Pattern Recognition and Machine Learning Minor Project: Face Mask Detection

Gattu Hemanth *(B19EE030)* and Mohan Chhabaria *(B19EE096)*

Abstract

The paper depicts various machine learning algorithms and their performance on the Real-world masked world dataset and self -built masked face dataset. The dataset includes almost 90K sample images of objects belonging to 2 different classes. The used classifiers are Support Vector Machines, K nearest neighbour, Multilayer Perceptron and Convolutional Neural Network

Index Terms

KNN, SVM, CNN, MLP masked, unmasked classification, recognition classifier, train, test, accuracy

I. INTRODUCTION

THIS paper talks about the implementation of image classification using a variety of classifiers on the Real world masked face dataset and self-built masked face dataset. The dataset contains 96K samples with nearly 6000 images of faces with mask and 90k without masks.

Image Classification is one of the hot topics in computer vision and pattern recognition research. In recent years scene understanding and object classification has received intense attention as it is beneficial for many practical applications such as autonomous driving, medical applications, robot vision and content-based image retrieval. Image classification came into existence for decreasing the gap between the computer vision and human vision by training the computer with the data. The image classification is achieved by differentiating the image into the prescribed category based on the content of the vision.

II. IMPLEMENTATION

A. Data Visualisation

1) Here are some sample images with their class labels from the training set

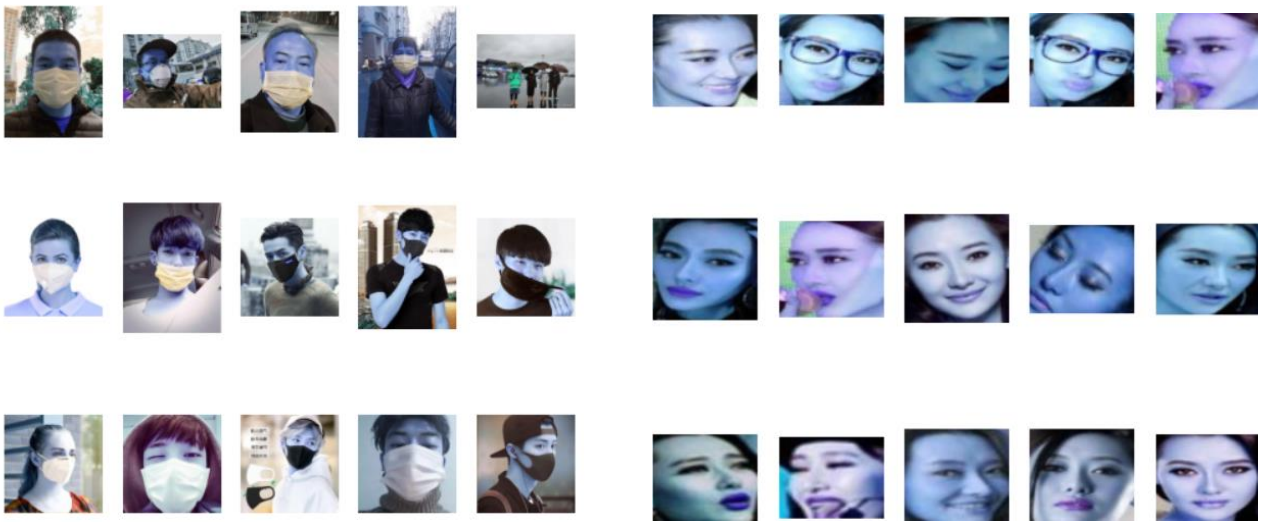


Fig. 1: Some samples of the data

B. Data pre processing

The dataset given is very much unbalanced as it has 90k samples of one class (unmasked) and almost 6000 samples of the other class. So, we took lesser samples from the first class so to create a well-balanced dataset. Also, as the images are very of high-quality images it is not possible to use these large number of images as it can crash a normal system as it requires very large computation power. The images are of varying shapes so to get a common shape we used average value of shape that came out to be (175,175,3). Also, we used a validation dataset including 500 samples of each class.

C. Support Vector Machines

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. The goal of the SVM algorithm is to create the best line or decision boundary (hyperplane) that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category for unknown data points. It uses support vectors that are data points closest to the margin or lie on the margins of the hyperplane.

To start with we used a standard scaler to scale the data so that the distribution is a normal distribution for all features. It subtracts the mean from all the feature values and then is divided by the variance. This helps us bring the data into a limited range
Now, as the number of attributes or pixel values is very high, that is 3072 so we used Linear Discriminant Analysis (LDA) which is a feature reduction technique. The number of components used is 9, it is selected because of the following equation:

No of components = min (L-1, no of features), where L = no of classes (here 1)

Next, we need to tune the hyperparameters for our SVC models, so we used grid search CV, which uses an iterative method to give us the best parameters for our model. The final or best hyperparameters came out to be [Kernel: 'rbf' and C: 0.1]. The trained classifier gave test accuracy of 99.8% and train accuracy of nearly 100%. The classification report and the confusion matrix can be seen in figure 2 and figure 3 respectively. Also, the cross-validation sores for the validation dataset were pretty much near to 1.0, it may be a possibility that the model is overfit.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	489
1	1.00	1.00	1.00	511
accuracy			1.00	1000
macro avg	1.00	1.00	1.00	1000
weighted avg	1.00	1.00	1.00	1000

Fig. 2. Classification Report for SVM

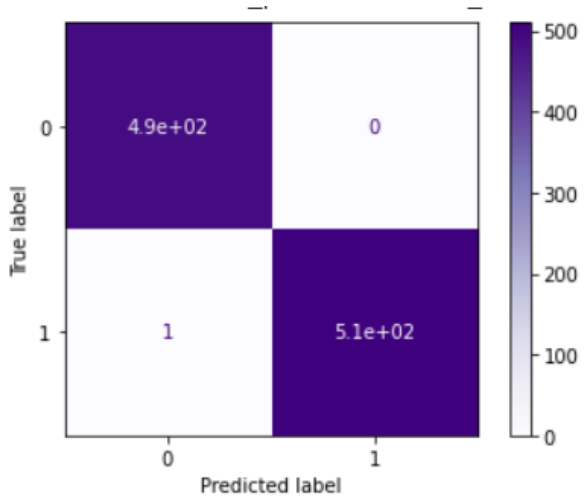


Fig. 3. Confusion Matrix for SVM

C. K- Nearest Neighbour

KNN is also a supervised machine learning technique that can be used for classification as well as regression problems. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).
To select the value of K that works best for our data was our first task, so for that we used the KNN algorithm several times with different values of K using Grid Search CV for our training set and chose the K that best fits our model. The list of values used for this is [1, 2, 3, 4] and also leaf nodes were given as parameters. Fig. 4. shows the outcomes of this cross validation.

We used the best estimator provided by grid search cv and then the test accuracy for this classifier came out to be 99.6 % and the test accuracy was almost 100%, other results can be seen from the classification report Fig.4. and confusion matrix (Fig. 5.) that detail the predictions.

	precision	recall	f1-score	support
0	1.00	0.99	1.00	517
1	0.99	1.00	1.00	483
accuracy			1.00	1000
macro avg	1.00	1.00	1.00	1000
weighted avg	1.00	1.00	1.00	1000

Fig. 4. Classification report for KNN

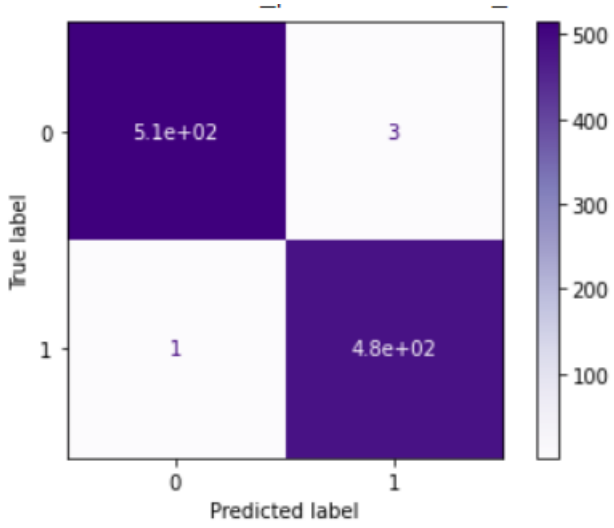


Fig. 5 Confusion Matrix for KNN

The cv scores were pretty much convincing and are listed here `[1.0, 1.0, 0.99, 0.99, 1.0]`

D. Convolutional Neural Network

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to the various objects in the image which are differentiable from each other. A CNN is built with many layers and we have used various layers in our sequential model, which takes these layers and adds them in a sequence to make a network. We built 2 different networks one was for RGB images and the other for grayscale images.

RGB images: To begin with we divided the arrays (image arrays) by 255 as the pixel value ranges from 0-255, this enables us to get data in a common range of 0-1. The initial layers included a pair of Conv2D and Maxpool layers, followed by a series of flatten and dense layers. This model provided better results and the accuracy came out to be greater than 99%. The classification report and the performance is shown in Fig. 6 and Fig. 7 respectively.

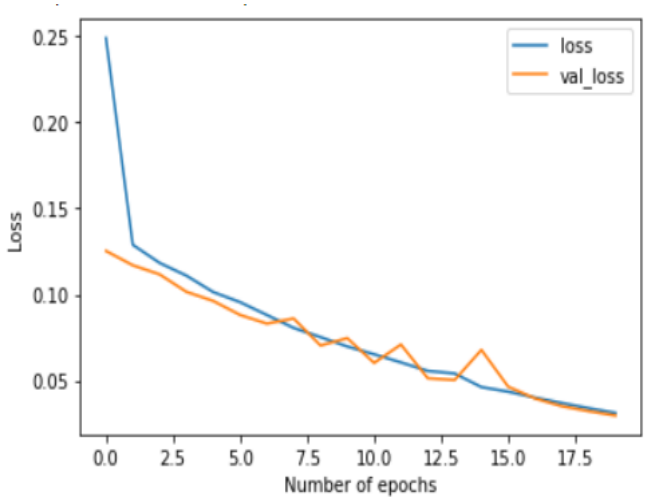


Fig.6 Loss for validation and training data

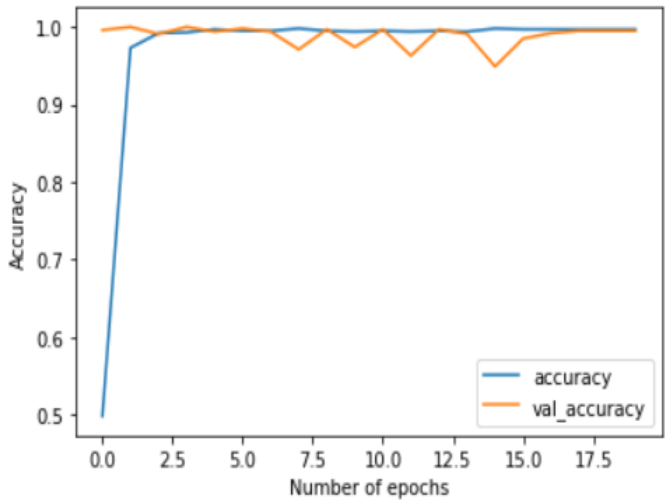


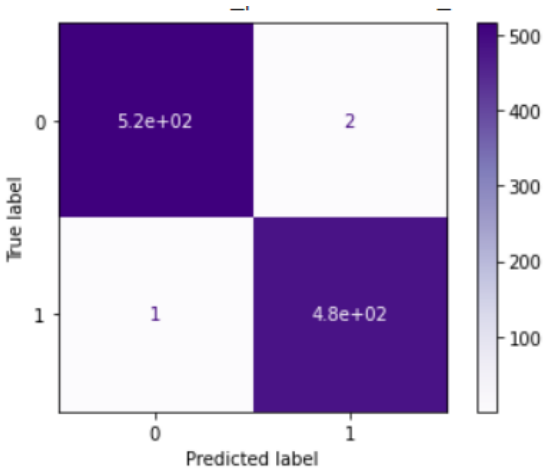
Fig. 7. Accuracy for validation and training data number for CNN

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 173, 173, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 86, 86, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 84, 84, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 42, 42, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 112896)            0
_________________________________________________________________
dense (Dense)                (None, 128)               14450816
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_3 (Dense)              (None, 10)                330
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 14,480,885
Trainable params: 14,480,885
Non-trainable params: 0
```

Table 1. Architecture of CNN

E. Multilayer Perceptron

Multilayer Perceptron is a feed forward Artificial Neural Network. In the implementation we have used one the in-built model of Sklearn that is Multilayer perceptron classifier, it takes the inputs and using the weights for various layers gives the class label, we are required to give hidden layers, loss function, learning rate etc as parameters. The best params were retrieved using Grid CV search which gave these as the best params. [`activation: tanh, alpha:0.0001, hidden layer sizes: (150,)`] The accuracy we got with this model was nearly 99%.



	precision	recall	f1-score	support
0	1.00	1.00	1.00	517
1	1.00	1.00	1.00	483
accuracy			1.00	1000
macro avg	1.00	1.00	1.00	1000
weighted avg	1.00	1.00	1.00	1000

Fig. 8 Confusion matrix and classification report for MLP

III. CONCLUSIONS

1. Comparing all the classification algorithms used we can state that SVM has performed the best, still the performances of other classifiers can't be ignored as they are very near to the best. The plot in Fig. 9 and Fig. 10 compares the accuracy of every classification described in this paper.
2. Data Augmentation can be used to improve accuracy of CNN as it can increase the number of samples. It obviously requires us to change the architecture of the network and find an efficient one but still is an improvement. Here are some examples of samples generated by augmentation shown in Fig. 11.

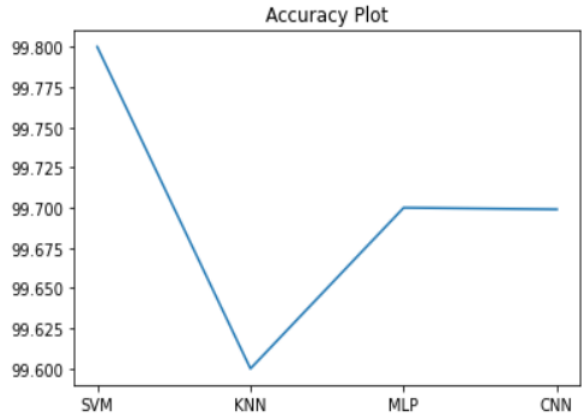


Fig. 9. Comparison among used classifiers

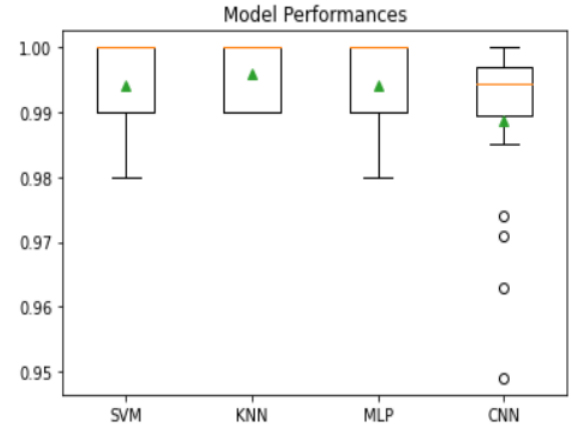


Fig. 10. Boxplots comparing validation scores

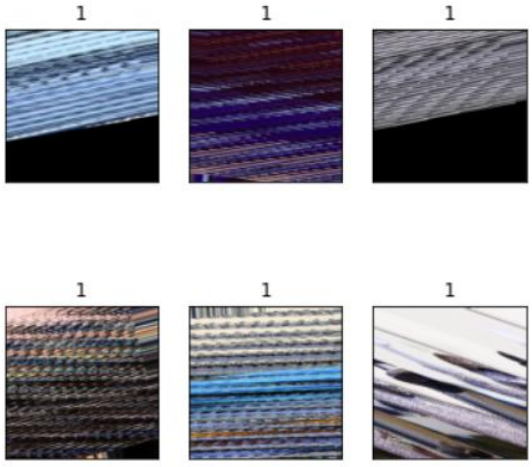


Fig. 11. Images created using data augmentation (175*175*3)

Contributions

Mohan Chhabaria: Data pre-processing, data Cleaning, data Visualization, built a model using KNN and CNN and tried to improve accuracy by changing hyperparameters. Performed Dimensionality reduction by trying different methods such as PCA, feature selection methods.

Hemanth Gattu: Built a model using SVM and MLP and tried to improve accuracy by changing hyperparameters, and standardizing the data. Performed Dimensionality reduction by trying different methods such as LDA

Report was written in the presence of each team member.

References

1. Sklearn documentation
2. Keras documentation
3. Tensorflow documentation
4. https://github.com/moritzhambach/Image-Augmentation-in-Keras-CIFAR-10
5. https://www.tensorflow.org/tutorials/images/cnn