NAME: JUBESH JOSEPH
EMP-ID: 242017
DATE: 16.07.2024

PROBLEM STATEMENT:

To Classify the drone signals into FSK vs NOT FSK.

Step 1: Collection of Data

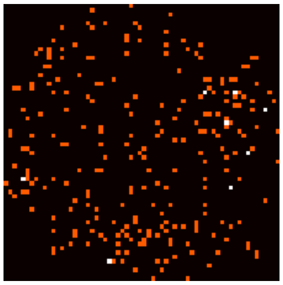
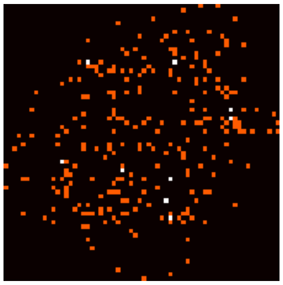
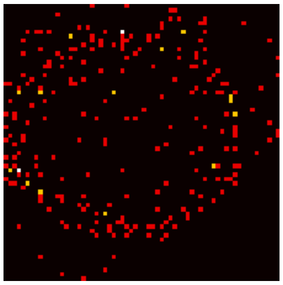
I collected some data for different modulation types using the Vector Signal Generator.
The collected data includes modulations:

1. 2FSK
2. 4FSK
3. 8FSK

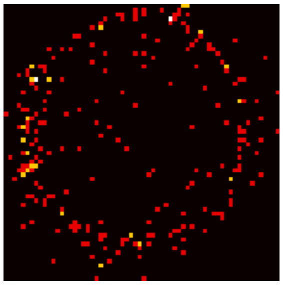
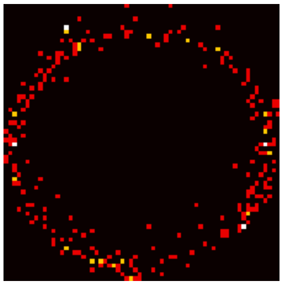
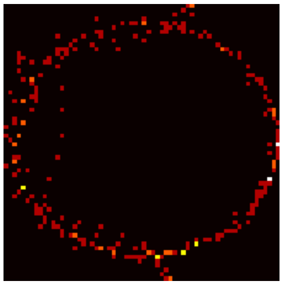
Step 2: Collected some telemetry data of RFD_900

Step 3: Plot the constellation Diagrams.

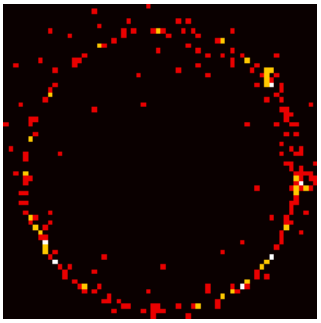
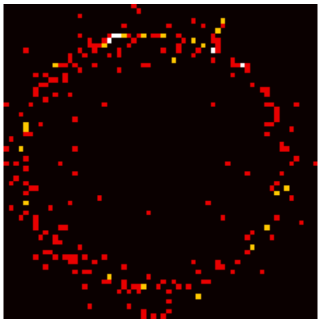
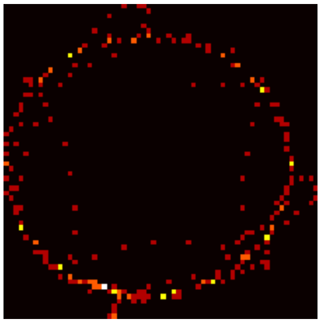
1. 2FSK

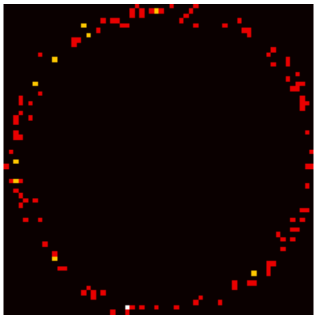
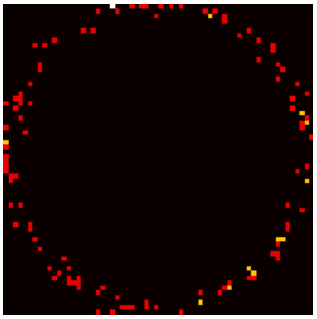
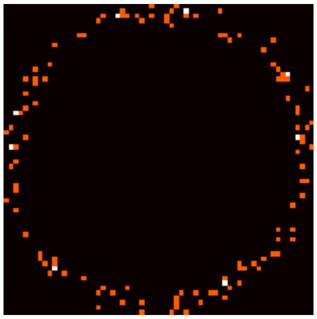
2. 4FSK

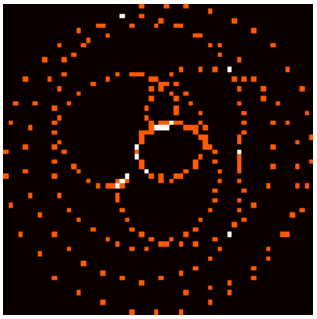
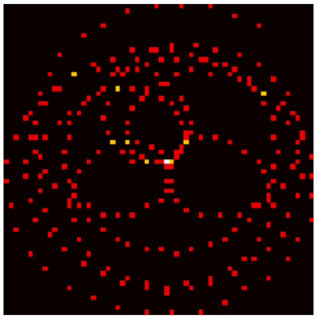
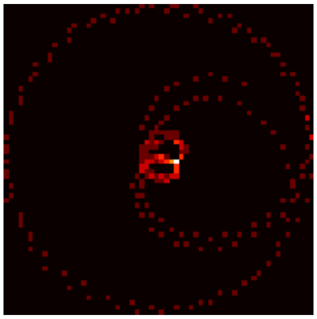
3. 8FSK

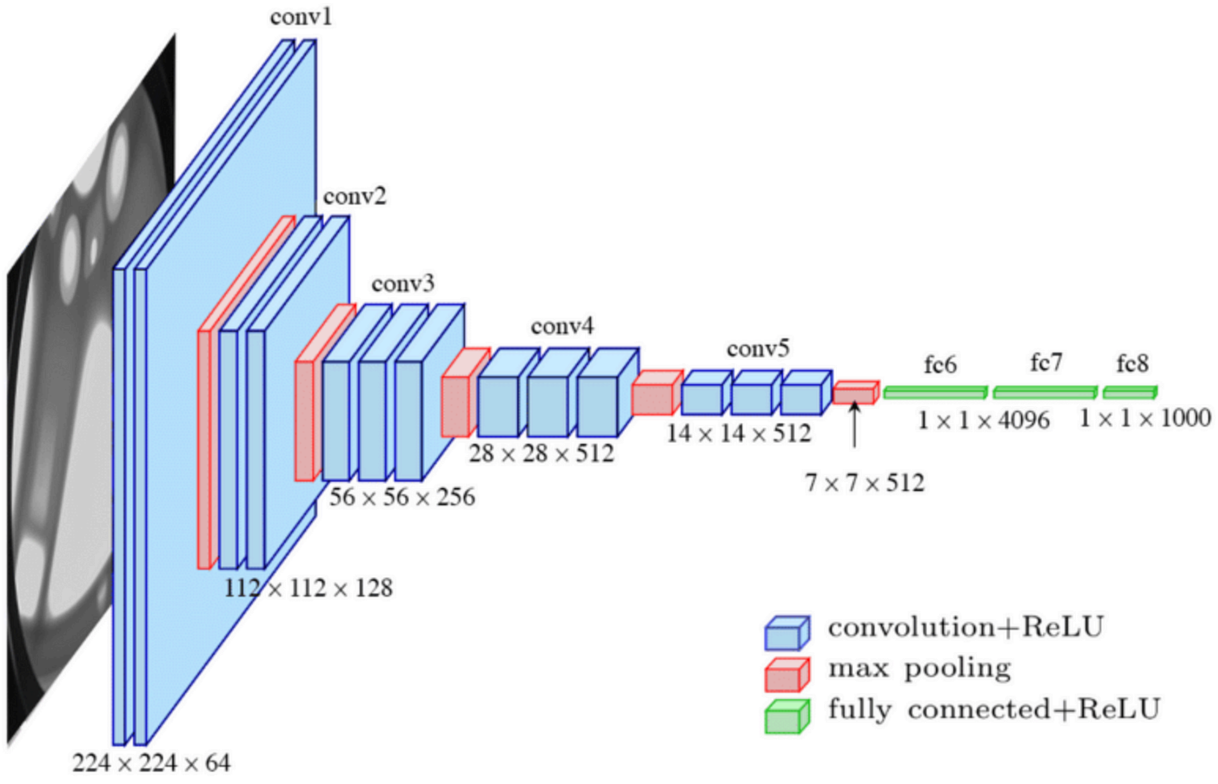
4. Telemetry Data

NON_FSK Constellations:

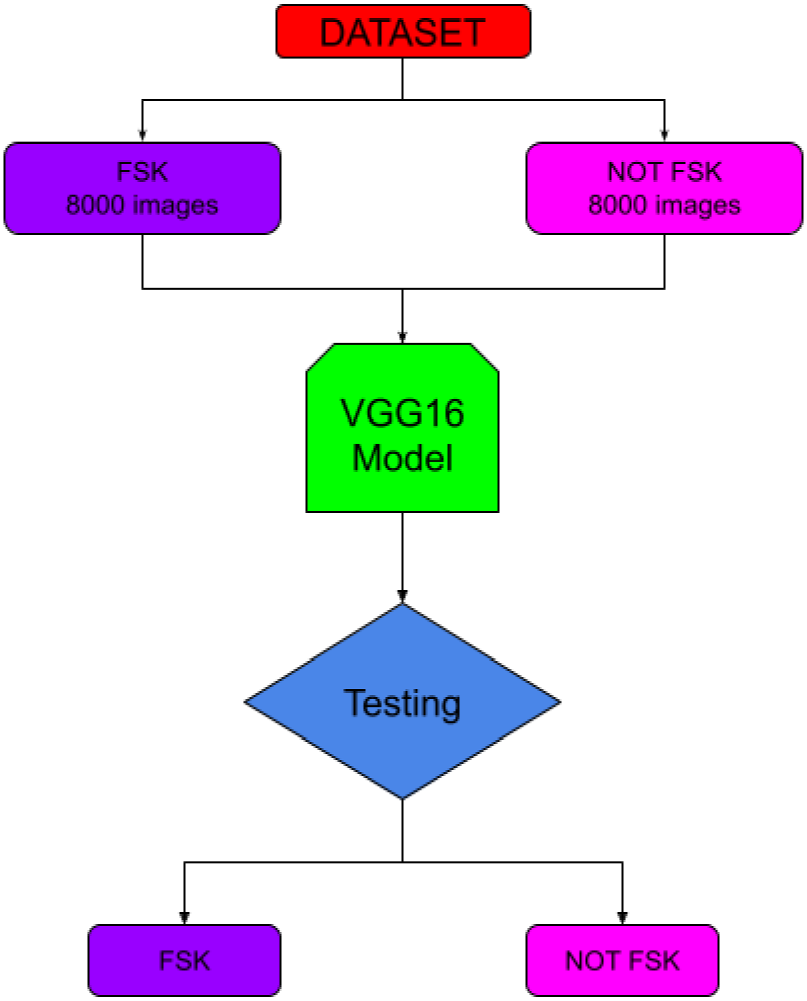
Model Used: VGG16 (without Top Layers)



Details of the model:

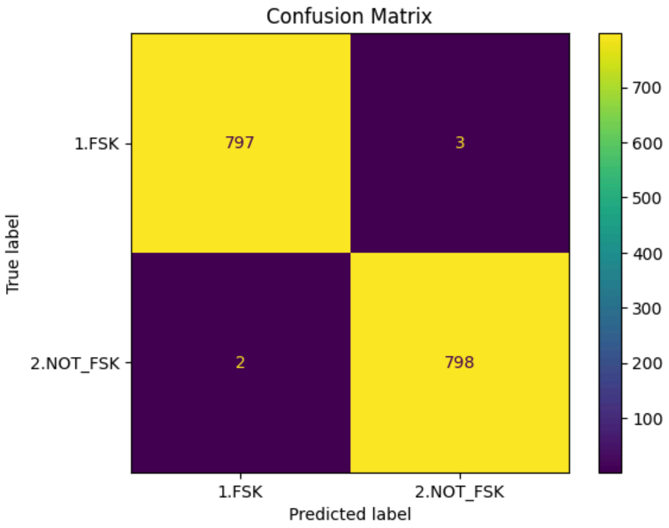
Weights: imagenet
TopLayers: False
Last Layer: Sigmoid
Entropy: binary-crossentropy
Training Time: 27 mins (With GPU)

Flow Chart:

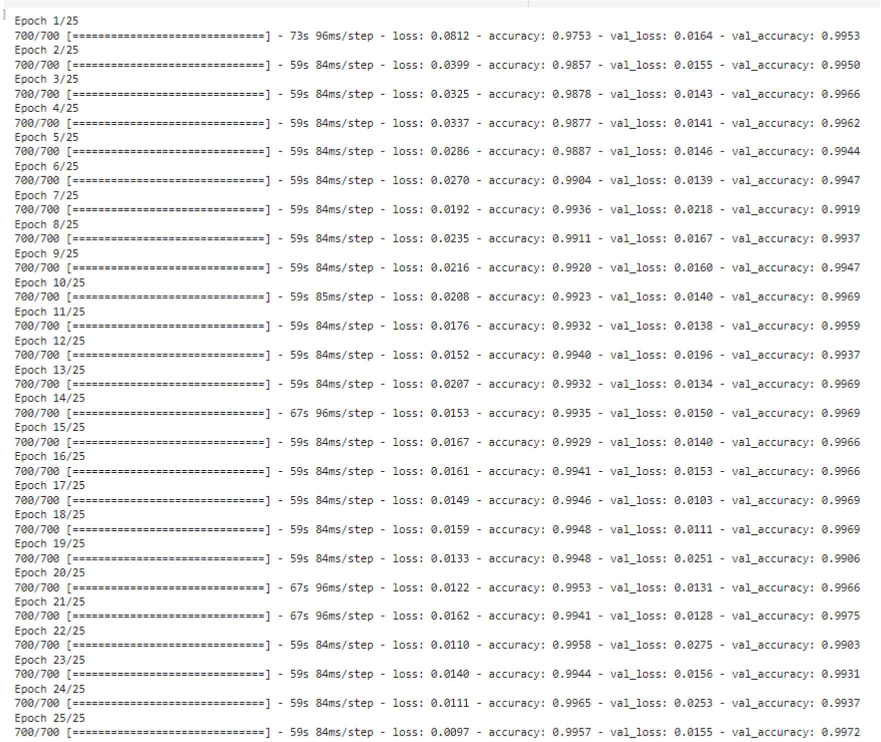

DATASET
FSK
8000 images
NOT FSK
8000 images
VGG16
Model
Testing
FSK
NOT FSK

Outcomes:

1. Confusion Matrix

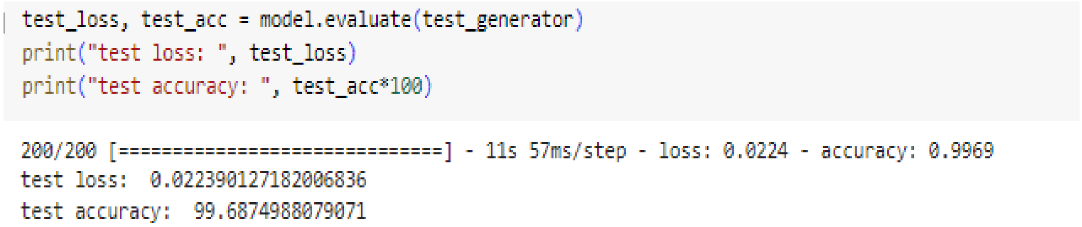


2. Training Progress



```
Epoch 1/25
700/700 [==============================] - 73s 96ms/step - loss: 0.0812 - accuracy: 0.9753 - val_loss: 0.0164 - val_accuracy: 0.9953
Epoch 2/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0399 - accuracy: 0.9857 - val_loss: 0.0155 - val_accuracy: 0.9950
Epoch 3/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0325 - accuracy: 0.9878 - val_loss: 0.0143 - val_accuracy: 0.9966
Epoch 4/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0337 - accuracy: 0.9877 - val_loss: 0.0141 - val_accuracy: 0.9962
Epoch 5/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0286 - accuracy: 0.9887 - val_loss: 0.0146 - val_accuracy: 0.9944
Epoch 6/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0270 - accuracy: 0.9904 - val_loss: 0.0139 - val_accuracy: 0.9947
Epoch 7/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0192 - accuracy: 0.9936 - val_loss: 0.0218 - val_accuracy: 0.9919
Epoch 8/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0235 - accuracy: 0.9911 - val_loss: 0.0167 - val_accuracy: 0.9937
Epoch 9/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0216 - accuracy: 0.9920 - val_loss: 0.0160 - val_accuracy: 0.9947
Epoch 10/25
700/700 [==============================] - 59s 85ms/step - loss: 0.0208 - accuracy: 0.9923 - val_loss: 0.0140 - val_accuracy: 0.9969
Epoch 11/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0176 - accuracy: 0.9932 - val_loss: 0.0138 - val_accuracy: 0.9959
Epoch 12/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0152 - accuracy: 0.9940 - val_loss: 0.0196 - val_accuracy: 0.9937
Epoch 13/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0207 - accuracy: 0.9932 - val_loss: 0.0134 - val_accuracy: 0.9969
Epoch 14/25
700/700 [==============================] - 67s 96ms/step - loss: 0.0153 - accuracy: 0.9935 - val_loss: 0.0150 - val_accuracy: 0.9969
Epoch 15/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0167 - accuracy: 0.9929 - val_loss: 0.0140 - val_accuracy: 0.9966
Epoch 16/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0161 - accuracy: 0.9941 - val_loss: 0.0153 - val_accuracy: 0.9966
Epoch 17/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0149 - accuracy: 0.9946 - val_loss: 0.0103 - val_accuracy: 0.9969
Epoch 18/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0159 - accuracy: 0.9948 - val_loss: 0.0111 - val_accuracy: 0.9969
Epoch 19/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0133 - accuracy: 0.9948 - val_loss: 0.0251 - val_accuracy: 0.9906
Epoch 20/25
700/700 [==============================] - 67s 96ms/step - loss: 0.0122 - accuracy: 0.9953 - val_loss: 0.0131 - val_accuracy: 0.9966
Epoch 21/25
700/700 [==============================] - 67s 96ms/step - loss: 0.0162 - accuracy: 0.9941 - val_loss: 0.0128 - val_accuracy: 0.9975
Epoch 22/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0110 - accuracy: 0.9958 - val_loss: 0.0275 - val_accuracy: 0.9903
Epoch 23/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0140 - accuracy: 0.9944 - val_loss: 0.0156 - val_accuracy: 0.9931
Epoch 24/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0111 - accuracy: 0.9965 - val_loss: 0.0253 - val_accuracy: 0.9937
Epoch 25/25
700/700 [==============================] - 59s 84ms/step - loss: 0.0097 - accuracy: 0.9957 - val_loss: 0.0155 - val_accuracy: 0.9972
```

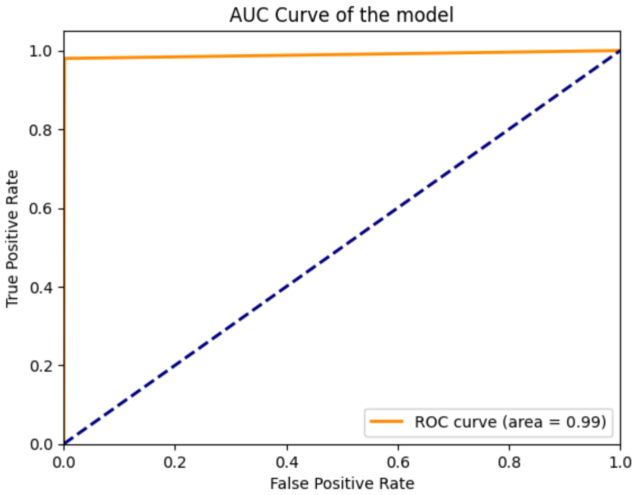
3. Test Accuracy : 99.68 %



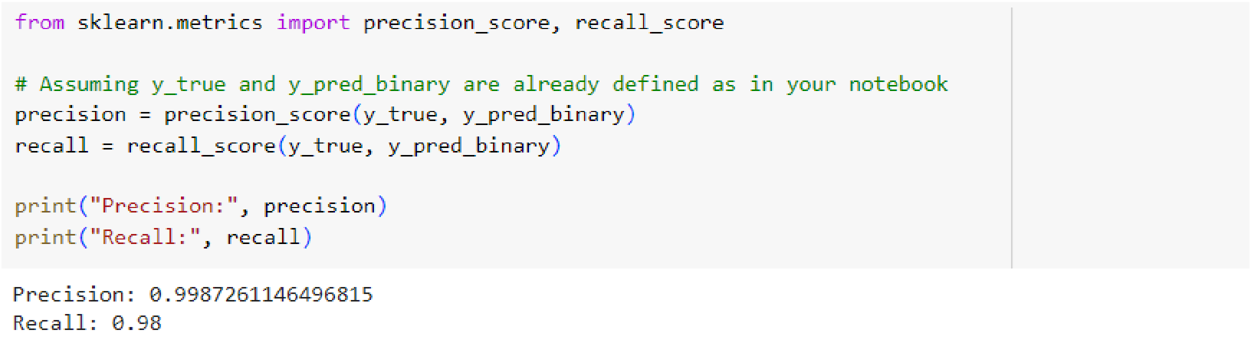
```
test_loss, test_acc = model.evaluate(test_generator)
print("test loss: ", test_loss)
print("test accuracy: ", test_acc*100)
```

```
200/200 [==============================] - 11s 57ms/step - loss: 0.0224 - accuracy: 0.9969
test loss:  0.022390127182006836
test accuracy:  99.6874988079071
```

4. AUC Plot



5. Precision and Recall Scores:



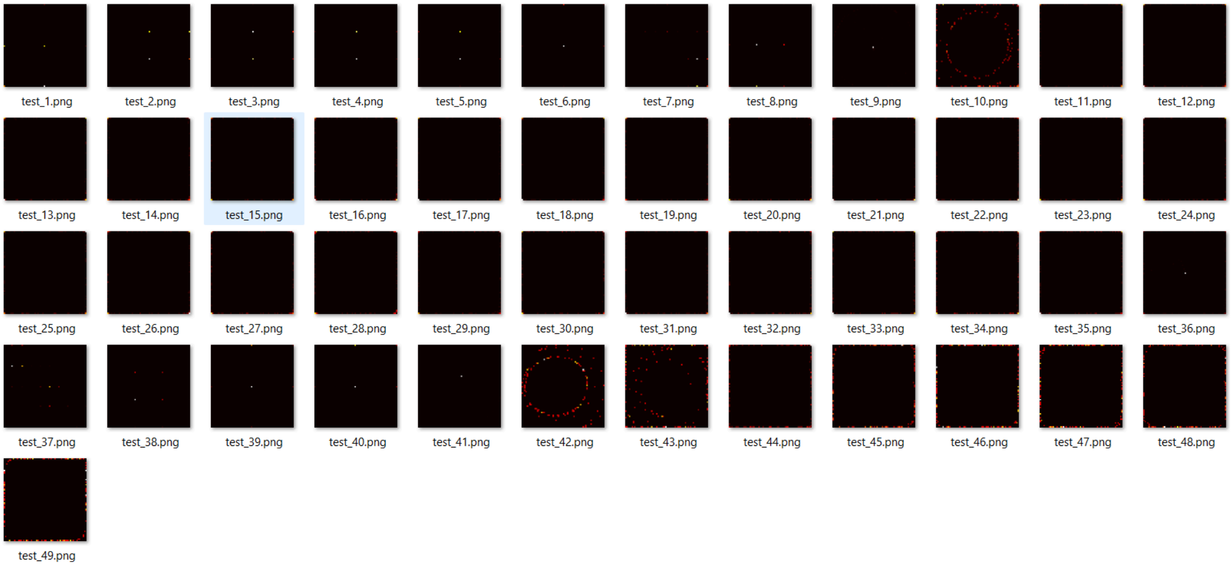
```
from sklearn.metrics import precision_score, recall_score

# Assuming y_true and y_pred_binary are already defined as in your notebook
precision = precision_score(y_true, y_pred_binary)
recall = recall_score(y_true, y_pred_binary)

print("Precision:", precision)
print("Recall:", recall)
```

```
Precision: 0.9987261146496815
Recall: 0.98
```

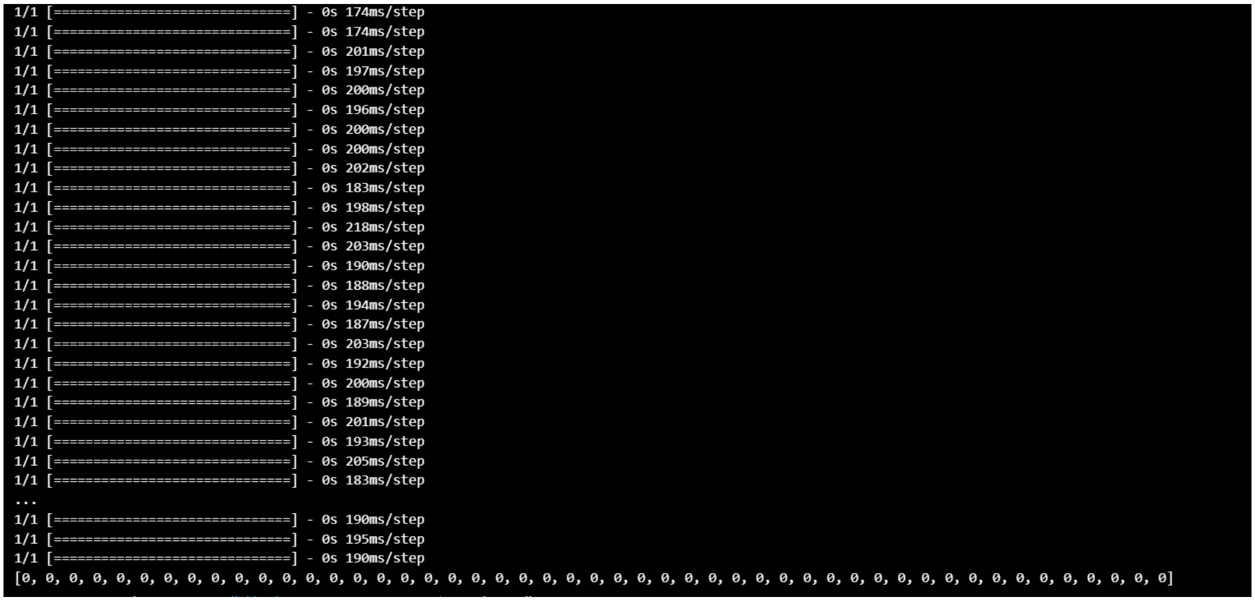
Actual testing

Test 1: rfd900_injection0

1. Generated random 50 frames from the given IQ data.



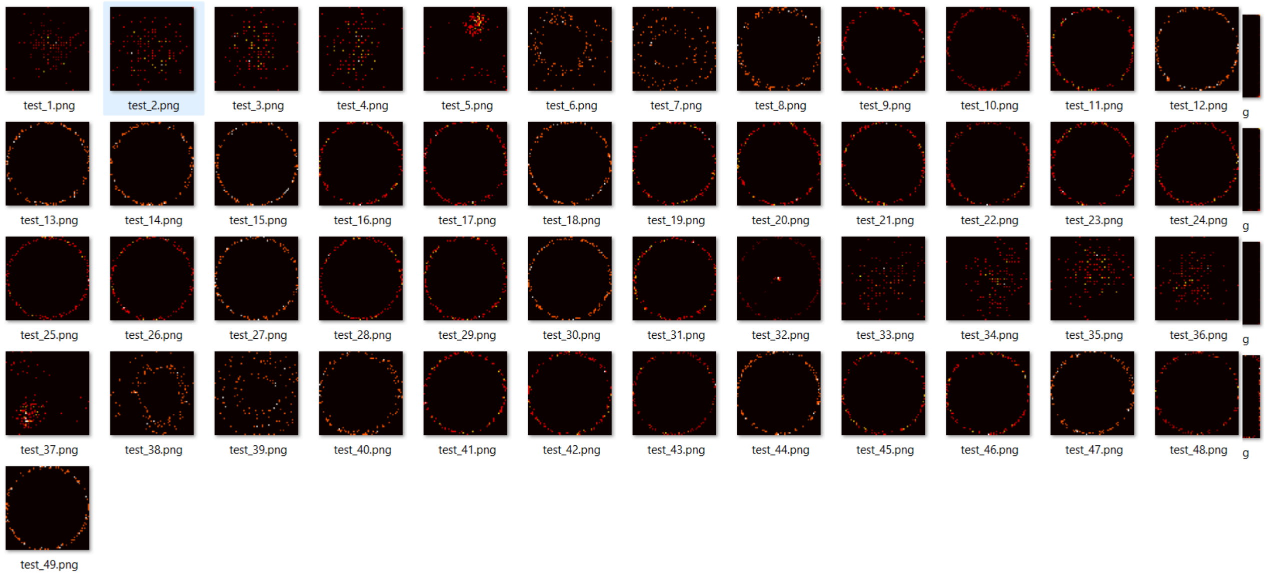
2. Tested the model on them.
3. Outcome we got.



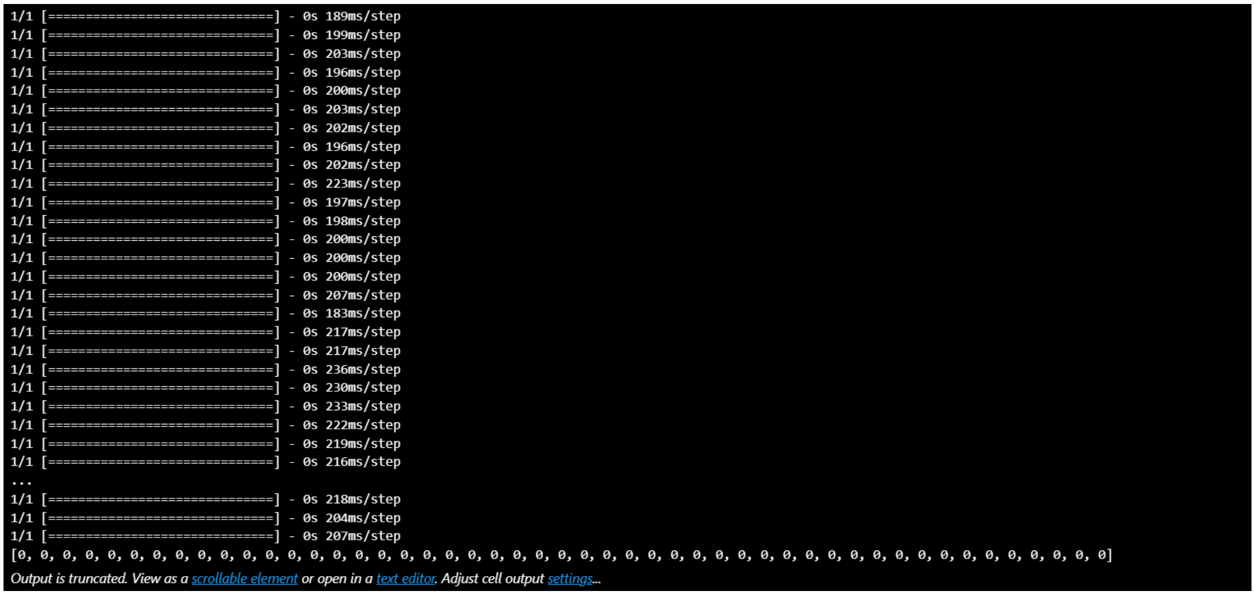
```
1/1 [==============================] - 0s 174ms/step
1/1 [==============================] - 0s 174ms/step
1/1 [==============================] - 0s 201ms/step
1/1 [==============================] - 0s 197ms/step
1/1 [==============================] - 0s 200ms/step
1/1 [==============================] - 0s 196ms/step
1/1 [==============================] - 0s 200ms/step
1/1 [==============================] - 0s 200ms/step
1/1 [==============================] - 0s 202ms/step
1/1 [==============================] - 0s 183ms/step
1/1 [==============================] - 0s 198ms/step
1/1 [==============================] - 0s 218ms/step
1/1 [==============================] - 0s 203ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 194ms/step
1/1 [==============================] - 0s 187ms/step
1/1 [==============================] - 0s 203ms/step
1/1 [==============================] - 0s 192ms/step
1/1 [==============================] - 0s 200ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 201ms/step
1/1 [==============================] - 0s 193ms/step
1/1 [==============================] - 0s 205ms/step
1/1 [==============================] - 0s 183ms/step
...
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 195ms/step
1/1 [==============================] - 0s 190ms/step
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Test2: **rfd900_net25**

1. Generated random 50 frames from the given IQ data.



2. Tested the model on them.
3. Outcome we got



```
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 199ms/step
1/1 [==============================] - 0s 203ms/step
1/1 [==============================] - 0s 196ms/step
1/1 [==============================] - 0s 200ms/step
1/1 [==============================] - 0s 203ms/step
1/1 [==============================] - 0s 202ms/step
1/1 [==============================] - 0s 196ms/step
1/1 [==============================] - 0s 202ms/step
1/1 [==============================] - 0s 223ms/step
1/1 [==============================] - 0s 197ms/step
1/1 [==============================] - 0s 198ms/step
1/1 [==============================] - 0s 200ms/step
1/1 [==============================] - 0s 200ms/step
1/1 [==============================] - 0s 200ms/step
1/1 [==============================] - 0s 207ms/step
1/1 [==============================] - 0s 183ms/step
1/1 [==============================] - 0s 217ms/step
1/1 [==============================] - 0s 217ms/step
1/1 [==============================] - 0s 236ms/step
1/1 [==============================] - 0s 230ms/step
1/1 [==============================] - 0s 233ms/step
1/1 [==============================] - 0s 222ms/step
1/1 [==============================] - 0s 219ms/step
1/1 [==============================] - 0s 216ms/step
...
1/1 [==============================] - 0s 218ms/step
1/1 [==============================] - 0s 204ms/step
1/1 [==============================] - 0s 207ms/step
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...