Turkish Journal of Computer and Mathematics Education Volume 10, Issue 3 (2019), 1609-1621
Research Article

Automated Facial Recognition in Older Photographs Using One-Shot Learning in Siamese Networks and Transfer Learning

Viratkumar K. Kothari[1] and Dr Sanjay M. Shah[2]

[1]Ph.D. Scholar, Kadi Sarva Vishwavidyalaya, Gandhinagar, Gujarat
[2]Director, Narsinhbhai Institute of Computer Studies & Management, Kadi, Gujarat

Article History: Article Received: *28th June 2019* ***Accepted:*** *16th October 2019* ***Published Online:*** *14th November 2019*

Abstract: A lot of historical information comes in various forms, such as old documents, papers, photographs, videos, audio, and even artefacts and sculptures. Photographs, audio, and videos are especially important because they effectively convey information. When we convert these into digital versions, it becomes easy to share, access online or offline, copy, move around, back up, and store in numerous places. However, a challenge with digital content is that it is often difficult to search due to the absence of readable text. Consequently, we cannot effectively analyse and utilise critical information. To make it useful, we manually look at pictures and add tags to create labels. While basic labels suffice for most searches, it becomes more complicated when dealing with a large number of photographs. Enhancements in search capabilities are needed to make the process easier, quicker, and more efficient. Fortunately, recent technological advances, such as artificial intelligence, provide us with facilities to simplify this process. This paper explores how artificial intelligence can streamline this process, enhancing search efficiency and enabling automatic identification and tagging of individuals in photos, thus facilitating easier access and analysis of digital archives.
It is anticipated that manual tagging efforts could be reduced by approximately 80%, and the searchability of photographs could be enhanced by about 84%.

Keywords*:* Face Recognition, Face Identification, Image Processing, Face Verification, One-shot Learning, FaceNet, Convolutional Neural Network, Image Classification, Image Processing, Machine Learning, Computer Vision, Siamese Networks, Artificial Intelligence

I. INTRODUCTION

Facial recognition in older photographs presents a significant challenge due to factors such as image degradation and changes in appearance over time. Traditional facial recognition systems often struggle to perform accurately in such scenarios, especially when limited training data is available. Our study introduces a unique method that employs one-shot learning within Siamese network frameworks in the field of computer vision aimed at enhancing the facial recognition process in historical photographs. The Siamese network architecture enables effective feature extraction and comparison, while one-shot learning minimises the need for large annotated datasets. Experimental results on a synthetic dataset demonstrate the efficacy of the proposed method in accurately identifying individuals in older photographs, showcasing its potential in various domains such as forensic investigations, historical research, and genealogy.

Upon conversion to digital format, physical photographs offer several advantages. However, they inherently exhibit limited searchability and are heavily dependent on metadata. Therefore, enhancing metadata can significantly improve searchability and facilitate seamless linking of photographs according to various criteria, including people in the photographs, locations, and events. Technology such as artificial intelligence show promise in this.

The experiments are set to be applied to improve the digital archives related to Mahatma Gandhi on the Gandhi Heritage Portal (www.gandhiheritageportal.org), which houses a wide range of digitised materials, such as letters, books, and audiovisual content. Despite the richness of the collection, the portal's search capabilities are hampered by metadata limitations, often derived from manually tagged, poor-quality images by experts whose availability is scarce. Using a Face Recognition Dataset, sourced from the comprehensive Labeled Faces in the Wild Dataset, will help streamline this process by employing AI for more accurate identification and metadata creation, thereby enhancing the portal's usability and accessibility for research and educational purposes.

Automated classification of photographs using various machine learning techniques typically requires a large number of images to enhance accuracy. However, with older photographs, there is often a very limited number available, leading to lower accuracy in automated image classification. Therefore, a machine learning technique that requires fewer samples for training is necessary. The Siamese Network is a technique that can achieve good accuracy with fewer sample photographs.

Facial recognition technology is widely used in areas such as security and interactive systems, but it faces challenges with accurately identifying individuals in aged or degraded photographs. Conventional methods, which typically require extensive labelled data, falter under these conditions. This paper proposes a new method utilising Siamese networks coupled with one-shot learning to overcome these obstacles, offering a promising solution for recognising faces in historical images.

This paper is organised as follows:

Section I: Introduction, Section II: Related Work, Section III: Information about the digital data, image recognition architecture, and approach, Section IV: Environmental Setup, Section V: Results and Observations of the Digital Platform and Section VI: Conclusion and Future Work.

II. RELATED WORK

Alex Krizhevsky et al. [1] have introduced deep convolutional neural networks (CNNs) for image classification, achieving a significant breakthrough in accuracy on the ImageNet dataset. The proposed model utilised multiple layers of convolutional and pooling operations and non-linear activation functions to learn hierarchical features from raw image data. Karen Simonyan et al. [2] pioneered the VGG network architecture, which comprises stacked convolutional layers featuring small 3x3 filters. Despite its simplicity, VGG achieved state-of-the-art results on the ImageNet dataset by focusing on depth in the network architecture.

Shaoqing Ren et al. [3] introduced Faster R-CNN – a unified framework for object detection, combining region proposal generation and object detection into a single network. By utilising region proposal networks (RPNs) to generate region proposals, Faster R-CNN achieved real-time performance on object detection tasks while maintaining high accuracy. Joseph Redmon et al. [4] extended the YOLO (You Only Look Once) object detection framework to handle a large number of object classes simultaneously. YOLO9000 improved accuracy and detection performance across a wide range of object categories by incorporating hierarchical classification.

Gregory Koch et al. [5] introduced Siamese neural networks for one-shot image recognition tasks, where the network learns to distinguish between pairs of images by embedding them into a common feature space. Siamese networks achieved impressive performance on tasks requiring only a single example for training, demonstrating their effectiveness in learning discriminative features from limited data. Kaiming He et al. [6] introduced ResNet – a residual learning block to address the vanishing gradients in deep neural networks. By using shortcut connections to skip layers, ResNet allowed the training of very deep networks (VDN) with hundreds of layers, achieving state-of-the-art accuracy on image classification benchmarks such as ImageNet. Kaiming He et al. [7] extended Faster R-CNN to Mask R-CNN, the object detection framework to perform instance segmentation, enabling accurate pixel-level segmentation of objects in images. By predicting segmentation masks alongside bounding boxes and class labels, Mask R-CNN achieved state-of-the-art performance in instance segmentation tasks.

Christian Szegedy et al. [8] introduced the Inception-v4 and Inception-ResNet architectures, which combined the benefits of Inception modules with residual connections. By leveraging residual connections, the proposed architectures achieved improved training convergence and accuracy on image classification tasks compared to previous Inception models. Gao Huang et al. [9] introduced DenseNet, a densely connected convolutional network, where each layer receives input from all preceding layers and passes its feature maps to all subsequent layers. DenseNet achieved state-of-the-art performance on image classification benchmarks by fostering feature reuse and encouraging feature propagation while maintaining parameter efficiency.

Jifeng Dai et al. [10] proposed R-FCN, a region-based fully convolutional network (R-FCN) for object detection, which shared convolutional features across the entire image. By leveraging position-sensitive score maps for object localisation, R-FCN achieved high accuracy on object detection tasks with improved computational efficiency compared to previous region-based approaches. Andrew G. Howard et al. [11] introduced MobileNets – a family of efficient convolutional neural networks (CNNs) optimised for mobile and embedded vision applications. By utilising depthwise separable convolutions and model parameterisation techniques, MobileNets achieved state-of-the-art accuracy with significantly fewer parameters and computational resources compared to traditional CNN architectures.

Yunpeng Chen et al. [12] introduced Dual Path Networks (DPN), a novel architecture with dual paths within each residual unit, enabling information flow through two paths with different transformations. By promoting feature diversity and interaction, DPN achieved improved accuracy and generalisation on image classification tasks, surpassing the performance of conventional ResNet models. Forrest N. Iandola et al. [13] proposed SqueezeNet, a highly efficient convolutional neural network (CNN) architecture with significantly fewer parameters compared to traditional models like AlexNet. By using a combination of 1x1 convolutions, aggressive down-sampling, and fire

modules, SqueezeNet achieved comparable accuracy to larger networks while being suitable for deployment on resource-constrained devices.

Hyeonwoo Noh et al. [14] proposed a novel deconvolution network architecture for semantic segmentation tasks, which learned to predict pixel-wise class labels from input images. The proposed network achieved accurate segmentation results by combining encoder and decoder modules with skip connections while preserving spatial information and reducing checkerboard artefacts commonly seen in deconvolution-based approaches. Liangming Pan et al. [15] introduced a Siamese network approach for knowledge graph embedding, where the network learns to compare entities based on their relationships in the knowledge graph. By embedding entities into a common feature space, Siamese networks enabled efficient comparison and retrieval of related entities, facilitating knowledge graph-based applications such as entity matching and recommendation.

Prior research in facial recognition has primarily focused on improving accuracy in controlled environments with high-quality images. While these methods have achieved significant success, their performance degrades when applied to older photographs due to factors such as image degradation and changes in appearance. Few studies have explored techniques tailored explicitly for handling degraded or historical images. Siamese networks have proven to be an effective method for one-shot learning, efficiently performing feature extraction and comparisons using minimal training data.

III. INFORMATION ABOUT THE DIGITAL DATA, FACE RECOGNITION ARCHITECTURE AND APPROACH

A. The Dataset

The Gandhi Heritage Portal (www.gandhiheritageportal.org) is an important digital archive offering extensive digitised resources for scholarly exploration of Mahatma Gandhi's life and legacy. This comprehensive online repository boasts diverse digital assets, including photographs, audio clips, videos, and textual documents. Among its offerings, the portal features an extensive collection of photographs that are integral to historical research and analysis. However, these photographs pose specific challenges that affect their use in research. Being historical, the photographs are in black and white, vary in quality, and come in different sizes and resolutions. Such variations can significantly impact the clarity and detail visible in the images.

Various Machine Learning (ML) or Artificial Intelligence (AI) models for facial recognition could be employed here, but a critical limitation within this context is the limited number of photographs available for any given individual. Unlike contemporary datasets, where thousands of images might be available to train a model, the Gandhi Heritage Portal's collection, substantial as it is, offers a relatively sparse number of images per individual. This scarcity of data points hampers the development of robust, accurate models capable of navigating the nuanced task of identifying historical figures in archival photographs. In the domain of digital archival research, these challenges necessitate innovative approaches to digital archiving, image analysis, and the application of artificial intelligence. Exploring solutions to overcome these hurdles is crucial for enhancing access to and engagement with historical photographic archives, thereby enriching our understanding.

B. Face Classification and Face Recognition

In machine learning, face classification and face recognition are two related but distinct tasks that involve recognising and analysing faces in images or videos. These tasks leverage computer vision and deep learning techniques to interpret the content of an image or sequence of images.

Face Classification

Face classification is a process that involves categorising faces into one or more predefined categories or classes. This can be based on various attributes such as emotion (happy, sad, angry), age (child, adult, senior), or even specific characteristics like wearing glasses or not. In a broader sense, face classification can also refer to determining whether a face belongs to a predefined group, such as distinguishing between employees and visitors in a corporate setting. The key aspect of face classification is that it categorises faces into general groups rather than identifying individual identities.

Face recognition

Face recognition, on the other hand, is a process that seeks to identify the specific individual to whom a face belongs. This task compares one or more faces against a database of known individuals to find a match. Essentially, face recognition asks the question, "Who is this person?" and attempts to find the individual's identity by comparing their facial features against a database of faces. This process involves extracting unique features from the face and using them as a basis for comparison. It's widely used in security systems, smartphone unlocking, and in identifying individuals in photographs and videos.

Computer vision involves a variety of tasks that allow computers to analyse and comprehend visual data from images and videos:

1. Image Classification: This process involves the computer categorising images into predefined groups. For example, an algorithm might be trained to sort images into categories such as "mountains" or "oceans."
2. Localisation: Beyond simply classifying images, localisation aims to pinpoint the exact position of an object within an image. For instance, if a photo contains a cat, the system not only identifies the presence of a cat but also outlines its location within the image, often by drawing a bounding box around it.
3. Object Detection: This task extends to identifying multiple objects within an image. For example, the system identifies each object in an image featuring a cat, a dog, and a bike. It highlights them, usually with bounding boxes, but does not specify what each object is.
4. Object recognition: Going a step further, object recognition involves recognising and naming the different objects in an image. If an image includes a dog, the system not only detects its presence but also labels it as a "dog," effectively combining detection and classification.
5. Instance Segmentation: This sophisticated process involves identifying objects in an image and precisely outlining their shapes instead of merely placing bounding boxes around them. This technique provides a detailed boundary that follows the exact contours of each object, offering a clear distinction between different objects and their backgrounds.

Each of these tasks represents a layer of complexity in how machines can analyse and understand visual data, moving from simple categorisation to detailed, nuanced recognition and analysis. The scope of this research paper is to identify faces from images.

C. Siamese Network for Face Recognition

A Siamese Network is a specialised architecture in machine learning used for tasks like face recognition, among others. It's unique because it focuses on learning the similarity between two comparable inputs rather than classifying inputs into predefined categories. This makes it particularly effective for face recognition, where the goal is often to determine whether two face images represent the same person.

Architecture

The architecture of a Siamese Network consists of two identical subnetworks joined at their outputs. These subnetworks are "siamese twins" — hence the name — because they share the same parameters and weights during training. Each subnetwork processes one of the two input images. The key idea is that these subnetworks learn to get feature vectors from the images, which represent the essential characteristics of the faces.

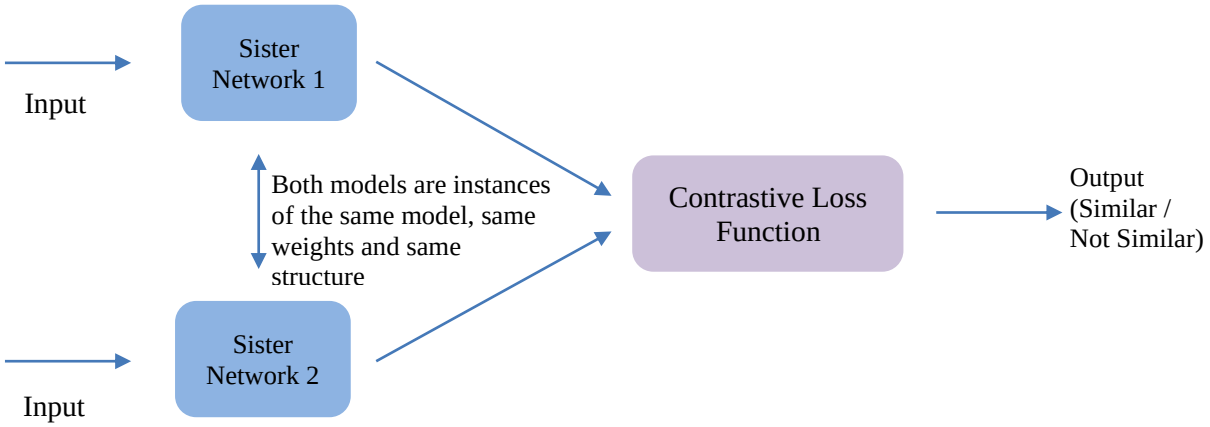

Figure 1: Simple Architecture of Siamese Network

Machine learning excels in tasks like web search, fraud detection, and speech and image recognition but struggles with limited data for supervised learning. One particularly challenging task is image classification under the constraint of limited data size, i.e., we might have only one example of each possible class before making a prediction. This is known as one-shot learning, where the algorithm looks at an image only once and learns features to distinguish it from the rest. One-shot learning is particularly useful in facial recognition challenges. This approach is widely used across several sectors.

One-shot learning has been effectively realised through Siamese networks, which are a special type of neural network. Unlike traditional models that learn to classify inputs, a Siamese network learns to recognise the similarity between two inputs and then differentiates them. The Siamese neural network considers the distance or

difference in similarity, essentially measuring how distinct the inputs are from each other. In simple words, a Siamese network features two closely related or identical neural networks, often referred to as sister networks. Each one processes a different input image. The outputs from the final layers of these sister networks are fed into a contrastive loss function. This function measures how similar or different the two images are.

D. Properties of the model

The main goal of the Siamese neural network is to distinguish between input images rather than classify them. It uses a contrastive loss function to assess how effectively the sister networks can distinguish a specific pair of images.

The process begins with feeding an image into the first sister network, which then passes through several layers designed for feature extraction—such as convolution, pooling, and fully connected layers—resulting in a feature vector f(x1). This vector f(x1) represents the encoded version of the input image (x1). Following this, the second operation involves inputting into the second sister network, which is identical to the first one, to obtain another encoded version f(x2) of a different input image (x2).

Next, we measure the distance d between the two encodings f(x1) and f(x2). If this distance d falls below a certain threshold (a hyperparameter), it suggests that the two images represent the same person. Otherwise, the images are of different individuals. Through this method, the machine effectively distinguishes between different faces. The following image will help you understand this.

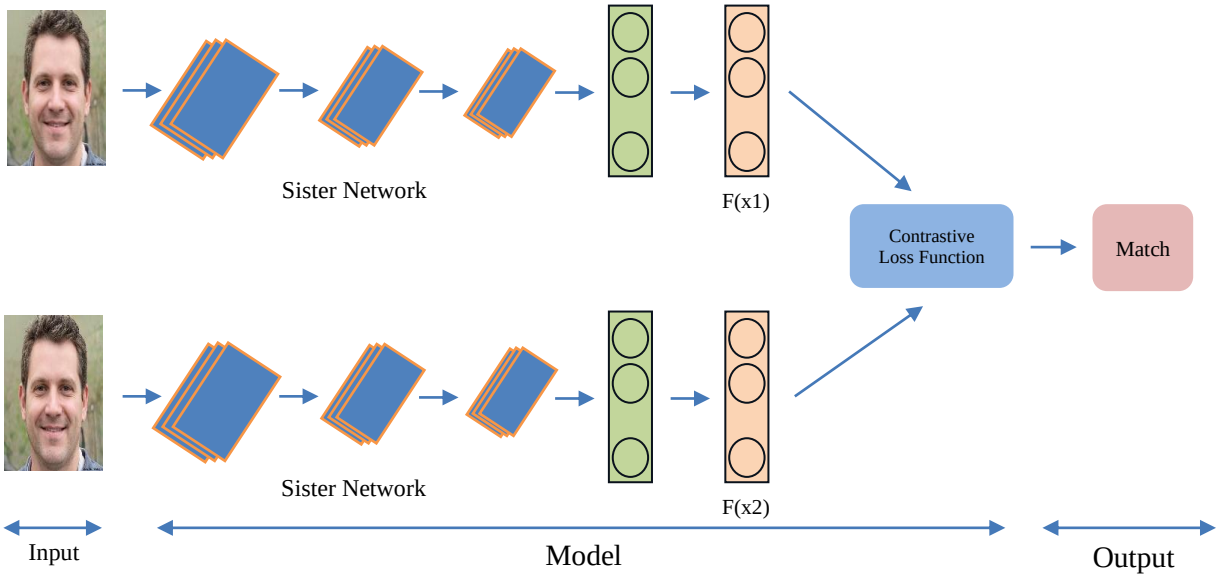

Figure 2: Image Processing in Siamese Network

The distance between two encodings is calculated as follows:

The distance d(x1, x2) is equal to the Euclidean Distance between the features f (x1), f(x2).

When x(i) and x(j) signify the same person, the distance d(x(i),x(j)) is minimal. Conversely, if x(i) and x(j) depict different individuals, then d(x(i),x(j)) is significantly greater.

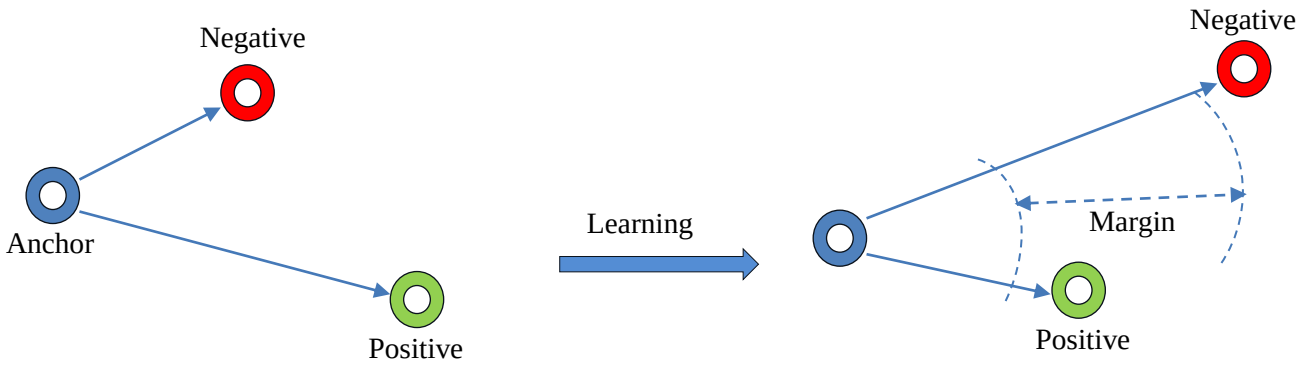

Figure 3: Find the similarity or difference between two photographs

To enhance the quality of image encoding, you can optimise the parameters by applying gradient descent on a triplet loss function, commonly known as contrastive loss. This involves calculating the loss using three images: an anchor, a positive (which is the same as the anchor), and a negative image (which is different).

Since the positive and the anchor images are identical, the distance d(Anchor, Positive) between their encodings will be smaller than the distance d(Anchor, Positive) between the encoding of the anchor and the negative image.

The distance between the encodings of the anchor and the positive image, d(f(Anchor),f(Positive)), should be minimal, whereas the distance between the encodings of the anchor and the negative image, d(f(Anchor),f(Negative)), should be significantly larger, fulfilling the condition:

$d(f(\text{Anchor}), f(\text{Positive})) < d(f(\text{Anchor}), f(\text{Negative}))$

The loss function is defined as:

$L(\text{Anchor}, \text{Positive}, \text{Negative}) = \max(||f(\text{Anchor}) - f(\text{Positive})||2 - ||f(\text{Anchor}) - f(\text{Negative})||2 + \alpha, 0)$

Here, the "max" indicates that the triplet loss function L(Anchor, Positive, Negative) will be zero as long as the expression $[d(\text{Anchor}, \text{Positive}) - d(\text{Anchor}, \text{Negative}) + \alpha]$ is less than or equal to zero. If this expression is greater than zero, indicating a need for adjustment, the loss will be positive, and the function will aim to minimise this loss towards zero.

Finally, the cost function (J), which represents the sum of all individual losses from different triplets across the entire training set, is calculated as:

$J = \sum L(\text{Anchor}(i), \text{Positive}(i), \text{Negative}(i))$

This sum provides a comprehensive measure of the model's performance over all training examples, guiding the optimisation process to improve face recognition accuracy.

The Siamese Network offers various benefits:

1) Effectiveness with Small Datasets: Siamese Networks can be effective even with smaller datasets because they learn from pairs of images, effectively augmenting the amount of training data.
2) Robustness to Class Imbalance: They are not trained to classify inputs into fixed categories, making them less susceptible to issues like class imbalance.
3) Versatility: While especially known for face recognition, Siamese Networks are also used in other applications requiring similarity comparisons, such as signature verification and anomaly detection.

Siamese Networks represent a powerful approach to learning fine-grained similarity measures between images, making them a cornerstone technique for face recognition and beyond.

E. Customisation in the layers of the model

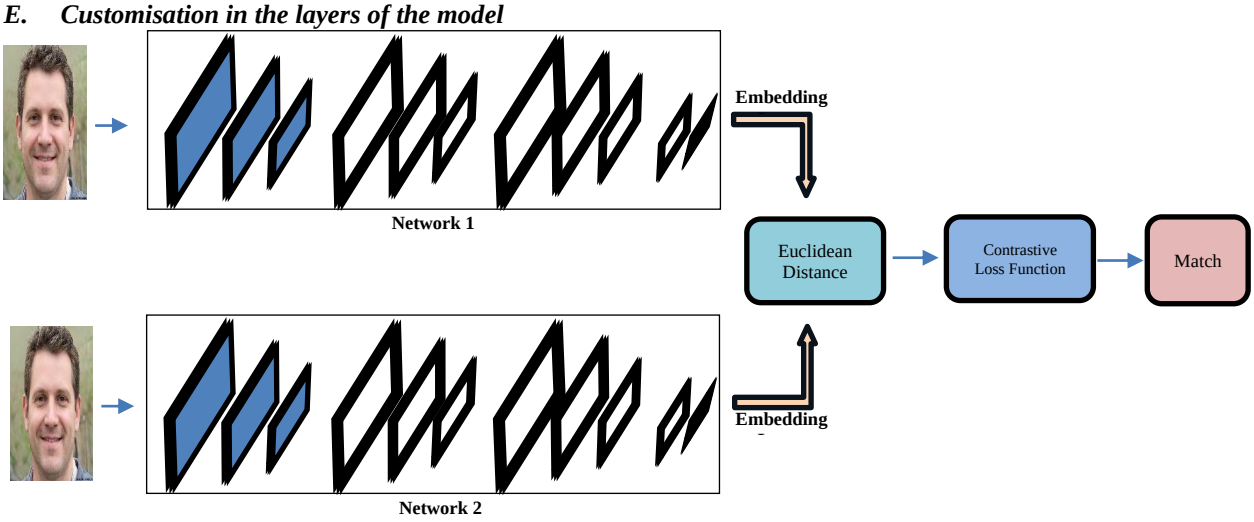

Figure 4: Implementation of the model

In contrast to traditional CNNs that categorise images into specific groups or labels, the Siamese Network focuses on measuring the distance between pairs of images. Its goal is to adjust its internal parameters—weights and biases—to reflect the relationship between the images. The network aims to minimise the distance between images that are similar or share the same label, indicating their similarity. Conversely, images with different labels aim to maximise the distance, signalling their dissimilarity.

- The process begins by selecting a pair of images from the dataset for processing by the parallel networks.
- These networks are identically structured, applying the same processing tasks to each image.
- Towards their conclusion, these networks feature Fully Connected Layers, the final of which comprises 128 nodes, producing the Embedding Layer Representation. Consequently, each image within the pair yields its distinct Embedding Layer Representation after processing.
- Subsequently, the network calculates the Euclidean distance between the two embedding layers. Identical individuals across the images will likely result in closely matching embeddings and a reduced distance. In contrast, images representing different individuals are anticipated to exhibit a greater distance.
- To standardise the distance values, they are subjected to a Sigmoid Function, adjusting the range to between 0 and 1.
- The sigmoid's outcome is then evaluated using a loss function, encouraging the network to refine its weights and biases. For this purpose, Binary Cross Entropy is utilised as the loss function, with both networks undergoing identical adjustments to their weights and biases.

F. Customisation in the layers of the model

The default Xception model cannot be used straight forward for the custom problem of image recognition.

Figure 5: Default Xception architecture

The first layer – the input layer, is designed to receive images for processing within the model. The default input image size for the Xception model is 299 x 299 pixels (Keras), with the '3' indicating that the images are expected to be coloured, making the complete input size 299 x 299 x 3. After this layer, the images undergo processing through a series of convolutional and max-pooling layers.

The Encoder plays a crucial role in transforming images into feature vectors. We will use a pre-trained model, the Xception model, which is an advancement of the Inception_V3 model. Transfer learning helps significantly reduce both training time and the volume of the dataset.

1) We will incorporate the Xception model's pre-trained weights into our new model.
2) We will change the default input layer to accept images of the size 128 x 128.

3) *include_top=False:* this means that the top layer or the fully connected layers at the end of the network, typically used for classification, will not be included. This is useful for feature extraction in transfer learning scenarios where we want to add custom layers for specific tasks.

4) *pooling='avg':* this applies global average pooling to the output of the last convolutional block. This means that for each feature map coming out of the previous convolutional block, it calculates the average of all values, reducing the spatial dimensions (height and width) to a single scalar average per feature map. This operation effectively converts the 2D feature maps into a 1D feature vector, which can be beneficial for reducing model complexity and computational cost in tasks that require feature extraction.

5) We will make all the layers *trainable = fales* except the layer 27 using the following code.

```
for i in range(len(pretrained_model.layers)-27):
    pretrained_model.layers[i].trainable = False
```

Doing this freezes the weights and biases of these layers, meaning they will not be updated during further training. This is a common practice in transfer learning, where the general features learned by the model are preserved, while only the final layers are trained to adapt to a new task.

1) Parameter tuning of the model

Following the encoding process, the model is connected to Fully Connected (Dense) layers, culminating in a final layer that employs L2 Normalization. L2 Normalization adjusts the dataset so that the sum of the squares of the values in each row equals 1, effectively standardising the data for further processing.

The following parameters will be tuned in order to get better performance from Xception:

1) *weights='imagenet':* this is to use weights from a pre-trained model

2) input_tensor=input_layer: this is to add a custom input layer as follows:

```
input_layer=layers.Input(shape=(128,128,3))
```

3) The following set of custom layers may be added to improve the performance:

```
encode_model = Sequential([
        pretrained_model,
        layers.Flatten(),
        layers.Dense(512, activation='relu'),
        layers.BatchNormalization(),
        layers.Dense(256, activation="relu"),
        layers.Lambda(lambda x: tf.math.l2_normalize(x, axis=1))
 ], name="Encode_Model")
```

It starts with a pretrained_model as the base for feature extraction, followed by a Flatten layer to convert the features into a 1D array. Then, it adds two Dense layers with 512 and 256 neurons, respectively, both using ReLU activation for non-linear transformation. A BatchNormalization layer is included for stabilising learning by normalising the input layer by re-centering and re-scaling. The final Lambda layer applies L2 normalisation to the output, ensuring feature vectors have a unit length.

IV. ENVIRONMENTAL SETUP

We will use the following to test the model:

1) Jupyter Notebook in Kaggle environment (default)

2) 4 CPU Cores

3) GPU: 1 Nvidia Tesla P100 GPU (default)

4) RAM: 29 Gigabytes of RAM

5) TensorFlow: 2.4.1 or above

6) Image dataset: We are utilising the Face Recognition Dataset, a carefully selected compilation of JPEG celebrity images designed to enhance face detection and recognition technology. This dataset, derived from the Labeled Faces in the Wild Dataset (LFW), includes a variety of photos found online. Further information and dataset details can be accessed on its official website: http://vis-www.cs.umass.edu/lfw/. The following is the more information on it:

a. Each photo prominently features a single face against an RGB-encoded background, originally sized at 250 x 250 pixels.

b. It comprises 1680 folders, each dedicated to a different celebrity, with 2 to 50 photos per folder.

c. The dataset comprises a total of 13,233 images featuring 5,749 individuals, among whom 1,680 have two or more images.

d. Faces were extracted from these photos using the Haar-Cascade Classifier in OpenCV and resized to 128 x 128 pixels, maintaining RGB encoding.

e. The length of the training list is 1191, and the length of the testing list is 133.

f. We utilise training and testing lists to assemble triplets consisting of an anchor, a positive (the same person as the anchor), and a negative (a different person from the anchor) for face data.

V. RESULTS AND OBSERVATIONS

In this section, the observations are presented based on tests. The Siamese Network model looks like the following:

```
Model: "Siamese_Network"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
Anchor_Input (InputLayer)       [(None, 128, 128, 3) 0
__________________________________________________________________________________________________
Positive_Input (InputLayer)     [(None, 128, 128, 3) 0
__________________________________________________________________________________________________
Negative_Input (InputLayer)     [(None, 128, 128, 3) 0
__________________________________________________________________________________________________
Encode_Model (Sequential)       (None, 256)          22043944    Anchor_Input[0][0]
                                                                 Positive_Input[0][0]
                                                                 Negative_Input[0][0]
__________________________________________________________________________________________________
distance_layer (DistanceLayer)  ((None,), (None,))   0           Encode_Model[3][0]
                                                                 Encode_Model[4][0]
                                                                 Encode_Model[5][0]
==================================================================================================
Total params: 22,043,944
Trainable params: 9,583,800
Non-trainable params: 12,460,144
```

Figure 6: Default Xception architecture

This output describes the architecture of a "Siamese_Network" model designed for learning from image triplets (anchor, positive, negative), each with a size of 128x128 pixels across 3 colour channels. The model uses a shared "Encode_Model" to transform these images into 256-dimensional feature vectors. The "DistanceLayer" then computes distances between the encoded features of the anchor, negative, and positive images. The model has a total of 22,043,944 parameters, with 9,583,800 of them being trainable and 12,460,144 non-trainable. This setup is typical for tasks that require learning from relative comparisons between different images to improve similarity measures.

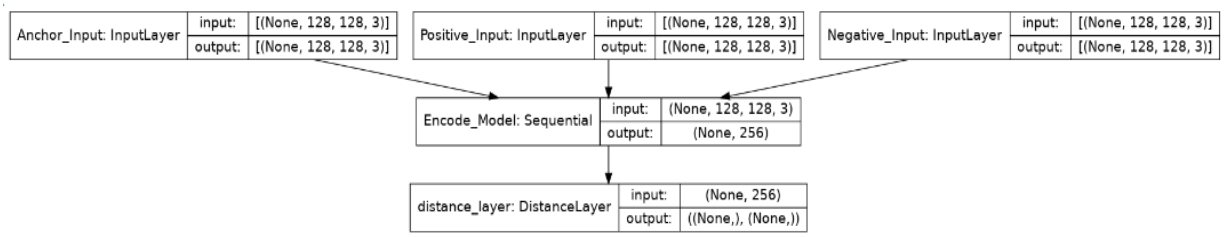

Figure 7: Architecture of a Siamese Network built for face recognition

The chart illustrates the architecture of a Siamese Network built for face recognition. It comprises three input layers (Anchor, Positive, and Negative), each taking an image of size 128x128x3, which is the standard format for colour images. These inputs are then fed into a shared 'Encode_Model' (a Sequential model), which processes the images to generate a 256-dimensional feature vector (embedding) for each. Finally, a 'DistanceLayer' computes the distances between the embeddings of the anchor-positive and anchor-negative image pairs. The model outputs

these distances, which are used to measure how similar or dissimilar the images are, with the expectation that distances for images of the same person (anchor-positive) will be smaller than those of different people (anchor-negative).

We will train the siamese_model using batches of triplets, monitoring training loss and evaluating additional metrics from tests after each epoch. Whenever the model surpasses its previous maximum accuracy, we'll save its weights. Our aim is to gather more metrics to find ways to enhance the model's accuracy further. The number of epochs has been configured to stay within Kaggle's time limits. We are keeping the batch size of 128 for the experiment. The model will gather metrics such as accuracy, means, and standard deviations by making predictions on the training data.

We will analyse accuracy after a few epochs until we get the best accuracy. The following are the observations.

Combination - 1:

Sr. No.	Processor	No. of added layers	Activation	Dence Value	Epoch	Batch Size	Loss in Train	Accuracy	Interpretation
1	1 Nvidia Tesla P100	1D (512) + 1BN + 1D(256) + 1L	Relu	100	1	128	0.5019	0.8664	Poor accuracy

We added the following combination of layers, but it gives poor accuracy.

layers.Flatten(),
layers.Dense(512, activation='relu'),
layers.BatchNormalization(),
layers.Dense(256, activation= "relu"),
layers.Lambda(lambda x: tf.math.l2_normalize(x, axis=1))

Combination - 2:

Sr. No.	Processor	No. of added layers	Activation	Dence Value	Epoch	Batch Size	Loss in Train	Accuracy	Interpretation
1	1 Nvidia Tesla P100	1D (512) + 1BN + 1L	Relu	100	50	128	0.0009	0.92955	Accuracy Improved but still there is a loss

We have added the following combination of layers, which gives better accuracy.

layers.Flatten(),
layers.Dense(512, activation='relu'),
layers.BatchNormalization(),
layers.Lambda(lambda x: tf.math.l2_normalize(x, axis=1))

Combination - 3:

Sr. No.	Processor	No. of added layers	Activation	Dence Value	Epoch	Batch Size	Loss in Train	Accuracy	Interpretation
1	1 Nvidia Tesla P100	1BN + 1D(256) + 1L	Relu	100	100	128	0.0004	0.92632	Accuracy Improved but still there is a loss

We have added the following combination of layers, which improved the accuracy.

layers.Flatten(),
layers.BatchNormalization(),
layers.Dense(256, activation= "relu"),
layers.Lambda(lambda x: tf.math.l2_normalize(x, axis=1))

Combination - 4:

Sr. No.	Processor	No. of added layers	Activation	Dence Value	Epoch	Batch Size	Loss in Train	Accuracy	Interpretation
1	1 Nvidia Tesla P100	1D (512) + 1BN + 1D(256) + 1L	Relu	100	168	128	0	0.94089	Accuracy Improved but loss cannot be improved further

We have added the following combination of layers, but it gives better accuracy than other combinations of Dense and BatchNormalization layers.

layers.Flatten(),
layers.Dense(512, activation='relu'),
layers.BatchNormalization(),
layers.Dense(256, activation= "relu"),
layers.Lambda(lambda x: tf.math.l2_normalize(x, axis=1))

Combination - 5:

Sr. No.	Processor	No. of added layers	Activation	Dence Value	Epoch	Batch Size	Loss in Train	Accuracy	Interpretation
1	1 Nvidia Tesla P100	1D (512) + 1BN + 1D(256) + 1L	Relu	100	256	128	0	0.94089	Accuracy remained stable now and loss is already 0 so it cannot be improved further

We have added the following combination of layers, but it gives the best accuracy under Seamses Network.

layers.Flatten(),
layers.Dense(512, activation='relu'),
layers.BatchNormalization(),
layers.Dense(256, activation= "relu"),
layers.Lambda(lambda x: tf.math.l2_normalize(x, axis=1))

Let us examine the results in the chart to understand it better:

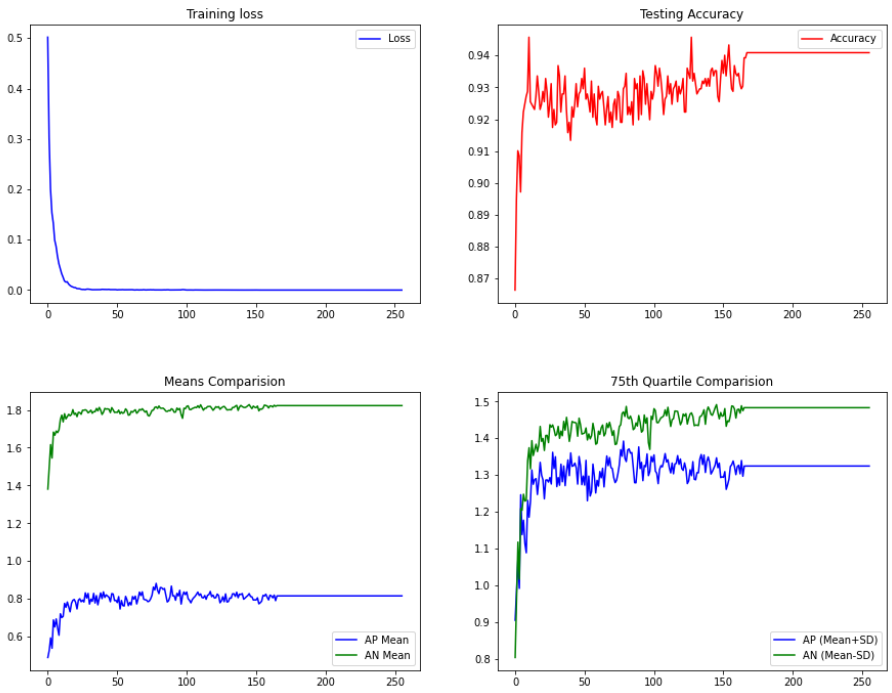

Figure 8: Training loss, testing accuracy along with mean comparison and 75th Quartile comparison

We will use a distance formula to measure the distance between image encodings. If the computed distance exceeds a certain threshold, the images are classified as "different"; if it falls below this threshold, they are considered "same".

VI. CONCLUSIONS AND FUTURE WORK

Following is the summary of the experiment with various combinations of layers added to the model. The best accuracy with very little overfit is available by applying the default Xception model and adding some custom layers.

Sr. No.	Processor	No. of added layers	Activation	Dence Value	Epoch	Batch Size	Loss in Train	Accuracy	Interpretation
1	1 Nvidia Tesla P100	1D (512) + 1BN + 1D(256) + 1L	Relu	100	1	128	0.5019	0.8664	Poor accuracy
2	1 Nvidia Tesla P100	1D (512) + 1BN + 1L	Relu	100	50	128	0.0009	0.92955	Accuracy Improved but still there is a loss
3	1 Nvidia Tesla P100	1BN + 1D(256) + 1L	Relu	100	100	128	0.0004	0.92632	Accuracy Improved but still there is a loss
4	1 Nvidia Tesla P100	1D (512) + 1BN + 1D(256) + 1L	Relu	100	168	128	0	0.94089	Accuracy Improved but loss cannot be improved further
5	1 Nvidia Tesla P100	1D (512) + 1BN + 1D(256) + 1L	Relu	100	256	128	0	0.94089	Accuracy remained stable now and loss is already 0 so it cannot be improved further

Figure 9: Summary of the combination of the layers

The table presents results from experiments using a Siamese network for image recognition with various configurations on a Nvidia Tesla P100 processor. The network architectures differ by the number and sequence of dense (D), batch normalisation (BN), and Lemda (L) layers. Across different batch sizes and epochs, the network shows progressive improvements:

- The first configuration yielded poor accuracy.
- Subsequent adjustments led to improved accuracy despite some loss.
- Further tweaking enhanced accuracy, though some loss persisted.
- Increasing the epoch to 168 achieved better accuracy with zero loss, indicating no further loss minimisation was possible.
- Epoch size of 256 maintained this peak accuracy, with zero loss sustained, suggesting optimal performance under the given conditions.

Now, let us calculate the model's accuracy by 'True Similar', 'False Similar', 'False Different', and 'True Different.' The following confusion matrics show the accuracy that we achieved:

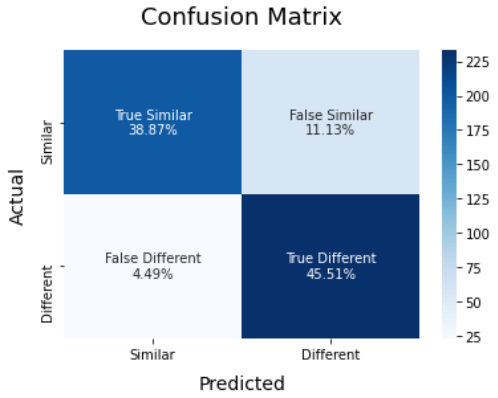

Figure 10: Confusion matrix based on results

This confirms the accuracy of about 84.37%. The approach of using a Siamese Network for face recognition may be extended to recognise one or more people in the old photographs. That will help identify more people and thus create better metadata.

REFERENCES

[1] Alex Krizhevsky et al., "Imagenet classification with deep convolutional neural networks," *Journal of Machine Learning Research,* 2012.

[2] Karen Simonyan et al., "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[3] Shaoqing Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI),* 2015.

[4] Joseph Redmon et al., "YOLO9000: Better, Faster, Stronger," *arXiv preprint arXiv:1612.08242,* 2016.

[5] Gregory Koch et al., "Siamese Neural Networks for One-shot Image Recognition," *Advances in Neural Information Processing Systems (NeurIPS),* 2015.

[6] Kaiming He et al., "Deep Residual Learning for Image Recognition," Proceedings of the *IEEE conference on computer vision and pattern recognition (CVPR),* 2016.

[7] Kaiming He et al., "Mask R-CNN," Proceedings of the *IEEE international conference on computer vision (ICCV),* 2017.

[8] Christian Szegedy et al., "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," Proceedings of the *AAAI Conference on Artificial Intelligence,* 2016.

[9] Gao Huang et al., "DenseNet: Densely Connected Convolutional Networks," Proceedings of the *IEEE conference on computer vision and pattern recognition (CVPR),* 2017.

[10] Jifeng Dai et al., "R-FCN: Object Detection via Region-based Fully Convolutional Networks," *Advances in Neural Information Processing Systems (NeurIPS),* 2016.

[11] Andrew G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861,* 2017.

[12] Yunpeng Chen et al., "Dual Path Networks," Proceedings of the *IEEE international conference on computer vision (ICCV),* 2017.

[13] Forrest N. Iandola et al., "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," *arXiv preprint arXiv:1602.07360,* 2016.

[14] Hyeonwoo Noh et al., "Learning Deconvolution Network for Semantic Segmentation," Proceedings of the *IEEE international conference on computer vision (ICCV),* 2015.

[15] Liangming Pan et al., "Learning to Compare: Siamese Network for Knowledge Graph Embedding*," arXiv preprint arXiv:1702.03814,* 2017.