Introduction To Deep Learning Project Report

Group 4

Group members:

- Aryanti Harpal (2080706)
- Beliz Pekkan (2066519)
- Goja Modic Sila (2071091)
- Sherwin Lee (2077086)

Data Loading and Processing

The provided training dataset was loaded from local directories in .pkl format. The individual files were then converted into Mel spectrograms and then normalized. Using train_test_split, the dataset was split into training (95%) and validation (5%) sets.

Feature extraction was done on audio data and labels were prepared for both sets to facilitate model training and performance evaluation. This ensures our model learns effectively from the training data and is validated accurately.

Architecture design

The project utilizes fully connected layers. The first layer consists of 256 units using ReLU activation, then followed by batch normalization and dropout. This allows the input features to be transformed into a higher-dimensional space, allowing the model to capture complex relationships. The second layer consists of 128 units that utilize the same ReLU activation, batch normalization, and dropout. The function of this layer is to further refine the features extracted by the previous layer.

The third and final layer comprises 64 units using ReLU activation, batch normalization, and dropout. As the output layer, its role is to prepare the features for output. The output layer itself is a single-unit dense layer with linear activation that predicts the valence value, giving a continuous output.

In this architecture, components like batch normalization, dropout, and ReLU activation were chosen for specific purposes. Batch normalization ensures stable and faster training by normalizing the outputs of each layer. Dropout helps to prevent overfitting by randomly dropping units during training. ReLU activation introduces non-linearity, enabling the network to learn complex patterns.

The design of the architecture aims to balance complexity and generalization, using the aforementioned regularization techniques to enhance performance on unseen data.

Experiments

Bayesian optimization was used to tune our hyperparameters. In this setup, a defined optimize function takes as input the features and labels for both the training and validation datasets.

Inside the optimize function, a nested objective function represents the model's performance based on the input learning rate and batch size. With those two values, this function trains and evaluates the model, returning the evaluation result.

The hyperparameters' lower and upper bounds are defined in pbounds where the learning rate is searched between 0.0001 and 0.005, and the batch size is searched between 16 and 128.

A Bayesian Optimizer is created with the objective function and the bounds of the hyperparameters. The optimizer is then asked to maximize the objective function. The maximization process starts with 2 initial random points and then iterates 5 times to find the optimal learning rate and batch size.

The optimize function then returns the optimal hyperparameters and their corresponding maximum objective function value and prints them out into a table.

Iteration	Target	Batch Size	Learning Rate	MSE
1	0.6205	62.71	0.00363	0.6205
2	0.5287	16.01	0.001581	0.5287
3	0.6118	61.39	0.001776	0.6118
4	0.5456	73.83	0.005	0.5456
5	0.7194	128.0	0.001546	0.7194
6	0.5729	121.2	0.003511	0.5729
7	0.5802	125.9	0.002324	0.5802

Table 1: Results of Bayesian optimization

Results

After hyperparameter tuning, the best parameters were approximated as a batch size of 128 and a learning rate of 0.0015. The resulting Mean Squared Error (MSE) achieved with these parameters on the training data was 0.8713.

Furthermore, to assess the generalization performance of our model, we evaluated it on a separate validation dataset. The MSE obtained on this validation set was 0.5068, indicating how well our model performs on unseen data. Please read Appendix A for more details in regards to the MSE obtained per epoch.

These results suggest a reasonable level of performance in terms of minimizing prediction errors within our model. However, further analysis may be required to explore potential areas of improvement, such as feature engineering or model architecture adjustments, to enhance the predictive capability further.

Conclusion

- The model performed well with the chosen tuned hyperparameters.
- The architecture is effective for the chosen task.
- Further improvements could be made with additional data augmentation and tuning.
- Some discrepancies in the results suggest areas for further investigation.

Proposed Architecture

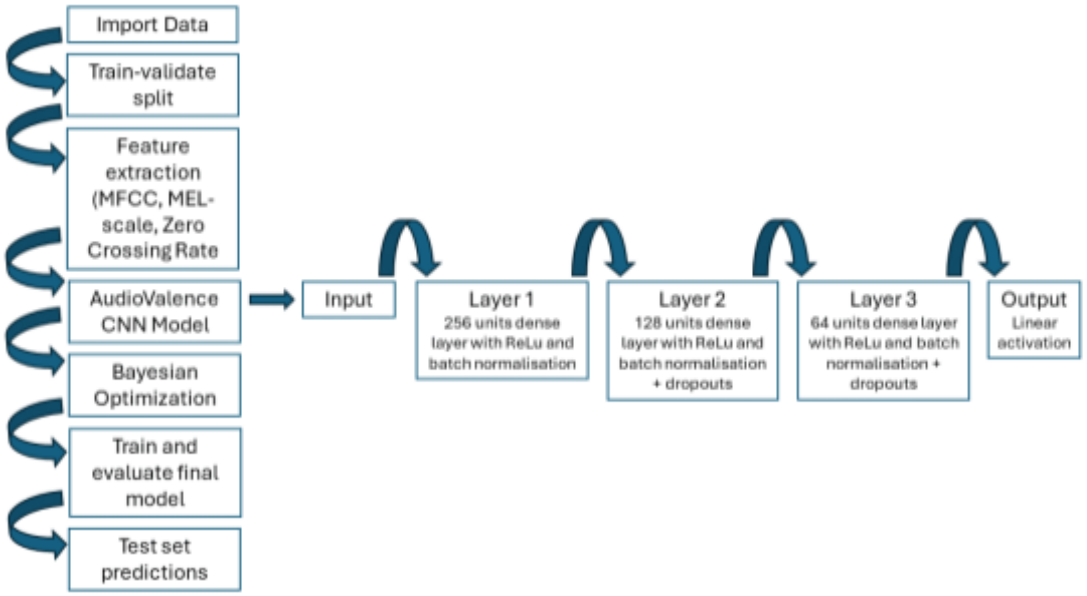

Figure 1: Diagram summarizing the proposed architecture

Appendix

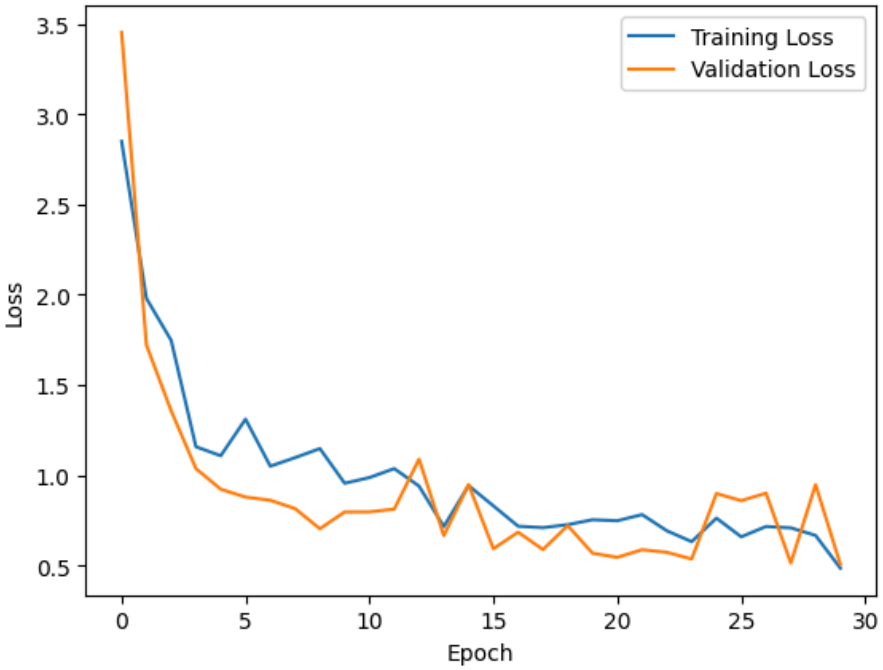

Figure 2: Training and validation loss per epoch