Cryptocurrency Price Prediction Using Attention Mechanism & Transformer Architecture

Shayan Alipour
Deep Learning 2020 - Prof. Rodolà
Sapienza Università di Roma
`alipour.1889595@studenti.uniroma1.it`

Abstract

This project addresses the issue of cryptocurrency price movements using deep learning methods. Our specific goal was to classify the up and downs movements after each window of 30 minutes period. We trained convolutional neural network (CNN), long short-term memory (LSTM), and Transformer for this task. The goal we pursue in this project was to inspect the performance of the mentioned models on price forecasting and recognize unanticipated challenges in working with each of them. The best F1-score we garnered was 61.42 with a multilayer LSTM.

1 Data Preprocessing

We used four different cryptocurrencies datasets for Bitcoin (BTC), Ethereum (ETH), Litecoin(LTC), and Bitcoin Cash (BCH) in which each of them has 92204 records and the following structure:
Time, Low, High, Open, Close, Volume

We selected the Close and Volume of each of those four datasets and joined them based on the Time column because for sure there is a correlation between various cryptocurrencies price movements. We calculated and added 8 different technical indicators to the mentioned dataset in order to observe the output and performance in each case.

The technical indicators we used are 7 and 21 minutes moving average, 12 and 26 minutes exponential moving average, MACD, 20 minutes standard deviation, upper and lower bands. Fig 1 shows 6 hours of the Litecoin's close price with some of the technical indicators.

At first, we followed the proposed approach in the proposal and trained the LSTM model based on Apple Inc stock market values, but we couldn't get

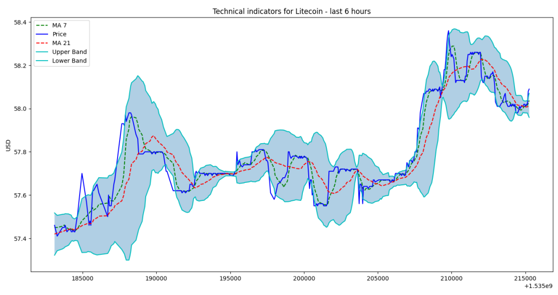

Figure 1: Litecoin dataset

acceptable results, because of lack of data records, 9950 logs only were available. Only daily stock prices are available online for free, we needed 1-min interval data, therefore, we switched to what we were able to find free of charge which was the Cryptocurrency market status dataset. We didn't mention Apple Inc.'s procedure in this report, though we submitted its notebook as a proof of work to show our efforts to deal with what we proposed in the first draft.

1.1 Creating Sequences

We considered a 30-minute window and added the next record close price as a future price that we deem to predict. In other words, our sequence length is 30 and our forecast period is 1. The task we have done is a classification problem, we added another column, in addition to the Future, named Label, If the prices increases we put 1 otherwise 0 in the Label column.

After creating the label, we had to build the sequences, for that purpose, we pick 30 records and take the last records' label in each sequence. Before that we dropped the future feature because we only needed it to create labels, then we applied scaling to the dataset features. After creating sequences, it was important to have a balanced dataset, therefore we checked how many up and downs we had, considered the lowest occurrence, and cut the other category. In the end, we shuffled all the sequences to have better results.

2 Methods

2.1 LSTM

Figure 2 shows the architecture of the model that had the best results among other LSTMs (Hochreiter and Schmidhuber, 1997). The model consists of two LSTM layers with 128 and 64 hidden units and a sequential layer for input projection consist of two linear layers and for the output classification we used a sequential layer which contain three linear layers. We used ReLu activation for the first two dense layer and a linear activation for the final dense in output sequential. We tried several hidden units but we only showed the best one in this section, some other results presented in the Section 3 under 'Models Result' title.

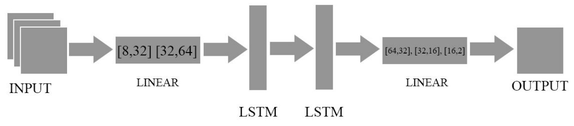

Figure 2: LSTM Architecture

2.2 CNN

We used 1D Convolutional Neural Networks in order to extract features from sequences data in this project. A 1D CNN is very effective for deriving features from a fixed-length segment of the overall dataset.

Three 1D CNN or according to PyTorch jargon, Conv1d used back to back in which we used different input and out channels, kernel size of 5 and stride of 1 for the first two conv1d and 2 for the last one. After convolution layers, we flattened the CNN output and feed it into two fully connected linear layers to finally get 2 items at the end. We used ReLu Activation for the first dense layer.

2.3 Transformer

It was the first time we worked with Transformers network, We put a lot of thoughts and research in order to understand how it works and figure out the proper structure for our task. What we reached so far was that we don't need to generate anything like the well-known translation task. For price prediction, we only need the encoder part of the transformer network.

According to the PyTorch tutorial, the source and output shape are the same and it is *(S, N, E)*. Where S is the input (source) sequence length, N is the batch size, and E is the number of features.

For mapping the price forecasting to the above-mentioned variables and garnering desire output, we added a randomly generated day price status to each 30-day window, in the end, we had a 31 Attentions. we choose the last attention, though we could select any of them, it was not like in LSTM that we have to select the last item. After choosing which attention we're going to work with, we respected that selection during training and testing phrases.

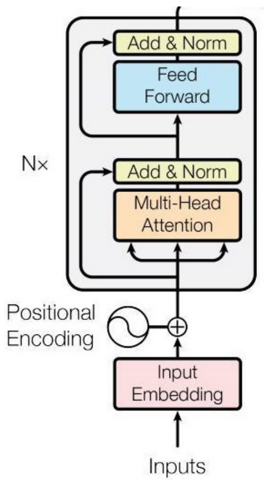

Figure 3: Transformer Encoder Architecture

We implemented Positional Encoder as an embedding layer based on PyTorch Documentation (PyTorch), which was bottomed on the original paper titled 'Attention is All you need' (Vaswani et al., 2017), in the paper they used Sine and Cosine functions to generate the vector.

Same as previous models, we added the input projection and output classifier layer. We used various head numbers in the multi-head attention models and sub-encoder-layers in the encoder in addition to other parameter tunings.

Model	Features	Dropout	Variant	Train F1	Test F1
LSTM	8	0.2	[128, 128, 128]+BiLSTM	60.51	59.23
LSTM	8	0.2	[128, 128, 128]+30epochs	63.37	57.80
LSTM	16	0.2	[128, 64]	62.41	59.37
LSTM	8	0.2	[128, 64]	62.59	61.42
LSTM	8	0	[64]+BiLSTM	61.41	60.12
LSTM	8	0.3	[32, 64]+BiLSTM	60.43	59.65
CNN	16	0	[64, 128, 256]	66.86	53.66
CNN	16	0.2	[64, 128, 256]	65.37	51.75
CNN	16	0.2	[64, 128, 256]+50epochs	87.88	53.40
Trans	8	0	4heads+6layers+20epochs	61.49	58.54
Trans	16	0.2	4heads+6layers+20epochs	56.62	56.16

Table 1: Litecoin Price Classification Results

3 Experiments/Results/Discussion

3.1 Evaluation and Hyperparameters

For hyperparameters, we conducted several random hand-tuning such as optimizer learning rate, dropout rate, specific parameters for each approach, and loss function behavior. We use Cross-entropy loss which is commonly applied as a loss function for classification problems.

Although we added technical indicators to the original dataset to have more features, the predictions resulted for dataset with technical indicators are weaker.

We classified the predictions into the confusion matrix, after training models for 10-20-50 epochs. Adam Optimizer is used to minimize the loss function during back-propagation(Kingma and Adam, 2014). Adam proved to be one of the best choices for deep learning and has benefited from previous optimizers. Different learning rates have been tested, the best one was 1e-3, though we used 5e-4 for transformer models so that they learn slower. we trained the tests with a batch size of 64.

3.2 Models Result

Table 1 shows just a portion of our conducted hand hyperparameter tunning. In the above table, for LSTM models the tuples refer to the hidden size of LSTM layers in which the length shows the number of used layers, and the BiLSTM indicated that whether the LSTMs are Bidirectional (Graves et al., 2005) or not, For the CNN models the tuple refers to Conv1d output channels. Last but not least, for Transformer models the 'variant' column refers to the number of attention heads and the encoder layers. The default epoch value is 10 if we trained them more, we mentioned it in the table.

The best result, achieved with LSTM model with two layers and 8 features. We observe by training more epochs(less than 50), LSTM and Transformer models won't surpass F1 score of 70 for trainset, however, the CNN showed different results.

3.3 Models Performance

Figure 4, 5, and 6 show the heat map visualization of the confusion matrix for LSTM, CNN, and Transformer models models respectively. The accuracy of LSTM model outperforms Transformer model in predicting label '0', however for label '1', Transformer shows higher accuracy. Figure 4 shows 62.04% accuracy in predecting **0** label, and 60.81% in **1** label.

4 Conclusion

In this project, we investigate out the performance of deep learning methods in cryptocurrency price prediction task by constructing multiple models with different variants. The challenges regarding implementing in LSTM, CNN and Transformer models lie not only on understanding how each of those works but also on how to prepare proper data for each model. The data preprocessing was pretty much the same for all of them, but the data preparation such as changing shapes, flattening, and etc needed to be done for feeding the data and using the results. We observed that overfitting in CNN models happens much faster than the other two.

We are glad that we were able to implement all three models and achieved acceptable results with this simple dataset. For achieving better results more detailed dataset is required and still there's a lot of room for improvement regarding working with Transformer architecture. For my task, the only reference I had was PyTorch documentation, lack of online content was obvious during the whole developing process.

References

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. pages 799–804.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Diederik P Kingma and Jimmy Ba. Adam. 2014. Adam: A method for stochastic optimization. *arXiv preprint*, arXiv:1412.6980.

PyTorch. Sequence-to-Sequence Modeling with nn.Transformer and Torchtext.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

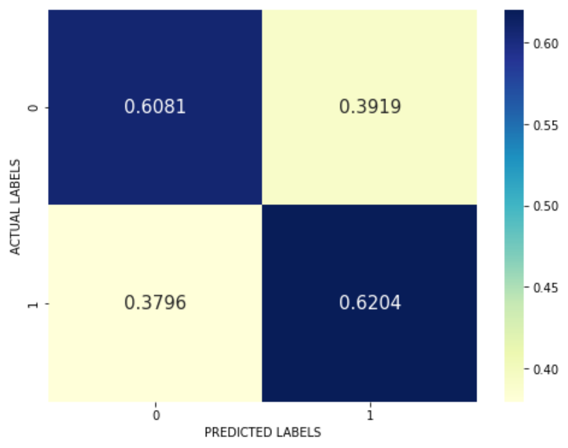

Figure 4: LSTM model confusion matrix heat map

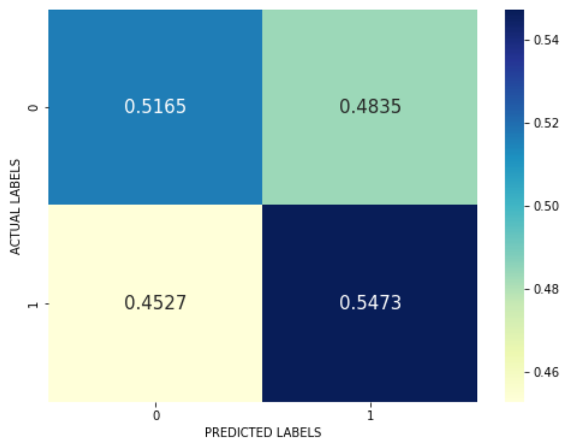

Figure 5: CNN model confusion matrix heat map

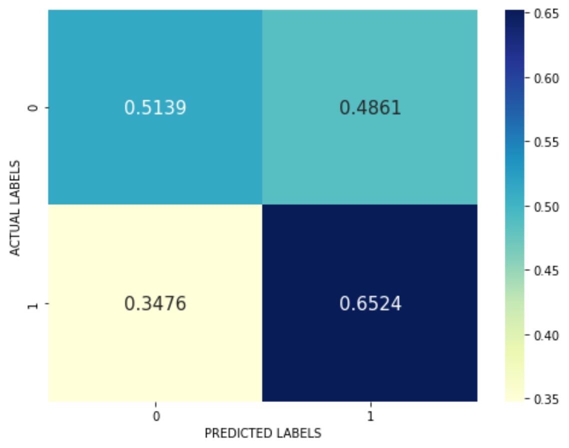

Figure 6: Transformer model confusion matrix heat map