CSC413 Final Project

Arnaud Deza, Stephen Brade, Susan Sun

Abstract

Online deep learning (ODL) represents a class of deep learning algorithms that optimizes deep models for data streams. Sahoo et. al proposes Hedge Backpropagation (HBP) for optimizing models of this class [1]. We re-implement the results of this paper, apply the method to predicting the price movements of three stock market indexes over a 20 years, and compare the results of HBP to regular online gradient descent (OGD). The extension to stock data was effective but did not exhibit any clear advantages of HBP over OGD.

1 Introduction

In typical deep learning contexts, a dataset must be available in its entirety for training. However, data that arrives sequentially, is too large for storage, or exhibits concept drift may be unsuitable for these training methods. Online Deep Learning (ODL) is a sub-field that focuses on adapting the optimization process for deep models for data streams. One method created by Sahoo et. al is hedge backpropogation (HBP) which makes predictions that are robust against concept drift while being able to capture nuanced trends in deep layers.

In this work we will re-implement the results from [1] and apply the proposed algorithm, HBP, to a new domain. We chose three stock market indices because stock data arrives sequentially and experiences concept drift due to evolving market trends [2]. We will train models with HBP to predict whether prices will rise or fall and compare it's performance to a shallow network and a deep network using, both using online gradient descent (OGD).

2 Related Works

Existing methods for online deep learning with regular stochastic gradient descent typically still require some form of mini-batch training [1; 3]. Meanwhile, the HBP method developed by Sahoo et. al [1] is designed to be trained on data streams directly and is competitive to other learning methods.

Historical stock data typically includes the daily open, high, low, close, adjusted close prices, and volume. From these values, technical indicators that measure momentum, stochasticity, and other trends can be derived. Different models have used a combination of historical prices and technical indicators to predict whether stock price will increase or decrease. Qui and Song used an Artificial Neural Network (ANN) trained with vanilla SGD and thirteen technical indicators to predict the price movement of Japanese stock market indexes [4]. Fisher and Krauss adopted a LSTM network using a rolling window of stock returns to predict the price movement of the S&P500 over 23 years [5]. The LSTM model outperformed previous random forest and deep neural net (DNN) methods but struggled to generalize on data after the 2008 financial crisis. More recently, there have been attempts to represent a combination of stock prices and technical indicators as images and use Convolutional Neural Networks (CNN) to classify price movement [6; 7]. The CNN models reduce the generalization issues of the LSTM models but still require all data to be available in advance.

Existing performance in this problem space is relatively well established with many of these models attaining prediction accuracies of up to 90% [4; 8]. Instead of trying to improve the classification

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

accuracy, our primary goal for this extension is to explore whether ODL can be applied on data streams and adapt to changes in the market.

3 Methods

We consider classification problems in an online setting, by training a model, $\mathbf{F} : \mathcal{R}^D \rightarrow \mathcal{R}^C$, based on a sequence of training examples, $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}$, with binary labels, $y_i \in \{0, 1\}^C$, that arrive sequentially. A condensed version of the algorithm from [1] is reproduced below and further details are available in the original paper. In contrast to [1], we only focused on DNN models trained through OGD and its variants versus HBP on 5/7 datasets due to resource constraints.

Algorithm 2 NB: the below pseudocode is adapted from Sahoo et. al [1]

Input: learning rate parameter η, discount paramater $\beta \in (0, 1)$, smoothing parameter s
Output: Trained prediction function $\mathbf{F}(\mathbf{x})$ with parameters: α and $W^{(l)}, \theta^{(l)}\ \forall l \in \{1, \cdots, L\}$
Initialize the prediction function, a deep neural network, $\mathbf{F}(\mathbf{x})$ with L hidden layers and L+1 classifiers $\mathbf{f}^{(l)}, \forall l \in \{1, \cdots, L\}$ and $\alpha^{(l)} = \frac{1}{L+1}\ \forall l \in \{1, \cdots, L\}$

procedure ONLINE DEEP LEARNING WITH HBP
  for `t = 1,...,T` **do**
    Receive input instance: $x_t \in \mathcal{R}^d$
    Predict $\hat{y}_t = \mathbf{F}_t(x_t) = \sum_{l=0}^{L} \alpha_t^{(l)} \mathbf{f}_t^{(l)}$ according to ▷ weighted sum of classifiers
    $\mathbf{f}^{(l)} = \text{softmax}(\mathbf{h}^{(l)}\Theta^{(l)}), \quad \forall l \in \{0, \cdots, L\}$ ▷ L classifiers
    $\mathbf{h}^{(l)} = \sigma(W^{(l)}\mathbf{h}^{(l-1)}), \quad \forall l \in \{1, \cdots, L\}$ ▷ feature representation
    $\mathbf{h}^{(0)} = \mathbf{x}$
    Reveal the true value y_t
    Set $\mathcal{L}_t^{(l)} = \mathcal{L}(\mathbf{f}_t^{(l)}(\mathbf{x}_t), y_t), \forall l \in \{1, \cdots, L\}$ where $\mathcal{L}$ is the cross-entropy loss function
    Update $\Theta_{t+1}^{(l)} \forall l \in \{0, \cdots, L\}$ using $\Theta_{t+1}^{(l)} \leftarrow \Theta_t^{(l)} - \eta\alpha^{(l)}\nabla_{\Theta_t^{(l)}}\mathcal{L}(\mathbf{f}^{(l)}, y_t)$
    Update $W_{t+1}^{(l)} \forall l \in \{1, \cdots, L\}$ using $W_{t+1}^{(l)} \leftarrow W_t^{(l)} - \eta \sum_{j=l}^{L} \alpha^{(j)}\nabla_{W^{(l)}}\mathcal{L}(\mathbf{f}^{(j)}, y_t)$
    Update $\alpha_{t+1}^{(l)} \forall l \in \{0, \cdots, L\}$ using $\alpha_{t+1}^{(l)} \leftarrow \alpha_t^{(l)}\beta^{\mathcal{L}_t^{(l)}}$
    Smoothing $\alpha_{t+1}^{(l)} = max(\alpha_{t+1}^{(l)}, \frac{s}{L}), \forall l \in \{0, \cdots, L\}$
    Normalize $\alpha_{t+1}^{(l)} = \frac{\alpha_{t+1}^{(l)}}{Z_t}$ where $Z_t = \sum_{l=0}^{L} \alpha_{t+1}^{(l)}$
  end for
end procedure

For our extension, we used the Alpha Vantage API to obtain historical daily stock price data for three electronically traded funds (ETFs), SPY, QQQ, and DIA that track the stock market indexes S&P 500, NASDAQ, and Dow Jones Industrial Average respectively. Based on existing literature, we tested two methods of representing stock data. The first method involved calculating nine technical indicators with their formulas outlined in the appendix. The second method was a rolling window approach where w = the size of the sliding window and the closing prices of the previous w days are appended are as outlined in Equation (1). We constructed the input data for two different window sizes, 15 days and 30 days which is representative of 3 weeks and 6 weeks of trading.

$$x(t) = \frac{AC_{t-w}}{AC_{t-(w+1)}}, \ldots, \frac{AC_{t-1}}{AC_{t-(w+1)}} \quad (1) \qquad y(t) = \begin{cases} 1, & AC_t \geq AC_{t-1} \\ 0, & AC_t < AC_{t-1} \end{cases} \quad (2)$$

Conventional normalization methods such as min-max scaling would not be appropriate for our data because they embed information about the future price movements into prior data. Instead, we normalized values at each sample using the adjusted closing price of the day before the first day in the rolling window. Finally, we labeled the data as outlined in Equation (2).

Per dataset with each corresponding to a specific window size and ETF, we trained and optimized the hyperparameters for three models: an MLP with 20 layers trained using HBP, an MLP with 20 layers trained using OGD, and a single layer model that used OGD. Every model used a softmax output activation and had an output size of two, corresponding to the two possible labels described above. This output representation was consistent with the codebase from Sahoo et. al. Additionally, the MLP models used ReLU hidden activations.

To tune hyperparameters, we used a simple grid search approach and tuned the hidden layer size $\{10, 25, 50, 100\}$ and learning rate $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$. Per ETF, window size, and data representation, we selected the hyperparameter configuration which resulted in the highest accuracy at the final iteration for each of the three models.

4 Experiments

4.1 Reimplementation

The reimplemntation part of this project consisted of verifying the main results of [1], focusing on the performance of shallow and deep neural networks trained with HBP vs vanilla OGD and its variants. The specific datasets we used for training are the real world physics datasets, Higgs, Susy and Syn8 and the synthetic curated datasets, CD1 and CD2, in [1]. All of these datasets consist of classification problems with 2 classes and range between 4-5 million instances. We used the publicly available source code of the original authors, which required some minor modifications. The results of our reimplementation experiments can be found in table 1.

Table 1: Reimplementation results (Final error)

Method	Number of Layers	Higgs	Susy	Syn8	CD1	CD2
Online Gradient Descent (OGD)	2	29.41	20.28	39.71	43.58	42.11
Online Gradient Descent (OGD)	3	27.33	20.15	39.27	43.20	41.60
Online Gradient Descent (OGD)	4	26.81	20.12	39.15	43.12	41.54
Online Gradient Descent (OGD)	8	26.86	20.16	39.39	43.29	41.62
Online Gradient Descent (OGD)	16	27.39	20.44	40.28	49.33	43.38
Online Gradient Descent (OGD)	20	27.91	20.56	47.63	49.31	48.79
OGD + Momentum	20	27.41	**20.07**	39.55	43.68	42.20
OGD + Nesterov	20	26.78	**20.07**	39.82	46.61	46.22
Hedgeback Propagation	20	**26.26**	20.17	**38.94**	**42.83**	**41.22**

4.2 Sensitivity Analysis

We performed a sensitivity analysis for 2 model hyper parameters on the Higgs and CD1 dataset. In particular we investigated the effect of varying learning rate and hidden dimension size on model performance. In Figure 1, we can observe the model performance as a function these parameters. The range for the learning rate was chosen to demonstrate two extremes; very large or small update steps and its relationship with model performance for OGD and HBP. The range of the hidden dimension was chosen in order to allow for both very complicated (many parameters) and simpler (fewer parameters) models to be trained.

4.3 Extension

Included in Figure 2 are the results for the rolling window method in which HBP and the single layer OGD models tracked very closely, with the 20 layer OGD model converging very slowly and never achieving the same accuracy. The results for the models trained on the technical indicators are included in the Appendix D for completion – none of the models converged on this data and we expect it requires a more advanced understanding of financial indicators to tune and select appropriate features.

The results of the grid search for the rolling window representation are included in Appendix C. These show that 20 layers using HBP converges at least as fast as 1 layer with OGD, while achieving a very similar if not better accuracy. Additionally, 20 layers with HBP performs significantly better than 20 layers with OGD with respect to convergence rate and accuracy.

5 Discussion

The results shown in the reimplementation experiments are very similar to the results presented in [1], however there are some discrepancies likely due to varying Keras versions. We observe that the

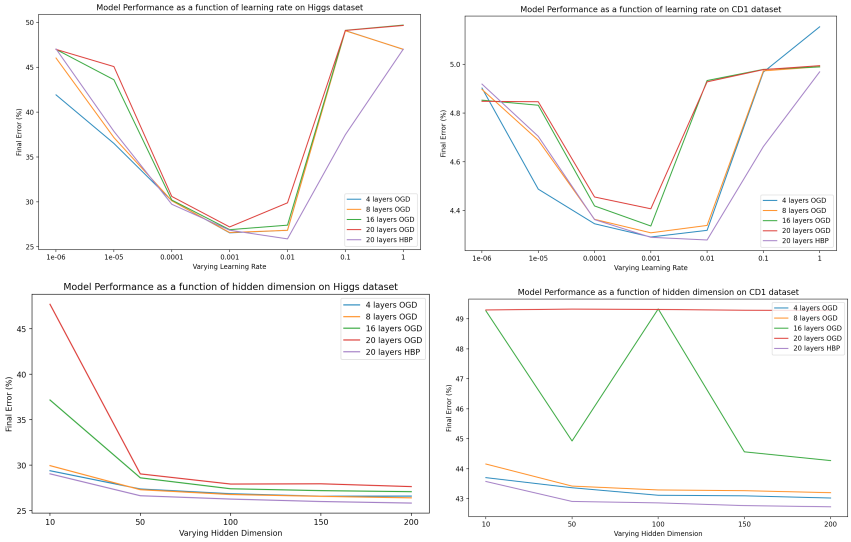

Figure 1: Sensitivity analysis

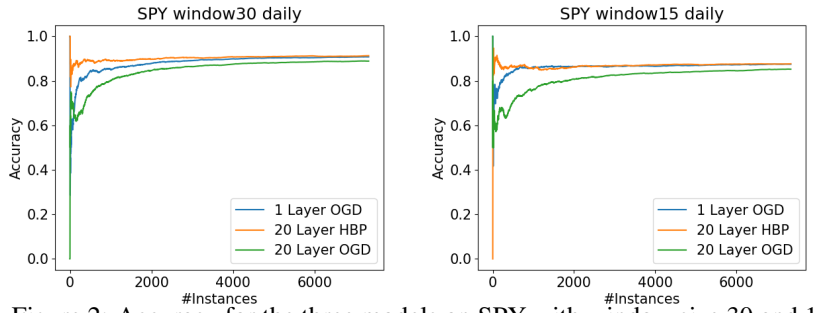

Figure 2: Accuracy for the three models on SPY with window size 30 and 15.

proposed algorithm outperforms all baselines on 4 out of the 5 datasets. The discrepancy between our reimplementation results is that OGD with momentum or nesterov performs better than HBP. However, the performance difference is very small and HBP's strength is clearly demonstrated.

The sensitivity analysis reinforces the strength of HBP . We observe that that when varying learning rate and hidden dimension size, HBP almost always produces the lowest error in comparison to vanilla OGD. From the learning rate experiments, we observe that OGD and HBP perform similarly for very small or large step sizes. This makes sense given the challenges learning algorithms have with non calibrated weight updates.

Applying HBP to stock price movement prediction shows some initial success as deep MLP trained with HBP outperforms the deep MLP trained with OGD. However, we cannot say that the deep HBP model gained any advantage from its depth as it performed roughly the same as a 1 layer OGD model. We expect this is due to the simplicity of the data – perhaps predicting price movements for multiple stocks would be more fitting for ODL.

6 Conclusion

In conclusion, we have demonstrated the strengths of HBP through the implementation of key experiments conducted in [1]. Due to the lack of sensitivity analysis in [1], we have demonstrated the overall robustness of the proposed HBP algorithm to model hyperparameters (learning rate and hidden dimension size). We observed that HBP remains competitive when varying these parameters.

The extension to stock data shows initial promise but shows that the selected task was too simple to make use of deep layers trained with HBP. Our feature construction and selection was constrained by our rudimentary understanding of financial markets. However, in the future this HBP algorithm could also be attempted with both time series price data and technical indicators which may produce

a stronger signal. Additionally, we could expand to other securities such as single stock or foreign currencies that are potentially more volatile than index ETFs.

References

[1] D. Sahoo, Q. Pham, J. Lu, and S. C. H. Hoi, "Online deep learning: Learning deep neural networks on the fly," 2017.

[2] B. Silva, N. Marques, and G. Panosso, "Applying neural networks for concept drift detection in financial markets," *CEUR Workshop Proceedings*, vol. 960, pp. 43–47, 01 2012.

[3] S.-W. Lee, C. yeon Lee, D. Kwak, J. Kim, J. Kim, and B.-T. Zhang, "Dual-memory deep learning architectures for lifelong learning of everyday human behaviors," in *IJCAI*, 2016.

[4] M. Qiu and Y. Song, "Predicting the direction of stock market index movement using an optimized artificial neural network model," *PLoS ONE*, vol. 11, 2016.

[5] T. G. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *Eur. J. Oper. Res.*, vol. 270, pp. 654–669, 2018.

[6] O. Sezer and M. Ozbayoglu, "Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach," *Applied Soft Computing*, vol. 70, 04 2018.

[7] X. Zhang, S. Liu, and X. Zheng, "Stock price movement prediction based on a deep factorization machine and the attention mechanism," 2021.

[8] L. Khaidem, S. Saha, and S. R. Dey, "Predicting the direction of stock market prices using random forest."

[9] "Investopedia technical indicator definitions and formulas."

A Contributions

Responsibilities were shared across members are outlined below.

Team Member	Overall responsibilities and contributions
Arnaud Deza	• Got source code working with minor Keras fixes. • Conducted reimplementation and sensitivity analysis experiments • Contributed to Methods, Experiments, Discussion and Conclusion
Stephen Brade	• Applied Sahoo et. al code to stock data • Completed grid search for all extensions • Contributed to Introduction, Discussion, Conclusion, and Abstract
Susan Sun	• Conducted background research on stock data prediction methods • Acquired, cleaned, and pre-processed stock data • Contributed to Abstract, Related Works, Methods, and Conclusion

B Technical Indicators Equations

Table 2: Formulas for various technical indicators [9]

Technical Indicator	Formula
Simple Moving Average (SMA)	$SMA = \frac{AC_1+AC_2+\cdots+AC_n}{n}$ n = number of days AC = Adjusted Close Price
Moving Average Convergence or Divergence (MACD)	$EMA_{12} - EMA_{26}$
Fast Stochastic Indicator (%K)	$(\frac{C-L_{14}}{H_{14}-L_{14}}) * 100$ C = The most recent closing price L_{14} = The lowest price in the 14 previous trading days H_{14} = The highest price in the 14 previous trading days
Slow Stochastic Indicator (%D)	3 day moving average of %K
Relative Strength Index (RSI)	$100 - \left(\frac{100}{1+\frac{\text{Average gain}}{\text{Average loss}}}\right)$
William's %R	$\frac{H_{14}-C}{H_{14}-L_{14}}$ C = The most recent closing price L_{14} = The lowest price in the 14 previous trading days H_{14} = The highest price in the 14 previous trading days
Percentage Price Oscillator (PPO)	$\frac{EMA_{12}-EMA_{26}}{EMA_{26}} 100$
Exponential Moving Average (EMA)	$EMA_t = (AC_t * (\frac{smoothing}{1+n})) + EMA_{t-1} * (1 - (\frac{smoothing}{1+n}))$ n = number of days AC = Adjusted Close Price smoothing = 2 (most commonly)

C Results of Grid Search for Rolling Window Datasets

Table 3: Results of grid search on DIA

Window Size	Model	Best Hidden Size	Best Learning Rate	Final Accuracy
15	20 Layers HBP	100	0.01	**0.880**
15	20 Layers OGD	50	0.01	0.852
15	1 Layer SGD	10	0.01	**0.880**
30	20 Layers HBP	25	0.01	0.900
30	20 Layers OGD	100	0.01	0.877
30	1 Layer SGD	25	0.01	**0.901**

Table 4: Results of grid search on QQQ

Window Size	Model	Best Hidden Size	Best Learning Rate	Final Accuracy
15	20 Layers HBP	50	0.01	0.867
15	20 Layers OGD	100	0.01	0.821
15	1 Layer SGD	100	0.001	**0.870**
30	20 Layers HBP	100	0.01	**0.908**
30	20 Layers OGD	100	0.01	0.876
30	1 Layer SGD	100	0.001	0.904

Table 5: Results of grid search on SPY

Window Size	Model	Best Hidden Size	Best Learning Rate	Final Accuracy
15	20 Layers HBP	25	0.01	**0.876**
15	20 Layers OGD	100	0.01	0.852
15	1 Layer SGD	25	0.01	0.875
30	20 Layers HBP	50	0.01	**0.913**
30	20 Layers OGD	100	0.01	0.889
30	1 Layer SGD	100	0.001	0.908

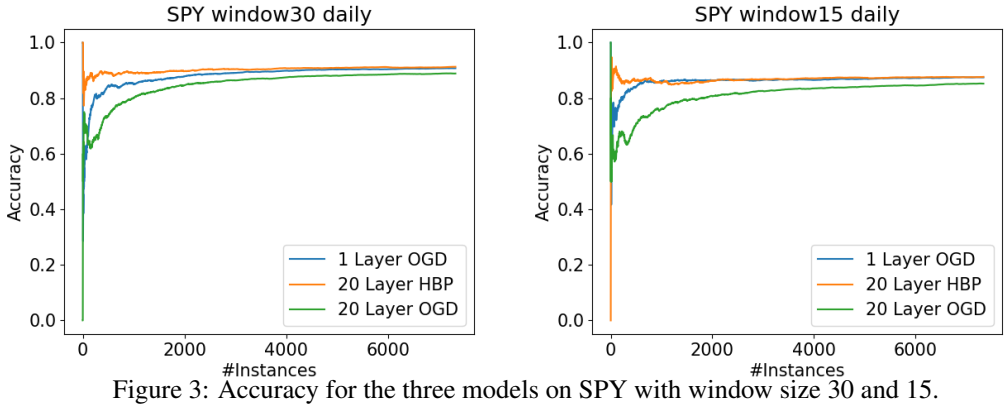

Figure 3: Accuracy for the three models on SPY with window size 30 and 15.

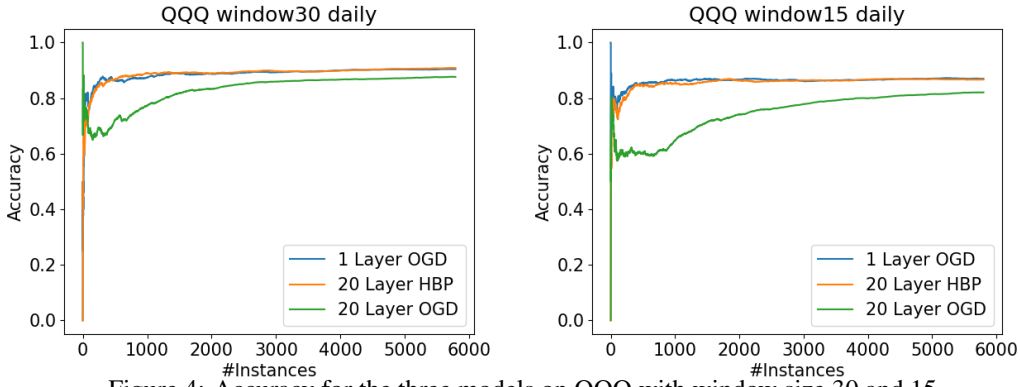

Figure 4: Accuracy for the three models on QQQ with window size 30 and 15.

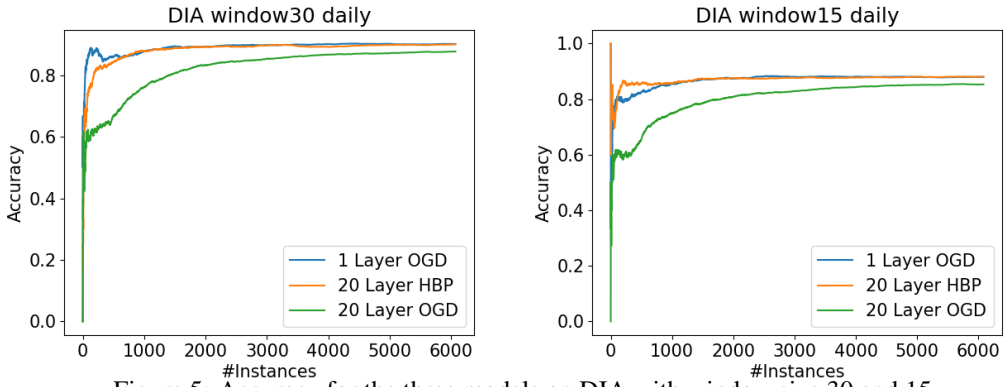

Figure 5: Accuracy for the three models on DIA with window size 30 and 15.

D Results of Grid Search for Technical Indicator Datasets

NB: None of the models converged so we excluded the tables as they did not show any additional information.

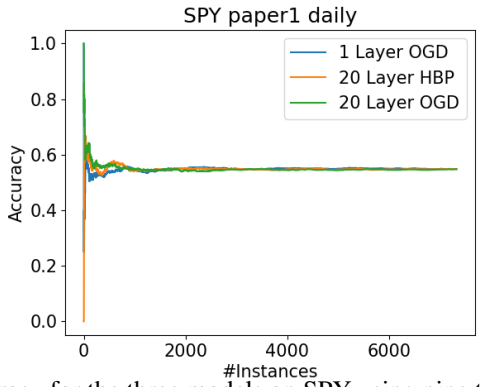

Figure 6: Accuracy for the three models on SPY using nine technical indicators.

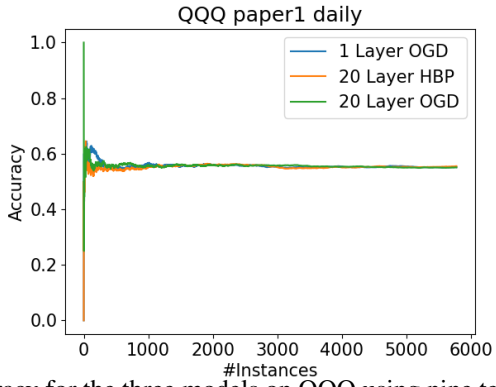

Figure 7: Accuracy for the three models on QQQ using nine technical indicators.

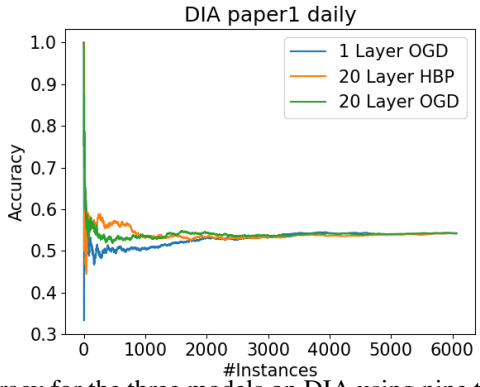

Figure 8: Accuracy for the three models on DIA using nine technical indicators.