Written document of about 5 pages , summary of the most important findings of the work undertaken.

Things learned during These past weeks

What I’ve learned from week to week

* First week was collecting data – easiest part of the assignment
* Searching through google scholar for similar type projects (release data and cites)
* While searching for baseline you’d get an understanding of the most commonly accepted ways of doing things (Example , sliding window , splitting data into training validation and test set . A model for each in particular crypto currency)
* Picking baseline and simulating their model while understanding their reasoning behind (while noticing their flaws). (Lstm and Gru)
* Creating my first tcn network and making sure the dilations and padding produces the right model description .
* While studying an in particular network it was hard for me to picture it without knowing exactly how it worked . Most of these networks took a long time to comprehend leading to many sketches , give examples .
* Did I get similar results to baseline following their metrics
* If not , why do you think ?
* What hyperparameters did I change , what impact did they have
* What reults I was getting for my baseline
* How I improved upon my tcn basline.

Questions During Meetings for Clarification

* Usually multivariate data does better than univariate as there is more data
* There is no standard value for the typical window length – it purely depends on your data , so trial and error.
* Rolling window technique was new to me
* Learned to find a baseline model , which would be a good metric to judge your work on .
* Learned that there could be possible leakage in published articles online , so you have to take things with a pinch of salt.
* Max-pooling layers don’t minimize the image it takes the most prominent features using the least possible neurons
* How can we tell long term behaviour from short term behaviour , empirical evaluation rather than mathematical (example is something show more a consistent trend).

**A Final Year Research Project On:**

**Predicting Price Fluctuations of Cryptocurrencies Using A Temporal Convolutional Network**

**Presented To**

**The School Of Computer Science And Information Technology.**

**University College Cork**

**Project Student : Jack Featherstone**

**Project Supervisor : Andrea Visentin**

Picking Baseline Model

The first course of action that had to be taken was establishing a baseline model.

After an appropriate amount of reading and enquiring on numerous articles, we decided on

“A Deep Learning-based Cryptocurrency Price Prediction Scheme for Financial Institutions”.

The reasons behind picking this in particular model was that it was recently published - 04/06/2020, it had been moderately cited and its subject matter suited our project description.

As a little time had passed since the publishing of this article, we knew that there would be more cryptocurrency data available to use for training. This could only have been deemed a good thing as the more training data available the less likely of overfitting the model. This meant there was the possibility of improving the baseline model.

The reasons behind picking a baseline initially was that we could directly compare my findings with said baseline and also get some practice building their models from this article.

A benefit that I found in reading similar projects was that you would learn the accepted standard for data preparation with time series data. For example, a common occurrence across every article I read was the same manipulation to the data (rolling window/ normalizing the data).

Replicating The Baseline’s Model

Reproducing the baseline’s model was a relatively simple task, the source article provided an in-depth description and had numerous diagrams of their model, making it a rather simple job to replicate.

However, the training process for this baseline model seemed to be too much of a challenge for my local machine to handle.

The estimated time it would have taken to run one epoch was nine minutes, as you can imagine if you have quite a large number of epochs, like we did, then this training process seemed like quite a frivolous one. One would be completely deterred from testing out different values for all the hyper-parameters as a result of the time it would take.

Google Cloud

At this point in the project, I was quite disheartened as it seemed completely unfeasible to run on my local machine, purely because of the time it would take.

The solution? Creating an instance on Google Cloud.

This means that a virtual machine was doing all the heavy computation in the cloud.

The process of creating an instance on Google Cloud was rather easy, although saying this, at first, I was not getting the results I had hoped. Rather than getting nine minutes per epoch, I was now getting twelve minutes per epoch, far from ideal!

After a large amount of research and a great deal of help from some of Andrea’s students, the problem was not with Google Cloud, rather a testament to my code.

My training runs were not making the most out of the GPU supplied by the virtual machine.

Increasing the batch size, and also getting rid of the data generators increased the usage of the GPU. By calculating the batches in parallel in the GPU memory, this reduced the communication overhead in the CPU. However, it was about finding the balance because I didn’t want the batch sizes too big so that the model did not converge to the best it could.

The specification of my model are as below:

GPU: NVIDIA Tesla T4

CPU: 8, 30gb memory

Pre – Packaged with: TensorFlow2.8/Keras.CUDA11.3.GPU

Boot Disk Image:  Debian GNU/Linux 10 (buster) with 200gb (standard persistent disk).

The NVIDIA Tesla T4 was picked because it was specifically designed for high performance computing and deep learning training. This GPU believe it or not, was the least powerful GPU on offer. The option was there to upgrade the hardware including the GPU in the virtual machine, although I did not find it necessary as I did not have an extremely complex model, nor an enormous amount of data.

My reasoning behind picking the remaining specification is because this is what was recommended on the deep learning, pre-built instance. This pre-built instance was extremely helpful as I did not have to look for all the necessary software myself as it came pre-loaded with all the necessary packages out of the box.