Note: N is the number of data points. These are not all the architectures we tested but are rather ones with presentable results.

Model	Training Parameters	Architecture
CNN Model 1	Xavier Uniform Initialization Number Epochs: 1000 K-Folds: 4 Splits Batch Size: 500 Adam Optimizer Learning Rate: 0.005 Criterion (Loss Function): Cross Entropy Loss	Input: N * 1 * 22 * 1000 Convolutional Layer: stride: (1,3), kernel: (1, 50), filter number: 50 Batch Normalization: momentum 0.1 Exponential Linear Units (ELU) with activation map N*22*317*50 Convolutional Layer: stride: (1, 1), kernel: (22, 1), filter number: 50 Batch Normalization: momentum 0.1 Exponential Linear Units (ELU) with activation map N*1*317*50 Average Pooling: stride: (1, 15), kernel: (1, 75) Dropout (p = 0.3) Fully Connected Layer: (50 * 17, 4) Softmax Output: N * 4
CNN Model 2	Xavier Uniform Initialization Number Epochs: 1000 K-Folds: 4 Splits Batch Size: 500 Adam Optimizer Learning Rate: 0.005 Criterion (Loss Function): Cross Entropy Loss	Input: N * 1 * 22 * 1000 Convolutional Layer: stride: (1, 2), kernel: (1, 40), filter number: 40 Batch Normalization: momentum 0.1 Exponential Linear Units (ELU) with activation map N*22*481*40 Convolutional Layer: stride: (1, 1), kernel: (22, 1), filter number: 40 Batch Normalization: momentum 0.1 Exponential Linear Units (ELU) with activation map N*1*481*40 Average Pooling: stride: (1, 15), kernel: (1, 75) Dropout (p = 0.3) Fully Connected Layer: (40 * 28, 4) Softmax Output: N * 4
CNN Model 3	Xavier Uniform Initialization Number Epochs: 1000 K-Folds: 4 Splits Batch Size: 500 Adam Optimizer Learning Rate: 0.005 Criterion (Loss Function): Cross Entropy Loss	Input: N * 1 * 22 * 1000 Convolutional Layer: stride: (1, 1), kernel: (22, 1), filter number: 40 Batch Normalization: momentum 0.1 Exponential Linear Units (ELU) with activation map N*22*481*40 Convolutional Layer: stride: (1, 2), kernel: (1, 40), filter number: 40 Batch Normalization: momentum 0.1 Exponential Linear Units (ELU) with activation map N*1*481*40 Average Pooling: stride: (1, 15), kernel: (1, 75) Dropout (p = 0.3) Fully Connected Layer: (40 * 28, 4) Softmax; Output: N * 4
CNN Model 4	Same as CNN Model 2 but Number Epochs is 50 for each model, which has 9 models for each subject	Same as CNN model 2
CNN Model 5	Same as CNN Model 2	Same as CNN model 2, but Fully Connected Layer becomes (40 * 28, 9), and the softmax outputs (N * 9)
RNN Model 1	Number Epochs: 200 K-Folds: 5 Splits Batch Size: 50 Adam Optimizer Learning Rate: 0.001 Criterion (Loss Function): Categorical Cross Entropy Loss	Input: N * 1 * 22 * 1000 Bidirectional LSTM Layer: a visible layer with (22, 1000) input, and a hidden layer with 128 LSTM neurons, with return sequence, 0.01 l2 kernel, bias, and recurrent regularizer. Batch Normalization: momentum 0.99 Dropout (p = 0.5) Flatten layer Dense with ELU Activation layer(32) Dropout (p = 0.5) Output: N * 4, using Adam optimizer with 0.002 kernel regularizer
RNN Model 2	Same as RNN Model 1	Same as RNN Model 1, except for the input with size (N * 1 * 199 * 220)
RNN Model 3	Number Epochs: 200 K-Folds: 5 Splits Batch Size: 50 Adam Optimizer Learning Rate: 0.001 Criterion (Loss Function): Categorical Cross Entropy Loss	Input: N * 1 * 22 * 1000 Bidirectional LSTM Layer: a visible layer with (22, 1000) input, and a hidden layer with 128 LSTM neurons, with return sequence, 0.01 l2 kernel, bias, and recurrent regularizer. Batch Normalization: momentum 0.99 Dropout (p = 0.5) Bidirectional LSTM Layer: a visible layer with (22, 1000) input, and a hidden layer with 64 LSTM neurons, with return sequence, 0.01 l2 kernel, bias, and recurrent regularizer. Flatten layer Dense with ELU Activation layer(32) Dropout (p = 0.5) Output: N * 4, using Adam optimizer with 0.002 kernel regularizer
RNN Model 4	Same as RNN Model 3	Same as RNN Model 3, except for the input with size (N * 1 * 199 * 220)
CRNN Model 1	Number Epochs: 200	Part 1: Input: N * 1 * 199 * 220

	K-Folds: 5 Splits Batch Size: 50 Adam Optimizer Learning Rate: 0.005 Criterion (Loss Function): Categorical Cross Entropy Loss	Convolutional Layer: stride: (1, 1), kernel: (1, 10), filter number: 16 Exponential Linear Units (ELU) with activation map N*22*211*16 Batch Normalization: momentum 0.99 Convolutional Layer: stride: (1, 1), kernel: (21, 1), filter number: 32 Exponential Linear Units (ELU) with activation map N*1*211*16 Batch Normalization: momentum 0.99 MaxPool: (1, 4) Convolutional Layer: stride: (1, 1), kernel: (1, 10), filter number: 64 Exponential Linear Units (ELU) with activation map N*10*202*64 Batch Normalization: momentum 0.99 MaxPool: (1, 4) ——— Part 2: Dropout (p = 0.4) Permute(2, 3, 1) TimeDistributed Flatten Layer Bidirectional LSTM Layer: 128 LSTM neurons with return sequence Bidirectional LSTM Layer: 64 LSTM neurons with return sequence Bidirectional LSTM Layer: 32 LSTM neurons Dropout (p = 0.4) Output: N * 4, using Adam optimizer with 0.01 kernel regularizer
CRNN Model 2	Same as CRNN Model 1	Part 1 from CRNN Model 1 Convolutional Layer: stride: (1, 1), kernel: (1, 10), filter number: 128 Exponential Linear Units (ELU) Batch Normalization: momentum 0.99 MaxPool: (1, 4) Part 2 from CRNN Model 1
CRNN Model 3	Same as CRNN Model 1	Part 1 from CRNN Model 1 Convolutional Layer: stride: (1, 1), kernel: (1, 10), filter number: 256 Exponential Linear Units (ELU) with activation map N*1*193*256 Batch Normalization: momentum 0.99 Part 2 from CRNN Model 1, but add 'Bidirectional LSTM Layer: 128 LSTM neurons with return sequence' after 'TimeDistributed Flatten Layer'

Models	Train/Test Accuracy (%)	Final Training Loss
CNN Model 1	1.0/.62	.77
CNN Model 2	1.0/.64	.79
CNN Model 3	1.0/.69	.77
CNN Model 4.1	1.0/.44 (Average)	.75
CNN Model 4.2	1.0/.63 (Average)	.78
CNN Model 5	1.0/.99	1.37
RNN Model 1	.91/.32	2.16
RNN Model 2	.84/.40	1.05
RNN Model 3	.96/.30	1.64
RNN Model 4	.96/.42	1.05
CRNN Model 1	1.0/.57	.08
CRNN Model 2	1.0/.63	.14
CRNN Model 3	.97/.41	.24

For CNN Model 4, all of the various test accuracies for each individual subject network can be found in the report.