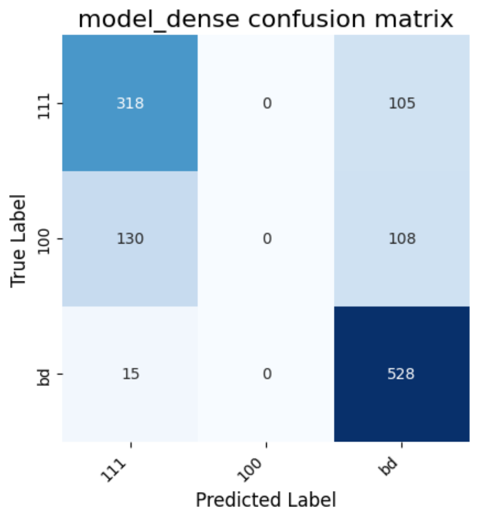
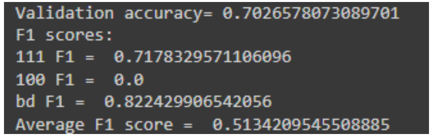
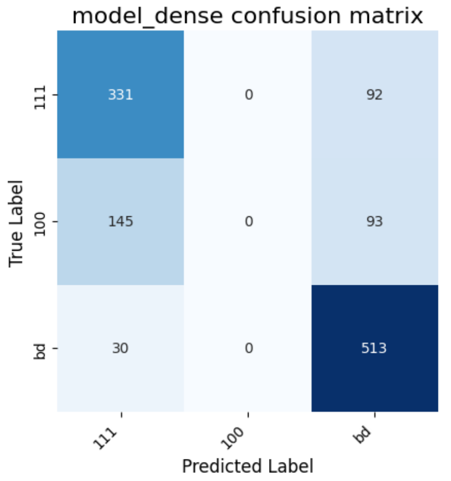
Assessment Figures


model_dense confusion matrix

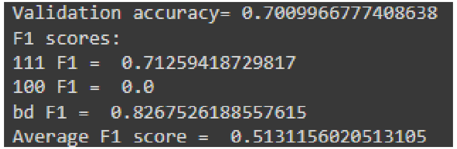
True Label \ Predicted Label	111	100	bd
111	318	0	105
100	130	0	108
bd	15	0	528



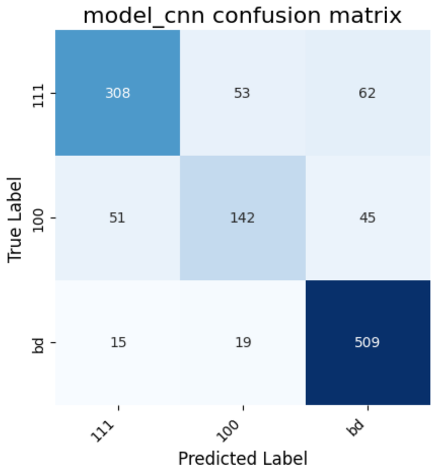
```
Validation accuracy= 0.7026578073089701
F1 scores:
111 F1 =  0.7178329571106096
100 F1 =  0.0
bd F1 =  0.822429906542056
Average F1 score =  0.5134209545508885
```


model_dense confusion matrix

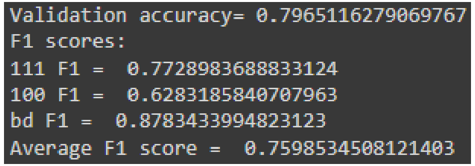
True Label \ Predicted Label	111	100	bd
111	331	0	92
100	145	0	93
bd	30	0	513



```
Validation accuracy= 0.7009966777408638
F1 scores:
111 F1 =  0.71259418729817
100 F1 =  0.0
bd F1 =  0.8267526188557615
Average F1 score =  0.5131156020513105
```


model_cnn confusion matrix

True Label \ Predicted Label	111	100	bd
111	308	53	62
100	51	142	45
bd	15	19	509



```
Validation accuracy= 0.7965116279069767
F1 scores:
111 F1 =  0.7728983688833124
100 F1 =  0.6283185840707963
bd F1 =  0.8783433994823123
Average F1 score =  0.7598534508121403
```

ML4ER - Assignment 7 Activities

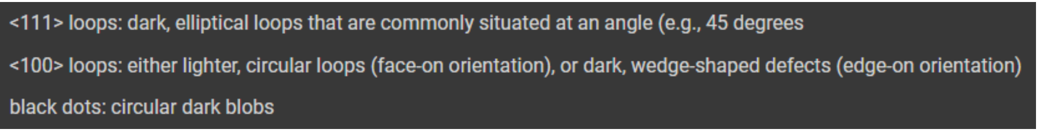
Muhammad Zain Azeem,

Informatics Skunkworks (**non-credits)**, Week 4

10/08/2024

Progress

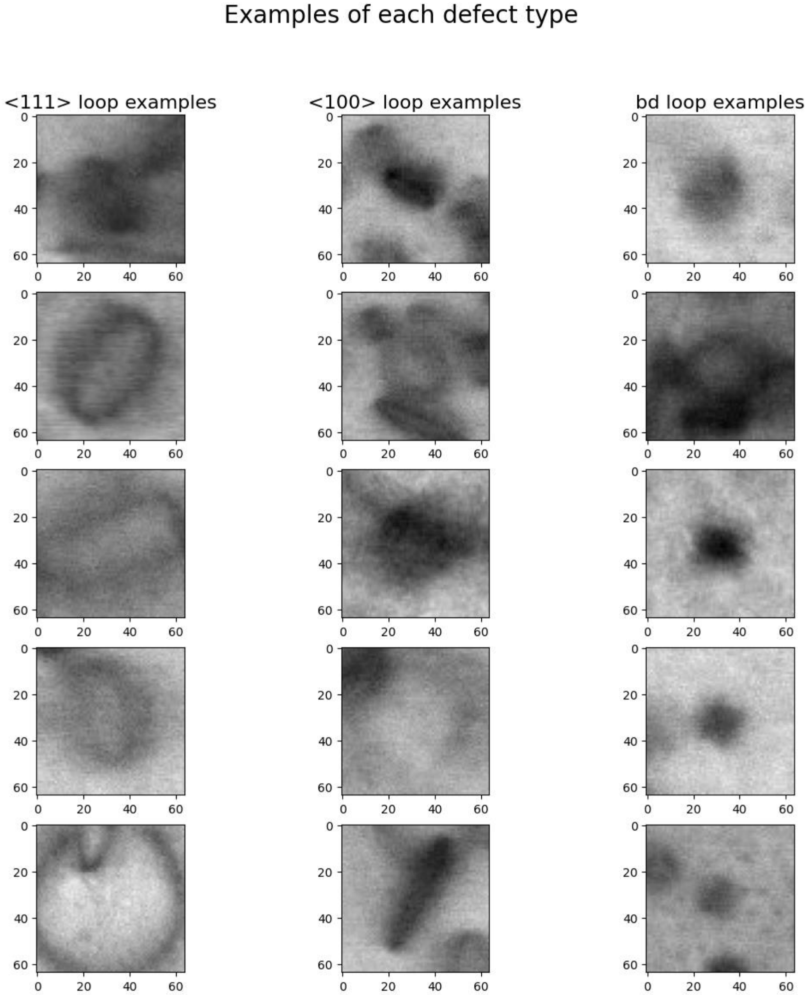
- Examining images of defects

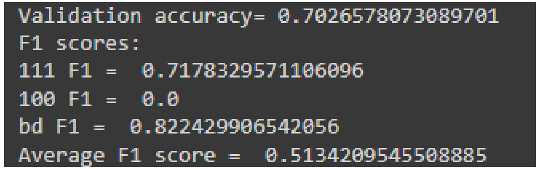


<111> loops: dark, elliptical loops that are commonly situated at an angle (e.g., 45 degrees

<100> loops: either lighter, circular loops (face-on orientation), or dark, wedge-shaped defects (edge-on orientation)

black dots: circular dark blobs

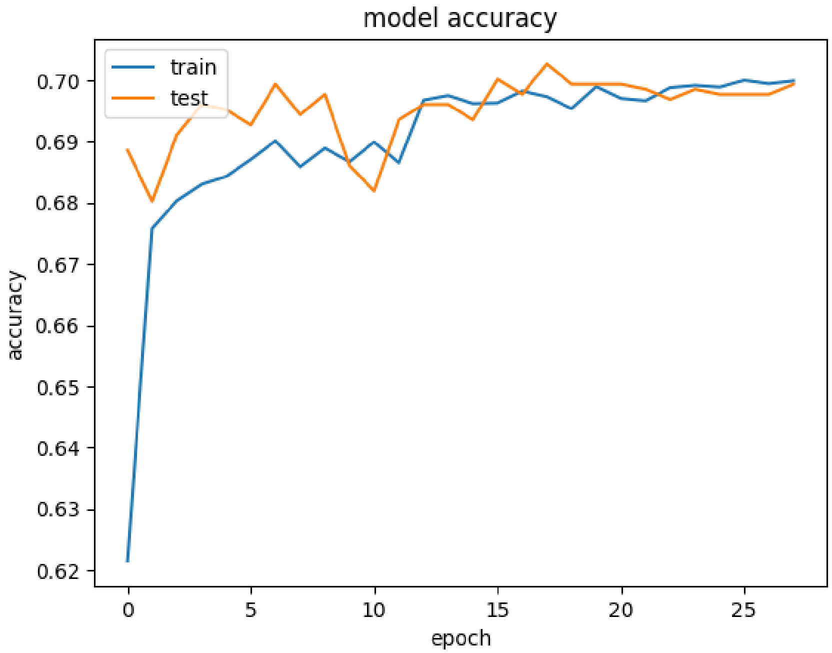


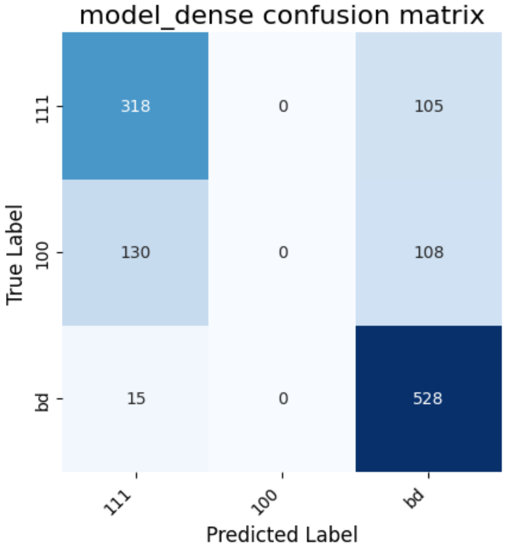


```
Validation accuracy= 0.7026578073089701
F1 scores:
111 F1 =  0.7178329571106096
100 F1 =  0.0
bd F1 =  0.822429906542056
Average F1 score =  0.5134209545508885
```

Progress

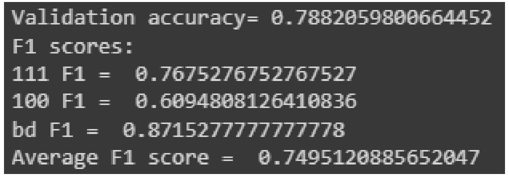
- Defect classification with FCN





	111	100	bd
111	318	0	105
100	130	0	108
bd	15	0	528

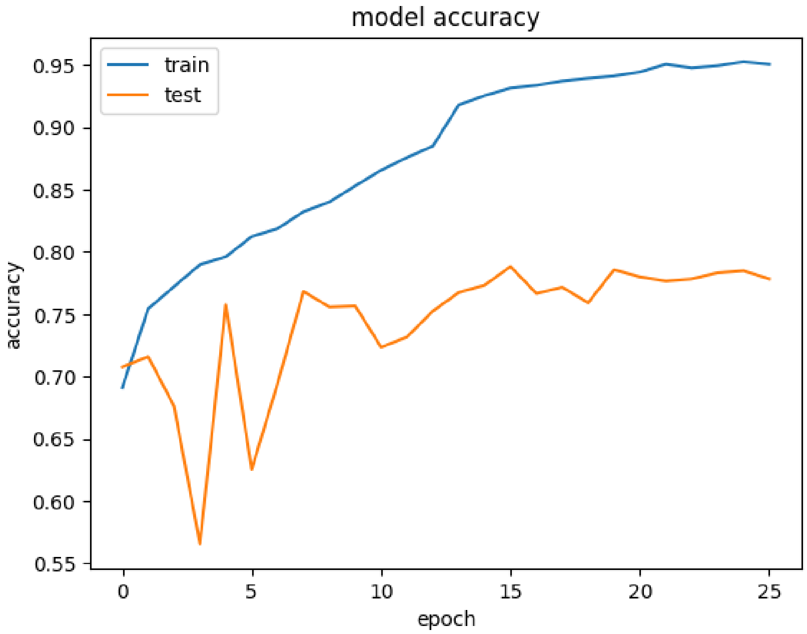
- The model fails to categorize <100> loops, with an average F1 score of 0.51

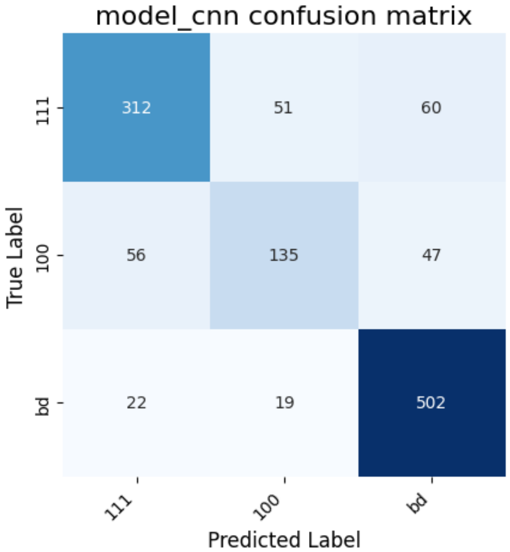


```
Validation accuracy= 0.7882059800664452
F1 scores:
111 F1 =  0.7675276752767527
100 F1 =  0.6094808126410836
bd F1 =  0.8715277777777778
Average F1 score =  0.7495120885652047
```

Progress

- CNN layer on FCN





	111	100	bd
111	312	51	60
100	56	135	47
bd	22	19	502

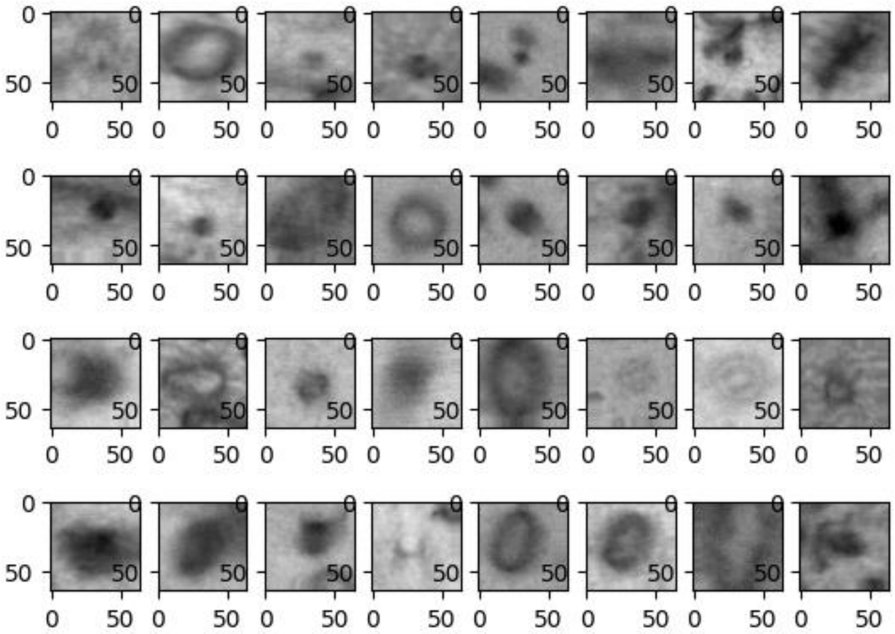
Original shape: 64x64 pixels

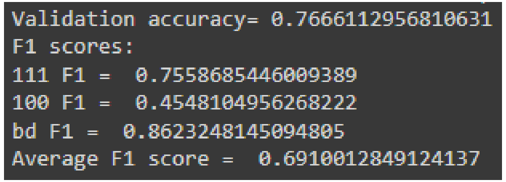
- Adding CNN layers enhanced the model's accuracy and average F1 score (0.74)

Progress

Solving the overfitting problem in CNN:

- Dropout
- Data augmentation (ImageDataGenerator class)

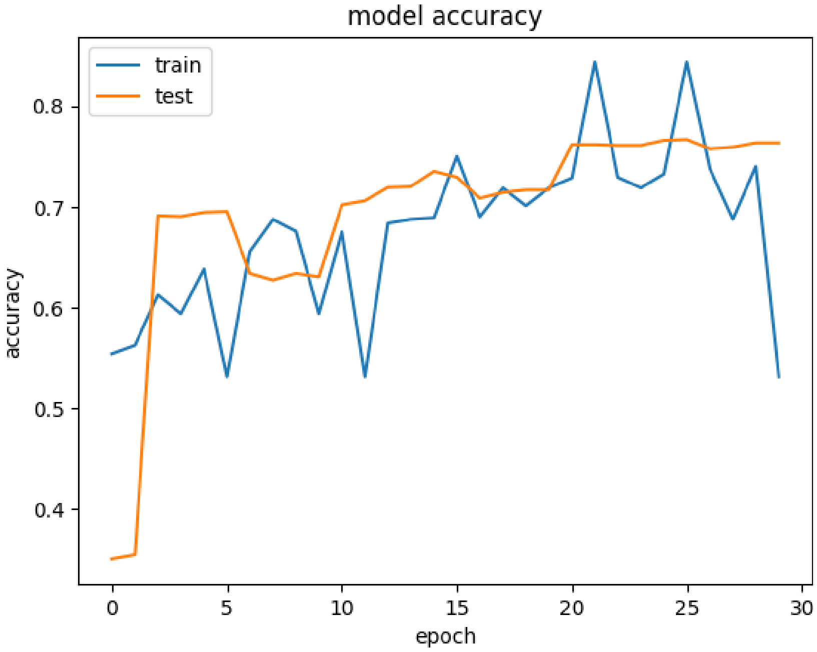


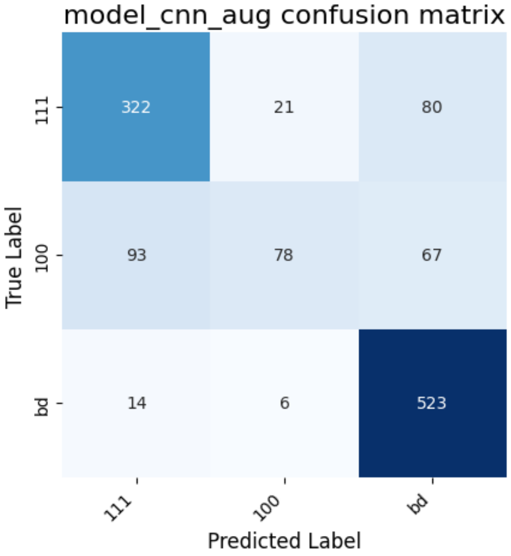


```
Validation accuracy= 0.7666112956810631
F1 scores:
111 F1 =  0.7558685446009389
100 F1 =  0.4548104956268222
bd F1 =  0.8623248145094805
Average F1 score =  0.6910012849124137
```

Progress

Solving the overfitting problem in CNN:



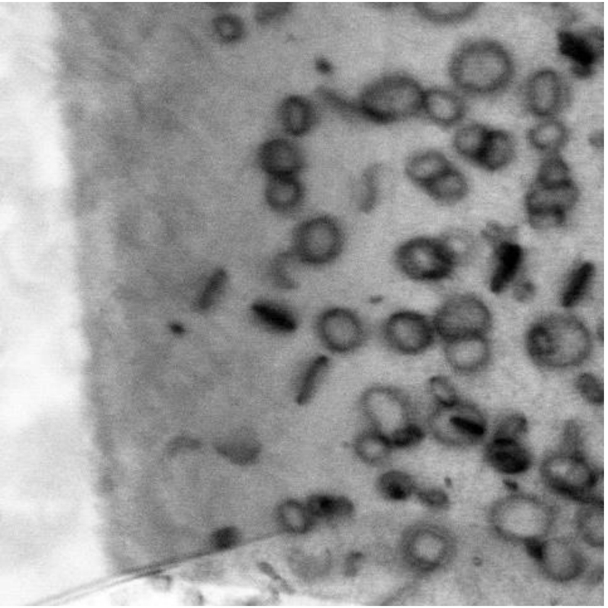


	111	100	bd
111	322	21	80
100	93	78	67
bd	14	6	523

- The model's F1 score and accuracy improved slightly to 0.735 and 79%, respectively.

Progress

Object detection using the You Only Look Once (YOLO) model



STEM micrograph (random image)

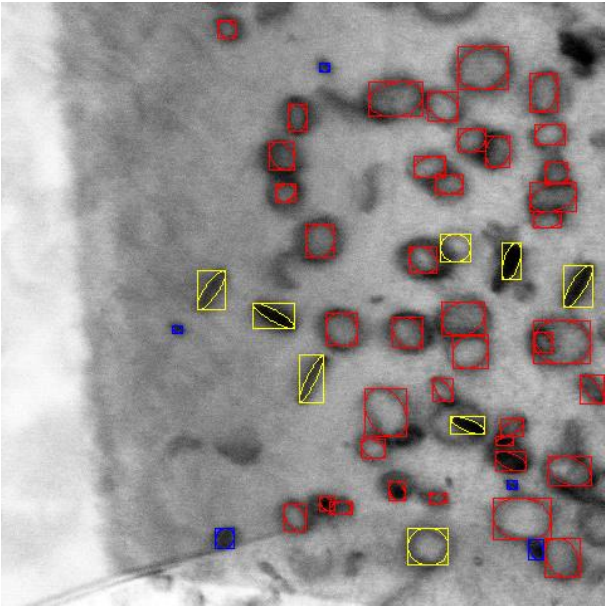
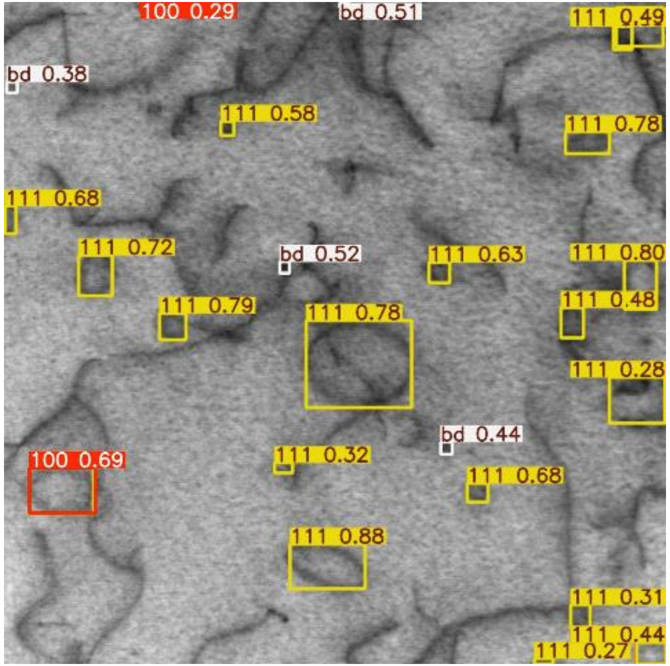


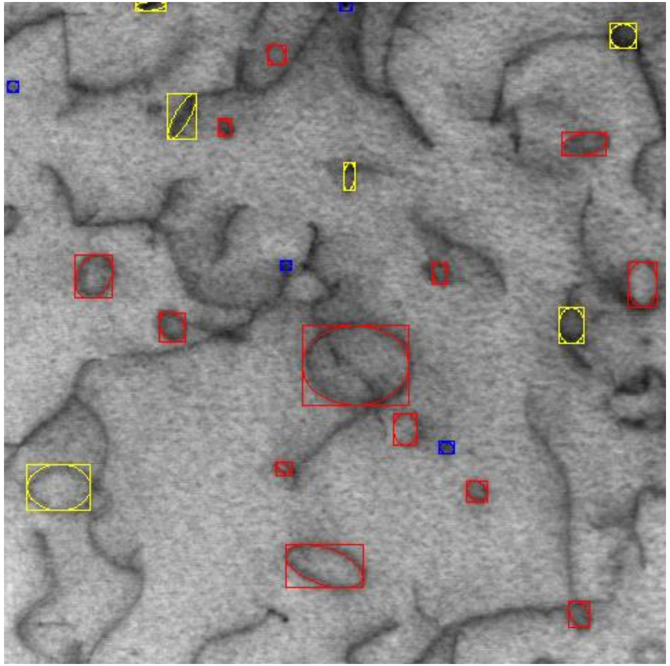
Image labelling

- Blue: black dots
- Red: <111> loops
- Yellow: <100> loops

Progress



PREDICTED



GROUND TRUTH

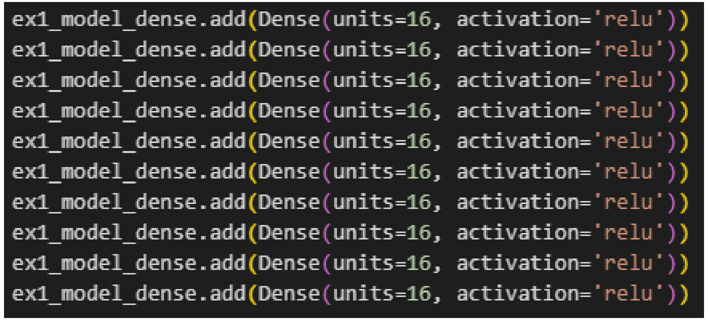
Progress

Exercise 1: Revisit adding layers to our FCN

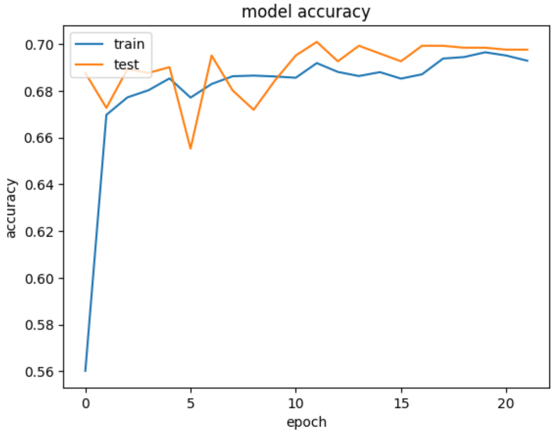
Task:

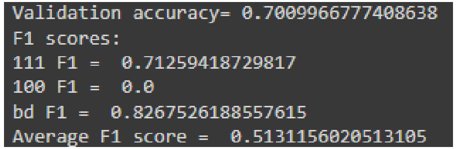
Examine the claim from section 1 that adding more layers to our Fully Connected Network from Section 1 will not significantly improve performance.

- Added 8 more layers!

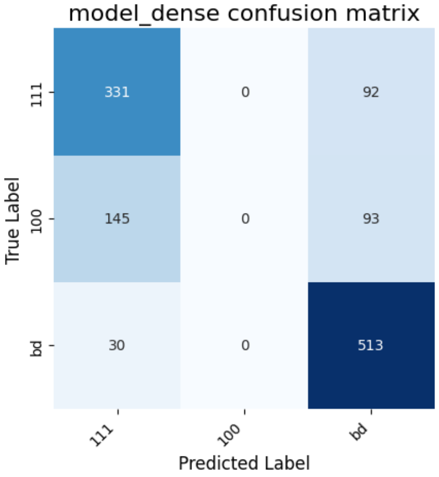


```
ex1_model_dense.add(Dense(units=16, activation='relu'))
ex1_model_dense.add(Dense(units=16, activation='relu'))
ex1_model_dense.add(Dense(units=16, activation='relu'))
ex1_model_dense.add(Dense(units=16, activation='relu'))
ex1_model_dense.add(Dense(units=16, activation='relu'))
ex1_model_dense.add(Dense(units=16, activation='relu'))
ex1_model_dense.add(Dense(units=16, activation='relu'))
ex1_model_dense.add(Dense(units=16, activation='relu'))
ex1_model_dense.add(Dense(units=16, activation='relu'))
ex1_model_dense.add(Dense(units=16, activation='relu'))
```





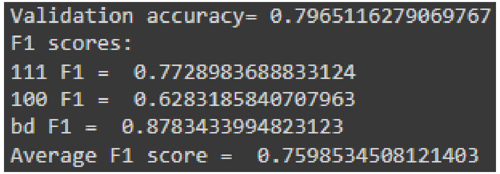
```
Validation accuracy= 0.7009966777408638
F1 scores:
111 F1 =  0.71259418729817
100 F1 =  0.0
bd F1 =  0.8267526188557615
Average F1 score =  0.513115602051310 5
```



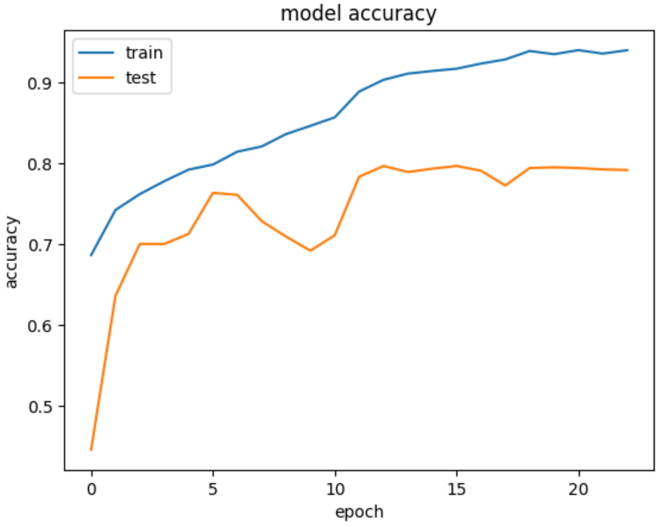
1) Number of fully connected layers: **8**
2) Validation Accuracy: **0.7009966777408638**
3) Any 100-type defects? **No**

- Did the model predict anything in the second column? **No**

Progress



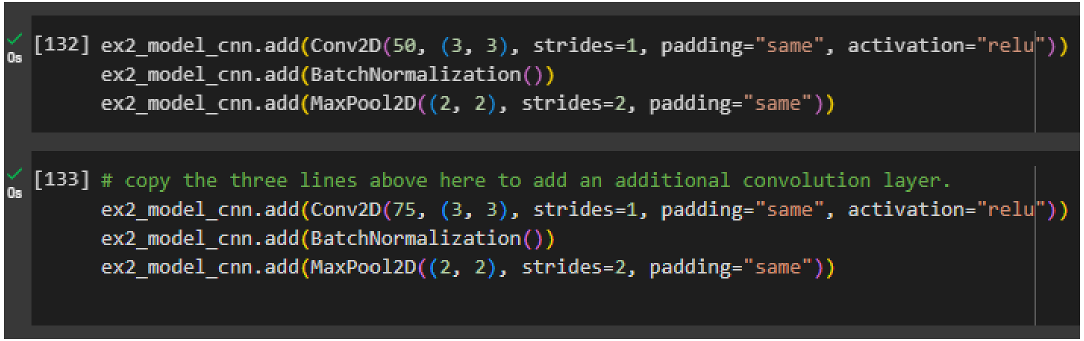
```
Validation accuracy= 0.7965116279069767
F1 scores:
111 F1 =  0.7728983688833124
100 F1 =  0.6283185840707963
bd F1 =  0.8783433994823123
Average F1 score =  0.7598534508121403
```



Exercise 2: Modifying Convolution Layers

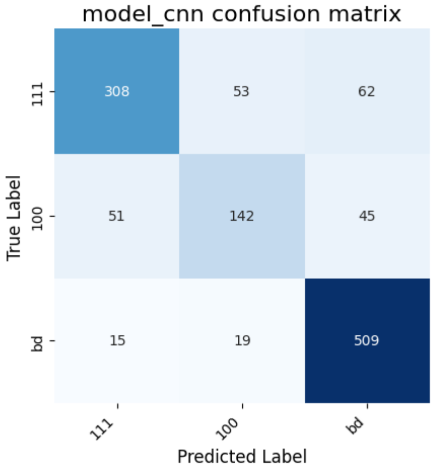
Task: Examine the sensitivity of the convolution structure to modifications

- Added 1 more layer!



```
[132] ex2_model_cnn.add(Conv2D(50, (3, 3), strides=1, padding="same", activation="relu"))
      ex2_model_cnn.add(BatchNormalization())
      ex2_model_cnn.add(MaxPool2D((2, 2), strides=2, padding="same"))

[133] # copy the three lines above here to add an additional convolution layer.
      ex2_model_cnn.add(Conv2D(75, (3, 3), strides=1, padding="same", activation="relu"))
      ex2_model_cnn.add(BatchNormalization())
      ex2_model_cnn.add(MaxPool2D((2, 2), strides=2, padding="same"))
```



1) Did you add or remove one? **Added**
- Output size of the convolution layer? **64x64x50**
1) Validation Accuracy: **0.7965116279069767**
2) Qualitatively, did your model predict any 100 type defects? **Yes**
- Did the model predict anything in the second column? **Yes**; **142**