Part 2 – Wigglies

Om Borda (Matriculation no: 10000722)
Zaranaben Savani (Matriculation no: 10001007)

July 2025

Supervisor: Prof. Dr. Magda Gregorová

Submission Archive: `part2.zip` (Notebook, Code, Report, CSV)

Overview

The main aim of this project was to classify a time-series dataset known as "wigglies" using deep learning methods. We trained two types of neural network models—a Fully Connected Neural Network (FNN/MLP) and a Convolutional Neural Network (CNN)—on the provided training dataset to predict the class of each wiggly instance. Our objective was to develop accurate models that could generalize well to unseen data. Throughout the project, we experimented with various model architectures, data preprocessing strategies, and optimization techniques to improve classification performance and generalization. Finally, we generated predictions for the test dataset and evaluated the results based on model performance.

All Submissions Overview

During this project, we made a total of 8 Kaggle submissions. Each submission reflects a different stage in our training and evaluation process. We adjusted model parameters, data splits, and training strategies across several iterations. Below is a breakdown of the key methods (M1 to M4) that led to our final model.

M1: Baseline FNN and CNN (1 submission)

- Started by exploring the data to understand the shape, distribution, and variability of the input signals.
- Built a simple Feedforward Neural Network (FNN) with 1 hidden layer using ReLU activation and a Convolutional Neural Network (CNN) with 2 convolutional layers, followed by a flatten layer and a fully connected output layer.
- Used the Adam optimizer with a learning rate of 0.01 to fast convergence.
- Trained both models on the entire training dataset without a validation split to establish a baseline.
- Used cross-entropy loss as the objective function for multi-class classification.
- Models were trained for 10 epochs with a batch size of 64.

Result: The CNN model performed better then the FNN in terms of training accuracy, which indicate CNN is able to capture local features. However, both models overfitted quickly in the training process. This step helped us to learn more about how model works and nature of the dataset.

M2: Validation Split and Improved Architecture (2 submissions)

- Used an 80/20 train-validation split to monitor generalization and track overfitting.
- Increased the number of training epochs to 50 for better convergence.
- Added dropout layers in both FNN and CNN models to reduce overfitting.
- Increased the hidden layer size of the FNN to 512 units to improve learning capacity.
- Added an additional pooling layer to the CNN model to further reduce spatial dimensions.
- Decreased the learning rate to 0.001 to slow down training and allow for better weight updates.

Result: Both models achieved higher accuracy compared to the baseline. Overfitting was reduced to some extent, and the validation split provided better visibility into model performance during training.

M3: Data Normalization and CNN Enhancements (2 submissions)

- Applied normalization to the input data to ensure all features were on a similar scale, improving training stability.
- Reduced batch size from 64 to 32 to allow finer gradient updates and better generalization.
- Added a third convolutional layer to the CNN model to improve feature extraction.
- Introduced Batch Normalization layers after each convolutional layer to stabilize and speed up training.

Result: The FNN did not show significant improvement in accuracy. However, the CNN model became more effective in classifying the data, and the validation loss consistently decreased with each epoch, indicating improved training stability.

M4: Final Training Strategy (3 submissions)

- Implemented early stopping with patience of 10 epochs.
- The best model(CNN) was saved and used for final prediction.

Result: FNN did not show any improvement after all changes, CNN model gave the best results overall. The final submission file was `cnn_prediction4.csv`, and the model was saved as `cnn_model.pth`.