Assignment No 4 : ECG Anomaly Detection

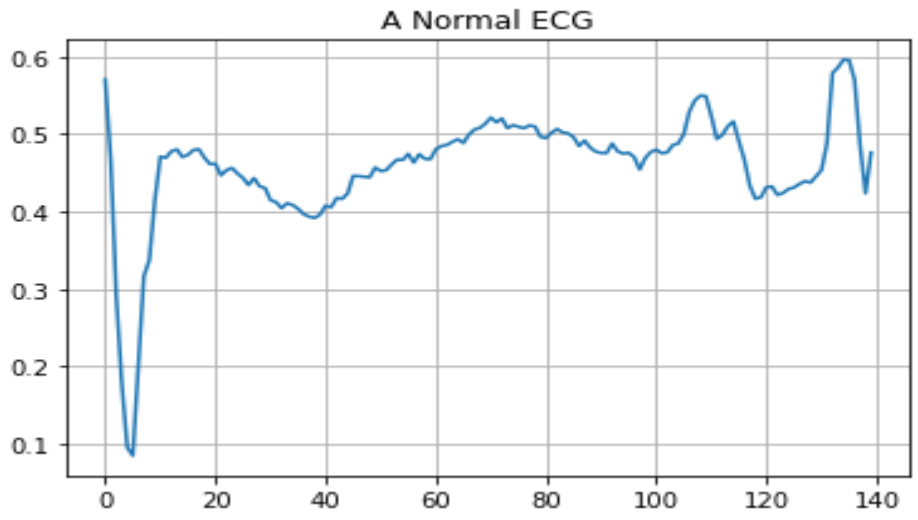
Code:

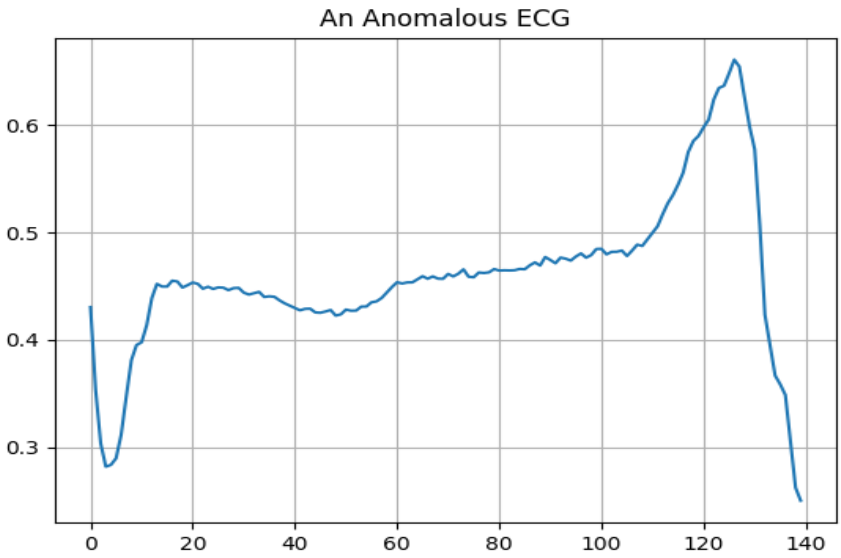
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tensorflow as tf

from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers, losses
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Model

# Download the dataset
dataframe = pd.read_csv('http://storage.googleapis.com/download.tensorflow.org/data/ecg.csv&#39;,
header=None)
raw_data = dataframe.values
dataframe.head()
# The last element contains the labels
labels = raw_data[:, -1]

# The other data points are the electrocadriogram data
data = raw_data[:, 0:-1]

train_data, test_data, train_labels, test_labels = train_test_split(
data, labels, test_size=0.2, random_state=21
)
min_val = tf.reduce_min(train_data)
max_val = tf.reduce_max(train_data)

train_data = (train_data - min_val) / (max_val - min_val)
test_data = (test_data - min_val) / (max_val - min_val)

train_data = tf.cast(train_data, tf.float32)
test_data = tf.cast(test_data, tf.float32)
train_labels = train_labels.astype(bool)
test_labels = test_labels.astype(bool)

normal_train_data = train_data[train_labels]
normal_test_data = test_data[test_labels]

anomalous_train_data = train_data[~train_labels]
anomalous_test_data = test_data[~test_labels]
plt.grid()
plt.plot(np.arange(140), normal_train_data[0])
plt.title("A Normal ECG")
```

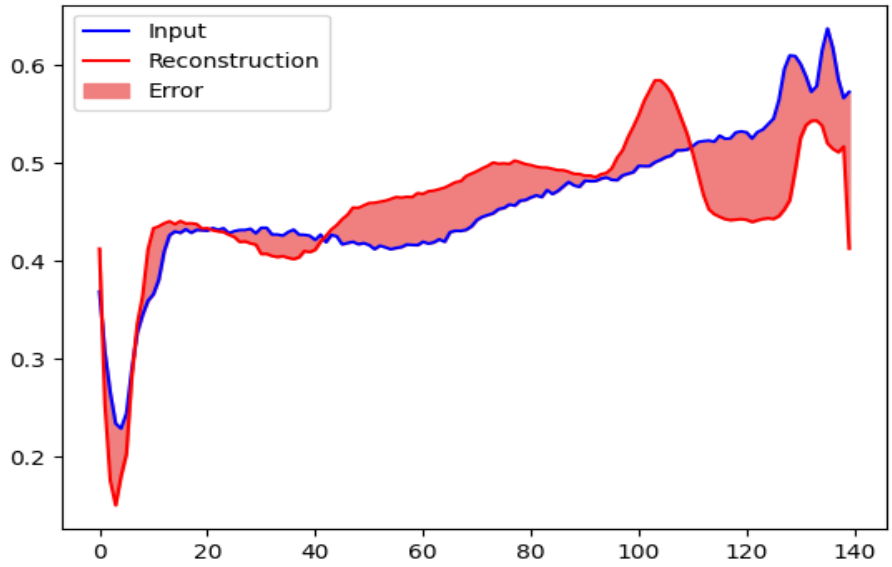
```
plt.show()
plt.grid()
plt.plot(np.arange(140), anomalous_train_data[0])
plt.title("An Anomalous ECG")
plt.show()


class AnomalyDetector(Model):
def __init__(self):
super(AnomalyDetector, self).__init__()
self.encoder = tf.keras.Sequential([
layers.Dense(32, activation="relu"),
layers.Dense(16, activation="relu"),
layers.Dense(8, activation="relu")])

self.decoder = tf.keras.Sequential([
layers.Dense(16, activation="relu"),
layers.Dense(32, activation="relu"),
layers.Dense(140, activation="sigmoid")])

def call(self, x):
encoded = self.encoder(x)
decoded = self.decoder(encoded)
return decoded
autoencoder = AnomalyDetector()
autoencoder.compile(optimizer='adam', loss='mae')
history = autoencoder.fit(normal_train_data, normal_train_data,
epochs=20,
batch_size=512,
validation_data=(test_data, test_data),
shuffle=True)
plt.plot(history.history["loss"], label="Training Loss")
plt.plot(history.history["val_loss"], label="Validation Loss")
plt.legend()
encoded_imgs = autoencoder.encoder(normal_test_data).numpy()
decoded_imgs = autoencoder.decoder(encoded_imgs).numpy()

plt.plot(normal_test_data[0],'b')
plt.plot(decoded_imgs[0],'r')
plt.fill_between(np.arange(140), decoded_imgs[0], normal_test_data[0], color='lightcoral' )
plt.legend(labels=["Input", "Reconstruction", "Error"])
plt.show()
encoded_imgs = autoencoder.encoder(anomalous_test_data).numpy()
decoded_imgs = autoencoder.decoder(encoded_imgs).numpy()

plt.plot(anomalous_test_data[0],'b')
plt.plot(decoded_imgs[0],'r')
```

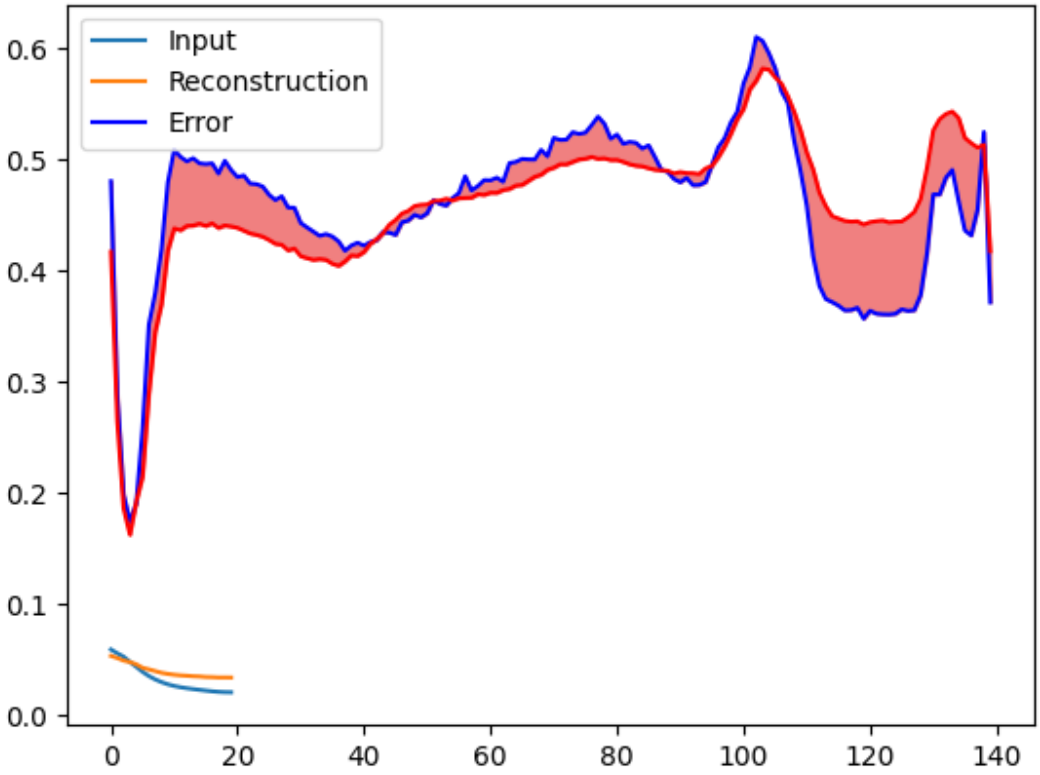
```
plt.fill_between(np.arange(140), decoded_imgs[0], anomalous_test_data[0], color='lightcoral' )
plt.legend(labels=["Input", "Reconstruction", "Error"])
plt.show()

reconstructions = autoencoder.predict(normal_train_data)
train_loss = tf.keras.losses.mae(reconstructions, normal_train_data)

plt.hist(train_loss, bins=50)
plt.xlabel("Train loss")
plt.ylabel("No of examples")
plt.show()
```

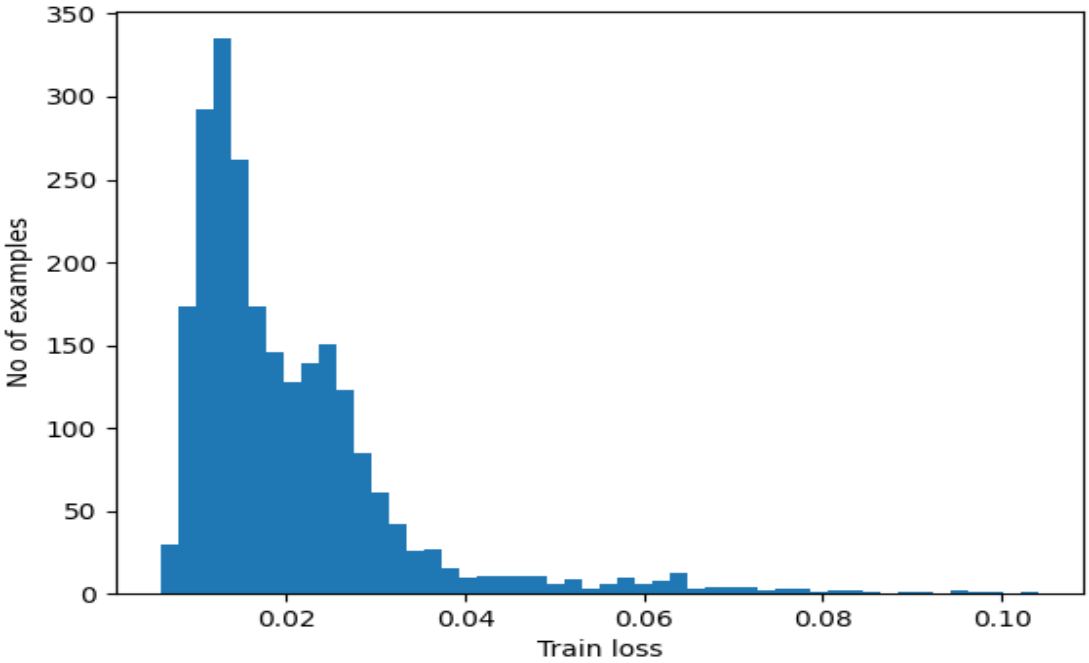
Output:

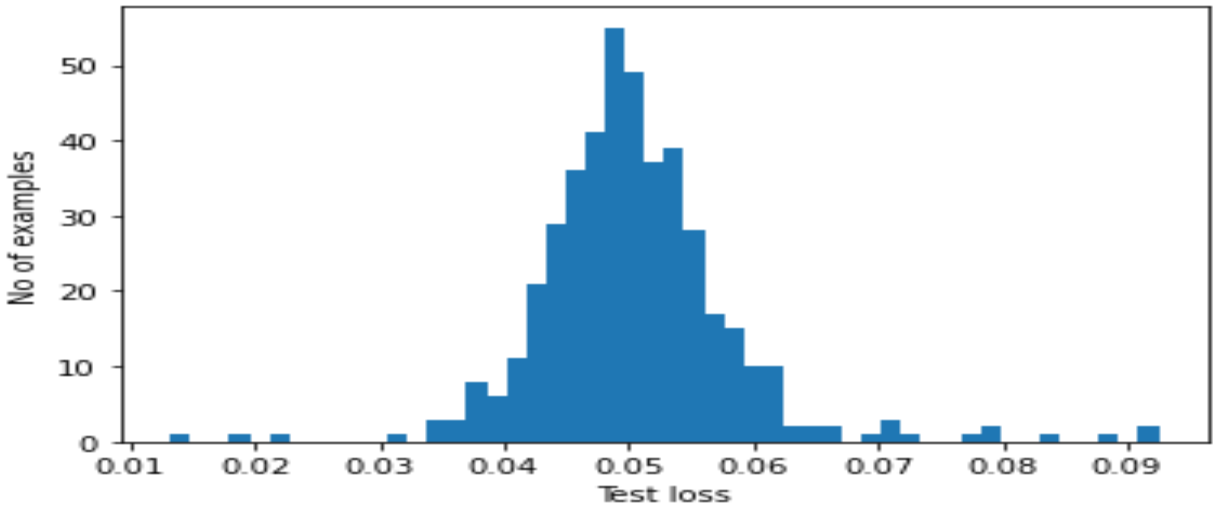



An Anomalous ECG
0.6
0.5
0.4
0.3
0
20
40
60
80
100
120
140


Input
Reconstruction
Error
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100
120
140


Input
Reconstruction
Error
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80
100
120
140


No of examples
350
300
250
200
150
100
50
0
0.02
0.04
0.06
0.08
0.10
Train loss


No of examples
0
10
20
30
40
50
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
Test loss