In [ ]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv1D, MaxPooling1D, UpSampling1D, L

# Load data
data = pd.read_excel('D:\\finalyear\sine-wave-denoise\Dataset\ex1.xlsx')
original_amplitude = data['Amplitude'].values
noised_amplitude = data['Amplitude_noise'].values

# Reshape and scale data
original_amplitude = original_amplitude.reshape(-1, 1)
noised_amplitude = noised_amplitude.reshape(-1, 1)

scaler = MinMaxScaler()
original_amplitude = scaler.fit_transform(original_amplitude)
noised_amplitude = scaler.transform(noised_amplitude)

# Windowing function
window_size = 4500

def create_windows(data, window_size):
    windows = []
    for i in range(len(data) - window_size + 1):
        windows.append(data[i:i + window_size])
    return np.array(windows)

X = create_windows(noised_amplitude, window_size)
y = create_windows(original_amplitude, window_size)

# Reshape for training
X_train = X.reshape(-1, window_size, 1)
y_train = y.reshape(-1, window_size, 1)
```

In [ ]:
```
# Conv1D Autoencoder Model
input_signal = Input(shape=(window_size, 1))
x = Conv1D(16, 3, activation='relu', padding='same')(input_signal)
x = MaxPooling1D(2, padding='same')(x)
x = Conv1D(8, 3, activation='relu', padding='same')(x)
encoded = MaxPooling1D(2, padding='same')(x)

x = Conv1D(8, 3, activation='relu', padding='same')(encoded)
x = UpSampling1D(2)(x)
x = Conv1D(16, 3, activation='relu', padding='same')(x)
x = UpSampling1D(2)(x)
decoded = Conv1D(1, 3, activation='relu', padding='same')(x)

autoencoder = Model(input_signal, decoded)
autoencoder.compile(optimizer='adam', loss='mean_squared_error')

# Train Conv1D Autoencoder
autoencoder.fit(X_train, y_train, epochs=20, batch_size=64, validation_split=0.2
denoised_signal = autoencoder.predict(X_train)
denoised_signal = scaler.inverse_transform(denoised_signal.reshape(-1, 1))
```

```
Epoch 1/20
7/7 [==============================] - 209s 909ms/step - loss: 0.3078 - val_loss:
0.2603
Epoch 2/20
7/7 [==============================] - 3s 415ms/step - loss: 0.2175 - val_loss:
0.1621
Epoch 3/20
7/7 [==============================] - 3s 417ms/step - loss: 0.1217 - val_loss:
0.0714
Epoch 4/20
7/7 [==============================] - 3s 412ms/step - loss: 0.0513 - val_loss:
0.0395
Epoch 5/20
7/7 [==============================] - 3s 376ms/step - loss: 0.0453 - val_loss:
0.0479
Epoch 6/20
7/7 [==============================] - 3s 398ms/step - loss: 0.0426 - val_loss:
0.0329
Epoch 7/20
7/7 [==============================] - 3s 385ms/step - loss: 0.0319 - val_loss:
0.0311
Epoch 8/20
7/7 [==============================] - 3s 397ms/step - loss: 0.0283 - val_loss:
0.0237
Epoch 9/20
7/7 [==============================] - 3s 389ms/step - loss: 0.0210 - val_loss:
0.0162
Epoch 10/20
7/7 [==============================] - 3s 403ms/step - loss: 0.0136 - val_loss:
0.0095
Epoch 11/20
7/7 [==============================] - 3s 387ms/step - loss: 0.0074 - val_loss:
0.0040
Epoch 12/20
7/7 [==============================] - 3s 405ms/step - loss: 0.0030 - val_loss:
0.0016
Epoch 13/20
7/7 [==============================] - 3s 385ms/step - loss: 0.0016 - val_loss:
0.0016
Epoch 14/20
7/7 [==============================] - 3s 386ms/step - loss: 0.0018 - val_loss:
0.0016
Epoch 15/20
7/7 [==============================] - 3s 393ms/step - loss: 0.0016 - val_loss:
0.0013
Epoch 16/20
7/7 [==============================] - 3s 402ms/step - loss: 0.0014 - val_loss:
0.0012
Epoch 17/20
7/7 [==============================] - 3s 385ms/step - loss: 0.0013 - val_loss:
0.0012
Epoch 18/20
7/7 [==============================] - 3s 405ms/step - loss: 0.0013 - val_loss:
0.0011
Epoch 19/20
7/7 [==============================] - 3s 406ms/step - loss: 0.0012 - val_loss:
0.0011
Epoch 20/20
7/7 [==============================] - 3s 413ms/step - loss: 0.0012 - val_loss:
```

```
0.0011
16/16 [==============================] - 4s 96ms/step
```

In [ ]:

```
# LSTM Autoencoder Model
input_signal_lstm = Input(shape=(window_size, 1))
encoded_lstm = LSTM(64, activation='relu', return_sequences=True)(input_signal_l
encoded_lstm = LSTM(32, activation='relu', return_sequences=False)(encoded_lstm)

decoded_lstm = RepeatVector(window_size)(encoded_lstm)
decoded_lstm = LSTM(32, activation='relu', return_sequences=True)(decoded_lstm)
decoded_lstm = LSTM(64, activation='relu', return_sequences=True)(decoded_lstm)
decoded_lstm = Conv1D(1, 3, activation='relu', padding='same')(decoded_lstm)

lstm_autoencoder = Model(input_signal_lstm, decoded_lstm)
lstm_autoencoder.compile(optimizer='adam', loss='mean_squared_error')

# Train LSTM Autoencoder
lstm_autoencoder.fit(X_train, y_train, epochs=20, batch_size=64, validation_spli
denoised_signal_lstm = lstm_autoencoder.predict(X_train)
denoised_signal_lstm = scaler.inverse_transform(denoised_signal_lstm.reshape(-1,
```

```
Epoch 1/20
7/7 [==============================] - 279s 38s/step - loss: 0.3386 - val_loss:
0.2740
Epoch 2/20
7/7 [==============================] - 277s 40s/step - loss: 0.2089 - val_loss:
0.1600
Epoch 3/20
7/7 [==============================] - 281s 40s/step - loss: 0.1541 - val_loss:
0.1377
Epoch 4/20
7/7 [==============================] - 263s 38s/step - loss: 0.1405 - val_loss:
0.1297
Epoch 5/20
7/7 [==============================] - 278s 40s/step - loss: 0.1350 - val_loss:
0.1280
Epoch 6/20
7/7 [==============================] - 351s 52s/step - loss: 0.1307 - val_loss:
0.1279
Epoch 7/20
7/7 [==============================] - 301s 44s/step - loss: 0.1292 - val_loss:
0.1278
Epoch 8/20
7/7 [==============================] - 280s 41s/step - loss: 0.1279 - val_loss:
0.1276
Epoch 9/20
7/7 [==============================] - 283s 39s/step - loss: 0.1268 - val_loss:
0.1275
Epoch 10/20
7/7 [==============================] - 270s 39s/step - loss: 0.1261 - val_loss:
0.1276
Epoch 11/20
7/7 [==============================] - 726s 115s/step - loss: 0.1256 - val_loss:
0.1277
Epoch 12/20
7/7 [==============================] - 192s 28s/step - loss: 0.1256 - val_loss:
0.1277
Epoch 13/20
7/7 [==============================] - 214s 31s/step - loss: 0.1256 - val_loss:
0.1277
Epoch 14/20
7/7 [==============================] - 262s 37s/step - loss: 0.1255 - val_loss:
0.1276
Epoch 15/20
7/7 [==============================] - 2350s 385s/step - loss: 0.1255 - val_loss:
0.1277
Epoch 16/20
7/7 [==============================] - 173s 25s/step - loss: 0.1255 - val_loss:
0.1278
Epoch 17/20
7/7 [==============================] - 202s 29s/step - loss: 0.1254 - val_loss:
0.1277
Epoch 18/20
7/7 [==============================] - 1001s 162s/step - loss: 0.1254 - val_loss:
0.1279
Epoch 19/20
7/7 [==============================] - 179s 26s/step - loss: 0.1253 - val_loss:
0.1280
Epoch 20/20
7/7 [==============================] - 218s 31s/step - loss: 0.1252 - val_loss:
```

```
0.1279
16/16 [==============================] - 76s 5s/step
```

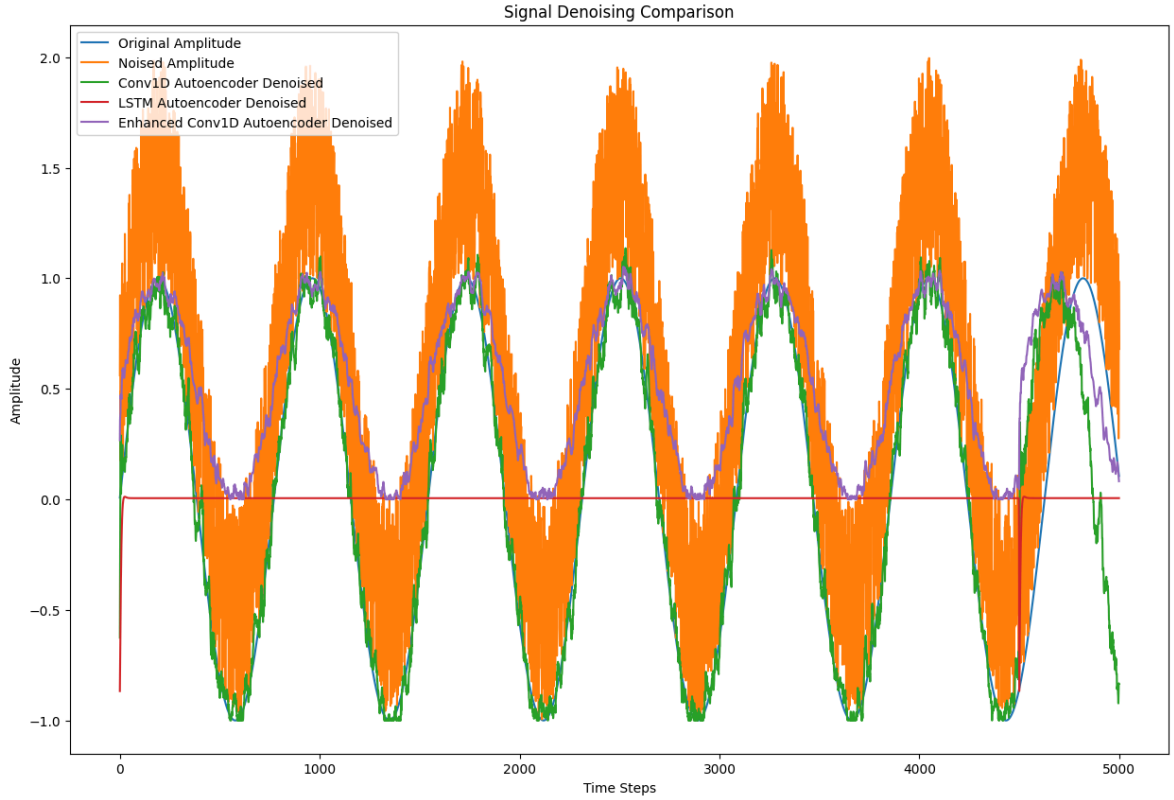
In [ ]:

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv1D, MaxPooling1D, UpSampling1D
from tensorflow.keras.optimizers import Adam

# Define the input layer
input_signal = Input(shape=(4500, 1))

# Build the convolutional layers
x = Conv1D(16, kernel_size=3, activation='relu', padding='same')(input_signal)
x = MaxPooling1D(pool_size=2, padding='same')(x)
x = Conv1D(8, kernel_size=3, activation='relu', padding='same')(x)
encoded = MaxPooling1D(pool_size=2, padding='same')(x)

# Build the upsampling layers
x = Conv1D(8, kernel_size=3, activation='relu', padding='same')(encoded)
x = UpSampling1D(size=2)(x)
x = Conv1D(16, kernel_size=3, activation='relu', padding='same')(x)
x = UpSampling1D(size=2)(x)
decoded = Conv1D(1, kernel_size=3, activation='relu', padding='same')(x)

# Compile the model
conv2_autoencoder = Model(inputs=input_signal, outputs=decoded)
conv2_autoencoder.compile(optimizer=Adam(), loss='mean_squared_error')

# Train the model
conv2_autoencoder.fit(X_train, y_train, epochs=20, batch_size=64, validation_spl
denoised_signal_conv2 = conv2_autoencoder.predict(X_train)
```

```
Epoch 1/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 6s 234ms/step - loss: 0.3319 - val_loss: 0.2741
Epoch 2/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 194ms/step - loss: 0.2456 - val_loss: 0.1772
Epoch 3/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 195ms/step - loss: 0.1488 - val_loss: 0.0755
Epoch 4/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 194ms/step - loss: 0.0585 - val_loss: 0.0341
Epoch 5/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 320ms/step - loss: 0.0380 - val_loss: 0.0439
Epoch 6/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 197ms/step - loss: 0.0408 - val_loss: 0.0285
Epoch 7/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 196ms/step - loss: 0.0280 - val_loss: 0.0283
Epoch 8/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 193ms/step - loss: 0.0274 - val_loss: 0.0243
Epoch 9/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 195ms/step - loss: 0.0231 - val_loss: 0.0205
Epoch 10/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 194ms/step - loss: 0.0201 - val_loss: 0.0176
Epoch 11/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 201ms/step - loss: 0.0169 - val_loss: 0.0143
Epoch 12/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 275ms/step - loss: 0.0134 - val_loss: 0.0094
Epoch 13/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 324ms/step - loss: 0.0083 - val_loss: 0.0052
Epoch 14/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 195ms/step - loss: 0.0046 - val_loss: 0.0024
Epoch 15/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 195ms/step - loss: 0.0023 - val_loss: 0.0016
Epoch 16/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 194ms/step - loss: 0.0019 - val_loss: 0.0016
Epoch 17/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 194ms/step - loss: 0.0018 - val_loss: 0.0014
Epoch 18/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 198ms/step - loss: 0.0016 - val_loss: 0.0014
Epoch 19/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 323ms/step - loss: 0.0015 - val_loss: 0.0013
Epoch 20/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 199ms/step - loss: 0.0014 - val_loss: 0.0013
16/16 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step
```

In [ ]:

```
# Visualization
time_steps = np.arange(len(data))
plt.figure(figsize=(15, 10))
plt.plot(time_steps, data['Amplitude'], label='Original Amplitude')
plt.plot(time_steps, data['Amplitude_noise'], label='Noised Amplitude')
plt.plot(time_steps, denoised_signal.flatten()[:len(data)], label='Conv1D Autoen
plt.plot(time_steps, denoised_signal_lstm.flatten()[:len(data)], label='LSTM Aut
plt.plot(time_steps, denoised_signal_conv2.flatten()[:len(data)], label='Enhance

plt.xlabel('Time Steps')
plt.ylabel('Amplitude')
plt.title('Signal Denoising Comparison')
plt.legend()
plt.show()
```


Signal Denoising Comparison
Original Amplitude
Noised Amplitude
Conv1D Autoencoder Denoised
LSTM Autoencoder Denoised
Enhanced Conv1D Autoencoder Denoised
Amplitude
Time Steps
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
1000
2000
3000
4000
5000