Face Recognition With Siamese Networks and Triplet Loss

Berke Bayraktar[1] Sare Naz Ersoy[1]

Abstract

In today's world, task of face verification and recognition holds a very important space in our day to day lives. Each of us use or are subjected to some type of face verification in our daily lives. Whether as an authentication method for our personal technological products such as smartphones or computers or as an industrial scale face recognition systems that are used for employee authentication at big companies and warehouses. Due to it's high relevancy to our lives, face recognition and verification has been and continues to be a very popular field of research. One of the earlier and influential method for face recognition before neural networks is a method called eigenfaces[1] and was developed in 1991. Later with popularity of neural networks, the field has shifted from using manually extracted features to using facial features that were extracted by neural networks.

1. Introduction

In the domain of face recognition, it is important to understand the difference and relation between a face verification system and face recognition system. Face verification and face recognition are both components of facial biometric technology, but they serve distinct functions and are used in different scenarios. Face verification is the process that involves comparing a presented facial image with a single stored facial image (or facial template) to confirm whether they both belong to the same individual. The aim is to verify if the person is who they claim to be. For example, a smartphone may use face verification to unlock the device, verifying that the user presenting their face to the camera is the same user whose face image is stored in the device's memory. Face recognition is a more complex process that involves comparing a presented facial image against multiple stored facial images in a database to find a match. The goal is to determine the identity of the person, not merely to verify a claim of identity. For example, law enforcement might use face recognition technology to identify a suspect captured in surveillance footage by comparing the image against a database of known individuals. Similarly the earlier given example of employee authentication systems might use face recognition depending on how they are implemented and are expected to work. In essence, face verification is a one-to-one matching process, whereas face recognition is a one-to-many matching process. The distinction between the two lies in the task complexity and the range of possible matches: verification decides whether a person is who they say they are, while recognition identifies who a person is among a broader population.

Ever since the earlier days of image recognition, face recognition has been a popular field for researches. Starting with the previously mentioned, influential method called eigenfaces[1] which is a face recognition technique that uses principal component analysis (PCA) to extract the most important facial features then represents each face as a linear combination of eigenfaces, which are the eigenvectors of the covariance matrix calculated from a training set of face images. During verification, the input face is projected onto the eigenface subspace, and its similarity to known faces is measured based on the Euclidean distance or cosine similarity in the reduced-dimensional space. This approach was popular at the time but with the developments in deep neural network domain and especially with emergence of convolutional neural networks which are very good at extracting intrinsic features from images automatically, they have replaced the aforementioned earlier approaches. Specifically with the introduction of works like DeepFace[2] in 2014 and FaceNet[3] in 2015. Both of these works have introduced novel approaches to the task of face recognition which we will go over in detail in the following sections.

In this work, for the dataset, we are using a famous face recognition dataset called Labelled Faces in Wild (LFW)[4] This dataset contains images of various celebrities around the world. The dataset contains 13,233 face images accross 5749 people (labels). This same dataset is also used on the previously mentioned works DeepFace and FaceNet alongside other datasets.

2. Related Work

Advancements in the field of face recognition have been fueled by an increasing demand for technologies that can accurately identify and verify individuals. Recent years have seen an influx of studies leveraging deep learning techniques and innovative loss functions to improve the performance and reliability of face recognition systems. This section delves into two significant contributions that have shaped the current research on face recognition with Siamese networks and triplet loss: the DeepFace and FaceNet models.

In 2014, Facebook's AI research team published a paper detailing a significant breakthrough in face recognition, known as the DeepFace model. The principal strength of DeepFace lies in its ability to reduce the inherent variance in facial appearances due to extraneous factors such as viewing angle, illumination, and facial expressions. DeepFace achieves this by first identifying faces in images and then applying a 3D alignment transformation to render them in a consistent, frontal pose.

The model utilizes a nine-layer deep neural network to learn a compact representation of faces. The first four layers consist of locally connected layers, each followed by a max-pooling layer, which are more flexible than standard convolutional layers, allowing the network to model complex and highly variable facial features. This is followed by two fully connected layers and a softmax classifier. The DeepFace system maps each face image into a 4096-dimensional Euclidean space, where distances directly correspond to face similarity. This sophisticated architecture, coupled with an extensive labeled dataset, allows DeepFace to achieve near-human accuracy on standard benchmarks such as LFW (Labeled Faces in the Wild), marking a significant leap forward for facial recognition models.

Another pioneering work in the field is the FaceNet model, published by Google's research team in 2015. Unlike DeepFace and previous models that used pairs of images for training, FaceNet introduced a new methodology: the triplet loss function. FaceNet employs a deep convolutional network that transforms an input image into a compact 128-dimensional vector. The vector, also known as an embedding, represents the learned features of the face.

The triplet loss function is central to FaceNet's approach. Each training iteration uses three images: an anchor image, a positive image (same identity as the anchor), and a negative image (different identity than the anchor). The aim of the training process is to minimize the distance between the anchor and positive images (intra-class distance) while maximizing the distance between the anchor and the negative images (inter-class distance) in the embedded space. This results in an embedding space where the same identities are located closely together while different identities are further apart.

Moreover, to prevent the network from collapsing all representations to zero vectors, a margin is introduced in the loss function. This margin defines a 'radius' around each embedding within which all same-identity faces should reside, thereby providing a structured approach to learning a similarity measure for faces. The FaceNet model achieved state-of-the-art results on multiple benchmark datasets including LFW and YouTube Faces DB, cementing its role as a significant milestone in face recognition research.

These seminal works, DeepFace and FaceNet, have laid the groundwork for contemporary research into face recognition, ushering in an era of deep learning methodologies and advanced loss functions. The current study uses the same approach as the FaceNet model by utilizing Siamese Networks and Triplet Loss for training of the model.

3. Implementation

As stated before, in this work we are utilizing a Siamese Network approach with triplet loss. Before diving into an explanation in greater detail of our approach, we will discuss the concepts of Siamese Networks and trriplet loss.

3.1 Siamese Networks and Triplet Loss

The core premise behind a siamese network is that it leverages the power of learning feature representations by comparing pairs of inputs. Named after the biological term "Siamese Twins" due to its characteristic architecture, the siamese network processes two distinct inputs through shared weights and computes their similarity or difference.

The network consists of two identical subnetworks (or "twins"), each receiving a distinct input, and the aim is to encode the inputs into a representation in such a way that similar inputs have similar representations. Therefore, the term "siamese" comes from the fact that there is a symmetry in how the network treats both inputs and that they share

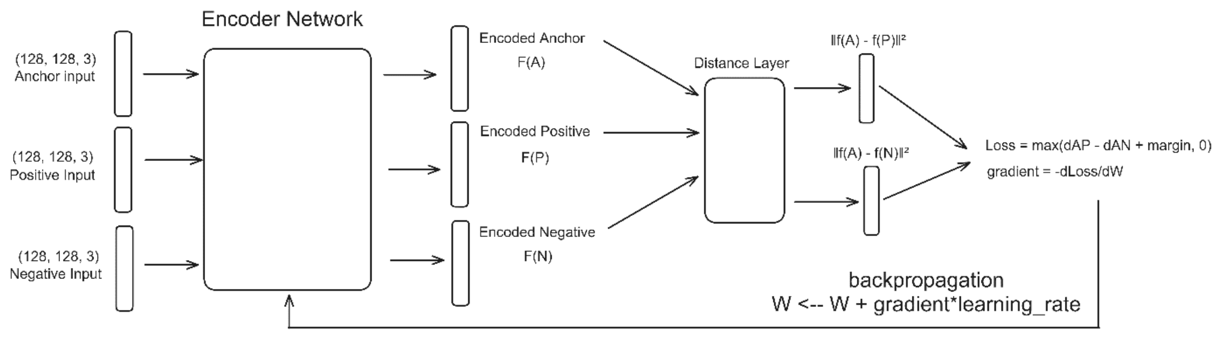


Figure 1 Illustration of Methodology

the same parameters or weights. This unique architecture of siamese networks makes them well-suited for various tasks that require learning from the similarity between pairs, such as signature verification, image retrieval, and particularly face recognition.

A fundamental enhancement to the siamese architecture is the use of triplet loss. Instead of relying on pairs of inputs, the triplet loss considers three inputs at a time: an anchor, a positive input of the same class as the anchor, and a negative input from a different class. The goal of the triplet loss is to learn representations such that the distance between the anchor and the positive input in the embedding space is less than the distance between the anchor and the negative input by a certain margin. This helps the model learn more robust and generalizable representations.

Given an anchor sample (x_a, y_a) and another positive sample (x_p, y_p) such that $y_a = y_p$ and with another negative example (x_n, y_n) such that $y_a \neq y_n$ the triples loss function is defined as:

$$L_{triplet} = max\left(0, d_p^2 - d_n^2 + \alpha\right)$$

$$\text{where } d_p^2 = D\left(f_\theta(x_a), f_\theta(x_p)\right) \text{ and is } l_2 \text{ norm}$$

$$\text{where } d_n^2 = D\left(f_\theta(x_a), f_\theta(x_n)\right) \text{ and is } l_2 \text{ norm}$$

The margin, represented by the alpha term, specifies the minimum desired difference between these distances. It enforces that the distance between the anchor and the positive sample should be smaller than the distance between the anchor and the negative sample by at least the margin value. In other words, it ensures that the positive sample is closer to the anchor than the negative sample by a certain threshold. By incorporating the margin in the triplet loss, the model learns to maximize the discrimination between positive and negative samples. It encourages the model to push positive samples closer to the anchor while simultaneously pushing negative samples farther away, increasing the overall separation between classes in the embedding space. Setting an appropriate margin value is crucial because a too small value may lead to the loss being easily satisfied without significant separation, while a too large value can make the optimization problem more difficult and lead to convergence issues.

3.2 Methodology

Next, we construct an encoder model upon this base model, which includes several additional layers. A GlobalMaxPooling2D layer is used to reduce the spatial dimensions of the output from the InceptionResNetV2. A Dropout layer then randomly sets a fraction of input units to 0 at each update during training time, which helps prevent overfitting. Then, a Dense layer, without biases, is employed for the transformation of features, followed by Batch Normalization to accelerate the training process. Lastly, we employ a Lambda layer to apply L2 normalization to the output, ensuring that all output vectors have a magnitude (or length) of one. This encoder structure is summarized in figure 2. Our siamese network comprises three encoders, each for the anchor, positive, and negative image. Each image is independently passed through an encoder, and the embeddings thus generated are subsequently sent to a custom DistanceLayer. This layer computes the squared Euclidean distance between the embeddings of the anchor-positive pair and the anchor-negative pair, serving as a measurement of similarity or dissimilarity. Visual representation of this model can be observed in figure 1. This model utilizes the triplet loss function, aiming to ensure that the

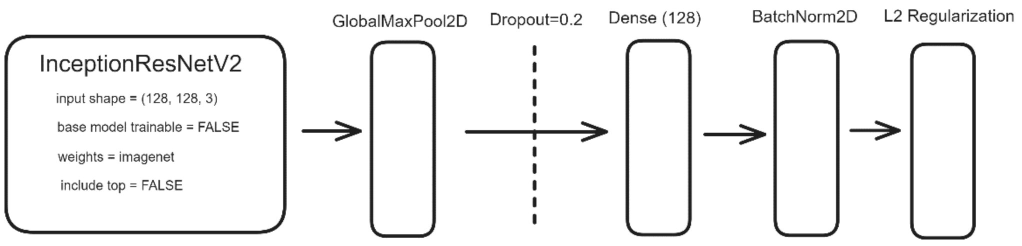


Figure 2 Encoder Network In Detail

positive image is closer to the anchor than the negative image by a pre-defined margin in the embedding space. During training, the model automatically updates the gradients of the weights to minimize this loss.

Construction of these mentioned triplets are another thing to consider which is called “Triplet Mining”. There are various ways to construct triplets from the dataset for training. In this work we use 2 mining strategies called Random Triplet Mining, Semi-Hard Triplet Mininig and Hard Triplet Mining. In random triplet mining triplets are formed by random selection of images from the dataset. More specifically, to form a random triplet, first a random anchor image is selected, later a postive image which belongs to same class (person) as the anchor image selected randomly from all the images of this said person. Finally, negative image is selected randomly from all the images except those that belong to the anchor person. While this method is simple and can work reasonably well, it has a major drawback: most of the randomly selected triplets are easy triplets where the current model already gives a correct answer (i.e., the anchor is already closer to the positive than the negative). Therefore, such triplets don't contribute much to the improvement of the model. To overcome the limitation of random triplet mining, hard triplet mining can be used. This strategy involves selecting the hard triplets, i.e., triplets where the negative instance is almost as close to the anchor as the positive instance. By focusing on these hard triplets, the model can learn more effectively and often achieve better performance. It's a form of focusing on the most challenging examples, similar to the concept of hard negative mining in other contexts.

4 Dataset

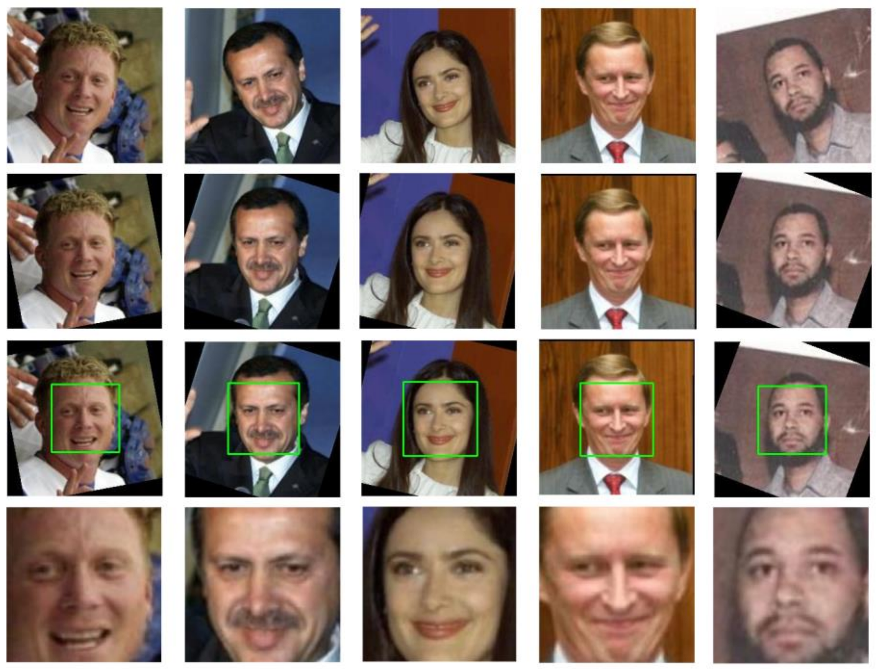
There are various datasets that are proposed for the task of face recognition as listed in below figure. However as it can be seen many of these datasets are way too big for us to use due to time and computational resource limitations. Therefore we have selected to use the Labelled Faces in the Wild (LFW) dataset which had managable amount of data.

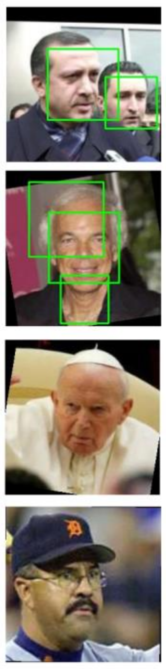
Dataset	# Person	# Images
LFW [4]	5,749	13,000
VGGFace [5]	2,622	2.6M
ms-celeb-m1 [6]	100,000	10M
umdfaces [7]	8,277	367,888

4.1 Preprocessing

Providers of the LFW faces dataset provide various versions of this dataset with different preprocessing techniques such as face-alignment preapplied to images. We are specifically using the “lfw-deepfunneled” dataset which includes faces that are preprocessed with 2d face alignment using deep neural networks. After obtaining the aligned faces, we run a face detection algorithm on these aligned faces using the Haar Cascade [8] method. From this method (x,y,w,h) values which specify location of the face in the image are returned and later these values are used for cropping the relevant area of the image. If we consider first 5 columns from figure 3, first row of images show the data from LFW dataset where as second row of images are from LFW-Deepfunneled dataset where 2d face alignment is applied. In third row we see faces that are detected by the Haar Cascade face detection algorithm and in last row we see region of the images that we are interested in being cropped and resized into uniform sizes since in each image face region is not necessarily the same size or aspect ratio.

Finally, as it can be seen from rightmost column on figure 3 there are some edge cases that need to be considered. On some cases, there are faces on the background belonging to some other person which the face detection algorithm picks up. And in other scenerios, same face can be detected multiple times or areas that do not really classify as faces get picked up as being faces. For dealing with these edge cases a simple method is used. In the first case where background people are getting picked up, a simple solution is to simply accept only the biggest





bounding box as the valid one. As for the other cases where same region gets picked up multiple times, non max suppression is used in order to declare a single region of interest. Finally in some cases, faces do not get detected at all. Such cases were in minority so they are simply ignored.

5 Experiments

In this section we will explain several experiments that we have run and discuss their results. In these experiments we have tried setting different values for hyperparameters such as: type of LFW dataset used, type of triplet generation strategy, base model for the encoder, number of layers to fine-tune in this mentioned base model for encoder and finally margin for triplet loss.

Before getting into experiments, we want to briefly talk about our metrics and how they are calculated. Our model's output is not a class label like it would be the case in regular classification networks. As explained earlier, we are utilizing the difference between the distance of image embeddings anchor-positive and anchor-negative. Since there isn't a single class output like usual, we have to define our own way determining a correct and a false classification and metrics that are dependent on these.

As explained in the triplet loss section, our overall goal with triplet loss is to increase the distance between anchor and negative images and decrease the distance between anchor and positive images. Following the same idea we define true positives, false negatives, true negatives and false positives as follows:

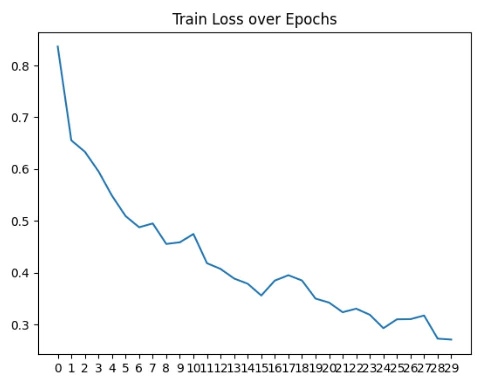
$$d_{ap}, d_{an} = model.predict(a, p, n)$$

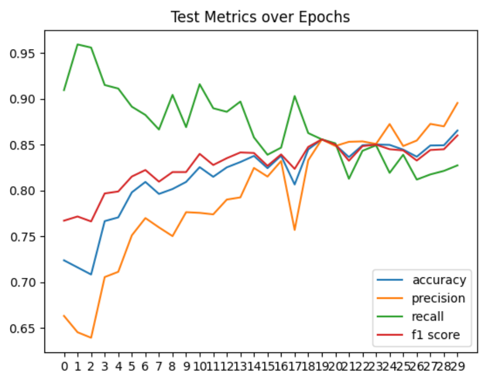
$$\begin{aligned} TP &= count(d_{ap} < threshold) \\ FN &= count(d_{ap} \geq threshold) \\ TN &= count(d_{an} > threshold) \\ FP &= count(d_{an} \leq threshold) \end{aligned}$$

Since model.predict returns a batch of distances between anchor-positive and anchor-negative images, we can count distances from these returned values which satisfy our goal of increasing the distance between anchor-negative images and decreasing the distance between anchor-positive images. There are simpler approaches for calculating accuracy such as; for a given triplet if distance(a, p) < distance(a, n) then this triplet is considered to be true end in other situation, triplet is considered to be classified false. However unlike this simpler approach, the approach above allows us to calculate more metrics such as precision, recall and f1. The value of threshold can be configured as pleased for the task and data at hand. In our situation we have selected the threshold as 0.5

The below experiment uses lfw-deepfunnelled dataset which has the faces aligned. Random triplet mining strategy along with a base model of

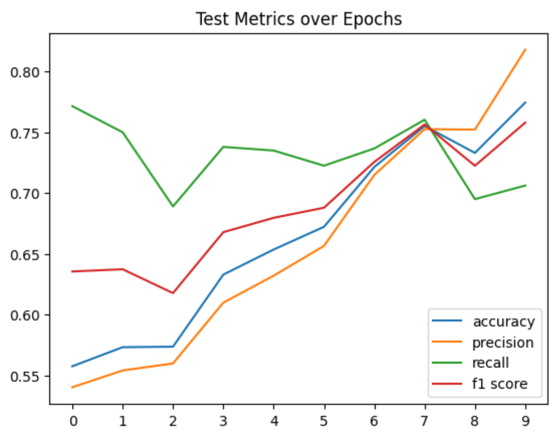
InceptionResNetV2 with it’s last 15 layers configured as trainable is used.





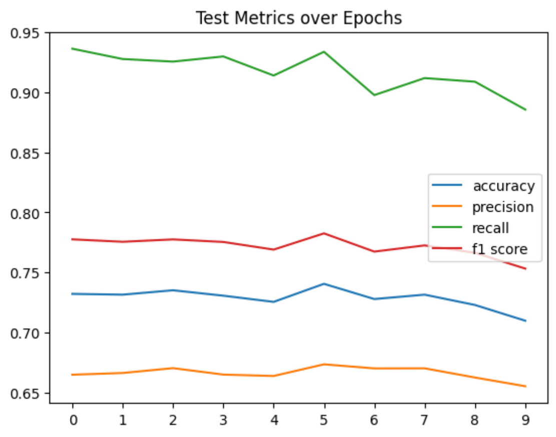
As we can see from figures above the model trains as expected, lowering the training loss over time and converging an accuracy value of 0.86 in the test metrics over epochs graph. This model has been trained for 30 epochs. This configuration with these hyperparameters and configuration was the best achieving model excluding hard triplet mining strategy which yielded the overall result. All the other experiments we will look at after this one are run with 10 epochs. This model was continued for 20 more epochs after the initial 10 epochs. Since it was the most promising configuration, we wanted to see if there would be any gain in performance over incearsed epochs. Longer training did increase the epoch from below 0.85 in epoch 10 to 0.85 in epoch 19 and above 0,85 in epoch 29 how ever they were marginal gains. The other, even more promising configuration which has achieved 0.90 accuracy in 10 epochs was not subjected to longer training like this configuration because hard triplet mining is a very computationally demanding task due to requirement of computing distance between every image in a given batch. With these details in mind. Below we provide more experiments and findings.

In the next experiment, we change the dataset from the previous configuration to use the non-aligned faces dataset (non-preprocessed) to see what effect face alignment has on face recognition. Results from this experiment are as below.



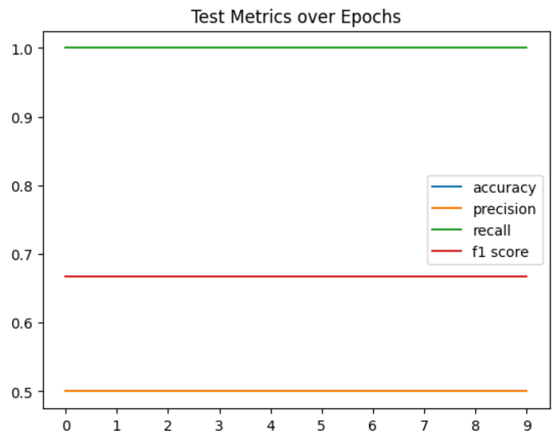
As we can see the accuracy is 0.76 which is 0.1 points lower than the dataset with face alignment applied. This result is expected because face alignment makes extraction of the features easier for the encoder network. This is further proven by more sophisticated approaches like FaceNet and DeepNet employing 3d facial alignment as novel approaches for increasing performance of face verification systems.

In the next experiment we test the effect of base model that was used in construction of the encoder network. In our study we tested two different base models: InceptionResNetV2 and VGG19. Results are provided in below graph.



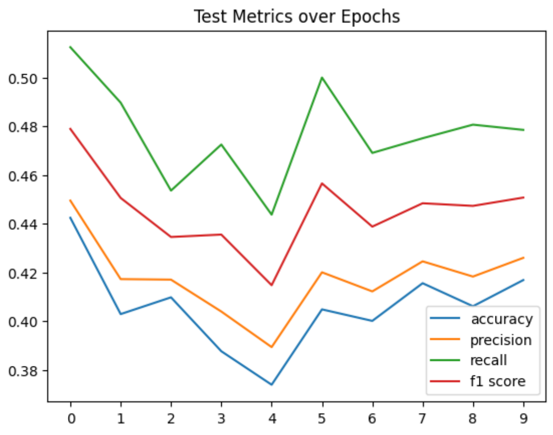
As shown in the earlier (first) experiment, InceptionResNetV2 base model achieves an accuracy score if 0.86 where as under same hyperparameters, our outher model, VGG19 achieves a score of 0.77.

In our next experiment we test how the number of fine tuned layers change the performance of the overall network. So far in our experiments we have been fine tuning only the last 15 layers of the InceptionResNetV2 network. In below experiment we provide results for allowing fine tuning of the last 100 layers.



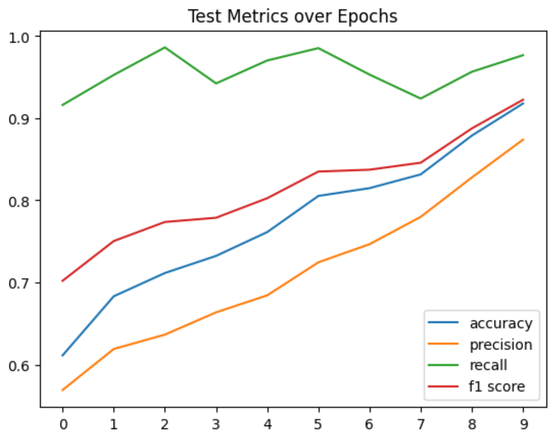
Results are interesting because as we can see there has been absolutely no optimization on the network after 10 epochs. None of the anchor, negative pairs have been classified as such. Instead all of these pairs are wrongly classified as belonging to same person. Where as since model classifies every pair as belonging to same person, the true positive count is maximun and recall score is 1.0. Since the dataset is balanced (in the sense that there are equal number of (anchor, positive) pairs which have all been classified correctly as (anchor, negative) pairs which have all been classified falsely), an overall accuracy score of 0.5 is returned. This line is under the precision line in above graph and can't be seen. We are not really sure why this outcome has occured. To our understanding, no matter how many layers are freezed or not, the result, at the beginning epochs should not be affected much. If we were to set trainable layer weights to all 0 and try learning them from scratch, this situation could be explained with 10 being a limited amount of epochs to even start seeing an optimization for a network with 100 layers. However in the case of fine-tuning, the existing weights should be started as is and updated from there. In summary, we are not sure how to comment on this particular experiment.

In our next experiment we increase the margin size of the triplet loss defined by the alpha term as discussed in earlier sections of this report. For this experiment, we increase our default margin size of 1.0 to 5.0 to see it's effect on performance.

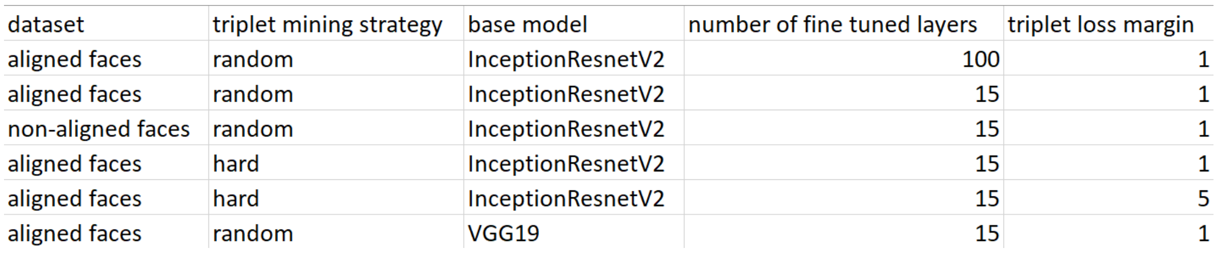


As we can see wit han accuracy score within the range of 0.45 - 0.48 it is the worst scoring training configuration overall. 5.0 was a too big margin for our task and caused network to have convergence issues. Therefore it makes sense to stick to a margin of 1.0.

In our next and final experiment we are finally moving onto hard triplet mining strategy instead of random. The results from using hard triplet mining for triplet generation is provided below.



As it can be seen from the graph there is a clear learning curve and it is more defined compared to earlier experiments. Moreover, the final score reaches an accuracy level of 0.90 which is a 0.04 increase over our baseline experiment of 0.86. This result is expected as hard triplet mining achieves to remedy the randomness in training process caused by random triplet generation, by providing triplets of images that should help with optimization of the model. We would like to experiment more with hard triplet mining strategy and other non-random mining strategies however they are computationally very expensive and are not feasible for us to run on our local machines.



dataset	triplet mining strategy	base model	number of fine tuned layers	triplet loss margin
aligned faces	random	InceptionResnetV2	100	1
aligned faces	random	InceptionResnetV2	15	1
non-aligned faces	random	InceptionResnetV2	15	1
aligned faces	hard	InceptionResnetV2	15	1
aligned faces	hard	InceptionResnetV2	15	5
aligned faces	random	VGG19	15	1

Summary of all the experiments and their scores can be observed from table above.

Finally, we compare our work to other studies that were benchmarked on the LFW dataset in the table below

Model	Accuracy
Our model	0.90
FaceNet (2015)	0.9963 ± 0.09
GhostFaceNets (2023)	0.998667

As it can be seen from the table above, our appraoch is for from the state of the art of even the year 2015. Today in 2023 state of the art is GhostFaceNets[9] model with an accuracy of 0.998667 and the other reference model which was infuelential and state of the art in it’s time is FaceNet with an accuracy of 0.9963. Our model performs at 0.90 accuracy. We see it as a success that we were able to reach such an accuracy score with transfer learning on limited resources. These two models are state of the art of their time and hence a good point of comparison. Nonetheless, as we said earlier face recognition is a popular research domain and many work are published constantly with different approaches. In below table we provide some of the model’s that come close to today’s state of the art.

Model	Accuracy
Prodpoly [10]	0.99833
DiscFace [11]	0.9983
QmagFace [12]	0.9983
ArcFace + MS1MV2 + R100 [13]	0.9983
DCQ [14]	0.998
AdaFace + WebFace4M + R100 [15]	0.9980
CircleLoss [16]	0.9973
FaceTransformer+OctupletLoss [17]	0.9973
DigiFace-1M [18]	0.9617
OcularAI-Face [19]	0.945

6 Future Work

In this section we will briefly discuss what could have been made better in thus study. There are mainly 2 points that we want to expand upon: triplet mining strategies and pretrained models. In this study we have already employed two different triplet mining strategies which are random and hard triplet mining. Although the hard triplet mining strategy has proven itself to contribute to the performance of the network, there are many other mining strategies such as semi-hard, adaptive and online mining strategies that could be tried for further improvement. In this study we were especially restricted with computational resources. We believe using a triplet strategy which utilizes the embeddings instead of the images themselves could both increase performance and reduce computational complexity at the same time since eucledian distances would be calculated within embedded images on a smaller feature space instead of raw images. The other point we want to discuss is use of pretrained networks. In this study we have used two pretrained networks with ImageNet weights which are InceptionResNetV2 and VGG19 models. Although they have been shown to contribute to accuracy performance, they are not exactly optimized for extraction of features from human faces since they were trained on ImageNet which is a general purpose dataset. Earlier we have mentioned that there are other datasets with millions even 10s of millions of images for face recognition which we haven’t included in our work due to computational limitations. Instead of using these datasets for training, in future works, models that are pretrained on these kinds of face datasets that are optimized for face feature extraction could be utilized instead of the general purpose backbones such as the InceptionResNetV2 and VGG19. We believe that such improvement in backbone architechture which makes up the main portion of our encoder network would be really beneficial to overall model performance.

7 Conclusion

In this study we have developed a face recognition system using siamese networks and triplet loss function. We have also determined the effect of various hyperparameters and training settings' effect on model performance such as triplet mining strategy, using a pretrained network for transfer learning, number of layers to apply fine-tuning on these imported weights for transfer learning, the effect of a very important preprocessing step of the data in face recognition which is face alignment. We have also tested different margin values for our triplet loss function. In the end we provided the reader with our findings from these experiments and finally in the previous section we have explained how our methodology is very open for future development with simple, easy to apply additions to the current state of the work.

8 References

[1] Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of cognitive neuroscience*, *3*(1), 71-86.

[2] Serengil, S. I., & Ozpinar, A. (2021, October). Hyperextended lightface: A facial attribute analysis framework. In *2021 International Conference on Engineering and Emerging Technologies (ICEET)* (pp. 1-4). IEEE.

[3] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).

[4] Huang, G., Mattar, M., Lee, H., & Learned-Miller, E. (2012). Learning to align from scratch. *Advances in neural information processing systems*, *25*.

[5] Ramos-Cooper, S., Gomez-Nieto, E., & Camara-Chavez, G. (2022). VGGFace-Ear: an extended dataset for unconstrained ear recognition. *Sensors*, *22*(5), 1752.

[6] Guo, Y., Zhang, L., Hu, Y., He, X., & Gao, J. (2016). Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14* (pp. 87-102). Springer International Publishing.

[7] Bansal, A., Nanduri, A., Castillo, C. D., Ranjan, R., & Chellappa, R. (2017, October). Umdfaces: An annotated face dataset for training deep networks. In *2017 IEEE international joint conference on biometrics (IJCB)* (pp. 464-473). IEEE.

[8] Viola, P., & Jones, M. (2001, December). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001* (Vol. 1, pp. I-I). Ieee.

[9] Alansari, M., Hay, O. A., Javed, S., Shoufan, A., Zweiri, Y., & Werghi, N. (2023). GhostFaceNets: Lightweight Face Recognition Model from Cheap Operations. IEEE Access.

[10] Chrysos, G. G., Moschoglou, S., Bouritsas, G., Deng, J., Panagakis, Y., & Zafeiriou, S. (2021). Deep polynomial neural networks. IEEE transactions on pattern analysis and machine intelligence, 44(8), 4021-4034.

[11] Kim, I., Han, S., Park, S. J., Baek, J. W., Shin, J., Han, J. J., & Choi, C. (2020). Discface: Minimum discrepancy learning for deep face recognition. In Proceedings of the Asian conference on computer vision.

[12] Terhörst, P., Ihlefeld, M., Huber, M., Damer, N., Kirchbuchner, F., Raja, K., & Kuijper, A. (2023). Qmagface: Simple and accurate quality-aware face recognition. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 3484-3494).

[13] Kim, M., Jain, A. K., & Liu, X. (2022). Adaface: Quality adaptive margin for face recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 18750-18759).

[14] Li, B., Xi, T., Zhang, G., Feng, H., Han, J., Liu, J., ... & Liu, W. (2021). Dynamic class queue for large scale face recognition in the wild. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 3763-3772).

[15] Kim, M., Jain, A. K., & Liu, X. (2022). Adaface: Quality adaptive margin for face recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 18750-18759).

[16] Sun, Y., Cheng, C., Zhang, Y., Zhang, C., Zheng, L., Wang, Z., & Wei, Y. (2020). Circle loss: A unified perspective of pair similarity optimization. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 6398-6407).

[17] Knoche, M., Elkadeem, M., Hörmann, S., & Rigoll, G. (2023, January). Octuplet loss: Make face recognition robust to image resolution. In 2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG) (pp. 1-8). IEEE.

[18] Bae, G., de La Gorce, M., Baltrušaitis, T., Hewitt, C., Chen, D., Valentin, J., ... & Shen, J. (2023). DigiFace-1M: 1 Million Digital Face Images for Face Recognition. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 3526-3535).

[19] Sethuraman, S. C., Tadkapally, G. R., Mohanty, S. P., Galada, G., & Subramanian, A. (2023). MagicEye: An Intelligent Wearable Towards Independent Living of Visually Impaired. arXiv preprint arXiv:2303.13863.