Comparison between DL models' performance on the regression task based on the UCI Heart Disease Dataset

Averoldi Lorenzo VR521277

Motivations

Statistically, parkinson, with Alzheimer's, is one of the most frequent degenerative diseases in the world.

At this day, there is no cure for parkinson; the only thing medicine can do for patients is try to treat symptoms and improve their quality of life.

The task in this case is regression, trying to predict parkinson’s progression in time, in particular the 2 progression indexes: motor UPDRS and total UPDRS (Unified Parkinson’s Disease Rating Scale).

State of the Art

Using MSE as a metric for comparison, the baseline is represented by a simple MLP, which achieved an MSE between 5 and 15.

Then in the scoreboard we see the models we use also in this work, Vanilla RNN, LSTM, LSTM and BiLSTM which reaches an MSE well below the baseline's one.

The current SOTA is represented by Pure Transformer (0.1-0.4 MSE) and LSTM + Attention + Ensemble, with an MSE lower than 0.1

Data preprocessing

The dataset is composed of 19 features, **age** is the only integer one, while the others are continuous.

The entries are biomedical voice measurements from 42 patients with parkinson in early stage, who joined a recruitment for trial of a telemonitoring device for remote symptom progression monitoring.

For MLP, we preprocess data with 2 simple Scalers, respectively for labels and data, then we create a dataloader with the training dataset and one with the testing one, and we are ready to train the MLP.

For the other models instead, we need to treat the data in sequences of observations, so we decide a window size(5 obsv.) and riorganize data in sequences of 5 observations, then we use again 2 simple scalers and the dataloaders for training and testing

Models

MLP: A simple Multilayer Perceptron to use as a baseline, it’s composed by 3 linear layer fully connected with 2 ReLu between the 3 linear layers.

Vanilla RNN: A simple recursive neural network, As weights we use 2 linear fully connected layers, **Wxh** for the input sequence and **Whh** for the memory in time.

Vanilla LSTM: Similar but very different from RNN: it includes two states: a hidden state and a cell state, both vectors of length n, the cell contains long term information, and LSTM can read, write or erase from the cell. The selection of which information is erased/written/read is controlled by three corresponding gates, vectors again of length n.

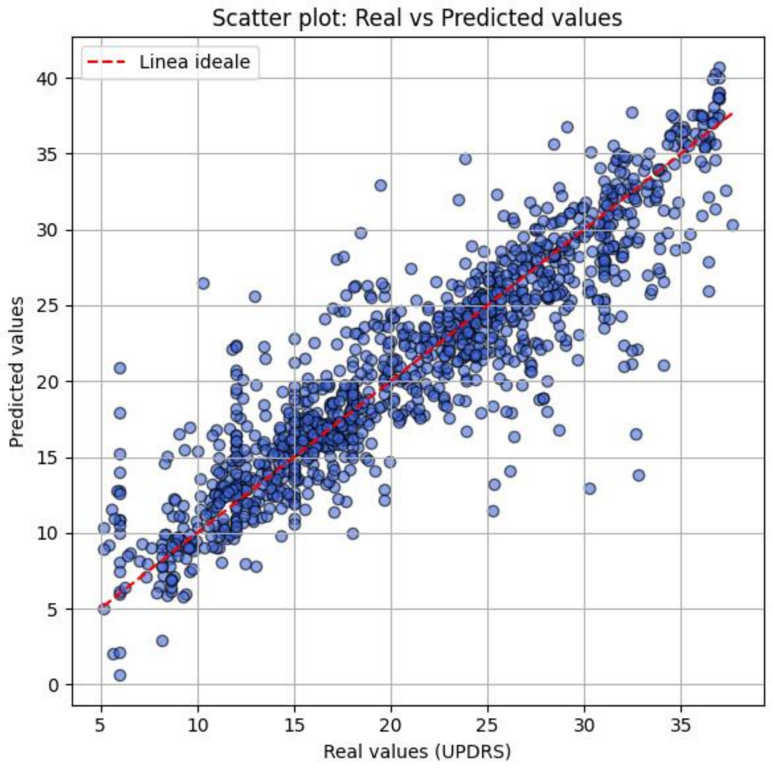
Attention-LSTM: The mechanism it’s identical to Vanilla LSTM, but we also have a list that during the forward pass accumulates and memorizes all the hidden states. Then, in the main LSTM, we compute the attention weights from that output sequence and the context vector from the attention weights multiplied by the output, then we use a last linear layer for the final prediction.

Bi-LSTM: It uses a vanilla LSTM and an LSTM that works backwards, meaning it takes in input the reversed sequence, then we get the final hidden states from the 2 LSTMs and concatenate them together, to finally pass the concatenated state through a last linear layer.

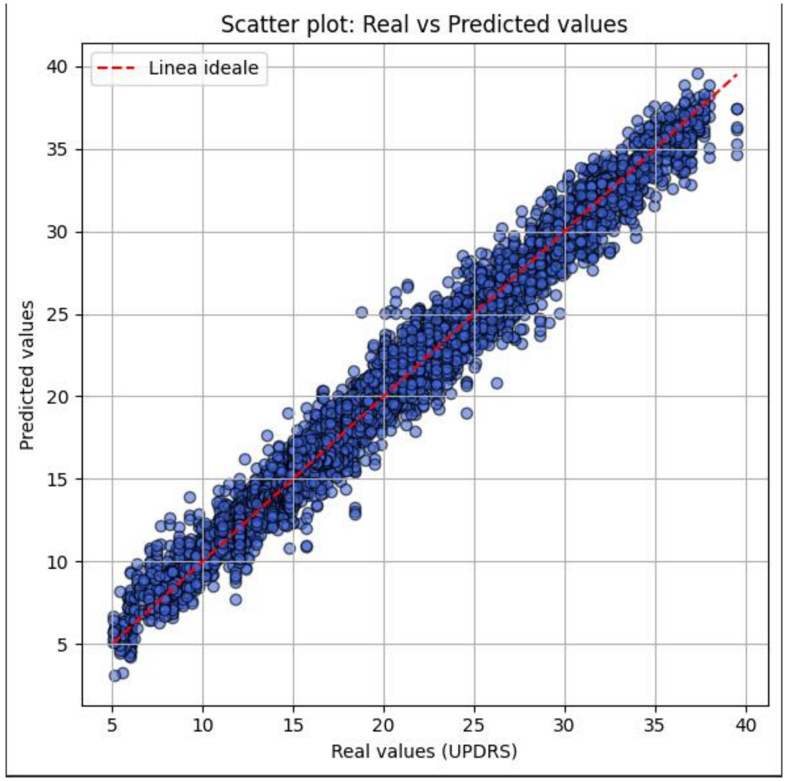
It’s powerful, but slow, it takes twice the time of training respect to a normal LSTM

Results

MLP BASELINE

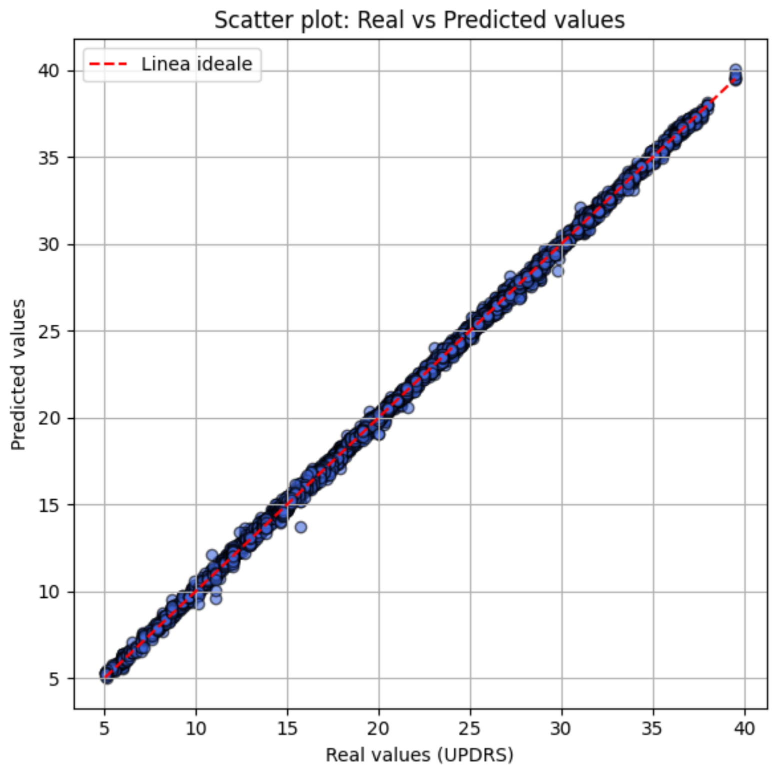


Vanilla RNN

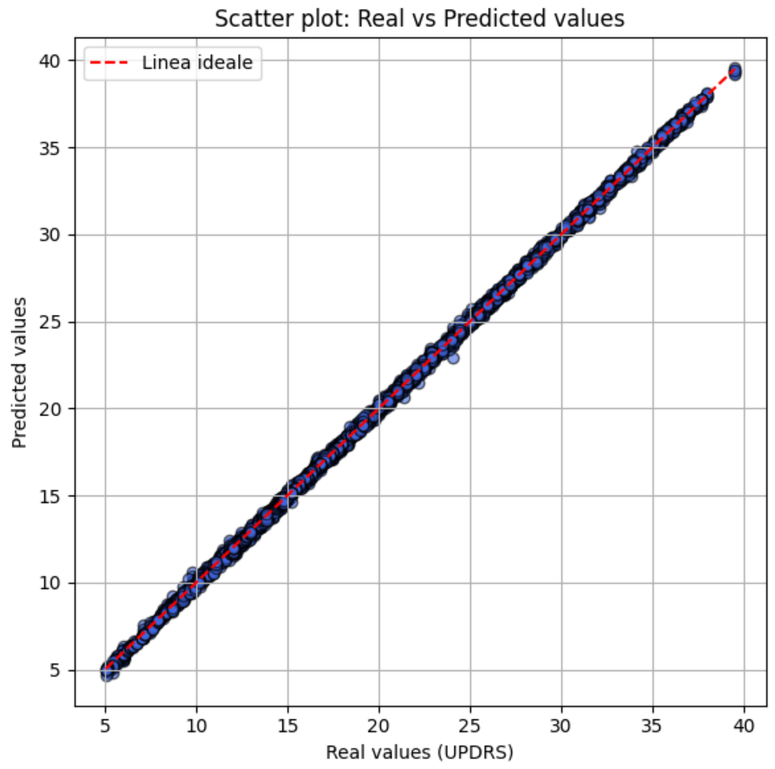


Results

Vanilla LSTM



Bi-LSTM



Conclusions

The models performed as expected, with performance improving proportionally with the complexity of the model’s memory.

Possible future improvements should focus on trying also with the other target (total UPDRS) and trying also a regression with both the targets in multioutput.