CMPE 492

Comparison of Gated Recurrent Neural Networks and Transformer Network for Time Series Classification and Regression

Salih Bedirhan EKER

Hakan BALIK

Ece Dilara ASLAN

Advisor:
İnci Meliha Baytaş

Table of Contents

1. Introduction

In today's world the reputation of Artificial Intelligence (AI) has been increased a lot. Some of the big reasons behind this are: AI helps us automate tasks and processes, it provides data and insights that we can not obtain on our own, it can interact with machines in a more natural way such as through voice recognition and Natural Language Processing (NLP), it has a good role on extending cognitive abilities such as providing memory assistance, it is used to create personalization such as through recommendations based on past behaviors, and much more. Also, as the big hardware companies manufacture better CPUs, GPUs and one that is specifically created for this domain which are TPUs, we now have much more computation power which is a huge necessity to run healthy AI models.

As the interest deepened, people came up with a more specialized domain which is Machine Learning (ML). ML differs from AI in a way that it mainly focuses on creating a technique such that a computer can learn from data. One other distinction is that ML focuses on identifying the patterns and improving the prediction accuracy whereas AI just focuses on mimicking the wise behaviors that humans perform. It is important to note that AI encapsulates ML such that every ML model can be also labeled as an AI model.

Going one more deeper in this encapsulation chain one can see a more advanced area of AI and also ML, the Deep Learning (DL). As discussed earlier, ML focuses on learning from data; in accordance to that DL also focuses on the same problem but the main difference is that it uses a structure called "Neural Networks". One can simply define a Neural Network as a computer system that is modeled after the brain; it is composed of a large number of interconnected processing nodes, or neurons, that work together to solve specific problems. So the main distinction of these two are: ML is not explicitly programmed whereas DL uses a much more complex structure which is modeled on the human brain. Similar to the relation of AI and ML, DL is a subset of ML so any DL model can be seen as an ML model.

In our project, we seek to compare the popular Deep Learning models which are Gated Recurrent Neural Networks (Gated RNN) and Transformer Networks. The comparison is based on the training time and train & test accuracy among different datasets with various sequence sizes, statistical analysis (univariate / multivariate) and classification / regression. It is also fruitful to mention what classification and regression are. Classification is the process of predicting the category of the given data entries; on the other hand, regression is a technique which helps in finding the dependence between variables and predicts the continuous output. More detailed information is available under *2.1 Project Definition* and *5.1 System Requirements* sections.

2. Project Definition and Planning

2.1 Project Definition

As described briefly in the previous section, the aim of our project is to compare Gated RNN and Transformer Networks. The following are definitions and properties of these models.

RNN is a type of neural network that is designed to handle sequential data. RNNs are used in a variety of tasks, including machine translation, speech recognition, and image captioning. In RNN, weights and biases for all the nodes in the layer are the same. The gradient descent function of RNN is: *w(n+1) = w(n) - (eta x gradient(cost function)).*

Gated Recurrent Unit (GRU) is a version of RNN designed to overcome the vanishing gradient problem which states the earlier inputs' effects on the final model are becoming negligible. GRUs use connections through a sequence of nodes to perform machine learning tasks associated with memory and clustering, for instance, in speech recognition. GRU consists of two gates: Reset which is used to decide whether the previous cell state is important or not and Update, which decides if the cell state should be updated with the candidate state (current activation value) or not.

One other version of RNN is *Long Short-Term Memory (LSTM)* which is well-suited to learn from sequences of data. LSTM networks are well-suited to tasks such as natural language processing. There are two extra gates compared to GRU in LSTM: Forget, which decides how much information from the previous state should be kept and forget remaining and Output, which determines what the next hidden state will be. In general, GRU uses less memory and is faster than LSTM, however, LSTM is more accurate when using datasets with longer sequences.

Lastly, there is the Transformer Networks which differs from the previous ones such that it benefits from parallelization whereas aforementioned ones work iteratively. They are composed of a series of interconnected transformers. These networks are often used for tasks such as image recognition and natural language processing. The attention-mechanism looks at an input sequence and decides at each step which other parts of the sequence are important. Transformer Networks consist of Encoder and Decoder. Both Encoder and Decoder are composed of modules that can be stacked on top of each other multiple times.

This project aims to display the differences between these 3 models: GRU, LSTM and Transformer for both classification and regression. The results could be used on different domains since these models are very fundamental. Even our classmates, who are working on different AI projects, pointed out that it would be fruitful for them to use the outcomes of this project.

2.2 Project Planning

2.2.1 Project Time and Resource Estimation

The cumulative time spent up until this progress report are as follows:

Table 2.1. Duration table.

Task	Duration
Assembling team, reaching out to advisor and hand-shaking on the project	1 week
Overall meetings with advisor	~5 hours
Overall internal meetings	~25 hours
Literature review and researching about domain terminology	1 week
Researching appropriate datasets	5 days
Implementation	1 week
Testing	1 day
Listing down the results and findings	4 hours
Creating the Progress Report	2 days

Regarding resources, members basically used their own personal laptops to accomplish any computation required tasks. The datasets that were analyzed until now did not require huge computation power, but for the remaining datasets online tools such as Google Colab might be needed.

2.2.2 Success Criteria

The success criteria for our project is simply being able to compare different datasets with different characteristics and come up with data & observation dependent results. It would be also great to generate insights, coming up with universal or near results using a small sample.

2.2.3 Risk Analysis

Even though it looks like there are a lot of online resources for open datasets, most of them are erroneous and illogical. So it was troublesome to come up with proper ones. We even had to find a new dataset for the category *"long sequence, univariate, many-to-one classification"* since during our testing phase it resulted in poor accuracies around ~5%. It took a while to deduce the data was not trainable. So in short, the risky part of the project is to find the appropriate datasets to work on. Also since training and testing the models take a considerable time, one should implement necessary debugging mechanisms to ensure everything is working as intended during and after the execution.

2.2.4 Teamwork

Even though in most of the projects, more people result in more problems. This is mainly due to the job assignment, finding the convenient time to have meetings and lastly merging the progress done by different colleagues. But in our case, we managed to hold meetings internally & with our advisor in a great manner since each member of our team is responsible and easy-going. As a result, we organized our roadmap properly, distributed the work among us, took necessary actions, and most importantly we worked asynchronously. For example, while one member was working on the implementation task another one was working on the progress report. It is also important to point out that this approach did not result in one member focusing on only the task that was assigned to him/her. During the meetings with our advisor, every member took notes to prevent missing out points and to double check. We also held frequent online & face to face meetings to show our individual and collaborative progresses.

3. Related Work

Dealing with time series problems is a popular area and there are various research and approaches. We will give some information about different papers and researches on this area that we have used.

[1] Investigate the differences between time series transformers and LSTM with self attention, and compare performances. The research shows that LSTM with self attention is a very powerful solution approach. We may use this approach on our datasets in order to compare the performances.

[2] Measure the performance of Recurrent Neural Networks on time series classification using a large dataset archive (UCR) and this research shows a comprehensive result of LSTM and simple RNN usage on time series classification.

[3] Tries to compare forecasting power of LSTM and GRU on multivariate financial data. Although used data is very tricky, the research brings comprehensive results that GRU outperforms LSTM on that task.

[4] It also works with Recurrent Neural Networks, but this time research tries to cope with forecasting problems. The perspective of this research tries to investigate different aspects of Recurrent Neural Networks on forecasting. We can see that forecasting problems are difficult even if the used structure is very strong.

4. Methodology

Our task is comparing different deep learning solution approaches for several time series tasks. In this section we explained our solution approaches, difficulties we faced and our future work plans.

Firstly we choose PyTorch as our main development tool, decide solution methods (LSTM, GRU and Transformer) and decide tasks (task details are explained below sections) with our advisor.

We used LSTM, GRU and Transformer models in order to overcome the tasks. We categorize tasks according to:

- Task (many-to-one classification / many-to-many classification / many-to-one regression / many-to-many regression)
- Feature size (univariate / multivariate)
- Sequence length (short / long time sequence)

In order to compare the solution approaches, we need to learn models. We read papers, blogs, documentations and watch tutorial videos so that we can distinguish advantageous and disadvantageous sides of the models. Then we learn different implementation details of the models.

After that, we find a lot of different datasets for different tasks. Up to now we use 3 different datasets and we will give details about two of them and one of them was not applicable on current models. We also will explain difficulties about it later on. Information about many-to-one classification datasets are on Table 4.1.

Table 4.1. Datasets of many-to-one classification task.

	Name	Training Size	Test Size	Number of Features	Number of Classes	Sequence Length
Dataset #1	SmoothSubspace	150	150	1	3	15
Dataset #2	SyntheticControl	300	300	1	6	60
Dataset #3	PenDigits	7494	3498	2	10	8
Dataset #4	EEG	122	-	64	2	256

Then we implement the models on datasets. We implement LSTM, GRU and Transformer on Dataset #1 and Dataset #2. We train successful models using LSTM and GRU however, we could not obtain satisfying results using Transformer. In the future we will try to build a stronger Transformer model. Finally, we obtain the results and discuss them in the next chapters.

We tried to use normalization techniques in our datasets even though our datasets are univariate time series. First we ran our model without normalization, then we normalized our data and ran the model. Finally we used dynamic normalization and ran the model. After obtaining the results, we noticed that our accuracy did not get better (accuracy results got worse 1-2%). Therefore we decided not to use normalization techniques on the first datasets. In the future we will try these normalization techniques on multivariate datasets as well.

We used a dataset called Adiac (about identification of unicellular algae) as a long sequence dataset. Ist time series length is 176 and the number of classes is 37. We trained LSTM and GRU models on this dataset. However we haven't made enough progress. The data is hardly classifiable as we can see from Figure 4.1, different classes are represented with different colors, and our current models can not learn much information from the data. Then we have decided to leave this dataset for now. If we come up with new improvement methods, we can train new models on this dataset in the future.

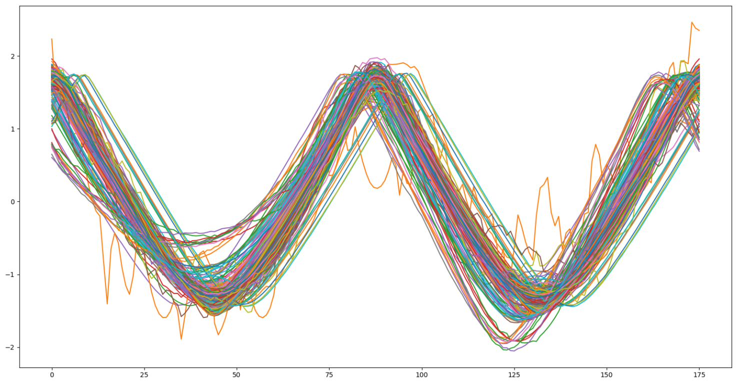

Figure 4.1. Adiac dataset.

5. Requirements and Modeling

5.1 System Requirements

We used an anaconda work environment in order to organize our libraries and packages.

Libraries and software versions are as follows:

- Python v3.10.8,
- PyTorch v1.13.0
- Matplotlib v3.6.2

Hardware information are as follows:

- CPU: Intel Core i5-10300H 2.5(4.5GHz)
- GPU: Nvidia GeForce 1650 Ti
- Memory: 16GB

5.2 System Architecture

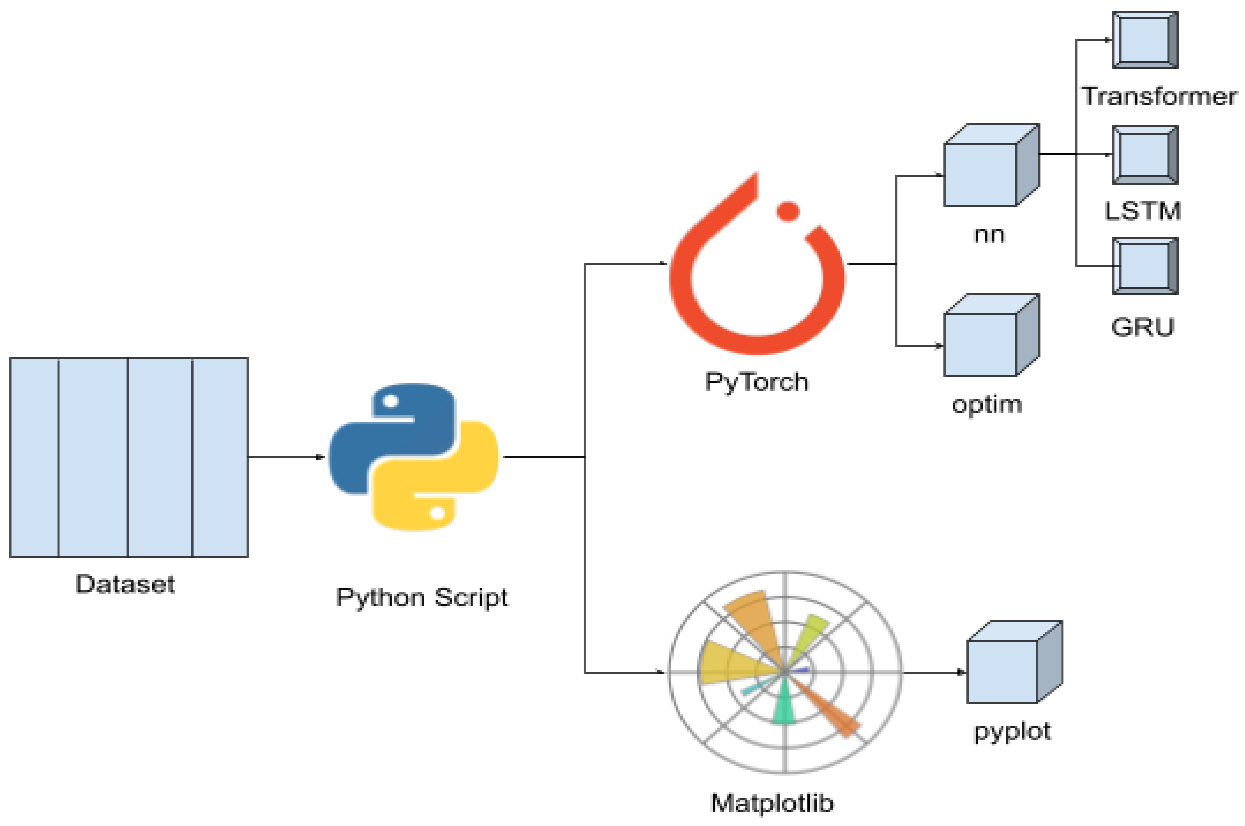

Figure 5.2.1. Architecture.

6. Design, Implementation and Testing

6.1 Implementation

In this section I will explain the coding part. We stored our code in a private repository. We have dealt with two tasks up to now:

- Short sequence, univariate, many-to-one classification (Dataset #1)
- Long sequence, univariate, many-to-one classification (Dataset #2)

We knew that the datasets are .txt format and do not have header line and index columns and also datasets are univariate. Therefore reading is relatively easy and we did not need any additional libraries (in the future we will need pandas in order to obtain the next datasets). Then we visualized the data so that we have an opinion about the pattern of the dataset (like seasonality) and get a clearer vision. We just plotted and did not apply heatmap since univariate data can be visualized well enough.

We implemented our models in separate files. The main difference between implementation files occurs because nn models. Other parts are the same (other than data reading and visualization). Therefore, I will explain the general structure of the code.

After reading (then visualizing if you want) the data as PyTorch tensors, we made device CUDA settings at the beginning of the code. Then we run the code 10 times and save the results. I will explain the process of every run.

Firstly we create our model object. We define our model in a class. In the class we used nn.Module and initialized the model. We used LSTM and GRU models up to now. As a model output we obtain hidden states. We want to build a many-to-one model therefore we get only the last value of the output.

Then we created an optimizer and loss function. We used torch.optim.Adam optimizer and nn.CrossEntropyLoss. Finally we trained our model. Training function takes input data, labels, model, optimizer, loss function, device and epoch number. In the train function, first we give the data to the model, then we take the predictions and calculate the loss. After that, we used the optimizer and made backpropagation. Lastly, we print the epoch result (every 10 epochs) and calculate time.

After training the model, we test our model with train and test sets and get the accuracy values and time results.

6.2 Testing

We measure training time and accuracy values as our testing criteria. In order to test accurately we keep the hardware the same. We run the LSTM and GRU models on dataset #1 and dataset #2. Also we run the models 10 times because of accuracy and time instability reasons. We got average and max/min values.

7. Results

Training time, testing time of training set, testing time of test set, training accuracy and testing accuracy of all runs and their average, maximum and minimum values which are all listed under the Appendix A: Execution Metrics. Results are demonstrated as tables for each dataset and model.

Table 7.1 and Table 7.2 are prepared by using the results of all runs. Table 7.1 shows the values of LSTM using dataset #1 and GRU using dataset #1. Table 7.2 shows the values of LSTM using dataset #2 and GRU using dataset #2. Both Table 7.1 and Table 7.2 provide average training time of 10 runs in seconds, average training accuracy, average testing accuracy and maximum testing accuracy of 10 runs.

Table 7.1. Dataset #1 results.

	Average Training Time (s)	**Average Training Accuracy**	**Average Testing Accuracy**	**Maximum Testing Accuracy**
Dataset #1 with LSTM	155.2734375	0.9406666667	0.9206666667	0.9866666667
Dataset #1 with GRU	161.6546875	0.8713333333	0.8673333333	0.9266666667

Table 7.2. Dataset #2 results.

	Average Training Time (s)	**Average Training Accuracy**	**Average Testing Accuracy**	**Maximum Testing Accuracy**
Dataset #2 with LSTM	247.1765625	0.7226666667	0.694	0.8333333333
Dataset #2 with GRU	272.56875	0.78	0.7656666667	0.8766666667

8. Conclusion

As we can see from Table 7.1 and Table 7.2, GRU has a longer training time regardless of dataset, although dataset #1 and dataset #2 have different sequence length, different number of classes and different sample size. This observation conflicts with our prior research. GRU has fewer gates and so fewer number of weights and parameters to update during training. Therefore, GRU must be faster to train theoretically. We will try different datasets to figure this situation out.

Average training accuracy and average testing accuracy are higher using LSTM for dataset #1. On the other hand, average training accuracy and average testing accuracy are higher using GRU for dataset #2. The main difference of dataset #1 and dataset #2 is the difference of their sequence lengths. Dataset #1 has a sequence length of 15 and dataset #2 has a sequence length of 60. In conclusion, it is observable that LSTM gives better results for shorter time-series sequences and GRU gives better results for longer time-series sequences.

9. Future Work

We need to improve the Transformer model for dataset #1 and dataset #2 in order to get acceptable results. We need to implement the LSTM, GRU and Transformer models for dataset #3 and dataset #4. Moreover, another comparison criteria of memory usage will be added for existing models. For other task cases (many-to-many classification, many-to-one regression and many-to-many regression), short / long sequence and univariate / multivariate dataset cases will be found. LSTM, GRU and Transformer models will be implemented according to datasets.

Training time, training accuracy, testing accuracy and memory usage will be found for datasets of remaining cases. Moreover, heatmaps will be generated for hidden states of LSTM and GRU and for attention values of Transformer. These results will be analyzed. The decision of using LSTM, GRU or Transformer for a problem will be optimized according to the dataset properties of the problem.

References

[1] Katrompas, A., Ntakouris, T., Metsis, V. (2022). Recurrence and Self-attention vs the Transformer for Time-Series Classification: A Comparative Study. In: Michalowski, M., Abidi, S.S.R., Abidi, S. (eds) Artificial Intelligence in Medicine. AIME 2022.

[2] Smirnov, Denis & Mephu Nguifo, Engelbert. (2018). Time Series Classification with Recurrent Neural Networks.

[3] Garcia Torres, Douglas & Qiu, Hongliang. (2018). Applying Recurrent Neural Networks for Multivariate Time Series Forecasting of Volatile Financial Data.

[4] Petneházi, Gábor. (2018). Recurrent Neural Networks for Time Series Forecasting.

Appendix A: Execution Metrics

Table 1. Dataset #1 - LSTM results.

	Training Time (s)	**Testing Time of Training Set (s)**	**Training Accuracy**	**Testing Time of Test Set (s)**	**Testing Accuracy**
Run 1	148.609375	0.125	0.96	0.125	0.9866666667
Run 2	147.734375	0.125	0.9066666667	0.125	0.84
Run 3	157.625	0.125	0.9533333333	0.125	0.94
Run 4	152.859375	0.125	0.9466666667	0.125	0.9333333333
Run 5	161.578125	0.125	0.94	0.125	0.9533333333
Run 6	159.203125	0.125	0.94	0.125	0.8866666667
Run 7	153.671875	0.125	0.9066666667	0.125	0.9066666667
Run 8	160.203125	0.109375	0.9733333333	0.125	0.94
Run 9	160.890625	0.125	0.9466666667	0.125	0.8666666667
Run 10	150.359375	0.125	0.9333333333	0.125	0.9533333333
Avg	155.2734375	0.1234375	0.9406666667	0.125	0.9206666667
Max	161.578125	0.125	0.9733333333	0.125	0.9866666667
Min	147.734375	0.109375	0.9066666667	0.125	0.84

Table 2. Dataset #1 - GRU results.

	Training Time (s)	Testing Time of Training Set (s)	Training Accuracy	Testing Time of Test Set (s)	Testing Accuracy
Run 1	159.15625	0.140625	0.9266666667	0.125	0.9266666667
Run 2	159.1875	0.15625	0.9266666667	0.125	0.9066666667
Run 3	160.1875	0.15625	0.9066666667	0.15625	0.8733333333
Run 4	163.484375	0.140625	0.9066666667	0.140625	0.9066666667
Run 5	162	0.15625	0.8933333333	0.140625	0.9
Run 6	158.59375	0.140625	0.78	0.15625	0.8133333333
Run 7	162.46875	0.15625	0.84	0.140625	0.88
Run 8	164.671875	0.15625	0.7	0.171875	0.6866666667
Run 9	165.234375	0.140625	0.92	0.15625	0.88
Run 10	161.5625	0.1875	0.9133333333	0.15625	0.9
Avg	161.6546875	0.153125	0.8713333333	0.146875	0.8673333333
Max	165.234375	0.1875	0.9266666667	0.171875	0.9266666667
Min	158.59375	0.140625	0.7	0.125	0.6866666667

Table 3. Dataset #2 - LSTM results.

	Training Time (s)	Testing Time of Training Set (s)	Training Accuracy	Testing Time of Test Set (s)	Testing Accuracy
Run 1	248.6875	0.828125	0.57	0.84375	0.5666666667
Run 2	243.140625	0.84375	0.8466666667	0.84375	0.8333333333
Run 3	248.015625	0.875	0.7866666667	0.875	0.7466666667
Run 4	252.421875	0.859375	0.6833333333	0.84375	0.6133333333
Run 5	252.625	0.890625	0.61	0.859375	0.6
Run 6	253.15625	0.890625	0.7366666667	0.875	0.7233333333
Run 7	247.515625	0.859375	0.7633333333	0.84375	0.7466666667
Run 8	243.9375	0.828125	0.65	0.84375	0.6266666667
Run 9	241.296875	0.84375	0.8033333333	0.828125	0.7733333333
Run 10	240.96875	0.828125	0.7766666667	0.828125	0.71
Avg	247.1765625	0.8546875	0.7226666667	0.8484375	0.694
Max	253.15625	0.890625	0.8466666667	0.875	0.8333333333
Min	240.96875	0.828125	0.57	0.828125	0.5666666667

Table 4. Dataset #2 - GRU results.

	Training Time (s)	Testing Time of Training Set (s)	Training Accuracy	Testing Time of Test Set (s)	Testing Accuracy
Run 1	254.484375	1.03125	0.7833333333	1.046875	0.8033333333
Run 2	256.609375	0.984375	0.8266666667	0.984375	0.8066666667
Run 3	268.453125	1.015625	0.9333333333	1.0625	0.8766666667
Run 4	269.171875	1.03125	0.6733333333	1	0.6633333333
Run 5	266.578125	1.25	0.7533333333	1.328125	0.7133333333
Run 6	269.484375	0.984375	0.8233333333	1.078125	0.82
Run 7	266.171875	1.046875	0.7133333333	1.078125	0.7066666667
Run 8	292.21875	1.09375	0.81	1.09375	0.8033333333
Run 9	290.65625	1.125	0.79	1.125	0.7933333333
Run 10	291.859375	1.203125	0.6933333333	1.171875	0.67
Avg	272.56875	1.0765625	0.78	1.096875	0.7656666667
Max	292.21875	1.25	0.9333333333	1.328125	0.8766666667
Min	254.484375	0.984375	0.6733333333	0.984375	0.6633333333