Abstract

Given the current pandemic, masks have become a necessity which calls for an enhancement in the current facial recognition models. Traditional algorithms rely on specific facial features which are now obscured. The main objective of this project is to develop a facial recognition model for masked faces using the triplet loss function. The aim to implement this by using transfer learning models like Inception-ResNet-V1, ResNet-50 and VGG16. Through experimentation, it was observed that Inception-ResNet-V1 produces the most optimized results achieving an accuracy of 87%.

Introduction

Since the beginning of the pandemic, a lot of research has been directed towards finding ways to mitigate the spread of COVID-19, and the most recommended method is to use a face mask. This has inadvertently led to challenges in working on current facial recognition algorithms, such as security checks and face ID. These algorithms work by generating an embedding of facial features. The model is trained. It minimizes the distance between embeddings of different images of the same person and maximizes the distance between images of different people. Since masks cover a significant portion of the face, it makes feature extraction more complicated. The main objective of this project is to build a model capable of recognizing masked faces.

The project is inherently a Facial Recognition task. The main goal is to extend to include functionality for masked faces by training the existing standard models like VGG16, InceptionResNetV2, and ResNet-50 with datasets consisting of masked faces. In addition to retraining the models, a triplet loss function has been implemented, proven to show good results for standard facial recognition systems. The inspiration behind this project and the idea to implement the triplet loss function have been derived from this project [9].

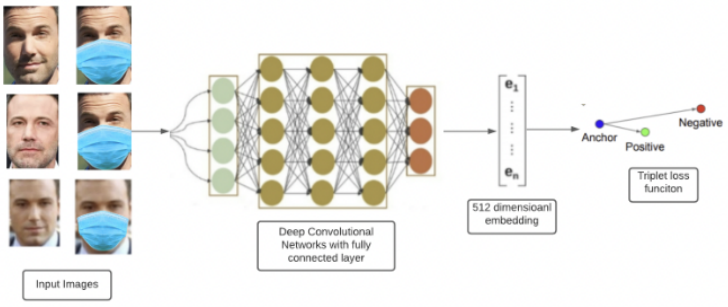

Figure 1. Facial Recognition System using masked and unmasked images

Approach

A supervised learning approach is followed to implement the facial recognition system, as shown in figure 1. Transfer learning models like InceptionResNetV1 [1], ResNet-50 [2] and VGG16 [3] are most commonly used models for this task. This project explores the different transfer learning models. All the images were resized into 224 x 224 x 3 pixels and were fed to the model

through the Data Loader to the convolutional layers for spatial processing. The data is further fine-tuned to fit the network by normalization and data augmentation. Normalizing data enables faster and more stable network training by maintaining a good data distribution. Additionally, this technique also handles the problem of exploding gradients. Data Augmentation was performed using standard image transformations like horizontal flip, vertical flipping, and resizing. Convolutional layers are used as feature extractors that create feature maps. Pooling layers are used to reduce the dimension of the feature maps; by reducing the number of learning parameters of the network. The network classifier then outputs a 512-dimensional image embedding using a fully-connected layer. This embedding represents all the important features of the input image. Triplet Loss function[4] was used to compute the cost of the predictions. To better understand the working of the triplet loss function, the code was borrowed from [7,8], to better understand how to select the anchor, positive, and negative images to feed into the model. Initially, our main approach was to use the Siamese Network, which finds the similarity between two inputs and is used in facial recognition. However, it was observed that the training time is significantly large to obtain comparable results. The project implements convolutional neural networks as a base model to address this limitation.

Triplet Loss

This loss function was introduced by Google's FaceNet [4] for the traditional face recognition task. The main intuition is choosing an anchor, a positive and a negative image as shown in figure 2. The positive image belongs to the same class as the anchor, and the negative image belongs to a different class. This loss function renders similar embedding (small euclidean distance) images belonging to the same person and outputs very different embeddings (larger euclidean distance) for images with different people. This loss function was explored as a masked face recognition system differs from a traditional one simply in the input given to the system. Triplet loss requires a pairing process for the training data; this involves selecting triplet pairs and the frequency of repairing the dataset. There are many different ways the triplet pairs are selected - easy triplet, hard triplet, and semi-hard triplet. A triplet was chosen where the distance between the negative image and the anchor is smaller than the distance between the anchor and the positive image.

The main aim is to make sure that the anchor x_i^a is very close to x_i^p, the positive sample of the same person than it is to any image x_i^n, which is the negative sample. Thus, the expected behavior is as shown in (1):

$$\| f(x_i^a) - f(x_i^p) \|_2^2 + \alpha < \| f(x_i^a) - f(x_i^n) \|_2^2 \qquad (1)$$

$$\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \mathsf{T} \qquad (2)$$

Here, α is the set margin between positive and negative pairs and T is the set of possible triplets that can be considered for the training phase of the model. Thus, the loss function L is minimized as shown in (3),

$$\sum_i^N [\| f(x_i^a) - f(x_i^p) \|_2^2 - \| f(x_i^a) - f(x_i^n) \|_2^2]_+ \qquad (3)$$

Figure 2. Triplet Loss aims to maximize the distance between anchor and negative and minimize the distance between anchor and positive.

Experiments and Result

The implemented facial recognition model uses transfer learning techniques to save training time, exploit these models' features, and use it as a starting point to evaluate our custom dataset. These models are pre-trained on tasks involving rich data, which can be extended to fit any task or application. This leads to great flexibility and improved performance and has emerged as a helpful learning tool for automating procedures. Three different transfer learning models were explored: InceptionResNetV1, ResNet-50, and VGG16. All the work has been implemented on Google Colab's built-in GPU resources and PyTorch framework.

a. Baseline Approach

The project analyzed the performance of VGG16 as a baseline model. This model was pre-trained on ImageNet datasets with an accuracy of 92.7%. However, it was observed that while the model performed well for a few niche cases, it could not generalize well for unseen data. This can be attributed to the fact that VGG16 is more robust for object detection, which is not consistent with our project objective. This model achieved an accuracy of 76.23%.

b. InceptionResNetV1

The PyTorch framework provides a pre-trained InceptionResNetV1 model, which employs a FaceNet architecture trained on the VGG2 dataset, producing 99.65% accuracy. The image features were extracted from the pre-trained model using the last layer until the final fully connected layer. It is essential to freeze this layer as the networks are usually trained for a specific task. The model's performance was improved by weights of this pre-trained model to be updated in the training process. This model outputs 512 in its last layer, consistent with 512-dimensional embedding. The model uses Callbacks functions like CSVLogger, EarlyStopping, ReduceLROnPlateau to keep track of loss and accuracy trends during the learning phase. The architecture of the model is shown in figure 3. The model was trained using different optimizers and loss functions to gauge the performance during prediction. A batch size of 120 was where each batch had 20 identities, where each identity consisted of 3 masked and three unmasked images. Layers like dropout and batch normalizations were used to reduce overfitting. It was observed that Adam Optimizer, with a decaying learning rate starting with a value of 0.05, predicted values with more confidence and also showed more generalization to

unseen data. The model produced an accuracy of 87%, and a plot of accuracy and loss trend can be seen in figure 4.

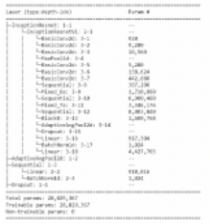

Figure 3. Model Architecture

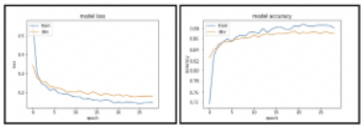

Figure 4. Plot of model loss and model accuracy for InceptionResNetV1.

c. ResNet-50

This model was also implemented using the FaceNet architecture for the VGG2 dataset, which produces an accuracy of 95.6%. The same training process was followed as before, and it was observed that this model performed better with RMSProp. This model produced an accuracy of 85.23%.

Dataset

The dataset consists of both unmasked and masked images. The unmasked images were obtained from the Labeled Faces in the Wild (LFW) database [1], composed of photographs of faces mainly designed to study face recognition. Approximately 13250 images of 5749 people are centered using the Viola-Jones face detector. The original dataset is a deep-funneled version consisting of four distinct sets of images and three different kinds of aligned images. The masked images were obtained from the Simulated Masked Face Recognition (SMFR) Dataset [5], composed of a mask-covered version on the existing LFW dataset. These simulated masked face images of people were generated using the Dlib library [6]. A combination of ~25K images of masked and unmasked versions will be used for the face recognition task. All the images were organized into separate folders, each corresponding to a person. The dataset was separated into training, testing, and validation sets in a 70-20-10 ratio.

Model Evaluation

The model's performance was compared using simple embedding matching. Initially, in its training phase, each of the above models computes a 512-dimensional embedding for each identity representing the ground truth. A testing sample represents an identity consisting of 3 masked and 3unmasked images. A sample is fed to the model during testing, which outputs an embedding describing the facial features. The matching technique is performed by computing the Euclidean distance between the ground truth and the newly obtained embedding, rendering a single value output. This output is then normalized, so it lies in the range of [0,1]. This Euclidean distance is compared with a preset threshold value which determines if the person has

been recognized or not. Additionally, to decrease the number of false alarms, the final output was based on deciding if a person is valid or not by checking the embedding matching criteria for at least three images passed from the test sample. It was observed that InceptionResNetV1 was the best performing model amongst the three implemented with an accuracy of 87%. In addition, the model was able to generalize well enough, and a few success and failure cases are discussed and shown in Figures 5 and 6. Hyperparameter tuning was also perfomed to compare the model's performance under different settings, as shown in Table 1.

Model	Accuracy	Parameters	Threshhold
InceptionResNetV1	87.97%	LR = 0.05, Loss = Triplet, Optimizer = ADAM, Epochs = 25	0.728
ResNet-50	85.61%	LR = 0.0001, Loss = Triplet, Optimizer =RMSProp, Epochs = 25	0.643
VGG16	76.23%	LR = 0.005, Loss = Triplet, Optimizer = ADAM, Epochs = 25	0.884

Table 1. Comparison of results obtained from different models

Figure 7. Correctly recognized masked faces

Figure 6. Incorrectly recognized masked faces

Conclusion and Future Works

This project successfully implemented a facial recognition system using transfer learning techniques and achieved comparable results. The model produced an accuracy of 87% and extended generality to unseen data. The accuracy can be improved by incorporating a loss function with two additional parameters like a masked positive and masked negative to accommodate the masked facial images. While producing satisfactory results, the supervised learning approach can include the task of labeling datasets, which is time-consuming. A different approach, like unsupervised learning, can be explored where this recognition task is implemented using generative data modeling.

References

1. nscozzaro, "facenet-pytorch." https://github.com/timesler/facenet-pytorch, 2019.
2. Masked Face Recognition using ResNet-50 (Masked Face Recognition using ResNet-50)
3. Simonyan, Karen and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition."
4. Schroff, Florian & Kalenichenko, Dmitry & Philbin, James. (2015). FaceNet: A unified embedding for face recognition and clustering. 815-823.
5. Zhongyuan Wang et al. "Masked Face Recognition Dataset and Application" In: arXiv:2003.09093 [cs.CV] 2020. http://dlib.net/
7. https://medium.com/@mohitsaini_54300/train-facenet-with-triplet-loss-for-real-time-face-recognition-a39e2f4472c3
8. Face recognition using triplet loss - Github.
9. Masked Face Recognition - paper.