Deep Learning Assignment 2 Report

This is a summarization of the work we did in this assignment; all the answers and solutions are available in the notebooks with considerably more depth to them.

Time Series:

1.a + b:

While exploring the data we found and fixed a few interesting problems.

i. First, we saw that a lot of the heart rate measurements are missing because of the way it was sampled. The heart rate was sampled much less frequently than other measurements making it seem discrete in comparison. We fixed this issue using liner interpolation with smoothed the values over. We also decided to fill in all the other missing values using liner interpolation.
ii. Next was the issue of the ninth subject, its data was very odd, containing only one activity, and very little time steps compared to the others. We decided to ignore this subject to not taint our models learning ability with unrepresentative data.
iii. Lastly, we found a strong correlation between the high-resolution sensors and their low-resolution counter parts. We decided to drop the low-resolution measurements.

1.c:

We could attempt to use self-supervision in order to fill the missing readings in any of the features. Though as we showed above, heart rate changes very slowly and as such is probably left to interpolation. Another use could be predicting the heart rate of the subject in the next time step.

2.a:

We use 5-fold cross validation, to evaluate our models, and reshape the data into examples that contain some number of time steps each. (TIME_SAMPLE_SIZE – is the variable controlling the number of time steps in each example.)

2.b:

We created a very simple naïve model that assigns the probability of being in each class according to a normal distribution with its location being the probability of having the same class across the entire training set. We achieved around 0.08 accuracy on both validation and testing.

2.c:

For the classical machine learning model we used a SGD model which achieved much better results with around 0.6 validation accuracy and 0.4 testing accuracy. The analysis of these results can be found here https://wandb.ai/zshoham/dl_assignment2/reports/Analysis-of-SGD-Results--VmlldzozNzQ4NDA.

2.d:

We start with a CNN network which performed very well, achieving around 0.95 validation accuracy and 0.7 testing accuracy. The analysis of the results can be found here https://wandb.ai/zshoham/dl_assignment2/reports/Analysis-of-Shallow-CNN-Results--VmlldzozNzQ4OTk

2.e:

For the pretrained model we decided to train a similar network to the previous CNN as a regressor predicting the heart rate at the next time step. We also needed to reshape the data a little, removing the last time step in each example and taking its heart rate to be the target for that example. We achieved very good results predicting to within 1 heartbeat of the real heart rate. We then took the pretrained regressor and changed its Dense layer to fit a classification problem. The results we got were not as good as the original CNN classifier though they still were much better than the SGD model.

2.f:

We suggest three improvements for our CNN model:

- The first is very simple, improving the CNN model by making it deeper, adding more connections, et cetera. The shallow model is minimal and has much room for experimentation.
- Secondly, we could try different model architectures. Time series tasks are generally thought to benefit from RNN architectures.
- Finally, we could always optimize the model by changing hyperparameters like batch size, optimizer, and the number of epochs.

2.g:

We chose to implement the first and second options, meaning we created two more models. A deeper CNN model, and an RNN model. A summary and comparison of the algorithm is available here https://wandb.ai/zshoham/dl_assignment2/reports/Summarizing-Report-for-Activity-Classification--VmlldzozNzQ5NDY

Additionally, a table with all our runs is available here https://wandb.ai/zshoham/dl_assignment2/table?workspace=user-zshoham

3.a:

Our group name will be BGU-DL2021_YHB.

We joined the competition and try to explore the data. We saw the data is spread over few tables so we tried to merge all the data we saw relevant- all the data except for the holidays and the transactions tables, which we chose not to include- the transactions because we don't have this information for the test time, and the holiday because it was hard to integrate (there are local holidays, and national, which is hard to take in consideration) and we thought the work isn't worth it.

3.b:

We chose to use Random forest regressor for our classical ML model. We tried several other algorithms such as SVM and lightGBM. SVM tool a lot of time to fit, and lgbm gave bad results so we chose RF-100 trees with max-depth of 11 (number of features fed to the model). The model results were average-private and public score of 1.44- below the score for the sample submission where it is all zeroes predictions.

3.c:

The required preprocessing for creating an embedding network is to create encoding for each of the categorical features (to be between 0 to n-1)- which we already did before.

3.d:

For the embedding network we chose to use all the categorical features we had- class, on promotion, store, type, city, state, cluster, item, family, day. We tried several networks with different depths and got to conclusion the network should be shallow, because the deeper it got the worst results it gave.

We received for the validation NWRMSLE result of 0.687852 and could see nice correlation between the predicted values to the received values in the graph we plotted. Yet, for the test set we received private and public score of 1.34- better than before, but yet poor results.

3.e:

Here we chose to add to the embedding network from the previous question the oil prices as another feature.

We received for the validation NWRMSLE result of 0.695713 and could see nice correlation between the predicted values to the received values in the graph we plotted. Yet, for the test set we received private and public score of 1.33- better than before, but by very little.

3.f:

We chose to look at the embedding of the store numbers, compared to the original store clustering received in store dataset. We plotted all the stores after embeddings and transformation from 15 dimensional vector to 2D vector, and saw that all the stores in clusters 16, 4, 15 aligned in one line- which might suggest the model was able to learn good embedding to part of the stores, but not to others (like cluster 2 which is very scattered).

3.g:

In order to use the embeddings as feature extractor, we tuned the embedding model from 3.d right after the flatten layer to receive all of the new features produced by the embeddings. Then, to create the new dataset- we fed the network with all our data to receive the new features. We received 125 new features.

After some tries on different algorithms like SVM and RF which took too much time to run, and SGD which took too much memory which we didn't have at this point, we used the LGB model, which gave very poor results- NWRMSLE score of 0.848414 for the validation set, and 1.52 private and public score for the test set.

4.

The notebook is available on Kaggle to view:

https://www.kaggle.com/hodtwito/bgu-dl2021-yhb