Online Knowledge Distillation for Financial Timeseries Forecasting

Pavlos Floratos, Avraam Tsantekidis, Nikolaos Passalis, and Anastasios Tefas
Computational Intelligence and Deep Learning Group
Artificial Intelligence and Information Analysis lab, Department of Informatics
Aristotle University of Thessaloniki, Thessaloniki, Greece
Emails: {pfloratos, avraamt, passalis, tefas}@csd.auth.gr

***Abstract*—Recent advances in Deep Neural Networks (DNNs) led to enormous progress in many different fields, covering a wide range of applications, including financial time-series analysis. Many different financial time-series forecasting tasks have been successfully tackled using such approaches, including predicting the next day's return of FOREX currency pairs. However, using DNNs for such tasks is not always straightforward due to training stability issues that often arise. Indeed, the noisy nature of the data can often cause considerable different behaviors between DL models, despite following the same training process, model architecture, and hyper-parameters. At the same time, the methods proposed for generating the training labels can sometimes further reinforce such issues. All these phenomena can reduce the reliability of training DL models for financial forecasting tasks, while also making the training process especially time-consuming, requiring several validation and back-testing runs. To overcome these limitations, we propose an ensemble-based online distillation method that can significantly reduce this behavior. The proposed method is efficient since, in contrast to offline distillation approaches, it works in a single step, while it also allows for reducing the number of hyper-parameters to be tuned, e.g., the number of epochs for training the teachers. As demonstrated in the conducted experiments, the soft labels extracted through the proposed approach can mitigate the effect of noisy annotation that often exists in FOREX data, leading to significant performance improvements.**

I. Introduction

Equities, bonds, currencies, and derivatives are traded in financial markets forming the backbone of the economy. They serve multiple roles, ranging from enabling commercial activity between different countries through foreign exchange currency markets (FOREX) to allowing businesses secure the required capital to expand and grow. At the same time, financial markets allow investors to profit through speculation. This has led to a significant amount of research for analyzing and predicting the future behavior of various assets, allowing for taking the appropriate market position that would eventually lead to profitable trades. Even though the behavior of assets is determined by many different factors, including external social parameters and the actions of a large number of investors, several different methodologies have been proposed to model the - occasionally irrational - behavior of financial markers. This is usually achieved through appropriate modelling of the markets' behavior using tools that range from traditional quantitative analysis to modern machine learning methodologies.

Indeed, many tools have been developed to effective forecast and analyze financial markets, leading to an increasing participation of machine learning-based agents in financial markets [1]. The preceding difficulties have been tackled to a great degree by the development of Deep Learning (DL) tools [2], that enabled automated agents to exploit the vast amount of data collected from financial markets, outperforming to a significant degree the methods used until then [3]–[7]. A wide range of different methods have been used to this end, ranging from supervised learning approaches using Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) recurrent neural networks (RNNs) [8], [9] to Deep Reinforcement Learning (DRL) methodologies [10], [11]. Recent literature also attempts to fuse multimodal information by incorporating additional information sources, such as information extracted from natural language processing and sentiment analysis of news articles, social media [12]–[14] and keyword trends of internet search engines [15], [16], to enable for even more accurate forecasts.

The aforementioned methods approach the problem of financial trading by employing different problem formulations and model architectures. However, most of them are vulnerable to the excessive noise that often exist in financial data and usually reduces the stability when training neural networks. Often, this can lead to overfitting the training data, which can negatively impact the ability of the models to generalize to unseen data. This can be attributed to the existence of many almost equivalent hypotheses that can fit the training data, which can lead to very different predictions for the test data. To better demonstrate this behavior, we have conducted an experiment where we trained a DL-based agent to predict the next day's return of the *EUR/USD* currency. The evaluation results are reported in Fig. 1, where we plotted the results in unseen data for the exact same residual CNN architecture for five different training runs. The only difference between the experiments is the used seed. It is worth noting that even though all agents have the same architecture and they are trained on the same data, they end up having different behavior during (and after) the optimization. Indeed, the best agent can achieve a PnL of almost 10%, while the worst one can reach a PnL of -10% during the optimization. We also observe similar instabilities for the test accuracy. Such behavior makes training a DL-based agent for financial trading much more

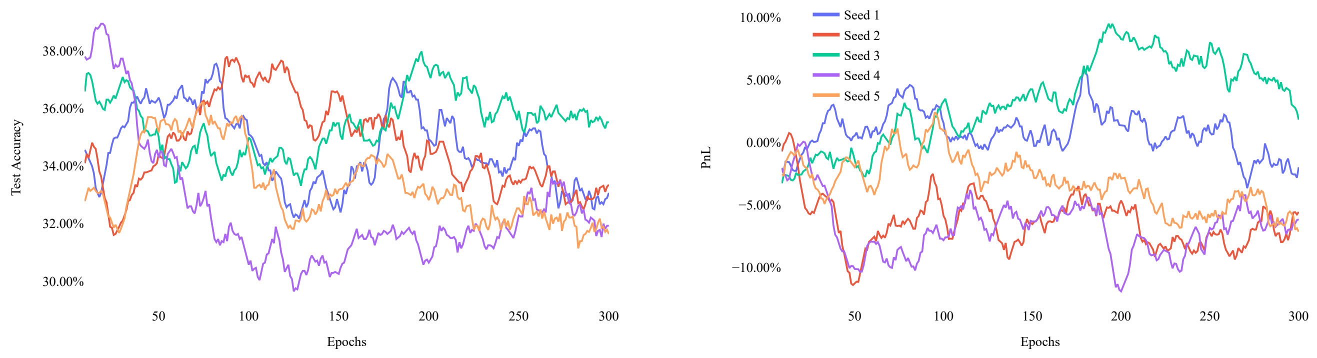


Fig. 1: Comparing five different training runs using different initializations of the same deep residual CNN. Both the forecasting accuracy and profit and loss (PnL) in the test set are reported. These experiments demonstrate the instabilities arising from the noisy nature of financial data when training DL models for financial forecasting tasks.

difficult compared to standard approaches that are typically followed in other domains, e.g., computer vision. Employing various regularization methods, such as dropout [17], weight decay [18] or using smaller networks [19], do not usually allow for significantly reducing these phenomena. Formulating trading as a classification task, where the model decides between when to *buy* or *sell* an asset (or in some formulations, also to *exit* the market) can make the learning problem easier compared to regressing the exact price of an asset. However, at the same time, it introduces an additional hyper-parameter for generating the ground truth labels, i.e., selecting which price movements should be assigned to each of the two/three classes.

In this work, we suggest that these phenomena can be partially attributed to the noise that exists in the annotations of the data. For example, two input data samples that are similar can often have different ground truth annotations (e.g., regression targets or labels). Indeed, this is often the case when due to external causes, e.g., government announcements, the price of an asset can move in vastly different directions. In this paper, we conjecture that the noise in data annotations can reduce the performance of DL agents trained for financial tasks. Another source of such behavior is the chaotic nature of some financial time-series [20]. These phenomena are often recognized in recent literature, even though not always clearly stated. For example, the development of handcrafted labels, which are usually produced by assigning a class to each sample based on some criterion, e.g., [3], [4], can be considered as a way to reduce the impact of noisy labels. Indeed, when the threshold for the classes is correctly selected, these approaches can mitigate these effects. However, the core issue remains: when two similar input samples carry different annotations, the network is forced to overfit the noise component of the data, learning highly nonlinear decision surfaces that can fit the input training data. Some other recent approaches employ distillation schemes to reduce the impact of noisy labels by mimicking the annotations extracted from other DL models [21], [22]. Another line of research includes introducing prior knowledge, e.g., as extracted from human indicators, into the training process to remove non-stationary noise [23]. However, these methods typically require using human annotations, which can be especially hard to acquire.

The main contribution of this paper is an online distillation scheme that can be used to mitigate the effect of the aforementioned phenomena. The proposed method allows the ground truth labels to shift, reducing in this way the noise that often exists in the ground truth data. To this end, the proposed method builds upon knowledge distillation, where an ensemble of multiple teachers is trained using the original, yet potentially noisy, labels. Then, this ensemble is used to extract the updated and filtered annotations that will be used for training the student models. Even though using an ensemble of multiple teacher models enables us to more reliably estimate the ground truth annotations, it also significantly increases the training complexity. Indeed, the training process is now a two-step process, which includes first training the teachers' ensemble and then using the extracted annotations to train the final model. To overcome this limitation, we propose employing an online training setup, where the teachers' ensemble is trained simultaneously with the student model. In this way, the proposed method is significantly easier to apply, since it does not require separately training the teacher and student models, allowing for training all models in just one step. This also reduces the impact of some hyper-parameters, e.g., the number of training epochs for the ensemble, since all models are trained at the same time. To the best of our knowledge, the proposed method is the first online ensemble-based financial-oriented distillation scheme that is implemented in an end-to-end pipeline allowing for performing one-step distillation overcoming the need for time-consuming two-stage training pipelines (e.g., [21], [22]). We have conducted extensive experiments on large-scale FOREX datasets that include 37 different currencies, demonstrating that the proposed method can significantly improve the accuracy of financial forecasting and trading.

The rest of the paper is structured as follows. First, we introduce the necessary background, along with the proposed method in Section II. Then, we provide the experimental evaluation and discuss the obtained results in Section III. Finally, Section IV concludes the paper.

II. Proposed Method

In this Section we first provide the necessary background on financial trading and training DL agents. Then, we present and discuss the proposed method for online distillation for financial trading.

This work focuses on DL-based models that work by mapping an input $\mathbf{x} \in \mathbb{R}^N$ to an output $\mathbf{y} \in \mathbb{R}^C$, i.e., $\mathbf{y} = f_{\mathbf{W}}(\mathbf{x})$, where N denotes the input dimensionality, the notation C is used to refer to the dimensionality of the output space and $\mathbf{W}$ denotes the trainable parameters of the DL model. For the rest of this Section we will focus on classification tasks, where usually the so called *one hot* encoding is used, i.e., each class is mapped into an output neuron, which estimates the probability of each input sample belonging to the corresponding class. We also use the Open–High–Low–Close (OHLC) price level technique for encoding the input data, reducing the trading data into four values for a specified time interval. This is without loss of generality, since the input can be provided in any form to the network.

Using OHLC candlesticks allows for providing a clear picture of the price movement in every interval. The daily returns for every currency pair, when OHLC candlesticks are used, can be calculated as:

$$R_t = \frac{P_{o,t+1} - P_{o,t}}{P_{o,t}}, \tag{1}$$

where R_t denotes the daily return value, $P_{o,t+1}$ denotes the next day's open price and $P_{o,t}$ denotes the current day's open price. We also focus on solving a price direction forecasting problem instead of regressing the exact price of an asset, following the models used in the recent literature on financial trading [3], [4]. The daily return R_{t+1} is calculated for the next day and then we generate the ground truth labels using three possible price directions (up, down, and stationary). These directions define the three classes that dictate the agent's actions, i.e., *buy*, *sell* or *exit* the market. It is worth noting that we employed a three-class formulation, which enables us to account for commission fees and price slippage, given that appropriate return thresholds have been defined. The class label for the t-th timestep can be then defined as:

$$l_t = \begin{cases} 0, & \text{if } R_t < -r_{down} \text{ (sell)} \\ 1, & \text{if } R_t > r_{up} \text{ (buy)} \\ 2, & \text{otherwise (exit)} \end{cases}, \tag{2}$$

where r_{down} and r_{up} are used to define the return thresholds. Afterwards, we can use the cross-entropy loss for training the network as:

$$H(\mathbf{y}, \mathbf{l}) = -\sum_{i=1}^{C} [\mathbf{l}]_i \log([\mathbf{y}]_i), \tag{3}$$

where the one hot encoding of the label of a training sample is denoted by $\mathbf{l}$, while $[\mathbf{x}]_i$ denotes the i-th element of a vector $\mathbf{x}$. Gradient descent-based algorithms, such as Adam [24], can be then used for optimizing the network.

Recent findings in the literature have already highlighted the positive effect of using knowledge distillation approaches for training trading agents using deep refinement learning tasks [21], [22]. Knowledge distillation is a method that enables training *student* models under the supervision of a stronger *teacher* network. The soft outputs produced by the teacher are more informative about the data than standard hard labels, as they are enriched with information about hidden data similarities, and can be used to more efficiently train the student network. Even though distillation methods were originally proposed for model compression purposes, transferring the knowledge from a large and complex neural network into smaller and more efficient ones [25], [26], the knowledge distillation process can also introduce a regularization effect that allows for improving the accuracy of various models.

Let $\mathbf{y}^{(T)} = f_{T,\mathbf{W}_T}(\mathbf{x}) \in \mathbb{R}^C$ denote the *teacher* network and $\mathbf{y}^{(S)} = f_{S,\mathbf{W}_S}(\mathbf{x}) \in \mathbb{R}^C$ denote the *student* network, where the notation $\mathbf{W}_T$ and $\mathbf{W}_S$ is used to refer to the trainable parameters of these models. To avoid cluttering the used notation, we will directly refer to these models by using their output, i.e., $\mathbf{y}^{(T)}$ and $\mathbf{y}^{(S)}$. Strong teacher models usually overfit the data and tend to be very confident regarding their predictions. As a result, valuable class similarity information can often diminish as values close to one and zero are predicted by the teacher models. This issue is resolved by neural network distillation by using a temperature hyper-parameter in the softmax function:

$$[\mathbf{y}^{(T,soft)}]_i = \frac{\exp([\mathbf{z}^{(T)}]_i/M)}{\sum_j \exp([\mathbf{z}^{(T)}]_j/M)}, \tag{4}$$

where the notation $[\mathbf{y}^{(T,soft)}]_i$ is used to refer i-th class's probability, as estimated by the teacher network, $[\mathbf{z}^{(T)}]_i$ are the logits of the teacher network and M is used to refer to the temperature hyper-parameter. Note that the regular softmax function can be recovered if $M = 1$ is used. Using higher temperature allows for generating softer distributions that provide more information regarding the similarities between a data sample and different classes. However, it is worth noting that increasing the temperature is only necessary when the teacher network is overly confident and it is not required for more difficult tasks or when smaller teachers are used [27].

The student network can be then optimized using both the probabilities extracted from the teacher network and the ground truth labels. Therefore, the loss function used for distillation is defined as:

$$\mathcal{L}_{KD} = \alpha_S H(\mathbf{y}^{(S,soft)}, \mathbf{y}^{(T,soft)}) + \alpha_H H(\mathbf{y}^{(S)}, \mathbf{l}) \qquad (5)$$

where $\mathbf{y}^{(S,soft)}$ denotes the softened output of the student network after raising the temperature of the softmax function to M, while $\mathbf{y}^{(S)}$ denotes the regular output of the student network, i.e., when temperature is set to $M = 1$. Also, α_S and α_H denote the weight of the soft labels (distillation loss) and the hard ground truth labels (regular cross entropy) in the training process.

In this paper, we directly rely on the soft labels generated by the teacher model, since the original labels are noisy, negatively affecting the training process. In addition, to provide a more reliable estimation of these soft labels, we propose employing an ensemble of smaller teacher models instead of a larger teacher model. This can allow for extracting a more reliable estimation, since one powerful teacher can overfit and overestimate the probability of the correct class [21]. Handcrafted labels and customized class splits in financial time-series classification tasks also come with a discontinuity problem, since there will always exist very similar data that eventually belong to different classes, e.g., one with next day's return 0.53% and another with 0.55%. Using soft labels allows for overcoming this discontinuity by allowing the teacher model to generate new ground truth targets, minimizing the negative effects that can arise from forcing the model to learn such highly nonlinear phenomena and potentially intensifying overfitting issues.

To this end, we first train a set of teacher models for a predefined number of epochs using the hard ground truth labels $\mathbf{l}$. The regular cross-entropy loss can be used for training the teachers:

$$\mathcal{L}_k = H(\mathbf{y}^{(T,k)}, \mathbf{l}), \qquad (6)$$

where $\mathbf{y}^{(T,k)}$ refers to the output of the k-th teacher. The final soft labels of the ensemble are calculated as the average of the probability distribution output of all teachers as:

$$\mathbf{y}^{(T,soft)} = \frac{1}{K}\sum_{t=1}^{K} \mathbf{y}^{(T,k)}, \qquad (7)$$

where K is the total number of teacher networks. Then, the student model is trained by directly using these soft labels as:

$$\mathcal{L}_{KD,en} = H(\mathbf{y}^{(S,soft)}, \mathbf{y}^{(T,soft)}). \qquad (8)$$

In our work the teacher and student models use the same architecture, while the only difference between them is their initialization. However, this is without loss of generality, since the proposed method can be also employed for models that have different architectures, given that the final output dimensionality (number of classes) is the same for different models.

The aforementioned process can improve the accuracy of the models, as we also experimentally demonstrate in Section III. However, it is computationally intensive, since it requires first training the teacher ensemble and then proceeding with

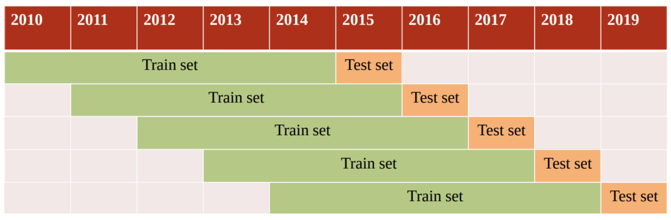


Fig. 2: We used a walk forward setup to evaluate all methods. For each split, we used five years for training and one year for backtesting.

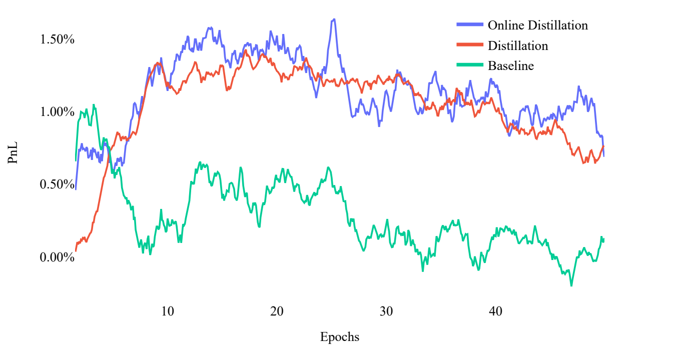
training the student model. This process also introduces an additional hyper-parameter to be tuned: the number of epochs that the teacher models will be trained for. Specifying this hyper-parameter for financial trading problems, where DNNs tend to overfit from the very first epoch, is not a trivial task and can affect the performance of the student model. In this work, we propose employing an online distillation scheme to overcome this limitation. To this end, we propose simultaneously training the teachers' ensemble and transferring the knowledge to the student model. Therefore, we apply (6), (7), and (8) *on-the-fly* by simultaneously training both the teacher and the student models. The proposed pipeline can be summarized as follows:

1) First, train each of the teacher model using the hard labels l.
2) Then, aggregate the output of the ensemble to form the soft targets that will be used for training the student model.
3) Finally, train the student for one epoch using these targets.
4) The previous three steps are repeated for a total of N training epochs.

III. Experimental Evaluation

The experimental evaluation of the proposed method is provided in this Section. First, we describe the used datasets, feature extraction procedures, as well as the architecture of the employed agents. Then, we proceed by performing several experiments to examine the effect of performing both offline ensemble-based distillation, as well as the proposed online distillation approach. For all evaluation experiments, we used the same hyper-parameters and network architectures, altering only the distillation methodology, to ensure a fair comparison between different methods.

We used FOREX trading data that belong to 37 different currencies, such as *EUR/USD*, *CHF/JPY*, *GBP/CAD*, *USD/NOK*. The trading data were collected in the period of 2010-2019, while they were sub-sampled using the Open–High–Low–Close (OHLC) price level technique using daily candles. A total of 114,234 samples were collected using this approach. For the evaluation we also employed a walk forward setup using five train/test splits, as shown in Fig. 2. Each train split consists of five years of train data and one year



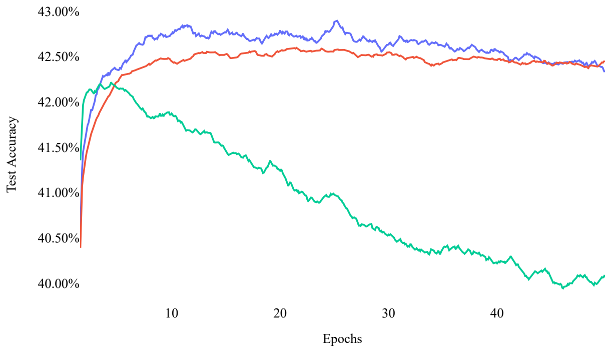


Fig. 3: Comparison (test PnL and accuracy) between DL agents trained with a) hard labels, b) plain distillation and c) online distillation.

of test data. For example, the first train split contains data from 2010 to 2014 and the evaluation is performed on 2015, the next train split contains data from 2011 to 2015 and the evaluation is performed on 2016, etc. For each experiment we evaluated the performance of each method on all these splits, unless otherwise noted. The average classification accuracy and PnL on the test set is reported for all the conducted experiments. The PnL metric corresponds yearly percentage of gains or losses that were achieved based on the trading decisions. For example, a 10% PnL for an initial investment of 100$ would lead to a total of 110$. On the other hand, the same initial investment with -4% PnL would become 96$. Both testing accuracy and PnL are frequently measured during the training iterations. In this way, we are able to provide more details regarding the stability of the agents during the training process, as well as their performance in unseen data. Note that a small moving average filter was applied on all the provided plots in order to improve their readability. Also, a commission of $5 * 10^{-5}$ per trade was used for the calculation of the PnL.

The input to the DL models consist of a window of $W = 30$ daily return values. The models were also trained to perform trading using all available data, i.e., we did not train separate agents for each currency pair. Instead, we train each network using data from all currency pairs and we do report the average accuracy and PnL over the 37 available currencies. Furthermore, we used standardization scaling to normalize the input data, leading to data with zero mean and unit variance. Regarding label extraction, the thresholds r_{up} and r_{down} in (2) were selected in such a way so that all classes will be perfectly balanced in the training set, i.e., 1/3 of the data will have the label *buy*, 1/3 of the data will have the label *sell* and 1/3 of the data will have the label *exit*, in order to avoid class imbalance issues.

The network architecture used for the conducted experiments was based on a residual network that consists of two blocks. More specifically, each block consists of three convolutional layers. The first convolutional layer and the first residual block use a kernel size of 5, while the stride was set to 1. For these layers we used padding equal to 2. For the last residual block we used a kernel size of 3, while the stride was set to 1 and padding of 1 was used. All convolutional layers have 35 filters and are followed by batch normalization and ReLU activation functions. Then, average pooling (stride 3) is applied to this representation, which then is fed into a fully connected layer followed by the softmax activation function with temperature $M = 1$. The output of the model corresponds to the probability distribution over the three available actions, i.e., *buy*, *sell*, *exit*. The Adam optimization algorithm [24] using its default hyper-parameters and a learning rate of 0.0004 was used for training all models. Finally, we used a batch size of 512, while we regularized all models using weight decay equal to 0.5.

First, we evaluate the impact of using an offline variant of the proposed knowledge distillation approach. An ensemble of five teachers was used to this end. Each of the agents was trained for 10 epochs using the ground truth labels. The output of each of the training agents was averaged and then used to train the student agent, as described in Section II. We report the PnL and accuracy for three different evaluation setups in Fig. 3. The aforementioned distillation process is denoted by “Distillation” in this Figure. Using this approach, we can significantly improve both the profitability, as well as the accuracy of the agents, compared to the baseline agent that was trained with hard labels. The regularization effect arising from using knowledge distillation is especially evident, since the baseline agent transits into an overfitting regime in less than 10 epochs. Then, we proceeded with evaluating the proposed online distillation methodology. This method is denoted by “Online Distillation” in Fig. 3. As the experimental results suggest, using online distillation leads to faster convergence. At the same time, the positive regularization effect arising from distillation is maintained, while we can even achieve slightly better performance compared to offline distillation. At the same time, the proposed method allows to train the agents more easily, since it consists of a single-step pipeline. Moreover, it allows for reducing the number of hyper-parameters that need to be tuned, since the teachers are trained along with the students, removing the dependency on the number of epochs used for training the teachers' ensemble.

IV. Conclusions

In this paper, we proposed a novel online distillation method to transfer the knowledge from an ensemble of teacher models into one single student model, allowing to reduce the effect of noisy labels that often exist in financial forecasting tasks. The experimental results suggest that a teachers' ensemble, which was trained with potentially noisy labels for a predefined number of epochs, can be successfully used to extract the new ground truth annotations, based on which the student network was trained. In this way, we were able to extract more reliable estimations for the final ground truth annotations, reducing the impact of the existing noisy labels. To further improve the computational complexity of the proposed method, we employed an online training setup, where the teachers' ensemble was trained together with the student model. As a result, the proposed method was significantly easier to apply, since it does not require separately training the teacher and student models, reducing the whole process to just one single-step training pipeline.

It is worth noting that the proposed method can be still affected by the noise that exists in the training labels, since the teachers' ensemble is trained using these annotations. The effect of this noise can be potentially further reduced by employing self-distillation approaches for the teacher models [28]. Such approaches could help further increase the accuracy of the resulting models, without significantly affecting the training or inference complexity of the proposed method.

Acknowledgements

This work has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH - CREATE - INNOVATE (project code: T2EDK-02094).

References

[1] R. Haynes and J. S. Roberts, "Automated trading in futures markets," *CFTC White Paper*, 2015.

[2] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[3] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *Proceedings of the IEEE Conference on Business Informatics*, vol. 1, 2017, pp. 7–12.

[4] Z. Zhang, S. Zohren, and S. Roberts, "Deeplob: Deep convolutional neural networks for limit order books," *IEEE Transactions on Signal Processing*, vol. 67, no. 11, pp. 3001–3012, 2019.

[5] M. Dixon, D. Klabjan, and J. H. Bang, "Classification-based financial markets prediction using deep neural networks," *Algorithmic Finance*, vol. 6, no. 3-4, pp. 67–77, 2017.

[6] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PloS One*, vol. 12, no. 7, p. e0180944, 2017.

[7] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 5, pp. 1407–1418, 2018.

[8] S. Mehtab and J. Sen, "Stock price prediction using convolutional neural networks on a multivariate timeseries," *arXiv preprint arXiv:2001.09769*, 2020.

[9] S. Mehtab, J. Sen, and S. Dasgupta, "Robust analysis of stock price time series using cnn and lstm-based deep learning models," in *Proceedings of the International Conference on Electronics, Communication and Aerospace Technology*, 2020, pp. 1481–1486.

[10] A. Tsantekidis, N. Passalis, A.-S. Toufa, K. Saitas-Zarkias, S. Chairistanidis, and A. Tefas, "Price trailing for financial trading using deep reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[11] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 653–664, 2016.

[12] S. Mehtab and J. Sen, "A robust predictive model for stock price prediction using deep learning and natural language processing," *Available at SSRN 3502624*, 2019.

[13] T. H. Nguyen, K. Shirai, and J. Velcin, "Sentiment analysis on social media for stock movement prediction," *Expert Systems with Applications*, vol. 42, no. 24, pp. 9603–9611, 2015.

[14] N. Oliveira, P. Cortez, and N. Areal, "The impact of microblogging data for stock market prediction: Using twitter to predict returns, volatility, trading volume and survey sentiment indices," *Expert Systems with Applications*, vol. 73, pp. 125–144, 2017.

[15] T. Preis, H. S. Moat, and H. E. Stanley, "Quantifying trading behavior in financial markets using google trends," *Scientific Reports*, vol. 3, no. 1, pp. 1–6, 2013.

[16] M.-H. Fan, M.-Y. Chen, and E.-C. Liao, "A deep learning approach for financial market prediction: Utilization of google trends and keywords," *Granular Computing*, vol. 6, no. 1, pp. 207–216, 2021.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Proceedings of the Advances in Neural Information Processing Systems*, 1992, pp. 950–957.

[19] S. Oymak, "Learning compact neural networks with regularization," in *Proceedings of the International Conference on Machine Learning*, 2018, pp. 3966–3975.

[20] J. E. Sandubete and L. Escot, "Chaotic signals inside some tick-by-tick financial time series," *Chaos, Solitons & Fractals*, vol. 137, p. 109852, 2020.

[21] A. Tsantekidis, N. Passalis, and A. Tefas, "Diversity-driven knowledge distillation for financial trading using deep reinforcement learning," *Neural Networks*, vol. 140, pp. 193–202, 2021.

[22] ——, "Improving deep reinforcement learning for financial trading using neural network distillation," in *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing*, 2020, pp. 1–6.

[23] J. Fang and J. Lin, "Prior knowledge distillation based on financial time series," in *Proceedings of the IEEE International Conference on Industrial Informatics*, vol. 1, 2020, pp. 429–434.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[26] N. Passalis, M. Tzelepi, and A. Tefas, "Probabilistic knowledge transfer for lightweight deep representation learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 5, pp. 2030–2039, 2020.

[27] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," *Proceedings of the Advances in Neural Information Processing Systems*, vol. 30, 2017.

[28] L. Zhang, C. Bao, and K. Ma, "Self-distillation: Towards efficient and compact neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.