Face Recognition - Few Shot Learning

Ayush Singal
2020365
ayush20365@iiitd.ac.in

Pranav Sharma
2020395
pranav20395@iiitd.ac.in

Tanish Gupta
2020344
tanish20344@iiitd.ac.in

Abstract

This research paper presents an analysis of three different models used for face recognition in a few-shot learning dataset. This dataset is unique in the sense that it has over 1320 classes, each having images ranging from as low as 2 to as high as 46 images. We implement various models, like the old fashioned K-Nearest Neighbours, the classic Support Vector Classifier, and newwr models like Siamese Neural Network. We use the dlib library(amongst others) to extract faces from the images in the former 2 models, while the logic of triplets in the latter. The paper explores the challenges of few-shot learning datasets and investigates different methodologies, including novelty models such as the implementation of a threshold to detect new faces in the dataset.

Keywords – face recognition, few-shot learning, KNN, SVC, Siamese NN, dlib, accuracy score, f1 score, cutoff, optimization

1 Introduction

1.1 Problem Statement

Facial recognition is becoming increasingly important in today's world for a variety of applications, including security and authentication systems. However, achieving accurate and efficient face recognition is still a challenge, particularly in real-world scenarios where there may be variations in lighting, facial expressions, and pose.

1.2 Motivation

The motivation behind our project is to develop a facial recognition system that can accurately and efficiently recognize faces even in challenging conditions. By using state-of-the-art machine learning techniques and deep learning models, we aim to create a system that can be applied in real-world scenarios, such as security systems and access control. Additionally, our project aims to explore the potential of different algorithms and models in the field of facial recognition, comparing their accuracy and efficiency to determine the most suitable method for different applications. Overall, our project aims to contribute to the advancement of facial recognition technology, improving its accuracy and efficiency for a range of practical applications.

1.3 Background

Machine learning models such as K-Nearest Neighbors (KNN) and Support Vector Machines (SVMs) have been used for face recognition, but they often require feature extraction techniques to be applied

on the images first. Recently, Siamese Neural Networks (SNNs) have gained popularity in the field of face recognition, as they are designed to learn discriminative features from pairs or triplets of images. The use of deep learning techniques for feature extraction has also shown promising results in improving the accuracy of face recognition models. In particular, the use of pre-trained deep neural networks, such as VGG-Face and deepface algorithms, has been shown to yield good results in one-shot learning scenarios. Given the challenges of one-shot learning datasets, there is a need for further exploration of different models and feature extraction techniques to improve the accuracy of face recognition systems. This project aims to contribute to this area of research by implementing and analyzing different models for face recognition on a one-shot learning dataset, and exploring novel approaches to improve the efficiency and accuracy of these models.

In addition to the aforementioned models, we also introduce a new method to detect if a face is present in our dataset or not. This approach can aid in the efficient management of large datasets by facilitating easy addition of new faces to the dataset. Moreover, we aim to optimize the inference time for feature extraction and label prediction to enable real-time face recognition. We believe this can have significant practical applications, such as in surveillance systems, security systems, and access control systems.

1.4 Related Work

One-shot learning is a challenging task where the model must recognize a new object/class from only a single training sample. A popular method for one-shot learning is Siamese neural networks, as proposed in the paper "Siamese Neural Networks for One-Shot Image Recognition" by Koch et al. (2015). The paper introduced a unique structure of neural networks that can learn similarity between inputs and generalize to new classes from unknown distributions. The authors achieved near state-of-the-art performance on one-shot classification tasks using a convolutional architecture.

In facial recognition, one of the main challenges is handling variations in facial appearance such as lighting, pose, and expression. Deep learning models such as convolutional neural networks (CNNs) have shown promising results in facial recognition tasks. In particular, facial feature extraction using pre-trained CNN models such as VGG-Face, Inception-ResNet, and FaceNet has become a popular approach for facial recognition.

Another approach for facial recognition is using facial landmarks and geometry. Dlib, a popular library for facial landmark detection and facial feature extraction, has been used in several studies. For example, Zhang et al. (2016) proposed a facial recognition system that combines dlib facial landmarks and a support vector machine (SVM) classifier. The authors achieved high accuracy on the LFW dataset, which is a popular benchmark for facial recognition.

Overall, one-shot learning and facial recognition are active areas of research in computer vision, and deep learning models have shown promising results in addressing these challenges.

1.5 Objectives

The objective of our project is to explore and implement various models for face recognition in a one-shot learning dataset. We aim to compare the performance of different models, including KNN, SVC, and Siamese NN, and investigate the challenges faced in few-shot learning datasets. Additionally, we attempt to introduce a method for detecting new faces in the dataset and optimize the inference time for real-time face recognition. The ultimate goal is to achieve high accuracy in face recognition and provide insights into the possibilities and limitations of few-shot learning datasets.

1.6 Scope

The scope of this project is to explore different machine learning models for face recognition in a few-shot learning dataset. Specifically, we aim to implement and compare the performance of three different models: K-Nearest Neighbors, Support Vector Classifier, and Siamese Neural Network. We will use the provided dataset, which has over 1650 classes, each with a varying number of images ranging from 2 to 46. We will also explore the use of different feature extraction techniques such as dlib and deepface to optimize the performance of our models. Additionally, we aim to investigate the challenges associated with few-shot learning datasets and propose novel approaches to overcome them. For instance, we plan to explore the implementation of a cutoff to detect new faces in the

dataset and improve the time taken to extract and recognize faces. The results of this project will provide insights into the effectiveness of different machine learning models for face recognition in a one-shot learning dataset. Additionally, our proposed approaches can have significant implications for real-world applications, such as surveillance systems, where the ability to recognize new faces quickly and accurately is crucial.

1.7 Impact

The impact of this project lies in its potential to improve the accuracy and efficiency of face recognition systems in few-shot learning datasets. Few-shot learning datasets are particularly challenging as they have limited samples per class, making it difficult to train accurate recognition models. By exploring different machine learning models and feature extraction techniques, this project can provide insights into the best approaches to tackle the challenges of few-shot learning datasets. Moreover, the proposed novel approaches, such as the implementation of a cutoff to detect new faces in the dataset, can have significant implications for real-world applications such as security systems, where the ability to recognize new faces quickly and accurately is crucial. Overall, this project can contribute to the development of more accurate and efficient face recognition systems, which can have a significant impact on various industries such as security, law enforcement, and marketing.

2 Materials and Methods

2.1 Dataset

The one-shot learning dataset used in this project consists of over 1320 classes(1324 classes to be exact), with images ranging from 2 to 46 per class. The dataset has 2 folders, “Face Data”, and “Extracted Faces”. The latter consists of the faces that have been extracted from the images present in the former folder. Each of these folders consist of 1324 folders, with the names of folders as class/label names, and each of them, in turn, have images belonging to that class.

The images in the label classes range from as low as 2 to as high as 46, with an average of about 5 images per class.

Our dataset has been taken from kaggle.
`https://www.kaggle.com/datasets/stoicstatic/face-recognition-dataset`

2.2 Exploratory Data Analysis (EDA)

Our dataset has 1324 classes, with 2 to 46 images per class. The dataset has a total of 6107 images, each image is of size 128x128, and is encoded in RGB. Further, these images already only contain the extracted faces using Haar-Cascade Classifier(cv2).
Here are some sample images from 3 classes
Here is atleast 1 extracted face image of each class

Figure 1: sample images

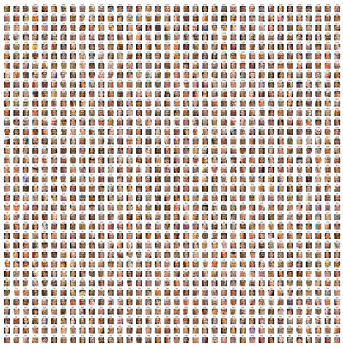

Figure 2: images of all labels

2.3 Models

As stated before, we have implemented 3 different models, namely, KNN, SVC and SNN.

2.4 Methodology

We will talk about the methodology of each of them separately now.

2.4.1 K-Nearest Neighbours

K-Nearest Neighbours (KNN) is a supervised machine learning algorithm for classification and regression problems. It is a simple algorithm that works by finding the k nearest data points in the training set to a given test data point and then using the majority class among those k neighbors to predict the class of the test point.

In the case of classification problems, the KNN algorithm calculates the distances between the test point and each point in the training set and selects the k nearest neighbors based on this distance. The most common distance metric used is the Euclidean distance. Once the k nearest neighbors are identified, the algorithm assigns the most frequent class among these neighbors to the test point.

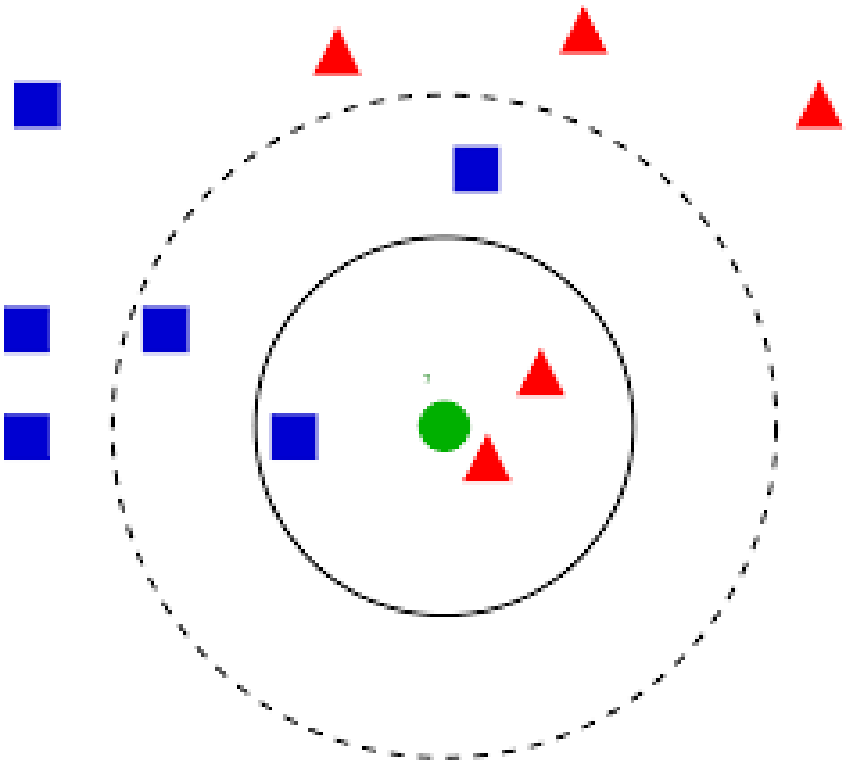

Figure 3:

KNN has the advantage of being a non-parametric algorithm, meaning it does not assume any underlying data distribution. It also has the flexibility to adapt to any number of classes and can work well with high-dimensional data. However, it can be computationally expensive for large datasets, as it requires calculating the distance between the test point and all the data points in the training set.

KNN was chosen because of the nature of our dataset, which focuses on fewer data points.

We first implemented KNN on flattened images, i.e., the 128x128 images were flattened into a one-dimensional vector of size 16384.

Then we extracted the facial features from the images first and then implemented KNN on them. This was done using various models of DeepFace, including FaceNet, VGG-Face, OpenFace, DeepFace, and ArcFace, which in turn helped us get much better results.

Finally, we decided on using the dlib library to extract facial features from images, which uses a pre-trained face detection model to locate the faces in the image, and then applies facial landmark detection to identify critical points on the face such as the eyes, nose, mouth, and eyebrows. These landmarks align the face and remove any pose or expression variations. After alignment, the image is passed through a pre-trained deep neural network called a face descriptor to extract a set of features that represent the unique characteristics of the face.

We decided to split the dataset in such a manner so that for each of the 1324 classes, half the images were in the training set, and half were in the test set.

Since our dataset has classes with images as low as 2, we could not set up a validation set since there was only one image in the training set.

The first step in implementing KNN was to extract the facial features of the images in our train and test images. As mentioned, this was done via Idlib, and the output was a 128-size vector of features for each image.

We plotted the accuracy vs. k graph for the KNN classification. For each value of k from 1 to 11, the KNN classifier of sklearn was fitted with the train data, and a custom prediction function was used to predict the classes of the test data.

The custom predictarr() function used includes detecting a face that is not in the dataset. More specifically, a cutoff was set, and the distance(euclidean) from the closest training point was noted. If the distance were higher than the cutoff threshold, the predicted outcome would be "-1", indicating the face was not present in the dataset. Otherwise, the usual prediction function of the KNN classifier was used.

We even noted the time it took for the complete process of prediction of an image(from feature extraction to prediction). This was done to keep an eye on the speed since it plays a vital role in our next objective, using the model for real-time face recognition.

The first method of flattening the images yielded an accuracy of 3

Using DeepFace increased the accuracy, and models like FaceNet, VGG-Face, OpenFace, DeepFace, and ArcFace yielded accuracy vs K graph as follows:

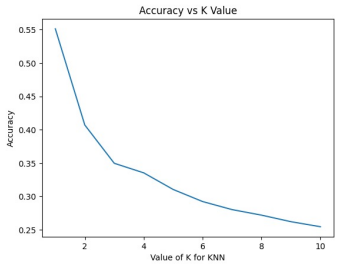

Figure 4: Face Net

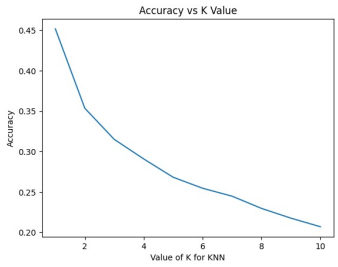

Figure 5: VGG-Face

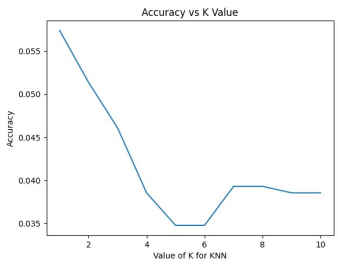

Figure 6: OpenFace

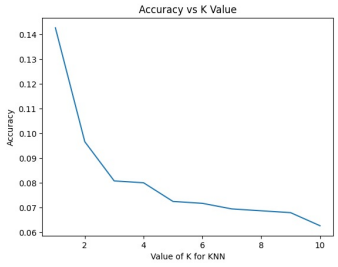

Figure 7: DeepFace

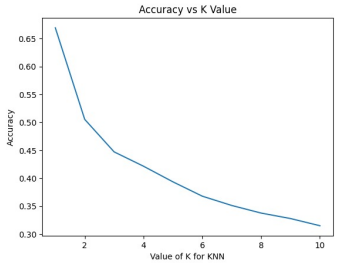

Figure 8: ArcFace

Finally, coming to the dlib feature extraction method, we got the following results.

We had two hyperparameters to be tuned in this model of ours. The first was the value of K, and the second was the cutoff value.

The plot below shows the accuracy vs. k graph of the KNN model:

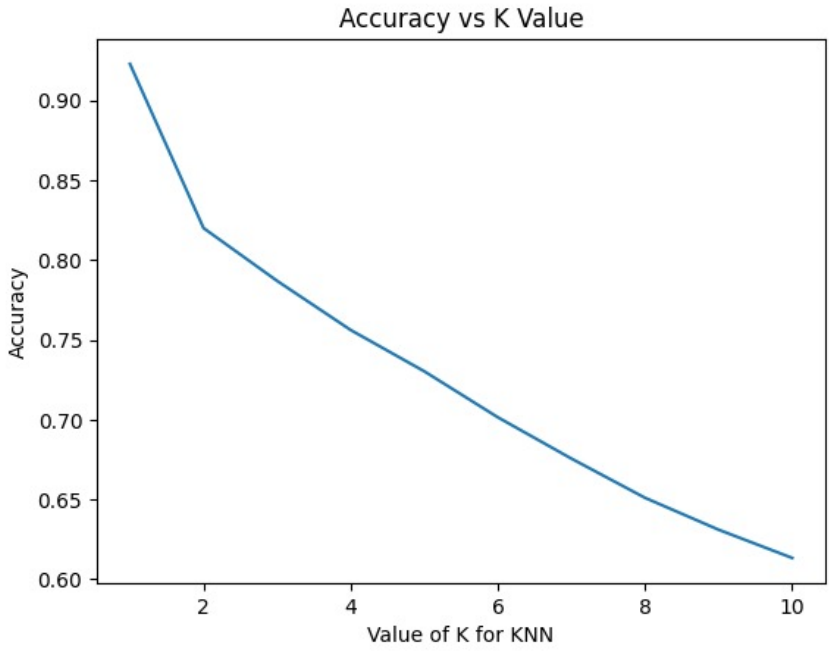

Figure 9: Accuracy vs K

We can see the nature of the graph, which indicates that the accuracy is consistently decreasing for increasing values of k.
This is because of the uniqueness of our dataset. Because the total number of images is less than 1 for a label in the training set, the test image would be closest to significantly fewer train images of the same label. Hence, our graph shows the right tendency of decreasing accuracy with increasing k.

Hence the optimal value of k is 1.

For the cutoff value, we noted the distance of the test data point(not in our dataset) with the closest train point and noted it to be greater than 0.5 units. Hence, the cutoff value was set to 0.5 in this model.

For these hyper parameters, we got an accuracy of 92 percent, an F1 score of 87 percent, and the total inference time for a single image to be 0.28 seconds.

2.4.2 Support Vector Classifiers

Before training an SVC model, it's important to prepare the dataset by splitting it into training and testing sets. In our case, since the dataset was not pre-separated, we created a new dataset by randomly splitting the data into two parts: one for training the model and the other for evaluating its performance. A common approach is to use a 50:50 split, where half of the data is used for training and the other half for testing. For our scenario of few-shot learning this splitting technique is optimal. To remove unnecessary features from the raw dataset, we extracted the face details from the images using the dlib face recognition model. This allowed us to obtain a numerical representation of the facial features and remove any unwanted elements that may be present in the dataset. Our dataset is few shot, meaning it has a limited number of images per subject.

Support Vector Classification (SVC) is not the most suitable algorithm for few-shot learning tasks.Few-shot learning is a type of machine learning task where the algorithm needs to recognize new objects or classes from very few examples, usually just one or a few.

SVC is a popular algorithm for classification tasks because it can handle both linearly separable and non-linearly separable data, and it is relatively insensitive to the presence of outliers. However, SVC has some drawbacks, such as being computationally expensive for large datasets and requiring careful tuning of hyperparameters, such as the regularization parameter and the kernel function parameters.

After feature extraction, a simple, untuned Support Vector Classification (SVC) model was trained on the extracted features. The SVC model was trained using default hyperparameters, which means that no tuning was performed on the model. Two types of kernels were used in the SVC model, namely the radial basis function (RBF) kernel and the linear kernel. The model was trained separately with each kernel type to compare their performances. The accuracy of the model was evaluated using a test dataset, and it was observed that the untuned SVC model achieved an accuracy of 13 percent with the RBF kernel and an accuracy of 16 percent with the linear kernel. These results suggest that the untuned SVC model with default hyperparameters did not perform well on the dataset. After observing that the untuned SVC model did not perform well on the dataset, a grid search was performed to find optimal hyperparameters that would improve the model's performance. The grid search involved testing a wide range of hyperparameters values for the SVC model. The hyperparameters that were tuned included the regularization parameter (C), the kernel type (linear, RBF, polynomial), and the kernel coefficient (gamma). The grid search was performed using cross-validation, where the dataset was split into multiple folds, and each fold was used as a validation set while the other folds were used as training sets. The model was trained and evaluated for each combination of hyperparameters, and the combination that resulted in the highest accuracy was selected as the optimal hyperparameters for the model. The results of the grid search showed that the performance of the SVC model improved significantly

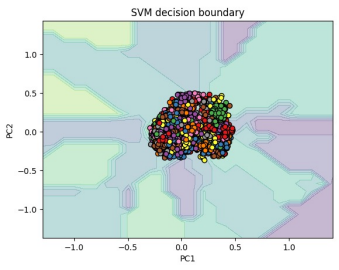

Figure 10: boundry

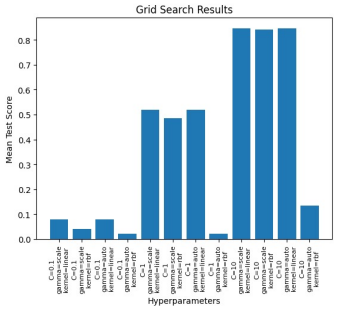

Figure 11: parameter

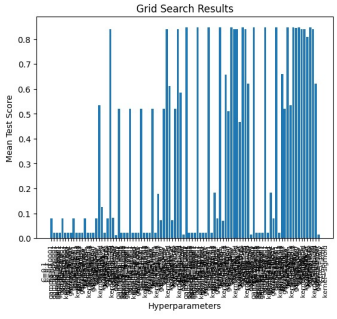

Figure 12: parameter

A prediction time of 0.082 seconds per image is relatively fast and reliable for real-life scenarios. This means that the model can process multiple images within seconds, making it suitable for real-time or near real-time applications too.

2.4.3 Siamese Neural Network

Siamese neural networks are designed to excel at few-shot learning, where a model is required to recognize a new sample with only one or a few examples of that sample. In this dataset, we have extracted faces of random people, and the goal is to train the Siamese model to recognize similar faces. The Siamese model requires triplets of images for training, where each triplet consists of an anchor, a positive, and a negative image.

To create these triplets as shown in figure [4], we have divided the data into folders based on the classes rather than taking some images from each class. This approach helps us address class imbalance, as it ensures that we have enough data from each class for the training process. Once the data is divided into folders, we select an anchor and a positive image from the same folder, as they share similar features. We then randomly select a negative image from another folder. This triplet is then used to train the Siamese network.

Since the minimum number of images per class is 2, we must use the triplet strategy to ensure that we have enough data to train our model effectively. Additionally, this approach is more effective at addressing class imbalance, which is common in many real-world datasets. With this method, the

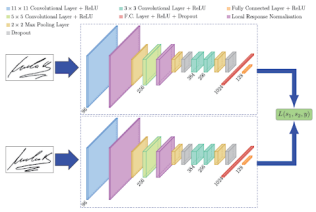

Figure 13: SNN

Figure 14: triplets

Siamese network can recognize images in the future with very few images per class, making it a powerful tool for few-shot learning tasks.

A siamese neural network (SNN) is a neural network architecture consisting of two or more identical sub-networks that learn to extract features from the input data as shown in figure [3] the sub networks are most commonly CNN models. The idea behind the architecture is that by sharing weights and parameters, the network can learn to recognize similarities between inputs based on their feature vectors.

The SNN works by taking two or more inputs, passing each of them through their own identical sub-networks to extract features, and then comparing the feature vectors to determine their similarity. The similarity is typically calculated using a distance metric such as Euclidean distance or cosine similarity. The output of the SNN is a single value that indicates how similar the inputs are.

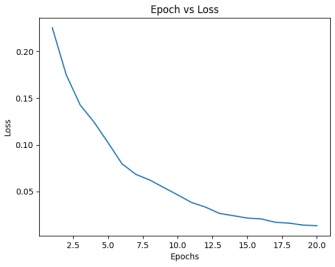

Figure 15: Mobilenet

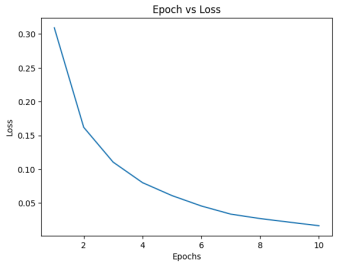

Figure 16: VGG19

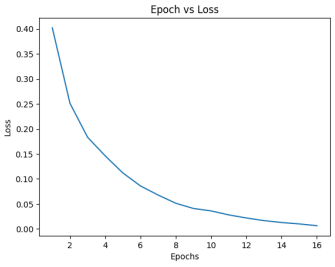

Figure 17: Efficientnet

The SNN is trained using pairs of inputs and a target output, which indicates whether the inputs are similar or dissimilar. During training, the weights and parameters of both sub-networks are updated in parallel to minimize the loss between the predicted output and the target output. The loss function used in SNNs is typically a contrastive loss or a triplet loss, which encourages the network to learn features that are discriminative for the task of comparing inputs.

Since the cross entropy loss is not valid for this model therefore the loss that we use is Triplet Loss. These plots shows the loss vs epoch graphs and for this we took different combinations of the hyper-parameters in order to tune the model and improve accuracy. So we used three pretrained models here that are VGG 19, Mobile net v2, Efficient Net as well after training these models the VGG 19 model was the best model to get the accuracy of 89 percent.

the test accuracy for these three models were as follows : VGG 19 - 89 percent MobileNetV2 - 88 percent EfficientNet - 86 percent

Among the various models that we experimented with, the VGG 19 architecture yielded the best performance. However, the downside of this model was its heavy computational cost and longer training time. To address the challenge of detecting faces in the dataset, we implemented a technique that excluded images where the face was not present. We tested this on a sample of images and found that a threshold value of 0.5 to 0.6 worked well in determining if the image belonged to the dataset or not.

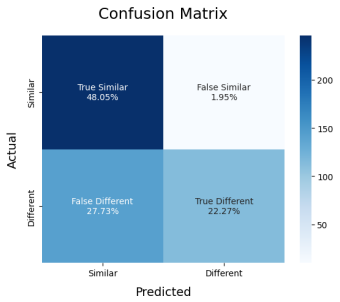

Figure 18: Mobilenet

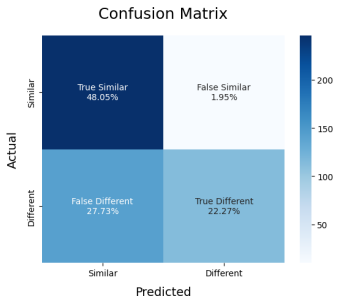

Figure 19: VGG19

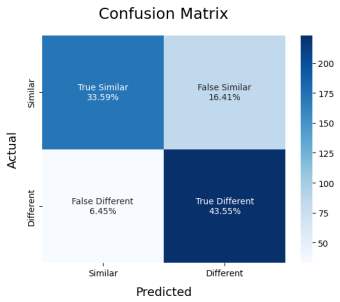

Figure 20: Efficientnet

Another crucial aspect that we considered was the reduction of inference time. This is important because a model that takes a long time for inference may not be practical for real-time applications. In our experiments, we found that the MobileNet model provided the best inference time, taking only 0.14 seconds per image. This made it a more suitable candidate for real-world applications where quick processing is essential. Overall, by considering factors such as model performance, face detection, and inference time, we were able to optimize our model for improved results and practical usage.

2.5 Novelty

The novelty of our project lies in several aspects. Firstly, we are analyzing a few-shot learning dataset that has over 1320 classes, with images ranging from as low as 2 to as high as 46 images per class. This kind of dataset poses a significant challenge for face recognition models, which are traditionally trained on large datasets with many images per class. Secondly, we are comparing the performance of three different models for face recognition: the K-Nearest Neighbors model, the Support Vector Classifier, and the Siamese Neural Network. By comparing these models, we aim to identify the strengths and weaknesses of each approach in the context of few-shot learning.

Additionally, we have tried to implement a cutoff to detect new faces in the dataset, which has not been explored much in previous studies. This feature will make our model more robust in the sense that it can detect faces that are not present in our dataset, which is an important consideration in real-world applications of face recognition.

Finally, we aim to keep the inference time of our models low, which is essential for real-time face recognition systems. By inference time, we mean the time taken for the entire process of extracting facial features from an image, and predicting the label of that image, or predicting that the face is not present in our dataset.

Overall, our project's novelty lies in its combination of analyzing a challenging dataset, comparing different models, implementing an algorithm to detect new faces, and optimizing our approach for real-time inference.

2.6 Applications

There are several potential applications of our project. One of the most important is in the field of security, where face recognition technology is commonly used to identify individuals in a crowd or at a security checkpoint. With the ability to accurately recognize faces in a few-shot learning scenario, our project could help enhance security measures in public spaces, airports, and other high-security areas.

Another application is in the field of human-computer interaction, where face recognition technology is used to provide personalized experiences to users. For example, a computer could use face recognition technology to automatically log in a user and adjust settings such as font size and display preferences based on the user's individual preferences.

Additionally, our project could have applications in the entertainment industry, where face recognition technology is used to personalize content recommendations for users. For instance, a streaming service could use our technology to recommend movies and TV shows based on the user's viewing history and facial features.

Overall, the potential applications of our project are numerous and diverse, and we believe that it has the potential to make a significant impact in a variety of industries.

2.7 Evaluation Metric(s)

The evaluation metric for our project will be accuracy, which is the number of correctly classified images divided by the total number of images in the dataset.

Since our task is to recognize faces, accuracy is a suitable metric as it measures the percentage of correctly identified faces. We will also consider other metrics such as F1 score to evaluate the performance of our models. F1 score is the harmonic mean of precision and recall and provides a balanced measure of the two.

Additionally, we have also used confusion matrix to analyze the performance of our models and identify the classes that are being misclassified.

3 Results and Discussions

For K - Nearest Neighbor We made our KNN classifier more efficient by implementing DeepFace and dlib libraries for facial extraction, from 3 percent to 92 percent. We were able to successfully detect the face not in our dataset, and also noted the inference time for predicting an image, which came out to be 0.28 seconds.

For Siamese Neural Network We used three pre trained models here that are VGG 19, Mobile net v2, Efficient Net as well after training these models the VGG 19 model was the best model to get the accuracy of 89 percent. the test accuracy for these three models were as follows :
VGG 19 - 89 percent
MobileNetV2 - 88 percent
EfficientNet - 86 percent

We found that the MobileNet model provided the best inference time, taking only 0.14 seconds per image.

For Support Vector Classifier A prediction time of 0.082 seconds per image was obtained and an accuracy: of 91 percent.

4 Conclusion

In conclusion, the few shot learning problem was addressed using three different models - SNN, SVC, and KNN. The SNN and SVC models were able to achieve high accuracy rates of 89 percent and 91 percent which is a good performance given the complexity of the problem. However, the KNN model outperformed both SNN and SVC models with an accuracy of 92 percent.

Although the SVC model performed slightly better than the SNN model, it's important to note that the choice of model ultimately depends on the specific requirements and constraints of the problem at hand. Although KNN gives the maximum accuracy, its inference time is so high, that it is not suitable for real time applications. The SVC model's fast prediction time makes it a reliable choice for real-life applications, while the SNN model's ability to learn complex patterns and relationships may be beneficial in more complex tasks.

Overall, the results of the experiments demonstrate the effectiveness of these models in addressing the one shot learning problem, with the KNN model emerging as the most accurate model. However, the selection of the most appropriate model will depend on various factors, including inference time, the size of the dataset, the computational resources available, and the specific requirements of the problem.

5 Distribution of work

Tasks	Team Members
Data Collection	Pranav
Data Cleaning	Tanish
Pre-processing	Pranav , Ayush and Tanish
Data Visualization	Ayush
Feature Extraction	Tanish, Pranav and Ayush
SNN	Pranav
KNN	Tanish
SVC	Ayush
Hyperparameter Tuning, Check for Overfitting and underfitting of models, and selection of best model	Tanish, Pranav and Ayush

Table 1: Work contribution

6 References

[1] Siamese Neural Networks for One-Shot Image Recognition: https://www.cs.cmu.edu/ rsalakhu/papers/oneshot1.pdf

[2] FaceNet: A Unified Embedding for Face Recognition and Clustering: https://arxiv.org/pdf/1503.03832.pdf.

[3] https://builtin.com/machine-learning/siamese-network

[4] DeepFace: Closing the Gap to Human-Level Performance in Face Verification: https://www.cs.toronto.edu/ ranzato/publications/taigman$_c vpr14.pdf$

[5] VGG Face Descriptor: https://www.robots.ox.ac.uk/ vgg/software/vgg$_f ace/$

[6] OpenFace 2.0: Facial Behavior Analysis Toolkit: https://cmusatyalab.github.io/openface/