Face Mask Detection

Udacity Machine Learning Engineer Nanodegree Capstone Report

DaeMyung Kang
Sep 06, 2020

I. Definition

Project Overview

In this project, we performed Face Mask Detection using Deep Learning. These days, COVID-19 is very serious problem in the world now. Wearing mask is most helpful to prevent the spread of COVID-19.[1].

To prevent spreading of COVID-19, It is very helpful to encourage to wear a mask in public transportation. but It is hard to tell someone that doesn't wear a mask. so automated notification system that tells someone who don't wear a mask to wear a mask. Deep Learning can make this.

Let's find someone who don't wear a mask. it is very hard using human eyes.

There are some face mask datasets in Kaggle. I used this dataset.

Dataset Link: https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset?

Problem Statement

This is a binary classification problem.(wearing a mask or not), Inputs are image of peoples, and the goal is to detect peoples wear a mask or not. Given an image of some peoples, the model will detect the persons to wear a mask or not.

Metric

We used the F1 Score as the model final assessment. Because the real data is imbalanced. Most of people will wear a mask to protect them from COVID-19, F1-score is better than Accuracy in this situation.

	True Condition Positive	**True Condition Negative**
Predicted Condition Positive	TP = True Positive	FP = False Positive
Predicted Condition Negative	FN = False Negative	TN = True Negative

Accuracy, Precision and Recall can be derived from the above matrix.

- Accuracy = (TP + TN)/(TP + FP + TN + FN)
- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1_SCORE = 2 * (Precision * Recall)/(Precision + Recall)

II. Analysis

Data Exploration

For this project we used face mask dataset in Kaggle. The dataset contains 11792 images.

The datasets have only face images and tag as a directory name. there are 3 main directories and each one has two tag directories.

		Count
Train	WithMask	5000
	WithoutMask	5000

Test	WithMask	483
	WithoutMask	509
Validataion	WithMask	400
	WithoutMask	400

The images size are diverse. Below images are each sample for WithMask and WithoutMask.

```
train_withmask = list_imagenames("Train/WithMask", 0)
train_withoutmask = list_imagenames("Train/WithoutMask", 1)
test_withmask = list_imagenames("Test/WithMask", 0)
test_withoutmask = list_imagenames("Test/WithoutMask", 1)
validation_withmask = list_imagenames("Validation/WithMask", 0)
validataion_withoutmask = list_imagenames("Validation/WithoutMask", 1)

imagenames = []

# Make All Images as one
# To test rightly, we will chose test set randomly.
for c in [train_withmask, train_withoutmask, test_withmask, test_withoutmask,
validation_withmask, validataion_withoutmask]:
    print(len(c))
    imagenames.extend(c)

print(len(imagenames))
```

```
5000
5000
483
509
400
400
11792
```

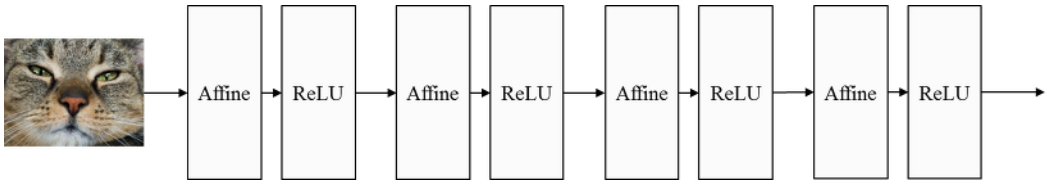
Algorithm and techniques

In this project, we use CNN as the structure of deep learning model and Transfer Learning as a learning method. Below is a brief description of our CNN and Transfer Learning.

Convolution Neural Network

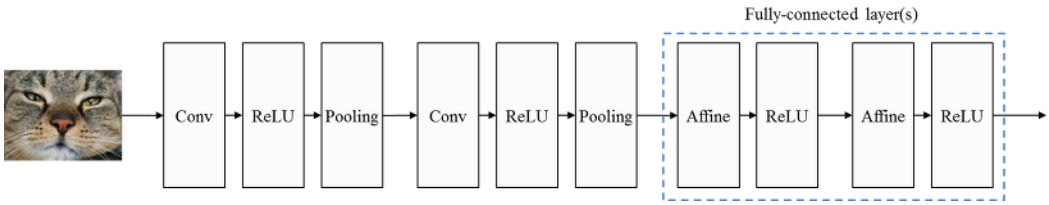
CNN was first introduced in (LeCun et al., 1989) to process images more effectively by applying filtering techniques to artificial neural networks, and later in (LeCun et al., 1998), a type of form currently used in deep learning. CNN was proposed. Existing filtering techniques processed images using fixed filters as shown in Figure 1. The basic concept of CNN is "Let each element of the filter expressed as a matrix be automatically learned to be suitable for data processing". For example, when we want to develop an image classification algorithm, we can improve the classification accuracy by using a filtering technique. However, one problem is that you have to decide which filter to use in the algorithm through human intuition or iterative experimentation. If CNN is used in this situation, the algorithm can automatically learn a filter that maximizes image classification accuracy.

A typical artificial neural network is a structure in which several layers defined by the synthesis of a fully-connected operation specified as affine and a nonlinear activation function such as ReLU as shown in Figure 1 are stacked.



[Figure 1]

As shown in Figure 2, CNN adds a new layer called a convolutional layer and a pooling layer before the fully-connected layer to apply a filtering technique to the original image, and then perform a classification operation on the filtered image. It is configured to be.

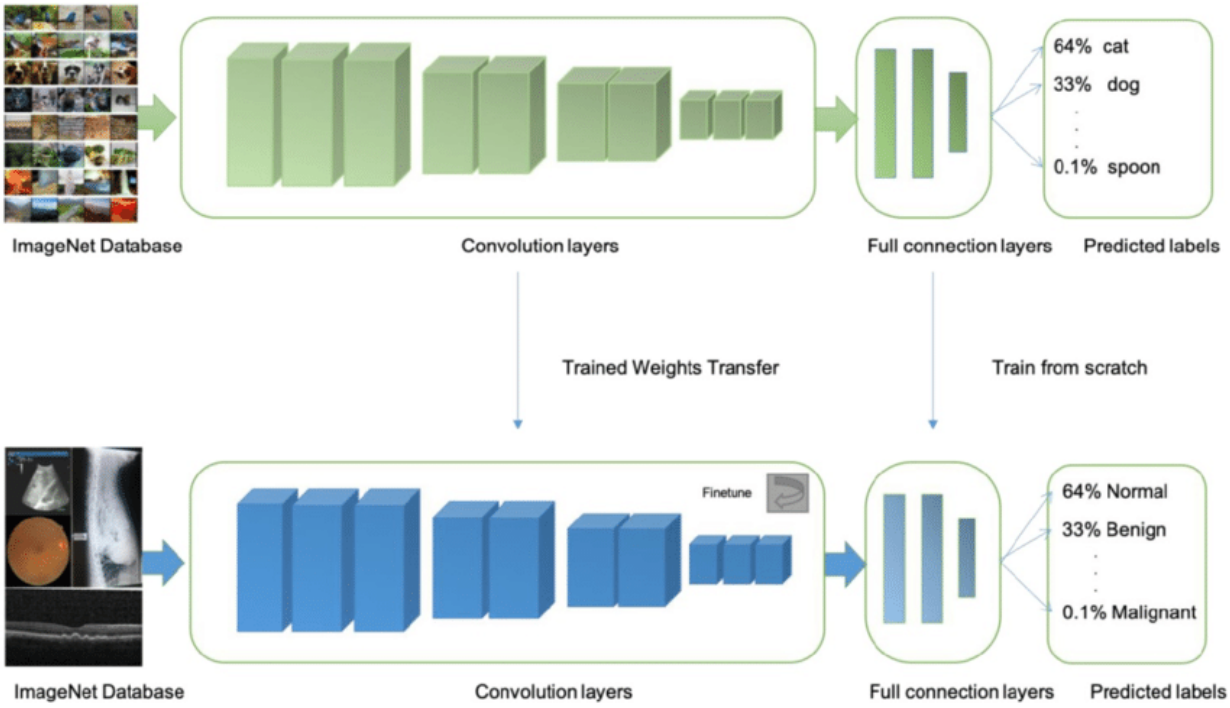


[Figure 2]

I choose CNN Model as a solution.

Transfer Learning

Transfer learning is one of the popular methodologies in the field of computer vision because high accuracy can be achieved in a relatively short time (Rawat & Wang 2017). With transfer learning, even when solving problems different from those that have already been learned, it is possible to apply patterns that have already been learned instead of building models from the bottom up.



Transfer learning in computer vision mainly refers to the use of pre-trained models. A pre-trained model is a model that has already been trained with large-sized data similar to the problem I am trying to solve.

learning the dataset of the target task using finetuning based on the pretrained backbone is the best way to gain performance. There is a paper "Pre-Training Can Improve Model Robustness and Uncertainty"(https://arxiv.org/abs/1901.09960), this project uses the Transfer Learning method to train the model to classify Face Mask Dataset. Performance is improved by applying several finetuning methods based on the most suitable existing backbone

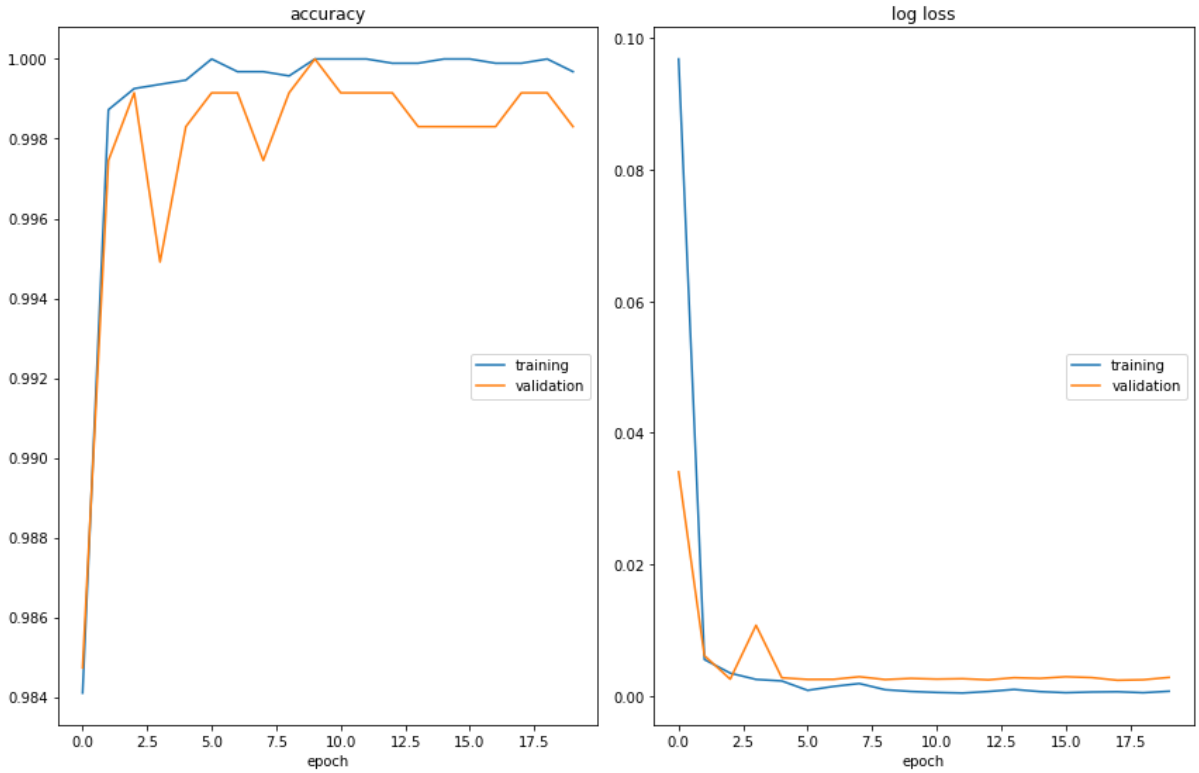
Benchmark

We create a simple CNN model, train it, and use it as a benchmark model. This model is a very simple model with two CNN layers. Since the resolution of the image is not large, I decided that

deep learning model with small layers would be a good baseline.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 220, 220]           2,432
         MaxPool2d-2         [-1, 32, 110, 110]               0
            Conv2d-3         [-1, 64, 108, 108]          18,496
         MaxPool2d-4           [-1, 64, 54, 54]               0
            Conv2d-5          [-1, 128, 52, 52]          73,856
         MaxPool2d-6          [-1, 128, 26, 26]               0
            Conv2d-7           [-1, 64, 24, 24]          73,792
         MaxPool2d-8           [-1, 64, 12, 12]               0
            Linear-9                 [-1, 512]       4,719,104
          Dropout-10                 [-1, 512]               0
           Linear-11                 [-1, 256]         131,328
           Linear-12                   [-1, 2]             514
================================================================
Total params: 5,019,522
Trainable params: 5,019,522
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 25.55
Params size (MB): 19.15
Estimated Total Size (MB): 45.27
----------------------------------------------------------------
```

Apply the same process as Data Preprocessing defined in Methodology below, and check the result of applying Simple CNN Model only above model. The optimizer for the Simple CNN Model is adam, and the learning rate is set to 0.001. The result of training 20 epochs of the above benchmark model is as follows. The accuracy is 0.991518. It is also good score.



III. Methodology

Data Processing

There are generally Two methods: Resize, Augmentation.

Resize

Each image is resized to 224x224. CNN requests the same size input, first of all, we should convert all images as the same size, we make them as 224x224 square.

Augmentation

When dataset is small, to prevent overfitting, we can apply augmentation. Randomly, change the original image and make it as new image. But this dataset already has augmented data. The filename has prefix "Augmented_.". so we don't apply augmentation for this dataset

There is good augmentation tool at https://github.com/aleju/imgaug

[https://github.com/aleju/imgaug]

Implementation

We process following steps

- Backbone selection for Transfer Learning
- Tuning
- Evaluation

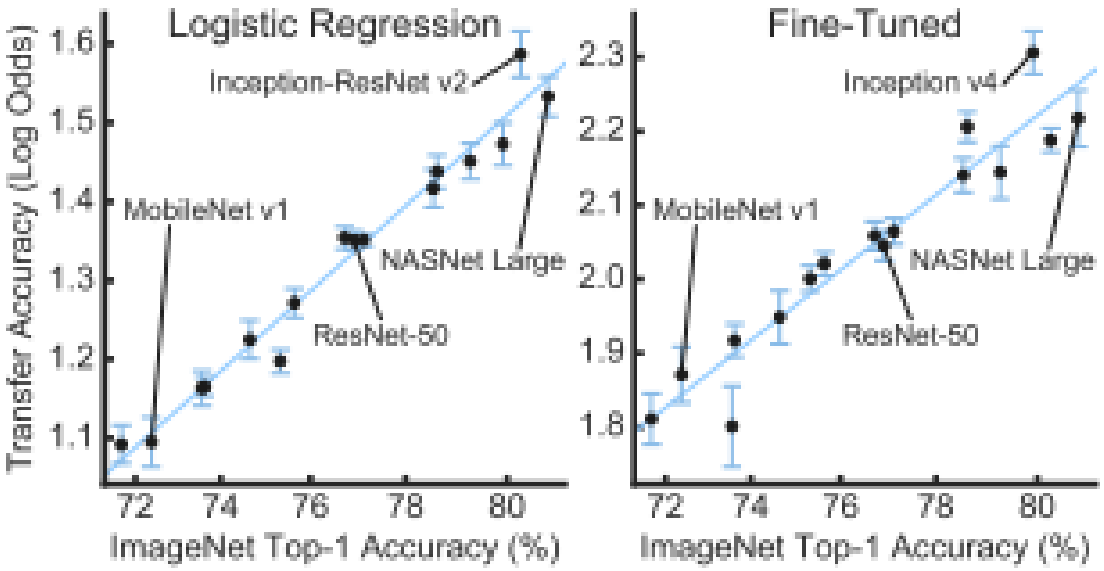
Backbone Selection for Transfer Learning

The final model greatly depends on the backbone model. To get good performance from trained model, It is important to choose a good backbone to train. When choosing a pretrained backbone model, you need to determine whether the model is trained from the appropriate dataset and how well the model performs.

Backbone model dataset and Performance

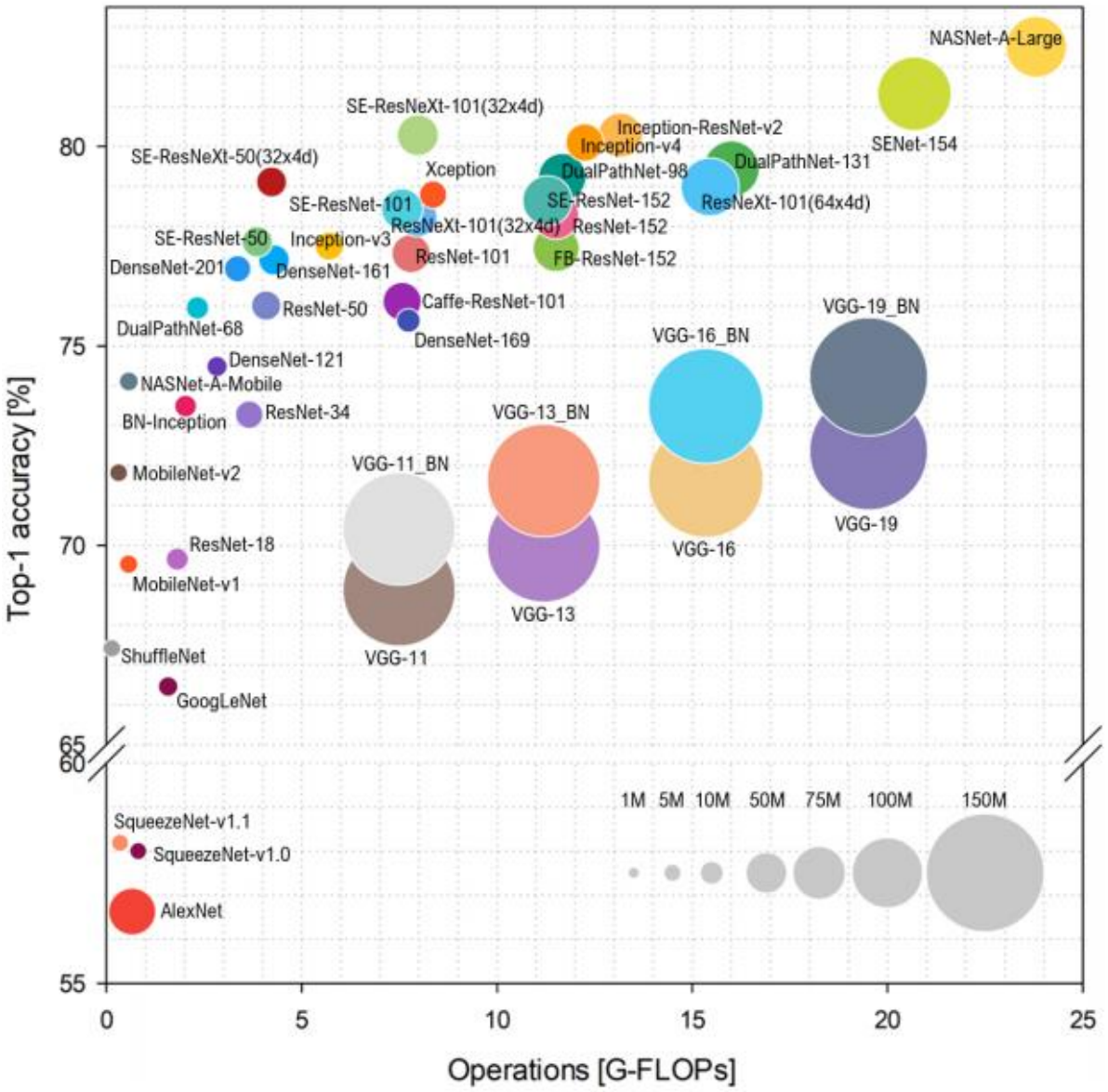
Based on Do Better ImageNet Models Transfer Better?(https://arxiv.org/pdf/1805.08974.pdf), It appeals following things.

- The performance is better if transfer learning is performed using a model with better performance for ImageNet as a backbone.
- In this paper, it compared 12 datasets and 16 classification models.
- The correlation between the actual backbone and the transfer task was very high



Backbone Selection

There are many reference models in this paper "Benchmark Analysis of Representative Deep Neural network Architectures"(https://arxiv.org/pdf/1810.00736.pdf)



Large model needs bigger memory and more time to train, so we exclude large model because of complexity and scalability issues. We use resnet50 and mobilenet_v2 as pretrained backbones that provided by pytorch. It is to detect to wear a mask or not. So it should be small to use in buses or other public place easily.

Traning & Evalution

The steps are following.

- Data preprocessing
- In order to determine the earning rate, backbone, and optimizer, we train 3 epochs each

to determine the hyperparameter that works best.

- Based on the hyperparameter determined above, the final model is determined and the model is trained.
- Proceed to the detailed hyperparameter tuning of the model.
- Evaluate the final model.

Refinement

To find a optimized model, it is important to find an appropriate hyperparameter early. And deep learning training is very expensive. In order to find the right backbone and hyperparameter for the dataset, the initial 3 epcoh training is done according to the following conditions.

- backbone = ['mobilenet_v2', 'resnet50']
- learning rate = [0.01, 0.01, 0.001]
- optimizer = ['adam', 'sgd']
- backbone last layer = ['fc']

Based on the combination of the above conditions, we train the model 3 epochs each for 12 learning conditions. It is essential to find good conditions because the learning performance of hyperparameters is very different. The backbone last layer is a way to replace the last layer of the backbone model.

	model	optimizer	learning_rate	validation_loss	train_loss	val_accuracy	train_accuracy
0	MobileNetV2	sgd	0.010	0.005754	0.005487	0.997455	0.998198
1	ResNet	sgd	0.010	0.005877	0.003690	0.997455	0.999046
2	ResNet	adam	0.001	0.011965	0.026002	0.995759	0.990990
3	MobileNetV2	sgd	0.100	0.015895	0.036916	0.994911	0.988764
4	ResNet	sgd	0.001	0.016357	0.021638	0.995759	0.995124
5	MobileNetV2	adam	0.001	0.017049	0.012766	0.996607	0.995230
6	MobileNetV2	sgd	0.001	0.017675	0.023220	0.995759	0.993852
7	ResNet	sgd	0.100	0.059682	0.028410	0.980492	0.990248
8	MobileNetV2	adam	0.010	0.141388	0.109975	0.966921	0.962900
9	ResNet	adam	0.100	0.946574	0.149217	0.645462	0.948060

Based on the above data, we trained two models in total. But almost of them shows above 90% accuracy.

Backbone	Optimizer	Learing Late
Resnet	sgd	0.01
MobileNetV2	sgd	0.01

IV. Results

Model Evaluation and Validation

We trained three models, Simple CNN Model as a benchmark model, and MobileNetV2 and Resnet50. We trained 20 epochs for each model. After training each model, we applied the test dataset to produce test loss, accuracy and F1-Score indicators. The result is shown below.

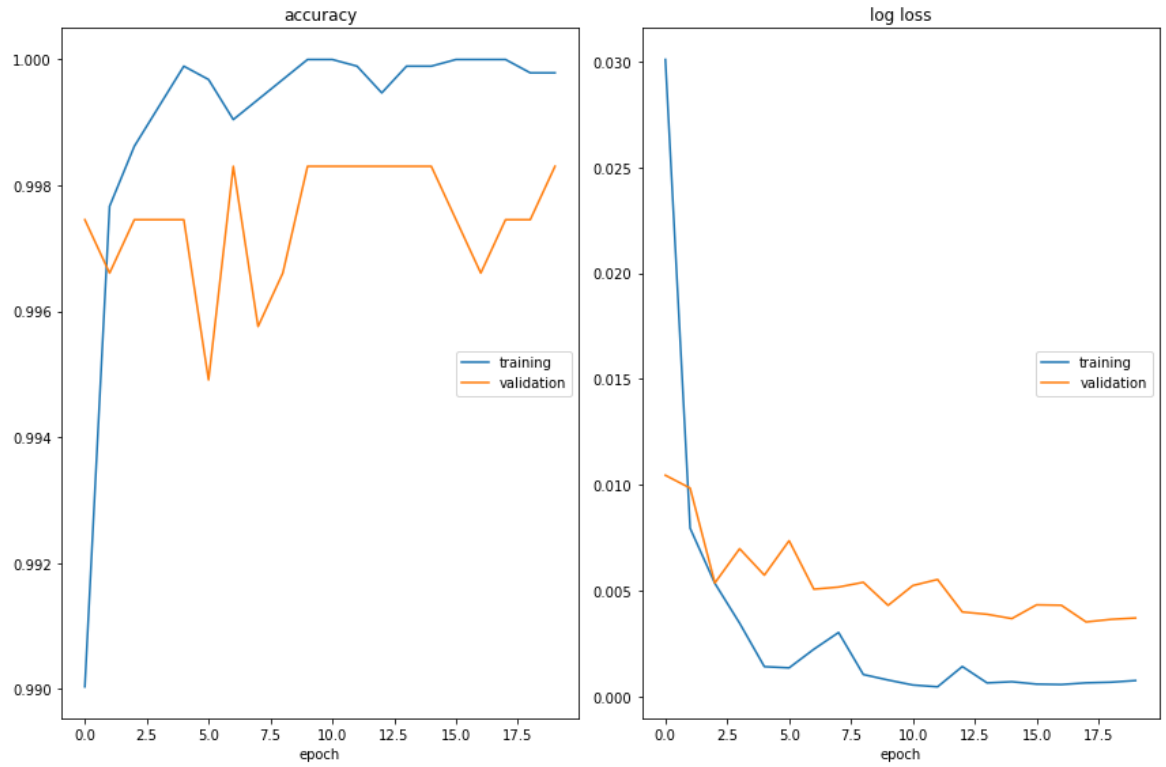
	model	last_layer	train_loss	val_loss	test_loss	train_accuracy	val_accuracy	test_accuracy	test_f1
0	SimpleCNNModel	None	0.008212	0.034712	0.025196	0.997456	0.989822	0.991518	0.991518
1	MobileNetV2	fc	0.000795	0.004319	0.006442	1.000000	0.998304	0.997455	0.997455
2	ResNet	fc	0.000666	0.002652	0.004008	1.000000	1.000000	0.999152	0.999152

Even the benchmark model performance is over 99%, all models with Transfer Learning show better performance compared to benchmark model.

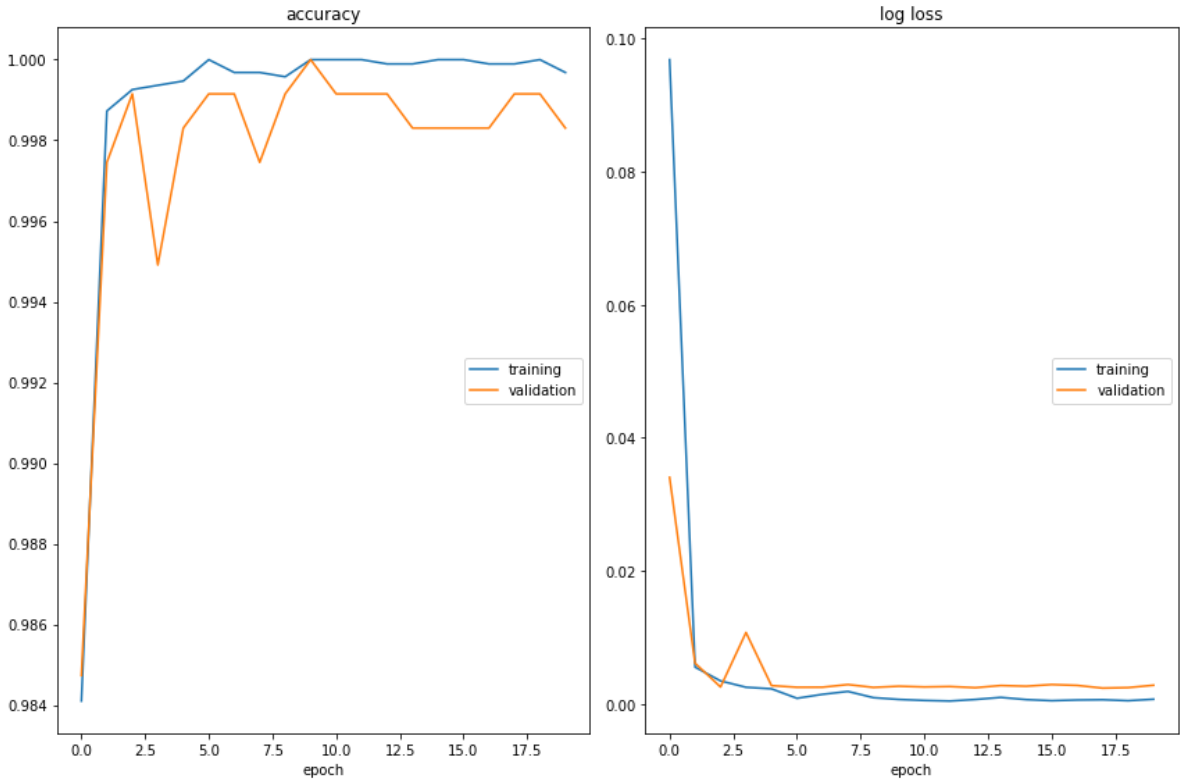
There was no significant difference in MobileNetV2 and Resnet50.but resnet50 shows more resistant to overfitting than the using MobileNetV2.

Due to the limitation of learning resources, we could not confirm adaptability to overfitting due to the small number of trainings as small as 10 epochs. And just using fully connected layer as last layer However, if you proceed further, we can test batch normalization and dropout.

MobileNetV2



Resnet50



Justification

Models trained with Transfer Learning have up to 1% higher accuracy when compared to the Simple CNN Model. Even it is very small performance difference. To detect to wear a mask or not is just binary problem. And tuned model shows a little bit better accuracy.

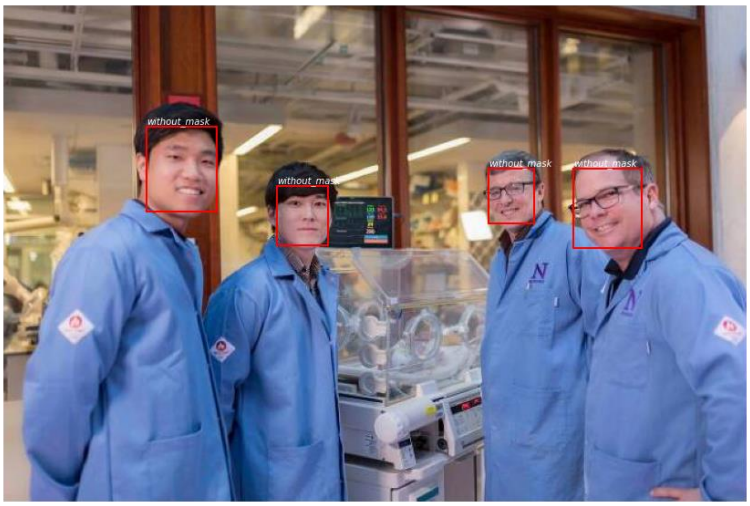
	3 epoch validation_accuracy	20 epoch
MobileNetV2	0.997455	0.998304
Resnet50	0.997455	1.000000

V. Conclusion

Visualization

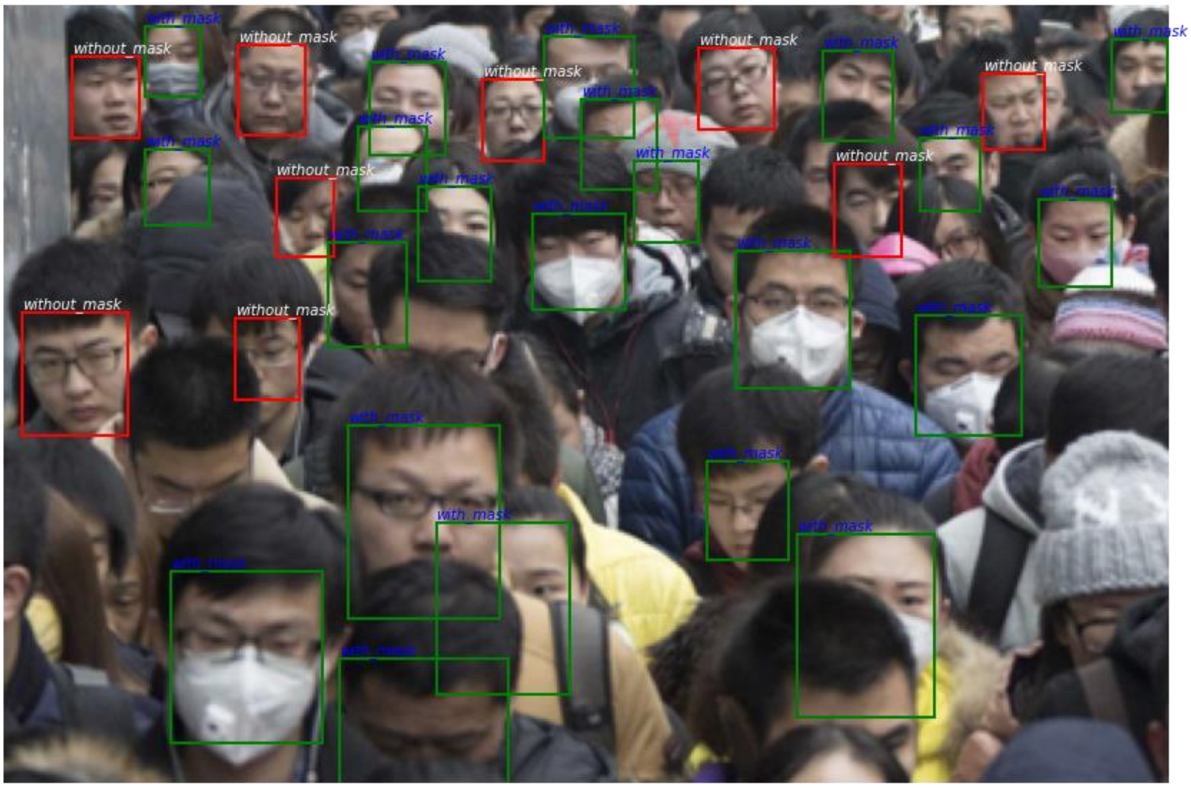
We tested some images with Resnet50 model. Red Box shows face which doesn't wear a mask .Green Box shows face which wear a mask.

1] There are 4 people not to wear a mask.



2] There is one person not to wear a mask.

3] There are many people to wear or not.



It shows misrecognition when there are some objects that partially covers the face.

Reflection.

The goal of this project is to detect people to wear a mask or not. We created a deep learning model to detect it. We trained the model using Transfer Learning. MobileNetV2 and Reset50 based

model shows in 99.999% accuracy, which is more accurate than the benchmark CNN Model.

We tested several deep learning models for Transfer Learning and put the results in a project report. We can tune model to modify the hyperparameter(learning rate, backbone, optimizer, etc) the difference of these hyperparameters shows different performance.(In this project,, the gap is not too much.)

Unfortunately, the limit of GPU resources and time limitations, we couldn't test various methods of transfer learning(we could just use 40Hours GPU in a week in Kaggle).

I also can't apply batch normalization and dropout for this project.

Improvement

There are two problems with the project. First of all, It shows low performance when there is an object to cover someone's face. It misrecognizes the object as mask. The second is not to apply batch normalization and dropout.

To get more performance even there are some objects. I think to learn the mask feature with face together. It can prevent to misrecognize other object as mask.

Improvement

- Do Better ImageNet Models Transfer Bettter? https://arxiv.org/pdf/1805.08974.pdf
- Benchmark Analysis of Representative Deep Neural Network Architectures
  - https://arxiv.org/pdf/1810.00736.pdf