Stock Selection Framework Using Fundamental Analysis & Deep Learning*

孫瑞遙[1] and 郭慧芸[1]

[2]JP Morgan Chase

Abstract— There are a great deal of recent works in an attempt to solve time series classification (TSC) problem. Among these works, however, very few of them look for linkages to financial application especially in portfolio allocation. The proposed framework is specifically designed for addressing stock picking and portfolio allocation problem using long-term fundamental information and short-term time series classification with multi-scale convolutional neural network (MCNN). In general, the result of our framework significantly outdistances other compared benchmark.

I. INTRODUCTION

Stock picking and portfolio allocation problems have long been researched extensively. However, there have not been many research efforts to bring the domain knowledge of finance into deep learning approach. In this research, we introduce a framework specifically designed for addressing stock picking and portfolio allocation problems. Our framework are made up of long-term fundamental information and short-term time series classification with multi-scale convolutional neural network (MCNN). In addition, it has an adaptive way to update its hyperparameters automatically based on the current performance of portfolio. Through the comparison of both return and window-moving sharpe ratio, we show that our proposed framework steadily outperforms two benchmarks.

II. RELATIVE WORK

A. Multi-Scale Convolutional Neural Network

Multi-Scale Convolutional Neural Networks (MCNN) includes three stages: transformation stage, local convolution stage, and full convolution stage [1] [2]. It's this structure that makes MCNN have ability to extract multi-scale and multi-frequency features automatically. An overview of MCNN is illustrated in Fig 1. In time series classification (TSC) problem, recent works have shown that MCNN is an efficient approach to learn the general representations in the time series and thus advances the state-of-the-art [1] [2] [5].

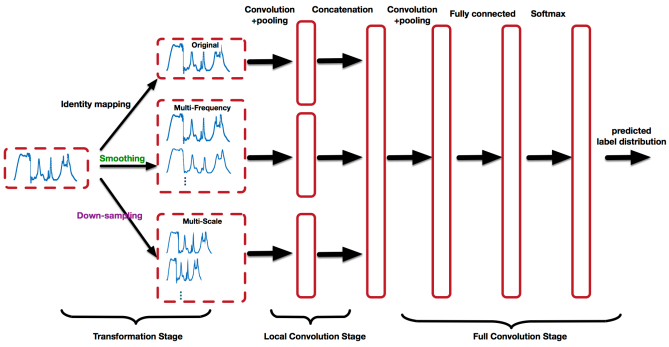

Fig. 1. MCNN Architecture

III. PORTFOLIO ALLOCATION with MCNN

Our framework starts from two phases. Phase I addresses the problem "given an investment universe, how to pick stocks with strong fundamentals by the comprehensive evaluation of fundamental information and its performance." On the other hand, phase II deals with the problem "given an investment universe, how to establish portfolio allocation based on the result of time series classification."

A. Phase I : fundamental analysis with ML

Phase I follows procedures in Fig 2. The quarterly fundamental data used in this phase would be obtained from Compustat and composed of 500 component stocks in S&P 500 index from January 2000 to March 2017.

Fig. 2. Phase I Process

On the basis of fundamental data, nine fundamental indicators listed below are computed for stock evaluation [6].

- Net Working Capital
- Retained Earnings / Total Assets
- Return on Total Assets
- Market Value of Equity / Total Liabilities
- Asset Turnover Ratio
- P/E Ratio
- Return on Equity
- Dividend Yield
- Current Ratio

However, our desired framework is on a monthly basis not quarterly; thus, except for stock prices which already have monthly time frame, the gaps in fundamental data between each financial statements announcement, usually three months or a quarter, should be filled forward up as Fig 3. Hence, it is now possible to construct monthly fundamental indicators for each stock, and the value of indicators related to stock price would vary from month to month.

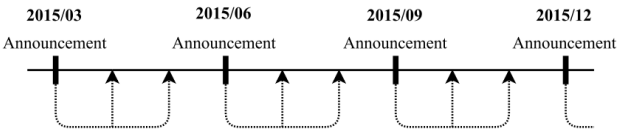

Fig. 3. Forward Filling financial statements gaps

1) Exponentially Weighted Moving Average (EWMA): For fundamental analysis with a long-term focus, one-quarter indicators tend to be sensitive to business cycle and then become noisy signals. To address this problem, exponentially weighted moving average (EWMA) is applied on each indicator, which involves averaging past observations with different weights. As the weighted mean of observation Y at time t and $EWMA$ at time $t-1$, a common form of the EWMA for time t is defined as:

$$EWMA_t = \lambda Y_t + (1-\lambda) EWMA_{t-1} \tag{1}$$

where $0 \leq \lambda \leq 1$, $t = 1, 2, ..., n$. Note that the time step here is three months for each t in our framework because of the monthly-filling preprocessing. It seems reasonable to emphasize on recent past rather than distant past on account of the characteristics of fundamentals. The number of observations in window $n = 12$, which stands three years, and $\lambda = 60/(60+1)$.

2) Class labels: Now we have the weighted fundamental indicators for each stock in each month. Aiming for providing a "link" between fundamentals and stock performance, we design our targets using 3-year stock return; then split into 10 classes according to the order of return. Given time t, stocks in class 1 indicate their returns are top 10% among 500 components at time t. Similarly, stocks in class 10 represents bottom 10%. Finally, these classes would be divided into four ranks for later use. Rank 1 is made up of classes 1, 2 and 3. Rank 2 is made up of classes 4 and 5. Rank 3 is made up of classes 6, 7 and 8. Rank 4 is made up of classes 9 and 10.

3) Train-test split: For machine learning purpose, our data would be split into training and testing set, and the time window would be 10 years and 1 year separately. The model would be annually update. That is, for example, if training set starts from 2001 to 2010, data in 2011 would be testing set, and so on.

B. Phase II : PORTFOLIO ALLOCATION with MCNN

1) MCNN and probability weighting: In time series classification (TSC) problem, recent works have shown that MCNN is an efficient approach to extract meaningful features and has ability to learn the general representations in the time series. However, among these works, very few of them look for linkages to financial application especially in portfolio allocation. Our proposed method incorporates fundamental information in phase I and MCNN into a single framework based on probability weighting to address portfolio allocation problem.

In this phase, the output of MCNN $O_i = \{u_i, c_i, d_i\}$ is the probability corresponding to three categories: uptrend, consolidation and downtrend for a time series T_i, where $u_i + c_i + d_i = 1$. And probability weighting based on the probability output in MCNN reflects the view of MCNN toward relatively short-term movement of a time series. Moreover, it has a flexibility to control the degree of the weight adjustment by our confidence toward MCNN.

To begin with, in phase I, given time t and a rank r, there are total n_t^r stocks picked in portfolio for time $t+1$. The weights of portfolio are initiated with equally weight: $W = \{W_1, W_2, ..., W_{n_{P1}}\}$, where $0 \leq W_i \leq 1$ and $W_i = 1/n_{P1}$.

For each MCNN output O_i, $Score_i$ is defined as:

$$Score_i = (u_i - d_i) + (u_i - c_i) \tag{2}$$

where $-1 \leq Score_i \leq 2$. *Score* represents the view of MCNN toward the future movement of time series T. It would then be applied exponential-transformation in order to add nonlinearity, and the hyperparameter σ is introduced to increase the degree of nonlinearity, which make those have high *Score* much higher. That is, $Shift$:

$$Shift_i = \exp(\sigma * Score_i) \tag{3}$$

However, the scale of $Shift$ would be different from *Score* due to the nonlinearity adjustment. To make the effect of σ regularized, it is necessary to do min-max normalization and shifting to transform $Shift$ into $Shift^{norm}$, where $-1 \leq Shift^{norm} \leq 1$. The comparison between $Shift$ and $Shift^{norm}$ for different values of σ is shown in Fig 4. Finally, the effect of $Shift^{norm}$ on the weight W is controlled by the hyperparameter τ, which reflects the confidence toward the weight adjustment from the output of MCNN.

$$W_i = max(0, W_i + \tau * Shift_i^{norm}) \tag{4}$$

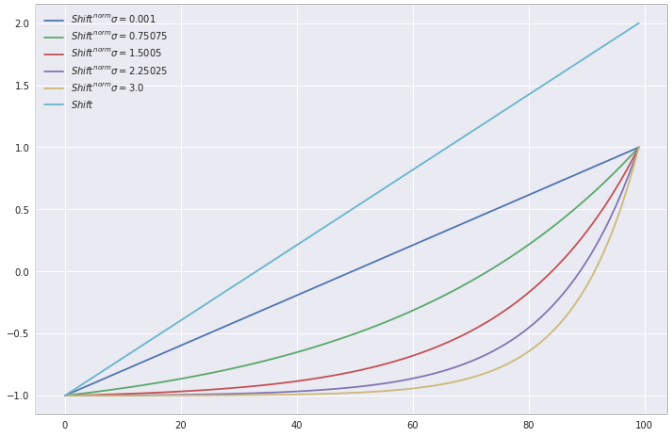

Fig. 4. Normalization & σ

As shown in Fig 5, larger σ leads weights to concentrate to higher *Score*, which are the stocks that might have tendency to go uptrend, and the effect of σ is increasing due to the characteristics of exponential functions.

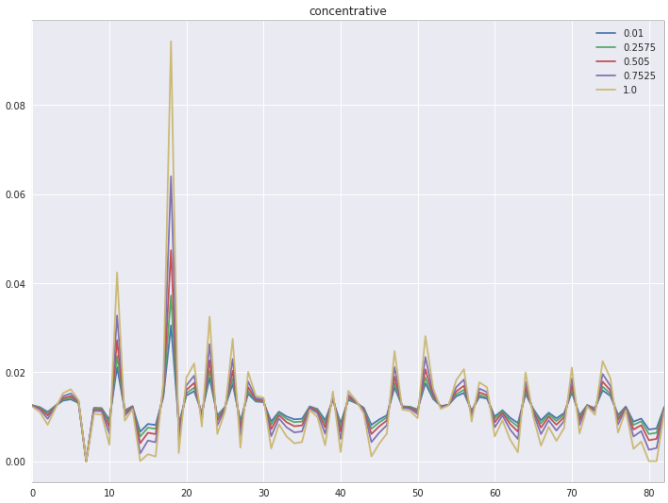

Fig. 5. σ sensitivity

On the other hand, the hyperparameter τ is the reflection of our confidence toward the output of MCNN. Apparently, if $\tau = 0$, it is nothing but the equally-weighting portfolio in phase I. The lager τ, the more confidence toward our model, which makes the effect of $Shift^{norm}$ on weights more significant, as shown in Fig 6. However, the effect of τ is diminishing due to the constraint on weights: $0 \leq W \leq 1$. Fig 7 shows the process of our framework for solving the portfolio allocation problem.

Fig. 6. τ sensitivity

2) Train-test split: The time series data from 2005 to 2010 is the base training set because our evaluation starts from 2011. It's intuitive to train and update MCNN model as same as Phase I do. However, MCNN

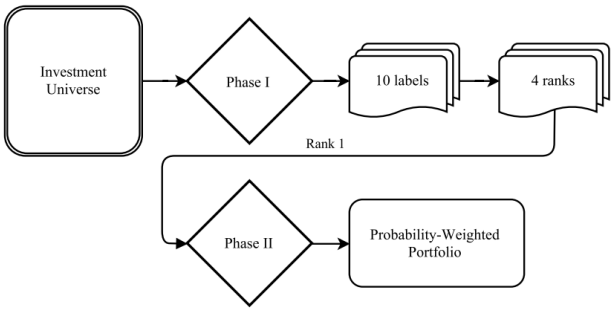

Fig. 7. Workflow in proposed framework

Algorithm 6 RMSprop with Nesterov momentum

Require: Global learning rate η, decay rate ρ, momentum coefficient α
Require: Initial parameter θ, initial velocity v
Initialize accumulation variable $r = 0$
while Stopping criterion not met **do**
  Sample a minibatch of m examples from the training set $\{x^{(1)}, \ldots, x^{(m)}\}$.
  Compute interim update: $\theta \leftarrow \theta + \alpha v$
  Set $g = 0$
  for $i = 1$ to m **do**
    Compute gradient:

$$g \leftarrow g + \nabla_\theta L(f(x^{(i)}; \theta)), y^{(i)}; \theta).$$

  end for
  Accumulate gradient: $r \leftarrow \rho r + (1 - \rho) g^2$
  Compute velocity update: $v \leftarrow \alpha v - \frac{\eta}{\sqrt{r}} g$ ($\frac{1}{\sqrt{r}}$ applied element-wise)
  Apply update: $\theta \leftarrow \theta + v$
end while

Fig. 8. RMSprop with Nesterov momentum

is a deep model with more complicated structure and more cost to retrain. To make our model both adaptable and efficient, instead of retraining the model, MCNN in Phase II would constantly update its weights annually just like what online learning do. When collecting enough new training data, MCNN runs mini-batch update with new data in relatively less epochs than the base training period to avoid over fitting.

3) Adaptive hyperparameter adjustment: It's no doubt that the situation of financial market varies from time to time, and so does the accuracy of our framework even with constantly updating. Thus, the values of both hyperparameters σ and τ, reflecting the degree of confidence toward MCNN, should not be treated as constants. The purpose of this part is to make these hyperparameters more dynamic and well-adapted to the current market. Here, we borrow ideas from a well-known optimization algorithm in deep learning called "RMSprop with Nesterov momentum" [3] [4] presented in Fig 8:

The gradient part in the algorithm is modified by using the difference of recent sharpe ratio between probability weighting portfolio, equally weighting portfolio, and SPY. With this modified algorithm, our framework would have ability to monthly update hyperparameters based on its performance. If probability weighting portfolio steadily outperforms equally weighting one and SPY, both hyperparameters will increase gradually, and vice versa. The whole updating process could be seen in Fig 9:

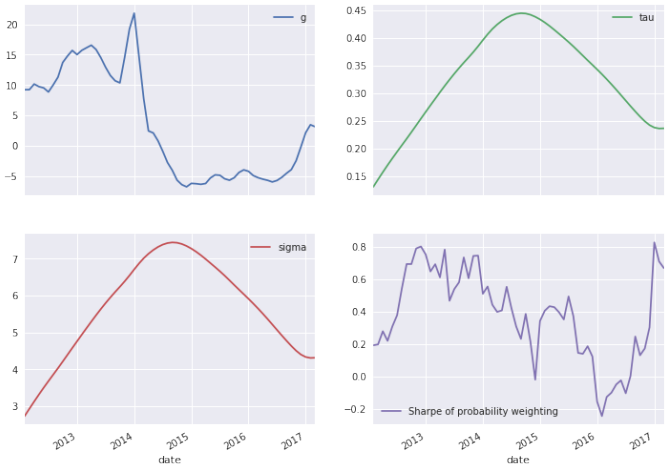

Fig. 9. Hyperparameter update

IV. EXPERIMENTS

A. Dataset

Daily price data P are used and obtained from Yahoo Finance and Google Finance, which consist of stocks in S&P 500 Index from January 2000 to March 2017. To robustly capture the general representation across stocks, price time series would be applied log-transformation first to make it conform more closely to the normal distribution.

Similarly, the effect of the lack of liquidity should also be taken into account. Therefore, in the training phase, only stocks with a certain degree of average trading volume would be selected into training set; then we are able to construct a univariate time series with identical interval n at the end of each month, denoted as $T = \left\{P_{t-n}^{tic}, P_{t-n+1}^{tic}, ..., P_t^{tic}\right\}$, where P_i^{tic} is the daily price of

stock *tic* at time stamp *i* and there are total *n* timestamps for each time series.

The proposed method is specifically designed for time series classification (TSC) problem, which is to build a classifier to predict a class label $y \in C$ given an input time series T. A time series is classified into three categories: uptrend, consolidation and downtrend. The class label y is calculated as follows:

$$y \in \begin{cases} u, & \text{if } threshold \leq \beta \\ c, & \text{if } abs(\beta) < threshold \\ d, & \text{if } \beta \leq -threshold \end{cases} \quad (5)$$

where β is the linear-regression slope of time series T and *threshold* is a constant. Setting the ideal *threshold* which makes these categories balanced is crucial for better model convergence and avoid overfitting.

B. Experimental Setting

In phase II, the main difference of MCNN in our research is that there is no down-sampling steps in the transformation stage, which means only multi-frequency branch (various degrees of smoothness) is used in time series augmentation. The main reason is that the down-sampling steps would lead to the inconsistence of data length, which results in worse convergence even with length-padding adjustment. The timestamps for each time series are set to be 60, which is about 3-month daily prices. The parameters of EWMA smoothness used in multi-frequency branch in MCNN are set to $\{2, 5, 8, ..., 29\}$.

C. Baseline

To evaluate proposed method, two benchmark are selected:

- SPY
- An equally-weighting portfolio of stocks with rank 1 in phase I (without probability weighting)

In addition to cummulative return, sharpe ratio with sliding window is also introduced for comprehensive evaluation.

V. RESULT

A. Phase I : fundamental analysis with ML

To measure variable importance for each fundamental indicators, random forest method is used in the phase I, which is a tree-based ensemble method and thus it is possible to measure the improvement of individual trees in the splitting criterion produced by each variable.

Variable importance plays an important role in discovering the relationship between fundamental information and the performance of corresponding stock. The result of importance analysis is shown in Fig 10 and the indicators with four largest importance are PE, RETA, MVQTL and STA.

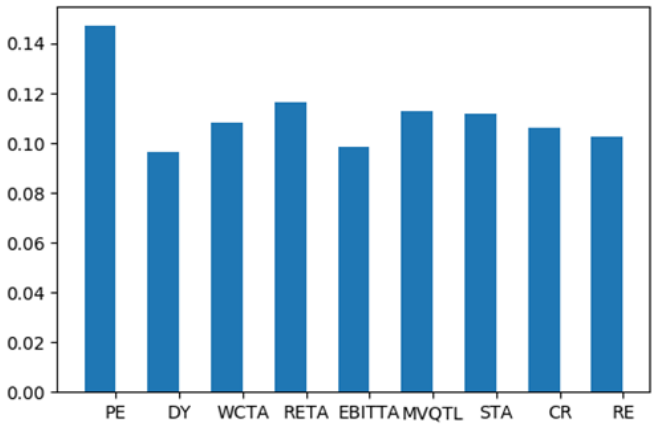

Fig. 10. Indicators importance

The accuracy of random forest classifier is 95% on average in the phase I. It's intuitive to compare each rank and SPY benchmark in the view of performance shown in Fig 11. Overall, the relationship of ranks is the same as expected; hence, phase I could be viewed as "a prefilter using fundamental information" of S&P 500 index components.

Fig. 11. phase I selection stock trend

B. Phase II : PORTFOLIO ALLOCATION with MCNN

Given a ranked portfolio in phase I, the difference between equally weighting and probability weighting is apparent as Fig 12 shown. In Fig 12, the x-axis represents S&P 500 index components in a specific rank; yellow bar with uniform distribution is equally weighting and the red one is probability weighting.

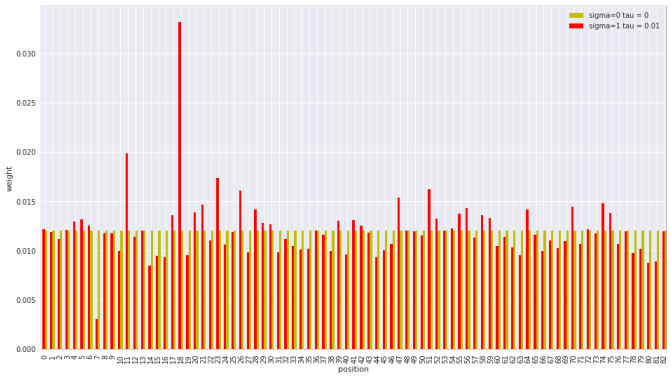

Fig. 12. Equally Weighting v.s. Probability Weighting

We also make a comparison of performance between two weighting methods. The cumulative return from January 2011 to March 2017 compared to SPY benchmark is shown in Fig 13, and the sharpe ratio with 2-year moving window is shown in Fig 14. The result of both equally weighting (phase I only) and probability weighting (phase I + phase II) methods are nearly consistently better than the SPY benchmark and the latter has a significant outperformance.

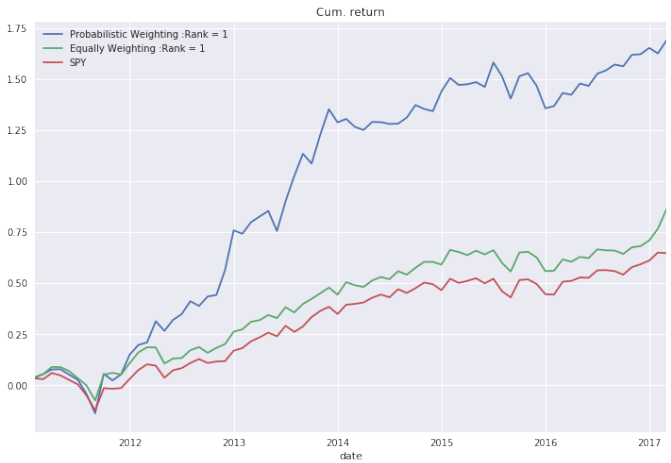

Fig. 13. cumulative return

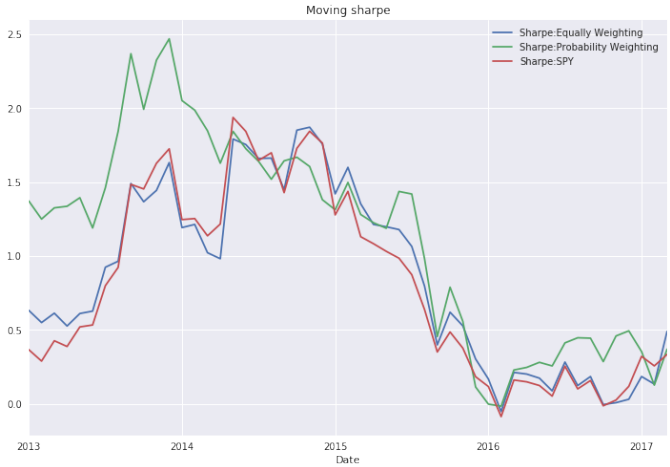

Fig. 14. moving sharpe

C. Sensitivity to hyperparameters

The proposed method includes two hyperparameters. The hyperparameter σ not only introduces nonlinearity adjustment to *Score* in probability weighting, but also controls the degree of weight concentration toward higher *Score*. The results of our model in the view of cumulative return and sharpe ratio for different constant values of σ are shown in Fig 15 and Fig 16 separately (keep τ fixed). The hyperparameter τ reflects the our confidence toward MCNN outcome. Larger τ implies stronger confidence and would make significant adjustment on weight. In the same way as σ, the results are separately shown in Fig 17 and Fig 18 (keep σ fixed).

Fig. 15. σ effect on cumulative return

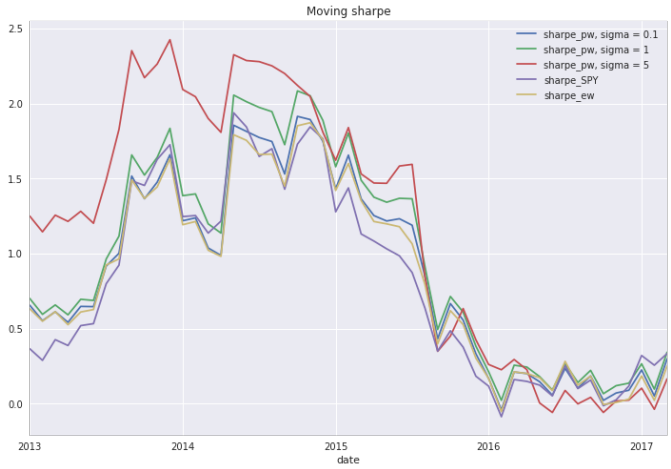

Fig. 16. σ effect on sharpe ratio

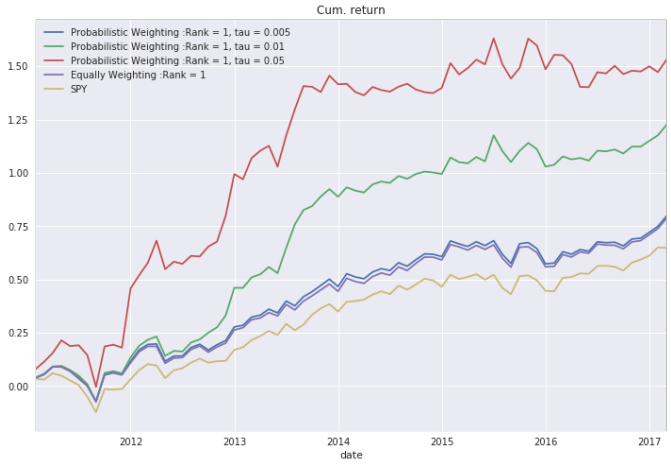

Fig. 17. τ effect on cumulative return

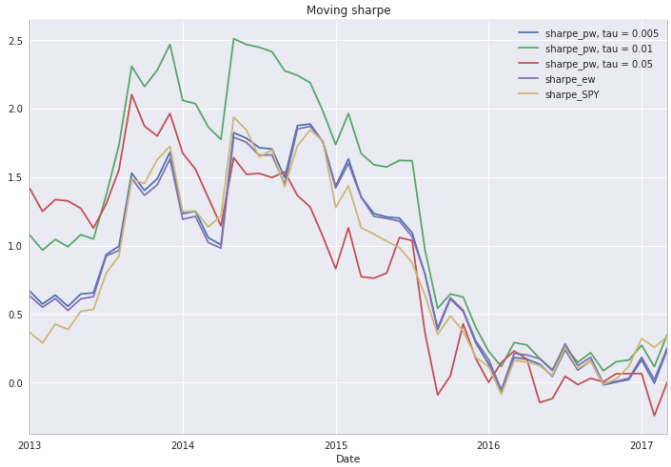

Fig. 18. τ effect on sharpe ratio

VI. CONCLUSIONS

In this paper, we have introduced a stock-picking and portfolio allocation framework to incorporate stock ranking with long-term fundamental information and short-term time series classification with MCNN. Pre-filter using random forest in phase I do capture the link between fundamental information and stock performance, and the probability weighting with MCNN in phase II gives superior results that significantly much better than the two benchmarks. There are several promising directions for future work. First, how to properly tune hyperparameters σ and τ according to MCNN in our current approach is an interesting directions. Moreover, for financial time series data, how to adapt multi-scale branch in MCNN is also the important part of our future work.

References

[1] Wang, W., Chen, C., Wang, W., Rai, P., & Carin, L. (2016). Earliness-Aware Deep Convolutional Networks for Early Time Series Classification. arXiv preprint arXiv:1611.04578.

[2] Cui, Z., Chen, W., & Chen, Y. (2016). Multi-scale convolutional neural networks for time series classification. arXiv preprint arXiv:1603.06995.

[3] Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, February). On the importance of initialization and momentum in deep learning. In International conference on machine learning (pp. 1139-1147).

[4] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[5] Zheng, Y., Liu, Q., Chen, E., Ge, Y., & Zhao, J. L. (2014, June). Time series classification using multi-channels deep convolutional neural networks. In International Conference on Web-Age Information Management (pp. 298-310). Springer, Cham.

[6] Milosevic, N. (2016). Equity Forecast: Predicting Long Term Stock Price Movement using Machine Learning. Journal Of Economics Library, 3(2), 288-294. doi:http://dx.doi.org/10.1453/jel.v3i2.750