Uncertainty Awareness for Predicting Noisy Stock Price Movements.

Yun-Hsuan Lien, Yu-Syuan Lin, and Yu-Shuen Wang ✉

National Yang Ming Chiao Tung University

Abstract. Predicting stock price movements is challenging because financial markets are noisy – signals and patterns in different periods are dissimilar and often conflict with each other. Consequently, irrespective of whether the price rises or falls, none of the previous methods achieve high prediction accuracy in this binary classification task. In this study, we consider aleatoric uncertainty and model uncertainty when training neural networks to forecast stock price movements. Specifically, aleatoric uncertainty is known as statistical uncertainty. It indicates that similar historical price trajectories may not lead to similar future price movements. On the other hand, model uncertainty is caused by the model's mathematical structures and parameter values, which can be used to estimate whether the models are familiar with the testing sample. Considering that most of the existing uncertainty estimation methods focus on model uncertainty, we transform the aleatoric uncertainty in financial markets to model uncertainty by removing samples with similar historical price trajectories and different future movements. The Bayesian neural network is then adopted to estimate the model uncertainty during inference. Experiment results demonstrated that the networks achieved high accuracy when they were certain about their predictions.

Keywords: Stock price movement prediction, aleatoric uncertainty, model uncertainty, uncertainty quantification

1 Introduction

Stock price movement prediction has attracted intense attention in the research field and the financial industry because investors can manage their portfolios to earn substantial profits from price fluctuations. The prediction can be considered as a binary classification problem, in which the inputs are historical prices, news, and company statistics, and the output is a rise or a fall in price at the next period (e.g., tomorrow). Theoretically, although price movements could exhibit a trend, in practice, a high level of uncertainty remains. The trend also changes over time so that knowledge learned from the past could be insufficient to predict future price movements. In fact, even the state-of-the-art method [8] achieved accuracy that was only slightly higher than a random guess in this binary classification task when taking historical prices as inputs.

In this study, we aim to quantify the uncertainty of stock price movement prediction, which is helpful for later investment strategies. On the one hand, stocks

that have similar price trajectories in the past may not result in similar movements in the future, and we refer to the phenomenon as *aleatoric uncertainty* [30]. Aleatoric uncertainty is known as statistical uncertainty, which indicates that the outcomes of an experiment in different runs can be different because of unknown or inherently random effects. A typical example is coin-flipping. In other words, the price movement cannot be accurately predicted due to currently unavailable information. On the other hand, mathematical structures and parameter values of a model that are used to map inputs to outputs result in *model* uncertainty. It is known that neural networks are composed of tremendous parameters, and the networks can be considerably dissimilar if they are trained from different initializations, although on the same dataset and using the same loss. In other words, given an input, particularly an input that is dissimilar to all samples in the training set, networks trained from different initializations are likely to output different results. This model uncertainty should also be considered when predicting stock price movements.

We transform the aleatoric uncertainty of stock price movements to model uncertainty for quantification. Specifically, since similar historical price trajectories do not necessarily lead to similar future movements, we treat rises and falls in price in the next period as noisy labels. The noisy label training is then applied to remove such samples from the training set. It deserves noting that, when removing the samples with noisy labels, the aleatoric uncertainty becomes model uncertainty because the network does not learn from these samples. We then apply the Bayesian neural network to quantify model uncertainty – whether the networks are familiar with a testing sample or they just guess the solution. Unlike general neural networks, a Bayesian neural network takes a single sample as input but outputs a label distribution. The variance of the distribution implies the level of uncertainty. We exploit the depth structure of neural networks to quantify uncertainty [2]. Our network contains multiple branches of different depths and outputs multiple labels to forecast future price movements. Since different branches imply different prediction functions, the predictions are likely diverse if inputs are dissimilar to any training sample. For this reason, we measure prediction entropy and variance to elucidate how certain the network is about its output.

We estimate the uncertainty of stock price movement prediction. We argue that our network is *uncertainty-aware* because it not only outputs the prediction but also reveals the level of uncertainty. For evaluation, we trained neural networks to predict stock price movements on the next day and after ten days on the *KDD17* and *ACL18* benchmark datasets. We also evaluated the prediction accuracy in terms of the network's prediction uncertainty on our collected dataset that contains bull, flat, and bear markets. Experiment results showed that high certainty of our network's prediction indeed leads to high accuracy. Below, we summarize our contributions:

- We estimate each prediction's level of uncertainty, which can assist investors to develop flexible investment strategies.

– We transform the aleatoric uncertainty of stock price movements to model uncertainty by removing samples with noisy labels. Then, we apply the Bayesian neural network to quantify model uncertainty.
– Our experiment results verified that high certainty of a network prediction indeed leads to high accuracy.

2 Related Work

Financial Forecasting. Recently, neural networks gained considerable attention in financial forecasting [13]. They are trained to predict stock prices [1], stock movements [27, 24, 5], and currency exchange rates [32]. Most of the networks considered historical prices and technical analysis [6] when making predictions because of "the market discounts everything theory", i.e., the price reflects all information in the market. Considering that financial markets are noisy and non-stationary, Feng et al. [7] applied adversarial training [16] to help networks learn robust representations and facilitate generalization. In addition to prices, several methods additionally considered limit order books [31, 28, 38], news events [4], and texts from social media [26, 34] for prediction. Their experiment results confirmed the benefits of these metadata. Rather than predicting future trends, Feng et al. [8] temporally captured stock relations and ranked the stocks according to their return ratios. The back-test results demonstrated that trading based on stock ranking were more effective than that based on future trends.

Uncertainty Estimation. In time series forecasting applications, training and testing data are from different periods and are likely to contain varied properties. Since standard networks are trained by maximum likelihood estimation (MLE), they are frequently over-confident with unfamiliar samples [25]. Due to the non-stationarity of financial markets, it is essential to extend a standard network to a Bayesian network with posterior inference from a probabilistic perspective. Under this formulation, the output becomes a distribution rather than a single point. When the network takes a novel observation as input, its output's distribution will have a large variance. Generally, the posterior inference can be approximated by dropout [10, 9], model ensembles [17], subspace inference [14], and depth [2]. The high variance of the outputs implies high uncertainty.

Noisy Labels. Since neural networks can fit even random variables [36], their robustness easily degenerates when training data contain noisy labels. To ease the problem, one of the directions taken was sample selection. These methods train two networks with the same architecture and select samples according to output disagreement [22], smaller loss values [12], and their combinations [35], to update network parameters. Besides sample selection, Wei et al. [33] pointed out that networks are likely to have the same outputs if the labels are correct. Hence, they updated the network by label loss and distance loss, and expected the two networks to become the same.

3 Method

Price movement prediction can be considered to be a binary classification problem, which takes historical price movements as inputs and forecasts price rises or price falls at the next period. Specifically, let the stock market data be $\mathcal{D} = \{(\mathbf{X}_t, y_{t+1})\}_{t=1}^{N}$, where $\mathbf{X}_t = [x_{t-T+1}, ..., x_t] \in \mathcal{R}^{F \times T \times W}$ with F-dimensional features (e.g., high and low prices of a period, and the corresponding technical analysis) in the lag of the past T time-steps, W is the number of moving averages, and y_{t+1} denotes the actual price movement at the next period. The goal is to build a network $f(\cdot)$ with $\mathcal{D}$. Specifically, $f(\mathbf{X_t})$ forecasts the price movement y_{t+1} based on the currently available features $\mathbf{X_t}$.

3.1 Transforming Aleatoric Uncertainty to Model Uncertainty

Financial market prices constitute non-stationary time series data. The price movements are drifting, and their distributions change over time. In addition, when the market does not have a clear direction, the price moves up and down in a small range, bringing many meaningless labels. In other words, although price movements could exhibit a trend, they contain aleatoric uncertainty. Considering that most existing methods focus on quantifying model uncertainty, we attempt to transform aleatoric uncertainty into model uncertainty by removing samples with noisy labels for quantification. Specifically, samples with noisy labels are those with similar $\mathbf{X}_t$ but different y_{t+1}. Since the network does not learn from the samples with aleatoric uncertainty, it can only guess the results when such samples appear during inference, and the uncertainty can thus be quantified.

Consistent and Inconsistent Labels. Differentiating correct and incorrect labels in a traditional classification problem is simple. However, in financial markets, there is no clear definition between them. All labels can be considered correct because the stock price movements indeed happen. Since prediction implies choosing the price movement that likely occurs, financial data can be partitioned into two groups: samples with consistent and inconsistent labels. Consistent labels are those that appear more frequently than others under the condition of similar historical patterns. This new definition fulfills the assumption in the noisy label training that correct labels are the majority, and networks can learn these labels at the early stage of training.

Let $\mathbf{X}_i$, $\mathbf{X}_j$ be the price features in consecutive days with $\|\mathbf{X}_i - \mathbf{X}_j\|_2 \leq \varepsilon$, and y_i, y_j be the corresponding ground truth labels. Li et al. [19] defined that the labels of $\mathbf{X}_i$ and $\mathbf{X}_j$ are inconsistent if $\|y_i - y_j\| \geq \delta$. They also proved that the network $f(\cdot)$ with randomly initialized parameters has to traverse a long distance to fulfill inconsistent labels. In other words, in the early iterations, models updated by the gradient descent method only fit samples with consistent labels, essentially ignoring the inconsistent ones. This motivates us to remove samples with inconsistent labels when training since they contain aleatoric uncertainty.

Sample selection. Previous experiment results show that deep neural networks tend to learn correct labels at the early stage of training and then memorize the incorrect ones [3]. In other words, at the early stage of training, the loss

of correct labels decreases quickly, whereas the loss of incorrect labels remains high. Therefore, removing inconsistent labels is equivalent to removing samples that have a large training loss [15, 29]. In our implementation, we adopt the co-teaching strategy in [12] and train two neural networks simultaneously. Each network selects samples with a small loss to train the other. Let f and g be the two networks with parameters ω_f and ω_g, respectively. Each mini-batch B will be reduced to

$$\tilde{B}_i = \underset{|\tilde{B}_i|=r|B|}{arg\min}\ \ell(B, i), \tag{1}$$

where $i \in \{f, g\}$, ℓ is the loss function and r is the ratio of data that we want to keep. To avoid accumulated error from the sample-selection bias, we train the network f using samples in $\tilde{B}_g$ and the network g using samples in $\tilde{B}_f$.

Deep neural networks tend to learn easy patterns first and gradually memorize hard samples as the number of training epoch increases. Therefore, we keep more samples at the early stage of training and gradually increase the filter out ratio. Let $R(n) \in [1-\tau, 1]$ be the proportion of the mini-batch samples that will be kept at the n^{th} training iteration. We formulate the proportion as

$$R(n) = 1 - \min\left\{\frac{n}{n_k}\tau, \tau\right\}, \tag{2}$$

where $n \in [1, N]$, $n_k \leq N$, N is the total number of epochs, and τ and n_k are hyperparameters. Therefore, we select samples

$$\tilde{B}_i = \underset{|\tilde{B}_i|=R(n)|B|}{arg\min}\ \ell(i, B) \tag{3}$$

at every iteration for network training.

3.2 Model Uncertainty Estimation

Experiments have shown that deep neural networks frequently suffer from erroneous predictions with high confidence [11, 25]. To remedy this problem in practice, methods were presented to estimate whether networks have learned how to handle a testing sample or they just guess the solution. The idea is based on the theory of Bayesian neural networks, and the output distribution can be used to quantify model uncertainty.

While a standard network considers the input sample and determines one predictive value, the Bayesian network computes a prediction distribution by

$$p(y|X, \mathcal{D}) = \int p(y|X, \theta)p(\theta|\mathcal{D})d\theta, \tag{4}$$

where X and y are the input sample and the corresponding label, respectively, θ indicates the network parameters, and $\mathcal{D}$ is the data set. Specifically, $p(y|X, \theta)$ tells us how well the network parameters θ explain the observation. The prediction $p(y|X, \mathcal{D})$ considers all possible parameter configurations weighted by the

their posterior probabilities $p(\theta|\mathcal{D})$. Apparently, computing the exact posterior distribution $p(\theta|\mathcal{D})$ is difficult due to the complexity of deep neural networks. In practice, the distribution can be approximated by training multiple models, and the disagreement of the models yields the uncertainty [17].

Bayesian Approximation. We adopt the method in [2] and exploit the network's depth to quantify uncertainty because of its accurate approximation and computational efficiency. The main idea is to add an auxiliary output layer at the back of each intermediate block. Then, we view each output as a sub-network's prediction result. This enables us to estimate uncertainty by checking the consistency of the predictions. An intuitive explanation to this Bayesian approximation is that the sub-networks' outputs would be consistent if they have learned the sample. Otherwise, the outputs would be diverse. To perform a Bayesian inference on the depth of a neural network, we consider the network's depth to be a random variable. Let a categorical prior over network's depth d be $p_\beta(d=i)=\beta_i=1/D$, where i is the layer index and D is the number of network layers. The marginal log likelihood of the model can be written as follows:

$$\log p(\mathcal{D};\theta)=\log\sum_{i=1}^{D}\left(p_\beta(d=i)\cdot\prod_{t=1}^{N}p(y_t|\mathbf{X}_t,d=i,\theta)\right), \tag{5}$$

where θ indicates the network parameters and $p(y|\mathbf{X},d=i,\theta)$ is the output of the i$^{\text{th}}$ layer. The posterior over depth, $p(d|\mathcal{D};\theta)=p(\mathcal{D}|d;\theta)p_\beta(d)/p(\mathcal{D};\theta)$, represents how well each layer explains the data.

Training. An intuitive approach to train the network is to maximize the marginal log likelihood through stochastic gradient descent. However, the approach does not work because each layer is weighted by the depth posterior. It leads gradients to vanish when the posterior collapses to a delta function. Therefore, we adopt the strategy in [2] and optimize the network by stochastic gradient variational inference. Let $q_\alpha(d=i)=\alpha_i$ be an approximate posterior. The evidence lower bound (ELBO) is written as

$$\log p(\mathcal{D};\theta)=\sum_{t=1}^{N}\mathbb{E}_{q_\alpha(d)}[\log p(y_t|\mathbf{X}_t,d;\theta)]-KL(q_\alpha(d)||p_\beta(d)).$$

Let B and N be the batch size and the dataset size, respectively. We maximize the ELBO of each mini-batch by using

$$\frac{N}{B}\sum_{t=1}^{B}\sum_{i=1}^{D}(\log p(y_t|\mathbf{X}_t,d=i;\theta)\cdot\alpha_i)-\sum_{i=1}^{D}\left(\alpha_i\log\frac{\alpha_i}{\beta_i}\right). \tag{6}$$

Inference. After network training, we marginalize depth with the variational posterior to predict price movements. For each new sample $\mathbf{X}^*$, the prediction is computed using Bayesian model averaging [21]:

$$p(y^*|\mathbf{X}^*,\mathcal{D};\theta)=\sum_{i=1}^{D}p(y^*|\mathbf{X}^*,d=i;\theta)q_\alpha(d=i). \tag{7}$$

Regarding the uncertainty of the network's output, we implement predictive entropy and variance because they are common measures of the uncertainty inherent in a distribution of possible outcomes. Then, we test their performances on predicting stock price movements. Specifically, predictive entropy is defined as

$$\mathbb{H}(y^*|\mathbf{X}^*, \mathcal{D}; \theta) = \sum_{c=0}^{1} p(y^* = c|\mathbf{X}^*, \mathcal{D}; \theta) \log p(y^* = c|\mathbf{X}^*, \mathcal{D}; \theta) \tag{8}$$

where $c \in \{0, 1\}$ is the class label. Moreover, the prediction's variance is formulated as

$$\sigma^2(y^* = 0|\mathbf{X}^*, \mathcal{D}; \theta) = \mathbb{E}_{q_\alpha(d)} \left(p(y^* = 0|\mathbf{X}^*, d; \theta)^2 \right) - \mathbb{E}_{q_\alpha(d)} \left(p(y^* = 0|\mathbf{X}^*, d; \theta) \right)^2 . \tag{9}$$

For a binary classification problem $p(y^* = 0|\mathbf{X}^*, d; \theta) = 1 - p(y^* = 1|\mathbf{X}^*, d; \theta)$ and $\sigma^2(y^* = 0|\mathbf{X}^*, \mathcal{D}; \theta) = \sigma^2(y^* = 1|\mathbf{X}^*, \mathcal{D}; \theta)$. Therefore, we only compute $\sigma^2(y^* = 0|\mathbf{X}^*, \mathcal{D}; \theta)$.

3.3 Implementation Details

Features. Recall that the input of our network is a tensor $\mathbf{X}_t = [x_{t-T+1}, ..., x_t] \in \mathcal{R}^{F \times T \times W}$, where F is the number of features, T indicates the past period that the network considers, and W is the number of moving averages. In our implementation, $F = 6$, $T = 30$, and $W = 4$. The feature x_t is composed of six price attributes, including *high*, *low*, *open*, *close*, *adjust close*, and *volume*. For each attribute, we computed the moving average according to four window sizes. Specifically, the attributes of the past 1, 3, 7, and 14 days were averaged. We also normalized the values in each sample individually to [0, 1] in the pre-processing stage.

Network architecture. We built a convolutional neural network (CNN) to forecast stock price movements. The network contained four convolutional layers and then two fully connected layers. The first three convolutional layers consider features along the temporal dimension, whereas the fourth layer exchanges information at the same period. In addition, we link the output of every convolutional layer to the shared fully connected layers to implement the depth uncertainty network. Figure 1 illustrates the detailed network architecture.

Parameters and Network training. We initialized network parameters using the Xavier initialization and set the batch size and the learning rate to 4096 and 0.003 for the *KDD17* dataset, and 1024 and 0.001 for the *ACL18* and our collected datasets, respectively. Since price movement data of an individual stock are rare, we followed the strategy in previous works and mixed samples of different stocks for training networks. The AdamW [20] optimizer was adopted to minimize binary cross-entropy loss. We repeated the training process for 300 epochs in our implementation. Subsequently, we selected the model that performed the best on the validation data and tested it on the testing data for evaluation.

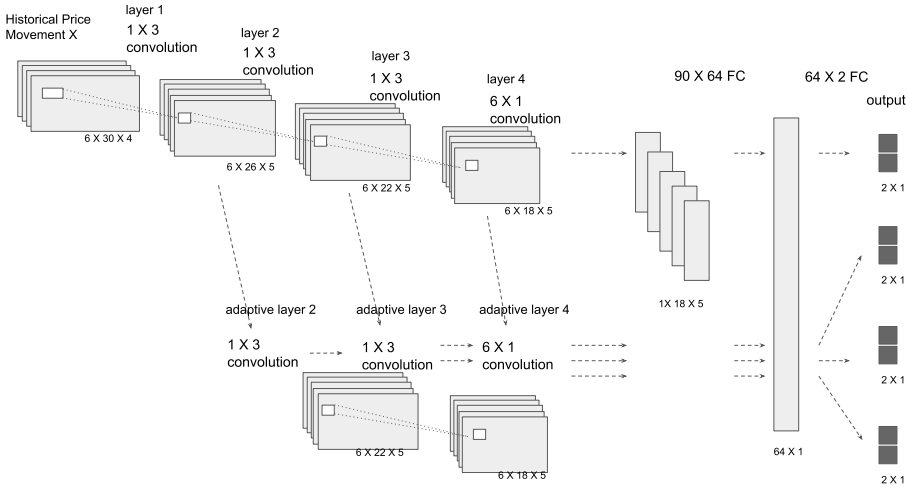

Fig. 1. The network architecture used in our study. Each intermediate layer has an auxiliary output. The adaptive layer is set to adjust the resolution. By exploiting the network's depth, we obtain the approximation of a Bayesian inference by one forward.

4 Results and Evaluations

4.1 Comparison to Baselines

We evaluated the performance of our method on five datasets: the first two are benchmark datasets *ACL18* [34] and *KDD17* [37], and the last three are our collected datasets in which the testing periods are *bull*, *flat*, and *bear* markets, respectively. We selected the stocks from the U.S. stock market that have the highest trading volumes for evaluation. Specifically, they are GOOG, NVDA, AMZN, AMD, QCOM, INTC, MSFT, AAPL, and BIDU. Table 1 shows the description, and the ranges used for training, validation, and testing. Let x_t and x_{t+1} be the closing prices of consecutive days. We computed the movement percent by $p_{t+1} = (x_{t+1}/x_t) - 1$. Typically, the label $y_{i+1} = 1$ if $p_{t+1} > 0$ and $y_{i+1} = 0$ otherwise. However, to compare with the state-of-the-art methods, we followed the setting of [8] and additionally defined the label $y_{i+1} = 1$ if $p_{t+1} \geq 0.55\%$ and $y_{i+1} = 0$ if $p_{t+1} \leq -0.5\%$. In other words, the samples with flat price movements were ignored in the experiment.

Baselines. We briefly describe the baselines in the following paragraph. Two of them were traditional technical analysis, and the others were the latest neural networks.

– Time Series Momentum Strategy (MOM) [23] was based on the belief that the current market trend will continue by taking the sign of returns over the last period. We used the trend in the last 10 days as the momentum indicator.

Dataset	# stocks	Training	Validation	Testing
ACL18	88	Jan-01-2014- Aug-01-2015	Aug-01-2015- Oct-01-2015	Oct-01-2015 Jan-01-2016
KDD17	50	Jan-01-2007- Jan-01-2015	Jan-01-2015- Jan-01-2016	Jan-01-2016 Jan-01-2017
Bull	9	Oct-20-2006 - Jun-20-2012	Jun-21-2012 - Mar-10-2013	Mar-11-2013 - Nov-20-2013
Flat	9	Jun-19-2003- Jan-03-2010	Jan-04-2010- Oct-25-2010	Oct-26-2010- Aug-19-2010
Bear	9	Jan-20-2001- Sep-26-2007	Sep-27-2007- Jul-23-2008	Jul-24-2008- May-20-2009

Table 1. The number of stocks and the data ranges for training, validation, and testing in our experiments.

- On-Line Moving Average Reversion (OLMAR) [18] was based on the belief that prices will revert to the long-term mean by taking the opposite sign of the difference between the last price and the moving average. We used the 30 days moving average as the mean reversion indicator.
- StockNet [34] was a variational autoencoder exploiting text and price signals to capture the market stochasticity. In the comparison, we discarded the text signals because they are often unavailable in practice.
- ADV-ALSTM [7] applied adversarial training to improve the generalization of a prediction model. They generated additional samples by adding small perturbations on input features and trained the model on both the origin and perturbed samples.

	Removing flat		Containing flat	
	ACL18	*KDD17*	*ACL18*	*KDD17*
MOM	46.61	49.12	47.92	48.84
OLMAR	52.7	49.85	52.21	50.23
StockNet	54.96	51.93	-	-
ADV-ALSTM	**57.2**	53.05	52.03	51.95
Ours	54.96	**53.49**	**53.5**	**52.69**
ADV-ALSTM (10%)	55.37	53.58	50.54	54.26
Ours (10%)	**64.98**	**55.81**	**60.8**	**55.37**

Table 2. Mean accuracy of the baselines on the *ACL18* and *KDD17* datasets. All of the methods performed only slightly better than a random guess. However, the mean accuracy considerably increases if the top 10% certain samples are evaluated. We highlight the highest accuracy in boldface.

Experiment Results. Table 2 shows the comparison results on *ACL18* and *KDD17* datasets. The reported statistics were the mean testing accuracy

Fig. 2. We compared our method to ADV-LSTM-Dropout on our collected datasets. The testing periods are bull, flat, and bear markets, respectively. We report the prediction accuracy from the cumulative lowest to highest uncertain samples for evaluation.

of five different runs. From the numbers, one can realize that predicting stock price movements is extremely difficult. Even the state-of-the-art, ADV-LSTM [7], achieved an accuracy slightly higher than a random guess. Since, to the best of our knowledge, there are no stock price prediction methods built upon deep neural networks and consider uncertainty, we extend ADV-LSTM and selected the top 10% certain samples for comparison. Because ADV-LSTM's network cannot exploit depth structures for uncertainty estimation, we apply the dropout approach to approximate the posterior inference [9]. The final result is the average of its five predictions. We named the extended ADV-LSTM as ADV-LSTM-Dropout. In our implementation, we set the dropout rate to 0.8; we also repeated the experiments five times to estimate the output's variance since our network contains five branches. As indicated in Table 2, our selected top 10% certain samples have higher prediction accuracy than the remains, whereas the samples selected by ADV-LSTM do not. We suspect the reasons could be: (1) dropout reduces the prediction accuracy during inference, and (2) the network cannot learn effective knowledge from the training data that contain aleatoric uncertainty. However, without theoretical proofs and thorough experiments, they may not truly explain the phenomenon.

We additionally compared our method with ADV-LSTM-Dropout regarding different degrees of uncertainty on our collected datasets. The testing periods are bull, flat, and bear markets, respectively. We mixed all stocks in the testing data and ranked the data based on the uncertainty of each sample in ascending order. The mean accuracy of the cumulative lowest 10% to 100% uncertain samples was computed. As indicated in Figure 2, the mean accuracy of our results gradually decreased as the uncertainty increased, which implies that our network can achieve higher accuracy when it is more certain about its prediction. Although ADV-LSTM-Dropout had a similar trend, it was not as clear as ours.

In the following sections, we evaluated the system on the whole testing set (i.e., containing flat movements) since removing them is unachievable in practice. Namely, $y_{i+1} = 1$ if $p_{t+1} > 0$ and $y_{i+1} = 0$ otherwise.

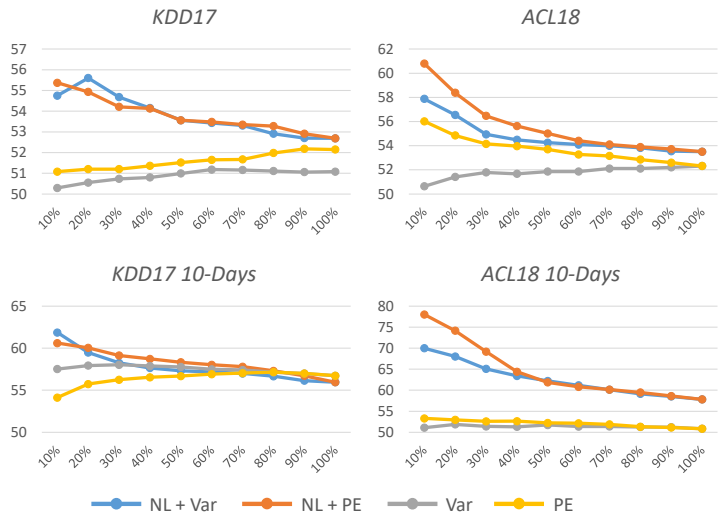

Fig. 3. The mean prediction accuracy of networks trained on a variety of conditions. We evaluated the accuracy of predictions from the cumulative lowest to highest uncertain samples. *NL*, *Var*, and *PE* are the abbreviations of noisy label training, uncertainty estimated by variance, and uncertainty estimated by predictive entropy, respectively. The line charts indicate the effectiveness of uncertainty awareness prediction. This phenomenon appeared both in the predictions of the next day and the next ten days.

4.2 Uncertainty Estimation using Variance and Predictive Entropy

We estimated the prediction uncertainty by computing the output's variance and predictive entropy. To evaluate the performance of these two strategies, we compare the accuracy regarding different degrees of uncertainty. In this experiment, we trained the network to predict the price movement of the next day and the next 10 days and plotted the results in Figure 3. All stocks in the testing data are mixed and then ranked for evaluation. As indicated, the mean accuracy of the stocks gradually decreased as the uncertainty increased. This phenomenon appeared both in predictions of the next day and the next 10 days, irrespective of whether variance or predictive entropy was used to represent uncertainty. Overall, the predictive entropy slightly performs better than the variance on the *ACL18* dataset, yet this advantage is not clear on the *KDD17* dataset.

In addition to mixing all stocks together, we evaluated each stock's mean prediction accuracy under the ranking of uncertainty. Let h_r^i be the mean accuracy of the $r\%$ lowest uncertain samples of stock i, where $r = \{0.1, 0.2, ..., 1.0\}$. We compute the Pearson correlation coefficient between r and h_r^i of each stock. The coefficient ranges from -1 to 1, which indicates negative and positive correlations, respectively. Figure 4 shows the correlation histogram results on the *KDD17*[37] and *ACL18* [34] datasets. The x and y axes represent the Pearson correlation coefficient and the number of stocks within each range, respectively.

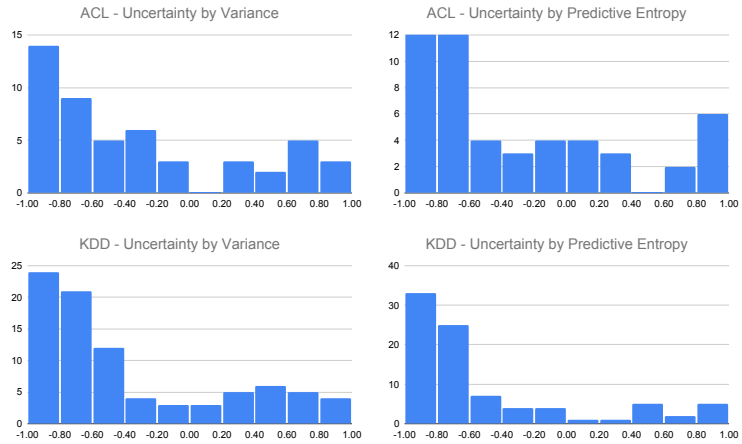

Fig. 4. The Pearson correlation coefficient between prediction uncertainty and accuracy. The x and y axes represent the range of the coefficient and the frequency, respectively. Clearly, in most stocks, low prediction uncertainty leads to high accuracy.

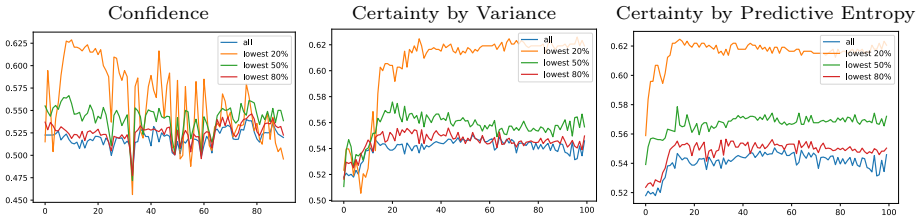

Fig. 5. Comparison of the mean accuracy of the samples selected according to confidence and two variants of certainty estimations. The lowest 20%, 50%, and 80% non-confident and uncertain samples, and all samples, in the testing data set during network training, were compared.

As indicated, most of the stocks have negative coefficients, which fulfilled the expectation, i.e., the lower is the uncertainty, the higher is the accuracy. Different from the results of mixing all stocks together, the variance in this experiment performs slightly better than the predictive entropy since samples in the bin of 0.8-1.0 are few.

4.3 The Effectiveness of Removing Aleatoric Uncertainty

We remove samples with aleatoric uncertainty by applying the noisy label training. In addition to the benefits of uncertainty quantification, this strategy improves network's generalization because it was not forced to memorize samples with inconsistent labels. To evaluate the effectiveness, we trained the networks with and without removing inconsistent samples and compared their predictive

accuracy on the benchmark datasets. The lines in Figure 3 verify that removing aleatoric uncertainty was effective. The improvement was about 1% accuracy if the whole testing set is evaluated. It is worth noting that the testing data were noisy. Even though the trained network achieved perfect generalization, it was not guaranteed to predict price movements accurately. In addition, removing samples that have inconsistent labels could advance uncertainty awareness of network predictions. The experiment results revealed that networks trained to fulfill all training data did not learn how to predict price movements. Their prediction accuracy was still low even though they were certain about the predictions of the selected samples.

Removing aleatoric uncertainty helps networks learn price trends in a dataset. Since samples with inconsistent labels have been removed, the network will be unfamiliar with them during testing. This strategy makes the network be certain about only the samples similar to those appearing in the training data and having consistent labels. That is why our system can considerably increase accuracy when the network is certain about its prediction.

4.4 Confidence v.s. Certainty

Neural networks output the probability of each label to represent their prediction. Since high probability is often interpreted as high confidence, we compare the relationship of these two measures to accuracy. Because the predicted probability in our implementation was a vector weighted from several outputs (Equation 7), which was an ensemble, we additionally trained a standard network that has the same depth, but did not contain branches, for the comparison. The parameters, such as kernel size, activation functions, and numbers of channels were unchanged. Let the output of this standard network be a 2D vector (p_ℓ, p_f), where p_ℓ and p_f are the probability of rise and fall, respectively, $0 \leq p_\ell, p_f \leq 1$, and $p_\ell + p_f = 1$. We computed the non-confidence of a prediction by $\min(1 - p_\ell, 1 - p_f)$. Overall, low values indicate high confidence. We then ranked the testing data based on the non-confidence of each sample in ascending order, and computed the mean accuracy of the lowest 20%, 50%, and 80% non-confident, and all samples. The accuracy of samples selected according to confidence and certainty estimations was compared.

Figure 5 shows the mean test accuracy during network training. As indicated, the accuracy lines of different uncertainties are clearly separated, while the lines of different non-confidence are not. Moreover, the high certainty samples enjoy high accuracy. This means that sample selection based on certainty was markedly reliable. It is also worth noting that the selected samples in each epoch were different because the network changed when training.

4.5 Compatibility to Other Network Structures

Our uncertainty framework is compatible with all types of network structures. In our implementation, we build a CNN with multiple branches to approximate a Bayesian network because of the balance between accurate approximation and

computational efficiency. The Bayesian approximation, however, can be achieved by ensemble models [17] or dropouts [10, 9]. Regarding the noisy label training, we utilize two networks with the same structure to remove samples with inconsiste nt labels [12]. This training strategy is irrelevant to network structures. Therefore, new uncertainty measures and noisy label training methods can be seamlessly integrated into our system to improve prediction accuracy further.

4.6 Limitations and Future Works

Although our experimental results indicate that considering uncertainty when forecasting stock price movements is effective, there is still a large space for improvement. After all, financial markets are noisy and uncertain. In addition, the amounts of daily stock price data are rare. While ancient samples could be useless for predicting future price movements, removing samples with aleatoric uncertainty would further reduce the data size. In our implementation, we mixed samples of different stocks to enlarge the data set. However, since stocks are of different properties, networks trained on the mixed data could be insufficient to predict the price movements of a specific stock. Therefore, in the future, we will consider stock relationships to overcome the problems caused by data mixing.

5 Conclusions

In this study, we consider aleatoric and model uncertainty when predicting stock price movements. The uncertainty estimation allows the network to know how certain the prediction is, which provides a high degree of freedom for investment strategies. While estimating aleatoric uncertainty during inference is challenging, we transform aleatoric uncertainty to model uncertainty by removing samples with inconsistent labels. The Bayesian inference is then applied to evaluate whether networks are familiar with a testing sample for estimating model uncertainty. In addition, removing samples with aleatoric uncertainty improves the network's generalization because networks only learn from samples with consistent labels. They are only certain about samples that have been seen in the training set and contain consistent labels. Experiment results verified that the high certainty of our network prediction indeed results in high accuracy.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work was supported by E.SUN Bank and the Ministry of Science and Technology, Taiwan (110-2221-E-A49 -062 - and 109-2221-E-009 -097 -).

References

1. Adebiyi, A.A., Adewumi, A.O., Ayo, C.K.: Comparison of arima and artificial neural networks models for stock price prediction. Journal of Applied Mathematics **2014** (2014)

2. Antorán, J., Allingham, J., Hernández-Lobato, J.M.: Depth uncertainty in neural networks. Advances in Neural Information Processing Systems **33** (2020)
3. Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M.S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y.: A closer look at memorization in deep networks. In: International Conference on Machine Learning. pp. 233–242 (2017)
4. Ding, X., Zhang, Y., Liu, T., Duan, J.: Using structured events to predict stock price movement: An empirical investigation. In: Conference on Empirical Methods in Natural Language Processing. pp. 1415–1425 (2014)
5. Dixon, M., Klabjan, D., Bang, J.H.: Classification-based financial markets prediction using deep neural networks. Algorithmic Finance **6**(3-4), 67–77 (2017)
6. Edwards, R.D., Magee, J., Bassetti, W.C.: Technical analysis of stock trends. CRC press (2018)
7. Feng, F., Chen, H., He, X., Ding, J., Sun, M., Chua, T.S.: Enhancing stock movement prediction with adversarial training. In: International Joint Conference on Artificial Intelligence. pp. 5843–5849 (2019)
8. Feng, F., He, X., Wang, X., Luo, C., Liu, Y., Chua, T.S.: Temporal relational ranking for stock prediction. ACM Transactions on Information Systems **37**(2), 1–30 (2019)
9. Foong, A.Y., Burt, D.R., Li, Y., Turner, R.E.: Pathologies of factorised gaussian and mc dropout posteriors in bayesian neural networks. stat **1050**, 2 (2019)
10. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: International conference on machine learning. pp. 1050–1059 (2016)
11. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015), http://arxiv.org/abs/1412.6572
12. Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., Sugiyama, M.: Co-teaching: Robust training of deep neural networks with extremely noisy labels. In: Advances in neural information processing systems. pp. 8527–8537 (2018)
13. Heaton, J.B., Polson, N.G., Witte, J.H.: Deep learning for finance: deep portfolios. Applied Stochastic Models in Business and Industry **33**(1), 3–12 (2017)
14. Izmailov, P., Maddox, W.J., Kirichenko, P., Garipov, T., Vetrov, D., Wilson, A.G.: Subspace inference for bayesian deep learning. In: Uncertainty in Artificial Intelligence. pp. 1169–1179 (2020)
15. Jiang, L., Zhou, Z., Leung, T., Li, L.J., Fei-Fei, L.: Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In: International Conference on Machine Learning (2018)
16. Kurakin, A., Goodfellow, J.I., Bengio, S.: Adversarial machine learning at scale. international conference on learning representations (2017)
17. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. In: Advances in neural information processing systems. pp. 6402–6413 (2017)
18. Li, B., Hoi, S.C.: On-line portfolio selection with moving average reversion. arXiv preprint arXiv:1206.4626 (2012)
19. Li, M., Soltanolkotabi, M., Oymak, S.: Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In: International Conference on Artificial Intelligence and Statistics (2020)
20. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)

21. Maddox, W., Garipov, T., Izmailov, P., Vetrov, D., Wilson, A.G.: A simple baseline for bayesian uncertainty in deep learning. arXiv preprint arXiv:1902.02476 (2019)
22. Malach, E., Shalev-Shwartz, S.: Decoupling" when to update" from" how to update". In: Advances in Neural Information Processing Systems. pp. 960–970 (2017)
23. Moskowitz, T.J., Ooi, Y.H., Pedersen, L.H.: Time series momentum. Journal of financial economics **104**(2), 228–250 (2012)
24. Nelson, D.M., Pereira, A.C., de Oliveira, R.A.: Stock market's price movement prediction with lstm neural networks. In: International joint conference on neural networks. pp. 1419–1426 (2017)
25. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: IEEE conference on computer vision and pattern recognition. pp. 427–436 (2015)
26. Nguyen, T.H., Shirai, K., Velcin, J.: Sentiment analysis on social media for stock movement prediction. Expert Systems with Applications **42**(24), 9603–9611 (2015)
27. Niaki, S.T.A., Hoseinzade, S.: Forecasting s&p 500 index using artificial neural networks and design of experiments. Journal of Industrial Engineering International **9**(1), 1 (2013)
28. Ntakaris, A., Magris, M., Kanniainen, J., Gabbouj, M., Iosifidis, A.: Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. Journal of Forecasting **37**(8), 852–866 (2018)
29. Ren, M., Zeng, W., Yang, B., Urtasun, R.: Learning to reweight examples for robust deep learning. In: International Conference on Machine Learning (2018)
30. Shaker, M.H., Hüllermeier, E.: Aleatoric and epistemic uncertainty with random forests. In: International Symposium on Intelligent Data Analysis. pp. 444–456. Springer (2020)
31. Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., Iosifidis, A.: Using deep learning to detect price change indications in financial markets. In: European Signal Processing Conference. pp. 2511–2515 (2017)
32. Walczak, S.: An empirical analysis of data requirements for financial forecasting with neural networks. Journal of management information systems **17**(4), 203–222 (2001)
33. Wei, H., Feng, L., Chen, X., An, B.: Combating noisy labels by agreement: A joint training method with co-regularization. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 13726–13735 (2020)
34. Xu, Y., Cohen, S.B.: Stock movement prediction from tweets and historical prices. In: Annual Meeting of the Association for Computational Linguistics. pp. 1970–1979 (2018)
35. Yu, X., Han, B., Yao, J., Niu, G., Tsang, I.W., Sugiyama, M.: How does disagreement help generalization against label corruption? arXiv preprint arXiv:1901.04215 (2019)
36. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530 (2016)
37. Zhang, L., Aggarwal, C., Qi, G.J.: Stock price prediction via discovering multi-frequency trading patterns. In: ACM SIGKDD international conference on knowledge discovery and data mining. pp. 2141–2149 (2017)
38. Zhang, Z., Zohren, S., Roberts, S.: Deeplob: Deep convolutional neural networks for limit order books. IEEE Transactions on Signal Processing **67**(11), 3001–3012 (2019)