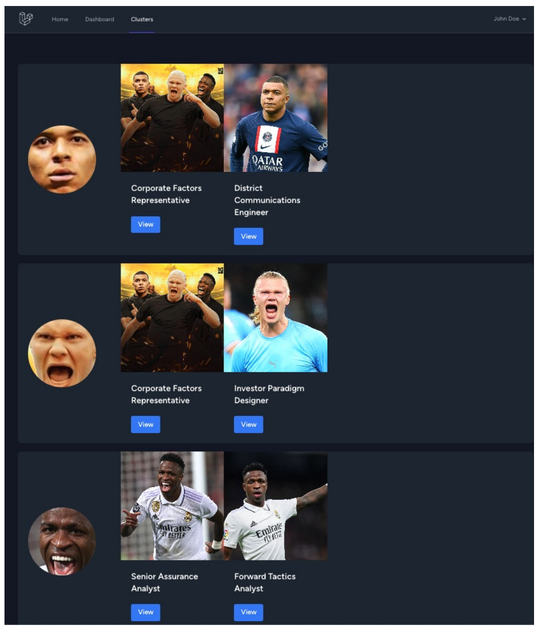
Home
Dashboard
Clusters
John Doe
Corporate Factors Representative
District Communications Engineer
View
View
Corporate Factors Representative
Investor Paradigm Designer
View
View
Senior Assurance Analyst
Forward Tactics Analyst
View
View

Fig. 11. Application created on Laravel and Flask stack.

We faced the challenge of limited resources for training the autoencoder model, which resulted in a suboptimal number of training epochs. The performance of the facial recognition system is highly dependent on the quality of the images used for training. Poor quality images, with low resolution or inconsistent lighting, can result in inaccurate embeddings and clustering results.

It is difficult to evaluate cluster performance and better evaluation metrics like Similarity Index and Likelihood Ratio Classifier can be used to get accurate insights into the research. The current metrics are reasonable but offer little insight into how the model can be improved.

Furthermore, the lack of exploration of other types of autoencoder models may have limited the performance of the system. The size and diversity of the dataset used for training and testing the model can impact the performance of the system. Unsupervised learning is highly dependent on the right balance of dataset, adding more data might improve performance. As far as the downstream task is concerned, this approach at this stage is not suitable for production and gives subpar results when deployed to an application. The application can be seen in figure 11.

REFERENCES

[1] A. Dertat, ”Applied Deep Learning Part 3: Autoencoders,” Towards Data Science, Oct. 2017. Available: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798. [Accessed: Apr. 17, 2023].

[2] L. Yang, D. Chen, X. Zhan, R. Zhao, C. C. Loy, and D. Lin, ”Learning to Cluster Faces via Confidence and Connectivity Estimation,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 13366-13375. doi: 10.1109/CVPR42600.2020.01338.

[3] M. Usman, S. Latif, and J. Qadir, ”Using deep autoencoders for facial expression recognition,” in 2017 13th International Conference on Emerging Technologies (ICET), Islamabad, Pakistan, 2017, pp. 1-6. doi: 10.1109/ICET.2017.8281753.

[4] D. P. Kingma and M. Welling, ”AutoEncoding Variational Bayes,” CoRR, vol. abs/1312.6114, 2013.

[5] I. D. Bhaswara, ”Exploration of autoencoder as feature extractor for face recognition system,” August 2020. Available: http://essay.utwente.nl/83138/. [Accessed: Apr. 17, 2023].

[6] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, “Adversarial Autoencoders,” arXiv:1511.05644, cs.LG, 2016.

[7] D. Jana, J. Patil, S. Herkal, S. Nagarajaiah, and L. Duenas-Osorio, “CNN and Convolutional Autoencoder (CAE) based real-time sensor fault detection, localization, and correction,” Mechanical Systems and Signal Processing, vol. 169, 108723, 2022. https://doi.org/10.1016/j.ymssp.2021.108723.

[8] A. Singh and T. Ogunfunmi, “An Overview of Variational Autoencoders for Source Separation, Finance, and Bio-Signal Applications,” Entropy, vol. 24, no. 1, pp. 55, Dec. 2021, doi: 10.3390/e24010055.

[9] ”What The Heck Are VAE-GANs, *Towards Data Science*,” Aug. 2018. [Online]. Available: https://towardsdatascience.com/what-the-heck-are-vae-gans-17b86023588a. [Accessed: May 14, 2023].

[10] ”Adversarial Autoencoders (with Pytorch), *Paperspace*,” 2017. [Online]. Available: https://blog.paperspace.com/adversarial-autoencoders-with-pytorch. [Accessed: May 14, 2023].