

Link para o trabalho <https://www.youtube.com/watch?v=c-1cBNhvNq8>

Caso o vídeo no youtube não apareça:

<https://drive.google.com/drive/folders/1VH9FXUeV4J1nAl1-UWfNzE6h1Vm5PaHM?usp=sharing>

O presente trabalho demonstra a aplicação de um autoencoder para a remoção de ruído em sinais biomédicos simulados.

O objetivo deste projeto foi desenvolver um autoencoder para redução de ruído em sinais biomédicos. O modelo é capaz de filtrar diferentes tipos de ruídos adicionados artificialmente a sinais de ECG, EEG, EMG e EOG.

Primeiramente, foi gerado sinais sintéticos representando Eletrocardiogramas (ECG) que simulam a atividade elétrica do coração, Eletroencefalogramas (EEG) que simulam a atividade elétrica do cérebro, Eletromiogramas (EMG) que simulam a atividade elétrica dos músculos e Eletrooculogramas (EOG) que simulam a atividade elétrica dos olhos. Exemplos de equipamentos que representam esses sinais:

- ECG: Monitores cardíacos, Holter.
- EEG: Capacetes de EEG, sistemas de monitoramento de sono.
- EMG: Equipamentos de eletromiografia, dispositivos de reabilitação muscular.
- EOG: Sistemas de rastreamento ocular, dispositivos de estudo do sono

Em seguida, adicionamos diferentes tipos de ruído a esses sinais para simular cenários reais, diferentes tipos de ruído foram adicionados aos sinais gerados. Os tipos de ruído incluem:

- Powerline Noise (Ruído de Linha de Potência)
 - Descrição: Este tipo de ruído é causado pela interferência de redes elétricas, geralmente a 50 ou 60 Hz, dependendo da região.
 - Origem: Fios de energia elétrica nas proximidades de dispositivos biomédicos.
 - Efeito no Sinal: Aparece como uma oscilação senoidal constante no sinal, o que pode mascarar as características do sinal biomédico real.
 - Simulação: Adicionado ao sinal com a equação $\text{noise_amplitude} * \text{np.sin}(2 * \text{np.pi} * 50 * t)$ onde 50 é a frequência da rede elétrica.
- Baseline Wander (Variação de Linha de Base)
 - Descrição: Variação lenta e de baixa frequência que pode ocorrer devido a mudanças na posição dos eletrodos ou respiração.
 - Origem: Movimento do paciente ou mau contato dos eletrodos.
 - Efeito no Sinal: Resulta em uma oscilação de baixa frequência que desloca a linha de base do sinal, dificultando a análise precisa.
 - Simulação: Gerado acumulando uma série de valores normais aleatórios e suavizando o resultado.
- Electrode Motion (Movimento do Eletrodo)

- Descrição: Ruído causado pelo movimento dos eletrodos em contato com a pele.
- Origem: Paciente se movendo ou ajustando os eletrodos.
- Efeito no Sinal: Aparece como picos ou ruídos transitórios, interferindo na precisão do sinal.
- Simulação: Adiciona picos aleatórios no sinal em posições selecionadas aleatoriamente.
- White Noise (Ruído Branco)
 - Descrição: Ruído de espectro plano, ou seja, todas as frequências têm igual intensidade.
 - Origem: Equipamento eletrônico, ambiente circundante ou outras fontes aleatórias.
 - Efeito no Sinal: Introduz variações aleatórias ao sinal que podem obscurecer as características do sinal real.
 - Simulação: Adicionado ao sinal com a equação `np.random.normal(0, noise_amplitude, len(signal))`.

A função `add_noise` foi utilizada para adicionar esses ruídos ao sinal original.

Os dados foram preparados para treinamento e teste através da função `prepare_data_with_noise`, que gerou amostras de sinais ruidosos a partir dos sinais originais.

O autoencoder foi construído utilizando a biblioteca Keras. A arquitetura do autoencoder incluiu:

- Encoder: Camadas densas com regularização L2 e dropout para prevenir overfitting.
- Decoder: Camadas densas para reconstruir o sinal original.

A função `build_autoencoder` foi utilizada para construir e compilar o modelo.

O modelo foi treinado utilizando um conjunto de dados de treinamento com 70% de divisão para validação. O treinamento foi monitorado usando EarlyStopping para evitar overfitting.

Após o treinamento, o modelo foi avaliado e os sinais denoised foram gerados. A função `plot_signals` foi utilizada para comparar os sinais originais, ruidosos e denoised, além de mostrar o histórico de erro do modelo.

Conclusão

O autoencoder desenvolvido se mostrou eficaz na redução de ruído em sinais biomédicos sintéticos. As técnicas de regularização e dropout ajudaram a evitar overfitting, e o uso de `EarlyStopping` garantiu que o modelo não fosse treinado em excesso. Os resultados indicam que a abordagem pode ser aplicada a dados reais, com ajustes adequados.

Este projeto demonstra a viabilidade de usar autoencoders para tarefas de denoising em sinais biomédicos, oferecendo uma base para pesquisas e aplicações práticas.

```
In [ ]: import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import Huber
from tensorflow.keras.regularizers import l2

def generate_signal(duration, frequency, signal_type):
    """Generates a synthetic biomedical signal of the specified type."""
    time = np.linspace(0, duration, int(frequency * duration))

    if signal_type == 'ecg':
        heart_rate = 1.2
        signal = (np.sin(2 * np.pi * heart_rate * time) +
                  0.5 * np.sin(2 * np.pi * 2 * heart_rate * time) +
                  0.2 * np.sin(2 * np.pi * 3 * heart_rate * time))
    elif signal_type == 'eeg':
        signal = (np.sin(2 * np.pi * 10 * time) +
                  np.sin(2 * np.pi * 20 * time) +
                  np.sin(2 * np.pi * 30 * time) +
                  np.random.normal(0, 0.1, len(time)))
    elif signal_type == 'emg':
        signal = np.random.randn(len(time))
    elif signal_type == 'eog':
        signal = np.sin(2 * np.pi * time) + 0.5 * np.random.randn(len(time))
    else:
        raise ValueError(f"Invalid signal type: {signal_type}")

    return time, signal

def add_noise(signal, noise_type, noise_amplitude):
    """Adds specific types of noise to a biomedical signal."""
    t = np.arange(len(signal)) / len(signal)

    if noise_type == 'powerline':
        noise = noise_amplitude * np.sin(2 * np.pi * 50 * t)
    elif noise_type == 'baseline_wander':
        noise = np.cumsum(np.random.normal(0, 0.1, len(signal))) * noise_amplitude
        noise = np.convolve(noise, np.ones(10) / 10, mode='same')
    elif noise_type == 'electrode_motion':
        num_spikes = int(noise_amplitude * len(signal) / 1000)
        spike_indices = np.random.choice(len(signal), num_spikes, replace=False)
        noise = np.zeros(len(signal))
        noise[spike_indices] = np.random.normal(0, noise_amplitude, num_spikes)
    elif noise_type == 'white_noise':
        noise = np.random.normal(0, noise_amplitude, len(signal))
    else:
        raise ValueError(f"Invalid noise type: {noise_type}")

    return signal + noise

def prepare_data_with_noise(signal, num_samples, noise_amplitude):
    """Prepares training data by adding various noise types to the original signal"""
    noise_types = ['powerline', 'baseline_wander', 'electrode_motion', 'white_no
```

```

X_train = np.tile(signal, (num_samples, 1))
X_noisy = []

for _ in range(num_samples):
    noise_type = np.random.choice(noise_types)
    noisy_signal = add_noise(signal, noise_type, noise_amplitude)
    X_noisy.append(noisy_signal)

return X_train, np.array(X_noisy)

def prepare_data(duration, frequency, signal_type, num_samples, noise_amplitude):
    """Prepares training and noisy data for the specified signal type."""
    time, signal = generate_signal(duration, frequency, signal_type)
    X_train, X_noisy = prepare_data_with_noise(signal, num_samples, noise_amplitude)
    return time, X_train, X_noisy

def build_autoencoder(input_shape):
    """Constructs a simple autoencoder model for noise reduction."""
    input_signal = Input(shape=input_shape) # Get the input shape of the signal

    # Encoder
    # Map the input signal to a representation
    encoded = Dense(256, activation='relu', activity_regularizer=l2(0.01))(input_signal)
    encoded = Dropout(0.2)(encoded) # Dropout Layer to prevent overfitting
    encoded = Dense(128, activation='relu')(encoded) # Second layer with 128 neurons

    # Decoder
    # Map the representation back to the original signal
    decoded = Dense(256, activation='relu')(encoded)
    decoded = Dropout(0.2)(decoded)
    decoded = Dense(input_shape[0], activation='linear')(decoded) # Linear activation

    autoencoder = Model(input_signal, decoded)
    autoencoder.compile(optimizer=Adam(learning_rate=0.001), loss=Huber())

    return autoencoder

def plot_signals(time, X_train, X_noisy, denoised_signals, signal_type, history):
    """Plots the original, noisy, denoised signals and the model's training history"""
    plt.style.use('seaborn-v0_8-whitegrid')
    fig, axes = plt.subplots(3, 3, figsize=(15, 12))

    for i in range(3):
        axes[0, i].plot(time, X_noisy[i], color='gray', linestyle='--', alpha=0.5)
        axes[0, i].fill_between(time, X_noisy[i], color='gray', alpha=0.2)
        axes[0, i].plot(time, X_train[i], color='blue', linestyle='--', alpha=0.3)
        axes[0, i].set_title(f'Original vs. Noisy {signal_type.upper()} (Example {i+1})')
        axes[0, i].set_xlabel('Time (s)', fontsize=10)
        axes[0, i].set_ylabel('Amplitude', fontsize=10)
        axes[0, i].legend(loc='upper right', fontsize=8)
        axes[0, i].grid(axis='y', linestyle='--')

        axes[1, i].plot(time, X_noisy[i], color='gray', linestyle='--', alpha=0.5)
        axes[1, i].fill_between(time, X_noisy[i], color='gray', alpha=0.2)
        axes[1, i].plot(time, denoised_signals[i], color='green', linestyle='--', alpha=0.3)
        axes[1, i].set_title(f'Denoised vs. Noisy {signal_type.upper()} (Example {i+1})')
        axes[1, i].set_xlabel('Time (s)', fontsize=10)
        axes[1, i].set_ylabel('Amplitude', fontsize=10)
        axes[1, i].legend(loc='upper right', fontsize=8)
        axes[1, i].grid(axis='y', linestyle='--')

```

```

axes[2, 0].plot(history.history['loss'], color='black')
axes[2, 0].plot(history.history['val_loss'], color='orange')
axes[2, 0].set_title('Model Error', fontsize=12)
axes[2, 0].set_xlabel('Epochs', fontsize=10)
axes[2, 0].set_ylabel('Error', fontsize=10)
axes[2, 0].legend(['Train', 'Validation'], loc='upper right')
axes[2, 0].grid(axis='y', linestyle='--')

axes[2, 1].axis('off')
axes[2, 2].axis('off')

fig.suptitle(f'Comparison of Original, Noisy, and Denoised {signal_type.upper}')
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

def main(signal_type):
    duration = 10
    frequency = 250
    num_samples = 1000
    noise_amplitude = 1.0

    time, X_train, X_noisy = prepare_data(
        duration,
        frequency,
        signal_type,
        num_samples,
        noise_amplitude
    )

    input_shape = (X_train.shape[1],)
    autoencoder = build_autoencoder(input_shape)

    early_stopping = EarlyStopping(
        monitor='val_loss',
        patience=10,
        restore_best_weights=True
    )















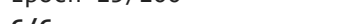







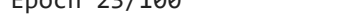
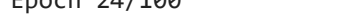
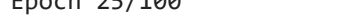
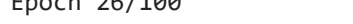
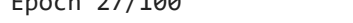

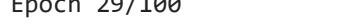
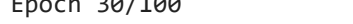
    history = autoencoder.fit(
        X_noisy,
        X_train,
        epochs=100,
        batch_size=128,
        validation_split=0.3,
        callbacks=[early_stopping]
    )

    denoised_signals = autoencoder.predict(X_noisy)

    plot_signals(time, X_train, X_noisy, denoised_signals, signal_type, history)

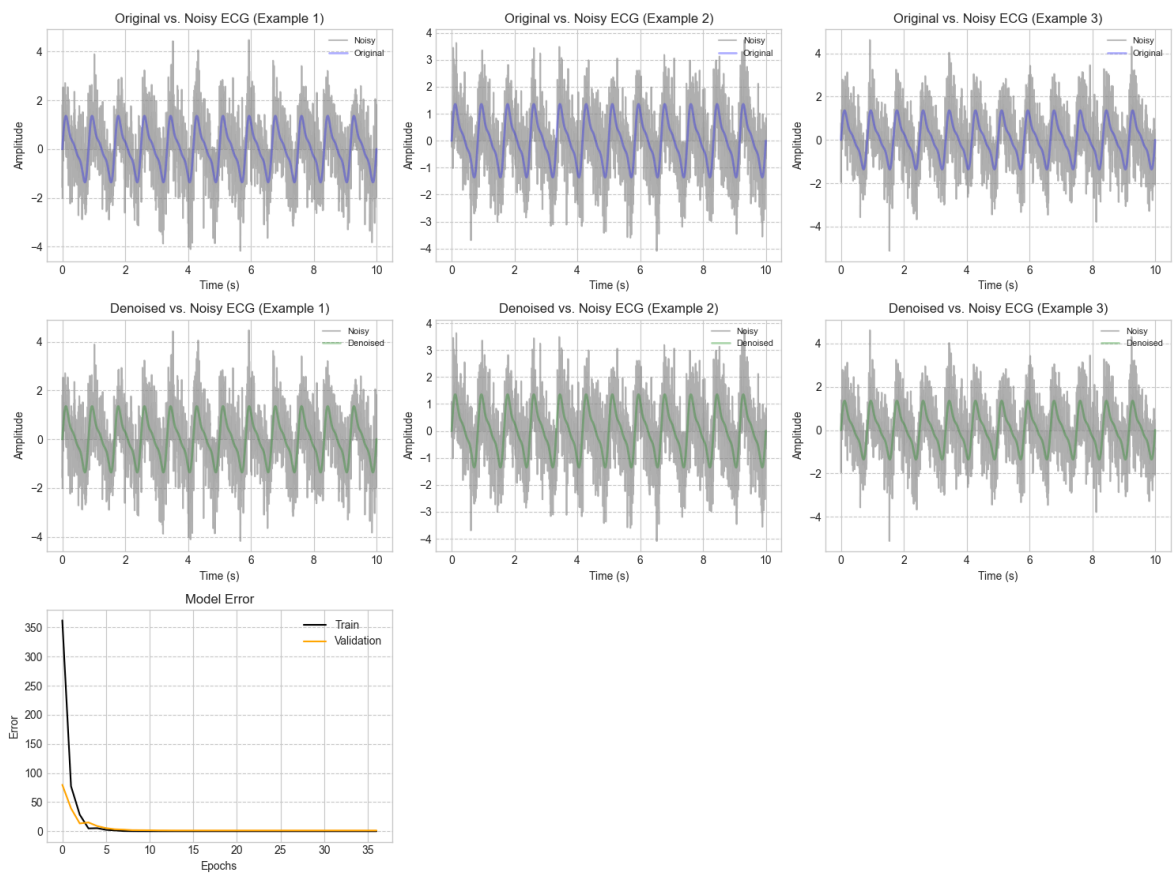
if __name__ == "__main__":
    for signal_type in ['ecg', 'eeg', 'emg', 'eog']:
        main(signal_type)































```

Epoch 1/100
6/6  2s 83ms/step - loss: 546.0801 - val_loss: 79.8028
Epoch 2/100
6/6  0s 35ms/step - loss: 89.3280 - val_loss: 39.3147
Epoch 3/100
6/6  0s 37ms/step - loss: 29.1501 - val_loss: 13.3264
Epoch 4/100
6/6  0s 35ms/step - loss: 5.6501 - val_loss: 15.0359
Epoch 5/100
6/6  0s 52ms/step - loss: 5.3456 - val_loss: 8.8320
Epoch 6/100
6/6  0s 37ms/step - loss: 2.6671 - val_loss: 5.3543
Epoch 7/100
6/6  0s 36ms/step - loss: 1.2242 - val_loss: 3.3686
Epoch 8/100
6/6  0s 36ms/step - loss: 0.2816 - val_loss: 2.7514
Epoch 9/100
6/6  0s 37ms/step - loss: 0.1258 - val_loss: 1.8279
Epoch 10/100
6/6  0s 34ms/step - loss: 0.0246 - val_loss: 1.7438
Epoch 11/100
6/6  0s 35ms/step - loss: 0.0171 - val_loss: 1.6267
Epoch 12/100
6/6  0s 36ms/step - loss: 0.0133 - val_loss: 1.3894
Epoch 13/100
6/6  0s 43ms/step - loss: 0.0020 - val_loss: 1.3283
Epoch 14/100
6/6  0s 57ms/step - loss: 0.0017 - val_loss: 1.2668
Epoch 15/100
6/6  0s 35ms/step - loss: 0.0012 - val_loss: 1.2611
Epoch 16/100
6/6  0s 36ms/step - loss: 0.0010 - val_loss: 1.2590
Epoch 17/100
6/6  0s 39ms/step - loss: 0.0010 - val_loss: 1.2581
Epoch 18/100
6/6  0s 37ms/step - loss: 9.2884e-04 - val_loss: 1.2576
Epoch 19/100
6/6  0s 40ms/step - loss: 9.6955e-04 - val_loss: 1.2571
Epoch 20/100
6/6  0s 36ms/step - loss: 9.7893e-04 - val_loss: 1.2569
Epoch 21/100
6/6  0s 37ms/step - loss: 9.2560e-04 - val_loss: 1.2568
Epoch 22/100
6/6  0s 34ms/step - loss: 9.0873e-04 - val_loss: 1.2569
Epoch 23/100
6/6  0s 53ms/step - loss: 9.1330e-04 - val_loss: 1.2568
Epoch 24/100
6/6  0s 35ms/step - loss: 9.2316e-04 - val_loss: 1.2567
Epoch 25/100
6/6  0s 32ms/step - loss: 9.7719e-04 - val_loss: 1.2567
Epoch 26/100
6/6  0s 35ms/step - loss: 8.9853e-04 - val_loss: 1.2568
Epoch 27/100
6/6  0s 35ms/step - loss: 8.8854e-04 - val_loss: 1.2567
Epoch 28/100
6/6  0s 43ms/step - loss: 8.9609e-04 - val_loss: 1.2567
Epoch 29/100
6/6  0s 35ms/step - loss: 8.8722e-04 - val_loss: 1.2568
Epoch 30/100
6/6  0s 35ms/step - loss: 8.7152e-04 - val_loss: 1.2568

Epoch 31/100
 6/6 ██████████ 0s 35ms/step - loss: 9.0248e-04 - val_loss: 1.2567
 Epoch 32/100
 6/6 ██████████ 0s 35ms/step - loss: 9.1162e-04 - val_loss: 1.2567
 Epoch 33/100
 6/6 ██████████ 0s 44ms/step - loss: 8.3957e-04 - val_loss: 1.2568
 Epoch 34/100
 6/6 ██████████ 0s 37ms/step - loss: 8.6062e-04 - val_loss: 1.2568
 Epoch 35/100
 6/6 ██████████ 0s 34ms/step - loss: 9.0151e-04 - val_loss: 1.2567
 Epoch 36/100
 6/6 ██████████ 0s 34ms/step - loss: 8.4756e-04 - val_loss: 1.2568
 Epoch 37/100
 6/6 ██████████ 0s 35ms/step - loss: 8.1085e-04 - val_loss: 1.2568
 32/32 ██████████ 0s 4ms/step

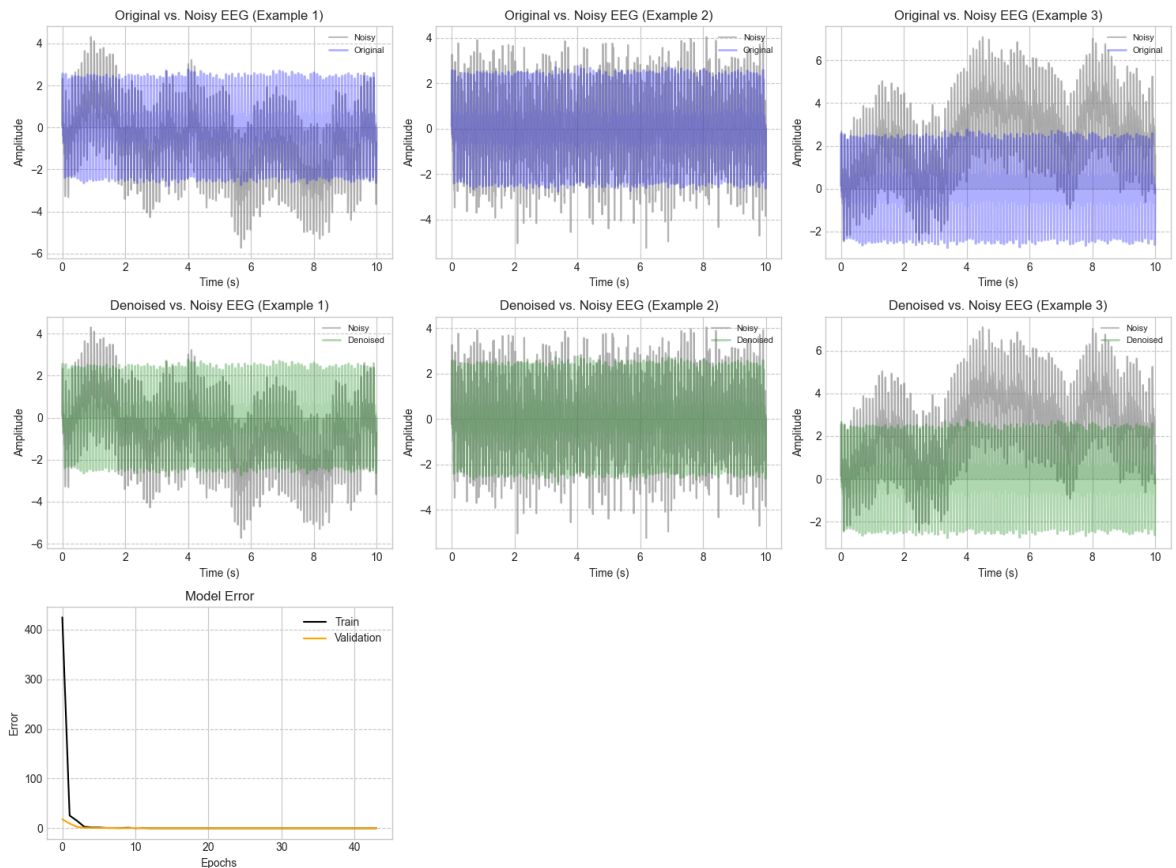
Comparison of Original, Noisy, and Denoised ECG Signals & Model Error


























Epoch 1/100
6/6  2s 74ms/step - loss: 685.8409 - val_loss: 17.9598
Epoch 2/100
6/6  0s 39ms/step - loss: 26.9819 - val_loss: 9.4819
Epoch 3/100
6/6  0s 41ms/step - loss: 16.0336 - val_loss: 2.6408
Epoch 4/100
6/6  0s 38ms/step - loss: 3.0089 - val_loss: 0.8843
Epoch 5/100
6/6  0s 42ms/step - loss: 2.0784 - val_loss: 0.6186
Epoch 6/100
6/6  0s 40ms/step - loss: 1.2711 - val_loss: 0.3699
Epoch 7/100
6/6  0s 46ms/step - loss: 0.4529 - val_loss: 0.2616
Epoch 8/100
6/6  0s 41ms/step - loss: 0.3848 - val_loss: 0.2317
Epoch 9/100
6/6  0s 41ms/step - loss: 0.3252 - val_loss: 0.0311
Epoch 10/100
6/6  0s 39ms/step - loss: 0.3686 - val_loss: 0.0133
Epoch 11/100
6/6  0s 38ms/step - loss: 0.0525 - val_loss: 0.5323
Epoch 12/100
6/6  0s 38ms/step - loss: 0.7526 - val_loss: 0.0077
Epoch 13/100
6/6  0s 39ms/step - loss: 0.0086 - val_loss: 0.0027
Epoch 14/100
6/6  0s 40ms/step - loss: 0.0045 - val_loss: 0.0018
Epoch 15/100
6/6  0s 39ms/step - loss: 0.0039 - val_loss: 5.4709e-04
Epoch 16/100
6/6  0s 42ms/step - loss: 0.0164 - val_loss: 5.4151e-04
Epoch 17/100
6/6  0s 62ms/step - loss: 0.0025 - val_loss: 3.5429e-04
Epoch 18/100
6/6  0s 40ms/step - loss: 0.0022 - val_loss: 1.2946e-04
Epoch 19/100
6/6  0s 38ms/step - loss: 0.0023 - val_loss: 6.5058e-05
Epoch 20/100
6/6  0s 39ms/step - loss: 0.0021 - val_loss: 5.5797e-05
Epoch 21/100
6/6  0s 45ms/step - loss: 0.0020 - val_loss: 3.9620e-05
Epoch 22/100
6/6  0s 39ms/step - loss: 0.0020 - val_loss: 1.6767e-05
Epoch 23/100
6/6  0s 39ms/step - loss: 0.0021 - val_loss: 1.0588e-05
Epoch 24/100
6/6  0s 44ms/step - loss: 0.0020 - val_loss: 1.0989e-05
Epoch 25/100
6/6  0s 40ms/step - loss: 0.0020 - val_loss: 1.2260e-05
Epoch 26/100
6/6  0s 56ms/step - loss: 0.0020 - val_loss: 7.0832e-06
Epoch 27/100
6/6  0s 39ms/step - loss: 0.0019 - val_loss: 1.0622e-05
Epoch 28/100
6/6  0s 39ms/step - loss: 0.0019 - val_loss: 3.0790e-05
Epoch 29/100
6/6  0s 39ms/step - loss: 0.0019 - val_loss: 7.7882e-06
Epoch 30/100
6/6  0s 39ms/step - loss: 0.0020 - val_loss: 3.8943e-06

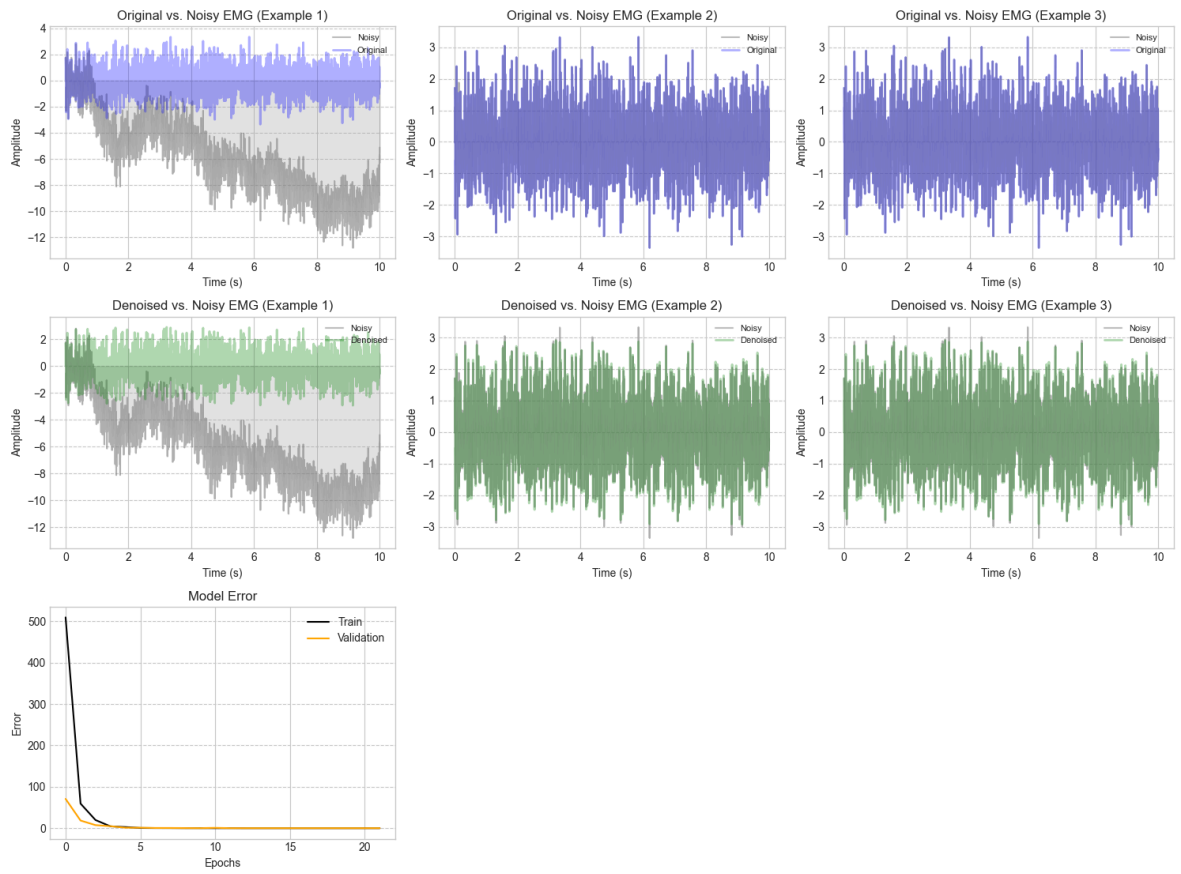
Epoch 31/100
6/6 ██████████ 0s 40ms/step - loss: 0.0018 - val_loss: 1.6540e-05
Epoch 32/100
6/6 ██████████ 0s 43ms/step - loss: 0.0019 - val_loss: 1.4527e-05
Epoch 33/100
6/6 ██████████ 0s 40ms/step - loss: 0.0019 - val_loss: 1.1196e-05
Epoch 34/100
6/6 ██████████ 0s 40ms/step - loss: 0.0018 - val_loss: 3.0858e-06
Epoch 35/100
6/6 ██████████ 0s 54ms/step - loss: 0.0018 - val_loss: 5.3707e-06
Epoch 36/100
6/6 ██████████ 0s 40ms/step - loss: 0.0018 - val_loss: 5.1938e-06
Epoch 37/100
6/6 ██████████ 0s 39ms/step - loss: 0.0018 - val_loss: 1.3806e-05
Epoch 38/100
6/6 ██████████ 0s 38ms/step - loss: 0.0018 - val_loss: 3.2163e-06
Epoch 39/100
6/6 ██████████ 0s 38ms/step - loss: 0.0019 - val_loss: 3.1363e-06
Epoch 40/100
6/6 ██████████ 0s 42ms/step - loss: 0.0019 - val_loss: 3.0681e-05
Epoch 41/100
6/6 ██████████ 0s 43ms/step - loss: 0.0018 - val_loss: 1.5027e-05
Epoch 42/100
6/6 ██████████ 0s 40ms/step - loss: 0.0019 - val_loss: 3.6436e-06
Epoch 43/100
6/6 ██████████ 0s 39ms/step - loss: 0.0018 - val_loss: 8.6759e-06
Epoch 44/100
6/6 ██████████ 0s 57ms/step - loss: 0.0018 - val_loss: 3.4999e-06
32/32 ██████████ 0s 4ms/step































Comparison of Original, Noisy, and Denoised EEG Signals & Model Error



Epoch 1/100
6/6  2s 78ms/step - loss: 772.2092 - val_loss: 70.8118
Epoch 2/100
6/6  0s 78ms/step - loss: 83.2332 - val_loss: 18.5554
Epoch 3/100
6/6  0s 51ms/step - loss: 16.0611 - val_loss: 7.4550
Epoch 4/100
6/6  0s 44ms/step - loss: 4.1333 - val_loss: 4.8868
Epoch 5/100
6/6  0s 44ms/step - loss: 2.4021 - val_loss: 1.1809
Epoch 6/100
6/6  0s 38ms/step - loss: 0.6207 - val_loss: 1.1634
Epoch 7/100
6/6  0s 38ms/step - loss: 0.3396 - val_loss: 0.5388
Epoch 8/100
6/6  0s 38ms/step - loss: 0.1953 - val_loss: 0.5093
Epoch 9/100
6/6  0s 38ms/step - loss: 0.0835 - val_loss: 0.3190
Epoch 10/100
6/6  0s 40ms/step - loss: 0.2216 - val_loss: 0.1410
Epoch 11/100
6/6  0s 38ms/step - loss: 0.0088 - val_loss: 0.8434
Epoch 12/100
6/6  0s 46ms/step - loss: 0.2270 - val_loss: 0.1249
Epoch 13/100
6/6  0s 38ms/step - loss: 0.0037 - val_loss: 0.1274
Epoch 14/100
6/6  0s 40ms/step - loss: 0.0028 - val_loss: 0.1282
Epoch 15/100
6/6  0s 38ms/step - loss: 0.0020 - val_loss: 0.1288
Epoch 16/100
6/6  0s 37ms/step - loss: 0.0018 - val_loss: 0.1294
Epoch 17/100
6/6  0s 37ms/step - loss: 0.0016 - val_loss: 0.1297
Epoch 18/100
6/6  0s 38ms/step - loss: 0.0015 - val_loss: 0.1299
Epoch 19/100
6/6  0s 38ms/step - loss: 0.0015 - val_loss: 0.1300
Epoch 20/100
6/6  0s 54ms/step - loss: 0.0014 - val_loss: 0.1300
Epoch 21/100
6/6  0s 37ms/step - loss: 0.0014 - val_loss: 0.1301
Epoch 22/100
6/6  0s 36ms/step - loss: 0.0015 - val_loss: 0.1301
32/32  0s 5ms/step

Comparison of Original, Noisy, and Denoised EMG Signals & Model Error



Epoch 1/100
6/6  2s 71ms/step - loss: 573.3018 - val_loss: 74.4102
Epoch 2/100
6/6  0s 40ms/step - loss: 57.4258 - val_loss: 15.3606
Epoch 3/100
6/6  0s 38ms/step - loss: 6.7631 - val_loss: 5.3075
Epoch 4/100
6/6  0s 36ms/step - loss: 5.2462 - val_loss: 2.6910
Epoch 5/100
6/6  0s 37ms/step - loss: 1.2287 - val_loss: 0.9434
Epoch 6/100
6/6  0s 36ms/step - loss: 0.3373 - val_loss: 0.5983
Epoch 7/100
6/6  0s 36ms/step - loss: 0.4096 - val_loss: 0.1848
Epoch 8/100
6/6  0s 35ms/step - loss: 0.0474 - val_loss: 0.2290
Epoch 9/100
6/6  0s 46ms/step - loss: 0.2496 - val_loss: 0.1025
Epoch 10/100
6/6  0s 38ms/step - loss: 0.0348 - val_loss: 0.0472
Epoch 11/100
6/6  0s 39ms/step - loss: 0.0524 - val_loss: 0.0524
Epoch 12/100
6/6  0s 42ms/step - loss: 0.0050 - val_loss: 0.1195
Epoch 13/100
6/6  0s 36ms/step - loss: 0.0573 - val_loss: 0.0469
Epoch 14/100
6/6  0s 37ms/step - loss: 0.0020 - val_loss: 0.0482
Epoch 15/100
6/6  0s 36ms/step - loss: 0.0016 - val_loss: 0.0636
Epoch 16/100
6/6  0s 37ms/step - loss: 0.0020 - val_loss: 0.0496
Epoch 17/100
6/6  0s 38ms/step - loss: 0.0059 - val_loss: 0.0432
Epoch 18/100
6/6  0s 36ms/step - loss: 0.0012 - val_loss: 0.0422
Epoch 19/100
6/6  0s 49ms/step - loss: 0.0012 - val_loss: 0.0985
Epoch 20/100
6/6  0s 37ms/step - loss: 0.0159 - val_loss: 0.0415
Epoch 21/100
6/6  0s 39ms/step - loss: 0.0011 - val_loss: 0.0900
Epoch 22/100
6/6  0s 38ms/step - loss: 0.0604 - val_loss: 0.0413
Epoch 23/100
6/6  0s 36ms/step - loss: 0.0013 - val_loss: 0.6500
Epoch 24/100
6/6  0s 36ms/step - loss: 0.3137 - val_loss: 0.0412
Epoch 25/100
6/6  0s 36ms/step - loss: 0.0057 - val_loss: 2.0062
Epoch 26/100
6/6  0s 35ms/step - loss: 1.0233 - val_loss: 0.2979
Epoch 27/100
6/6  0s 36ms/step - loss: 0.3517 - val_loss: 0.0411
Epoch 28/100
6/6  0s 36ms/step - loss: 0.0011 - val_loss: 0.0411
Epoch 29/100
6/6  0s 53ms/step - loss: 0.0011 - val_loss: 0.0410
Epoch 30/100
6/6  0s 41ms/step - loss: 0.0011 - val_loss: 0.0410

Epoch 31/100
6/6 ————— 0s 35ms/step - loss: 0.0010 - val_loss: 0.0410
Epoch 32/100
6/6 ————— 0s 37ms/step - loss: 0.0011 - val_loss: 0.0410
Epoch 33/100
6/6 ————— 0s 36ms/step - loss: 0.0010 - val_loss: 0.0410
Epoch 34/100
6/6 ————— 0s 37ms/step - loss: 0.0011 - val_loss: 0.0410
Epoch 35/100
6/6 ————— 0s 35ms/step - loss: 0.0010 - val_loss: 0.0410
Epoch 36/100
6/6 ————— 0s 37ms/step - loss: 0.0011 - val_loss: 0.0410
Epoch 37/100
6/6 ————— 0s 36ms/step - loss: 0.0011 - val_loss: 0.0410
Epoch 38/100
6/6 ————— 0s 45ms/step - loss: 0.0011 - val_loss: 0.0410
Epoch 39/100
6/6 ————— 0s 36ms/step - loss: 0.0010 - val_loss: 0.0410
Epoch 40/100
6/6 ————— 0s 39ms/step - loss: 0.0010 - val_loss: 0.0410
Epoch 41/100
6/6 ————— 0s 38ms/step - loss: 0.0010 - val_loss: 0.0410
Epoch 42/100
6/6 ————— 0s 37ms/step - loss: 9.7657e-04 - val_loss: 0.0410
Epoch 43/100
6/6 ————— 0s 38ms/step - loss: 0.0010 - val_loss: 0.0410
Epoch 44/100
6/6 ————— 0s 36ms/step - loss: 0.0010 - val_loss: 0.0410
32/32 ————— 0s 4ms/step

Comparison of Original, Noisy, and Denoised EOG Signals & Model Error

