Report TP4

Transfer Learning with MobileNetV2

Ghandouz Amina G2

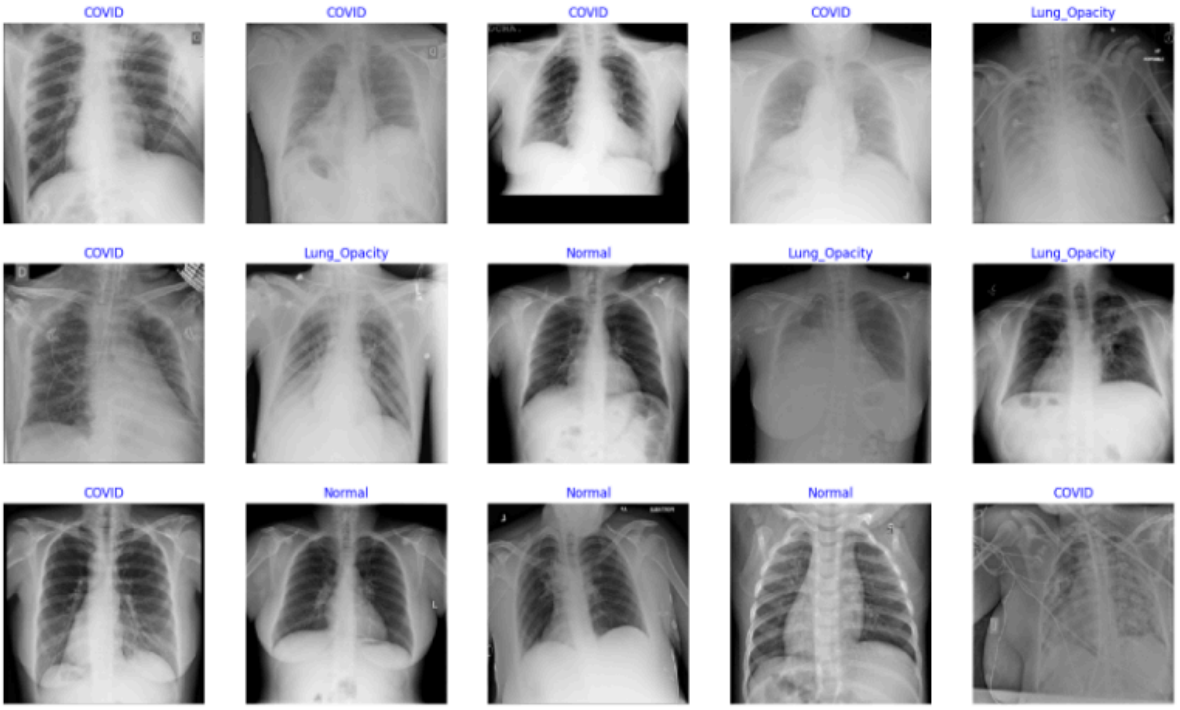
1. About the dataset: COVID-19 radiography dataset

Based on the url which we can find on *tp4 file*, COVID-19 Radiography dataset contains 33,920 chest X-ray images including:

- 11,956 covid-19
- 11,263 non-covid infections (viral or bacterial pneumonia)
- 10,701 normal

It contains also ground-truth lung segmentation masks for the entire dataset

The dataset we worked with has three main classes/categories: COVID-19, Normal, and non-covid infections (Viral Pneumonia and Lung_Opacity).In total we have ~21165 images for all the categories, we're using this dataset for multi-classification task.



2. Preprocessing

Before starting the training process, we preprocessed the dataset where we did:

- normalization (by dividing the pixel values by 255.0)
- resizing (resize all images to 244*244 pixels to match the input size expected by the MobileNet architecture)
- data splitting (we splitted the dataset into three parts: 70% for training, 15% for testing to evaluate the model during development and 15% for validation)

3. Loading and pre-trained architecture:

We imported the MobileNetV2 architecture and initialized it with pretrained weights from ImageNet and removed the original classification head to add our own classifier later.

Using the pretrained MobileNetV2 model, we're leveraging transfer learning and this allows us to:

- avoid the training process from scratch.
- use a model that already learned useful visual features.
- speed up convergence and often achieve higher accuracy with less data.

4. Loading and pre-trained the mobileNetV2 model:

The original top layers used for ImageNet classification have 1000 classes sp we excluded them to add a new sequence of fully connected layers:

```
model = keras.Sequential([
    base_model,
    layers.Flatten(),
    layers.Dense(1024, activation="relu"),
    layers.Dense(512, activation="relu"),
    layers.Dense(3, activation="softmax")
])
```

- *Flatten*(): Converts the 4D output into a 1D vector.
- *Dense*(1024, activation='relu')
- *Dense*(512, activation='relu')

- *Dense*(3, activation='softmax').

For the total parameters we got 67,011,140 parameters where 66,977,028 of them are trainable and the rest (34,112) are non-trainable parameters

Here's a summary of the model layers with their parameter counts:

Layer	output shape	params
MobileNetV2	(None, 7, 7, 1280)	2,257,984
Flatten	(None, 62720)	0
Dense (1024)	(None, 1024)	64,226,304
Dense (512)	(None, 512)	524,800
Dense (number of classes)	(None, 4)	2,052
total params		67,011,140

5. Transfer learning strategies:

To train the mobileNetV2 model, we used transfer learning with three different strategies and fixed hyper-parameters:

- number of epochs = 20
- batch size = 64
- optimizer = adam
- learning rate = 0.001

The strategies we have are:

a- Training only the last 3 fully connected layers:

Using this strategy, all the layers in mobileNetV2 are frozen except the last 3 fully connected layers (the ones we added before on step 5).

b- Training the last convolution layer and 3 fully connected layers:

The trainable layers are the last convolution layer and 3 fully connected layers we used on the previous strategy.

c- Training the last 2 convolution and the last 3 fully connected layers:

In this strategy we added an additional layer so we have trained the last 2 convolution layers and 3 fully connected ones.

6. The obtained results:

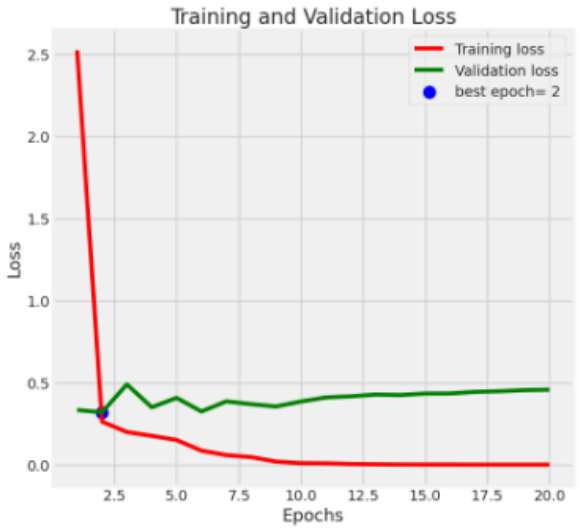
Training time:

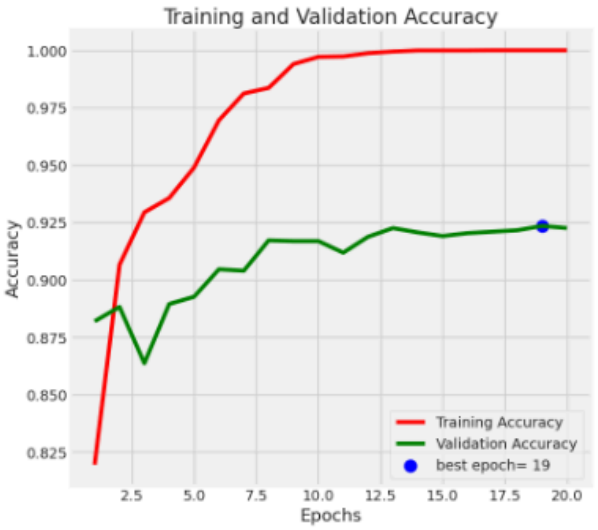
For all the three model we have **total params = 67,011,140**

model	layers	trainable params	non-trainable params	training time
model1	3 FC	64,753,156	2,257,984	25 min 2 s
model2	1 Conv + 3 FC	65,165,316	1,845,824	**24 min 18 s**
model3	2 Conv + 3 FC	65,473,156	1,537,984	26 min 59 s

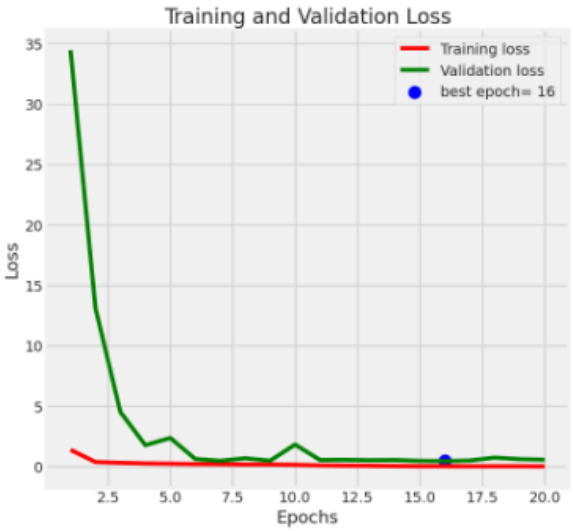
The convergence curve:

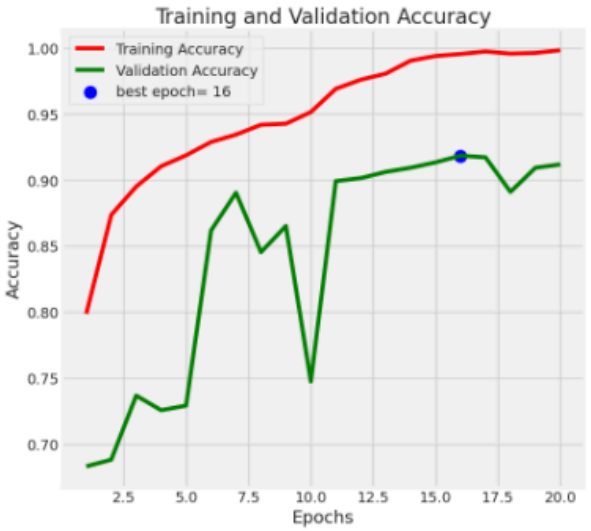
Model2: 1 Conv + 3 FC





Model3: 2 Conv + 3 FC





model	loss best epoch	accuracy best epoch	best training accuracy
model1	5	19	0.92
model2	**2**	19	0.925
model3	16	**16**	**0.93**

Accuracy and loss for the three sets:

model	loss/accuracy	training set	test set	validation set
model1	accuracy	0.96	0.908	0.909
	loss	0.101	**0.27**	**0.29**
model2	accuracy	0.91	0.89	0.88
	loss	0.22	0.29	0.31
model3	accuracy	**0.99**	**0.91**	**0.91**
	loss	**0.02**	0.46	0.44

	precision	**recall**	**f1-score**	**support**
COVID	**0.95**	**0.92**	**0.94**	**542**
Lung_Opacity	**0.91**	**0.81**	**0.86**	**902**

Normal	**0.89**	**0.95**	**0.92**	**1529**
Viral Pneumonia	**0.98**	**0.96**	**0.97**	**202**
accuracy			**0.91**	**3175**
macro avg	**0.93**	**0.91**	**0.92**	**3175**
weighted avg	**0.91**	**0.91**	**0.91**	**3175**

Accuracy, recall, precision, F1-score, and sensitivity on the validation set,

model 2 1 con + 3 fc

	precision	recall	f1-score	support
COVID	0.89	0.91	0.90	542
Lung_Opacity	0.91	0.79	0.85	902
Normal	0.88	0.95	0.91	1529
Viral Pneumonia	0.98	0.92	0.95	202

accuracy			0.89	3175
macro avg	0.92	0.89	0.90	3175
weighted avg	0.90	0.89	0.89	3175

model 3 2 conv + 3 fc

==========model_1==========

Train Loss: 0.020660514011979103

Train Accuracy: 0.9932332634925842

Validation Loss: 0.44501614570617676

Validation Accuracy: 0.9184252023696899

Test Loss: 0.4672064781188965

Test Accuracy: 0.9118109941482544

==========

	precision	recall	f1-score	support
COVID	0.94	0.92	0.93	542
Lung_Opacity	0.90	0.84	0.87	902
Normal	0.90	0.95	0.92	1529
Viral Pneumonia	0.99	0.92	0.96	202
accuracy			0.91	3175

macro avg	0.93	0.91	0.92	3175
weighted avg	0.91	0.91	0.91	3175