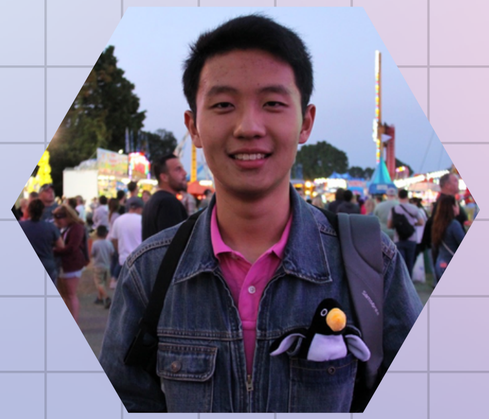
MSCA 31009 2 Final Project

June 2, 2021

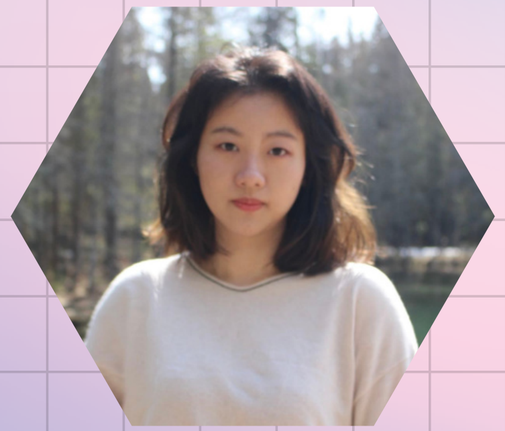
Application of deep learning for face mask detection in our COVID world

Team 15: Jeffery (Yan) Chen, Olivia (Caixin) Yang, Wilson McDermott, Zhizhuo Li

Our Team

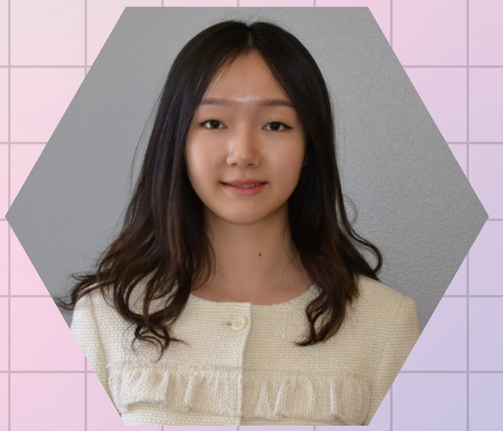


Jeffrey Chen



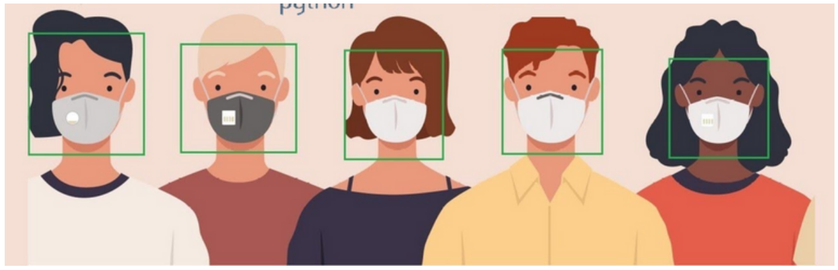
Zhizhuo Li

Wilson McDermott



Caixin (Olivia) Yang

Today's Agenda

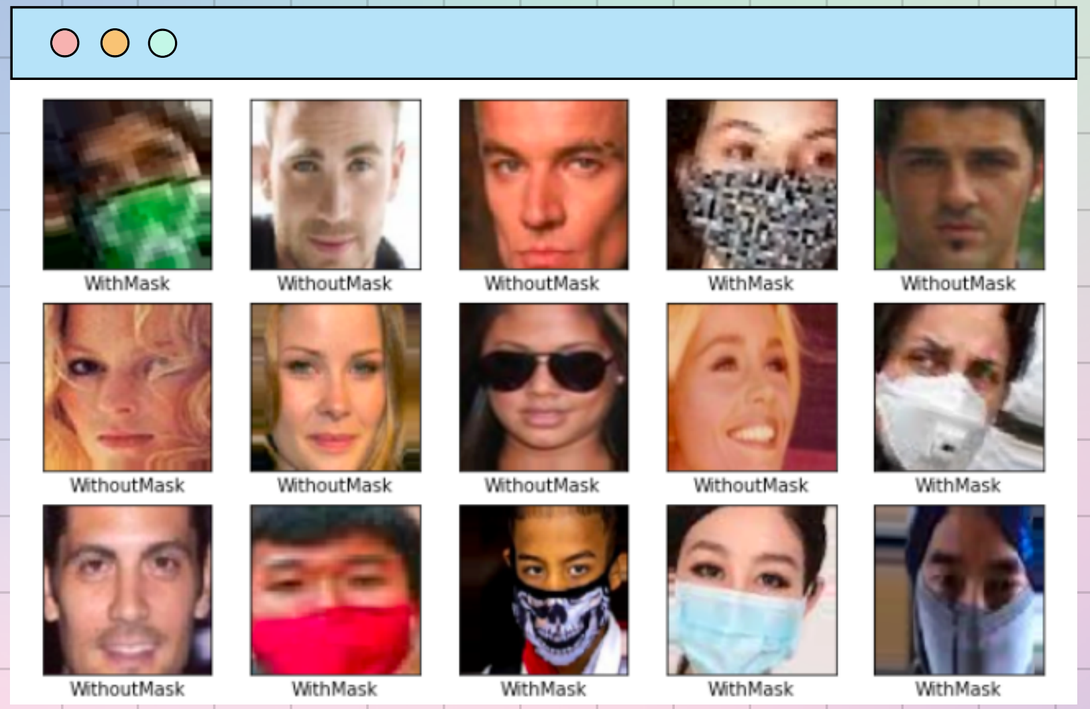


1. Problem Statement
2. Our Data
3. CNN Models
4. MobileNet Model
5. VGG Models
5. Model Performance
6. Conclusion & Future Work

Problem Statement

As mask regulations continue to evolve it is more important than ever for governments and businesses to recognize individuals following and not following the appropriate safety guidelines. Based thereon, our goals was to create a solution to better enable the aforementioned stakeholders to identify which individuals are and are not wearing masks

Our Data

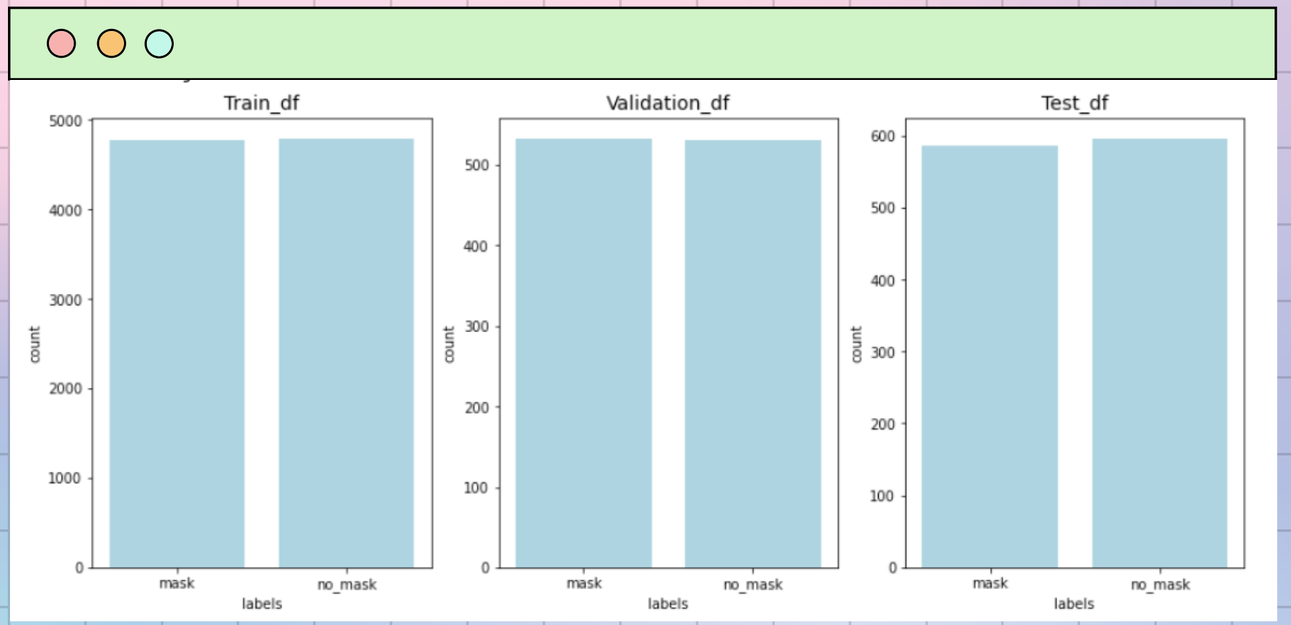


Face Mask Detection Dataset
https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset

Train:
- No mask 4785
- Mask 4773

Validation:
- No mask 536
- Mask 526

Test:
- No_mask 597
- Mask 584



Haar Cascade

haarcascade_frontalface_default.xml

Using haar cascade to detect faces

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.

- It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images.
- Used to detect objects in other images.
- We used a Haar Cascade Model trained to detect faces in order to obtain the bounding box coordinates of faces in an image.

CNN Models

Convolutional Neural Networks

What is a CNN?

Assign importance (learnable weights and biases) to various aspects / objects in the image

Classify

CNN Model 1:

Feature Extraction

Classification

Input
(128,128,3)

Conv2D
32, (3, 3)
Sigmoid

Maxpooling2D
(2,2)

Conv2D
64,(3,3)
Sigmoid

Maxpooling2D
(2,2)

Conv2D
64,(3,3)
Sigmoid

Maxpooling2D
(2,2)

Flatten

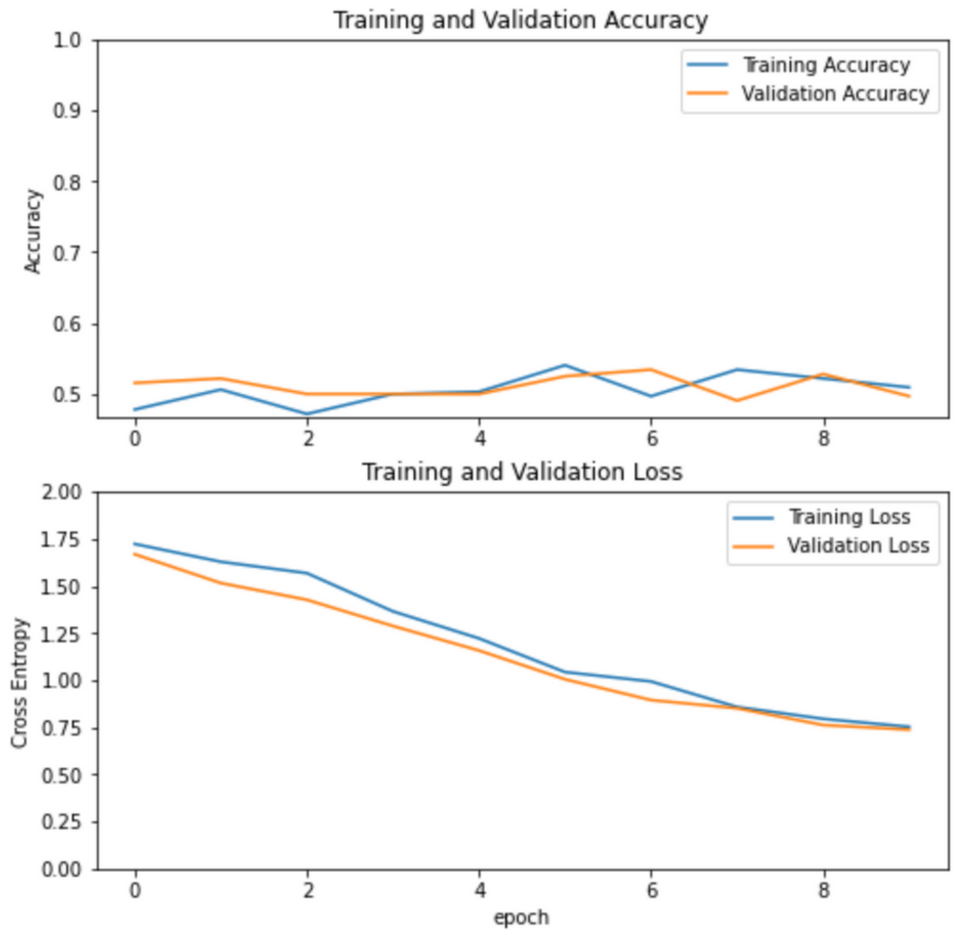
Dense
128
Sigmoid

Dense
2
Sigmoid

Performance

CNN Model 1

Test Accuracy: **49.2%**



CNN Model 2:

Feature Extraction | Classification

Input
(128,128,3)

Relu

Conv2D
32, (3, 3)

Maxpooling2D
(2,2)

Relu

Conv2D
64,(3,3)

Maxpooling2D
(2,2)

Relu

Conv2D
64,(3,3)

Maxpooling2D
(2,2)

Flatten

Relu

Dense
128

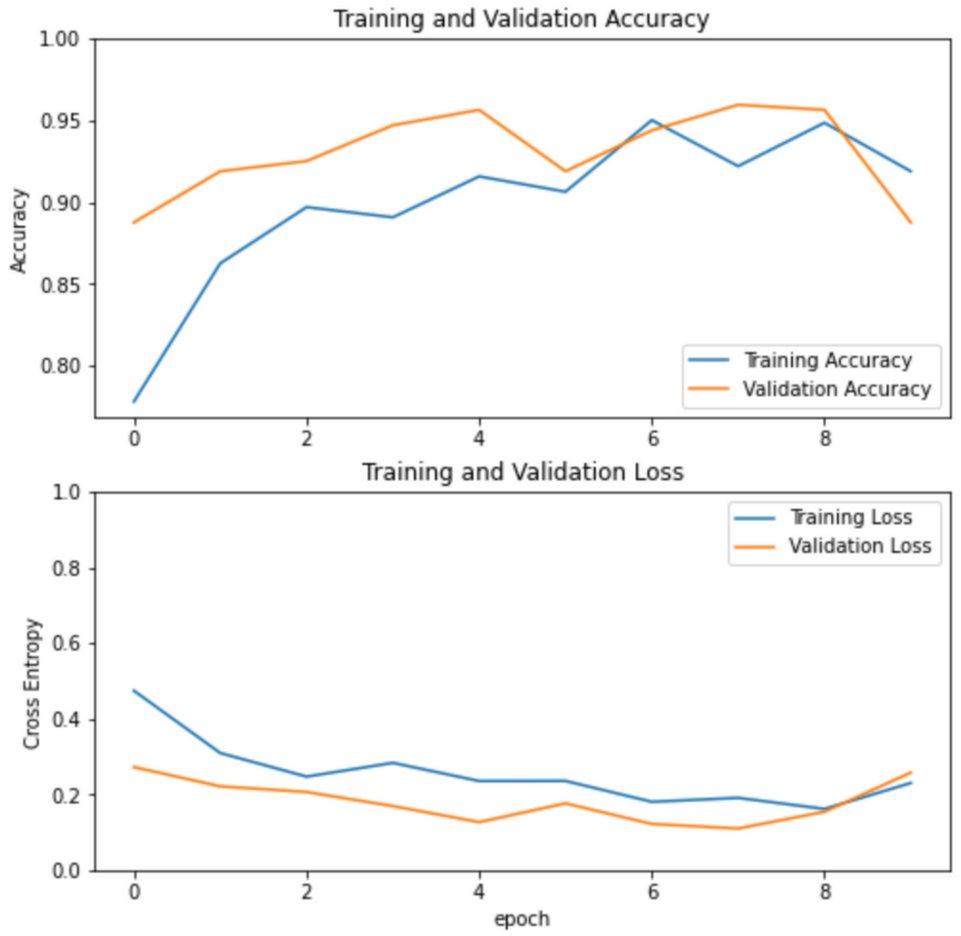
Sigmoid

Dense
2

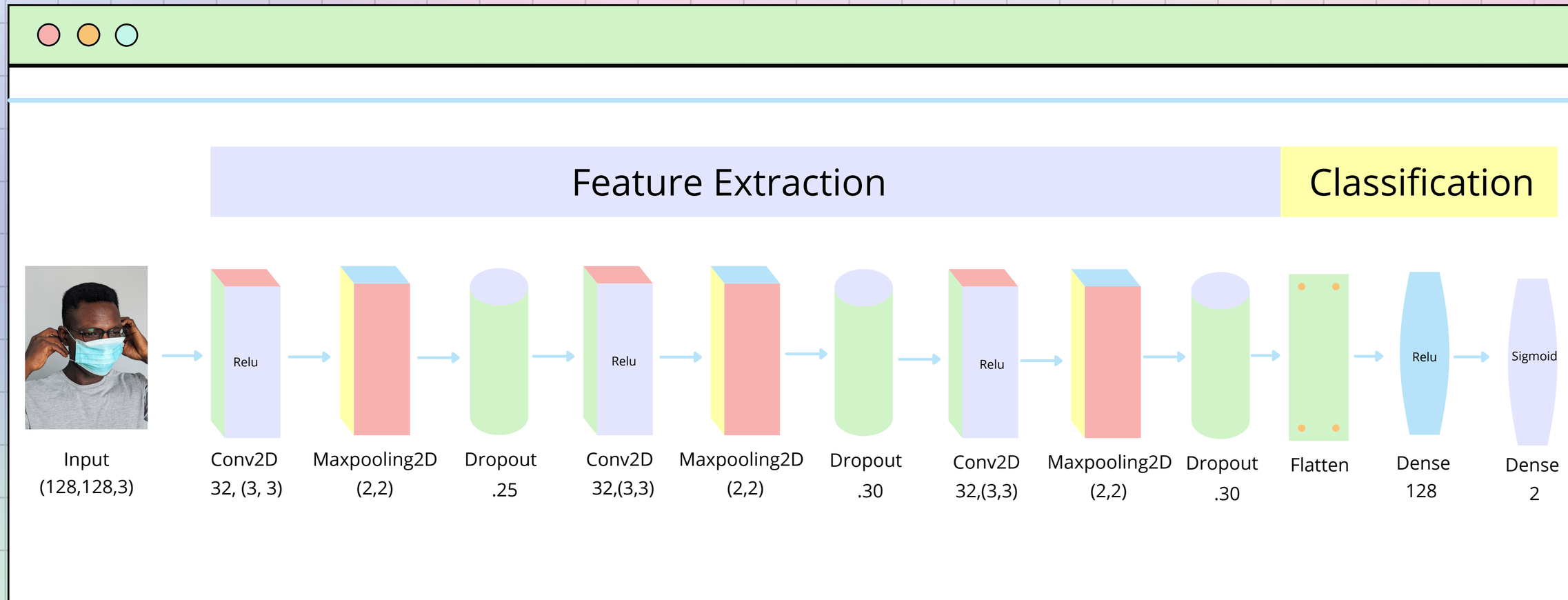
Performance

CNN Model 2

Test Accuracy: **86.4%**



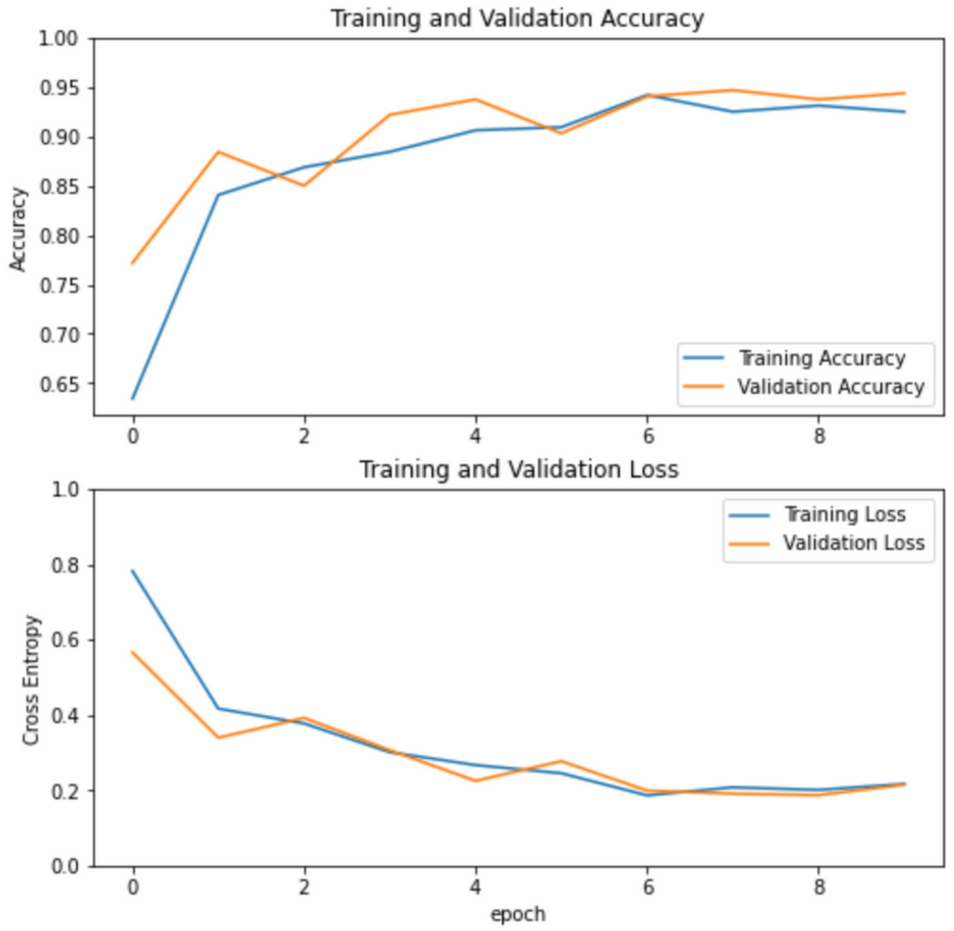
CNN Model 3:


Feature Extraction
Classification
Input
(128,128,3)
Relu
Conv2D
32, (3, 3)
Maxpooling2D
(2,2)
Dropout
.25
Relu
Conv2D
32,(3,3)
Maxpooling2D
(2,2)
Dropout
.30
Relu
Conv2D
32,(3,3)
Maxpooling2D
(2,2)
Dropout
.30
Flatten
Relu
Dense
128
Sigmoid
Dense
2

Performance

CNN Model 3

Test Accuracy: **93.8%**

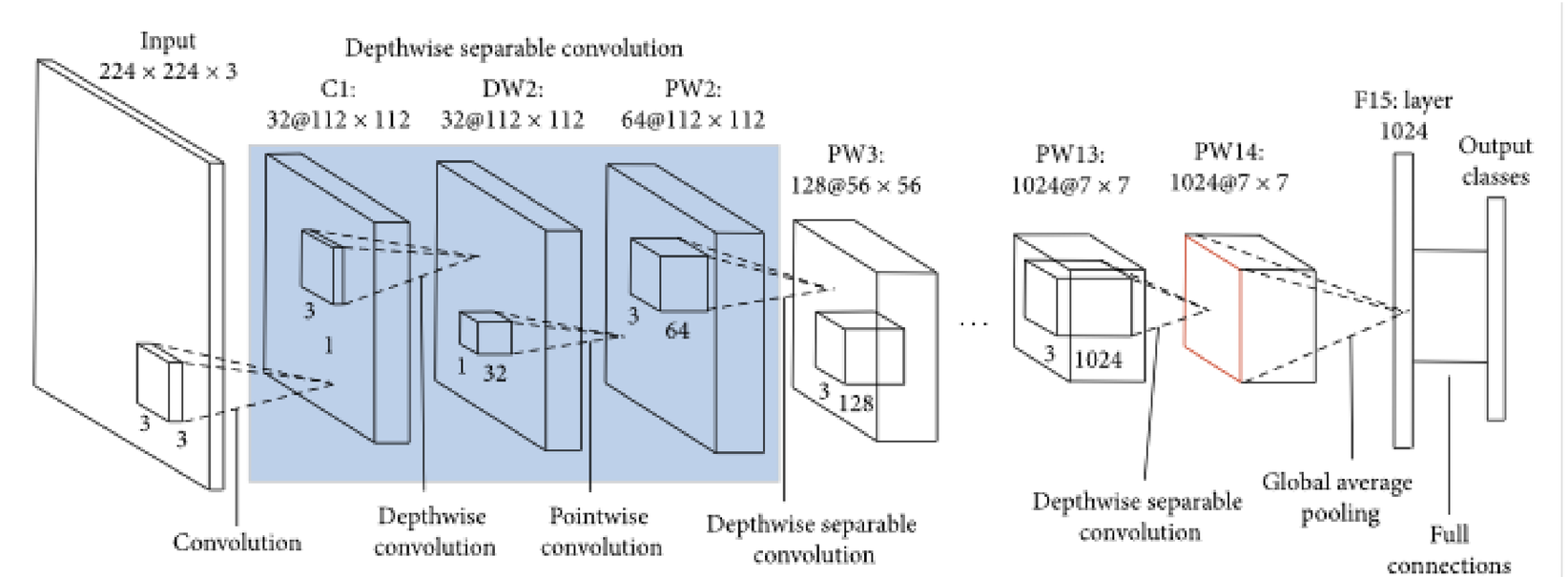


CNN Model Performance

	Test Accuracy	
Model 1:	49.2%	3 convolution layers Activation: only sigmoid
Model 2:	86.4%	3 convolution layers Activation: relu/sigmoid
Model 3:	93.8%	3 convolution layers 3 dropout layers Activation: relu/sigmoid

MobileNet Models

MobileNet Model



- depthwise separable convolutions
  - depthwise convolution
  - pointwise convolution

- Faster than the network with regular convolutions

MobileNet Model

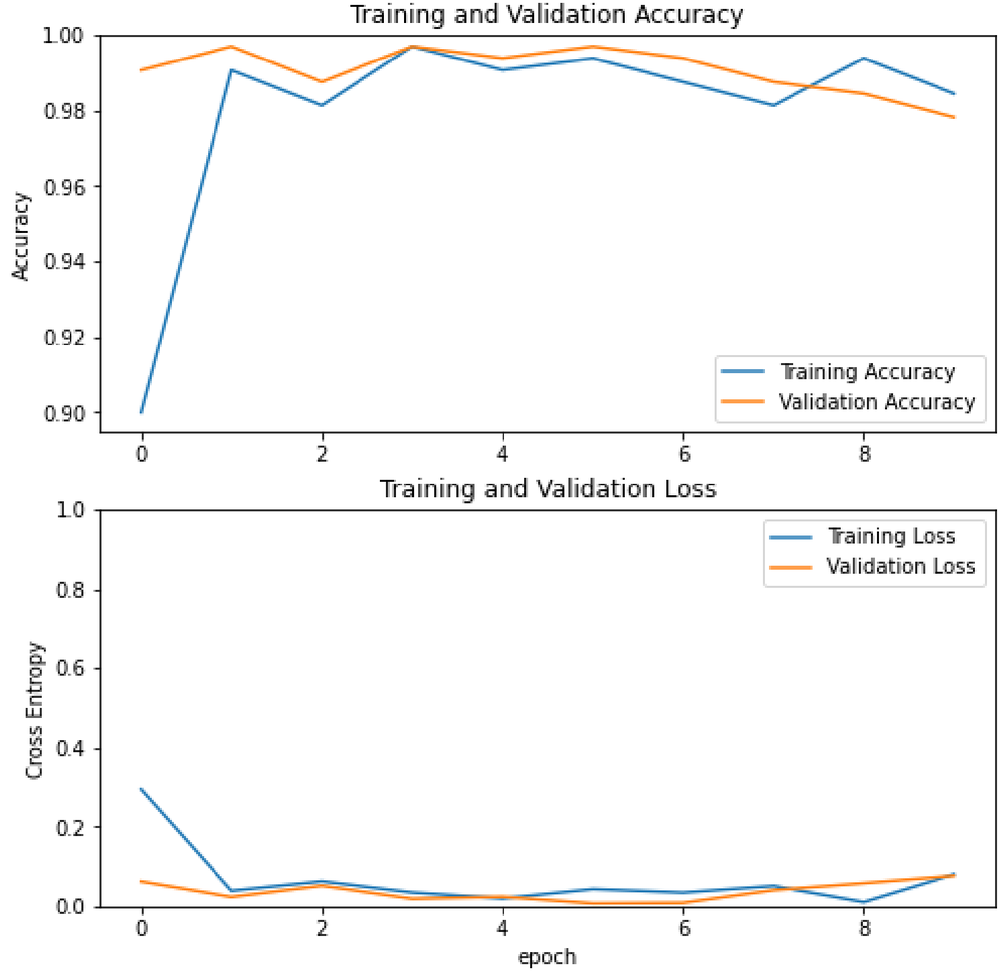
Type / Stride	Filter Shape
Conv / s2	$3 \times 3 \times 3 \times 32$
Conv dw / s1	$3 \times 3 \times 32$ dw
Conv / s1	$1 \times 1 \times 32 \times 64$
Conv dw / s2	$3 \times 3 \times 64$ dw
Conv / s1	$1 \times 1 \times 64 \times 128$
Conv dw / s1	$3 \times 3 \times 128$ dw
Conv / s1	$1 \times 1 \times 128 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw
Conv / s1	$1 \times 1 \times 128 \times 256$
Conv dw / s1	$3 \times 3 \times 256$ dw
Conv / s1	$1 \times 1 \times 256 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw
Conv / s1	$1 \times 1 \times 256 \times 512$
$5\times$ Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw
Conv / s1	$1 \times 1 \times 512 \times 1024$
Conv dw / s2	$3 \times 3 \times 1024$ dw
Conv / s1	$1 \times 1 \times 1024 \times 1024$
Avg Pool / s1	Pool 7×7
FC / s1	1024×1000
Softmax / s1	Classifier

- Depthwise Separable Convolutions
- Convolution layer
- Average Pooling layer
- Fully Connected layer
- Total params: 3,261,634
- Trainable params: 32,770
- Non-trainable params: 3,228,864

Performance

MobileNet

Test Accuracy: **98.6%**

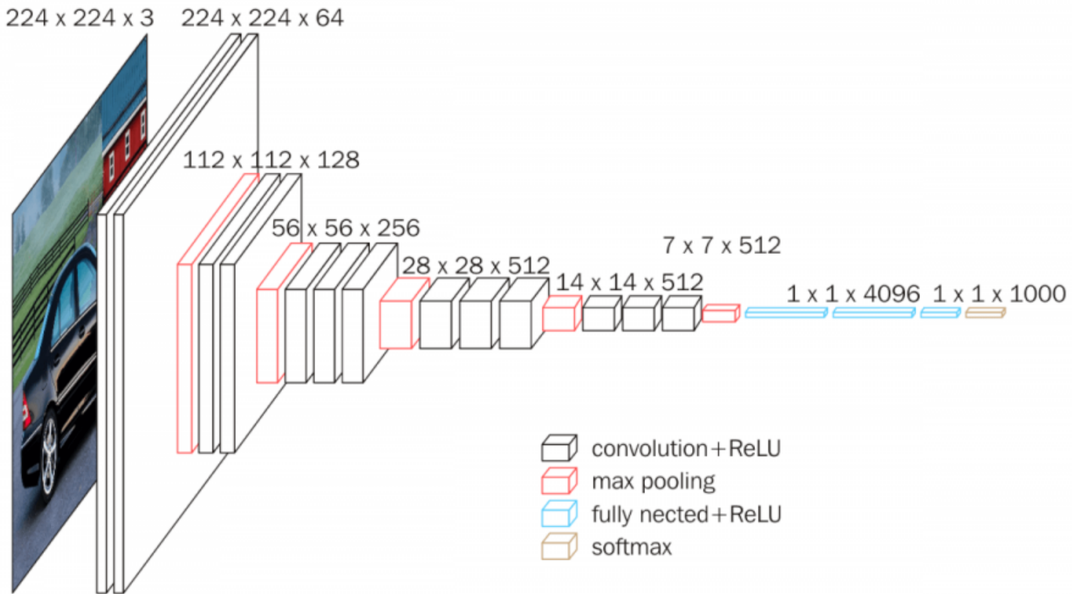


VGG Models

Transfer Learning & Fine-tuing

Base Model

Visual Geometry Group (VGG)



1. Trained on ImageNet dataset
2. Improvements based on AlexNet from large-sized kernels to multiple 3x3 kernels

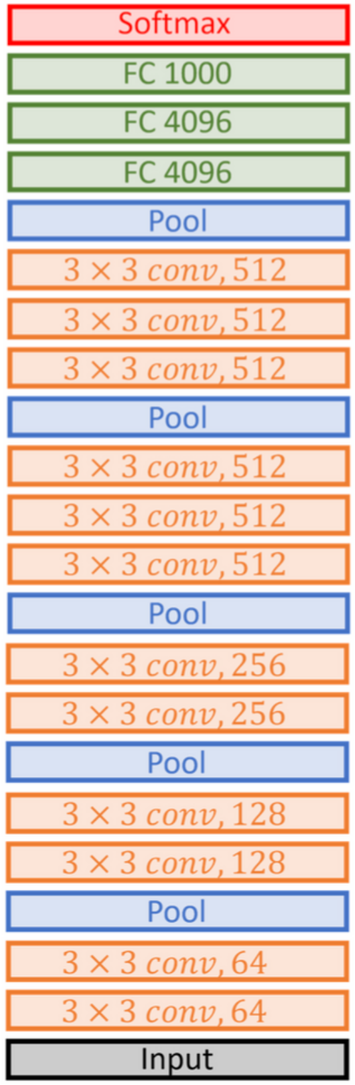
Over 14 million images& 1000 classes

Fixed input size 224x224 RGB image

Painfully slow to train & Large Architecture

VGG 16

1. 16 weighted layers
2. Relatively Large Network Architecture
3. ReLU non-linearity



CONVOLUTIONAL LAYERS X12

- 3X3 Filter Size
- Stride 1

MAXPOOL LAYERS X5

- 2X2 Filter Size
- Stride 2

FULLY CONNECTED LAYERS X3

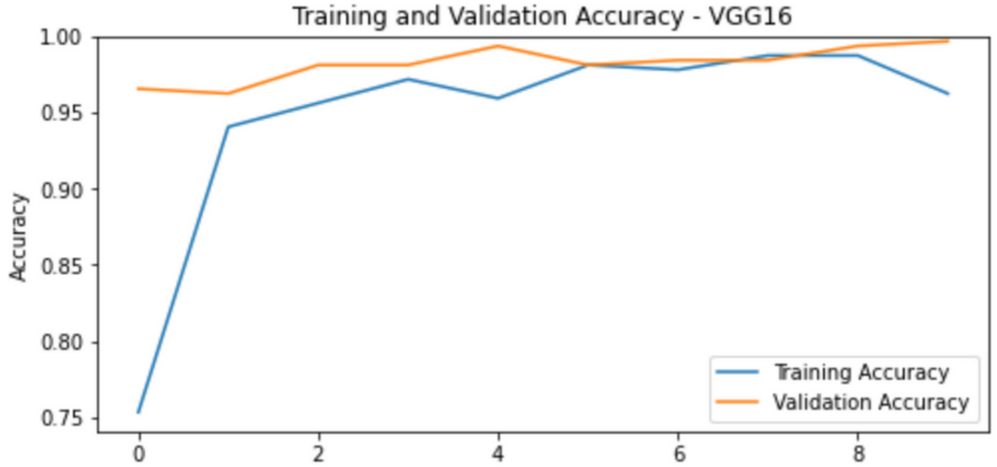
SOFTMAX LAYER X1

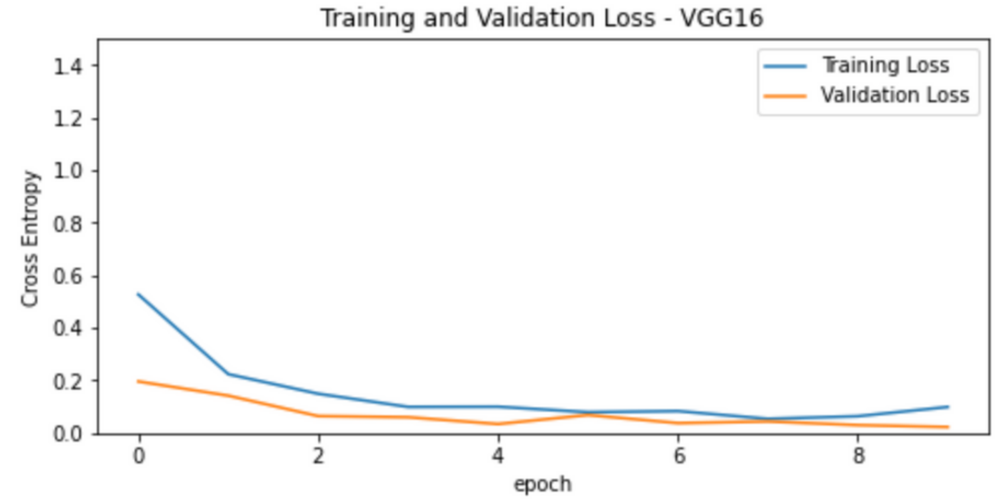
- Total parameters: 14,731,074
- Trainable parameters: 16,386
- Non-trainable parameters: 14,714,688

Performance

VGG16 Base Model

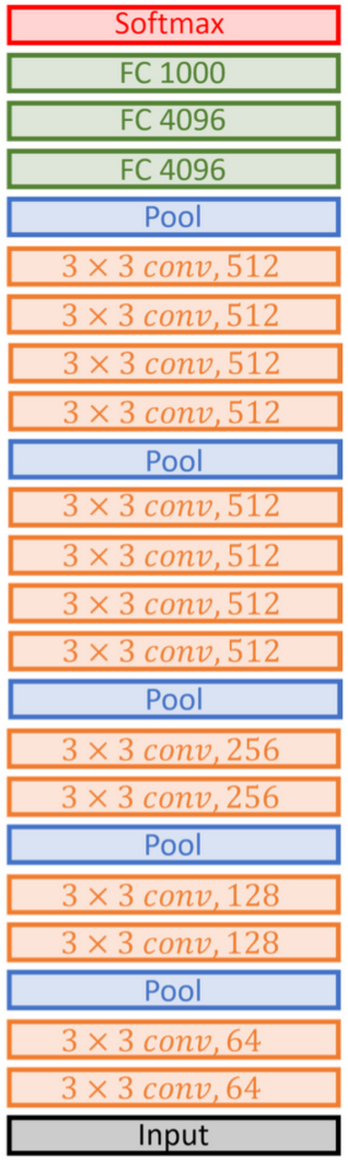
Test Accuracy: 99.2%





VGG 19

1. 19 Weighted Layers
2. +2 Convolutional Layers
3. Compare with VGG 16



CONVOLUTIONAL LAYERS X14

- 3X3 Filter Size
- Stride 1

MAXPOOL LAYERS X5

- 2X2 Filter Size
- Stride 2

FULLY CONNECTED LAYERS X3

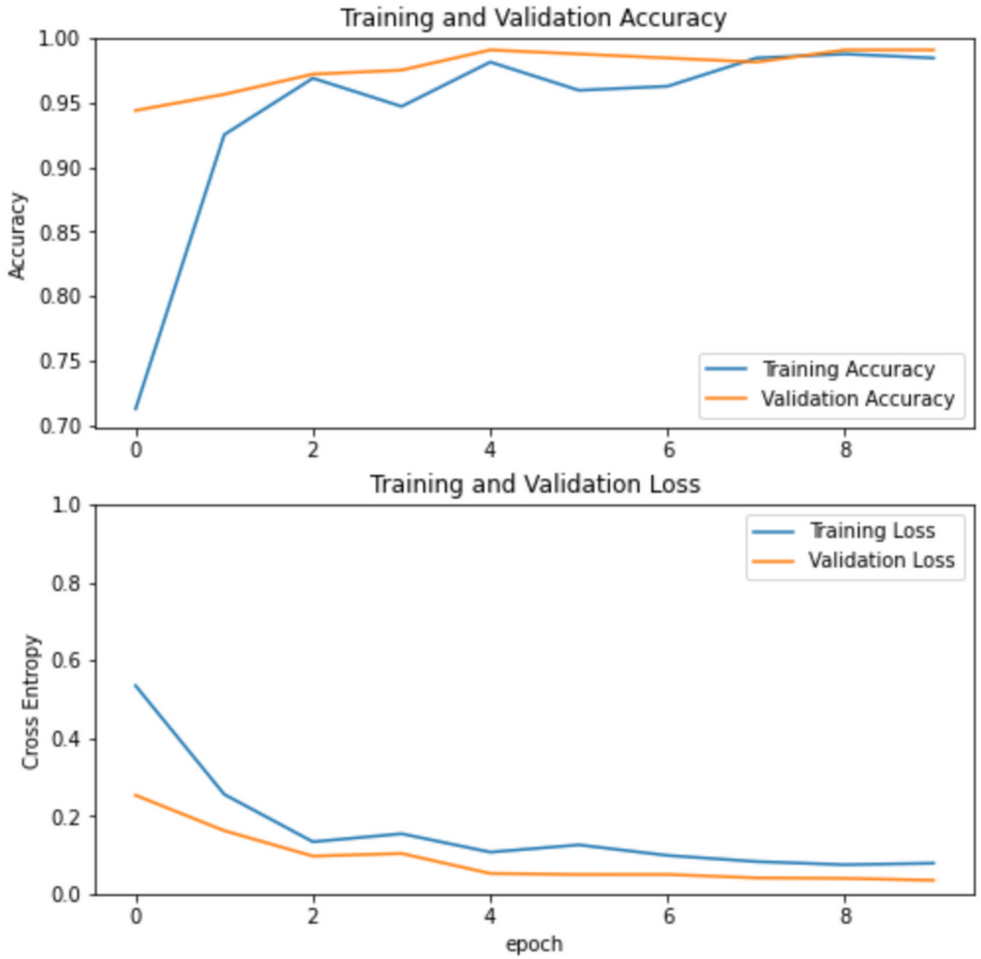
SOFTMAX LAYER X1

- Total parameters: 20,040,770
- Trainable parameters: 16,386
- Non-trainable parameters: 20,024,384

Performance

VGG19 Base Model

Test Accuracy: 98.7%



Fine-tuning

Action 1

Take layers from a previously trained model.

Action 2

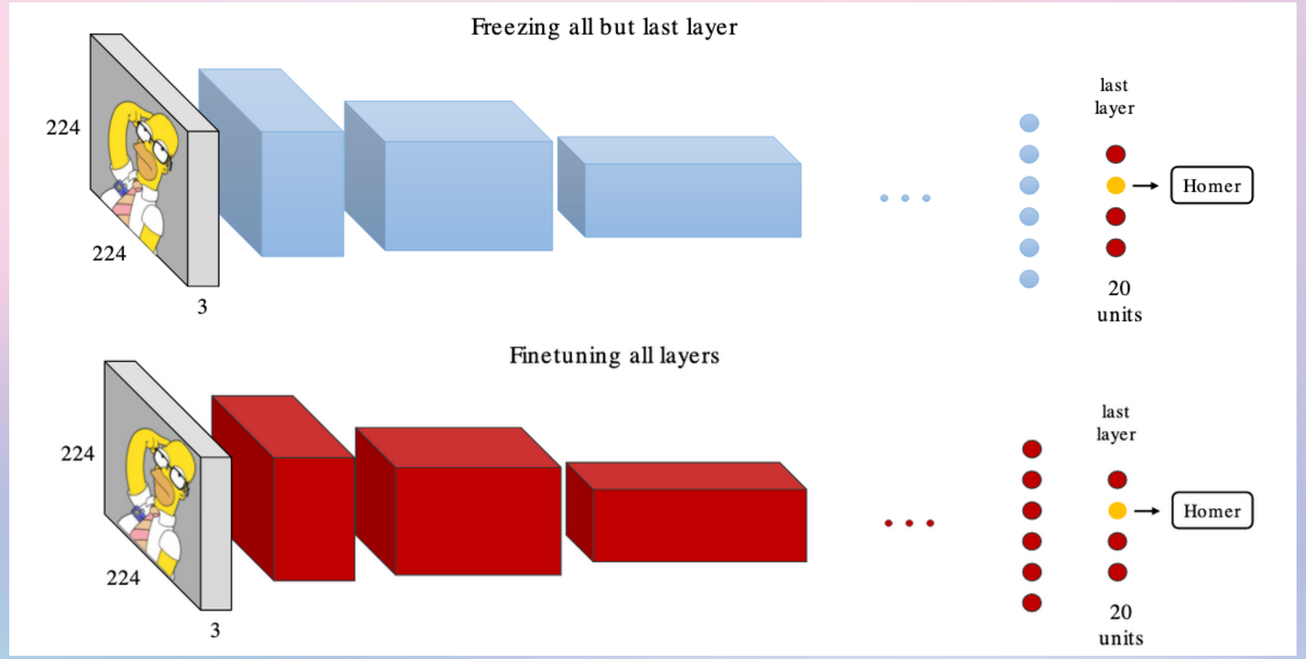
Freeze them, so as to avoid destroying any of the information they contain during future training rounds.

Action 3

Add some new, trainable layers on top of the frozen layers. Train the new layers on your dataset.

Action 4

Fine-tuning: consists of unfreezing the entire model you obtained above , and re-training it on the new data with a very low learning rate.



Fine-tuing & Callbcks

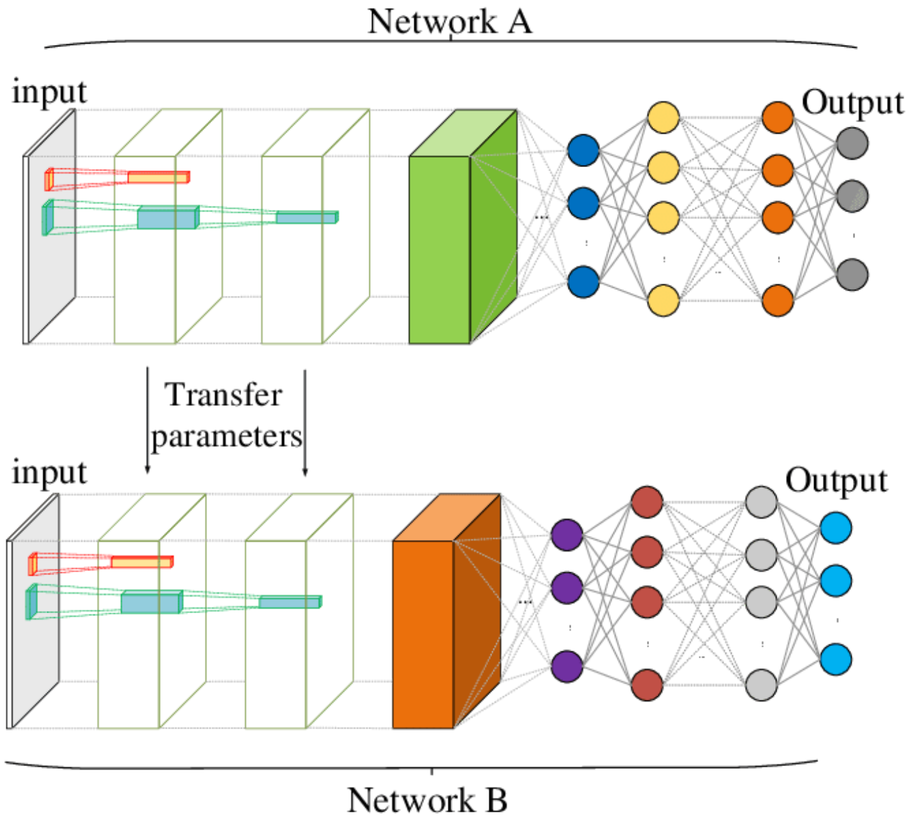
1. **Fine-Tuning**
   Unfreeze all the layer & Update weights.
2. **EarlyStopping**
   End training when val_accuracy is not improved for 4 consecutive times.
3. **ReduceLROnPlateau**
   The learning rate is reduced by half when val_accuracy is not improved for 2 consecutive times

Low Learning Rate

Adam
learning_rate=0.00001

Epoch 5
learning_rate=4.99 e-06

Epoch 7
learning_rate=2.49 e-06



Improvements

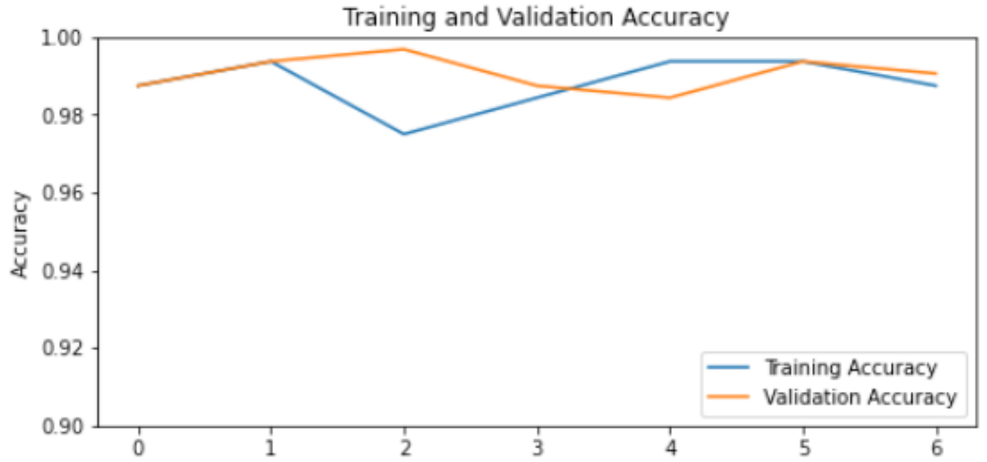
VGG19 Test Accuracy
98.73%

VGG19 Fine-tuning
Test Accuracy:
99.07%

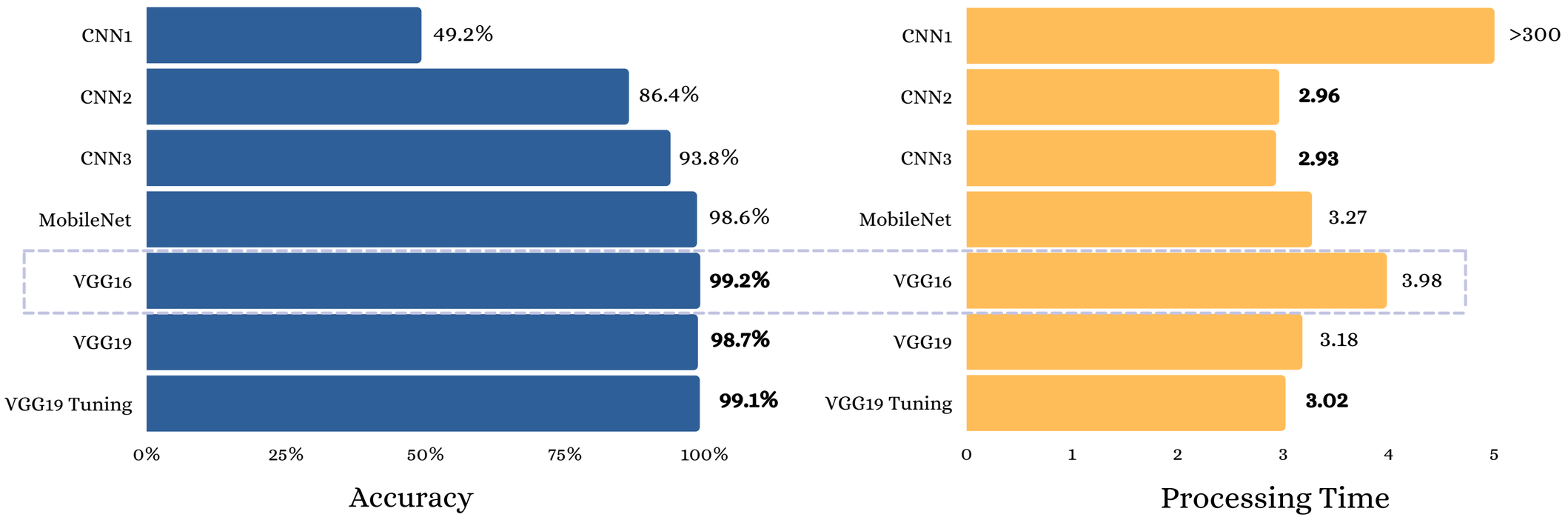
Performance

VGG19 Fine-Tuning

Test Accuracy: 99.1%



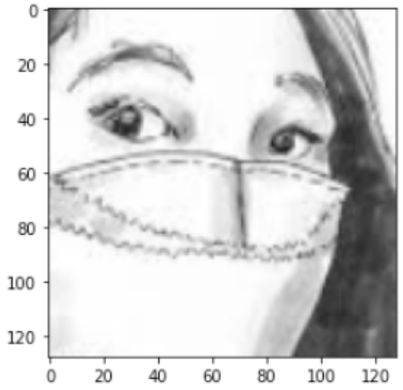
Sample Test & Mask Detection



Model Comparison Summary

Sample Test

WithMask Test



best_model.predict(sample_mask_img)

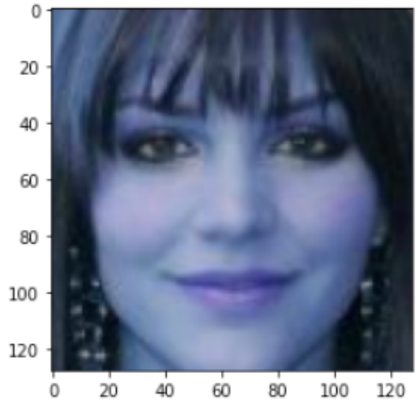
array([[9.9984312e-01, 2.4890372e-07]])

```
[ ]  pred = best_model.predict(sample_mask_img)
     predicted_class_indices = np.argmax(pred,axis=1)

     labels = (train_gen.class_indices)
     labels = dict((v,k) for k,v in labels.items())
     predictions = [labels[k] for k in predicted_class_indices]
     predictions
```

```
['mask']
```

WithoutMast Test



best_model.predict(sample_mask_img)

array([[1.2252026e-05, 9.9854195e-01]])

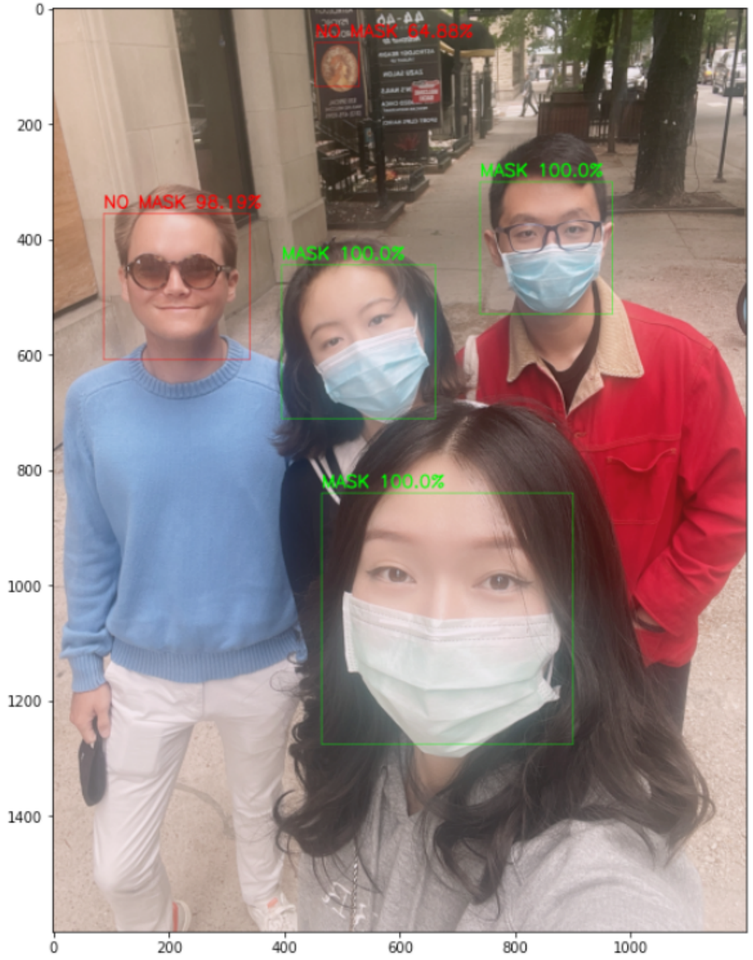
```
[ ]  pred = best_model.predict(sample_mask_img)
     predicted_class_indices=np.argmax(pred,axis=1)

     labels = (train_gen.class_indices)
     labels = dict((v,k) for k,v in labels.items())
     predictions = [labels[k] for k in predicted_class_indices]
     predictions
```

```
['no_mask']
```

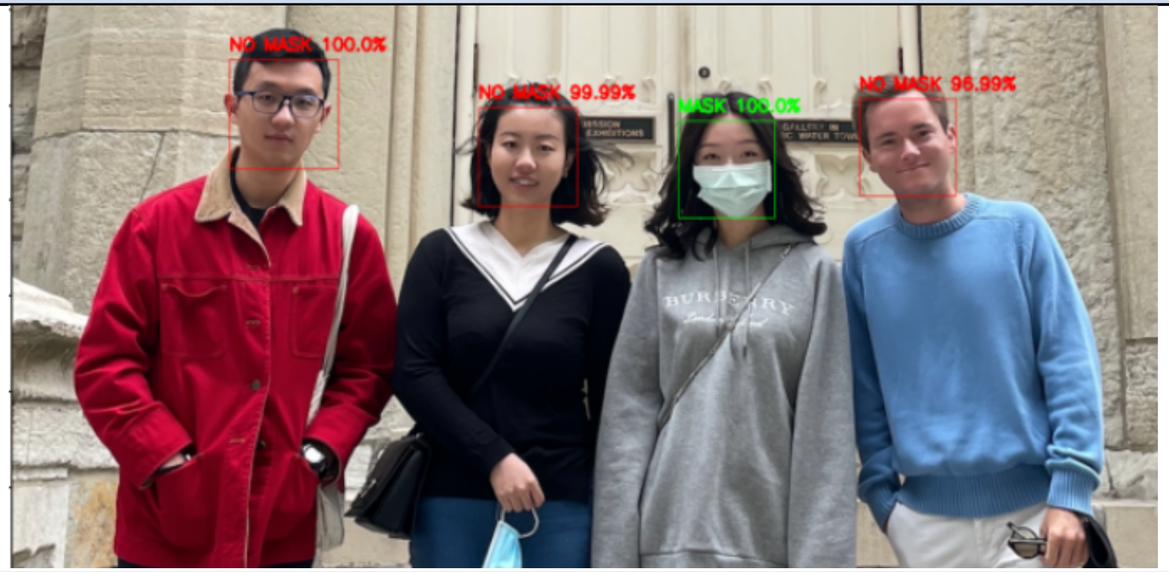
Mask Detection Test

Mask Detection 1



Mask Detection 2

Mask Detection 3



That's a wrap!

Thank you for listening.

References

Data

- https://www.kaggle.com/andrewmvd/face-mask-detection
- https://www.kaggle.com/lalitharajesh/haarcascades
- https://www.kaggle.com/nageshsingh/mask-and-social-distancing-detection-using-vgg19

Models

- https://www.robots.ox.ac.uk/~vgg/
- https://ai.googleblog.com/2017/06/mobilenets-open-source-models-for.html
- Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, & Hartwig Adam. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.