Project Report

Siamese Neural Networks for One-shot Image Recognition

Author: Rohit Chandra

Problem Description

- In the modern Deep learning era, Neural networks are almost good at every task, such as web search, spam detection, caption generation, and speech and image recognition but these neural networks rely on more data to perform well. But, for certain problems like **face recognition** and **signature verification**, we can't always rely on getting more data.
- We desire to generalize to these unfamiliar categories without necessitating extensive retraining which may be either expensive or impossible due to limited data or in an online prediction setting, such as web retrieval.
- One particularly **interesting task** is classification under the restriction that we may only observe a single example of each possible class before making a prediction about a test instance. This is called **one-shot learning** (Fei-Fei et al., 2006; Lake et al., 2011)
- To solve these kinds of tasks we have a new type of neural network architecture called **Siamese Networks**.
- Siamese neural networks are a type of neural network that are trained to learn a similarity metric between pairs of images. This metric can then be used to classify new images by comparing them to the known images.
- **One-shot** learning is different from **zero-shot** learning in which the model cannot look at any examples from the target classes (Palatucci et al., 2009).
- **Facial recognition technology** has potential benefits including improved security and more efficient identification processes, it is important to carefully consider the risks and potential harms associated with its implementation. Any use of facial recognition systems should be subject to robust **ethical and legal frameworks** to ensure that **individual rights** and **freedoms** are **protected**.

Project Objectives

- The **main goal** of this project is to develop a facial recognition application which is one of the applications of siamese neural network for one-shot learning
- We implement the siamese model from the same architecture which is mentioned in the research paper.
- The objective of the trained siamese model is to correctly classify my own image as verified and any other image of a different person should be classified as not verified in real time
- **Technologies** used: Python, openCV, Tensorflow, Keras, Kivy (front end application)
- Below is the screenshot of the app which is developed

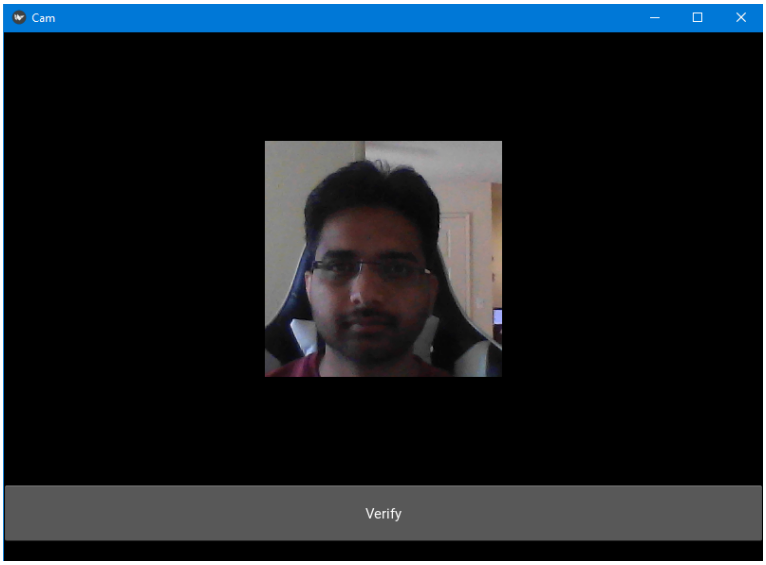

Fig 1. Facial Recognition app

- When the user clicks on the verify button, the siamese model will classify the image as either verified or not verified

Analysis

Step 1: Download and create own dataset to train the siamese neural network

- Download the Labelled Faces in the Wild Dataset and move them to negative folder link: http://vis-www.cs.umass.edu/lfw/#download

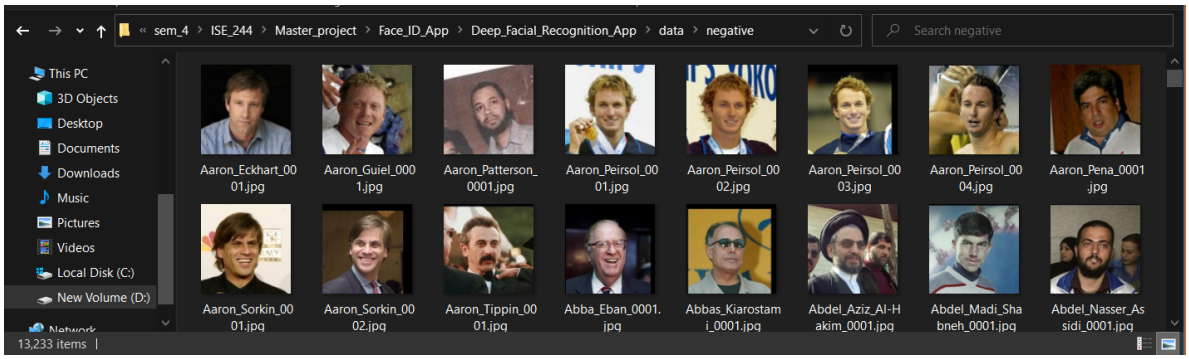

Fig 2. Labelled dataset in negative folder

- It consists of images of famous celebrities all around the world
- Dataset size = **13k** images in total.
- The **resolution** of the images is **100 * 100**
- Created my own dataset by capturing around **5k** images images of myself using **openCV, Python**. These images too have same resolution of **100 * 100**
- **Assumption:** Captured images of myself in different angles of the face with different facial expressions and light so as to simulate the labelled dataset that can aid the model in generalizing better

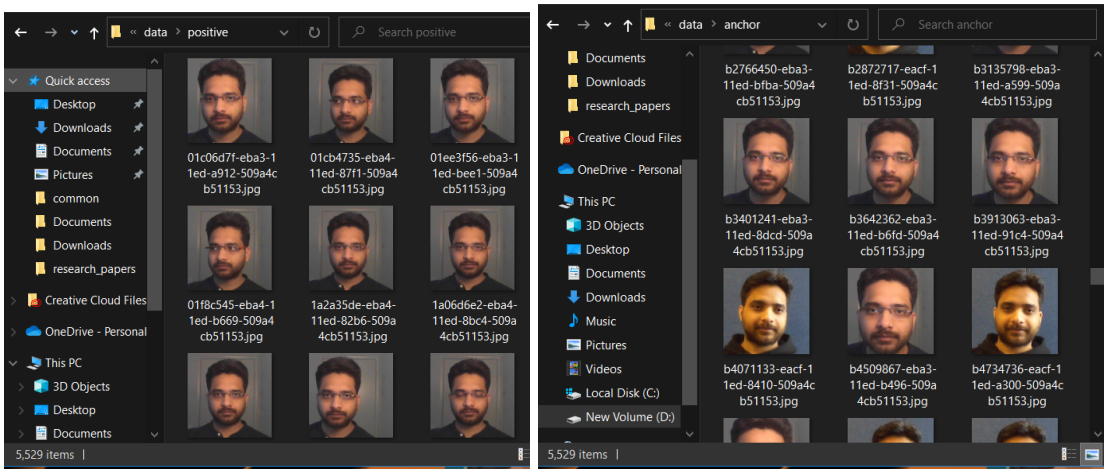

Fig 3. Images of myself in the positive folder and anchor folder

Step 3: Data Preprocessing Pipeline

- **Resize** images to 100 * 100 (in the research paper, the image resolution is 105 * 105)
- **Normalization**: scale all the image values between 0 - 1

- Return a **tuple** of **(anchor/input image, validation image, label**)
- Label 0 = anchor image and validation image are different
- Label 1 = anchor image and validation images are same

Step 4: Based on the of research paper build architecture of embedding model

- Below is the research paper architecture

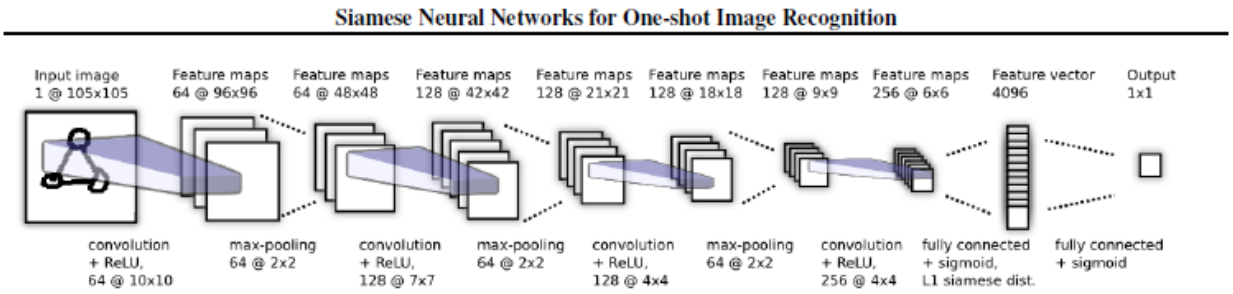

Fig 4. Architecture form the research paper

- Below is the embedding model architecture

```
Model: "embedding"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_image (InputLayer)     [(None, 100, 100, 3)]     0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 91, 91, 64)        19264
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 46, 46, 64)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 40, 40, 128)       401536
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 20, 20, 128)       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 17, 17, 128)       262272
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 9, 9, 128)         0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 6, 6, 256)         524544
_________________________________________________________________
flatten_2 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_2 (Dense)              (None, 4096)              37752832
=================================================================
Total params: 38,960,448
Trainable params: 38,960,448
Non-trainable params: 0
_________________________________________________________________
```

Fig 5. Embedding Model architecture

Step 5: Build L1 distance layer

- Developed a custom L1 layer to calculate L1 distance between two images by subtracting input embeddings from validation embeddings to find the (absolute) distance between them

Step 6: Build siamese model

- Once we get the embeddings from the embedding models for both input/anchor image and validation image, we calculate the the distance between those embedding and then classify the input image as verified or not

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

Fig 6. Siamese neural network model architecture

Step 7: Train the siamese neural network

- Parameters:
  - Loss function : Binary cross entropy
  - Optimizer: Adam
  - Learning rate: 0.0001
  - Epoch: 20
  - Batch Size: 32
- Shuffle and data in the tuple format and then split the tuple dataset: 75% training and 25 % validation set and train the model along with gradient descent to optimize the loss

Results: Evaluate model with performance metrics

Step 8: Evaluate siamese model by comparing the input image with a batch of 32 validation images of myself to make better overall predictions

- On training dataset
  - Recall : 0.98
  - Precision: 0.99

- On Validation Dataset
  - Recall : 1
  - Precision: 1

Step 9: Develop web app using Kivy, Python and openCV and verify images in real time

- Below is the screenshot of of the web application
- In the fig 7., we are verifying image of tom cruise (input image) against 32 images of myself(validation images), so the output should be not verified (label 0)and that's what we get as an output

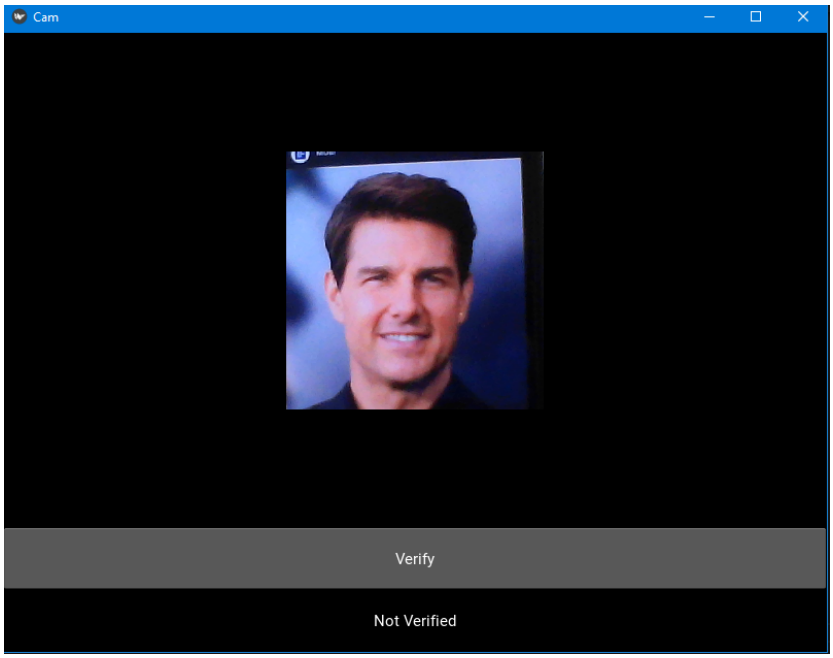

Fig 7. Not verified image

- In the fig 8., we test image of myself (input image) against 32 images of myself (validation image), so the output should be verified (label: 1)

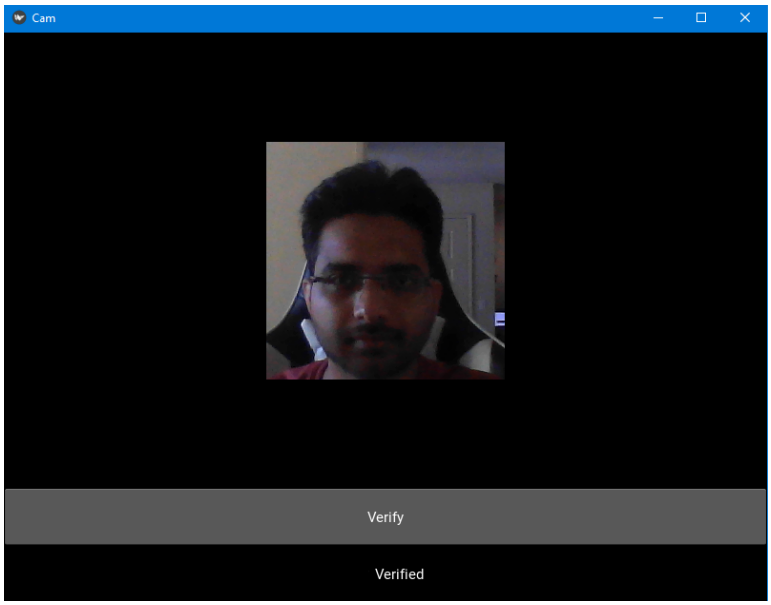

Fig 8. Verified Image

Discussion

- Gained experience of implementing a research paper
- Successfully developed end-to-end facial recognition app which is one of the applications of siamese neural network for one shot image recognition research paper
- Noticed that the model was overfitting because there were few scenarios of false negatives and false positives while testing on unseen data. **Possible solutions** would be to tune the hyperparameters, regularization, and collect more data with a better webcam of the laptop in a well lit environment.
- Since I collected my own images with a low resolution camera on my laptop, I believe that the model's performance is affected.
- Here, is the example of the false negative scenario

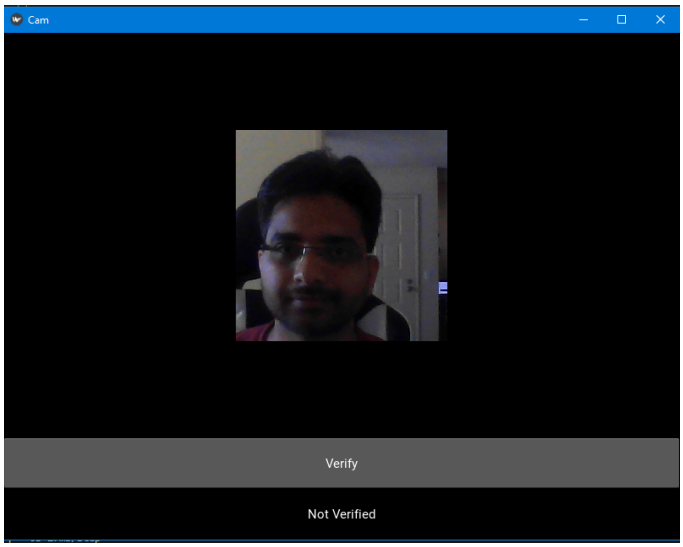

Fig 9. False negative case

Evaluation and Reflection

- Although the performance metrics like precision and recall of the training and validation data is pretty good but given the false positives and false negative cases, the model should be retrained with better resolution images since the current model is overfitting.
- We can also leverage regularization, hyperparameter tuning to reduce the overfitting in the model
- **Assumptions:** Changed the resolution of the images from 105 * 105 which is mentioned in the research paper to 100 * 100 for easier calculations and better understanding. Here, I assumed that this is not have any impact on the model's performance
- Assumed that by collecting and training 5000 images of positive, negative and validation images would be sufficient for the model to correctly classify the images in real time
- Also, assumed that the performance metrics - precision and recall are sufficient to evaluate the model. There's scope of calculating the other performance metrics and then compare the results
- Finally, assumed that the model's performance would be better if I try to verify an input image against a batch of 32 validation images and then take the average of the results to make a final classification
- **Applications**: There are multiple applications of implementing siamese neural networks like face unlock feature on smartphones. We can also implement an automated attendance system for hospitals, schools/colleges

References

- *Fei-Fei, Li, Fergus, Robert, and Perona, Pietro. One-shot learning of object categories. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 28(4):594–611, 2006.*
- *Lake, Brenden M, Salakhutdinov, Ruslan, Gross, Jason, and Tenenbaum, Joshua B. One shot learning of simple visual concepts. In Proceedings of the 33rd Annual Conference of the Cognitive Science Society, volume 172, 2011.*
- *Palatucci, Mark, Pomerleau, Dean, Hinton, Geoffrey E, and Mitchell, Tom M. Zero-shot learning with semantic output codes. In Advances in neural information processing systems, pp. 1410–1418, 2009*