CIS 522 – Final Project – Technical Report

Deep Learners

Team Members:

- Yashveer Singh Sohi; yashveer;
  Email: yashveer@seas.upenn.edu
- Shreyans Tiwari; shreytiw;
  Email: shreytiw@seas.upenn.edu
- Yiyasu Paudel; yiyasu;
  Email: yiyasu@seas.upenn.edu

***Abstract*—**
COVID-19 has impacted daily life of citizens and a major solution to this global pandemic is by masking up. Wearing mask correctly helps slow down the virus and thus, this project explores various deep learning image classification models to create an efficient masked facial detector. By using deep learning models like ResNet50 and MobileNetV2, the aim of the project is to create an efficient masked facial detector by achieving a higher accuracy score on spotting people who are masked up, not masked up or masked incorrectly. However, people masked correctly and incorrectly, and incorrectly and not masked are miss classified into the same bucket. Separating these classes makes the prediction extremely challenging.

I. Introduction

Face masking rules have been enforced on people to prevent the spread of COVID-19 pandemic. However, to wear or not to wear a mask has now been linked with social, political and ethical meanings, resulting in majority of them not wearing a mask or wearing them incorrectly. New observation from Health Hive has found that wearing a mask incorrectly does more harm and is no better than not wearing it at all. This project aims to tackle the spread of COVID-19 by flagging incorrect mask usage and ensure safe practices in public areas like airports, train stations and so on. The problem domain was chosen because all of us are well aware of the psychological and societal effects the pandemic has on the population and we aim to reduce the virus communication by creating an effective face mask detector.

This paper tries to solve the mask-detection problem using multiple deep learning approaches including a base CNN, a ResNet50 and a MobileNetV2 implementation. A baseline K-Nearest Neighbor is also implemented to compare performance. Overall, we were able to achieve a best accuracy of 98.93% in detecting the three categories of mask correctly worn, mask incorrectly and mask not worn. In this paper, one can find analysis on feature description, advanced deep learning architectures and the real world significance of our work with regard to the post-pandemic world.

II. Related Work

There are numerous dataset available on the internet and various methods have been used to tackle face mask detection. Most of the projects aim to detect masks in images (mask/no mask).

As mentioned by Neha Ann Nainan and team, the face mask detection by using inexpensive pretrained models such as MobileNetV2 and InceptionV3 and these fine tuned model managed to accurately predict the data [1].

The aim of the project Lien Nguyen and the team proposed were similar to our interest. Their paper focuses on creating a face mask detector that is pre-trained on MobileNetV2 with ImageNet pretrained weights, which was then used to detect three classes: correctly worn, incorrectly worn and not worn at all, by deploying a temperature sensor on Raspberry Pi 3. Similar results have been obtained by this paper which achieved a trained accuracy rate of 98.61% and 97.63% based on testing results [2].

Duy-Linh Nguyen and the team developed a facemask wearing alert system based on convolutional neural network (CNN) which includes two major stages: face detection and facemask classification to detect two classes: mask or no mask [3].

As far as we know, there is limited work on face mask detection which detects people with/without mask or wrongly masked. This paper intends to make public spaces safe by preventing the transmission of COVID-19 amongst humans, by making a comparative analysis between various models.

III. Dataset and Features

The final dataset is a combination of three dataset containing images from different sources; two from kaggle and one from GitHub. All the images were combined together with the following breakdown: 11553 (224x224x1) grayscale images in total; 3725 masking; 3828 not masking; 4000 incorrect masking. Figure 1, 2 and 3 shows the samples from the dataset. Out of these, 9478 were passed as training images, 1054 as validation images and 1171 as testing images having three labels: WIth Mask (label = 0), Without Mask (label =1) and Wrong Mask (label = 2). The final image was resized to 80x80n from 224x224 and Gaussian Blurring was applied to remove noise.

Fig. 1. With Mask (label=0

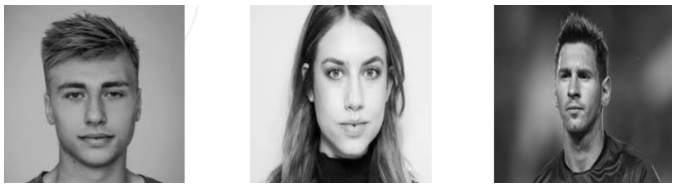

Fig. 2. Without Mask (label=1

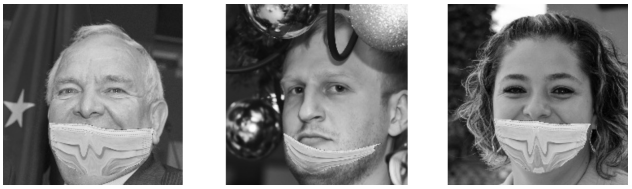

Fig. 3. Wrong Mask (label=2

IV. Methodology

The following is a list of all general pre-processing techniques that were used:

For our non-DL benchmark, we decided on using a KNN classifier. But since the images are 224x224, applying any non-DL model becomes complicated 50,176 features. To tackle this, we looked at feature description which allows for the most important information from images to remain preserved but reduces the size of the input to a great extent. For our implementation, we used Histogram of Gradients (HOG) feature description. The HOG feature descriptor counts the occurrences of gradient orientation in localized portions of an image. This allows for finding edges and their directions in the image efficiently by creating histograms for each localized region using the gradients and orientation of pixels.

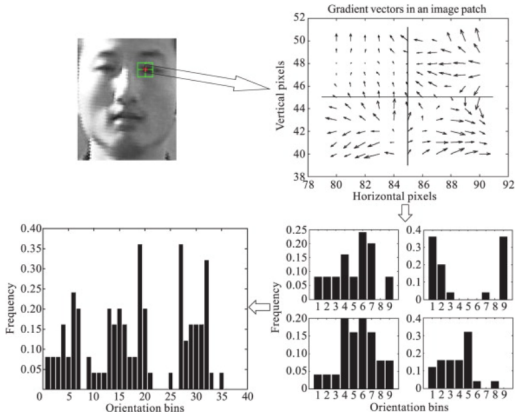

Fig. 4. Histogram of Oriented Gradients

For our implementation, the HOG desciptor managed to shrink the input features to a vector of size 2592 while detection the edges for the faces and masks (as shown below). The KNN Classifier was applied as the non-DL/standard ML model since KNN is a systematic and versatile algorithm that is used to solve classification problems. The sklearn library was used to import these tools and the dataset was split using train_test_split(). The plots are as follows:

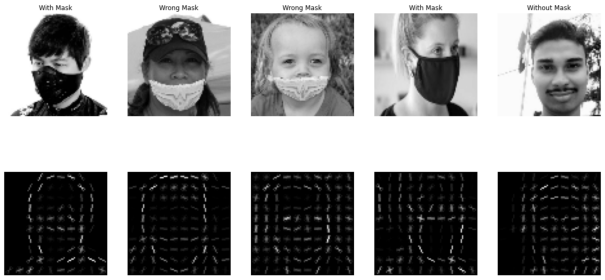

Fig. 5. HOG Filter

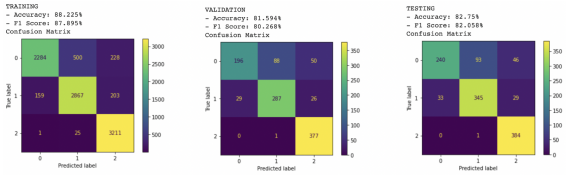

Fig. 6. Confusion Matrix: ML Model

A PyTorch Convolutional Neural Network (CNN) was chosen as the base-DL model since CNNs are one of the most effective methods for image classification. The architecture consists of five convolutional layers with a ReLU, activation, and max pool performed after each con- volutional layer. In addition, cross entropy loss function and adam optimizer with batch size = 32, epochs = 5 and initial learning rate = 0.0003 was used to train the model. The architecture summary is as follows:

The architecture summary is as follows:

```
class CNN(nn.Module):
  def __init__(self):
    super(CNN, self).__init__()
    self.conv1 = nn.Sequential(
      nn.Conv2d(in_channels=1,
      out_channels=16,
      kernel_size=3,
      stride=1,padding=2,),
      nn.ReLU(),
      nn.MaxPool2d(kernel_size=2),
    )
    self.conv2 = nn.Sequential(
      nn.Conv2d(16, 32, 5, 1, 2),
      nn.ReLU(),
      nn.MaxPool2d(2),
    )
    self.conv3 = nn.Sequential(
      nn.Conv2d(32, 64, 7, 1, 2),
      nn.ReLU(),
      nn.MaxPool2d(2),
    )
    self.conv4 = nn.Sequential(
      nn.Conv2d(64, 32, 5, 1, 2),
      nn.ReLU(),
      nn.MaxPool2d(2),
    )
    self.conv5 = nn.Sequential(
      nn.Conv2d(32, 16, 3, 1, 2),
      nn.ReLU(),
      nn.MaxPool2d(2),
    )
    # fully connected layer
    self.out = nn.Linear(784, 3)
```

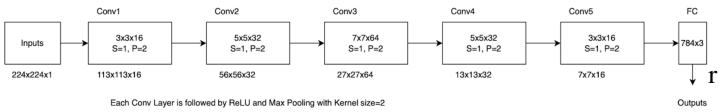

Fig. 7. CNN Classifier

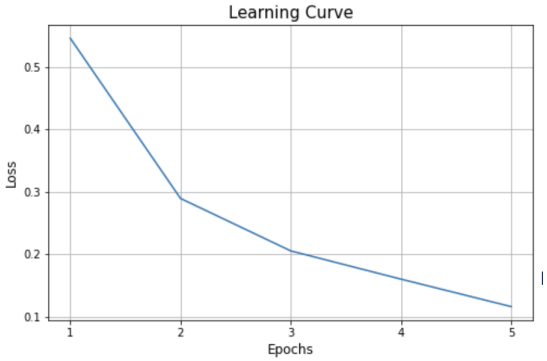

Fig. 8. CNN Model Performance

The pre-trained ResNet50 was implemented as one of the advanced architectures as it improves the efficiency of a deep learning network and has been found successful in other image classification tasks. For our implementation, we also took advantage of transfer learning for better performance and a smaller training time. The ResNet model was originally trained on the ImageNet dataset with 3 input channels and 1000 classes, which was fine-tuned to match our requirement of 1 input channel and 3 output classes. The architecture of the standard ResNet model was finetuned by changing the input channels for the conv1 later to 1. The loss function used was Cross Entropy loss and the optimizer used was Adam. In addition, a batch size of 32 was used for the dataloaders and the model was trained for 5 epochs with an initial learning rate of 0.0003. The architecture summary is as follows:

```
class ResNet(nn.Module):
  def __init__(self, in_channels=1):
    super(ResNet, self).__init__()

    self.model = models.resnet50(
    pretrained=True)
    self.model.conv1 = nn.Conv2d(
    in_channels, 64,
    kernel_size=7, stride=2, padding=3,
    bias=False)
    num_ftrs = self.model.fc.in_features
    self.model.fc = nn.Linear(
    num_ftrs, 3)
  def forward(self, x):
    return self.model(x)

resnet = ResNet()
```

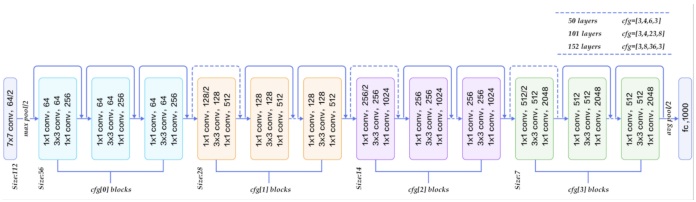

Fig. 9. ResNet50 Classifier

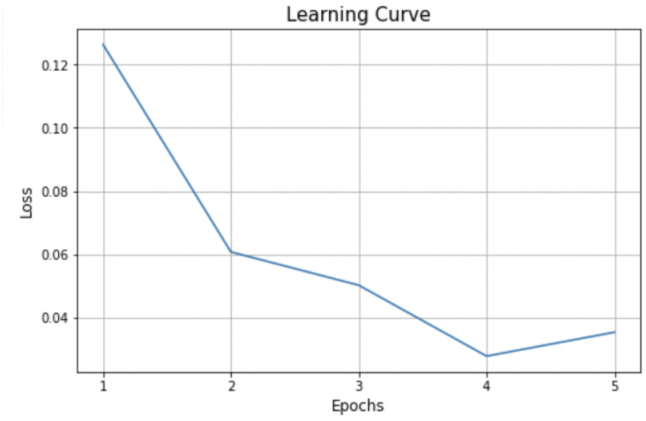

Fig. 10. ResNet50 Model Performance

As our next advanced deep learning model, we wanted to use a model that provided state of the art results while using limited computational resources. Thus, MobileNetV2 was deployed as another advanced architecture since it is known to be faster across the latency spectrum and perform well on mobile devices. Another reasons for choosing this model was that it works well with object detection. The main reason for these advantages provided by the MobileNetV2 architecture is its use inverted residuals [4] with depthwise convolutions. This allows for the effect of deep residual networks but in small blocks with bottlenecks requiring much less computational resources but showcasing very similar results to advanced networks. Similar to our ResNet50 implementation, transfer learning was used here as well for better performance. Since the original network was trained on the ImageNet dataset with 3 input channels and 1000 output classes, our network was finetuned by adding an additional convolutional layer at the beginning to inflate the input channels from 1 to 3, and an additional linear layer was added at the end to perform the classification task from the 1000 output classes from the MobileNetV2 model to the 3 output classes for our requirement. Similar to our previous adcanced deep learning model, Cross Entropy loss was used as the loss function and Adam was used as the optimizer with an initial learning rate of 0.0003. The batch size for the dataloaders was 32, and 3 epochs were used to train the model instead of 5 since the model converged very quickly for this use case. The architecture summary is as follows:

```
class MobileNet(nn.Module):
  def __init__(self, in_channels=1):
    super(MobileNet, self).__init__()
    self.conv = nn.Conv2d(in_channels,
    out_channels=3,
    kernel_size=5, padding=2)
    self.model = models.mobilenet_v2(
    pretrained=True)
    self.fc = nn.Linear(in_features=1000,
    out_features=3)

  def forward(self, x):
    x = self.conv(x)
    x = F.relu(self.model(x))
    return self.fc(x)
mobilenet = MobileNet()
```

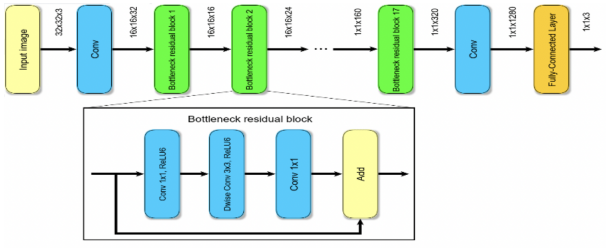

Fig. 11. MobileNetV2

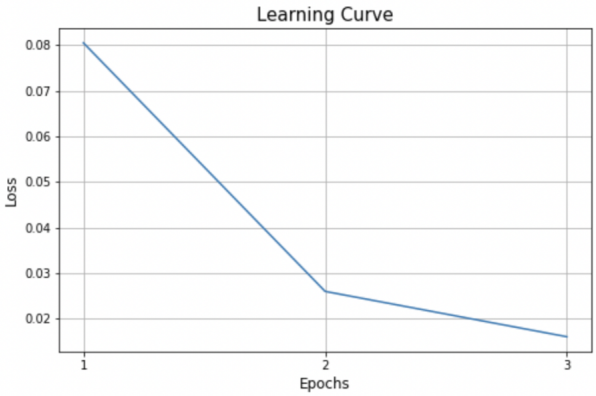

Fig. 12. MobileNetV2 Plot

V. Results

Dealing with a multi-class classification problem, we chose the Cross Entropy Loss as our loss function and we optimized each of the deep learning models using the Adam optimizer. The model's performance on the test set was measured using both Accuracy and F1 scores. Since, we have sampled enough images from all 3 classes and kept the dataset relatively balanced, both Accuracy and F1 scores are good measures of performance.

VI. Discussion

The data set has been trained by using various non DL and DL models. HOG features along with KNN classifier was used

Model Trained	Test Accuracy	F1-Score
HOG+KNN	82.75%	82.058%
CNN	93.44%	93.20%
ResNet50	98.82%	98.84%
MobileNetV2	98.93%	98.90%

TABLE I
MODEL PEFORMANCE SUMMARY

as a standard ML model which has a testing accuracy score of 82.75% and F1 score of 82.058%. As a base DL model, CNN classifier was chosen which obtained an accuracy score of 93.44% and F1 score of 93.20%. Furthermore, ResNet50 and MobileNetV2 were implemented as advanced DL models with an F1 score of 98.84% and 98.90% respectively.

A. Findings

Both the advanced DL models: ResNet50 and MobileNetV2 outperformed the base deep learning model (CNN) and base machine learning model (HOG feature + KNN Classifier) with an accuracy of 98.82% and 98.93% respectively, compared to that of CNN of 93.44%. The non deep learning model: KNN performed the worst with an accuracy score of 82.75%, but it still fared respectively well. This shows that the HOG descriptor proved effective in reducing input size but detecting important features. Still, the worse than DL performance is not surprising since the DL models used are the most popular networks in computer vision. Fine-tuned pre-trained MobileNetV2 outperformed ResNet50 by 0.11%.

B. Limitations and Ethical Considerations

On analysing the errors the incorrectly masked class shows some interesting characteristics. If a person wears a mask just over their mouth and leaves the nose open, then the image looks very similar to a person wearing the mask correctly. On the other hand, if a person wears the mask over the chin and leaves both the nose and the mouth exposed, then the resulting image is very close to a person not wearing a mask at all. Hence, for the incorrectly masked images, the models can miss-classify them into the other 2 classes, and just looking at the confusion matrix shows relatively balanced errors and hence does not highlight this issue until one does a manual deep dive.

C. Future Research Directions

The most promising areas of future research seem to be adding additional classes to split the incorrectly masked images into buckets like - incorrectly masked (nose uncovered), incorrectly masked (mouth uncovered), incorrectly masked (both nose and mouth uncovered). The confusion matrices from models trained using these labels would be the most informative about the bottlenecks in the model. After identifying these bottlenecks, one can further engineer features using custom edge and shape detection filters that can help mitigate some of these errors. Additionally, an ensemble of the strongest deep learning approaches could be created for a more robust classifier. Also, since our input images were clear faces with or without masks, future research may incorporate face detection algorithms to first filter the faces in a given scene and then apply our implementation on the filtered data.

VII. CONCLUSIONS

The impact of the COVID-19 pandemic has been drastic on many economic, and social fronts. The ideas of social distancing and face masking are the most effective non-vaccine approaches we have in our arsenal against this disease. This project aims to help institutions detect whether people are complying with healthy COVID practices. In most rudimentary applications of face mask detection, people have tried to detect the mere presence of a mask. However, as mentioned before, wearing a mask incorrectly is equally harmful as not wearing it at all. This is the idea that we tried to tackle using Machine Learning and Deep Learning approaches.

We attempted to build a KNN classifier with a HOG filter feature map of the input images as a simple ML approach. While extremely lightweight and fast, this approach did not achieve optimal performance. We went one step further and developed a 5 layer deep novel convolution neural network that gave a significant boost in our abilities to detect masking, but due to lack of resources we were unable to train deeper networks. Hence, we borrowed ideas from the SOTA deep learning models today - ResNet50 and MobileNetV2. The former helped us train deeper networks, while the latter can work well with smaller GPUs and addressed the computing power issue. Both SOTA models performed extremely well, and now our goal is to analyse the importance of additional classes, and create robust ensembles (as discussed in the section - *Future Research Directions*).

REFERENCES

[1] N. A. Nainan, J. H, R. Jalan, R. S. N, K. V. C, and S. P. Shankar, "Real time face mask detection using mobilenetv2 and inceptionv3 models," in *2021 IEEE Mysore Sub Section International Conference (MysuruCon)*, 2021, pp. 341–345.

[2] L. Nguyen, T. N. Cao, L. Huynh-Anh, and H. Dang-Ngoc, "An embedded machine learning system for real-time face mask detection and human temperature measurement," in *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, 2021, pp. 17–22.

[3] D.-L. Nguyen, M. D. Putro, and K.-H. Jo, "Facemask wearing alert system based on simple architecture with low-computing devices," *IEEE Access*, vol. 10, pp. 29 972–29 981, 2022.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2018. [Online]. Available: https://arxiv.org/abs/1801.04381