EMORY

An RL Framework for ECG Baseline Wander Removal

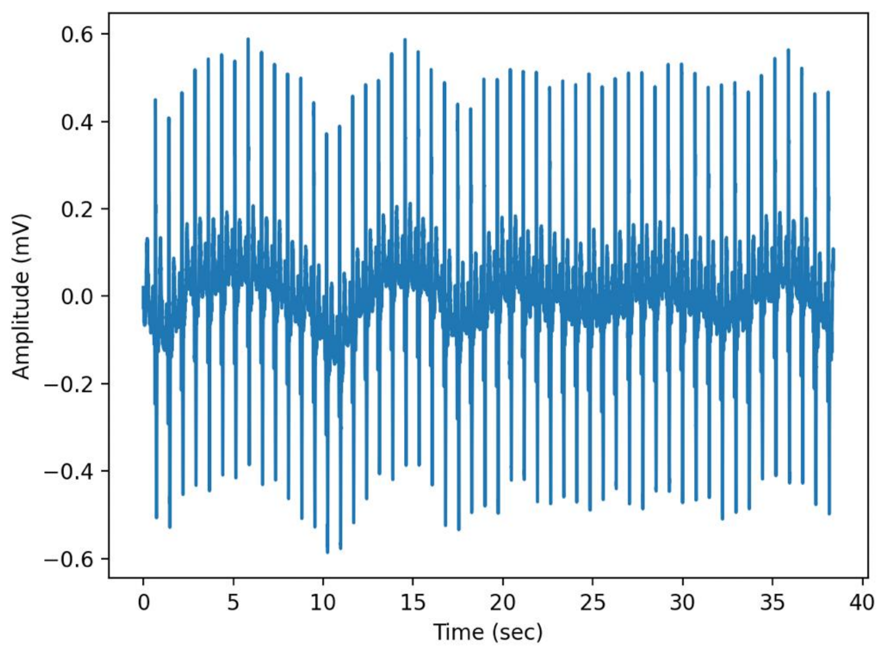
Masoud Nateghi

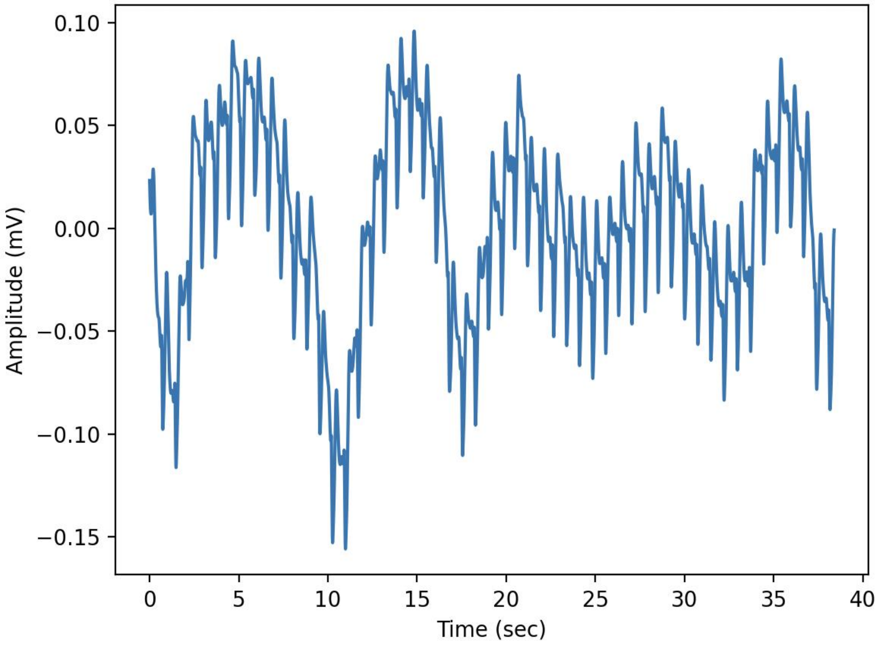
Background

- ECG is a vital diagnostic tool for cardiac health
- Signal quality is often compromised by various noise types:
  - Power line interference
  - Muscle artifacts
  - Baseline wander
  - Motion artifacts
- Clean signals are crucial for accurate diagnosis and interpretation
- Effective denoising is essential for both clinical and remote monitoring applications

The Challenge of Baseline Wander

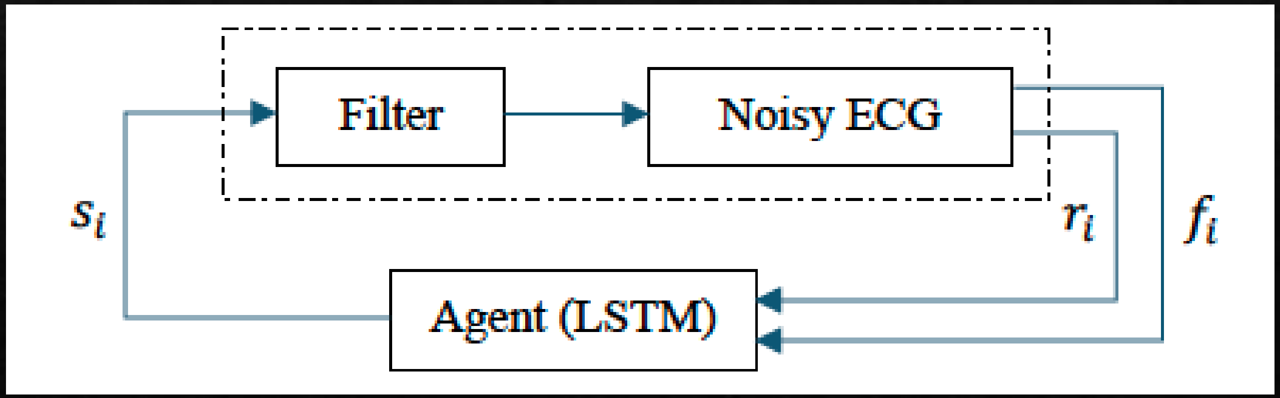
- Major source of ECG signal contamination
- Primary causes:
  - Patient respiration
  - Body movements
  - Poor electrode contact
- Technical challenges:
  - Very low frequency noise (<0.5 Hz)
  - Overlaps with important ECG components
  - Conventional filters can distort critical features
  - ST segment





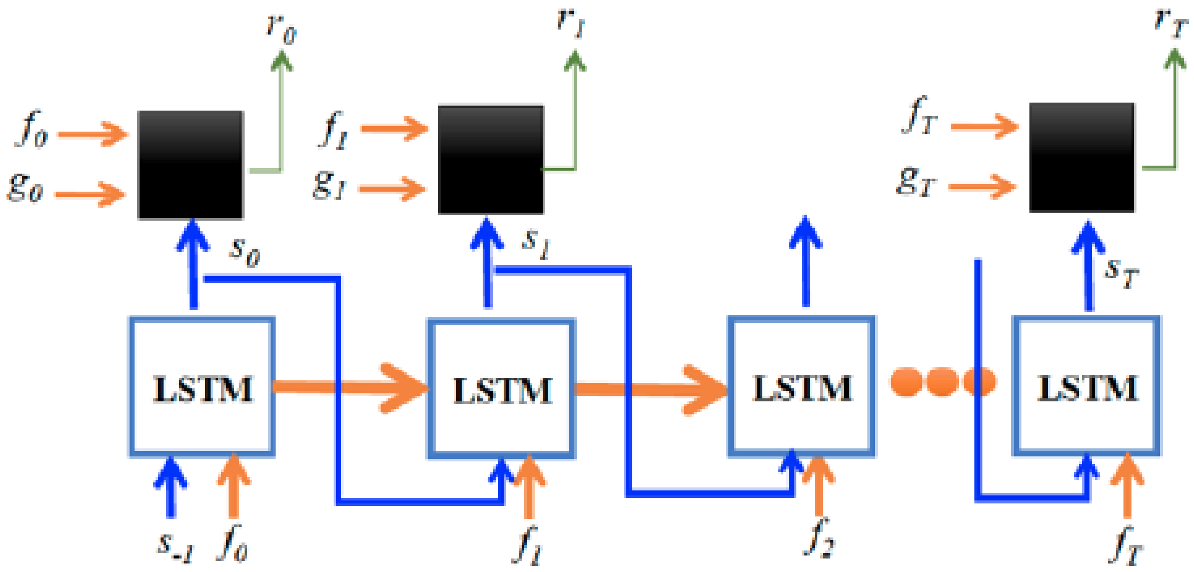
Problem Formulation

- Environment: Low Pass Filter + ECG signal segment
- States: Raw ECG measurements (s_i)
- Actions: Generate optimal filter parameters (s_i)
- Reward: Negative MSE between clean and denoised signal (r_i)



Proposed Solution

- RL-based adaptive filtering approach
- Key features:
  - Dynamic parameter adjustment
  - Real-time processing capability
  - Preservation of critical ECG feature
- Implementation:
  - Model-free
  - PPO algorithm
  - LSTM-based agent architecture



Dataset

- PTB Diagnostic ECG Database
  - 549 records from 290 subjects
  - 15 leads per record (12 conventional + 3 Frank leads)
  - High resolution (1000 samples per second)
  - Includes various cardiac conditions
  - Each record 30+ seconds in length
  - Digitized at 1000 Hz

- MIT-BIH Noise Stress Test Database (NSTDB)
- Provides realistic noise recordings
- Three noise types:
  - Baseline wander (BW)
  - Muscle artifact (MA)
  - Electrode motion artifact (EM)
- Sampled at 360 Hz

Data Preparation

- PTB Dataset
  - Apply an aggressive 1 Hz HPF
- MIT-BIH
  - Get a random segment of BW noise
  - Same length as the ECG
- Add it to ECG with SNR = 10 dB

$$\alpha = \sqrt{\frac{1}{10^{SNR/10}} \frac{P_{signal}}{P_{noise}}}$$

Training Hyperparameters

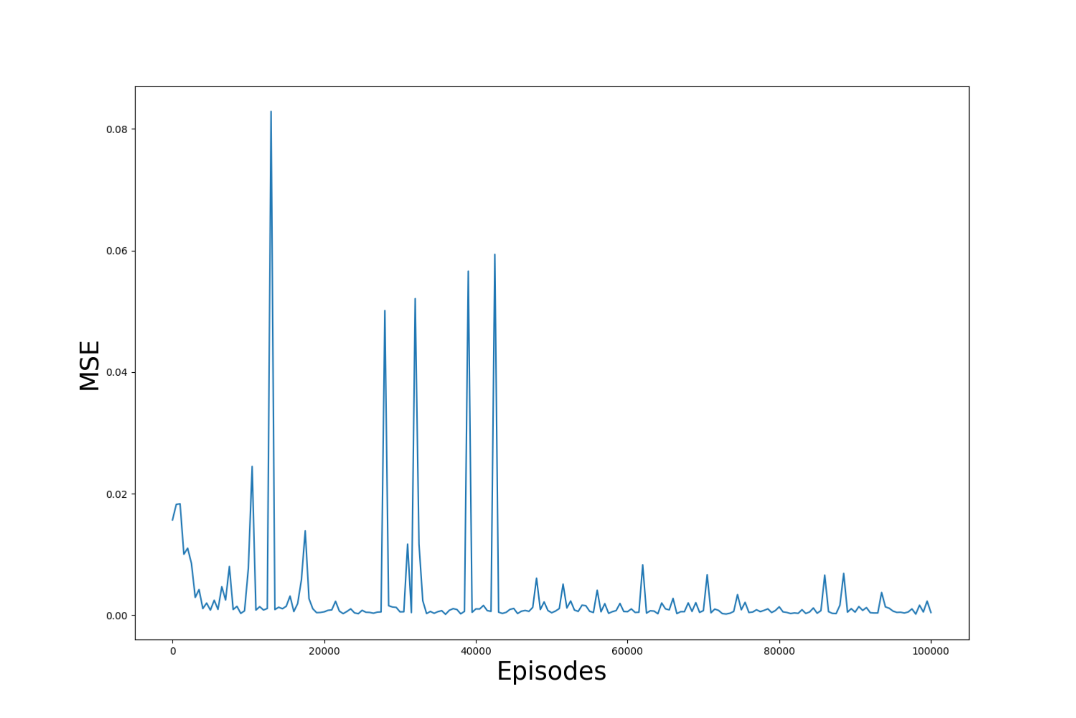
```
# LSTM network
self.lstm = nn.LSTM(
    input_size=self.segment_size + 1,  # signal + previous action
    hidden_size=64,
    batch_first=True
)

# Action mean and log std
self.action_mean = nn.Sequential(
    nn.Linear( in_features: 64,  out_features: 32),
    nn.ReLU(),
    nn.Linear( in_features: 32,  out_features: 1),
    nn.Sigmoid()
)
```

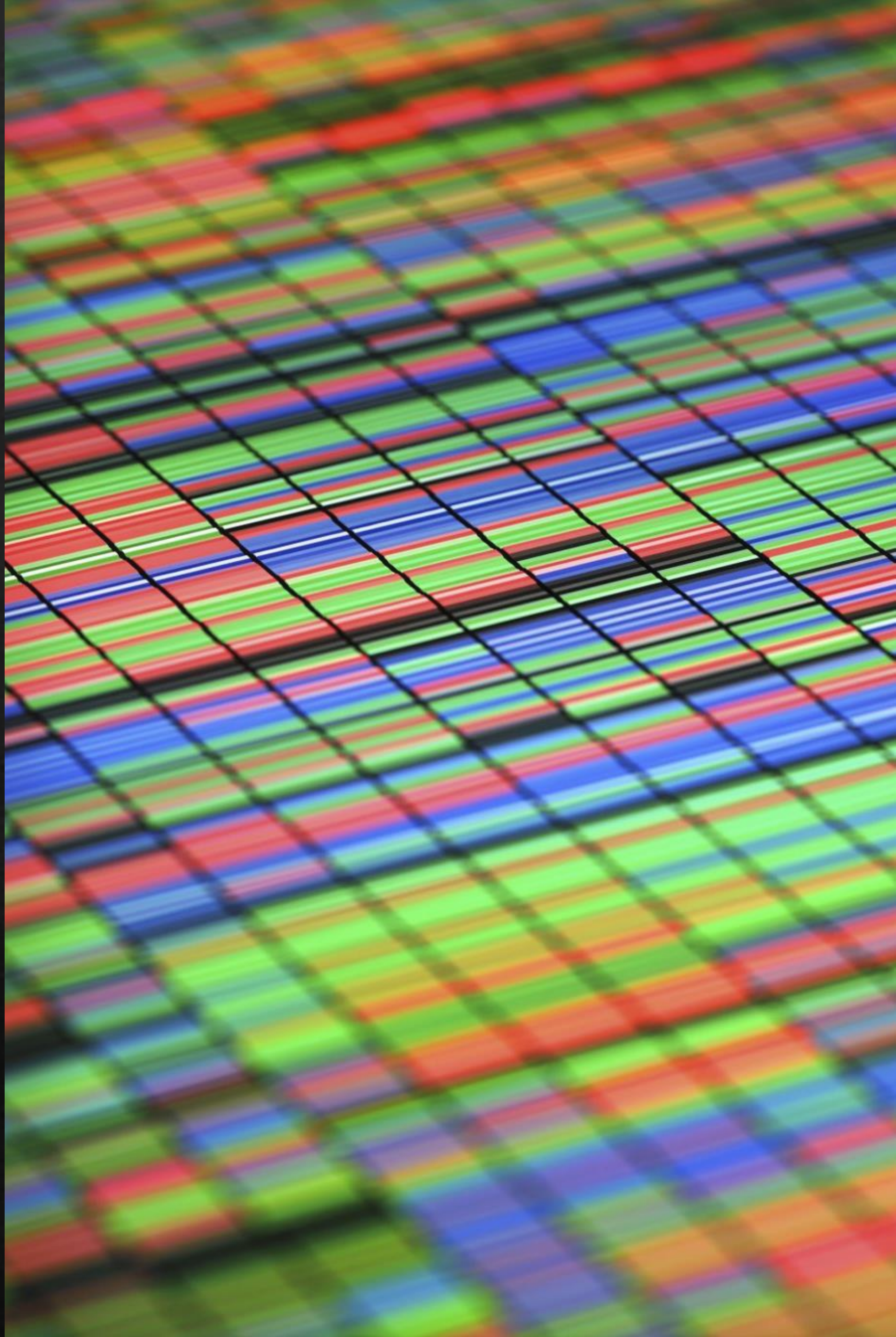
```
model, history = train_model(
    channel_data=data[channel],
    train_indices=list(np.arange(train_size)),
    baseline_wander=bw,
    window_seconds=5,
    fs=fs,
    snr_db=10,
    total_timesteps=100000,
    eval_freq=500
)
```

```
# Create and train model
model = PPO(
    CustomLSTMPolicy,
    env,
    verbose=1,
    learning_rate=3e-4,
    n_steps=2048,
    batch_size=64,
    n_epochs=10,
    gamma=0.99,
    gae_lambda=0.95,
    clip_range=0.2
)
```

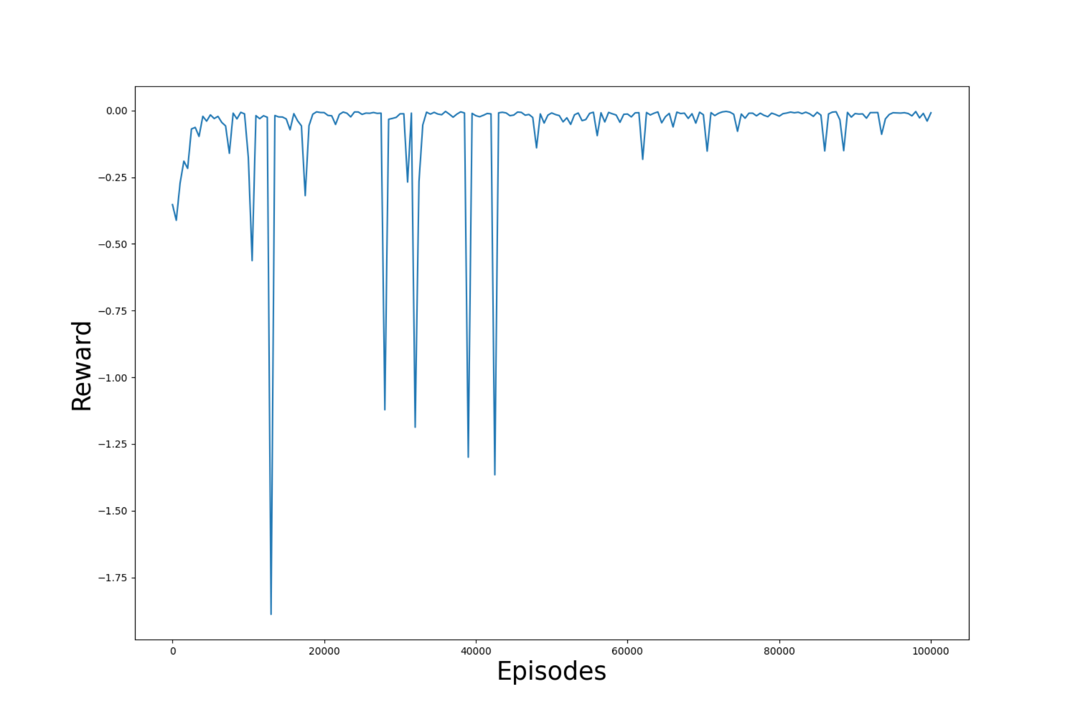
Results – MSE



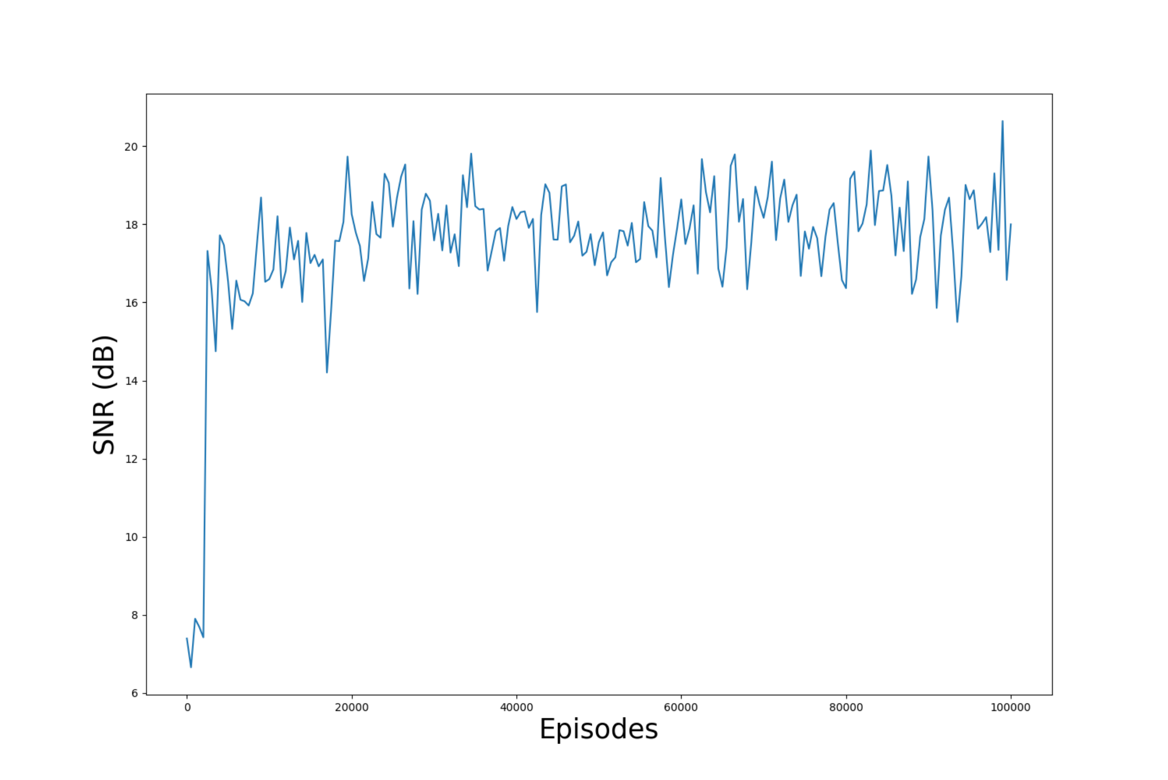
0.08
0.06
0.04
0.02
0.00
MSE
0
20000
40000
60000
80000
100000
Episodes



Results - Reward



Results - SNR



SNR (dB)
Episodes
20
18
16
14
12
10
8
6
0
20000
40000
60000
80000
100000