Rebuttal: Stock recommendation using multi-task optimization with momentum and rank-aware objective function

Anonymous Author(s)

CCS CONCEPTS

• **Computing methodologies** → *Multi-task learning*; • **Information systems** → *Learning to rank*; • **Applied computing** → *Economics.*

ACM Reference Format:
Anonymous Author(s). 2018. Rebuttal: Stock recommendation using multi-task optimization with momentum and rank-aware objective function. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

Thank you for all reviewers and your comments, your kind suggestion will made this work more comprehensive. We first address these common questions, then reply every reviewer's concern one by one. Thank you again for your patience!

1 COMMON QUESTION REBUTTAL

Most of reviewers mentioned more backbones or more datasets, hyperparameter sensitivity and experiments with pair-wise loss function or rise-fall prediction. These common questions will be analyzed in this section and the added experiments results are in Table 1(New backbones and dataset), Table 2(Hyperparameter Sensitivity) and Table 3(Rise-fall and pair-wise function).

1.1 Experiments on new backbones and datasets

We added more experiments using different backbones and datasets to further prove CQB's generalizability and effectiveness. We add backbone GRU [8] and RSR [10](a classic stock forecasting model) and a smaller CSI100 stock dataset, which is also used in the HIST paper [23]. For the CSI100 stock dataset, we run the Top 20 strategies since the scale of the stock decreases from 300 to 100. Due to the time limit, we select four commonly used SOTA MTL methods as backbones: EW, an equal weighting method; DB-MTL [14](second best delta m in current experiments); CAGrad [15](best MDD currently); NashMTL [18](second best RankIC and second best profit performance). The experimental results on task performance and profit performance on new baselines and datasets are shown in Table 1.

Generally, the HIST backbone achieves the best overall performance on the task metrics on CSI100 and 300 datasets, consistent with Qlib's official benchmark. One thing worth attention is that both RSR and GRU can't handle momentum classification problems in the single training setting; they achieve low accuracy on the task(around 40% accuracy in STL). Because of the lower complexity of model structures and a limited number of parameters, GRU and RSR have less predictive power than HIST.

However, it's challenging to determine which backbone has the best profit performance. This observation is consistent with the gap we addressed; a model with higher performance on task evaluation metrics does not necessarily promise higher profits.

All methods and backbones have a worse performance on CSI100 compared with their performance on CSI300 since CSI100 stocks are more volatile and less predictable. Their turnover rates and trading volumes are higher than other stocks because they have the highest market capitalization in the Chinese stock market.

Under backbones with different complexity and stock datasets of various scales, CQB demonstrates consistent performance. Specifically, CQB could achieve four bests and two second-bests on Δ_m, which means CQB has the overall best performance on all tasks under all settings. With the balancing mechanism, CQB could balance performance between two tasks.

Regarding the profit evaluation, all MTL methods fail while employing the RSR backbone; a single training RSR could achieve the highest returns on both CSI100 and CSI300 stocks. CQB achieved two best and two second-best profits for backbone HIST and GRU in all four experiments. The consistent profit performance could be attributed to the balancing between two tasks. The model could capture rank-related information from the classification task through multi-target optimization without sacrificing too much regression task performance. These results under different backbones and datasets prove CQB's effectiveness and generalizability.

1.2 Hyperparameter sensitivity on momentum line

We discuss how to choose hyperparameters l and s in momentum line construction. First, we discuss and select a rough range empirically and intuitively. Then, we provide evidence about how to choose l and s from a statistical perspective. Finally, we conduct a series of hyperparameter experiments under different settings of l and s and report the task and profit performance.

Parameter l determines the distance between two values to construct a momentum point. Since the momentum task is used to assist daily-based return ratio regression, the momentum point is expected to contain short-term trend information. From this perspective, the rough range of l is set to [4, 12] first, including at least one week and three weeks at most. A larger l captures long-term trend information; a smaller l could be close to a one-day return ratio and lost short-term information. Parameter s decides the length of the momentum line. Since the momentum line is constructed by two future data points (price at day T+1 and day T+2), the influence of future data will be diluted when s becomes larger since more

Backbone	Method	CSI100							CSI300						
		Task Evaluation			Profit Evaluation				Task Evaluation			Profit Evaluation			
		RankIC$\uparrow$	ACC$\uparrow$	$\Delta m\% \downarrow$	AR%$\uparrow$	MDD%$\uparrow$	SR$\uparrow$	CR	RankIC$\uparrow$	ACC$\uparrow$	$\Delta m\% \downarrow$	AR%$\uparrow$	MDD%$\uparrow$	SR$\uparrow$	AR$\uparrow$
GRU	STL	0.0444	42.21%		**9.8**	**-5.7**	**1.13**	**1.73**	**0.0610**	46.16%		6.2	-4.3	0.88	1.49
	EW	0.0427	74.22%	-36.00	4.4	-8.9	0.53	0.49	0.0541	**75.73%**	-26.37	-2.5	-5.5	-0.26	-0.46
	DB-MTL	0.0415	72.59%	-32.72	-6.5	-10.6	-0.79	-0.61	0.0598	75.70%	**-31.01**	2.7	-6.4	0.36	0.42
	CAGrad	0.0444	72.50%	-35.88	1.2	-8.2	0.18	0.15	0.0587	75.25%	-29.63	3.4	-6.1	0.46	0.57
	NashMTL	**0.0467**	73.88%	**-40.10**	1.7	-6.5	0.23	0.26	0.0585	75.71%	-29.96	7.5	-3.7	0.99	2.00
	CQB	0.0434	**74.80%**	-37.48	4.7	-5.9	0.55	0.79	0.0590	75.53%	-30.17	**8.3**	**-3.4**	**1.08**	**2.40**
RSR	STL	0.0478	37.66%		**20.4**	-6.3	**2.20**	3.24	0.0579	35.43%		**10.5**	-4.7	**1.31**	**2.34**
	EW	0.0444	73.65%	-44.23	18.4	**-4.7**	2.11	**3.90**	0.0555	**75.99%**	-55.16	6.1	-4.7	0.77	1.28
	DB-MTL	0.0442	72.44%	-42.41	0.4	-7.4	0.09	0.06	0.0586	74.63%	-55.92	6.8	-4.6	0.82	1.46
	CAGrad	0.0461	71.95%	-43.75	-1.4	-8.9	-0.13	-0.16	0.0567	75.51%	-55.53	5.5	-4.5	0.80	1.21
	NashMTL	0.0481	72.82%	-46.99	1.2	-6.1	0.18	0.20	**0.0589**	74.84%	-56.48	6.1	-6.3	0.74	0.97
	CQB	**0.0494**	**73.89%**	**-49.78**	-2.7	-12.0	-0.29	-0.22	0.0584	75.56%	**-57.06**	4.1	**-4.0**	0.59	1.02
HIST	STL	**0.0537**	**74.37%**		-3.3	-11.6	-0.34	-0.28	**0.0657**	75.66%		1.6	-5.6	0.25	0.30
	EW	0.0482	72.51%	6.37	-13.1	-16.0	-1.50	-0.82	0.0581	73.57%	7.17	6.1	-4.7	0.79	1.30
	DB-MTL	0.0531	73.17%	1.37	3.1	-6.8	0.39	0.45	0.0623	74.83%	3.14	5.7	-4.8	0.70	1.18
	CAGrad	0.0516	72.84%	2.98	**9.9**	-6.1	**1.19**	**1.61**	0.0596	73.68%	5.95	7.4	**-3.6**	0.94	2.03
	NashMTL	0.0513	72.89%	3.23	-2.7	-5.4	-0.26	-0.51	0.0627	74.33%	3.16	10.9	-4.2	1.33	2.62
	CQB	0.0531	74.13%	**0.72**	6.0	**-5.3**	0.73	1.14	0.0619	**75.91%**	**1.97**	**14.4**	-4.6	**1.72**	**3.13**

Table 1: Experiment results on CSI100 and CSI300 stock set. The best and second best results are highlighted in bold and underline.

historical data is used in the calculation. So, we first set the upper bound of s as 10.

After deciding the initial range l and s, we set $l = 4, 8, 12$ and $s = 4, 6, 8, 10$ as 12 groups of hyper-parameter pairs. Experiments are conducted using these 12 hyper-parameter pairs, the experimental results are shown in Table 2.

A good momentum indicator should be able to distinguish different momentum groups on daily return because we need to employ this classification task to assist the daily-return regression task. We first compute the average one-day return ratio of momentum groups on the Train, Valid, and Test datasets, then calculate the standard deviation between different momentum groups. The Table 2's Statistical Evaluation part shows that when $l = 4$, the standard deviation is higher, indicating a larger difference among different momentum groups. The observation that standard deviation increases as l becomes smaller is intuitive, because in the extreme case that l=1, the momentum $M_t = Price_t \breve{\ } Price_{t-l}$ equals the daily return.

Next, we run experiments on all s and l pairs and report the results in the table below. We report RankIC and investment simulation results since we focus on how the regression task performance and profits change with s and l. The results show that when $s = 6, l = 4$, the model could have the best performance in profit evaluation and second best RankIC. Combined with the previous standard deviation result, we choose $s = 6$ and $l = 4$ as the parameters of the momentum line in the main experiments.

1.3 Ablation study on rise-fall task and pair-wise function

1.3.1 Compared with rise-fall prediction. Compared with commonly used rise-fall prediction, the momentum line indicators are more informative in representing the short-term trend of stocks. We add experiments using traditional rise-or-fall tasks [7] and compare them with momentum indicators. The experiments using the Rise-Fall classification is shown in Table 3.

From the result, we can see that Rise-Fall has lower accuracy; the accuracy is merely a bit higher than guessing. Updating parameters through this task is unreliable; it has little positive effect on the regression task.

The noisy nature of Rise-Fall prediction has been discussed in the paper [3]. Not only in our experiments does the rise-fall prediction have an unconvincing performance, but in many works, improving the accuracy of rise-fall prediction is extremely challenging, especially when the stock pools become large. In the WWW'19 paper [9], KDTCN could achieve 69.77% accuracy in the DJIA index(30 stocks). A work from WSDM'18 achieved less than 48% accuracy on 2527 Chinese stocks [11]. In [16], the model achieved 65.64% accuracy in CITIC securities(14 stocks). In a KDD'19 paper [13], the model achieved 68.95% accuracy in CSI300 stocks.

1.3.2 Compared with pair-wise rank function. We add experiments using pair-wise ranking loss, which has been commonly used in previous work as a new ablation study. To demonstrate this, we implement the rank loss in the RSR paper [10] and the results are shown in Table 3.

First, the result shows that list-wise ranking loss has better predictive and profit performance. Second, the pair-wise loss function is not suitable for our target, recommending top stocks from a large stock pool to have better profit. The pair-wise loss function computes every pair of stocks, but investors pay attention yo the relative order between No.250 stock and No.251 stock because neither will be considered in a portfolio. The computation of the pair-wise rank function brings abundant information in the loss. And with list-wise Adaptive-k ApproxNDCG, the model could focus on the top-ranked stocks and consider the rank of all top stocks simultaneously, then directly give the ranking results. Since we need model

s	l	Statistical Evaluation			Task Evaluation	Profit Evaluation			
		Train Std/e^{-3} ↑	Valid Std/e^{-3} ↑	Test Std/e^{-3} ↑	RankIC↑	AR%↑	MDD%↑	SR↑	CR↑
4	4	**9.05**	$\underline{8.78}$	**6.75**	0.0590	-5.2	-6.5	-0.58	-0.81
4	8	7.18	7.44	5.41	0.0611	$\underline{5.5}$	**-4.2**	$\underline{0.73}$	$\underline{1.30}$
4	12	6.82	7.94	5.10	0.0614	3.2	-6.1	0.44	0.53
6	4	$\underline{8.28}$	7.54	$\underline{6.22}$	$\underline{0.0619}$	**14.4**	-4.6	**1.72**	**3.13**
6	8	5.55	5.44	4.14	0.0597	2.3	-5.5	0.31	0.42
6	12	5.18	5.25	3.71	0.0617	3.3	-4.9	0.46	0.48
8	4	7.94	7.91	$\underline{6.12}$	0.0608	-3.5	-7.1	-0.38	-0.50
8	8	4.87	5.06	3.74	0.0617	3.5	-5.9	0.45	0.59
8	12	4.42	4.53	3.34	**0.0635**	2.0	$\underline{-4.5}$	0.28	0.44
10	4	7.95	**8.83**	6.10	0.0577	-0.2	-5.9	0.02	-0.03
10	8	4.76	5.29	3.93	0.0611	-3.1	-6.5	-0.31	-0.48
10	12	4.16	4.09	3.14	0.0566	-2.0	-7.1	-0.19	-0.28

Table 2: Experiments on different hyper parameters s and l

pay more attention to the top profitable stocks, a list-wise rank function is more suitable.

	Task Evaluation		Profit Evaluation			
	RankIC ↑	ACC% ↑	AR% ↑	MDD% ↑	SR ↑	CR ↑
Rise-Fall	0.0373	51.52	3.4	-5.9	0.41	0.58
Pair-wise	0.0584	75.57	3.6	-3.9	0.49	0.94
Our method	**0.0619**	**75.91**	**14.4**	**-4.6**	**1.72**	**3.13**

Table 3: Ablation study of Rise-Fall task and Pair-wise function

Algorithm 1: Revised Adaptive-k ApproxNDCG

```
   Data: Output from classification layer; True label $Y_{class}$
   Result: Adaptive-k ApproxNDCG Loss
 1 Given a trading day t:
 2 $logits = Softmax(output)$
 3 $Countlist^t = [count^t_{level4}, count^t_{level3},..., count^t_{level0}]$
 4 $k$=0
 5 $threshold^t = 20\% sum(Countlist^t)$
 6 for $i$ in $Countlist^t$ do
 7     k=k+i
 8     if $k > threshold^t$ then
 9         Break
10     end
11 end
12 for stock $s$ do
13     $w^s_{class} = y^s_{class} * y^s_{class}$
14     $w^s_{pred} = \sum_{i=0}^{4} logit^s[i] * i^2$
15 end
16 NDCG@k = $ApproxNDCG(\pi_{w_{pred}}, w_{class}, k)$
17 $Loss = 0.5 * CrossEntropy(output, Y_{class}) + 0.5 * e^{-NDCG@k}$
```

Algorithm 2: Revised Converge-based Quad-Balancing

```
   Require: learning rate $\eta$, forgetting rate $\beta$, numbers of
            epochs E, numbers of iterations in one epoch M,
            numbers of task T;
 1 Randomly initialize $\theta_0$, $\{\psi_{t,0}\}_{t=1}^{T}$;
 2 Initialize $V_0 = 1$ for all tasks;
 3 for $e = 0, .., E-1$ do
 4     $decay_e = decay * sigmoid(-average(V_{e-1}))$
 5     for $m = 1, .., M$ do
 6         for $t = 1, ...T$ do
 7             $g_{t,m} = \nabla_{\theta_m} log(l_t + 10^{-8})$
 8             $\beta_{t,e} = \beta^{Sigmoid(V_{t,e})}$
 9             $\hat{g}_{t,m} = \beta_{t,e}\hat{g}_{t,m-1} + (1-\beta_{t,e})g_{t,m}$
10         end
11         $\alpha_m = max_{1 \le t \le T} \| \hat{g}_{t,m} \|_2$
12         $\widetilde{g}_m = \alpha_m \sum_{t=1}^{T} \frac{\hat{g}_{t,m}}{\|\hat{g}_{t,m}\|_2 + 10^{-8}}$
13         $\theta_{m+1} =_m -\eta \widetilde{g}_m$
14         for $t = 1, ...T$ do
15             $\psi_{t,m+1} = \psi_{t,m} - \eta \nabla_{\psi_{t,m}} log(l_t + 10^{-8})$
16         end
17     end
18     Test on train and validation dataset;
19     for $t = 1, ...T$ do
20         Record train and validation losses;
21         Compute $V_{t,e}$ using historical train and validation
             losses;
22     end
23 end
```

2 REPLY TO REVIEWER PXNW

2.1 Q1

We add a figure of the whole framework as Figure 1, and we will add the description to the beginning of section 3 in the paper.

The input of the framework is time series data. Initially, the data is leveraged to construct a momentum indicator, serving as the ground truth of the classification task. Subsequently, a deep learning backbone(GRU [8], RSR [10] or HIST [23]) is employed to encode the data into representation vectors. During the prediction stage, two fully connected layers are implemented to generate the predictions for classification and regression tasks. Utilizing the gradients and loss on two tasks, the CQB algorithm is implemented to balance the weight of two task gradients. The deep learn backbone and two prediction layers are updated in accordance with the balanced gradients.

2.2 Q2

We add hyperparameter experiments on momentum line construction and the results are shown in Table 2. 12 groups of hyperparameters are tested and the combination of $s = 6, l = 4$ could achieve best profits performance and second best RankIC. Given the results of statistical analysis, task evaluation and profits evaluation, we choose $s = 6, l = 4$ as the hyperparameter in our framework. The details please refer to Section 1.2.

2.3 Q3

New experiments are conducted on two new backbones GRU [8] and RSR [10], and one new dataset CSI100, the results are shown in Table 1. CQB could achieve overall best performance in these 6 settings.

3 REPLY TO REVIEWER VYWH

.

3.1 C1

3.1.1 C1.1. Thank you and we will add more explanations about the table. Here is the added description:

- We use Table 1 to show that lower training loss can't necessarily lead to better profit. Regression model R_1 is considered better since it achieves lower MSE than model R_2. However, R_2 has higher profits under top-2 stock investment.

3.1.2 C1.2. We add hyperparameter experiments on momentum line construction and the results are shown in Table 2. 12 groups of hyperparameters are tested and the combination of $s = 6, l = 4$ could achieve best profits performance and second best RankIC. Given the results of statistical analysis, task evaluation and profits evaluation, we choose $s = 6, l = 4$ as the hyperparameter in our framework. The details please refer to Section 1.2.

3.1.3 C1.3. The threshold choice is based on the financial market we choose. CSI 300 usually has 300 stocks and common quantitative investment focus on the top 50 profitable ones, and 50/300 is 16.67%; that's why we choose 20% as the threshold; we expect the model to consider at least 20% of stocks ranking in Adaptive-k ApproxNDCG.

As a loss objective in this framework. an NDCG function should satisfy three requirements: 1) decreasing monotonically with NDCG; 2) differentiable; 3) having a scale that is close to cross-entropy. And $e^{(-NDCG)}$ could satisfy the three requirements. First, it decreases monotonically. Second, it is differentiable. Third, since NDCG changes from 0 to 1, $e^{(-NDCG)}$ has a range $[1/e, 1]$, which can be combined with cross-entropy without additional scaling. That's why we chose the exponential function to form an NDCG loss function.

3.1.4 C1.4. We use the CSRank Normalization method provided by Qlib to preprocess time series data. Thank you for reminding us and we will add the details about the regularization method in the experimental setup.

3.1.5 C1.5. We plot the $beta_s$ during training. The Figure 2 shows that the beta of the two tasks changed continuously while training, and a higher beta indicates that the improvement of model performance on validation dataset slows down, even become negative. Compared with the fixed beta, CQB could the decrease the weight of new gradients since it deteriorates the model's performance on validation data.

3.2 C2

Thank you for reminding us. Your kind suggestions will make this work more comprehensive. We will correct these errors and revise the proof carefully. We will use more precise language in the proof. Algorithm 1 is the revised version, we change return k to break in line 9 and change the W in line 16 to w.

Revised Assupmtion 2: As overfitting occurs, the update of model parameters from gradient descent is not improving the model's performance on the validation set.

Revised part of Proof Sketch: First, we analyze the change of β_n when overfitting happens. According to Assumption 2, the model's performance on the validation set could deteriorate when overfitting happens and $\Delta loss_{valid}$ increases. Then, because of Assumption 1, V_n decreases. The forgetting rate $\beta \in (0, 1)$, as a result, β_n will increase while overfitting.

3.3 C3

In this work, our focus is to propose a general and model-agnostic multi-task framework instead of tuning the trading strategy. To ensure fairness, we use a default Qlib strategy provided by the official Qlib document , which is also used in the KDD23 paper [25] and KDD19 paper [13]. The investment simulation is notorious for its complexity due to factors such as risk degree, limit threshold, transaction fees, etc. A higher RankIC or ACC can't promise a better profit, and this is also our motivation to introduce ranking information and a multi-task framework for better stock recommendation. An example to demonstrate the performance gap is shown in Table 4. R_1 and R_2 have the same RankIC, C_1 and C_2 have the same ACC, but the profits of these models are not the same. From the main experimental result, we could also observe that the model achieve the highest RankIC is not always the most profitable one.

3.4 C4

We add a figure of the whole framework as Figure 1, and we will add the description to the beginning of section 3 in the paper.

The input of the framework is time series data. Initially, the data is leveraged to construct a momentum indicator, serving as the ground truth of the classification task. Subsequently, a deep learning

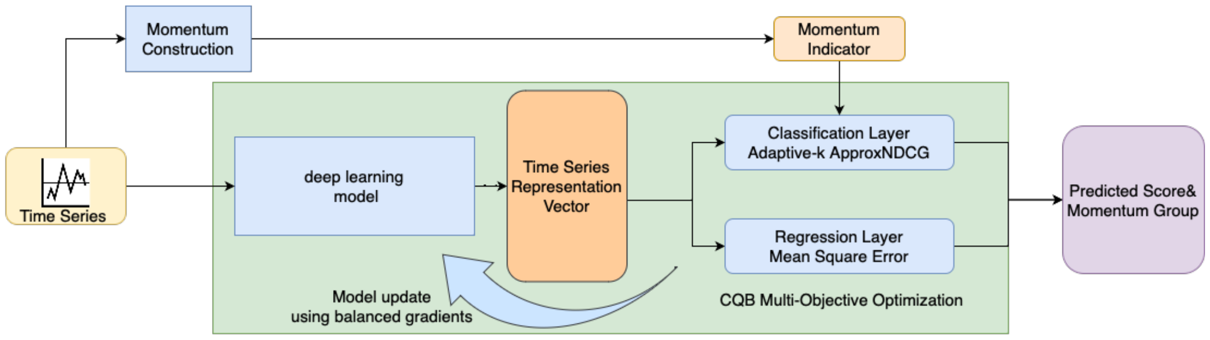
Momentum Construction
Momentum Indicator
Time Series
deep learning model
Time Series Representation Vector
Classification Layer Adaptive-k ApproxNDCG
Regression Layer Mean Square Error
Predicted Score& Momentum Group
Model update using balanced gradients
CQB Multi-Objective Optimization

Figure 1: Demonstration of the framework

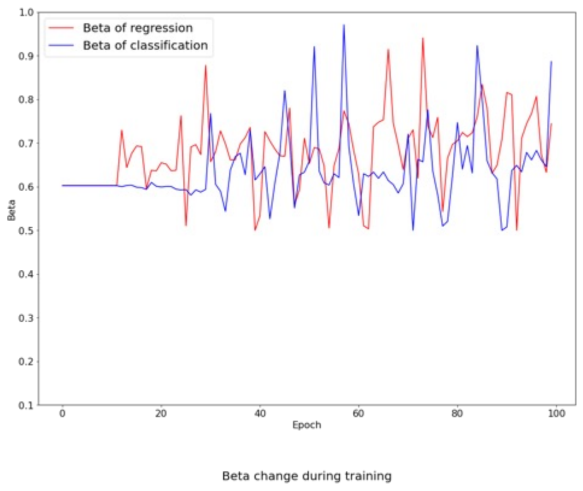

Figure 2: The change of $beta_s$ during training

Stocks	S_1	S_2	S_3	S_4	S_5	Metrics	Profits
Returns	-0.5	-0.4	-0.1	+0.3	+0.5		Top-2
Regression model						RankIC(↑)	
R_1	-0.3	-0.4	**+0.2**	-0.2	**+0.4**	0.80	0.20
R_2	-0.2	-0.3	0.1	**+0.4**	**+0.3**	0.80	0.40
Classification model						ACC(↑)	
C_1	**1**	0	0	**1**	0	0.40	-0.10
C_2	0	**1**	**1**	0	0	0.40	-0.25

Table 4: Example showing that conventional regression training can't promise higher profits.

backbone(GRU [8], RSR [10] or HIST [23]) is employed to encode the data into representation vectors. During the prediction stage, two fully connected layers are implemented to generate the predictions for classification and regression tasks. Utilizing the gradients and loss on two tasks, the CQB algorithm is implemented to balance the weight of two task gradients. The deep learn backbone and two prediction layers are updated in accordance with the balanced gradients.

3.5 C5

Thank you for reminding us. We will fix these typos in the revision.

3.6 Q1

Firstly, the ranking function is employed to serve as a plug-in function, and it should be simple, efficient, and ready to use without training. SetRank [19] and Rankformer [6] need training, which is complicated and time-consuming compared with NDCG.

Secondly, a list-wise rank function is more suitable for emphasizing the importance of top stocks. A pair-wise rank function needs to compute the relative order between every pair. However, investors may pay little attention to the relative order between No.250 and No.251 stock because neither will be considered in a portfolio. The computation of the pair-wise rank function brings abundant information in the loss. Pair-wise rank functions like LambdaRank [4] and LambdaMART [5] are not the optimal choice in our experiment. To demonstrate this, we implement the pair-wise rank loss in the RSR paper [10], and the results are shown in Table 3.

From the result, we can see that model training using list-wise, i.e., the NDCG ranking loss has better predictive and profit performance. We will add corresponding parts in related work to expand the research scope as well.

Thirdly, [17] builds an attention-based multi-task learning model and uses a linear combination of multi-task loss (equation 24 in [17]). The weight of each task is selected manually. Such manually selected parameters can't balance the gradients when gradients conflict or imbalances occur [24]. Their work focuses more on putting forward a new multi-task learning model and building attention between different tasks, not the optimization between multi-objectives. Our work aims to design a model-agnostic multi-task optimization method to balance the weights of gradients and loss during training. Therefore, we choose SOTA multi-objective optimization methods as baselines.

4 REPLY TO REVIEWER W8QA

4.1 C1

Compared with commonly used rise-fall prediction, the momentum line indicators are more informative in representing the short-term trend of stocks. We add experiments using traditional rise-or-fall tasks and compare them with momentum indicators. The experiments using the Rise-Fall classification are shown in Table 3. From the result, we can see that Rise-Fall has lower accuracy; the accuracy is merely a bit higher than guessing. Updating parameters through this task is unreliable; it has little positive effect on the regression task.

The noisy nature of Rise-Fall prediction has been discussed in the paper [3]. Not only in our experiments does the rise-fall prediction have an unconvincing performance, but in many works, improving the accuracy of rise-fall prediction is extremely challenging, especially when the stock pools become large. In the WWW'19 paper [9], KDTCN could achieve 69.77% accuracy in the DJIA index(30 stocks). A work from WSDM'18 achieved less than 48% accuracy on 2527 Chinese stocks [11]. In [16], the model achieved 65.64% accuracy in CITIC securities(14 stocks). In a KDD'19 paper [13], the model achieved 68.95% accuracy in CSI300 stocks.

For the generalizability of CQB, we compared it with 3 recent SOTA MTL algorithms [14, 15, 18], and CQB could have the best multi-task performance and investment performance on the stock task. We run experiments on new backbones and datasets to further prove CQB's generalizability and effectiveness. The details please reger to Section 1.1. CQB could achieve overall best performance in these 6 settings.

4.2 C2

We revised the related work part in the paper, and here is the new content:

Added content start from line 214: However, the pair-wise rank function computation brings abundant information in the loss. Investors pay little attention to the relative order between No.250 and No.251 stocks in portfolio building, but the pair-wise rank function computes the relative order between all stocks. To emphasize the weights of profitable stocks, our work uses a differentiable list-wise rank function, which could ignore the weight of less profitable stocks while ranking.

Added content start from line 244: Although SOTA MTL methods perform well on CV datasets, they could fail on stock recommendation since time series data suffers from distribution shifts and overfitting issues. To fill the gap and mitigate the overfitting issue in the stock multi-task framework, CQB detects performance deterioration and changes the forgetting rate and weights of different tasks.

Added content start from line 255: The widely used prediction is unreliable in MTL since the accuracy is around 60%. In many works, improving the accuracy of rise-fall prediction is extremely challenging, especially when the stock pools become large. In the WWW'19 papee [9], KDTCN could achieve 69.77% accuracy in the DJIA index(30 stocks). A work [11] from WSDM'18 achieved less than 48% accuracy on 2527 Chinese stocks. In [16], the model achieved 65.64% accuracy in CITIC securities(14 stocks). In a KDD'19 paper [13], Li, Song and Tao. achieved 68.95% accuracy in CSI300 stocks. To introduce a more reliable indicator in the multi-task setting, we construct a momentum line to represent the short-term trend of stock price. The concept of momentum has been proposed for decades and has proven effective in investment [1, 2, 12].

4.3 C3

To the best of our knowledge, this is the first work to combine differentiable NDCG loss functions in a multi-objective optimization framework in a stock recommendation task. Our framework achieved SOTA performance in task evaluations as well as profit evaluations. The challenges lie in introducing a less noisy and stable task as the second task, a list-wise rank function that pays more attention to high-rank stocks. And an MTL framework for handling serious overfitting problems in multi-objective optimization. We summarize our innovations as follows:

- We propose a new indicator called the momentum line to represent the short-term trend. Stock recommendation could benefit from training with this sub-task in a multi-objective fashion.
- We propose a novel list-wise rank objective function, which could be used without training. Using Adaptive-k Approx-NDCG, the MTL framework could have a better awareness of top stocks and better profit performance.
- We propose a new model-agnostic MTL method, CQB. By introducing converge-based balancing, our method outperforms SOTA MTL methods on stock recommendation tasks.

5 REPLY TO REVIEWER VCCK

5.1 W1

Please refer to Section 1.3.1 and Table 3. Rise-Fall prediction has lower accuracy. Updating parameters through this task is unreliable; it has little positive effect on the regression task.

5.2 W2

We add experiments using pair-wise ranking loss, which has been commonly used in previous work as a new ablation study. To demonstrate this, we implement the rank loss in the RSR paper [10] and the results are shown in Table 3. First, the result shows that list-wise ranking loss has better predictive and profit performance. Second, the pair-wise loss function is not suitable for our target, recommending top stocks from a large stock pool to have better profit.

5.3 W3

We added two new backbones and one new dataset CSI100 to further prove CQB's generalizability and effectiveness. The results are shown in Table 1 and detailed analysis please refer to Section 1.1. CQB could achieve overall best performance in these 6 settings.

For the No Square Weight in ablation study, the No Square Weight group gives weight of 0,1,2,3,4 to different momentum groups rather than 0,1,4,9,16. Compared with square weight, the weight difference between groups is less significant. To show the impact of using no square weight more directly, we compute the

	Avg@10	Avg@20	Avg@30	Avg@50
Cross-entropy	.0471	**.0601**	<u>.0587</u>	<u>.0577</u>
No adaptive-k	.0539	.0563	.0555	.0573
No Square Weight	<u>.0602</u>	.0577	.0548	.0569
Adaptive-k ApproxNDCG	**.0631**	<u>.0598</u>	**.0591**	**.0582**

Table 5: Average score of everyday Top N stocks of different classification objective function

average score of everyday top N predictions from different experiment settings in Table 5; the score is normalized from the one-day return ratio. No Square Weight setting has the lowest top 50 average scores, indicating that No Square Weight will return less profitable stocks as top 50 recommendations. The investment simulation is complicated; risk degree, limit threshold, and transaction fee are all considered in investment trading. The average score can't represent the profit, but we could intuitively conclude its failure since it returns average less profitable stocks as top ones.

5.4 W4

5.4.1 W4.1. Thank you for reminding us. We train the model on a single A100-SXM4-40GB GPU. The size of the representation vector is 128. We train every model 100 epochs with a learning rate of 2e-4. Since we have a ranking objective function, the batch size is the number of CSI300 stocks, and a batch contains CSI300 stock data on the same trading day. We will add these setting details in the paper.

5.4.2 W4.2. The fixed beta is 0.5. We chose this beta=0.5 because, in DB-MTL, we use beta=0.5, which is CQB's initial beta. We plot the change of beta while training in the CQB framework as Figure 2. The beta continuously changes with V_n changing, and a higher beta indicates that the improvement of model performance on validation dataset slows down, even become negative. Compared with the fixed beta, CQB could decrease the weight of new gradients since it deteriorates the model's performance on validation data.

About hpyerparameter sensitivity, experiments on s and l in momentum line are conducted. The results are shown in Table 2 and detailed analysis please refer to Section 1.2. Given the results of statistical analysis, task evaluation and profits evaluation, we choose $s = 6, l = 4$ as the hyperparameter in our framework.

5.5 W5

An ideal evaluation of stock forecasting is profit-related metrics [3] like accumulated return during investment. But the investment simulation is notorious for its complexity due to factors such as risk degree, limit threshold, transaction fees, etc. Given the complexity of trading, using profit-related metrics directly as an objective during training is unattainable.

For regression tasks, the most common approach is to use error-based metrics as objective functions and use RankIC to evaluate its performance. For the classification tasks, cross-entropy is used as an objective function, and confusion matrix-based metrics are used as an evaluation. However, a higher RankIC or ACC does not necessarily mean a better profit during trading. An example to demonstrate the performance gap is shown in Table 4. In this table, R1 and R2 have the same RankIC, and C1 and C2 have the same ACC but different profits. A higher RankIC or ACC can't promise a better profit, and this is also our motivation to introduce ranking information and a multi-task framework for better stock recommendation.

This gap exists through observation and related work [10, 20–22] because (1) the model ignores the direction while training the regression model. (2) The objective function ignores the rank between stocks. (3) The objective function didn't pay more attention to high-tier stocks, which received more attention from real-world investors. Our method aimed to mitigate the gap by introducing a direction-related task(momentum), a list-wise rank function(Adaptive-k ApproxNDCG), and a Multi-objective optimization method to mitigate overfitting in training(CQB). Through our experimental results, our methods could achieve better task performances and profits compared with other methods in table 1.

5.6 W6&W7

Thank you for reminding. We will revise these typos and update figures according to your advice.

6 REPLY TO REVIEWER HERV

6.1 C1

Thank you for your detailed check; your efforts have made this work better. We will modify these parts carefully, revise all typos, and polish notations and algorithms.

6.1.1 C1.1. The sentence starts with "Two tasks": "Two tasks have different coverage performance in the same training epoch."

6.1.2 C1.2. The $\hat{g}$ means the modified gradient after the EMA operation; $Loss_{n-1}$ is the loss on the validation dataset in the n-1 training epoch. $Beta_n^{m+1}$ is the m+1 square of $beta_n$, since for $\hat{g}_{n+m}$ and $\hat{g}_{n-1}$, there are m+1 iterations between them and the weight of $\hat{g}_{n-1}$ in g_{n+m} should be m+1 square of $beta_n$; The k is a typo; it should be "n" as well; We add equation 8 into algorithm 2 and modify algorithm 1. And Algorithm 1 and 2 are the revised versions.

6.2 C2

6.2.1 C2.1. We add hyperparameter experiments on momentum line construction and the results are shown in Table 2. 12 groups of hyperparameters are tested and the combination of $s = 6, l = 4$ could achieve best profits performance and second best RankIC. Given the results of statistical analysis, task evaluation and profits evaluation, we choose $s = 6, l = 4$ as the hyperparameter in our framework. The details please refer to Section 1.2.

New experiments are conducted on two new backbones GRU [8] and RSR [10], and one new dataset CSI100, the results are shown in Table 1. CQB could achieve overall best performance in these 6 settings.

6.2.2 C2.2. New experiments are conducted on two new backbones GRU [8] and RSR [10], and one new dataset CSI100, the results are shown in Table 1. CQB could achieve overall best performance in these 6 settings.

6.3 Q1

The early-stopping can detect overfitting, but won't affect the weights of different tasks. CQB and early stopping could be employed together to achieve better results. Our method is trying to change the weight of different objectives, and we expect the model to seek a better convergence of the model guided by the "less" overfitting objective. The effectiveness of CQB could be overserved in Figure 3 from original paper. Compared with DB-MTL, CQB mitigate the rise trend of losses of validation and test dataset after epoch 15.

6.4 Q2

The experiments shown in Figure 3 from original paper have already employed early-stopping. The $beta_s$ in CQB start increasing while overfitting is detected, so at the beginning of the training, the loss of DB-MTL and CQB have similar trends. When overfitting happens(after epoch 15), DB-MTL and CQB have different loss trends, which is the signal that CQB forgetting rate balancing and L2 regularization balancing works.

7 REPLY TO REVIEWER 3T6W

7.1 C1&Q1

We add hyperparameter experiments on momentum line construction and the results are shown in Table 2. 12 groups of hyperparameters are tested and the combination of $s = 6, l = 4$ could achieve best profits performance and second best RankIC. Given the results of statistical analysis, task evaluation and profits evaluation, we choose $s = 6, l = 4$ as the hyperparameter in our framework. The details please refer to Section 1.2.

7.2 C2

New experiments are conducted on two new backbones GRU [8] and RSR [10], and one new dataset CSI100, the results are shown in Table 1. CQB could achieve overall best performance in these 6 settings.

7.3 Q2

We report the change of k across different training instances(trading days) in Figure 3. The k value changes from 50 to 300, depending on the label distribution of the training instance. And most of k falls between 60 and 100. Here we display several cases below to illustrate why k changed.

In trading day 2007-09-13, there are more than 20% stocks belongs to Group 4, so the k is 86 in this trading day. And for trading day 2009-09-23 and 2013-08-07, most of stocks belongs to the same momentum group, so the k become larger(224 and 203). If we use a fixed k 50, No.50 and No.51 stocks have the same momentum group but the weight of No.51 stock will be ignored in NDCG@k function. The operation confuse the model and impact the ranking performance. But with adaptive-k mechanism, stocks belong to the same momentum groups will be guaranteed to have consistent weights. The choice of the k threshold is based on the financial market we choose. CSI 300 usually has 300 stocks and common quantitative investment focus on the top 50 profitable ones, and 50/300 is 16.67%; that's why we choose 20% as the threshold; we expect the model to consider at least 20% of stocks ranking in Adaptive-k ApproxNDCG.

Datetime	Group4	Group3	Group2	Group1	Group0
2007-09-13	86	68	12	99	10
2009-09-23	1	10	9	204	69
2013-08-07	4	193	14	4	73

Table 6: Examples of trading days. Number of stocks in each momentum line group varies in different trading days.

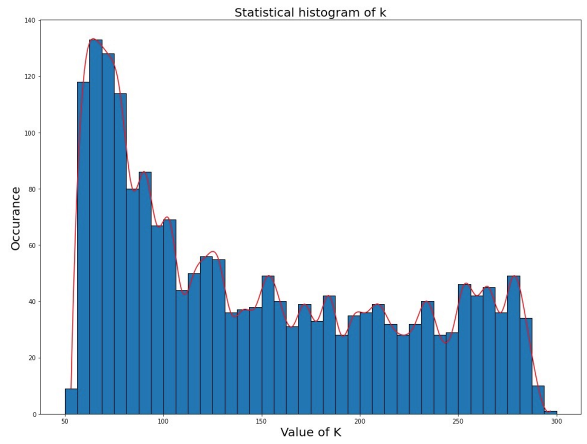

Figure 3: The distribution of adaptive-k in training instance.

REFERENCES

[1] Clifford Asness, Andrea Frazzini, Ronen Israel, and Tobias Moskowitz. 2014. Fact, fiction, and momentum investing. *The Journal of Portfolio Management* 40, 5 (2014), 75–92.
[2] Pedro Barroso and Pedro Santa-Clara. 2015. Momentum has its moments. *Journal of Financial Economics* 116, 1 (2015), 111–120.
[3] Yoshua Bengio. 1997. Using a financial training criterion rather than a prediction criterion. *International journal of neural systems* 8, 04 (1997), 433–443.
[4] Christopher Burges, Robert Ragno, and Quoc Le. 2006. Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems* 19 (2006).
[5] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.
[6] Maarten Buyl, Paul Missault, and Pierre-Antoine Sondag. 2023. RankFormer: Listwise Learning-to-Rank Using Listwide Labels. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3762–3773.
[7] Eunsuk Chong, Chulwoo Han, and Frank C Park. 2017. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications* 83 (2017), 187–205.
[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
[9] Shumin Deng, Ningyu Zhang, Wen Zhang, Jiaoyan Chen, Jeff Z Pan, and Huajun Chen. 2019. Knowledge-driven stock trend prediction and explanation via temporal convolutional network. In *Companion proceedings of the 2019 world wide web conference*. 678–685.
[10] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. 2019. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)* 37, 2 (2019), 1–30.
[11] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. 2018. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 261–269.
[12] Narasimhan Jegadeesh and Sheridan Titman. 1993. Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of finance* 48, 1 (1993), 65–91.

[13] Chang Li, Dongjin Song, and Dacheng Tao. 2019. Multi-task recurrent neural networks and higher-order Markov random fields for stock price movement prediction: Multi-task RNN and higer-order MRFs for stock price classification. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1141–1151.

[14] Baijiong Lin, Weisen Jiang, Feiyang Ye, Yu Zhang, Pengguang Chen, Ying-Cong Chen, Shu Liu, and James T Kwok. 2023. Dual-Balancing for Multi-Task Learning. *arXiv preprint arXiv:2308.12029* (2023).

[15] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. 2021. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems* 34 (2021), 18878–18890.

[16] Jiawei Long, Zhaopeng Chen, Weibing He, Taiyu Wu, and Jiangtao Ren. 2020. An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in Chinese stock exchange market. *Applied Soft Computing* 91 (2020), 106205.

[17] Tao Ma and Ying Tan. 2022. Stock ranking with multi-task learning. *Expert Systems with Applications* 199 (2022), 116886.

[18] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. 2022. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017* (2022).

[19] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. 2020. Setrank: Learning a permutation-invariant ranking model for information retrieval. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 499–508.

[20] Suman Saha, Junbin Gao, and Richard Gerlach. 2021. Stock ranking prediction using list-wise approach and node embedding technique. *IEEE Access* 9 (2021), 88981–88996.

[21] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, Tyler Derr, and Rajiv Ratn Shah. 2021. Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 497–504.

[22] Heyuan Wang, Tengjiao Wang, Shun Li, Jiayi Zheng, Shijie Guan, and Wei Chen. 2022. Adaptive long-short pattern transformer for stock investment selection. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. 3970–3977.

[23] Wentao Xu, Weiqing Liu, Lewen Wang, Yingce Xia, Jiang Bian, Jian Yin, and Tie-Yan Liu. 2021. Hist: A graph-based framework for stock trend forecasting via mining concept-oriented shared information. *arXiv preprint arXiv:2110.13716* (2021).

[24] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems* 33 (2020), 5824–5836.

[25] Lifan Zhao, Shuming Kong, and Yanyan Shen. 2023. DoubleAdapt: A Meta-learning Approach to Incremental Learning for Stock Trend Forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3492–3503.