Deep Convolutional Neural Networks for Limit Order Books

Akash Verma, Sarthak Sagar, Rahul Budhadev, Alec Madayan, Abdelmadjid Laouedj

October 9, 2021

1 Introduction

The original paper [1] develops a large-scale deep learning model to predict price movements from limit order book (LOB) data of cash equities. The architecture utilizes convolutional filters to capture the spatial structure of the limit order books as well as LSTM modules to capture longer time dependencies. We reference this model as DeepLOB for the remainder of this paper. We test our implementation of the model on FI-2010 dataset, which is used for bench marking against competing models. We also test it on our 3 month market data for liquid equities to see if the model is able to generalise over different time periods. We observe remarkably stable out-of-sample prediction accuracy for a variety of instruments. For comparision we implement another paper [2] which employs Transformers on LOB to predict movements. We refer to this model as TransLOB in the remainder of this paper. We are not able to replicate the results that were shown in the paper for TransLOB and feel that it could be a case of bumping up the accuracy of prediction by changing the computation of up, down and stationary categories. Finally, we attack DeepLOB with gradient based attacks to check the robustness of the model. Our conclusion is that, the model is able to generalize well and extract universal features from the data. This ability to extract robust features which translate well to other instruments is an important property of the model which has many other applications.

2 Data

We have used 2 datasets for this project:

FI-2010: This dataset is 10-day data from London Stock Exchange. The data has been used by a lot of models to benchmark performance on, thus we use it to check the performance of our implementation of DeepLOB and TransLOB architectures. The data is already labelled for 5 prediction horizions (k=10, 20, 30, 50, and 100).

3-month Data: We have used this larger dataset to confirm that the model generalizes well. We train and test our DeepLOB model on 3 months of LOB data for AMZN, GOOGL, FB, AAPL, and MSFT. Data contains 10 levels on each side and each level contains information on both price and volume Hence, we have a total of 40 features at each timestamp. We restricted the data to normal trading hours (9:30 AM to 4:00 PM). We used the following formulae to calculate the price change in the underlying price.

$$p(t) = \frac{p_{ask}(t) + p_{bid}(t)}{2}; m_-(t) = \frac{1}{k}\sum_{i=0}^{k} p_{t-i}; m_+(t) = \frac{1}{k}\sum_{i=1}^{k} p_{t+i}; r_k(t) = \frac{m_+(t) - m_-(t)}{m_-(t)}$$

We have labelled the data as up, down or stationary based on a threshold of $\alpha = 0.01\%$ on $r_k(t)$. The distribution of data is 1% up, 1% down, 98% stationary on short term horizon. The dataset becomes less unbalanced as the forecasting horizon increases.

3 Results of DeepLOB on FI-2010 dataset

3.1 Architecture

The model has 3 major blocks, the CNN block, the inception block and then the LSTM block. We go over each of the blocks in this section:

CNN Block : The main usage of the CNN layer is to condense the feature vector at each time step. The first CONV layer at each CNN block reduces the feature vector from 40(input) to 20 to 10 to 1. There are 3 convolution blocks at the end of which we have 16 channels of 82x1 matrix. Loss from 100 to 82, is due to padding='valid'. The 16 channels can be considered as 16 indicators derived from the LOB data. The channels don't have a lot of time dependence yet because the max kernel row length is (4, 1), thus taking the overlap of last 4 ticks.

Inception Block: The inception layer has been implemented to extract more channels (indicators) from the previously processed 16 channels. Each of the inception blocks work on the input parallelly to give 3 * 32 channels after concatenation. The first inception block has a kernel of (3, 1), while the second inception block has a kernel of (5, 1). The third inception block is different than the others because of the Maxpool layer. The final output has is 96 channels (indicators) with a timestep length of 82.

LSTM Block: The LSTM Layer converts 96 channels to 64 channels. The authors mention that this is primarily done to reduce the number of computation parameters as directly connecting the output to linear layer would have a lot of training parameters. The LSTM layers also helps to keep the temporal dynamics of the features intact. The LSTM output is connected to a fully connected neural network, and then a Softmax layer to categorize the signal as 'up', 'down', or 'stationary'.

3.2 Training process

We have used 7 days data for training and 3 days for testing. The data has 40 features including both prices and volumes by event. We didn't tune hyperparameters ourselves on the validation set: we decided to take the same as the authors even if the process is not clear (can also be computationally expensive). As previously mentioned, the labels are pre-computed in the original dataset (down, stationary, up).

3.3 Results

Model	Accuracy	Precision	Recall	F1	F1 CI 5%
Prediction Horizon k = 10					
NN	70.69	49.97	70.69	58.55	[58.29, 58.80]
CNN	62.62	51.79	62.62	50.95	[50.68, 51.21]
DeepLOB	80.24	78.41	80.24	78.13	[77.91, 78.34]
Prediction Horizon k = 50					
NN	50.28	49.13	50.28	49.21	[48.94, 49.47]
CNN	41.86	41.14	41.86	41.18	[40.92, 41.43]
DeepLOB	74.43	74.13	74.43	74.23	[74.01, 74.46]
Prediction Horizon k = 100					
NN	44.89	49.13	44.89	40.52	[40.26, 40.77]
CNN	38.02	39.42	38.02	37.98	[37.72, 38.23]
DeepLOB	71.97	72.40	71.91	72.03	[71.79, 72.26]

Table 1: Results of DeepLOB on FI-2010 dataset

We can draw three conclusions from this table. First, DeepLOB outperforms other baseline models, and this for all metrics. Second, as the forecasting time horizon increases, the F1 score decreases. It becomes naturally more challenging to forecast for a longer time horizon. Third, those results are in line with the paper [1]. We decided to go deeper into the analysis by looking at the performance by categories. This has not been clearly done for this dataset in the original paper [1].

3.4 Results by labels

Model	Precision	Recall	F1	F1 CI 5%
	Prediction Horizon k = 10			
Down	69.33	39.09	49.99	[49.32, 50.66]
Stationary	83.51	96.36	89.48	[89.28, 89.67]
Up	62.70	43.79	51.56	[50.86, 52.25]
	Prediction Horizon k = 50			
Down	68.42	66.05	67.21	[66.74, 67.68]
Stationary	80.53	84.82	82.62	[82.33, 82.91]
Up	68.34	64.08	66.14	[65.64, 66.63]
	Prediction Horizon k = 100			
Down	68.97	68.11	68.54	[68.12, 68.95]
Stationary	82.39	74.59	78.30	[77.93, 78.67]
Up	65.05	72.74	68.68	[68.24, 69.12]

Table 2: Results by label of DeepLOB on FI-2010 dataset

We would like to enhance two points. First, we have seen that, on average, the model performs better for short term horizon. However when looking at the labels, in short term horizon, the model seems to suffer in forecasting buy and sell signals as the F1 score points out: this element is not described in the paper. The quality of buy and sell signals actually increases with the prediction horizon, probably because the data are less unbalanced. Second, it could be interesting to train on a larger dataset to see if the model generalises well.

4 Results of DeepLOB on larger dataset

4.1 Training process

We have kept the architecture same as described above. Our objective was to see how the model performs on different (larger) dataset and if it generalises well. We are training on 1 month of data and testing on the same test set as for FI-2010 dataset. The purpose of choosing this test set was first to be computationally less costly, second to see if generalisation is also good within stocks.

4.2 Results

The results we got are as described in Table 3.

Model	Accuracy	Precision	Recall	F1	F1 CI 5%
		Prediction Horizon k = 100			
DeepLOB FI-2010	71.97	72.40	71.91	72.03	[71.79, 72.26]
DeepLOB Larger Dataset	76.17	62.22	76.17	66.11	[65.92, 66.29]

Table 3: Results of DeepLOB on large dataset

First, generalisation from smaller to larger dataset is somewhat good, and this for the most challenging time horizon. Second, the composition of stocks in the largest dataset is different from the ones in the test set. However, the models do not suffer from that meaning it is performing well in one-shot learning (forecast a label with no prior examples in the train set). Last, those results are in line with the paper [1].

5 Results of TransLOB on FI-2010 Dataset

Transformers have been used in NLP to overcome the shortcoming of LSTM by using the hidden state of each timestep for the output. They also reduce computation time when used as CNN + attention model. The TransLOB Model uses exact same features as DeepLOB, but employs a transformer module instead of an Inception + LSTM module. Transformer module consists of a Multihead attention block + fully connected NN. The output returned is the same shape as of input.

5.1 Architecture

The first 5 layers are 1-D CNNs which reduce the initial input of 100x40 to 100x14. As the CNNs are 1-D the 40-dimensional feature vector is considered as a channel. Kernel size is chosen as (2) for each layer, with causal padding. The Position Encoding layer concatenates another channel, which is indicative of the timestep position to make input to transformer block 100x15. The Transformer block has 3 self-attention blocks. The dimension of Query, Key and Value filter is 15x5 for each of the block. After the computation from each block, the values are concatenated(100x15) and the next steps are- residual connection, normalization, multi layer perceptron (hidden=60) to give 100x15 output size. After the transformation block the final steps are the flatten the features, run through a fully connected neural network to reduce the size to 3, which goes into a SoftMax layer to categorize the output.

5.2 Results

Below are the results we got for our implementation of the TransLOB model.

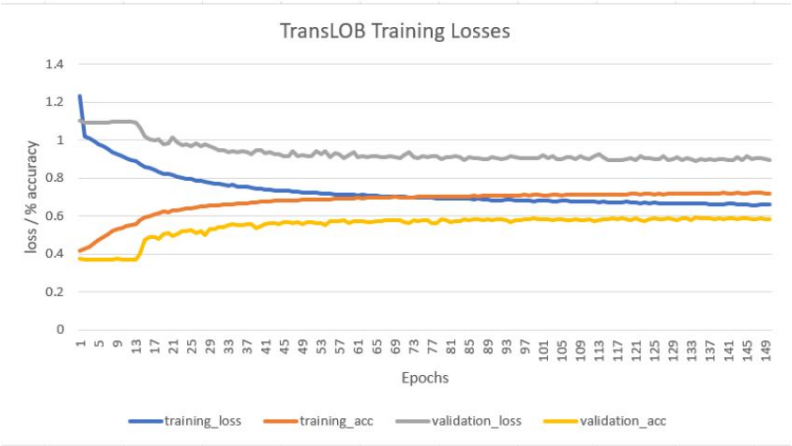


Figure 1: TransLOB training losses

Model	Accuracy	Precision	Recall	F1	F1 CI 5%
		Prediction Horizon k = 100			
NN	44.89	49.13	44.89	40.52	[40.26, 40.77]
CNN	38.02	39.42	38.02	37.98	[37.72, 38.23]
DeepLOB	71.97	72.40	71.91	72.03	[71.79, 72.26]
TransLOB	57.54	58.62	56.32	57.84	[57.64, 58.04]

Table 4: Results of TransLOB on FI-2010 Dataset

We note that our results differ from the paper which claims a 87.66% accuracy. The author chose a different way to compute the labels, changing both the signal computation and threshold α for the data, which can explain the higher

accuracy. But as the dataset used is FI-2010, which has been used to benchmark multiple models, we didn't try to change the labelling to match the papers results. We found another independent implementation of the paper which got 48% accuracy.

6 Robustness of models

In this section, we would like to test the robustness of the models. Especially, we will focus on the DeepLOB model with the FI-2010 dataset. One way to do experiment is by considering adversarial attacks. The idea is to perturb the initial images by a noise and see how the model reacts: we are looking at perturbation that could misclassify an image without being sizeable by humans. However, it could be challenging to choose a good attack for our purpose.

6.1 Adversarial attacks

We introduce two type of attacks. They are white-box attacks, meaning that we know the structure of the models when attacking. We note x the initial image, y the labels, ϵ and α hyperparameters for the pertubation and $L(\theta, x, y)$ the loss function. The first attack is called FGSM (Fast Gradient Sign Method). The idea is to perturb an image only one time using the sign of the gradients. The gradient gives the direction of greatest increase of the objective function. Therefore, perturb the image using the sign of the gradient makes sense to "stretch" the loss function.

$$FGSM(x) = x + \epsilon sign(\nabla_x L(\theta, x, y))$$

Another type of attack, more rude, is called PGD (Projected Gradient Descent). The idea here is to attack multiple times the images. Remember that the type of attacks should make sense, meaning a good attack should not been seen by humans. In this sense, we cannot indefinitely attack the images.

$$PGD : x^{t+1} = \Pi_{x+\delta}(x^t + \alpha sign(\nabla_x L(\theta, x, y)))$$

One way for a model to resist from those attacks is using adversarial defense.

6.2 Adversarial defense

The idea of adversarial defense is straightforward: for a model to be more robust against attacks, we feed it both original images and perturbed images in order to train. The loss function will then be divided between two components: a loss computed on original images and a loss computed using perturbed images. We decide in our project to weight both loss equaly (α is 0.5).

$$\hat{L}(\theta, x, y) = \alpha L(\theta, x, y) + (1 - \alpha) L(\theta, x + \epsilon sign(\nabla_x L(\theta, x, y))$$

In the appendix, you will find an example of a prototypical learning loss that helps the model to resist from attacks by creating distinct clusters in the latent space.

6.3 How to adapt the attack for our purpose?

Finding a way to adapt/create an attack for our purpose was very challenging and this for multiple reasons. The first one is that there is not much literature for attacks applied to finance. This is probably because CNNs are not extensively used in finance (except when developing NLP algorithms). Usually, the image is attacked globally However, we thought it would be interesting to apply the attack locally, based on random generated number. Therefore, we applied Fast Gradient Sign Method (attack on some local regions chosen randomly.

6.4 Results

Here you will find the results on 3 models. The first column refers to the original accuracy computed before. The second column refers to the accuracy of each model after the model being attack. The last column is the accuracy of the models trained using adversarial examples after being attack.

We can draw three conclusions from this table. First, DeepLOB model is relatively resistant to adversarial attack compared to baseline models. Second, using adversarial training, we are able to recover an important part of the accuracy.

Model	Accuracy Original Model	Accuracy with Adversarial Attack	Accuracy After Adversarial Training
NN	44.89	0	
CNN	38.02	0	
DeepLOB	71.97	24.65	55.91

Table 5: Results of DeepLOB on FI-2010 dataset following adversarial attack and defense, prediction horizon k = 100

However, some questions persists that we would have been willing to cover. Is the attack really appropriate and strong? How about a black box attack (attack without knowing the structure of the model)?

7 Perspectives

7.1 Literature and further research

You will find in the references section the literature we used. There are typically three areas we didn't experience and want to go deeper into. First, not considering white-box attacks but black-box attacks, meaning the attacks are independent of the structure of the models: [5], [7] . They are used in NLP and could make more sense here. Second, go deeper into the only paper we found dealing with both adversarial attacks and high frequency trading: [4]. Last, test the models not on metrics but using realistic trading strategies.

7.2 Recommendations

Prior to the project, we thought computational issue would be the main problem. Posterior to the project, it is not the case. The main problems were how did architectures have been chosen? Why there is so much difference in architectures between papers and github? Why the transformer architecture is working that bad relatively to what the paper is claiming? Why is there such a tiny parts of the papers dealing in testing the robustness of models? We then recommend to further investigate the issue with the **transformer performance** the intuition behind modeling should make sense for our problem. We also recommend to develop further attacks more appropriate for our problem we should think about some economic intuition behind this as well as testing **black-box attacks**.

References

[1] Zihao Zhang, Stefan Zohren, and Stephen Roberts
Deep Convolutional Neural Networks for Limit Order Books
Year: 2019
Accessible at: https://arxiv.org/pdf/1808.03668.pdf

[2] James Wallbriged
Transformers for limit order books
Year: 2020
Accessible at: https://arxiv.org/pdf/2003.00130.pdf

[3] M.T. Ribeiro, S. Singh, C. Guestrin
–*"Why should I trust you? Explaining the predictions of any classifier"*
Year: 2016
Accessible at: https://arxiv.org/pdf/1602.04938.pdf

[4] M Golblum, A Schwarzschild, A Patel, T Goldstein
Adversarial Attacks on Machine Learning Systems for High Frequency Trading
Year: 2020
Accessible at: https://arxiv.org/pdf/2002.09565.pdf

[5] A. Ilyas, L. Engstrom, A. Athalye, J. Lin
Black-box Adversarial Attacks with Limited Queries and Information
Year: 2018
Accessible at: https://arxiv.org/pdf/1804.08598.pdf

[6] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.- J. Hsieh
Zeroth order optimization based black-box attacks to deep neural networks without training substitute models
Year: 2017
Accessible at: http://doi.acm.org/10.1145/3128572.3140448

[7] N. Narodytska, S. P. Kasiviswanathan
Simple Black-Box Adversarial Perturbations for Deep Networks
Year: 2016
Accessible at: https://arxiv.org/pdf/1612.06299.pdf

[8] J. Snell, K. Swersky, R. Zemel
Prototypical Networks for Few-shot Learning
Year: 2017
Accessible at: https://arxiv.org/pdf/1703.05175.pdf

Appendices

A Technicals

Pipeline	Berkeley Cloud Computing
GPU	NVIDIA Tesla V100
Processor	Intel Xeon Platinium 8168 CPU @ 2.70GHz
RAM	1TB

Table 6: Technical Attributes

Model	Time to compute by epoch
DeepLOB FI-2010 Dataset	90 seconds
DeepLOB Larget Dataset	24 minutes
TransLOB FI-2010 Dataset	5 minutes

Table 7: Results of DeepLOB on FI-2010 dataset following adversarial attack and defense

Hyperparameters	Values
Batch size	32
Learning rate	0.001
Activation function	Leaky-ReLu

Table 8: Hyperparameters for DeepLOB

Hyperparameters	Values
Batch size	32
Adam Beta1	0.9
Adam Beta2	0.999
Learning rate	0.0001
Number of heads	3
Number of blocks	2
MLP activation	RELU
Dropout rate	0.1

Table 9: Hyperparameters for TransLOB

B Architectures

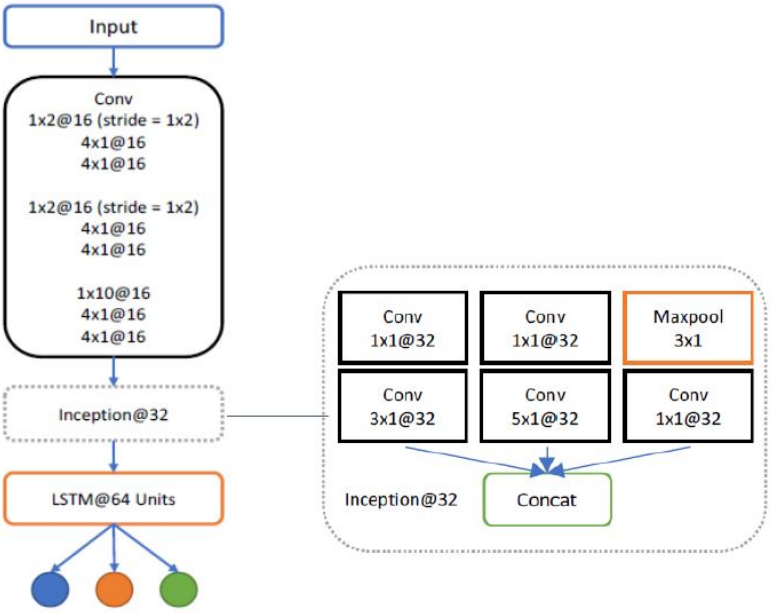


Figure 2: DeepLOB architecture

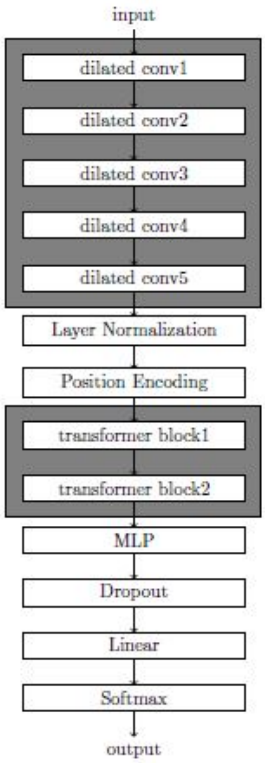
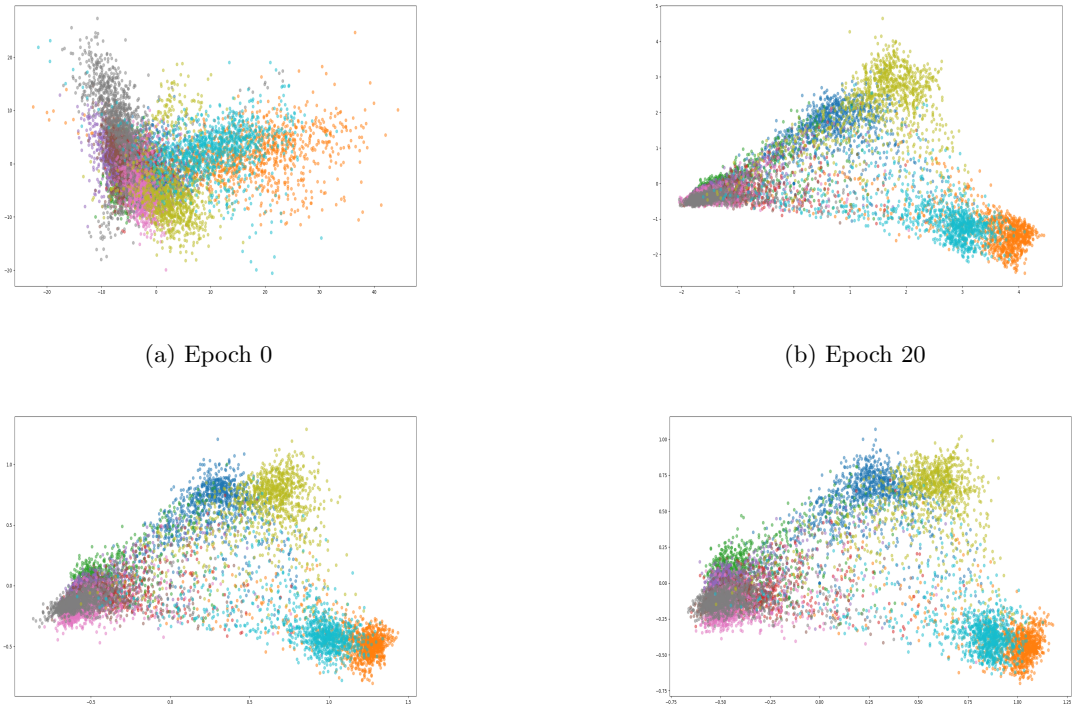


Figure 3: TransLOB architecture

C Adversarial attacks and defenses



(a) Epoch 0

(b) Epoch 20

(c) Epoch 50

(d) Epoch 70

Figure 4: Prototypical learning using a specific loss functions on CIFAR-10: clusters are created in the latent spaces which led the model to resist well to adversarial attacks