Dual Memory Neural Computer for Asynchronous Two-view Sequential Learning

Mohamed Amn and Roy A
{mamn2, ---}@illinois.edu

Group ID: 222
Paper ID: 150, Difficulty: hard
Presentation link: https://express.adobe.com/video/no7WmHR3BlxO7
Code link: https://github.com/mamn2/DNMC_Augmentation

1 Introduction

Multi-view learning is a sub-field of machine learning gaining a lot of attention recently for its strong performance and ability to integrate data from different sets. Typical examples are images of a 3D object being taken from different viewpoints - or 'views,' as they are referred to. Multi-view sequential learning refers to a multi-view paradigm in which each view forms part of a sequence. Examples are videos consisting of both video and audio streams, or videos of 3D objects from different angles, with a defined sequence. Typically research in this area has been focused on combining the views in synchronous data, however there are form of sequential data in which data within views do not have timestamp data - namely EMRs. In an EMR, we have information about each hospital visit such as the different drugs given, procedures, and patient diagnoses, but we don't necessarily know in what order these things happened, hence the asynchronously. The data in this case has three types of view interfaces: intra-view data such as the different ICD9 codes we find inside a prescription view, inter-view (early) such as the relationship between a diagnosis and procedure within a visit, and inter-view (late), which represents a mapping between the input views and output views.

The paper (Le et al., 2018b) presents a memory-augmented neural network aiming to model interactions between asynchronous sequential views while retaining long-term dependencies which can be quite important for predictions. It demonstrates a high level of efficacy on the MIMIC-III dataset compared to more traditional approaches such as Dual LSTM.

2 Scope of reproducibility

In order to test the claims from the paper, we test the metrics that were being claimed on the EHR data in comparison to Dual LSTM and WLAS using both the Diagnosis and Procedure views.

Specifically that between $DMNC_l$ and $DMNC_e$, the model provided better metrics across the board for AUC, F1, P@1, P@2, and P@5

2.1 Addressed claims from the original paper

To be more numerical, we will test that it was able to achieve an AUC, F1, P@1, P@2, and P@5 of 87.4, 73.2, 92.4, 88.9, and 82.6 respectively for $DMNC_l$ and 87.6, 73.4, 92.1, 89.9, and 82.5 respectively for $DMNC_e$. This is claimed in comparison to 85.8, 72.1, 91.6, 86.8, and 80.5 respectively for *Dual LSTM*, and 86.6, 72.5, 91.9, 88.1, and 80.9 for *WLAS*. Obviously the numbers don't have to be exact, but since we are provided with all the hyperparameters used in the paper, we will try to get results as close to those stated in the paper.

3 Methodology

Fortunately for our case, the author did provide code and some instructions to run it through their base repository. However, it was necessary to preprocess the *MIMIC-III* data into a compressed form for Pickle to be able to effectively read it before running the provided model. This was very convenient, as it gave us access to all the complex and large number of hyperparameters that are needed to replicate the results.

3.1 Model descriptions

The authors created a novel architecture they call *DMNC*, or Deep Memory Neural Computer. It consists of 2 encoders and 1 decoder, all interacting with 2 memory modules. Each memory module has temporal linkage and dynamic allocation. Each controller has its own embedding matrix, projecting a one-hot encoded representation of events in a d-dimensional space. Each encoder transforms

embedding vectors to h-dimensional hidden vectors. Since the input views are sequential in this model, the authors used *Dual LSTM* as the core of the encoders. This is the base architecture. The authors make two models from this architecture which differ in the way they treat the inter-view interactions. In one model, late-fusion memories are used, while in the other early-fusion memories are used.

In the late-fusion architecture, only late inter-view interactions are modeled. The keys produced by the read functions are used to address corresponding memory, ensuring the dynamics of computation in one view do not interfere with other views, only storing in-view contents in view-specific memories.

The early fusion architectures tend to be better when the correlation between the views is particularly strong. In this architecture, both memories share the same addressing space rather than keeping it separate. The read vectors for one encoder can come from either memory, meaning the value is dependent on the contents in both. The memories are also both updated by the encoders, while in the late-fusion model the memories are independent.

3.2 Data descriptions

We used the same dataset as the authors, *MIMIC-III*. This dataset consists of EHR records for over 40,000 patients who stayed in critical care units at Beth Israel Deaconess Medical Center covering a span between 2001-2012. The dataset is freely available online and has been anonymized so as not to reveal any sensitive information. For example, dates have been shifted so that birth dates and other personal information cannot be used to identify patients. It is worth noting, however, that we did not use all 40,000 patients to train our dataset. Because of the sequential nature of this model, all patients who were only in the hospital for a singular were automatically removed. This leaves us with data from 7,537 patients who had visited the unit multiple times over this time period.

3.3 Hyperparameters

All the hyperparameters were provided to us by the codebase which can be found at https://github.com/thaihungle/DMNC. This greatly simplified the process for us as there are quite a large number of hyperparameters throughout the various modules in this model. We used the exact same hyperparameters to get as close as possible to reproducing the exact results in the paper.

3.4 Implementation

The majority of the code was taken from the original repository linked from the paper itself. The implementation on our end mostly involved developing a pre-processing portion to encode our data in Google Drive via Pickle and changing the code to meet our environment specifications, such as adding GPU acceleration capability. This involved encoding information about each hospital visit which is then decoded and processed patient-wise. The code for this is available in the Jupyter Notebooks in the Github repository.

3.5 Computational requirements

The computational requirements were very large for the full dataset. While the demo dataset only took an hour or so to run as there were only 14 patients to fit the criteria, the full dataset took on average 8-12 CPU hours and a continuous 2 GB memory each. GPU acceleration actually slowed the process in our case and took about 12 GPU hours. We did not try using a TPU, which perhaps could have made the computation a little bit easier.

Due to the extreme computational requirements, we chose to run the programs in parallel on Google Colab servers. Unfortunately many of these programs crashed after running for 4-5 hours. This meant that some of the additional results that we wanted to include are not fully available. In order to remedy this we show the metrics at different checkpoints and compare them to the full set. More about this in Section 4.3.

4 Results

Due to the extreme computational requirements of our program, as well as the fact that the remote servers we ran on frequently crashed, we only have data available for the first 21,000 iterations so far. The remaining 21,000 iterations will be available on Github by the time you're reading this. Until then, let's take a look at the results halfway through.

As we can see, each of the models comes fairly close to the full 42,001 iteration run, and they are still learning, meaning it is likely that the results in the paper are reproducible.

We can also see very incremental improvements using the *DMNC* models in comparison to *LSTM* and *WLAS*, as expected.

Model	AUC	F1	P@1
Dual LSTM	85.4	71.4	90.6
WLAS	86.6	72.5	91.9
$DMNC_l$	87.4	73.2	**92.4**
$DMNC_e$	**87.6**	**73.4**	92.4

Table 1: Original Results (at 42,001 iterations)

4.1 Result

Table 1 summarizes the results presented in the original paper utilizing the full MIMIC-III dataset along with the hyper-parameter of 42,001 iterations (Le et al., 2018a).

Model	AUC	F1	P@1
Dual LSTM	93.4	65.1	58.1
WLAS	93.2	65.3	55.1
$DMNC_l$	**91.3**	66.5	56.8
$DMNC_e$	90.5	**66.8**	**58.8**

Table 2: Our Results (at 21,000 iterations)

Table 2 summarizes are results using Google Colab Pro, we compared the quality of the models with the full MIMIC-III dataset with a hyper-parameter of 21,000 iterations

4.2 Additional results not present in the original paper

An additional experiment we performed was reducing the size of the dataset. Since the computational requirements of this project were extremely high. We decided to try and run the models with only 1,000 patients instead of 7,537 patients. Unfortunately due to many sporadic server crashes, we were not able to run it through all 42,000 iterations. In order to get a glimpse into how the model was affected, we can instead look at how they compare at various checkpoint iterations. Here are the results of that:

Model	AUC	F1	P@1
Dual LSTM	94.7	66.6	58.5
WLAS	**94.9**	66.8	60.5
$DMNC_l$	94.1	66.7	57.8
$DMNC_e$	**94.9**	**68.2**	**69.1**

Table 3: Our Results (at 12,000 iterations)

5 Discussion

Give your judgement on if you feel the evidence you got from running the code supports the claims of the paper. Discuss the strengths and weaknesses of your approach – perhaps you didn't have time to run all the experiments, or perhaps you did additional experiments that further strengthened the claims in the paper.

The results from the paper were promising, showing slightly incremental increases over solutions like *WLAS* and *Dual LSTM*, however it is worth noting that the computational requirements for this model are about 3-4x as long. At the end of the day, there really is no preferred method, as all produce very close results and have their own trade-offs. If computational power is not really an issue, then *DMNC* is one of the best solutions available. However if ease of use and computational power is important, the practicality of models like *Dual LSTM* are still considerable better. The presentation reviews the replication efforts

5.1 What was easy

Understanding the model was fairly straightforward. We found that the authors presented the concepts in a very comprehensible manner. The analogies used were an excellent help in understanding the concept and no extremely technical terms specific to the domain matter went unexplained to any reasonable extent.

5.2 What was difficult

Running the code itself was not necessarily difficult, but very tedious. The code often crashed the host server and it made producing results very difficult and lengthy.

5.3 Recommendations for reproducibility

If we had to do it again, we would suggest using reliable remote Metal as a Service technologies or running locally on a powerful machine.

6 Acknowledgements

We are grateful for the comprehensive review of deep learning in health care and would like to thank Prof. Jimeng Sun and team. We found that the upcoming textbook was vital to our understanding of the material (Xiao and Sun, 2021).

References

H. Le, T. Tran, and S. Venkatesh. 2018a. Source code for dual memory neural computer. https://github.com/thaihungle/DMNC.

Hung Le, Truyen Tran, and Svetha Venkatesh. 2018b. Dual memory neural computer for asynchronous two-view sequential learning.

C. Xiao and J. Sun. 2021. *Introduction to Deep Learning for Healthcare*. Springer International Publishing AG.