Covid-19 Detection using X-ray images of chest

First we will import all the necessary python libraries, tensorflow functions and **Pre-trained VGG16** m other pre-trained models are available on `tensorflow.keras.applications`.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import numpy as np
import math
```

Setting the Hyperparameters

The initial learning rate `INIT_LR` is set to 0.001 and total number of epochs `EPOCHS` to 25.

The `dir2dataset` is the variable holding the path to the dataset folder.

The dataset folder contains 876 covid-19 positive xray images and 1341 covid-19 negative images

```
INIT_LR = 1e-3
EPOCHS = 25
dir2dataset = "dataset"
!tree --filelimit 10
```

```
.
├── covid [876 entries exceeds filelimit, not opening dir]
└── normal [1341 entries exceeds filelimit, not opening dir]

2 directories, 0 files
```

Train and validation split

The images are now divided into train and validation set with 20% of total images in validation set `va` images are rescaled and resized to 224 x 224 pixels.

Covid-19 positive images have been labeled as 0 and negatives as 1. There are 1774 images in traini validation set.

```
trainAug = ImageDataGenerator(
  rotation_range=15,
  fill_mode="nearest",
  validation_split=0.2,
  rescale=1. /255
  )

traindata = trainAug.flow_from_directory(
    dir2dataset,
    target_size=(224, 224),
    color_mode="rgb",
    classes=None,
    class_mode="categorical",
    batch_size=128,
    shuffle=True,
    seed=1337,
    save_to_dir=None,
    save_prefix="",
    save_format="png",
    follow_links=False,
    subset="training",
    interpolation="nearest",
)
print(traindata.class_indices)
print(traindata.n)
valdata = trainAug.flow_from_directory(
    dir2dataset,
    target_size=(224, 224),
    color_mode="rgb",
    classes=None,
    class_mode="categorical",
    batch_size=128,
    shuffle=False,
    seed=1337,
    save_to_dir=None,
    save_prefix="",
    save_format="png",
    follow_links=False,
    subset="validation",
    interpolation="nearest",
)
```

```
Found 1774 images belonging to 2 classes.
{'covid': 0, 'normal': 1}
1774
Found 443 images belonging to 2 classes.
```

The Model

base model is same as pretrained VGG16 but the head part is replaced with custom layers that outpu probability of image being covid-19 positive and other is probability of image being covid-19 negative already pretrained the updation of weights during training for these layers is turned off.

```
 for layer in baseModel.layers:
     layer.trainable = False
```

```
baseModel = VGG16(weights="imagenet", include_top=False,
  input_tensor=Input(shape=(224, 224, 3)))
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(4, 4))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(64, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)
model = Model(inputs=baseModel.input, outputs=headModel)
```

```
for layer in baseModel.layers:
  layer.trainable = False
```

Optimizer and model compilation

Adam optimizer is used with learning rate = `INIT_LR` and learning rate decay after each epoch given

```
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
  metrics=["accuracy"])
```

Training the model

The head part of the model which was added is now trained over the dataset for `EPOCHS` number of t

```
H = model.fit(
  traindata,
  steps_per_epoch=math.ceil(traindata.n / traindata.batch_size),
  validation_data=valdata,
  validation_steps=math.ceil(valdata.n / valdata.batch_size),
  epochs=EPOCHS)
```

```
[INFO] training head...
Epoch 1/25
14/14 [==============================] - 60s 4s/step - loss: 0.6257 - accuracy: 0.6691 -
Epoch 2/25
14/14 [==============================] - 61s 4s/step - loss: 0.3968 - accuracy: 0.8675 -
Epoch 3/25
14/14 [==============================] - 61s 4s/step - loss: 0.2844 - accuracy: 0.9222 -
Epoch 4/25
14/14 [==============================] - 61s 4s/step - loss: 0.2264 - accuracy: 0.9363 -
Epoch 5/25
14/14 [==============================] - 61s 4s/step - loss: 0.1910 - accuracy: 0.9476 -
Epoch 6/25
14/14 [==============================] - 61s 4s/step - loss: 0.1558 - accuracy: 0.9521 -
Epoch 7/25
14/14 [==============================] - 61s 4s/step - loss: 0.1421 - accuracy: 0.9566 -
Epoch 8/25
14/14 [==============================] - 61s 4s/step - loss: 0.1290 - accuracy: 0.9605 -
Epoch 9/25
14/14 [==============================] - 61s 4s/step - loss: 0.1206 - accuracy: 0.9611 -
Epoch 10/25
14/14 [==============================] - 61s 4s/step - loss: 0.1262 - accuracy: 0.9577 -
Epoch 11/25
14/14 [==============================] - 61s 4s/step - loss: 0.1123 - accuracy: 0.9645 -
Epoch 12/25
14/14 [==============================] - 61s 4s/step - loss: 0.0989 - accuracy: 0.9735 -
Epoch 13/25
14/14 [==============================] - 62s 4s/step - loss: 0.0950 - accuracy: 0.9696 -
Epoch 14/25
14/14 [==============================] - 61s 4s/step - loss: 0.0919 - accuracy: 0.9707 -
Epoch 15/25
14/14 [==============================] - 61s 4s/step - loss: 0.0905 - accuracy: 0.9696 -
Epoch 16/25
14/14 [==============================] - 61s 4s/step - loss: 0.0914 - accuracy: 0.9701 -
Epoch 17/25
14/14 [==============================] - 61s 4s/step - loss: 0.0808 - accuracy: 0.9769 -
Epoch 18/25
14/14 [==============================] - 61s 4s/step - loss: 0.0793 - accuracy: 0.9758 -
Epoch 19/25
14/14 [==============================] - 61s 4s/step - loss: 0.0729 - accuracy: 0.9752 -
Epoch 20/25
14/14 [==============================] - 61s 4s/step - loss: 0.0700 - accuracy: 0.9775 -
Epoch 21/25
14/14 [==============================] - 61s 4s/step - loss: 0.0785 - accuracy: 0.9729 -
Epoch 22/25
14/14 [==============================] - 61s 4s/step - loss: 0.0726 - accuracy: 0.9741 -
Epoch 23/25
14/14 [==============================] - 61s 4s/step - loss: 0.0667 - accuracy: 0.9797 -
Epoch 24/25
14/14 [==============================] - 61s 4s/step - loss: 0.0645 - accuracy: 0.9791 -
Epoch 25/25
14/14 [==============================] - 61s 4s/step - loss: 0.0646 - accuracy: 0.9763 -
```

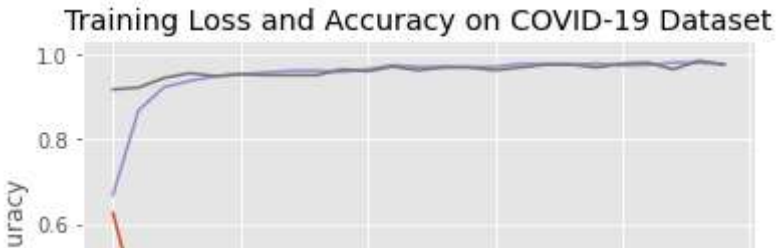
Accuracy and Plots

Accuracy, precision, recall, sensitivity, specificity of the model is shown and the losses as well as acc plotted

```
print("[INFO] evaluating network...")
predIdxs = model.predict(valdata,
steps= math.ceil(valdata.n / valdata.batch_size),
verbose=1)
predIdxs = np.argmax(predIdxs, axis=1)
labels = (valdata.class_indices)
labels = dict((v,k) for k,v in labels.items())

print(classification_report(valdata.labels, predIdxs,
  target_names=['covid','normal']))
```

```
[INFO] evaluating network...
4/4 [==============================] - 7s 2s/step
              precision    recall  f1-score   support

       covid       0.99      0.97      0.98       175
      normal       0.98      0.99      0.99       268

    accuracy                           0.98       443
   macro avg       0.98      0.98      0.98       443
weighted avg       0.98      0.98      0.98       443
```

```
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy on COVID-19 Dataset")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
now = datetime.now()
dt_string = now.strftime("%d%m%Y%H:%M:%S")
plt.savefig("outputs/"+"plot.png")
```




```
valdata.labels
cm = confusion_matrix(valdata.labels, predIdxs)
total = sum(sum(cm))
acc = (cm[0, 0] + cm[1, 1]) / total
sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])

# show the confusion matrix, accuracy, sensitivity, and specificity
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))
```

```
[[170   5]
 [  2 266]]
acc: 0.9842
sensitivity: 0.9714
specificity: 0.9925
```

Saving the model

```
model.save("outputs/"+"covid19detection.model", save_format="h5")
```