ATTENDANCE AUTHENTICATION USING SIAMESE NEURAL NETWORKS

Saadhikha Shree S (19BPS1075), Sruthika B (19BPS1112)

School of Computer Science and Engineering , VIT University, Chennai, India

saadhikhashree.s2019@vitstudent.ac.in

sruthika.b2019@vitstudent.ac.in

***Abstract*— The process of learning good features for machine learning applications can be very computationally expensive and may prove difficult in cases where little data is available. A prototypical example of this is the one-shot learning setting, in which we must correctly make predictions given only a single example of each new class. In this project, we have explored a method for learning "Siamese Neural Networks", which employs a unique structure to naturally rank similarity between inputs. Once a network has been tuned, we can then capitalize on powerful discriminative features to generalize the predictive power of the network not just to new data, but to entirely new classes from unknown distributions. Using a convolutional architecture, we are able to achieve strong results which exceed those of other deep learning models with near state-of-the-art performance on one-shot classification tasks. The aim of this project is to build a facial recognition model, which can recognize faces with just one shot and then integrate the model into a working app.**

***Keywords*— Face detection, Siamese networks, attendance systems, one-shot learning**

I. Introduction

The information age is quickly revolutionising the way transactions are taking place. Everyday actions are increasingly being handled electronically, instead of with pen and paper or face to face. This growth in electronic transactions has resulted in a greater demand for fast and accurate user identification and authentication. Access codes for buildings, bank accounts, and computer systems often use PIN's for identification and security clearances.

Using proper PIN gains access, but the user of the PIN is not verified. When credit and ATM cards are lost or stolen, an unauthorized user can often come up with the correct personal codes. Despite warnings, many people continue to choose easily guessed PIN's and passwords: birthdays, phone numbers.

This brings out the need for biometrics. A Biometric is a unique, measurable characteristic of a human being that can be used to automatically recognise an individual or verify an individual's identity. Biometrics can measure both physiological and behavioural characteristics.

- Physical biometrics include:
- Finger – scan
- Facial Recognition
- Iris – scan
- Retina – scan
- Hand – scan

II. Literature survey

Paper 1: "Deep Face Recognition: A Survey"[1]

The research paper aims to summarize the main advances in deep face recognition and, more in general, in learning face representations for verification and identification. This paper presents a clear, structured presentation of the principal, state-of-the-art (SOTA) face recognition techniques appearing within the past 5 years in top computer vision venues.

Paper 2: "Face Recognition from Video using Deep Learning"[2]

In this paper, they've proposed a face recognition system that makes searching for criminals easy and quick with less time and hence efficiently helps police and administration. There is lots of information about the applications of Facial Recognition in various domains.

Paper 3: "Deep face recognition for Biometric Authentication"[3]

This paper starts with how, most of the traditional methods lack robustness against varying illumination, facial expression, scale, occlusions and pose.

In this paper, they've presented a convolutional neural network based face recognition system which detects faces in an input image using Viola Jones face detector and automatically extracts facial features from detected faces using a pre-trained CNN for recognition.

Paper 4: "Systematic evaluation of deep face recognition methods"[4]

This study compares several popular model compression methods and shows that "MobileNet" has advantages over the others in terms of both compression ratio and robustness.

Paper 5: "Exploring Features and Attributes in Deep Face Recognition Using Visualization Techniques"[5]

This paper explores problems like: What effective features does a deep face model learn? and What do these features represent and what is the semantic meaning of them? By analyzing the classic network VGG Face using deep visualization techniques.

Table1: Comparison between existing literature

S.No	Summary of the papers		
	Journal	**Model**	**Performance**
1	IEEE	Deep convolutional neural networks (DCNN)	Proper comparison for face recognition techniques
2	IEEE	Deep convolutional neural networks (CNNs) and Classic network VGGFace	They've done a clear and great comparison and have stated the inner workings of the deep face model also found in [5].
3	IEEE	Advanced Deep Learning, CNN, Data scraping, Euclidean models.	Effective usage of CCTV footage for production of face data.
4	IEEE	CNN Model, Image Augmentation, Hyperparameter selection algorithm	With an overall accuracy of 98.76%, are obtained which depict the effectiveness of deep face recognition in automated biometric authentication systems.
5	Hindawi	Deep Learning algorithm strategies, Data augmentation, Network training algorithms.	A comprehensive evaluation framework and measures the effects of multifarious settings in five component has been provided, including data augmentation, network architecture, loss function, network training, and model compression

III. Proposed Work

One common type of recognition system, which is even used in our university is: Finger – scan based biometrics. But, due to the current pandemic situation, the usage of finger biometrics is not safe. That is the motivation to enhance the current facial recognition models, and present a more accurate model.

To achieve this, the following datasets are used. ANCHOR IMAGES: One shot images personally created while entering a user's facial information into the database. This is the image which is used to verify the users. POSITIVE IMAGES: Images that are similar and match the user's face (which is the anchor image). Output from the model would be 1. NEGATIVE IMAGES: Images that don't match any of the anchor images, and results in verification failure. Output from the model would be 0. The Anchor images and positive images dataset is personally created by me (user data), for the negative images used during model training has been obtained from the "Labelled faces in the wild" data set.

The main idea of a siamese network is to compare two images and find the similarity index between them, then authenticate with respect to the similarity index.

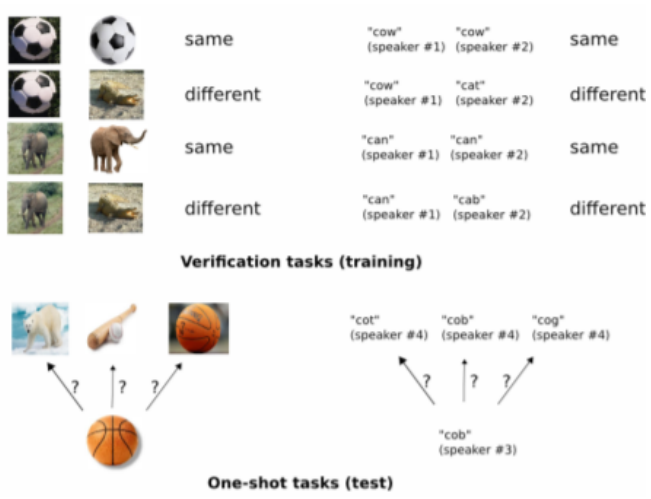

Fig. 1 One shot Image Authentication [6]

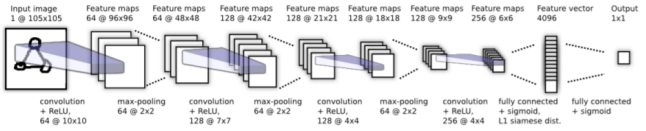

Fig. 2 Siamese Network Model [6]

IV. Experimental Setup and Results

Fig. 3 Verification idea

The figure 3 summarizes the verification idea used in the project. The overall development of the model and the application can be divided into 9 steps. Setup, Collecting positives and anchor images Loading and pre – processing images. Model Engineering, Training the model and saving the checkpoints, Evaluating the model, Saving the model, Real time testing of the model. Developing the application of the model.

A. Setup

Setting up the python project, importing dependencies. "Jupyter notebook" has been used to develop the Machine Learning model for this project. The process is as follows: Installing the dependencies required for this project, Importing the dependencies required for this project, Set GPU Growth to avoid computation errors and setting up the folders and making the required directories.

B. Collecting positives and anchor images

This step focuses on creating the positive and the anchor images for the app. It has two steps, Untar Labelled Faces in the Wild Dataset [9] and collects positive and anchor dataset.

C. Loading and pre - processing images

This module of the project is subdivided into 4 different parts. It involves getting image directories, preprocessing – Scale and resize, creating Labelled dataset and Building train and test partition. The creating step includes a bunch of positives and negatives for anchors. For example, when we compare a positive image with its anchor, the model has to give the comparison array as: (1,1,1,1,1), else the array has to be (0,0,0,0,0). Proceeding and comparing the similarity output model has to print 0 for unverified images and 1 for verified images.

D. Model Engineering

In this step, our focus is to build our model's layers one by one. Proceeding and comparing the similarity output model has to print 0 for unverified images and 1 for verified images.

It is done by Build Embedding Layer: Image processing happens here. The output of this model would be a single dimensional array.

Building Distance Layer: This layer compares the similarity between the arrays generated from the images.

Making Siamese Model: Processing the input and the anchor image and predicting the similarity output.

E. Training the Model

To train the model the following steps were taken. Firstly the Setup of Loss and Optimiser is done. Followed by this, the next steps are to Establish Checkpoints, Build Train Step function, Build Training Loop and finally train the model. While the train function focuses on training one batch, the loop here will be used to iterate over every batch in the dataset. In the end, The model has been trained for 50 EPOCHS, but can be increased to increase the model efficiency.

F. Evaluating the model

The model is evaluated by Importing Metrics, Making predictions, calculating metrics and visualizing results.

Here, Precision and recall are 2 different metrics which help us calculate the similarity between 2 images (0-1). Difference between Precision and Recall : Precision demonstrates what proportion of positive identification were actually correct, recall shows what proportion of actual positives were identified correctly.

Terms to remember are: Metrics above which a prediction is considered positive - Detection threshold and Proportion of positive predictions / total positive samples - Verification threshold.

Following this, the model is saved and taken for real time testing. A verification function and OpenCV real-time verification is done on the model.

Verification Function

```
os.listdir(os.path.join('application_data', 'verification_images'))

def verify(model, detection_threshold, verification_threshold):

# Build results array

results = []

for image in os.listdir(os.path.join('application_data', 'verification_images')):

input_img = preprocess(os.path.join('application_data', 'input_image', 'input_image.jpg'))

validation_img = preprocess(os.path.join('application_data', 'verification_images', image))

# Make predictions

# Wrapped up in list since we have only 1 input_image

result = model.predict(list(np.expand_dims([input_img, validation_img], axis=1)))

results.append(result)

detection = np.sum(np.array(results) > detection_threshold)

verification = detection / len(os.listdir(os.path.join('application_data', 'verification_images')))

verified = verification > verification_threshold

return results, verified

# Metrics above which a prediction is considered positive - Detection threshold

# Proportion of positive predictions / total positive samples - Verification threshold
```

OpenCV real time verification function:

```
cap = cv2.VideoCapture(0)

while cap.isOpened():

ret, frame = cap.read()

frame = frame[120:120+250,200:200+250, :]

cv2.imshow('Verification', frame)

# Verification trigger - click 'v'

if cv2.waitKey(10) & 0xFF == ord('v'):
```

```
#       Save       input       image       to
spplication_data/input_image folder

cv2.imwrite(os.path.join('application_data
','input_image','input_image.jpg'), frame)

# Run verification

results,   verified   ==verify(model,   0.5,
0.55)

print(verified)

if cv2.waitKey(10) & 0xFF ==ord('q'):

break

cap.release()

cv2.destroyAllWindows()

model.summary()
```

Past these procedures, a Kivy application is built with the model. The model has been combined into an application using Kivy [7,8]. By following the documentation and importing the code, the final result is visualized in the app.

V. Results

The results obtained from model verification and application interface are given below along with Siamese model statistics.

```
Model: "SiameseNetwork"
__________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
input_img (InputLayer)       [(None, 100, 100, 3)] 0
__________________________________________________________________________________________________
validation_img (InputLayer)  [(None, 100, 100, 3)] 0
__________________________________________________________________________________________________
embedding (Functional)       (None, 4096)              38960448  input_img[0][0]
                                                                 validation_img[0][0]
__________________________________________________________________________________________________
l1_dist_3 (L1Dist)           (None, 4096)              0         embedding[0][0]
                                                                 embedding[1][0]
__________________________________________________________________________________________________
dense_3 (Dense)              (None, 1)                 4097      l1_dist_3[0][0]
==================================================================================================
Total params: 38,964,545
Trainable params: 38,964,545
Non-trainable params: 0
__________________________________________________________________________________________________
```

Fig. 4 Siamese Model verification

Verification for a true condition (match found) is found in figure 5.

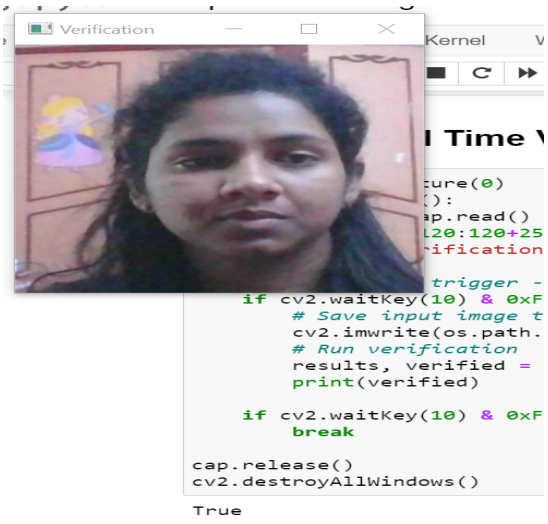

Fig. 5 Verified- match found from dataset

Verification for a false condition (match not found) is found in figure 6.

Fig. 6 Unverified- match not found from dataset

The application interface and verification results are given below. figure 7 and 8 represent verified and unverified conditions respectively.

Fig. 7 Verified- match found on the app from dataset

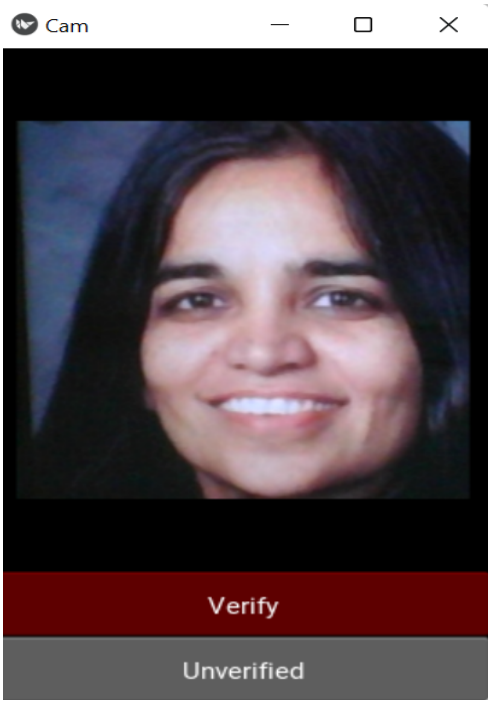

Fig. 8 Unverified- match not found on the app from dataset

The model's accuracy can be determined from the result verification. The model trained produces accurate results for upto 88.56%.

VI. Future Scope

This machine learning model can be integrated into any app which requires facial recognition. Other than that, this model can be expanded to store a variety of faces, instead of just a single person's data. This would be really helpful for credit card verifications too, because multiple users like family members can use the same card. This model can be used in IoT for creating smart doors, which could open the doors for the residents in the home.

VII. Conclusions

We can observe that the Siamese Neural Network model is a promising model, which when incorporated into an app will be really helpful for face authentication. The model gives a good accuracy level of 88.56%, which could be further improved using good clarity cameras, using higher EPOCHS during training and including more layers in the neural networks.

Acknowledgment

The authors are very grateful to the authorities of VIT University - Chennai campus for providing the necessary support and encouragement to perform this project work fruitfully.

References

[1] I. Masi, Y. Wu, T. Hassner and P. Natarajan, "Deep Face Recognition: A Survey," 2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 2018, pp. 471-478, doi: 10.1109/SIBGRAPI.2018.00067.

[2] Y. Zhong and W. Deng, "Exploring Features and Attributes in Deep Face Recognition Using Visualization Techniques," 2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019), 2019, pp. 1-8, doi: 10.1109/FG.2019.8756546.

[3] S. Manna, S. Ghildiyal and K. Bhimani, "Face Recognition from Video using Deep Learning," 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 1101-1106, doi: 10.1109/ICCES48766.2020.9137927.

[4] M. Zulfiqar, F. Syed, M. J. Khan and K. Khurshid, "Deep Face Recognition for Biometric Authentication," 2019 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), 2019, pp. 1-6, doi: 10.1109/ICECCE47252.2019.8940725.

[5] You, M., Han, X., Xu, Y., & Li, L. (2020). Systematic evaluation of deep face recognition methods. Neurocomputing, 388, 144-156.

[6] Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." ICML deep learning workshop. Vol. 2. 2015.

[7] Tensorflow Documentation,https://www.tensorflow.org/api_docs

[8] Kivy Documentation,https://kivy.org/doc/stable/guide-index.html

[9] Negative dataset – "FACES IN THE WILD DATASET" - http://vis-www.cs.umass.edu/lfw/

[10] Parmar, Divyarajsinh N., and Brijesh B. Mehta. "Face recognition methods & applications." arXiv preprint arXiv:1403.0485 (2014).