Temporally Correlated Task Scheduling for Sequence Learning

Xueqing Wu [1] Lewen Wang [2] Yingce Xia [2] Weiqing Liu [2] Lijun Wu [2] Shufang Xie [2] Tao Qin [2] Tie-Yan Liu [2]

Abstract

Sequence learning has attracted much research attention from the machine learning community in recent years. In many applications, a sequence learning task is usually associated with multiple temporally correlated auxiliary tasks, which are different in terms of how much input information to use or which future step to predict. For example, (i) in simultaneous machine translation, one can conduct translation under different latency (i.e., how many input words to read/wait before translation); (ii) in stock trend forecasting, one can predict the price of a stock in different future days (e.g., tomorrow, the day after tomorrow). While it is clear that those temporally correlated tasks can help each other, there is a very limited exploration on how to better leverage multiple auxiliary tasks to boost the performance of the main task. In this work, we introduce a learnable scheduler to sequence learning, which can adaptively select auxiliary tasks for training depending on the model status and the current training data. The scheduler and the model for the main task are jointly trained through bi-level optimization. Experiments show that our method significantly improves the performance of simultaneous machine translation and stock trend forecasting.

1. Introduction

Sequence learning (Sutskever et al., 2014; Bahdanau et al., 2014) is an important problem in deep learning, which covers many applications including machine translation (Wu et al., 2016; Vaswani et al., 2017), time series prediction (Qin et al., 2017a; Zhang et al., 2017a), weather forecasting (Shi et al., 2015; Kim et al., 2017), etc. In real-world applications, a sequence learning task is often associated with multiple temporally correlated tasks: (1) In simulta-

neous machine translation, we need to begin translation before reading the complete source sequence. In this application, the generation of output sentence is k word behind the source input, which is called *wait-k* strategy/task (Ma et al., 2019). *Wait-k* tasks with different k values are temporally correlated. (2) Stock forecasting aims to predict the trend of stocks at some future day (Dai & Zhang, 2013). Tasks of predicting stock prices at different future days constitute a set of temporally correlated tasks. Different from general multi-task learning, the input and output data of these temporally correlated tasks are highly overlapped, and only differ in how much future input to use when processing the current input (e.g., how many future source words to read to translate the current word in simultaneous translation) or which future data to predict (e.g., the price movement of which future day to predict in stock trend forecasting).

Clearly, a given sequence learning task (denoted as the main or target task) can be boosted by its temporally correlated tasks. Taking English-to-Vietnamese simultaneous translation as an example, assume the main task is *wait*-3, and the temporally correlated tasks are *wait*-$1, 2, 3, \cdots, 13$. By training a model on all those correlated tasks and evaluating the model on *wait*-3, we observe that the BLEU score is improved from 27.17 to 28.54. (BLEU (Papineni et al., 2002) is a widely used accuracy metric in machine translation, the larger the better.) A similar phenomenon is observed in stock price forecasting. Improvements across various applications motivate us to explore how to leverage those temporally correlated tasks more effectively to improve performance on the main task.

Given that there are usually multiple temporally correlated tasks, the key challenge lies in which tasks to use and when to use them in the training process. In this work, we introduce a learnable task scheduler for sequence learning, which adaptively selects temporally correlated tasks during the training process. The scheduler accesses the model status and the current training data (e.g., in current minibatch), and selects the best auxiliary task to help the training of the main task. The scheduler and the model for the main task are jointly trained through bi-level optimization: the scheduler is trained to maximize the validation performance of the model, and the model is trained to minimize the training loss guided by the scheduler. Experimental results on simultaneous machine translation and stock trend forecasting show

that such a task scheduler brings significant improvements.

Our main contributions are three-fold:

(1) To the best of our knowledge, we are the first to formulate and study the problem of leveraging temporally correlated tasks for sequence learning.

(2) We propose a new learning algorithm with a scheduler to adaptively select the temporally correlated tasks to boost the main task. The scheduler and the model of the main task are jointly trained without additional supervision signals.

(3) We empirically verify the effectiveness of our method. Experiments on four simultaneous translation tasks show that our method improves *wait-k* baseline by 1 to 3 BLEU scores. Experiments on a stock price forecasting dataset show that our method can improve the baselines in terms of Spearman's rank correlation coefficient and prediction loss.

Our code for simultaneous translation task is at `https://github.com/shirley-wu/simul-mt_temporally-correlated-task-scheduling`, and the code for stock price forecasting is at `https://github.com/microsoft/qlib/tree/main/examples/benchmarks/TCTS`.

2. Problem Setup and Our Algorithm

In this section, we first set up the problem of temporally correlated task scheduling for sequence learning, and then present our algorithm. Finally, we discuss the connection between our method and existing methods, and also provide an intuition why our method works.

2.1. Problem Setup

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input space and output space, and let $\mathcal{T} = \{T_1, \cdots, T_{|\mathcal{T}|}\}$ denote a set of temporally correlated tasks. Depending on applications, those tasks can be temporally correlated in different ways. We categorize them into two main classes and formalize them as follows.

(1) *Sequence processing with latency constraints*: Tasks in this category aim to process each token in an input sequence, one by one sequentially, under certain latency constraints. Representative examples include simultaneous translation (Ma et al., 2019), which targets at translating each source word without waiting for the end of the input sentence, and streaming speech recognition (He et al., 2019), which requires recognizing each word said by a speaker in real-time, etc.

Let x denote an input sequence and x_t denotes its t-th token. We focus on the setting where each token in x will be sequentially processed. Each task $T \in \mathcal{T}$ is associated with a non-negative integer δ_T, indicating that when processing the token x_t, how many additional tokens can be used. Note

that in conventional sequence learning where the complete source sequence is available, $\delta_T = \infty$. In the extreme case where no latency is allowed, $\delta_T = 0$. Let f_T denote the model for task T, which generates the output sequentially. $\tilde{y}$ denotes the output sequence of f_T based on x, and is initialized as an empty list. After processing x_t conditioned on $x_{\leq t+\delta_T} = (x_1, x_2, \cdots, x_{t+\delta_T})$, based on the processed representations of $x_t, x_{\leq t+\delta_T}$, and the already generated $\tilde{y}$, f_T needs to make a decision $\hat{y} = f_T(x_t, x_{\leq t+\delta_T}, \tilde{y})$. The decision could be to generate a single token, multiple tokens, or no token. After that, we append $\hat{y}$ to the end of $\tilde{y}$. We repeat the above steps to process all tokens in x.

(2) *Predicting the future*: Tasks in this category aim to predict some future values for a given time series. Representative examples include stock trend forecasting (Dai & Zhang, 2013), where we want to predict the price trend of an individual stock on some future days (e.g., tomorrow, the day after tomorrow, etc.) based on its historical prices, and weather forecasting (Shi et al., 2015; Kim et al., 2017), where we want to predict the weather of a city on a future day based on its historical weather.

Let x denote the input time series, and $y = (y_1, y_2, \cdots)$ denote the output sequence, where y_t is the value of the t-th future day/step. For a task $T \in \mathcal{T}$, the goal is to predict the value of the σ_T-th future day, i.e., $\tilde{y}_{\sigma_T}$, where σ_T is a positive integer and varies across different tasks T.

Each task has a different loss function $\ell(x, y, T)$, which is used to measure the distance between the predicted $\tilde{y}$ and the groundtruth y (in the first class of tasks) or $\tilde{y}_{\sigma_T}$ and y_{σ_T} (in the second class of tasks). Any task in $\mathcal{T}$ can be the main task, and the others are regarded as auxiliary tasks. In this work, our goal is to better leverage those auxiliary tasks to help the main task.

2.2. Our Algorithm

As aforementioned, our basic idea is to introduce a task scheduler, which adaptively selects an auxiliary training task to improve the performance of the main task. A basic assumption of our algorithm is that the models for all tasks in $\mathcal{T}$ can share parameters (except for a few individual parts like task embeddings), which is easy to satisfy in machine translation, stock trend forecasting, etc. Let $f(x; \theta)$ denote the model with parameter θ. Let $\varphi(x, y, \theta; \omega)$ denote the task scheduler parameterized by ω, which outputs an auxiliary task in $\mathcal{T}$ conditioned on an input-output pair (x, y) and the model $f(\cdots; \theta)$ of the main task. More specifically, we extract a low-dimensional feature $I_{x,y,\theta}$ to describe (x, y, θ) and use it as the actual input of the scheduler network φ. For ease of reference, we will alternatively use $\varphi(x, y, \theta; \omega)$ and $\varphi(I_{x,y,\theta}; \omega)$. φ can be different functions depending on applications, which will be discussed in Section 3 and 4.
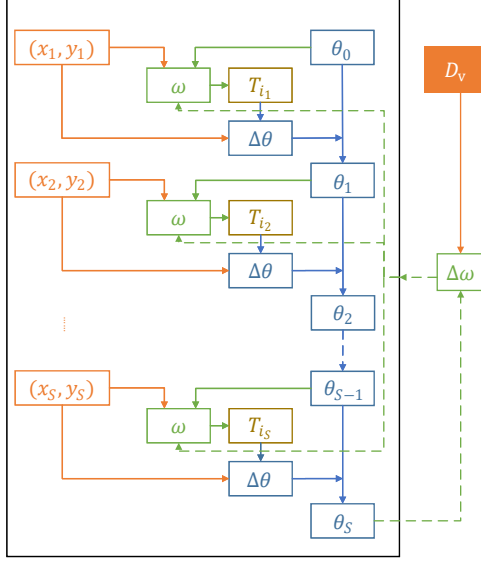
Figure 1: The optimization workflow of one episode. The solid lines and dashed lines represent the workflow of inner-optimization (*w.r.t.* θ) and outer-optimization (*w.r.t.* ω) respectively. At step s, with training data (x_s, y_s), the scheduler $\varphi(\cdots; \omega)$ chooses a suitable task T_{i_s} (green solid lines) to update the model $f(\cdots; \theta)$ (blue solid lines). After S steps, we evaluate the model f on the validation set D_v and update the scheduler φ (green dashed lines).

Now we have two sets of parameters to learn from the data: ω in the scheduler φ and θ in the model f. Let D_t and D_v denote the training and validation sets respectively. We formulate the training of f and φ as a bi-level optimization (Sinha et al., 2017), which is

$$\max_{\omega} \mathcal{M}(D_v; \theta(\omega)); \quad \text{s.t.,} \tag{1}$$

$$\theta(\omega) = \arg\min_{\theta} \frac{1}{|D_t|} \sum_{(x,y) \sim D_t} \mathbb{E}_{T \sim \varphi(x,y,\theta;\omega)} \ell(x, y, T; \theta).$$

In the above formulation, $\mathcal{M}(D_v; \theta(\omega))$ denotes the validation performance of the main task, and ℓ is the loss function of data (x, y) on task T. Intuitively, we expect that the scheduler can select a good auxiliary task for each input-output pair, so that eventually the validation performance of the model on the main task can be maximized.

Following the common practice (Liu et al., 2019; Fan et al., 2018), we optimize Eqn.(1) in an alternative way: We first fix ω and optimize θ, and then fix θ and optimize ω using the REINFORCE algorithm (Sutton & Barto, 2018). We repeat the process until convergence, as shown in Algorithm 1. The training process consists of E episodes (i.e., the outer loop), and each episode consists of S update iterations (i.e., the inner loop). The inner loop (from step 4 to step 9)

Algorithm 1 The optimization algorithm.

1: **Input:** Training episode E; internal update iterations S; learning rate η_1 of model $f(\cdots; \theta)$; learning rate η_2 of the scheduler $\varphi(\cdots; \omega)$; batch size B;
2: **for** $e \leftarrow 1 : E$ **do**
3:     Prepare a buffer to store states and actions: $\mathcal{B} = \{\}$;
4:     **for** $s \leftarrow 1 : S$ **do**
5:         Sample a minibatch of data $D_{e,s}$ from D_t with size B;
6:         For each $(x, y) \in D_{e,s}$, we sample a temporally correlated auxiliary task T according to the output distribution of $\varphi(x, y, \theta; \omega)$, and get $\tilde{D} = \{(x, y, T)\}$ where $|\tilde{D}| = B$;
7:         Add the inputs of φ (i.e., $I_{x,y,\theta}$) to $\mathcal{B}$:
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{(I_{x,y,\theta}, T) | (x, y, T) \in \tilde{D}\}$;
8:         Update model f:
        $\theta \leftarrow \theta - (\eta_1/B)\nabla_\theta \sum_{(x,y,T) \in \tilde{D}} \ell(x, y, T; \theta)$;
9:     **end for**
10:    Calculate the validation performance as the reward: $R_e = \mathcal{M}(D_v; \theta_{e,S})$;
11:    Update the scheduler:
        $\omega \leftarrow \omega + \eta_2 R_e \sum_{(I,T) \in \mathcal{B}} \nabla_\omega \log P(\varphi(I; \omega) = T)$.
12: **end for**
13: **Return:** θ.

---

aims to optimize θ, where we can update the parameter with any gradient-based optimizer like momentum SGD, Adam (Kingma & Ba, 2015), etc. The outer loop (from step 2 to step 11) aims to optimize ω. $\varphi(x, y, \theta; \omega)$ can be regarded as a policy network, where the state is the feature $I_{x,y,\theta}$, the action is the choice of task $T \in \mathcal{T}$, and the reward is the validation performance R_e (step 10). At the end of each episode, we update ω using REINFORCE algorithm (step 11). The workflow of one episode is shown in Figure 1.

2.3. Discussion

Our problem is related to several other learning settings.

(1) In multi-task learning (Zhang & Yang, 2017) (briefly, MTL), multiple tasks are jointly trained and mutually boosted. In MTL, each task is treated equally, while in our setting, we focus on the main task.

(2) Transfer learning (Pan et al., 2010) also leverages auxiliary tasks to boost a main task. Curriculum transfer learning (briefly, CL) (Shao et al., 2018; Dong et al., 2017) is one kind of transfer learning which schedules auxiliary tasks according to certain rules, e.g., from easy to difficult. Our problem can also be regarded as a special kind of transfer learning, where the auxiliary tasks are temporally correlated with the main task. Our learning process is dynamically controlled by a scheduler rather than some pre-defined rules.

(3) Our method is also related to Different Data Selection (DDS) (Wang et al., 2020). Both our work and DDS train a meta controller to improve the learning process on a given task via reinforcement learning. The main difference is that DDS aims to better select training data from a dataset,

while our work aims to better utilize temporally correlated tasks (and their corresponding data). Given a main task, our method constructs and schedules a series of temporal correlated tasks, while DDS works on the given task only.

Intuitively, training with temporally correlated auxiliary tasks helps the main task for two reasons: (1) The temporally correlated tasks either share the same output with overlapped inputs or the same input. Thus, joint training of them enables shared representation learning, just like general multi-task training (MTL), and consequently improves the main task; (2) The x/y pairs from correlated tasks can be viewed as noised augmentation of the training data for the main task, which helps the main task.

3. Application to Simultaneous Translation

In this section, we apply our algorithm to text-to-text simultaneous translation.

3.1. Task Description

Simultaneous neural machine translation (briefly, NMT) has attracted much attention recently. In contrast to standard NMT, where the NMT system can access the full input sentence, simultaneous NMT requires the model to conduct translation before reading the full input sentence. In this task, $\mathcal{X}$ and $\mathcal{Y}$ are collections of sentences in the source language and the target language. *Wait-k* (Ma et al., 2019) is a simple yet effective method in simultaneous NMT, where the translation is k words behind the source input. Formally, given a data pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ where the t-th token of x/y is x_t/y_t, when we read a prefix $(x_1, x_2, \cdots, x_k)$ (briefly denoted as $x_{\leq k}$), we need to output the first target token y_1; when reading $x_{\leq k+1}$, we should output y_2, etc. In comparison, for standard NMT, one can start translation after reading all tokens in x. A larger k makes the task easier and leads to better translation quality, but at the cost of a larger waiting time.

Under our framework, simultaneous translation belongs to the first class of tasks in Section 2.1. Let T_m denote *wait-m* task, and δ_{T_m} for task T_m is $m - 1$. Assuming *wait-k* is the main task T_{main}, then the temporally correlated tasks $\mathcal{T}$ are *wait-m* with different m values, i.e. $\mathcal{T} = \{T_1, T_2, \cdots, T_M\}$. In this work, we set $M = 13$. If the scheduler φ assigns a task T_m (i.e., *wait-m*) to a data pair (x, y), the loss function on this task is:

$$\ell(x, y, T_m; \theta) = \sum_{t=1}^{|y|} \log P(y_t | y_{\leq t-1}, x_{\leq t+m-1}; \theta). \quad (2)$$

3.2. Background

Previous work on simultaneous NMT can be categorized by whether using a fixed decoding strategy or an adaptive one.

Fixed strategies use pre-defined rules to determine when to read or to write a new token (Dalvi et al., 2018; Ma et al., 2019). *Wait-k* is a representative method (Ma et al., 2019), which achieves good results in terms of translation quality and controllable latency, and has been used in speech-related simultaneous NMT (Zhang et al., 2019; Ren et al., 2020).

For adaptive strategies, Cho & Esipova (2016) proposed wait-if-worse (WIW) and wait-if-diff (WID) methods which generate a new target word if its probability does not decrease (for WIW) or if the generated word is unchanged (for WID) after reading a new source token. Grissom II et al. (2014) and Gu et al. (2017) used reinforcement learning to train the read/write controller, while Zheng et al. (2019a) trained the controller in a supervised way. Alinejad et al. (2018) added a "predict" operator to the controller so that it can anticipate future source inputs. Zheng et al. (2019b) introduced a "delay" token into the target vocabulary indicating that the model should read a new word. Monotonic infinite lookback attention (MILk) used a hard attention model to determine when to read new tokens, and a soft attention model to perform translation (Arivazhagan et al., 2019). Ma et al. (2020) introduced multi-head attention to MILk and proposed monotonic multihead attention (MMA) with two variants: MMA-IL (Infinite Lookback) with higher translation quality and greater computational overhead, and MMA-H(ard) with higher computational efficiency.

Besides, Zheng et al. (2020a) extended *wait-k* to an adaptive strategy by training multiple *wait-m* models with different m's and adaptively selecting a decoding strategy during inference. Zheng et al. (2020b) explored a new setting, where at each timestep, the translation model over-generates the target words and corrects them in a timely fashion.

3.3. Settings

Datasets: For IWSLT'14 En→De, following (Edunov et al., 2018), we split $7k$ sentences from the training corpus for validation, and the test set is the concatenation of *tst2010, tst2011, tst2012, dev2010* and *dev2012*. For IWSLT'15 En→Vi, following (Ma et al., 2020), we use *tst2012* as the validation set and *tst2013* as the test set. For IWSLT'17 En→Zh, we concatenate *tst2013*, *tst2014* and *tst2015* as the validation set and use *tst2017* as the test set. For WMT'15 En→De, following (Ma et al., 2019; Arivazhagan et al., 2019), we use *newstest2013* as the validation set and use *newstest2015* as the test set. More details about datasets can be found in Appendix C.1.

Models: Following Ma et al. (2019), the translation model f is based on Transformer (Vaswani et al., 2017). However, different from Ma et al. (2019), we conduct incremental encoding, so that when encoding each source token, the model can only attend to its previous tokens. The computation complexity for encoding is $O(L_x^2)$ compared to $O(L_x^3)$

in Ma et al. (2019), where L_x is the length of the source sentence x. We find that our model is more efficient than Ma et al. (2019) with little accuracy drop. Detailed formulation and empirical comparison are in Appendix B.

For IWSLT En→Zh and En→Vi, we use the transformer small model, where the embedding dimension, feed-forward layer dimension, number of layers are 512, 1024 and 6 respectively. For IWSLT En→De, we use the same architecture but change the embedding dimension into 256. For WMT'15 En→De, we use the transformer big setting, where the above three numbers are 1024, 4096 and 6 respectively. The scheduler φ for each task is a multilayer perceptron (MLP) with one hidden layer and the `tanh` activation function. The size of the hidden layer is 256.

Input features of φ: The input of φ is a 7-dimension vector with the following features: (1) the ratios between the lengths of the source/target sentences to the average source/target sentence lengths in all training data (2 dimensions), i.e., $L_x/(\sum_{x'\in\mathcal{X}} L_{x'}/|\mathcal{X}|)$ and $L_y/(\sum_{y'\in\mathcal{Y}} L_{y'}/|\mathcal{Y}|)$; (2) the training loss over data (x, y) evaluated on the main task *wait-k*; (3) the average of historical training losses; (4) the validation loss of the previous epoch; (5) the average of historical validation loss; (6) the ratio of current training step to the total training iteration. The ablation study of feature selection is in Appendix E.1.

Choice of $\mathcal{M}$: The validation performance $\mathcal{M}$ is the negative validation loss with *wait-k* strategy. To stabilize training, the reward for the e-th episode (step 10 of Algorithm 1) is $R_e - R_{e-1}$, where R_{e-1} is the validation performance $\mathcal{M}$ of the previous episode.

Baselines: We implement the MTL and CL baselines discussed in Section 2.3: (1) For MTL, each task is randomly sampled. (2) For the curriculum transfer learning (briefly, CL), we start from the easiest task T_M and gradually move to the main task T_{main}. For *wait-k*, we implement another variant *wait-k**, where we train M baseline models on different *wait-m* tasks ($1 \leq m \leq M$), and pick the best model according to the validation performance on *wait-k* task. *Wait-k** is expected to bring additional improvements as pointed out by Ma et al. (2019). After that, we compare with several adaptive methods, including Wait-if-Worse (WIW), Wait-if-Diff (WID), MILk, MMA-IL and MMA-H (refer to Section 3.2 for a brief introduction). Finally, we combine our method with Zheng et al. (2020a). We leave the training details of all algorithms (optimizer, hyperparameter, etc) in Appendix C.2, and the details of baselines in Appendix C.3.

Evaluation: We use BLEU to measure the translation quality, and use Average Proportion (AP) and Average Lagging (AL) to evaluate the translation latency. AP measures the average proportion of source symbols required for transla-

tion, and AL measures the average number of delayed words (see Appendix A.1. for details). Following the common practice (Ma et al., 2019; 2020), we show the BLEU-AP and BLEU-AL curves to demonstrate the tradeoff between quality and latency. For IWSLT'14 En→De and IWSLT'15 En→Vi, we use `multi-bleu.perl` to evaluate BLEU scores; for IWSLT'17 En→Zh and WMT'15 En→De, we use `sacreBLEU` to evaluate the detokenized BLEU scores.

3.4. Results and Analysis

We first compare our method with MTL, CL and *wait-k** on IWSLT datasets. The BLEU-latency curves are shown in Figure 2, and the BLEU scores of En→Vi are reported in Table 1. For the *wait-k**, we also report the optimal training task (i.e., k^*) for each main task *wait-k*. BLEU scores of all language pairs are left in Appendix D. Experiments on more heuristic baselines are in Appendix E.2.

(a) BLEU-AP, En→Vi.

(b) BLEU-AL, En→Vi.

(c) BLEU-AP, En→De.

(d) BLEU-AL, En→De.
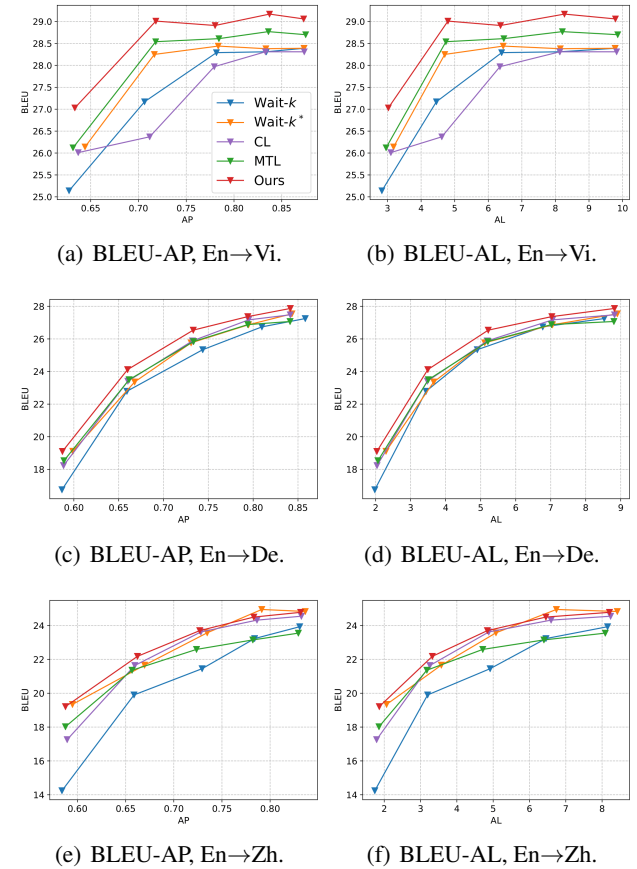
(e) BLEU-AP, En→Zh.

(f) BLEU-AL, En→Zh.

Figure 2: Translation quality against latency metrics (AP and AL) on IWSLT tasks.

We have the following observations:

(1) Generally, our method consistently performed the best across different translation tasks in terms of both translation

k	wait-k	wait-k^*/ k^*	CL	MTL	Ours
1	25.14	26.14 / 5	26.01	26.12	27.03
3	27.17	28.25 / 5	26.37	28.54	29.01
5	28.29	28.44 / 9	27.97	28.61	28.91
7	28.31	28.38 / 13	28.31	28.77	29.17
9	28.39	28.39 / 9	28.31	28.70	29.06

Table 1: BLEU scores on IWSLT En→Vi task.

quality and controllable latency. As shown in Table 1, our method achieves the highest BLEU scores among all baselines. In Figure 2, the curve for our method (i.e., the red one) is on the top in most cases, which indicates that given specific latency (e.g., AP or AL), we can achieve the best translation quality. The results of significance tests are left in Appendix D.

(2) MTL and CL can outperform the vanilla baseline, which demonstrates the effectiveness of leveraging the temporally correlated tasks. However, the improvements are not consistent, and it is hard to tell which baseline is better. CL slightly outperforms MTL on En→De. However, on En→Zh, CL performs better at lower latency while MTL performs better at higher latency. In comparison, the improvement brought by our method is much more consistent.

(3) Wait-k^* also outperforms wait-k. Specifically, we notice that the optimal training k^* is always larger than k for the main task. This is consistent with the observation in Ma et al. (2019). However, compared with CL and MTL, wait-k^* does not have a consistent advantage. This shows that the improvements of MTL, CL and our method is not due to the existence of some "best" task in $\mathcal{T}$, and further demonstrates the importance of adaptively using all auxiliary tasks.

The results on WMT'15 En→De, whose training corpus is larger, are shown in Figure 3. Wait-k^*, MTL and CL do not bring much improvement compared to vanilla wait-k. Our method consistently outperforms all baselines, which demonstrates the effectiveness of our method on large datasets. We further evaluate them on WMT'14 and WMT'16 test sets and obtain similar conclusions (see Appendix D for details).

Comparison with adaptive simultaneous NMT methods: We further compare our method with WIW, WID, MILk, MMA-IL and MMA-H on IWSLT En→Vi. The BLEU-AL curves are shown in Figure 4(a) and the BLEU-AP curves are in Appendix D. When AL ≥ 5.0, our method outperforms all baseline models, and when AL < 5.0, our method performs slightly worse than MMA-IL and MMA-H. A possible reason is that our method does not explicitly reduce the latency, but focuses on improving performance under given latency constraints.
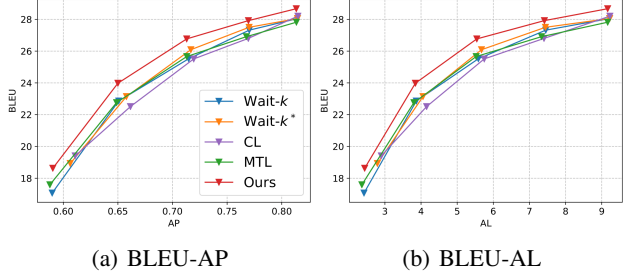
We further compare and combine our method with Zheng

(a) BLEU-AP   (b) BLEU-AL

Figure 3: Translation quality against latency metrics (AP and AL) on WMT'15 En→De.

et al. (2020a). We conduct experiments on two variants of Zheng et al. (2020a), where the wait-m models are obtained through vanilla wait-k (denoted by "Zheng et al.") and our strategy respectively (denoted by "Zheng et al. + Ours"). The BLEU-AL curves are shown in Figure 4(b), and the BLEU-AP curves are in Appendix D. We can see that: (1) our method catches up with Zheng et al. (2020a), which is built upon 10 models in total (wait-$1, 2, \cdots, 10$); (2) after combing our approach with Zheng et al. (2020a), the performance can be further boosted, which shows that our method is complementary to adaptive inference strategies like Zheng et al. (2020a).
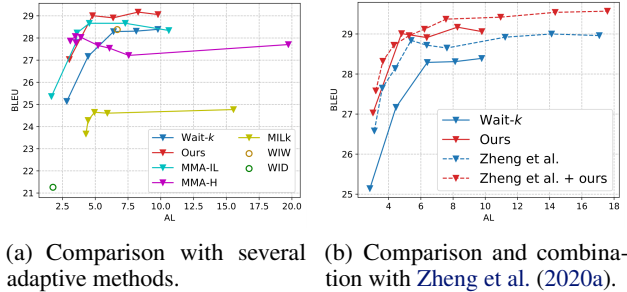

(a) Comparison with several adaptive methods.   (b) Comparison and combination with Zheng et al. (2020a).

Figure 4: BLEU-AL comparison between our method and baselines on En→Vi.

Computational overhead: To evaluate the additional computational overhead brought by our method, we compare the training speed of standard wait-k and our method on wait-3 (measured by the number of batches per second). Results on IWSLT datasets are summarized in Table 2. Our method requires $20\% \sim 30\%$ additional training time, which is acceptable considering the performance improvements. The major overhead comes from computing the training loss by wait-k. To verify that, we record the training speed of our method without the second and third input features of φ, i,e, the training loss over data (x, y) evaluated on wait-k, and the average of historical training losses (denoted by "Ours w/o feature (2,3)"). Without these two features, the training

speed of our method is close to *wait-k*.

Task	*wait-k*	Ours	Ours w/o feature (2,3)
En→De	5.3	4.0 (-23%)	5.2 (-2%)
En→Vi	1.5	1.1 (-27%)	1.4 (-7%)
En→Zh	2.5	1.8 (-28%)	2.4 (-4%)

Table 2: Comparison of training speed (batch / sec) between *wait-k* and our methods.

Strategy analysis: In Figure 5, we visualize the distribution of *wait-m* tasks obtained by the scheduler for two main tasks, *wait-3* and *wait-9* on IWSLT En→Zh dataset. We show the frequency of each *wait-m* task sampled by the scheduler φ at the 0th, 1st, 5th, 10th and 40th episode.

We observed that the scheduler uniformly samples different tasks at first, and then the strategies converge within 10 episodes. After convergence, the scheduler mainly samples several specific tasks, i.e., $m = 5, 9, 12$ for *wait-3*, and $m = 5, 12$ for *wait-9*. The task that both schedulers prefer most is $m = 12$, which is close to the optimal training task *wait-k^** ($k^* = 11$) for both *wait-3* and *wait-9*. This demonstrates that the scheduler can learn which task is more beneficial to the main task. However, it is worth noting that the scheduler also samples other tasks, which shows the importance of training with multiple tasks instead of one single optimal task. For example, the scheduler for *wait-9* samples both *wait-5* and 9 with a probability of about 0.1.

4. Application to Stock Trend Forecasting

4.1. Task Description

Stock trend forecasting is a core problem in stock market investment. Following the common practice, in this task, $\mathcal{X}$ represents the historical transactions data including both prices and trading volumes, and $\mathcal{Y}$ denotes the stock price movement in future days.

At the t-th day, a specific stock can be represented by a 5-dimension feature vector x_t including the opening price, closing price (denoted as p_t), the highest price, the lowest price and the trading volume. The task $T_k \in \mathcal{T}$ is to forecast the stock price rise percent of the next k-th day, i.e., $y_t = \frac{p_{t+k}}{p_{t+k-1}} - 1$. This task lies in the second type of classes as introduced in Section 2.1. In stock forecasting, we usually make predictions based on a window of historical inputs of length L, i.e., $(x_{t-L+1}, x_{t-L+2}, \cdots, x_t)$. We explore three different main tasks in this work: forecasting the stock rise trend of the next 1st, 2nd and 3rd days (i.e., T_1, T_2 and T_3). As mentioned by Tang et al. (2020), all the above three tasks are important because investors tend to invest in stocks with rising price trends in a continuous period, rather than caring the next trading day only.

4.2. Background

Traditional methods for stock trend forecasting are based on time-series analysis models, such as the Auto-regressive Integrated Moving Average (ARMA) and Kalman Filters (Yan & Guosheng, 2015). With the development of deep learning, applying neural network to stock forecasting becomes a new trend due to its better performance on noisy data. GRU (Chung et al., 2014) is one of the most representative deep neural networks in stock trend forecasting (Hu et al., 2018). By utilizing the reset gate and update gate, GRU can adaptively choose the information to be kept or dropped, which makes it relatively robust to the diverse and noisy sequential data in the stock market. Other types of recurrent neural networks are also leveraged such as LSTM, SFM (Zhang et al., 2017b), ALSTM (Qin et al., 2017b), etc. Recent works start to use graph neural networks (e.g. GAT (Veličković et al., 2017)) to model different stocks so that the information can be propagated along with the pre-defined relations (Feng et al., 2019; Kim et al., 2019).

In addition to neural networks, many machine learning methods have been successfully applied to forecast stock future trends. A particularly powerful branch is the tree-based approach, such as LightGBM (Ke et al., 2017) and Random Forest. They have outstanding performance and certain interpretability, so they are widely used in stock market investment (Khaidem et al., 2016; Tan et al., 2019).

4.3. Settings

We conduct experiments on individual stocks in the Chinese A-share market. We explore three groups of temporally correlated tasks: $\mathcal{T}_3 = \{T_1, T_2, T_3\}$, $\mathcal{T}_5 = \{T_1, T_2, \cdots, T_5\}$ and $\mathcal{T}_{10} = \{T_1, T_2, \cdots, T_{10}\}$.

Dataset : We use the historical transaction data for 300 stocks on CSI300 (CSI300, 2008) from 01/01/2008 to 08/01/2020. We split the data into training (01/01/2008-12/31/2013), validation (01/01/2014-12/31/2015), and test sets (01/01/2016-08/01/2020) based on the transaction time. All the data is provided by Yang et al. (2020).

Models: The forecasting model f is built on top of a 2-layer GRU with hidden dimension 32. The length of the input sequence (i.e., the L in Section 4.1) is 60. The scheduler φ is a single-layer feed-forward network with ReLU activation. The hidden size of φ is 32.

Input features of φ: The features consist of four parts: (1) the data $(x, y) \in D_t$ itself; (2) the training loss of the previous iteration; (3) the forecasting results of the previous iteration; (4) the labels of each task in $\mathcal{T}$.

Baselines: Similar to Section 3.3, we implement MTL and CL as baselines by using temporally correlated tasks. In CL, we start from the easiest task T_1, and then gradually move
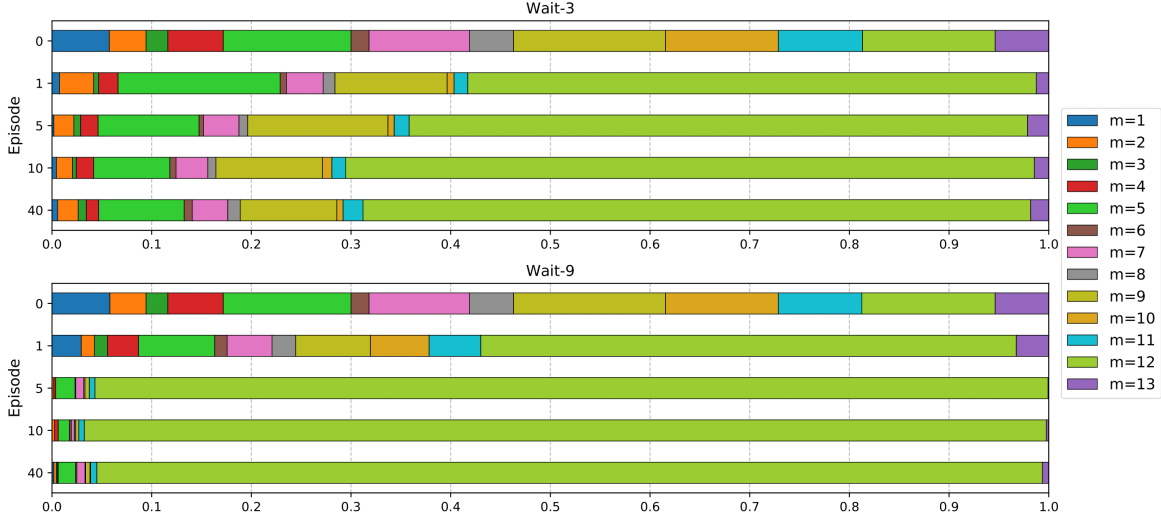
Figure 5: The distribution of *wait-m* tasks for *wait*-3 and *wait*-9 on En→Zh dataset.

to the most difficult one. Besides, we implement different models for stock forecasting for comparison, including simple feed-forward networks (denoted by MLP), lightGBM (briefly, LGB) (Ke et al., 2017) and GAT (Veličković et al., 2017; Feng et al., 2019). Note that for GAT, we do not leverage the specific relation between stocks.

Evaluation: We use rank correlation coefficient (Zhang et al., 2020) (briefly, RankIC) and the mean square error (briefly, MSE) to evaluate the forecasting results. RankIC can be regarded as a correlation between the prediction and the groundtruth results (the larger, the better). MSE measures the gap between prediction results and the groundtruth results (the smaller, the better). Due to space limitation, we put the evaluation results using ICIR (a measurement that takes stability into consideration) in Appendix F.

4.4. Results and Analysis

The results of stock trend forecasting are reported in Table 3. We have the following observations:

(1) With different sets of temporally correlated auxiliary tasks, i.e., $\mathcal{T}_3$, $\mathcal{T}_5$ and $\mathcal{T}_{10}$, our method significantly outperforms MTL and CL on most settings, which demonstrates the effectiveness of leveraging an adaptive scheduler.

(2) Using a larger temporally correlated task set does not always bring improvements. Taking task T_1 as an example, our method achieves the best performance when using 5 temporally correlated tasks, while using 10 hurts the performance. The reason is that more tasks will introduce additional noise to the training.

(3) There are more advanced methods than GRU like GAT, leveraging temporally correlated tasks can achieve superior

Methods	Task-1	Task-2	Task-3
GRU	0.049 / 1.903	0.018 / 1.972	0.014 / 1.989
MLP	0.023 / 1.961	0.022 / 1.962	0.015 / 1.978
LGB	0.038 / 1.883	0.023 / 1.952	0.007 / 1.987
GAT	0.052 / 1.898	0.024 / 1.954	0.015 / 1.973
MTL ($\mathcal{T}_3$)	0.061 / 1.862	0.023 / 1.942	0.012 / **1.956**
CL ($\mathcal{T}_3$)	0.051 / 1.880	0.028 / 1.941	0.016 / 1.962
Ours ($\mathcal{T}_3$)	**0.071 / 1.851**	**0.030 / 1.939**	**0.017** / 1.963
MTL ($\mathcal{T}_5$)	0.057 / 1.875	0.021 / 1.939	0.017 / 1.959
CL ($\mathcal{T}_5$)	0.056 / 1.877	0.028 / 1.942	0.015 / 1.962
Ours ($\mathcal{T}_5$)	**0.075 / 1.849**	**0.032 /1.939**	**0.021 / 1.955**
MTL ($\mathcal{T}_{10}$)	0.052 / 1.882	0.020 / 1.947	0.019 / 1.952
CL ($\mathcal{T}_{10}$)	0.051 / 1.882	0.028 / 1.950	0.016 / 1.961
Ours ($\mathcal{T}_{10}$)	**0.067 / 1.867**	**0.030 / 1.960**	**0.022 / 1.942**

Table 3: Results of stock trend forecasting. The first four rows are the results of various model architectures without leveraging temporally correlated tasks. The following rows are the results with different algorithms (MTL, CL and ours) and different auxiliary task sets ($\mathcal{T}_3$, $\mathcal{T}_5$ and $\mathcal{T}_{10}$). For each cell, the RankIC is put on the left and MSE on the right.

performance than the strong models. Currently, our model is based on GRU, and we will combine our method with GAT and lightGBM in the future for more improvements.

5. Conclusions and Future Work

In this work, we formulated and studied the problem of temporally correlated task scheduling, which can cover many applications including sequence processing with latency constraints (e.g., simultaneous machine translation) and sequence future prediction (e.g. stock trend forecasting). Ex-

periments on four translation tasks and one stock prediction task demonstrate the effectiveness of our approach.

For future work, we will apply our method to more applications like action prediction (Kong et al., 2020; Cai et al., 2019), streaming speech recognition, weather forecasting, game AI (Li et al., 2020; Vinyals et al., 2019), etc. We will also design better methods for the problem. Another interesting direction is to theoretically understand the problem and analyze our method.

References

Alinejad, A., Siahbani, M., and Sarkar, A. Prediction improves simultaneous neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3022–3027, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1337. URL https://www.aclweb.org/anthology/D18-1337.

Arivazhagan, N., Cherry, C., Macherey, W., Chiu, C.-C., Yavuz, S., Pang, R., Li, W., and Raffel, C. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1313–1323, Florence, Italy, 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P19-1126.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2014. URL https://arxiv.org/abs/1409.0473.

Cai, Y., Li, H., Hu, J.-F., and Zheng, W.-S. Action knowledge transfer for action prediction with partial videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 8118–8125, 2019.

Cho, K. and Esipova, M. Can neural machine translation do simultaneous translation? *CoRR*, abs/1606.02012, 2016. URL http://arxiv.org/abs/1606.02012.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

CSI300. Csi 300, 2008. URL http://www.csindex.com.cn/en/indices/index-detail/000300.

Dai, Y. and Zhang, Y. Machine learning in stock price trend forecasting, 2013.

Dalvi, F., Durrani, N., Sajjad, H., and Vogel, S. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 493–499, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2079. URL https://www.aclweb.org/anthology/N18-2079.

Dong, Q., Gong, S., and Zhu, X. Multi-task curriculum transfer deep learning of clothing attributes. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 520–529. IEEE, 2017.

Edunov, S., Ott, M., Auli, M., Grangier, D., and Ranzato, M. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 355–364, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1033. URL https://www.aclweb.org/anthology/N18-1033.

Fan, Y., Tian, F., Qin, T., Li, X., and Liu, T. Learning to teach. *CoRR*, abs/1805.03643, 2018. URL http://arxiv.org/abs/1805.03643.

Feng, F., He, X., Wang, X., Luo, C., Liu, Y., and Chua, T.-S. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)*, 37(2):1–30, 2019.

Grissom II, A., He, H., Boyd-Graber, J., Morgan, J., and Daumé III, H. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1342–1352, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1140. URL https://www.aclweb.org/anthology/D14-1140.

Gu, J., Neubig, G., Cho, K., and Li, V. O. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 1053–1062, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-1099.

He, Y., Sainath, T. N., Prabhavalkar, R., McGraw, I., Alvarez, R., Zhao, D., Rybach, D., Kannan, A., Wu, Y., Pang, R., Liang, Q., Bhatia, D., Shangguan, Y., Li, B.,

Pundak, G., Sim, K. C., Bagby, T., Chang, S., Rao, K., and Gruenstein, A. Streaming end-to-end speech recognition for mobile devices. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6381–6385, 2019. doi: 10.1109/ICASSP.2019.8682336.

Hu, Z., Liu, W., Bian, J., Liu, X., and Liu, T.-Y. 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*, pp. 261–269, 2018.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 3146–3154. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

Khaidem, L., Saha, S., and Dey, S. R. Predicting the direction of stock market prices using random forest. *arXiv preprint arXiv:1605.00003*, 2016.

Kim, R., So, C. H., Jeong, M., Lee, S., Kim, J., and Kang, J. Hats: A hierarchical graph attention network for stock movement prediction. *arXiv preprint arXiv:1908.07999*, 2019.

Kim, S., Hong, S., Joh, M., and Song, S. Deeprain: Conv-lstm network for precipitation prediction using multi-channel radar data. *CoRR*, abs/1711.02316, 2017. URL http://arxiv.org/abs/1711.02316.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/pdf/1412.6980.pdf.

Kong, Y., Tao, Z., and Fu, Y. Adversarial action prediction networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(3):539–553, 2020.

Li, J., Koyamada, S., Ye, Q., Liu, G., Wang, C., Yang, R., Zhao, L., Qin, T., Liu, T.-Y., and Hon, H.-W. Suphx: Mastering mahjong with deep reinforcement learning. *arXiv preprint arXiv:2003.13590*, 2020.

Liu, H., Simonyan, K., and Yang, Y. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Ma, M., Huang, L., Xiong, H., Zheng, R., Liu, K., Zheng, B., Zhang, C., He, Z., Liu, H., Li, X., Wu, H., and Wang, H. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3025–3036, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1289. URL https://www.aclweb.org/anthology/P19-1289.

Ma, X., Pino, J., Cross, J., Puzon, L., and Gu, J. Monotonic multihead attention. In *8th International Conference on Learning Representations*, 2020.

Pan, S. J., Yang, Q., Fan, W., and (ph. D, S. J. P. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 2010.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., and Cottrell, G. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017a.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., and Cottrell, G. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017b.

Ren, Y., Liu, J., Tan, X., Zhang, C., Qin, T., Zhao, Z., and Liu, T. Simulspeech: End-to-end simultaneous speech to text translation. In *ACL*, 2020.

Shao, K., Zhu, Y., and Zhao, D. Starcraft micromanagement with reinforcement learning and curriculum transfer learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 3(1):73–84, 2018.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., and Woo, W.-c. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pp. 802–810, Cambridge, MA, USA, 2015. MIT Press.

Sinha, A., Malo, P., and Deb, K. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Tan, Z., Yan, Z., and Zhu, G. Stock selection with random forest: An exploitation of excess return in the chinese stock market. *Heliyon*, 5(8):e02310, 2019.

Tang, Z., Zhang, T., Wu, J., Du, X., and Chen, K. Multistep-ahead stock price forecasting based on secondary decomposition technique and extreme learning machine optimized by the differential evolution algorithm. *Mathematical Problems in Engineering*, 2020, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Wang, X., Pham, H., Michel, P., Anastasopoulos, A., Carbonell, J., and Neubig, G. Optimizing data usage via differentiable rewards. In *International Conference on Machine Learning*, pp. 9983–9995. PMLR, 2020.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Yan, X. and Guosheng, Z. Application of kalman filter in the prediction of stock price. In *5th International Symposium on Knowledge Acquisition and Modeling (KAM 2015)*. Atlantis Press, 2015.

Yang, X., Liu, W., Zhou, D., Bian, J., and Liu, T.-Y. Qlib: An ai-oriented quantitative investment platform. *arXiv preprint arXiv:2009.11189*, 2020.

Zhang, C., Tan, X., Liu, J., Ren, Y., Qin, T., and Liu, T.-Y. Simuls2s: End-to-end simultaneous speech to speech translation. *Openreview*, 2019. URL https://openreview.net/pdf?id=Ske_56EYvS.

Zhang, F., Guo, R., and Cao, H. Information coefficient as a performance measure of stock selection models, 2020.

Zhang, L., Aggarwal, C., and Qi, G.-J. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2141–2149. ACM, 2017a.

Zhang, L., Aggarwal, C., and Qi, G.-J. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 2141–2149, 2017b.

Zhang, Y. and Yang, Q. A survey on multi-task learning. *CoRR*, abs/1707.08114, 2017. URL http://arxiv.org/abs/1707.08114.

Zheng, B., Zheng, R., Ma, M., and Huang, L. Simpler and faster learning of adaptive policies for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1349–1354, Hong Kong, China, November 2019a. Association for Computational Linguistics. doi: 10.18653/v1/D19-1137. URL https://www.aclweb.org/anthology/D19-1137.

Zheng, B., Zheng, R., Ma, M., and Huang, L. Simultaneous translation with flexible policy via restricted imitation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5816–5822, Florence, Italy, July 2019b. Association for Computational Linguistics. doi: 10.18653/v1/P19-1582. URL https://www.aclweb.org/anthology/P19-1582.

Zheng, B., Liu, K., Zheng, R., Ma, M., Liu, H., and Huang, L. Simultaneous translation policies: From fixed to adaptive. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2847–2853, 2020a. doi: 10.18653/v1/2020.acl-main.254. URL https://www.aclweb.org/anthology/2020.acl-main.254.

Zheng, R., Ma, M., Zheng, B., Liu, K., and Huang, L. Opportunistic decoding with timely correction for simultaneous translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 437–442, Online, July 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.42. URL https://www.aclweb.org/anthology/2020.acl-main.42.