Time Series Forecasting Model description

The project consists of a time series forecasting problem, with a provided input of 48000 time series of length 2776, all of which have already been applied padding of zero sequences to make them all of the same length of 2776. Also all samples are each part of 6 possible categories labeled 'A','B','C','D','E' and 'F'. The goal is to provide a model that is capable to predict future samples of these time series without any constraints regarding time domains and/or specific time contexts and with high generalization capabilities.

My role focused on implementing the final model following data cleaning and augmentation stages. I opted for an ensemble approach, combining a custom transformer-based model with the established `CONV_LSTM` model, to leverage both sequential and spatial features in the data effectively.

First we define a transformer encoder block. It normalizes the input and then utilizes a multi-head self-attention mechanism (MultiHeadAttention layer) with a specified key dimension, number of heads, and dropout rate. After a dropout the input tensor is added to the attention output, creating a residual connection. A simple convolutional feed forward part is then added sequentially and the final output is the sum of the second convolutional layer's output and the residual connection of the MultiHeadAttention block.

This block is now used in the function build_model. Transformer encoders are stacked using transformer_encoder function iteratively. Global average pooling is used to reduce the dimensions and dense layers with the specified number of units in each layer (mlp_units) are created followed by dropout.

The model is then compiled using `Mean Absolute Error` loss function with the `Adam_optimizer`. The metric used in our for the callbacks is the `Mean Absolute Error` on validation set.

The second "branch" consists of Convolutional Long Short-Term Memory (`CONV-LSTM`) model. The input layer is followed by a bidirectional `LSTM` layer with 64 units that processes sequences in both forward and backward directions. A 1D convolutional layer is added with 128 filters and kernel size of 3. The final 1D convolutional layer (output_layer) is used to match the desired output shape (with 1 channel). As usual, the model is compiled using Mean Squared Error loss and the Adam optimizer. Each of the two models is trained separately with a batch size of 128 for up to 50 epochs

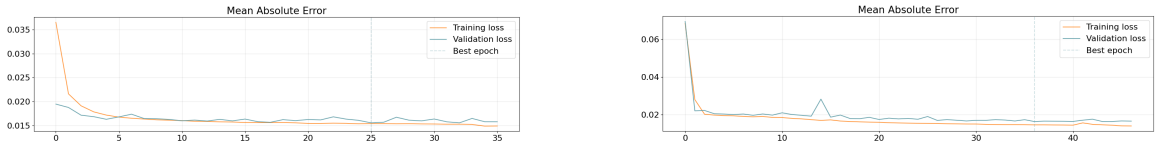

Figure 3: Combined Figures without Captions

(though training typically halts earlier with EarlyStopping and ReduceLROnPlateau callbacks to adjust the learning rate as needed). The outputs are then averaged (reshaping as necessary) to form the `ensemble_model`. The input for each model is structured as [X, X], where X represents an (n, 200) matrix, with n time series instances.

The results on a new, unseen test set are promising, achieving an `MSE` of 0.012245 and an `MAE` of 0.07767.