1.1 Data Source

I would like to acknowledge the authors of these task and images.
Kaggle problem: https://www.kaggle.com/andrewmvd/face-mask-detection
Source of the data: https://makeml.app/datasets/mask

This data set contains labelled images of 3 classes – Mask worn, Mask worn incorrectly and Mask not worn. The aim is to create a classification model.

1.2 Steps to execute the code

Step 1: Obtain the annotations and images folders from the source and save them into a folder. Ensure that the path is same in all the notebooks.

Step 2: Run FaceMaskClassification_XMLparsing.ipynb to parse given information
Step 3: Change the variable "model_num" accordingly for each model.

1 = PCA-SVM
2 = RandomForest
3 = CNN

Run FaceMaskClassification_FeatureEngineering.ipynb to extract faces and labelling

Choose each of step 4 for respective model.
Step 4a: Run FaceMaskClassification_RFModelTraining.ipynb for RandomForest
Step 4b: Run FaceMaskClassification_CNNModelTraining.ipynb for CNN
Step 4c: Run FaceMaskClassification_PCA_SVMModelTraining.ipynb for PCA & SVM

Step 5: Change the variable "model_num" accordingly for each model.

1 = PCA-SVM
2 = RandomForest
3 = CNN

Run FaceMaskClassification_RealTimeMaskDetection.ipynb for real time application

1.3 Data Integration

All coding was executed in python notebooks. The notebooks have been segregated based on the tasks performed like ETL and modelling. The data from one notebook is saved in various forms such as pickle format for models and .npy format for image data.

This allows debugging to be performed accurately.

1.4 Discovery and Exploration

This project is completed using Python in Jupyter notebook.

Data quality assessment

The source contains images and their necessary information in xml files. Hence, the details are parsed in the jupyter notebook named as FaceMaskClassification_XMLParsing using Beautifulsoup library. Beautifulsoup is simple to use for everyone. The information obtained are the size of images and the sizes of every bounding box for every face in all images. As the data is stored in string format in xml files, it is essential to convert them to integer types. The quality of data is first verified by visualizing these faces to ensure correct indication of faces. Also, check for any missing values have been done initially.

Feature engineering

This is coded in jupyter notebook named as FaceMaskClassification_FeatureEngineering.

- Image processing

  Image processing is necessary due to various reasons and is done using OpenCV and Numpy libraries. As the size of images are of various sizes, it important to standardize the size as the models only allow images of same sizes. As the colours of masks play no significance in mask detection, the images are converted to grayscale from RGB scale. Furthermore, the pixel values have been normalized between 0 and 1 by division of 255.

- Labelling

  As the 3 classes are in string format, Labelencoder function from Scikit-Learn library to convert them into 3 integer form for model fitting.

Algorithm

- PCA & SVM

  PCA is an unsupervised model that extracts the prominent features (eigenvectors that display great variance) of the images based on the eigenvalues. This reduces the dimension of features that are fed into the prediction model. Only the first few are selected as features out of the resulting principle components from PCA. SVM is a supervised classification model that uses hyperplane to classify images into various labels in the higher dimensions. The hyperplane is selected by maximizing the separation distance among the classes.

- RandomForest

  RandomForest is an ensemble technique that consists of many decision trees and classification is done with maximum vote Bootstrapping is done where a samples from the entire set of images are used for each decision tree. This decreases the variance and thereby preventing overfitting and allowing the model to be generalized for any set of images.

- CNN

CNN is a deep learning model widely used for images. CNN is a series of filters and polling with classification at the end of the model. The features are learnt through the filters. The classification aspect is done by the neurons that determine the weights and biases, resulting in the appropriate class. It is proven to be more effective than the traditional machine learning models for images.

Framework

- Scikit-Learn

  This library contains various machine learning algorithms in python integrated with other optimized libraries such as SciPy and Numpy.

- TensorFlow & Keras

  In Deep Learning, Tensorflow is considered to be low level API while Keras is high level API. Tensorflow is an end to end platform for machine learning with numerous flexible settings for model building. Keras is built on top of Tensorflow and which is easier to use allowing faster prototype. Hence, Keras will be used to build model at a faster pace.

Model performance indicators

- Validation accuracy and loss
  - These 2 parameters are the standard indicators for both test and validation data to have an idea of the overall performance of model for all 3 classes. This is more significant in CNN where these parameters for each epoch are plotted to identify the best combination of accuracy and loss for validation data.
- Confusion matrix
  - This is a significant indicator for classification task. It shows the precision and recall values for each of the 3 categories involved. Precision identifies the comparison between true and false positives while recall identifies the comparison between true positives and false negatives. Hence, it is clear to visualize how well each category can be classified into.
- Time taken for real time processing
  - As the test data are from real time face detection, it is essential to observe which model is able to process fast and output the result in the live image through the coloured boxes.

1.5 Applications / Data Products

As a data product, a real time face mask detector is built as an extension to the Kaggle problem using OpenCV which captures the video through laptop camera. Real time frame by frame image processing and classification are performed with the pre-trained models that I have built previously for this project. The images processed in real time will be test set of data to the models trained.

The real time detector will show the effectiveness of models in detecting masks. We can observe the time taken for classification to be output on video and the effectiveness in classifying accurately. This mask detector application is built with the intention to place the skills learnt to its proper use. This can help to resolve real word mask detection issues in the midst of Covid-19 crisis or be enhanced by other developers.

1.6 Observations

From observation, the PCA-SVM model takes the longest time as there is so much lag involved in displaying. Random Forest and CNN were able to provide better detection at real time. Nevertheless, the models do require more tuning for better possible performances.

Differentiating ‘mask worn’ and ‘mask worn incorrectly’ was indeed challenging and there were times where models are not able to accurately predict. The latter is mostly classified as former. Training with more images for faces worn incorrectly will help model performances as the dataset is imbalanced.

There are some observations that shows more variations in images or scenarios are required. The real time face mask detection for this classification task is sensitive to faces with beard or at night. However, this can be due to the quality of video camera. Hence, more testing required with video cameras to ensure the impact on real time classification from such external factors are minimized. This allows performances of models to be emphasized.