Face Unlock

Muhammed Abdullah Shaikh, Khurshed Fitter, Rishika Bhagwatkar, Shital Chiddarwar
{abdullahbinkhaled, khurshedpf, rishika.vnit}@gmail.com, shitalsc@mec.vnit.ac.in
Visvesvaraya National Institute of Technology, Nagpur, India

This is our project report that aims to understand and develop a face recognition framework. We used MNIST, AT&T and Labelled Faces in the Wild (LFW) datasets during this project. We performed experiment tracking, hyperparameter tuning, and sweeps throughout our work using WandB.
Source files are available at: https://github.com/ABD-01/Face-Unlock.

I. PROBLEM AND MOTIVATION

Face Recognition is a computer vision task that has been extensively studied for several decades. This project was developed at IvLabs[1], the AI and Robotics Lab of VNIT, Nagpur. The motivation was to learn and understand the working of face recognition algorithms and *one-shot* learning. Next, we decided to develop the framework for ourselves from scratch and further deploy it on hardware for a face-recognition-based door-lock system.

II. ONE-SHOT LEARNING

One-shot learning is a classification task where one example (or a very small number of examples) is given for each class, which is used to prepare a model, that in turn must make predictions about many unknown examples in the future.

This characterizes tasks seen in the field of face recognition, such as face identification and face verification, where people must be classified correctly with different facial expressions, lighting conditions, accessories, and hairstyles given one or a few template photos.

Modern face recognition systems approach the problem of one-shot learning via face recognition by learning a rich low-dimensional feature representation, called face embedding, that can be calculated for faces easily and compared for verification and identification tasks.

Historically, embeddings were learned for one-shot learning problems using a Siamese network. The training of Siamese networks with comparative loss functions resulted in better performance, later leading to the triplet loss function used in the FaceNet system by Google that achieved then state-of-the-art results on benchmark face recognition tasks.

III. MODEL ARCHITECHTURE

[**ResNet**] We use residual neural networks from He *et al.*[2] for this work, mostly ResNet18, but also experimented with ResNet26 and ResNet50. With ResNets, the gradients can flow directly through the skip connections backward from later layers to initial filters.

[**Siamese Network**] A Siamese Neural Network[3] is a class of neural network architectures that contain two or more identical subnetworks. 'identical' here means, they have the same configuration with the same parameters and weights. Parameter updating is mirrored across both sub-networks. It is used to find the similarity of the inputs by comparing their feature vectors, so these networks are used in many applications.

[**Triplet Loss**] Triplet loss was introduced in Schroff *et al.*[4] The loss function penalizes the model such that the distance between the matching examples is reduced and the distance between the non-matching examples is increased. Let $d(x, y) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ be a distance function in embedding space of dimension D. The loss function is defined as:

$$L = \sum_{\substack{a,p,n \\ y_a=y_p\neq y_n}} [d(x_a, x_p) - d(x_a, x_n) + \alpha]_+ \qquad (1)$$

This loss makes sure that, the projection of an anchor point x_a, and it's corresponding positive point x_p belonging to the same class (person) y_a is closer than the projection of a negative point x_n belonging to another class y_n, by at least a margin α.

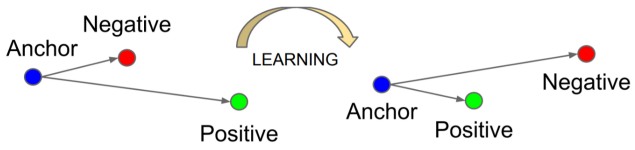

FIG. 1. Triplet Loss

IV. EXPERIMENTS

V. MNIST

Implemented as a warmup for the project to gain an understanding of one-shot learning and siamese nets. The model was trained only on 100 images of classes 0,1,2. Images of rest classes i.e 3-9 were kept hidden during the training phase. The evaluation was done on a 10-way 1-shot basis, wherein the support set Fig. 2 had only one sample from each class 0 to 9.

TABLE I. Result on MNIST Dataset

Class	0	1	2	3	4	5	6	7	8	9
Correct	5804	6648	5830	5877	5830	5274	5908	5589	5777	5849
Total	5923	6742	5958	6131	5842	5421	5918	6265	5851	5949
Accuracy	97.99%	98.60%	97.85%	95.85%	99.79%	97.28%	99.83%	89.20%	98.73%	98.31%

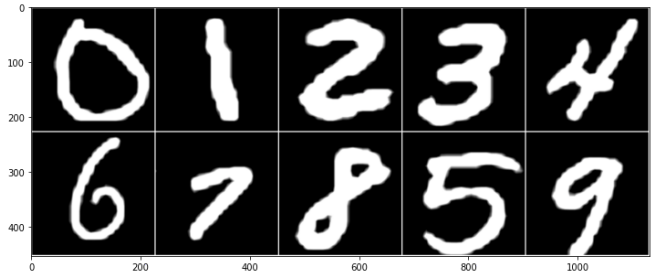

FIG. 2. Support Set for MNIST one-shot task.

VI. AT&T FACE DATASET

[**Dataset Information**] The AT&T face dataset, "(formerly 'The ORL Database of Faces'), contains a set of face images taken between April 1992 and April 1994 at AT&T Laboratories Cambridge. It contains 10 different images of 40 distinct people with 400 face images. The database was used in the context of a face recognition project carried out in collaboration with the Speech, Vision, and Robotics Group of the Cambridge University Engineering Department.

Besides the fact that the images have the same background and same size, the images were converted to a gray level and pixel values were scaled from 0 to 1.

[**Offline Triplet Mining**] In this approach, we first generate the triplets manually and then fit the data into the network. We used two different types of methods

1. All possible triplets - For each image in the dataset (anchor), we choose all possible combinations to make a positive pair, and for each pair choose all possible combinations from the remaining class as Negative.
2. Random triplets - For each image in data (Anchor), randomly sample a Positive image from its class. Then from each of the other classes sample one image (randomly) as the Negative. Hence, for every Anchor, you will have 1 randomly selected positive from its class and a randomly selected Negative from each of the n-1 classes where n is the total number of classes. For every anchor image, you would have n-1 triplets. So if you're having 3 classes of 10 images each then you would have 60 triplets

However this method being too expensive in terms of memory and computation, it was not performed on datasets other than MNIST.

[**Online Triplet Mining**] In this approach, we feed a batch of training data, generate triplets using all examples in the batch, and calculate the loss on it. This approach allows us to randomize the triplets and increase the chance to find triplets with high losses. This will help train the model faster.

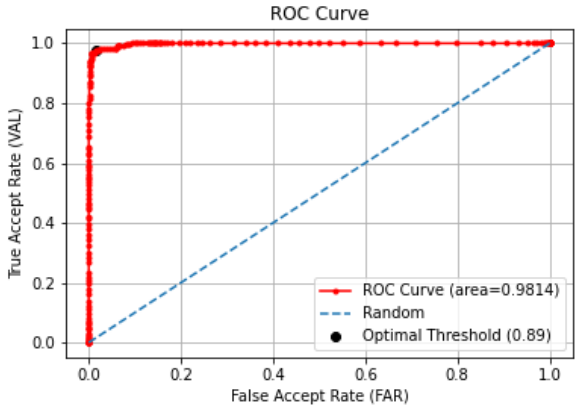

FIG. 3. An ROC curve is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: (i) True Acceptance Rate, (ii) False Acceptance Rate. The area under the ROC curve (AUC) provides an aggregate performance measure across all possible classification thresholds. AUC = 0.981

[**Outcome**] During this experiment, the batch size was 100, the face embedding dimension was 64 and the optimizer used was Adam. The results on this dataset are shown in Table II. We also studied the Regional Optical Characteristics (ROC) Fig. 3 and Equal Error Rate (EER) Fig. 4 curves for this setting. We further plotted the t-SNE chart of the face embeddings to see the clustering behavior of the model shown in Fig. 5.

VII. LABELLED FACES IN THE WILD (LFW)

[**Dataset Information**] Labeled Faces in the Wild[5] is a public benchmark for face verification, also known as pair matching. The data set contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the data set. The only constraint on these faces is that they were detected by the Viola-Jones face detector. A Deep Funneled set of LFW images was used for training and evaluation purpose.

TABLE II. Result AT&T Dataset.

Architechture	No. of learnable parameters	Epochs	Learning Rate	Train Accuracy	Test Accuracy
Plain CNN	4,170,400	16	0.001	92.67%	88.00%
ResNet-18	11,235,904	8	0.002	99.62%	94.73%
ResNet-26	17,728,064	20	0.002	93.00%	69.00%
ResNet-18†	11,235,904	200	0.0002	100%	98.4%

† represents results using online triplet mining, while the rest are using offline mining.

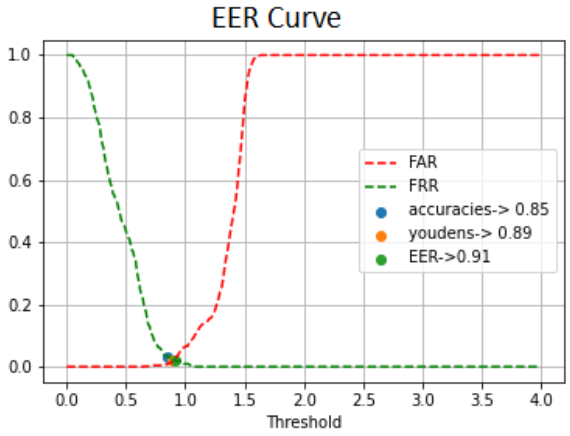

FIG. 4. The EER is the location on a ROC curve where the false acceptance rate (FAR) and false rejection rate (FRR) are equal. In general, the lower the equal error rate value, the higher the accuracy of the biometric system.

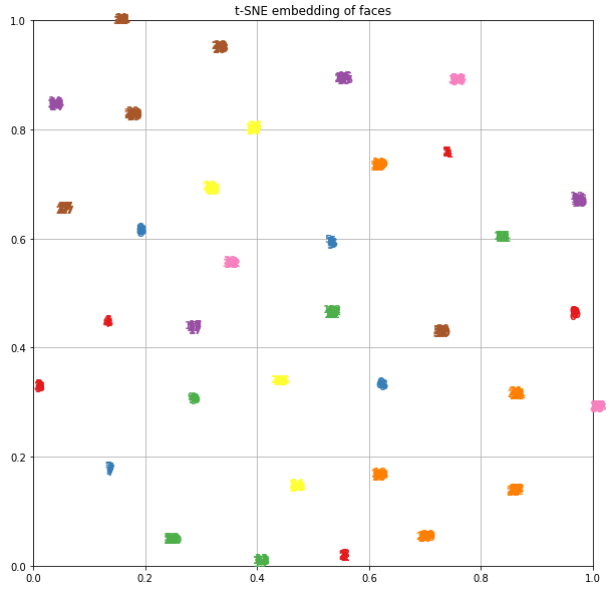

FIG. 5. t-SNE plot for AT&T dataset.

[**Hyperparameter Sweeps**] Hyperparameter search or tuning, or optimization is the task of finding the best hyperparameters for a learning algorithm. Hyperparameter sweeps, enable finding the best possible combinations of hyperparameter values for a specific setting by orchestrating a Run and managing experiments for the configuration of our training. code. We performed a thorough random search for these hyperparameters using WandB[6].

TABLE III. Search on model hyperparameters

Hyperparameter	Range of Search	Selected Value
Batch Size	40,100	256
Epochs	10,50,100,200	200
Learning Rate	0.002, 0.001, 0.0002, 0.0001	0.002
Embedding Dimension	64, 128, 256	128

[**Outcome**] We use the ResNet-18[2] architecture and the pre-trained weights. We discard the last layer and add a fully connected layer of size 128 corresponding to the embedding dimension. The optimizer used was Adam with a weight decay of 0.01 and rest hyperparameters as mentioned in III. We start with a learning rate of 2×10^{-3}, and divide it by 2 at every 50 epochs, the chance in learning rate is depicted in Fig. **??**. We train with batch hard triplet loss as shown in Hermans *et al.*[7]. We evaluate our performance referring to Tahamoqa[8] on benchmark for comparison, a 10-fold cross-validation split provided on the official website of LFW. Apart from accuracy, we also studied TAR, FAR, and precision of the model.

[**Biometrics**] There are four possible outcomes from a binary classifier. If the outcome from a prediction is p and the actual value is also p, then it is called a *true positive* (TP); however, if the actual value is n then it is said to be a *false positive* (FP). Conversely, a *true negative* (TN) has occurred when both the prediction outcome and the actual value are n, and *false negative* (FN) is when the prediction outcome is n while the actual value is p. The performance of biometric systems is expressed on the basis of the following error rates:

1. *Accuracy* $= \dfrac{\text{TP} + \text{TN}}{\text{TP+FP+FN+TN}}$
2. *True Acceptance Rate* also known as *sensitivity* or

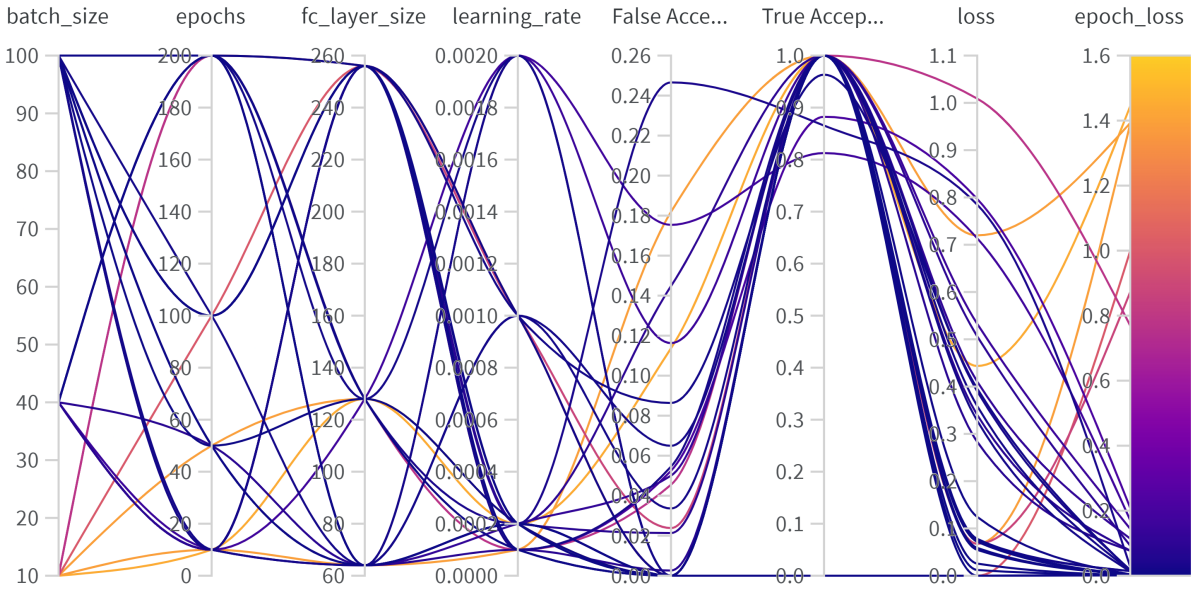

FIG. 6. Hyperparameter Sweep using WandB

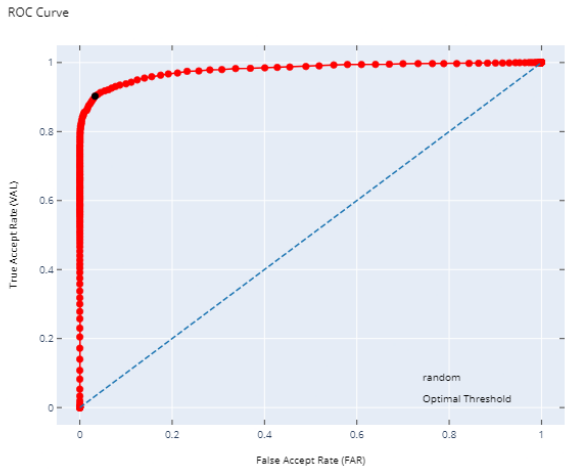

FIG. 7. ROC Curve. The Optimal threshold is found to be 1.10 at TAR=0.864, FAR=0.133. TAR@FAR=0.01 is found to be 0.5963

recall

$$\text{TAR} = \frac{\text{TP}}{\text{TP+TN}} = 1 - \text{FRR}$$

3. *False Acceptance Rate (FAR)*

$$\text{FAR} = \frac{\text{FP}}{\text{FP+TN}} = 1 - \text{TAR}$$

4. *True Rejection Rate (TRR)* also known as *specificity*
5. *False Rejection Rate (FRR)*

The ROC curve Fig. 7 is useful to understand the trade-off in TAR and FAR for different thresholds. We use Youden's J statistic to determine the best threshold for the classification.

$$J = \text{sensitivity} + \text{specificity} - 1$$
$$J = \frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}} - 1$$

VIII. IVLABS

[**Dataset Information**] A custom dataset of faces of Lab members. We used the MTCNN[9] framework to detect and align faces provided by Sandberg[10] to extract the dataset. The images of the members were taken from and can be viewed at `https://www.ivlabs.in/core-coordinators.html`. For face-detection on raspberry-pi, we used MobiFace[11]

[**Outcome**] We implemented both FaceNet[4] and ArcFace[12] frameworks for this dataset, using transfer learning. We used pre-trained weights on ImageNet for the ResNet18 backbone and trained on the LFW trainset. We fined tuned on IvLabs' face dataset as a classification task by freezing the weights of the backbone and adding a softmax classifier head. We used SGD with a momentum of 0.9, a learning rate of 0.0005, and a one-cycle scheduler from [Paper] for 100 epochs and max. value of the learning rate as 0.005. The predictions are shown in Fig. 10

FIG. 8. Epoch Loss

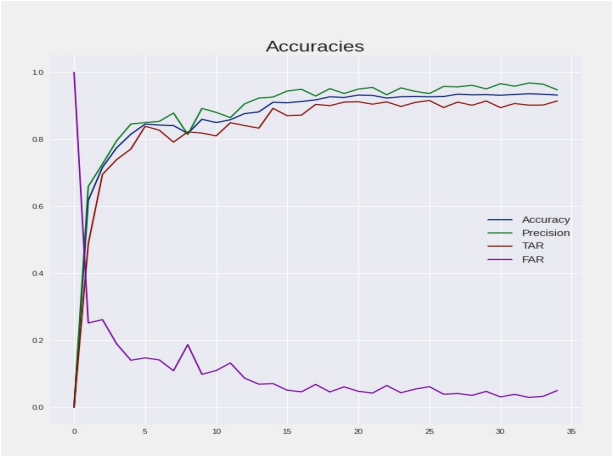

FIG. 9. Learning Curves

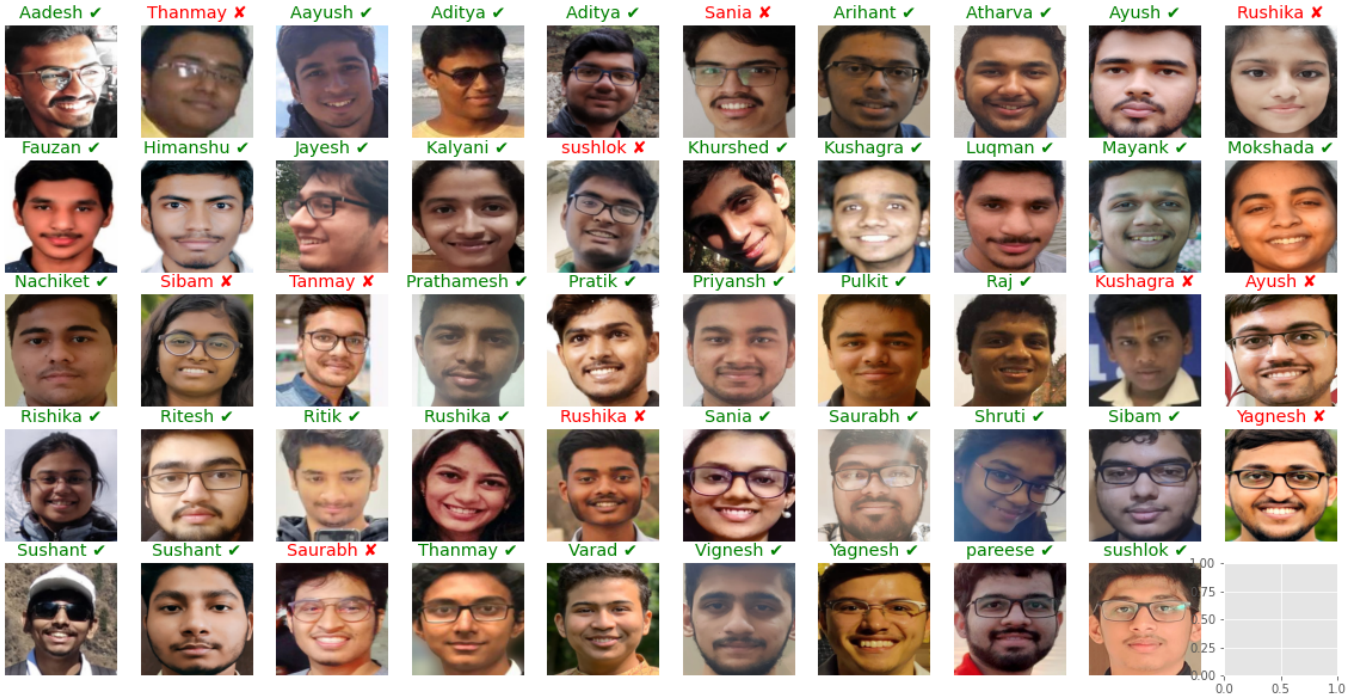

FIG. 10. Results on IvLabs' Face Dataset

---

[1] https://www.ivlabs.in/.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016) pp. 770–778.

[3] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah, "Signature verification using a "siamese" time delay neural network," in *Advances in Neural Information Processing Systems*, Vol. 6, edited by J. Cowan, G. Tesauro, and J. Alspector (Morgan-Kaufmann, 1993).

[4] Florian Schroff, Dmitry Kalenichenko, and James Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2015).

[5] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller, *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environ-*

ments, Tech. Rep. 07-49 (University of Massachusetts, Amherst, 2007).

[6] Lukas Biewald, "Experiment tracking with weights and biases," (2020), software available from wandb.com.

[7] Alexander Hermans, Lucas Beyer, and Bastian Leibe, "In defense of the triplet loss for person re-identification," (2017).

[8] Tamer Tahamoqa, "facenet-pytorch-glint360k," https://github.com/tamerthamoqa/facenet-pytorch-glint360k.

[9] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao, "Joint face detection and alignment using multi-task cascaded convolutional networks," IEEE Signal Processing Letters **23**, 1499–1503 (2016).

[10] David Sandberg, "facenet," https://github.com/davidsandberg/facenet/tree/master/src/align.

[11] Chi Nhan Duong, Kha Gia Quach, Ibsa Jalata, Ngan Le, and Khoa Luu, "Mobiface: A lightweight deep learning face recognition on mobile devices," (2018).

[12] Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Cotsia, and Stefanos P Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence , 1–1 (2021).