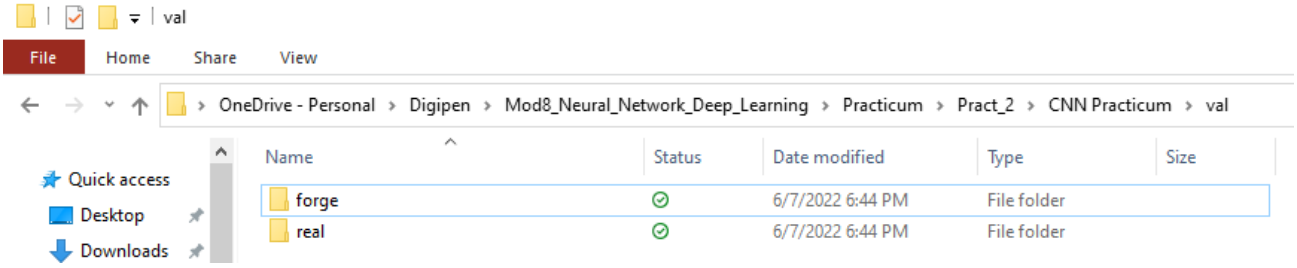
Artificial Neural Network & Deep Learning– Practicum 2- Convolution Neural Network

[Tee Li Lin]

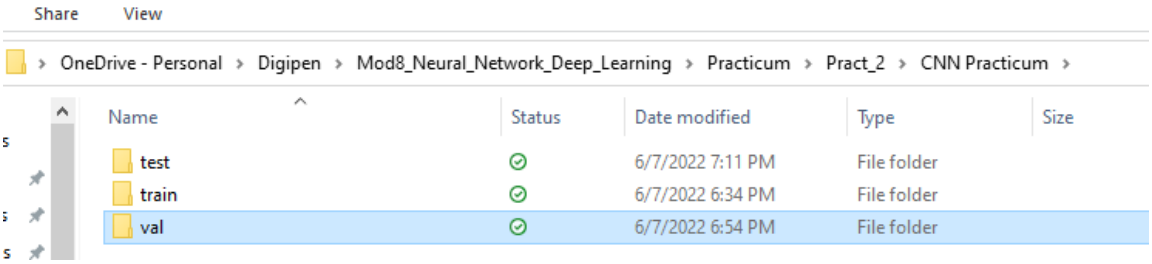
(Qns 1)

Arrange the all the images into 3 folders, namely “train”, “val” and test folders with 2 subfolders namely “forge” and “real”.

Subfolders:



Main folders:



Save the files’ directories into the variables as shown below:

In [3]:
```
train_folder= './CNN Practicum/train/'
val_folder = './CNN Practicum/val/'
test_folder = './CNN Practicum/test/'
```

Extract the images using keras “flow_from_directory” and convert all the images into greyscale format so as to reduce the “noise”. Standardise all images as per screenshot shown below before fit into the CNN model:

In [5]:
```
# Fitting the CNN to the images
# The function ImageDataGenerator augments your image by iterating through image as your CNN is getting ready to process tha

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)  #Image normalization.

training_set = train_datagen.flow_from_directory('../Pract_2/CNN Practicum/train/',
                                                 target_size = (64, 64),
                                                 color_mode = 'grayscale',
                                                 batch_size = 32,
                                                 class_mode = 'categorical')

validation_generator = test_datagen.flow_from_directory('../Pract_2/CNN Practicum/val/',
    target_size=(64, 64),
    color_mode = 'grayscale',
    batch_size=32,
    class_mode='categorical')

test_set = test_datagen.flow_from_directory('../Pract_2/CNN Practicum/test/',
                                            target_size = (64, 64),
                                            shuffle=False,
                                            color_mode = 'grayscale',
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

(Qns 2)

Below is the structure of CNN model using sequential with 2 convolution layers and 2 max pooling layers and run 30 epochs:

In [4]:
```
# let's build the CNN model

cnn = Sequential()

#Convolution
cnn.add(Conv2D(32, (3, 3), activation="relu", input_shape=(64, 64, 1)))

#Pooling
cnn.add(MaxPooling2D(pool_size = (2, 2)))

# 2nd Convolution
cnn.add(Conv2D(32, (3, 3), activation="relu"))

# 2nd Pooling layer
cnn.add(MaxPooling2D(pool_size = (2, 2)))

# Flatten the layer
cnn.add(Flatten())

# Fully Connected Layers
cnn.add(Dense(activation = 'relu', units = 128))
cnn.add(Dense(activation = 'softmax', units = 2))


# Compile the Neural network
cnn.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

In [6]:
```
cnn.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        320

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 128)               802944

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 812,770
Trainable params: 812,770
Non-trainable params: 0
_________________________________________________________________
```

In [7]:
```
cnn_model = cnn.fit(training_set,
        steps_per_epoch = len(training_set),
        epochs = 30,
        validation_data = validation_generator)
```

(Qns 3)

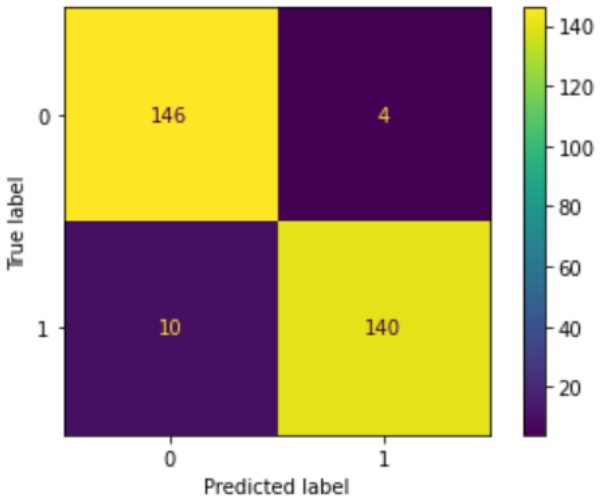
In [12]:
```
print(classification_report(test_set.classes, y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.94      0.97      0.95       150
           1       0.97      0.93      0.95       150

    accuracy                           0.95       300
   macro avg       0.95      0.95      0.95       300
weighted avg       0.95      0.95      0.95       300
```

In [13]:
```
cm=confusion_matrix(test_set.classes, y_pred_test)
cm
```

Out[13]:
```
array([[146,   4],
       [ 10, 140]], dtype=int64)
```

In [14]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```



Based on the classification report and confusion matrix shown above, the accuracy on the test dataset using this CNN model is around 95%. This seems that this model can detect unseen signatures at high accuracy rate. However, the structure of model is not optimal, and the training accuracy is not very high even though it can learn the signature using CNN. Thus, there are rooms for improvement in this model.