Examining the Performance of Training on EEG Data with CNN Model vs. CRNN Models

Lauren Mizner
UCLA - ECE C247
UCID# 005225768
lmizner@g.ucla.edu

Abstract

The purpose of this study is to build and optimize a Convolutional Neural Network model and multiple Convolutional Recurrent Neural Network models on electroencephalography (EEG) data. These models are further evaluated using additional studies to investigate how their test accuracies may differ under different dataset conditions. These conditions look at the differences in performance based on data size, specifically evaluating the classification accuracy for 1 test subject versus the classification accuracy across all test subjects for one study and evaluating the difference in classification accuracy due to varying sizes in time intervals of the data for another study. Results of this study are compared to theoretical expectations of these models to validate effects of hyperparameters and model performance.

1. Introduction

This study utilizes electroencephalography (EEG) data consisting of 2115 trials over 9 test subjects. Each trial corresponds to the EEG data sourced from 22 electrodes over 1000 time bins. Lastly, each trial is categorized into 1 of 4 tasks implemented by the test subject at the time the EEG data is measured. The tasks implemented by the test subjects are arm movement to the left, arm movement to the right, foot movement, and tongue movement.

The purpose of this study is to implement and evaluate the testing accuracy of various neural network models on categorizing the four tasks performed by the test subjects. The different model architectures implemented in this study consist of a Convolutional Neural Network (CNN) optimized across all test subjects, CNN evaluated over various sized time bins, CNN evaluated for an individual subject, Convolutional Recurrent Neural Networks (CRNN) optimized across all test subjects, CRNNs evaluated over various sized time bins, and CRNNs evaluated for an individual subject.

2. Method

This study focuses on two main classification models, a CNN model and CRNN models, with additional studies conducted on each model for further insight. Both models have been evaluated on the full dataset, on an individual subject dataset and on data of various timestep sizes.

Preprocessing on the main models, consists of reducing the data time bins from 1000 to 500 after initial inspection showed that the data from time bin 500 to time bin 1000 proved to be noisy and less conducive to training the models. Data was also split into training and validation sets using random splitting, with a dataset of 500 being set aside for validation purposes and the remaining data being used for the purpose of training the models. Labels for the training set, validation set, and test set were converted to categorical variables, 0 through 3, for the purpose of classifying the different tasks performed by the test subjects at the time the EEG data was measured.

2.1. Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a type of neural network commonly used for deep learning tasks, primarily classification. CNNs typically consist of convolutional layers, pooling layers, and fully connected layers. The convolutional layers apply filters or kernels to the input data to extract features and patterns. The pooling layers reduce the spatial dimensions obtained from the convolutional layers, while retaining important information. Lastly, the fully connected layers combine the features learned by the convolutional layers to assist in making predictions. In the case of this study, a batch normalization layer and a dropout layer have been incorporated to assist in the optimization of the model.

Four separate convolutional layer blocks were defined for this study's CNN model. The number of filters used per convolutional layer exponentially increases, starting at 25 and ending with 200, with each layer utilizing a kernel size, or filter size, of 10 x 1. To ensure that the output features maps maintain the same spatial dimensions as the input data, a padding parameter of 'same' is used. An ELU, or Exponential Linear Unit, activation function is implemented to help the network learn complex patterns

by introducing non-linearity. Lastly, an L2 kernel regularizer of 0.015 is included to penalize large weights in the model and assist with model optimization.

For the pooling layers a 3 x 1 pooling window is utilized along with a padding parameter of 'same' for the same reason described in the convolutional layer definition.

As previously mentioned, this study incorporates a batch normalization layer and dropout layer. The batch normalization layer is included in order to normalize the activations of the previous layer to make the training process more stable and accelerate convergence. The dropout layer is set to drop a fraction of input units during training, in this case 0.55, to help prevent overfitting of the model.

Lastly, the fully connected layer is made up of a flatten layer and a dense, or fully connected layer, itself. The flatten layer converts the multi-dimensional input data into a one-dimensional array. The dense, or fully connected layer, specifies a softmax activation function in order to convert the output into a probability distribution over the different classes.

2.2. Convolutional Recurrent Neural Network (CRNN)

A Convolutional Recurrent Neural Network combines the strengths of both CNNs and RNNs and consists of convolutional layers for feature extraction, followed by recurrent layers for sequential processing of the extracted features. Two CRNN models were considered for this study, CNN+LSTM and CNN+GRU. For both CRNN models, the CNN portion of the model maintains the same features and hyperparameters defined in the previous section, with the LSTM layer or GRU layer added at the end.

A Long Short-Term Memory model is a popular variant of RNNs that addresses the vanishing gradient problem that traditional RNNs suffer from, and is one of the specific RNN models that is utilized within this study's CRNN model. Within the LSTM layer we define a dropout rate of 0.50 and a recurrent dropout rate of 0.10 to be applied to the recurrent input units of the layer. Both dropout values assist the model in mitigating overfitting during training.

For the CNN+LSTM model the learning rate parameter did need to be decreased, as initial testing showed a much steeper loss curve. To smooth the loss curve, and promote increased test accuracy, the learning rate was decreased from 0.0009 with a 0.001 decay to 0.00075 without a decay. Additionally, the number of epochs was increased from 100 to 150, as it could be seen that the model was continuing to learn and increase accuracy past the 100 epoch mark.

A Gated Recurrent Unit model is another variant of RNNs that was explored for this study. In some cases, GRU can have some advantages over LSTM, specifically in memory usage and data processing speed.

Within the GRU layer we define a dropout rate of 0.20 and a recurrent dropout rate of 0 to be applied to the recurrent input units of the layer. Similar to the CNN+LSTM model, the learning rate was decreased from 0.0009 with a 0.001 decay to 0.00075 and a 0.001 decay. Additionally, the number of epochs was increased from 100 to 150, as it could be seen that the model was continuing to learn and increase accuracy past the 100 epoch mark.

2.3. Individual Subject Analysis

An additional study conducted on both the CNN model and the CRNN models, was evaluating how the training and testing accuracies change when evaluating over a single test subject instead of all subjects. For this study additional preprocessing was conducted in order to extract all data associated with subject 0.

2.4. Time Step Analysis

Another additional study conducted on both the CNN model and the CRNN models, was evaluating how the training and testing accuracies change when evaluating over different time intervals. For this study additional preprocessing was conducted in order to create datasets corresponding to time bins 0 through 200, 0 through 400, 0 through 600, 0 through 800, and 0 through 1000. By training and testing using these datasets insight could be gained on if the accuracies increase or decrease as the time interval of the dataset increases.

3. Results

The following sections summarize the test accuracy results of each model and the additional studies conducted on those models.

3.1. Convolutional Neural Network

The highest test accuracy of 70.65% was achieved with the basic CNN model, as shown in Figure 1 of Appendix A. This was accomplished through fine tuning the hyperparameters of dropout, learning rate, total epochs, and adding an L2 regularizer.

3.2. Convolutional Neural Network + LSTM

The CNN+LSTM model obtains a slightly higher test accuracy of 72.46%, as shown in Figure 3. Theoretically, a CRNN model should produce a higher test accuracy than either a CNN or RNN as it combines the strengths of

both CNNs and RNNs, so this result can be seen to match that intuition.

3.3. Convolutional Neural Network + GRU

The CNN+GRU model obtains a slightly higher test accuracy of 73.14%, as shown in Figure 5. Again, theoretically, a CRNN model should produce a higher test accuracy than either a CNN or RNN as it combines the strengths of both CNNs and RNNs, so this result can be seen to match that intuition.

3.4. Individual Subject Analysis

For both the CNN model and the CRNNs model tested on an individual test subject, the results showed lower accuracy than that of both models tested across all subjects. From the results, the CNN model test accuracy dropped from 70.65% across all models to 31.83% when tested on subject 1. Similarly, the CNN+ LSTM model test accuracy dropped from 72.46% across all models to 37.02% when tested on subject 1. Lastly, the CNN+GRU model test accuracy dropped from 73.14% to 35.21% when tested on subject 1.

3.5. Time Step Analysis

Time step analysis was performed on the CNN, CNN+LSTM, and CNN+GRU models, of which the results can be seen in Figures 10, 11, and 12, respectively. Overall, the CNN model and the CRNN model kept even pace with each other while handling different time intervals. The CNN model did perform slightly better than both of the CRNN models on the data sets using a total of 600 time bins or more.

4. Discussion

Based on the initial testing of the models, the results match expectations. Theoretically, a CRNN model should produce a higher test accuracy than either a CNN or RNN as it combines the strengths of both CNNs and RNNs. As seen in the results, the test accuracies of the models match this intuition, with both the CNN+LSTM and CNN+GRU performing better than the basic CNN model.

A more interesting take away from the results includes the drastic drop in accuracy seen when training and testing on one test subject as opposed to training and testing across all subjects. This result makes sense as too little data makes it more difficult for the model to learn and generalize the data. More data will always help to further establish any patterns in the data and training over one test subject reduces the dataset from 2115 data points to 237 data points. To improve the performance, while continuing to utilize the data of a single test subject, one option would be to utilize data augmentation techniques to artificially increase the data set, providing more data points for the model to train on, thus possibly improving the overall test accuracy.

Also, based on the test accuracy results seen in Figures 7, 8, and 9, it would be worthwhile to incorporate a callback for early stopping as neither model gains much improvement past about 25 epochs. This would save time and resources spent on further training and validating the model, as well concluding the model prior to overfitting and allowing it to generalize better on new unseen data.

Another interesting result was the performance seen among the models trained on datasets of various time intervals. It's expected for a model to develop a higher testing accuracy when the model has more data to train on. This can be seen in the results up to a certain extent. For both the CNN model and the CRNN models there is an increase in test accuracy as we increase the time interval from 200 to 400 to 600 time bins. The improvement goes further with the CNN model, increasing accuracy up to a time interval of 800 time bin. The test accuracy can be seen to decrease for both models when using a time interval of 1000 time bins.

From this it can be confirmed that increasing the data size in the form of the time interval can increase the test accuracy, but only up to a certain point. Although more data allows the models to improve generalization, reduce overfitting, provide a more robust evaluation, and more, these benefits can only go on for so long. In this case and many more, there are diminishing returns associated with increasing the data set size. This shows that data set size can be seen as yet another parameter that can be fine tuned to optimize model performance.

References

[1] Awan, Abid Ali. "A Complete Guide to Data Augmentation." *DataCamp*, DataCamp, 23 Nov. 2022, www.datacamp.com/tutorial/complete-guide-data-augmentation#.

[2] Chowdhury, Kuldeep. "10 Hyperparameters to Keep an Eye on for Your LSTM Model - and Other Tips." *Medium*, Geek Culture, 25 May 2021, medium.com/geekculture/10-hyperparameters-to-keep-an-eye-on-for-your-lstm-model-and-other-tips-f0ff5b63fcd4.

[3] Matthew Stewart, PhD. "Simple Guide to Hyperparameter Tuning in Neural Networks." *Medium*, Towards Data Science, 10 Feb. 2023, towardsdatascience.com/simple-guide-to-hyperparameter-tuning-in-neural-networks-3fe03dad8594.

[4] Pawar, Dipti. "Improving Performance of Convolutional Neural Networks!" *Medium*, Medium, 3 Oct. 2020, medium.com/@dipti.rohan.pawar/improving-performance-of-convolutional-neural-network-2ecfe0207de7.

Appendix A

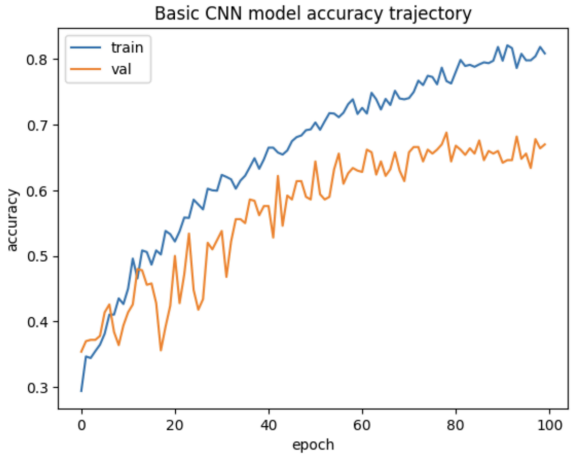

Figure 1: Basic CNN Training and Validation Accuracy

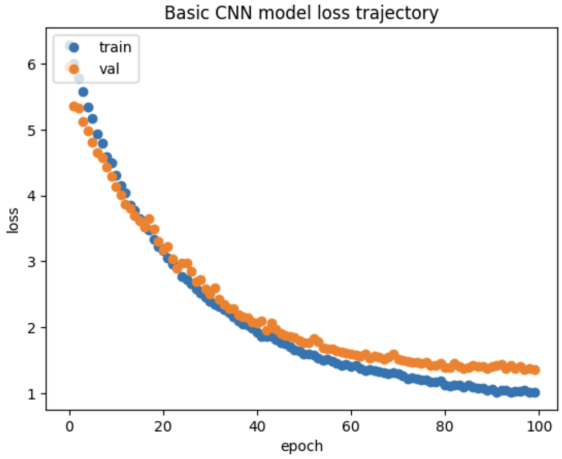

Figure 2: Basic CNN Loss Curve

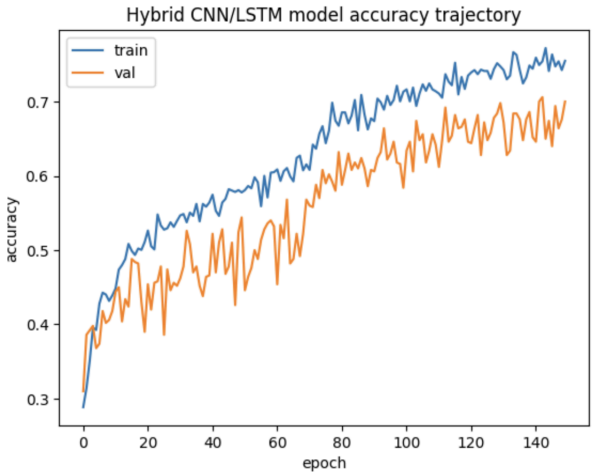

Figure 3: CNN + LSTM Training and Validation Accuracy

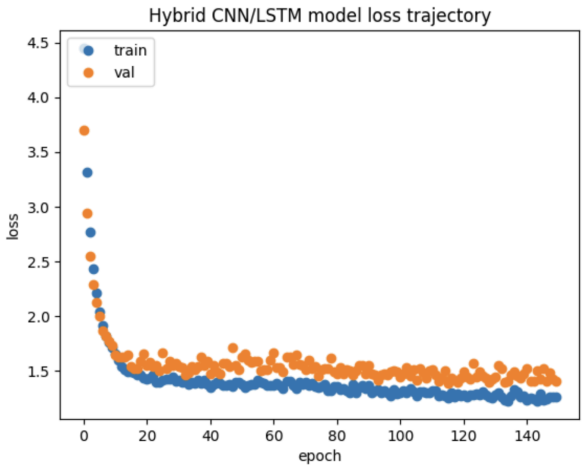

Figure 4: CNN + LSTM Loss Curve

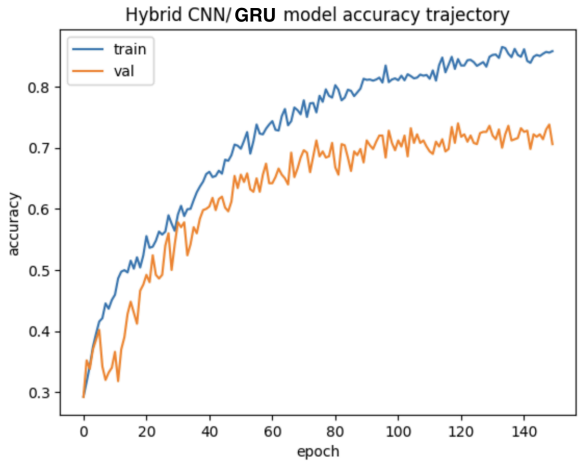

Figure 5: CNN + GRU Training and Validation Accuracy

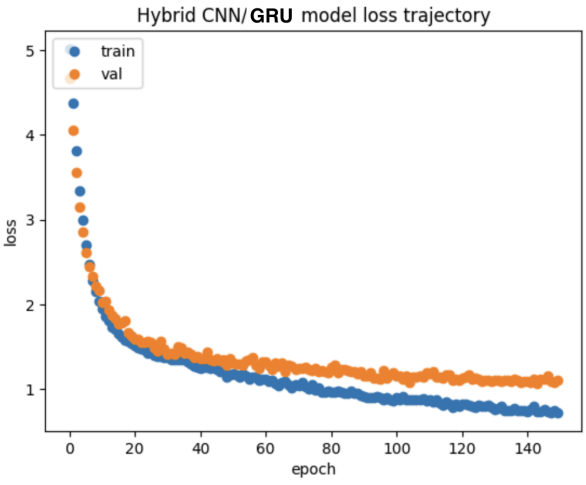

Figure 6: CNN + GRU Loss Curve

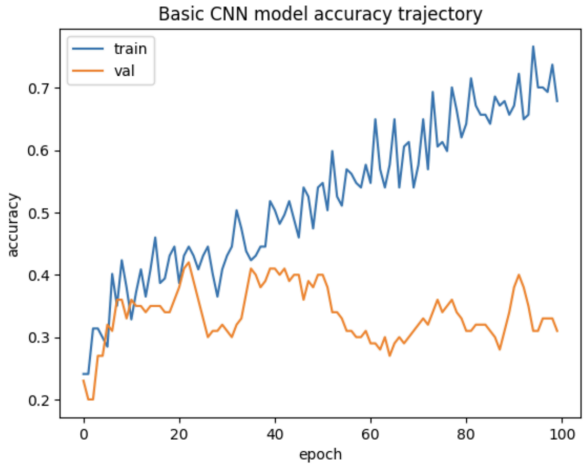

Figure 7: Basic CNN Training and Validation Accuracy on Subject 1

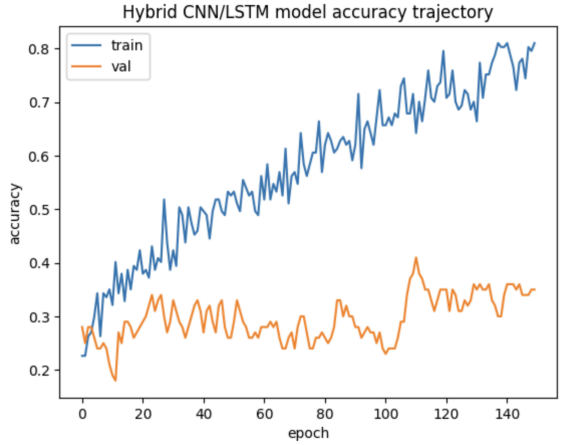

Figure 8: Basic CNN + LSTM Training and Validation Accuracy on Subject 1

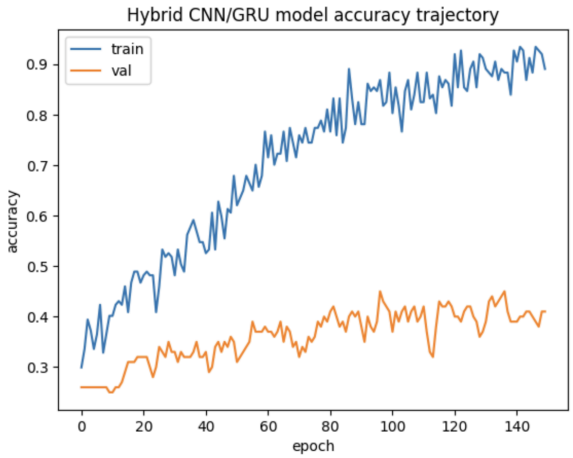

Figure 9: Basic CNN + GRU Training and Validation Accuracy on Subject 1

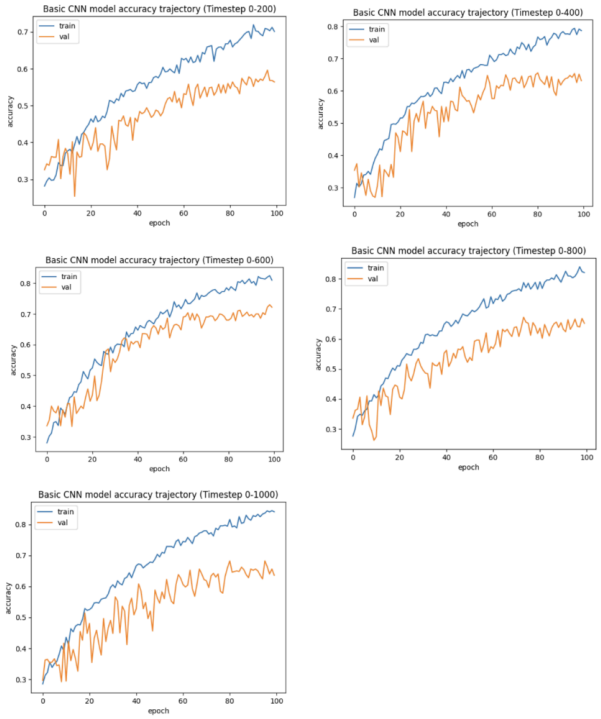

Figure 10: Basic CNN Model Over Various Time Intervals

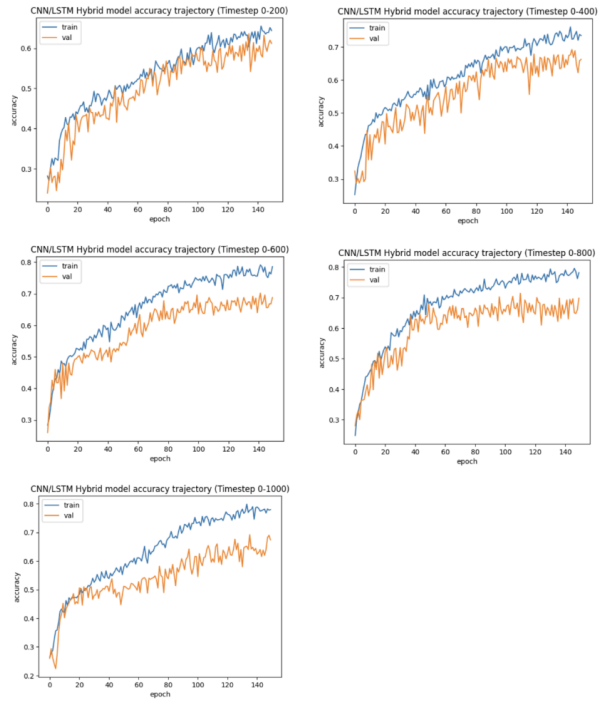

Figure 11: CNN + LSTM Model Over Various Time Intervals

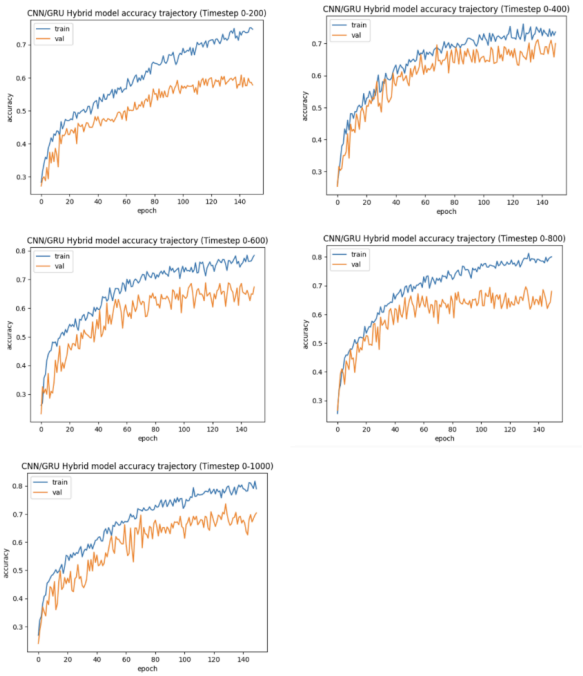

Figure 12: CNN + GRU Model Over Various Time Intervals

Appendix B

Table 1: Test Accuracy for Across All Subjects

Model	Test Accuracy (%)
CNN	70.65
CNN + LSTM	72.46
CNN + GRU	73.14

Table 2: Test Accuracy for Subject 1

Model	Test Accuracy (%)
CNN	31.83
CNN + LSTM	37.02
CNN + GRU	35.21

Table 3: Accuracy Across Various Time Intervals

Time Interval	CNN Test Accuracy (%)	CNN + LSTM Test Accuracy (%)	CNN + GRU Test Accuracy (%)
0-200	59.37	59.37	60.27
0-400	65.24	65.46	67.27
0-600	69.53	68.62	66.82
0-800	72.91	64.79	65.24
0-1000	67.04	64.79	67.72

Appendix C

Model Architectures

Convolutional Neural Network (CNN)

There are a total of 4 convolutional blocks, each specifying the convolutional layer, the pooling layer, the batch normalization, and the dropout. The 4 convolutional blocks are followed by a flattening layer and a fully connected layer.

Convolutional Layers:
- Filter sizes = 25, 50, 100, 200
- Kernel size = (10, 1)
- Padding = Same
- Activation Function = Elu
- L2 Kernel Regularizer = 0.015

Pooling Layers:
- Pool size = (3, 1)
- Padding = Same

Batch Normalization

Dropout:
- Dropout rate = 0.55

Flatten

Fully-Connected Layer:
- Units = 4
- L2 Kernel Regularizer = 0.015
- Activation Function = Softmax

Additional model parameters utilized in the compilation and training of the model include the following:
- Loss Function = Categorical Cross Entropy
- CNN Optimizer = Adam
- Learning Rate = 0.0009
- Decay = 0.001
- Epochs = 100
- Batch Size = 64

Convolutional Recurrent Neural Network (CRNN)

For CRNN, the 4 convolutional blocks for the CNN model are maintained with each specifying the convolutional layer, the pooling layer, the batch normalization, and the dropout. The 4 convolutional blocks are followed by an RNN block where a flattening layer, fully connected layer, reshape, and the RNN function, in this case LSTM and GRU, are defined. The RNN block is followed by a fully connected layer to complete the model. The two RNN models, LSTM and GRU are defined below.

Long Short-Term Memory (LSTM)

LSTM Layer:
- RNN Model = LSTM
- Units = 30
- Dropout = 0.50
- Recurrent Dropout = 0.10
- Return Sequences = False

Additional model parameters utilized in the compilation and training of the model include the following:
- Loss Function = Categorical Cross Entropy
- CNN Optimizer = Adam
- Learning Rate = 0.00075
- Epochs = 150
- Batch Size = 64

Gated Recurrent Unit (GRU)

GRU Layer:
- RNN Model = GRU
- Units = 30
- Dropout = 0.20
- Recurrent Dropout = 0
- L2 Kernel Regularizer = 0.015
- Return Sequences = False

Additional model parameters utilized in the compilation and training of the model include the following:
- Loss Function = Categorical Cross Entropy
- CNN Optimizer = Adam
- Learning Rate = 0.00075
- Decay = 0.001
- Epochs = 150
- Batch Size = 64