Similar Face Detection for Indian Faces using Siamese Neural Networks

Rama Devi P, Yashashvini R, Navyadhara G, Ruchitha M.
Department of Computer Science and Engineering, PES University, Bangalore, India
pramadevi@pes.edu, rachamalluyashashvini3@gmail.com,
navyadhara79@gmail.com, mruchitha333@gmail.com.

Abstract: Now a days facial recognition is a very common yet highly significant task all around the globe. Most of the times facial detection or facial similarity detection would lead the thought towards complex neural networks like CNN's. But the task of working with CNN's would be very computationally expensive and also require huge amounts of data for better results. In this work we focus on implementing the task of facial similarity detection using Siamese neural networks, a rather simple architecture which is capable of one-shot learning. A new dataset has been created and tested for facial similarity using Siamese neural networks and the model is able to predict the top similar images of a given image very accurately.

Keywords—Siamese Neural Network, One-shot learning, Contrastive Loss, Facial similarity detection, K-means clustering, Pytorch.

1 Introduction

Facial similarity is one task everyone does unknowingly. The Siamese network is analogous to how the human brain works, i.e. how the human brain stores information about the face through skin tone, eyes, the shape of nose etc. If a human brain sees a person with similar features it tries to match it with the faces which are already in its memory. If that face is similar then there is a high probability that it is the same person. If it sees a new face it stores that face in memory for future predictions. Similarly the main intuition behind Siamese Network is to learn face features and in the Siamese network, a function transforms the input image of the face into a tensor which contains a representation of the face's features. We then calculate the dissimilarity score between two.

1.1 Why is it important?

For suppose imagine a company which hires new employees very frequently and the company gives access to the employees through facial detection. If regular neural networks like CNN, RNN's are used in this case, they need a huge amount of data to be trained and give accurate results which is not feasible. Next, whenever a new employee is hired, to make the system recognize a new face, the whole model needs to be trained again which is computationally very expensive.

Siamese Neural networks can solve the above two challenges. Firstly, Siamese networks are capable of One-Shot learning, which solves the problem of huge data requirement. Next, Siamese neural networks do not train the model but instead train the loss function which is not computationally expensive.

1.2 Real World Applications

i. In a crime investigation centre where you get an image of the culprit who did the crime and wanted information regarding him. In such a case, you can run this image against the whole dataset and this model can give you the most similar image and information regarding him.
ii. In a company where face verification is used as a biometric [11]. Can consider two cases- a small company and a huge company
    a. In the first case, the dataset will be less, since there is less number of employees and when there is less data CNN tends to give a bad result.
    b. In the second case, since it is a huge company, recruiting happens each and every day and your database is updated very frequently. If you consider CNN the whole model is re-trained if there is a change in the dataset, it is highly expensive for the whole model to train each and every time.
    c. In both these cases, Siamese networks give a simple solution to this problem.
iii. A governmental agency could eradicate the use of ID cards and simply use a FRID(Facial re-identification) system to handle attendance, without worrying about a stranger stealing someone's ID to enter the system. [10]

2 Related work

2.1 One-Shot Learning

The process of learning good features for various applications can be very difficult in cases where little data is available. In these papers [1-2,9] they explored a method which handled this hurdle subtly i.e. One-Shot learning. This is a technique where it doesn't need much data to find similarities or find any patterns, maximum it needs one or two images. We also discovered from this paper that Siamese Network is highly compatible with this method. They worked on an omniglot dataset.

2.2 Advantage of Siamese Networks Over Other Available Networks

There are many advanced methods to solve the problem of 'Facial Verification'. But all of them equally need a huge dataset for their computation. In this paper [3] they have discussed how these advanced methods have performed on smaller datasets and what was the outcome. These methods have been tested against the Siamese model with the same dataset and surprisingly Siamese model outperformed those advanced models and the mentioned accuracy is claimed to be 95.62% on LFW [Labeled Faces in Wild] dataset.

2.3 Siamese Network Architecture

In these papers [4,7-8] they have proposed a detailed Siamese network architecture model which is used to train the similarity function. It contains 2 twin Convolution neural networks which share similar weights and architecture. Here we learnt that we won't be training the whole model but would be training the similarity function, hence computation would be really less and is much faster than regular approaches. This approach reached 98.9% accuracy on the LFW dataset and 99.1% accuracy on Arabian Face Dataset.

3 Model Pipeline

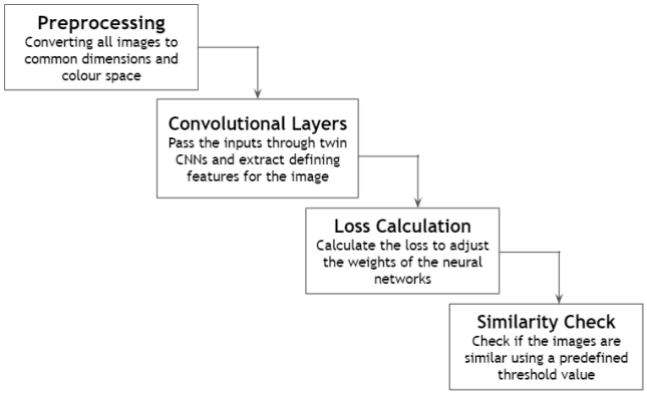

Fig 1. Pipeline of the implemented model

3.1 Dataset

The dataset we propose [5] is a combination of images from Indian Face Database, IIT Kanpur and images collected for this work by us. There are totally images of 18 different faces of which 10 are from Indian Face Database with 15 images of each face and 8 from the data collected newly with 10 images of each face. Some of the images has the face with spectacles and some do not of the same face and the model has handled such variance very well.

Fig. 2. Sample images from the dataset.

3.2 Image Pre-processing

The image class is passed to the pre-processing function, where images are picked at random such that approximately 50% of them belong to the same class and each image is pre-processed before feature extraction by transforming the image dimensions, converting RGB images to greyscale and inverting the image. The image pre-processing phase outputs a pair of images with a label 0 if the pair of images are belonging to the same class else 1.

3.3 Network Architecture

A custom class has been created which contains the architecture of the Siamese neural network implemented using Pytorch. Fig 4 clearly depicts each stage through the architecture. The model has 3 convolutional layers followed by a flattening layer. Images are converted to tensors before passing through the layers in the network. Table 1 describes different operations in the convolutional layer.

Function	**Usage**	**I/P tensor shape**	**O/P tensor shape**
ReflectionPad2d (padding_size=1)	Makes sure that edge features are also included	(64, 1, 100, 100)	(64, 1, 102, 102)
Conv2d (in_features=1, out_features=4, kernel_size=3)	Feature parameter sharing and dimensionality reduction	(64, 1, 102, 102)	(64, 4, 100, 100)
Relu	Non-Linear activation function	(64, 4, 100, 100)	(64, 4, 100, 100)
BatchNorm2d	Normalises the data	(64, 4, 100, 100)	(64, 4, 100, 100)

Table 1. Functions in the Convolutional layer 1 [I/P->Input, O/P-> Output]

Then the (64, 4, 100, 100) tensor is passed through the other 2 convolutional layers which give the final tensor of shape (64, 8, 100, 100). This tensor is later passed to the flattening layer. After passing through all the layers, 128 features are extracted from the image. Table 2 describes different operations in the flattening layer.

Function	**Usage**	**I/P tensor shape**	**O/P tensor shape**
Linear	To preserve the structure of the tensor	(64, 8, 100, 100)	(64, 500)
Relu	Non-Linear activation function	(64, 500)	(64, 500)
Linear	To preserve the structure of the tensor	(64, 500)	(64, 500)
Relu	Non-Linear activation function	(64, 500)	(64, 500)
Linear	To preserve the structure of the tensor function	(64, 500)	(64, 128)

Table 2. Functions in the Flattening layer [I/P->Input, O/P-> Output].

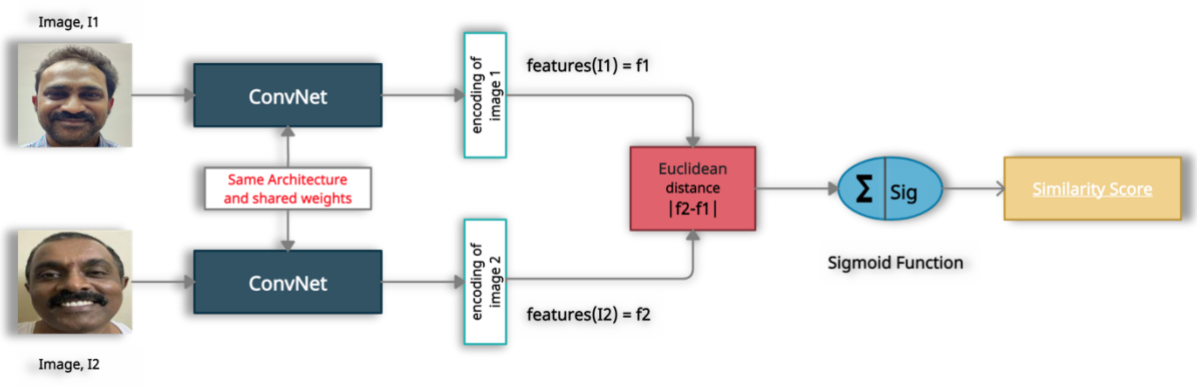

Fig 3. Siamese Network Architecture.

3.4 Loss Function

We used contrastive loss function to train our model[6, 12]. Contrastive loss is always calculated for a pair of images. The contrastive loss can be calculated as,

$$L_{contrast} = (1 - Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\max(0, m - D_W)^2$$

Equation 1: Contrastive loss function

Where,

- *Y* is the label i.e. 0 if similar, 1 otherwise.
- *Dw* is the Euclidean distance between the two images(Essentially distance between the two tensors).
- *m*, is the margin. *m>0*.

3.5 Training Phase

The pre-processed training images are loaded into a iterator with the help of "*dataloader*" function of pytorch. The "*dataloader*" also makes batches of the data. The data after being loaded into the iterator is passed to the training function, which trains the data over multiple epochs. The network is compiled using Adam optimizer with a learning rate of 0.0005 and contrastive loss function.

Then each batch is passed through the network, with two images being chosen at once from the batch sequentially. The images are passed through the Siamese network and image features are extracted. Contrastive loss function is applied over the extracted image features and the loss is used to perform gradient descent and back propagation of weights. By the end of training phase about all the images would have been passed through the network thus extracting the image features and also training our similarity function. The model is saved (into a .pth file). All the image features are collected into a data structure for performing classification in clustering phase[3.6].

Fig 4 clearly depicts different stages data has to pass through in training phase.

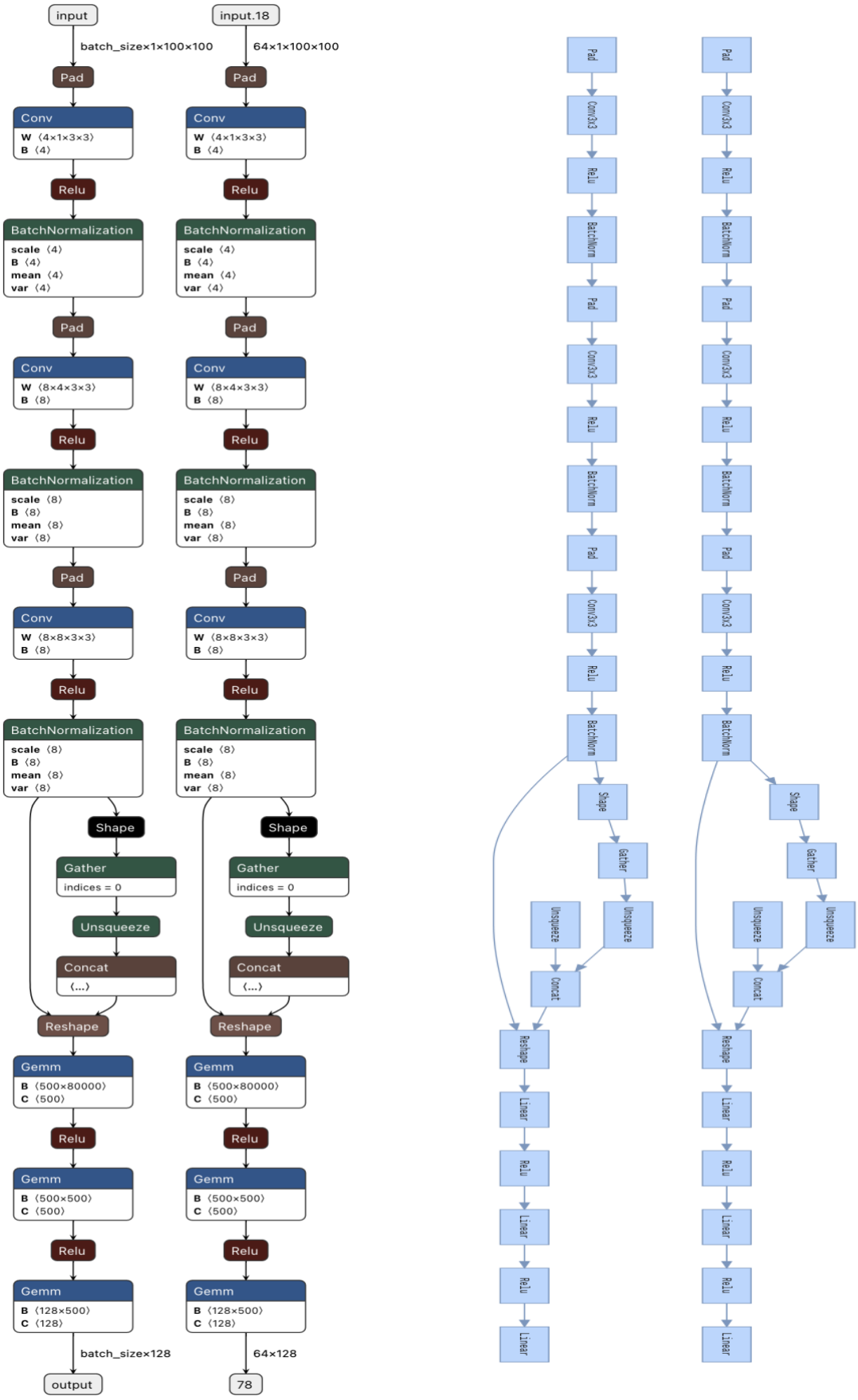

Fig 4. Clear view of stages in the model pipeline

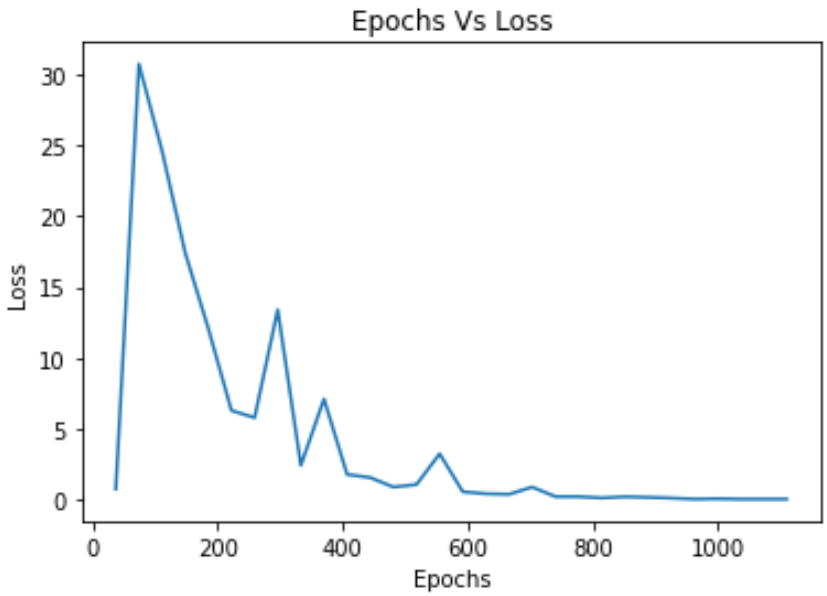

Fig 5. Loss Vs Epochs when trained over 30 epochs.

3.6 Clustering Phase

At the end of training phase, the image features(Unstructured Data) are extracted into a data structure along with image paths(To locate the image). These data structures are used to construct a data frame(Structured Data). The image features, which are tensors after extraction are converted into NumPy arrays before constructing the Pandas data frame. After pre-processing the data obtained from the image features but is unlabelled i.e. has no defined classes. In order to build classes for the data, K-means clustering algorithm is used and the data is differentiated into 18 different classes. The data frame with classes is stored into a comma separated file (c.s.v), and the clustering model is also saved.

4 Experimentation

4.1 Testing Procedure

The trained similarity function is tested over the test data. The test image is passed through the network, extracting the image features. The image features are converted into a NumPy array. The class of the image is decided using the clustering model, then the top similar images are predicted by calculating the Euclidean distance between the input image array and other arrays of the same class.

4.2 Results

Some of the predictions made by the model on the dataset are displayed below. Each image when predicted gave out 3 most similar images to the input image. The corresponding dissimilarity score and related details (here, only name of the person whose face is predicted as similar to the input image) has been displayed along with the images. Lower the dissimilarity score higher the similarity of the two images. The model was also able to handle the variance in faces, such as with and without spectacles or the face in a different angle very well.

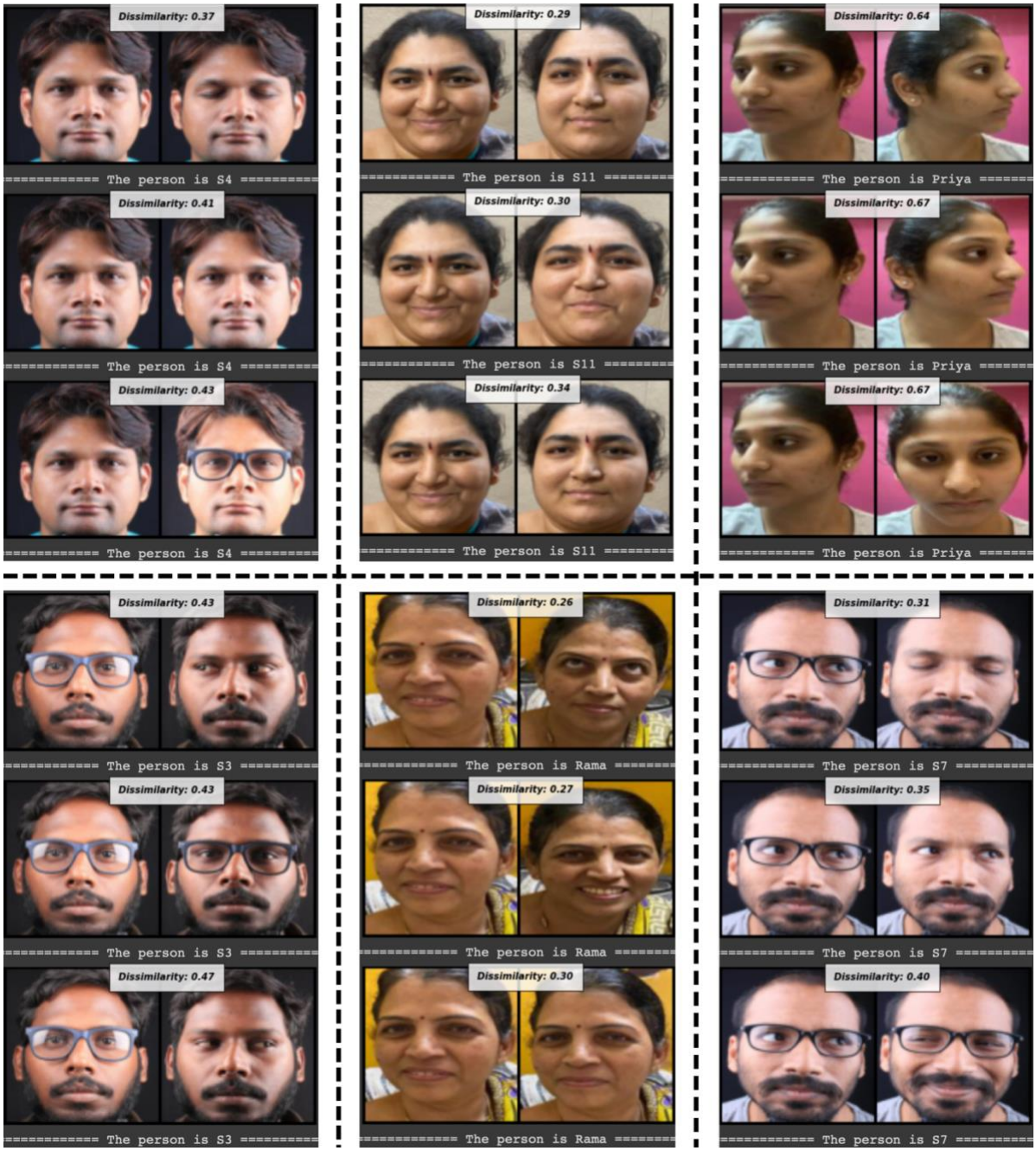

Fig 6. Results obtained from testing on different faces

5 Limitations and Future work

5.1 Limitations

The model was able to predict well in most of the cases for even other datasets like ATT dataset or Yale dataset, but sometimes produces erroneous results when the image is very dark. It also cannot handle images with wrong alignments (ex: An image turned Upside Down).

5.2 Future Works

1. Implementing image enhancement techniques to solve the limitation of erroneous predictions in case of dark or low light images.
2. Riemannian structures for alignment limitation. [5]

6 Conclusion

In this work we focus on proposing a solution to deal with the limitation of 'Less Data' in an efficient and in simple manner with the use of Siamese neural networks and One-Shot learning. We also propose a new dataset for Indian faces. This approach can be extended to find any similar objects, such as signature recognition, X-ray images etc., with the techniques of One-Shot learning, where we need to train only a single or few images of every possible class. Overall, this is an effective yet computationally inexpensive method of dealing facial similarity detection.

7 Acknowledgements

We would like to thank the teachers, friends and family who helped us greatly in the process of creating the dataset. We would also extend our thanks to IIT Kanpur, for providing a part of the dataset. We extend out hearty thanks to the authors of the Research Papers we have referred for their valuable insights, making our work a lot easier.

References

[1] Gregory Koch. *Siamese Neural Networks for One-Shot Image Recognition.* University of Toronto, 2015.

[2] Camilo Vargas, Qianni Zhang, and Ebroul Izquierdo. *One Shot Logo Recognition Based on Siamese Neural Networks.* Proceedings of the 2020 International Conference on Multimedia Retrieval. Association for Computing Machinery, New York, NY, USA, 321–325. DOI: https://doi.org/10.1145/3372278.3390734, 2020.

[3]. Nehal K. Ahmed, Elsayed E. Hemayed, Magda B. Fayek. *Hybrid Siamese Network for Unconstrained Face Verification and Clustering under Limited Resources*, Cairo University, Giza 12613, Egypt, 2020.

[4] Mohsen Heidari, Kazim Fouladi-Ghaleh. *Using Siamese Networks with Transfer Learning for Face Recognition on Small-Samples Datasets*. University of Tehran, Iran, 2020.

[5] Soumava Kumar Roy, Mehrtash Harandi. Richard Nock, Richard Hartley. *Siamese Networks: The Tale of Two Manifolds,* 2019.

[6] Raia Hadsell, Sumit Chopra, Yann LeCun. *Dimensionality Reduction by Learning an Invariant Mapping,* New York University, 2005.

[7] Roman Pflugfelder. *An In-Depth Analysis of Visual Tracking with Siamese Neural Networks,* 2018.

[8] A. Nandy, S. Haldar, S. Banerjee and S. Mitra, *A Survey on Applications of Siamese Neural Networks in Computer Vision*, International Conference for Emerging Technology (INCET), Belgaum, India, 2020.
[9] Koch, Gregory R. *Siamese Neural Networks for One-Shot Image Recognition*, (2015).

[10] Shou-Ching Hsiao, Da-Yu Kao, Zi-Yuan Liu, Raylin Tso. *Malware Image Classification Using One-Shot Learning with Siamese Networks*, Procedia Computer Science, Volume 159, 2019, Pages 1863-1871.

[11] H. Wu, Z. Xu, J. Zhang, W. Yan and X. Ma. *Face recognition based on convolution siamese networks*, 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Shanghai, 2017.

[12] Mohammad Shorfuzzaman, M. Shamim Hossain, MetaCOVID. *A Siamese neural network framework with contrastive loss for n-shot diagnosis of COVID-19 patients*, Pattern Recognition, 2020.