OUTPUT 1: MACHINE LEARNING PART

File name:main_detailed.py

```
PS C:\Users\hp\OneDrive\Desktop\EEG_Project_CNN> python main_detailed.py
Processing data and extracting features...
Processing files: 100%|████████████████████████████████████████| 72/72 [00:08<00:00,  8.32it/s]

Dataset statistics:
Total samples: 72
Features per sample: 320
Class distribution: [36 36]
Fold 1 - SVM: 1.0000
Fold 1 - KNN: 1.0000
Fold 1 - RF: 1.0000
Fold 1 - AdaBoost: 1.0000
Fold 2 - SVM: 0.8750
Fold 2 - KNN: 0.7500
Fold 2 - RF: 0.7500
Fold 2 - AdaBoost: 0.8750
Fold 3 - SVM: 1.0000
Fold 3 - KNN: 1.0000
Fold 3 - RF: 1.0000
Fold 3 - AdaBoost: 1.0000
Fold 4 - SVM: 0.8571
Fold 4 - KNN: 1.0000
Fold 4 - RF: 1.0000
Fold 4 - AdaBoost: 1.0000
Fold 5 - SVM: 0.8571
Fold 5 - KNN: 1.0000
Fold 5 - RF: 0.8571
Fold 5 - AdaBoost: 0.5714
Fold 6 - SVM: 1.0000
Fold 6 - KNN: 1.0000
Fold 6 - RF: 1.0000
Fold 6 - AdaBoost: 1.0000
Fold 7 - SVM: 1.0000
Fold 7 - KNN: 1.0000
Fold 7 - RF: 1.0000
Fold 7 - AdaBoost: 1.0000
Fold 8 - SVM: 1.0000
Fold 8 - KNN: 0.8571
Fold 8 - RF: 1.0000
Fold 8 - AdaBoost: 1.0000
Fold 9 - SVM: 1.0000
Fold 9 - KNN: 1.0000
Fold 9 - RF: 0.7143
```

Fold 9 - AdaBoost: 0.7143
Fold 10 - SVM: 1.0000
Fold 10 - KNN: 1.0000
Fold 10 - RF: 0.8571
Fold 10 - AdaBoost: 0.8571

Mean accuracy ± std across 10 folds:
SVM: 0.9589 ± 0.0629
KNN: 0.9607 ± 0.0821
RF: 0.9179 ± 0.1084

File name:main.py

PS C:\Users\hp\OneDrive\Desktop\EEG_Project> python main.py
Processing data and extracting features...
Processing files: 100%|
72/72 [00:09<00:00, 7.70it/s]

Dataset statistics:
Total samples: 72
Features per sample: 320
Class distribution: [36 36]

Evaluating models with feature selection...
SVM: 95.89% (+/- 6.29%)
KNN: 96.07% (+/- 8.21%)
Random Forest: 91.79% (+/- 10.84%)
AdaBoost: 90.18% (+/- 14.31%)

OUTPUT 2:DEEP LEARNING PART

File name: deep_learning.py

python deep_learning.py
2025-12-12 15:23:50.139847: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-12-12 15:23:58.213586: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
==========
EEG CLASSIFICATION - FAST VERSION WITH PROGRESS
==========
Feature: energy
==========

[1/5] Found 72 EDF files
   Loaded 72 recordings

[2-4/5] Loading cached normalized data...
   Loaded X: (72, 200, 200, 5), y: (72,)

Data shape: (72, 200, 200, 5)
Class distribution: [36 36]

==========
TRAINING
==========

>>> Training Simple CNN first (Architecture 2)...

[5/5] Training Simple CNN...
   Using 10-fold cross-validation
   Max epochs: 100, Batch size: 16

  Fold 1/10 - Training...
WARNING:tensorflow:From C:\Users\hp\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\backend\common\global_state.py:82: The name tf.reset_default_graph is deprecated. Please use tf.compat.v1.reset_default_graph instead.

2025-12-12 15:24:10.898216: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
    Fold 1 - Epoch 31: val_acc=1.0000, time=63.4s
   → Test Accuracy: 100.00%

  Fold 2/10 - Training...
    Fold 2 - Epoch 28: val_acc=0.8750, time=59.0s
   → Test Accuracy: 87.50%

  Fold 3/10 - Training...
    Fold 3 - Epoch 37: val_acc=1.0000, time=87.2s
   → Test Accuracy: 100.00%

  Fold 4/10 - Training...

Fold 4 - Epoch 29: val_acc=1.0000, time=65.9s
→ Test Accuracy: 100.00%

Fold 5/10 - Training...
Fold 5 - Epoch 29: val_acc=1.0000, time=67.7s
→ Test Accuracy: 100.00%

Fold 6/10 - Training...
Fold 6 - Epoch 29: val_acc=1.0000, time=61.3s
→ Test Accuracy: 100.00%

Fold 7/10 - Training...
Fold 7 - Epoch 29: val_acc=0.8571, time=66.0s
→ Test Accuracy: 100.00%

Fold 8/10 - Training...
Fold 8 - Epoch 34: val_acc=1.0000, time=79.2s
→ Test Accuracy: 100.00%

Fold 9/10 - Training...
Fold 9 - Epoch 29: val_acc=1.0000, time=72.2s
→ Test Accuracy: 100.00%

Fold 10/10 - Training...
Fold 10 - Epoch 31: val_acc=1.0000, time=65.8s
→ Test Accuracy: 100.00%

==========
Simple CNN: 98.75% ± 3.75%
==========

>>> Train ASPP model? (slower, ~3-5 min per fold)
Press Enter to train, or Ctrl+C to skip...

[5/5] Training ASPP CNN...
Using 10-fold cross-validation
Max epochs: 100, Batch size: 16

Fold 1/10 - Training...
Fold 1 - Epoch 85: val_acc=1.0000, time=344.0s

→ Test Accuracy: 100.00%

Fold 2/10 - Training...
  Fold 2 - Epoch 28: val_acc=0.8750, time=123.5s
→ Test Accuracy: 50.00%

Fold 3/10 - Training...
  Fold 3 - Epoch 100: val_acc=1.0000, time=455.8s
→ Test Accuracy: 100.00%

Fold 4/10 - Training...
  Fold 4 - Epoch 99: val_acc=1.0000, time=509.9s
→ Test Accuracy: 100.00%

Fold 5/10 - Training...
  Fold 5 - Epoch 100: val_acc=1.0000, time=411.2s
→ Test Accuracy: 100.00%

Fold 6/10 - Training...
  Fold 6 - Epoch 100: val_acc=1.0000, time=392.1s
→ Test Accuracy: 100.00%

Fold 7/10 - Training...
  Fold 7 - Epoch 100: val_acc=1.0000, time=404.4s
→ Test Accuracy: 100.00%

Fold 8/10 - Training...
  Fold 8 - Epoch 100: val_acc=1.0000, time=412.1s
→ Test Accuracy: 100.00%

Fold 9/10 - Training...
  Fold 9 - Epoch 100: val_acc=1.0000, time=406.2s
→ Test Accuracy: 100.00%

Fold 10/10 - Training...
  Fold 10 - Epoch 100: val_acc=1.0000, time=421.3s
→ Test Accuracy: 100.00%

  ==

ASPP CNN: 95.00% ± 15.00%
==========

==========
FINAL RESULTS
==========
Architecture 1 (ASPP): 95.00% ± 15.00%
Architecture 2 (Simple): 98.75% ± 3.75%
==========