Temporally Correlated Task Scheduling for Sequence Learning (Appendix)

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]

A. Mathematical definitions

A.1. Latency metrics definitions

Given the input sentence x and the output sentence y, let L_x and L_y denote the length of x and y respectively. Define a function $g(t)$ of decoding step t, which denotes the number of source tokens processed by the encoder when deciding the target token y_t. For *wait-k* strategy, $g(t) = \min\{t + k - 1, L_x\}$. The definitions of Average Proportion (AP) and Average Lagging (AL) are in Eqn.(1) and Eqn.(2) respectively.

$$\mathrm{AP}_g(x, y) = \frac{1}{|x||y|} \sum_{t=1}^{|y|} g(t); \tag{1}$$

$$\mathrm{AL}_g(x, y) = \frac{1}{\tau_g(|x|)} \sum_{t=1}^{\tau_g(|x|)} \left(g(t) - \frac{t-1}{|y|/|x|} \right), \tag{2}$$

$$\text{where} \quad \tau_g(|x|) = \min\{t | g(t) = |x|\}.$$

We use the scripts provided by Ma et al. (2019) to calculate AP and AL scores.

A.2. Mathematical formulation of curriculum transfer learning

For a group of temporally correlated tasks $\mathcal{T} = \{T_1, T_2, \cdots, T_M\}$, suppose $\mathcal{T}$ is ordered by task difficulty. That is, T_M is the easiest task, followed by T_{M-1}, T_{M-2}, etc, and T_1 is the hardest task. In the curriculum transfer learning (briefly, CL) baseline, we gradually change the training task T_m from the easiest task T_M to the main task, supposedly, T_k. The mathematical formulations are shown as follows:

$$m = M - \lfloor \frac{t-1}{t_{\max}} (M - k + 1) \rfloor \tag{3}$$

where t denotes the current update number ($t = 1, 2, ..., t_{\max}$), and $t_{\max}$ denotes the total update number.

B. Model architecture

Following Ma et al. (2019), our model for simultaneous NMT is based on Transformer model (Vaswani et al., 2017). The model includes an encoder and a decoder, which are used for incrementally processing the source and target sentences respectively. Both the encoder and decoder are stacked of L blocks.

In Ma et al. (2019), every time a new token is read, the encoder needs to re-encode the entire sentence, which is computationally expensive. To improve the efficiency, we adopt incremental encoding, so that after reading a new token, we only encode the new token rather than updating the representations of the whole sentence. The details are as follows:

(1) *Incremental encoding*: Let h_t^l denote the output of the t-th position from block l. For ease of reference, let $H_{\leq t}^l$ denote $\{h_1^l, h_2^l, \cdots, h_t^l\}$, and let h_t^0 denote the embeddings of the t-th token. An attention model $\mathtt{attn}(q, K, V)$, takes a query

[1]University of Science and Technology of China, Hefei, Anhui, China [2]Microsoft Research, Beijing, China. Correspondence to: Yingce Xia <yingce.xia@microsoft.com>.

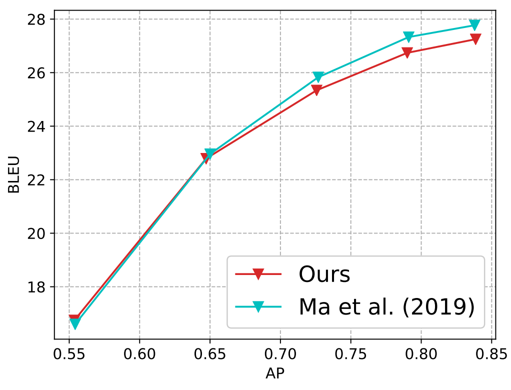
$q \in \mathbb{R}^d$, a set of keys K and values V as inputs. K and V are of equal size, $q \in \mathbb{R}^d$ where $d \in \mathbb{Z}_+$ is the dimension, $k_i \in \mathbb{R}^d$ and $v_i \in \mathbb{R}^d$ are the i-th key and value. `attn` is defined as follows:

$$\texttt{attn}(q, K, V) = \sum_{i=1}^{|K|} \alpha_i W_v v_i,\ \alpha_i = \frac{\exp((W_q q)^\top (W_k k_i))}{Z},\ Z = \sum_{i=1}^{|K|} \exp((W_q q)^\top (W_k k_i)), \tag{4}$$

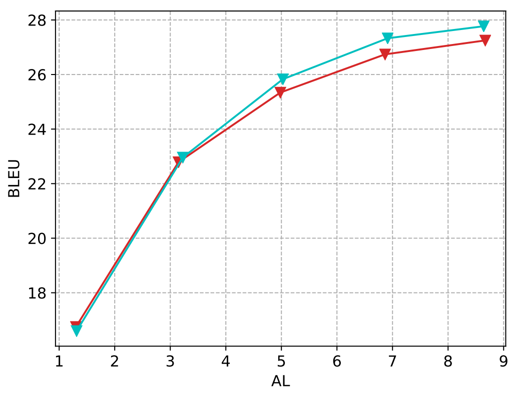
where W's are the parameters to be optimized. In the encoder side, h_t^l is obtained in a unidirectional way: $h_t^l = \texttt{attn}(h_t^{l-1}, H_{\le t}^{l-1}, H_{\le t}^{l-1})$. That is, the model can only attend to the previously generated hidden representations, and the computation complexity is $O(L_x^2)$. In comparison, (Ma et al., 2019) still leverages bidirectional attention, whose computation complexity is $O(L_x^3)$. We find that unidirectional attention is much more efficient than bidirectional attention without much accuracy drop (see Appendix B for details).

(2) *Incremental decoding*: Since we use *wait-k* strategy, the decoding starts before reading all inputs. At the t-th decoding step, the decoder can only read $x_{\le t+k-1}$. When $t \le L_x - k$, the decoder greedily generates one token at each step, i.e., the token is $y_t = \arg\max_{w \in \mathcal{V}} P(w | y_{\le t-1}; H_{\le t+k-1}^L)$, where $\mathcal{V}$ is the vocabulary of the target language. When $t > L_x - k$, the model has read the full input sentence and can generate words using beam search (Ma et al., 2019).

We compare the performance of original Ma et al. (2019) and our modified model on IWSLT'14 En→De dataset, and the results are in Figure 1, On IWSLT'14, we observe that the performance of our model slightly drops compared to Ma et al. (2019). However, the computational cost of Ma et al. (2019) is much larger than our modified model. For example, the inference speed of our *wait*-9 model is 57.39 sentences / second, while the inference speed of Ma et al. (2019) is 6.48 sentences / second.



(a) BLEU-AP, IWSLT'14 En→De



(b) BLEU-AL, IWSLT'14 En→De

Figure 1: Ablation study of different model architectures on IWSLT'14 En→De dataset and WMT'15 En→De dataset.

C. More detailed settings about experiments

C.1. Detailed introduction of the datasets

For IWSLT'14 En→De, following (Edunov et al., 2018), we lowercase all words, tokenize them and apply BPE with $10k$ merge operations jointly to the source and target sequences. We split $7k$ sentences from the training corpus for validation and the remaining $160k$ sequences are left as the training set. The test set is the concatenation of *tst2010, tst2011, tst2012, dev2010* and *dev2012*, which consists of 6750 sentences.

For IWSLT'15 En→Vi, following (Ma et al., 2020), we tokenize the data and replace words with frequency less than 5 by `<unk>`[1]. We use *tst2012* as the validation set and *tst2013* as the test set. The training, validation and test sets contains $133k$, 1268 and 1553 sentences respectively.

For IWSLT'17 En→Zh, we tokenize the data and apply BPE with $10k$ merge operations independently to the source and

[1]The data is downloaded from `https://nlp.stanford.edu/projects/nmt/`, which has been tokenized already.

target sequences[2]. We use the concatenation of *tst2013*, *tst2014* and *tst2015* as the validation set and use *tst2017* as the test set. The training, validation and test sets contains $235k$, 3874 and 1459 sentences respectively. For WMT'15 En$\leftrightarrow$De, we follow the setting in (Ma et al., 2019; Arivazhagan et al., 2019). We tokenize the data, apply BPE with $32k$ merge operations jointly to the source and target sentences, and get a training corpus with $4.5M$ sentences. We use *newstest2013* as the validation set and use *newstest2015* as the test set.

C.2. Detailed training strategy

For the translation model, we use Adam (Kingma & Ba, 2015) optimizer with initial learning rate 5×10^{-4} and `inverse_sqrt` scheduler (see Section 5.3 of (Vaswani et al., 2017) for details). The batch size and the number of GPUs of IWSLT En$\rightarrow$De, En$\rightarrow$Vi, En$\rightarrow$Zh and WMT'15 En$\rightarrow$De are 4096×1GPU, 16000×1GPU, 4000×1GPU and $3584 \times 8 \times 16$GPU respectively. For IWSLT tasks, the learning rate η is grid searched from $\{5 \times 10^{-4}, 5 \times 10^{-5}, 5 \times 10^{-6}, 5 \times 10^{-7}\}$ with vanilla SGD optimizer, and the internal update iteration S is grid searched from $\{\frac{1}{2}S_e, S_e, 2S_e\}$, where S_e is the number of updates in an epoch of the translation model training. For WMT'15 En$\rightarrow$De, due to resource limitation, we do not train the translation model from scratch. The translation model is warm started from pretrained *wait-k* model, the learning rate is set as 5×10^{-5}, and the internal update iteration S is 16.

C.3. Detailed baseline implementation

In this section, we introduce how we reproduce baseline models and get the results on En$\rightarrow$Vi task.

For MILk (Arivazhagan et al., 2019) and MMA (Ma et al., 2020), we do not reproduce the results, but directly use the results reported in Ma et al. (2020). Note that the results for MILk is reproduced by Ma et al. (2020).

For WIW and WID (Cho & Esipova, 2016), we pre-train a standard translation model, using the exact same architecture and training hyperparameters as our method and in (Ma et al., 2020). For fair comparison, we adopt bi-directional attention.

For Zheng et al. (2020), we use the pre-trained *wait-k* model and the model obtained through our results ($k = 1, 2, 3, ..., 10$). We use only single model rather than ensemble. As in Zheng et al. (2020), the thresholds of different k values are obtained in this way: $\rho_i = \rho_1 - (i-1) * (\rho_1 - \rho_{10})/9$, where we test with $\rho_1 \in \{0.2, 0.4, 0.6, 0.8\}$, $\rho_{10} = 0$; and $\rho_1 = 1$, $\rho_{10} \in \{0, 0.2, 0.4, 0.6, 0.8\}$.

D. Supplemental results

In this section, we report the specific BLEU scores and some additional results. The BLEU scores for IWSLT tasks and WMT'15 En$\rightarrow$De task are reported in Table 1 and Table 2 respectively. We further evaluate the baselines and our methods on WMT'14 and WMT'18 test sets, and report the BLEU-latency curves in Figure 3. The BLEU-AL curves of our methods and baselines on IWSLT'15 En$\rightarrow$Vi are reported in Figure 2(a), the BLEU-AL curves of our method and Zheng et al. (2020) are in Figure 2(b).

For significance tests, we compare our method with MTL on all 20 translation tasks (En$\rightarrow\{$Vi, Zh$\}$ tasks and two En$\rightarrow$De tasks, where each task is associated with 5 settings, i.e. *wait*-1,3,5,7,9), and find that our method significantly outperforms MTL on 16 tasks with $p < 0.05$, where 11 out of them are with $p < 0.01$.

E. Additional ablations and analysis on simultaneous translation

E.1. Feature selection

To emphasize the importance of the selected features in Section 3.3 in the main content, we provide four groups of ablation study, where in each group some specific features are excluded: (i) source and target sentence lengths; (ii) current training loss and average historical training loss; (iii) current validation loss and average historical validation loss; (iv) training step. We work on IWSLT'14 En$\rightarrow$De task and study the effect to *wait*-$3, 5, 7$.

The results are shown in Table 3. We report the BLEU scores only, since the latency metrics (AP and AL) are not significantly influenced. Removing any feature causes performance drop, indicating that they all contribute to the decision making. Specifically, network status information including validation performance (feature iii) and training stage (feature iv) is more

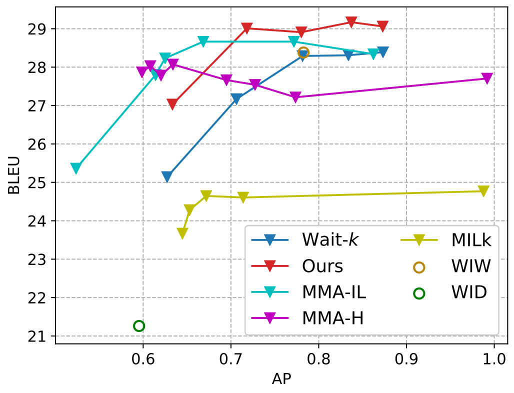
[2]The Chinese sentences are tokenized using Jieba (`https://github.com/fxsjy/jieba`).

Task	*wait-k*	*wait-k**/ best k^*	CL	MTL	Ours
En→De ($k = 1$)	16.75	19.11 / 9	18.23	18.53	19.10
En→De ($k = 3$)	22.79	23.36 / 13	23.41	23.50	24.12
En→De ($k = 5$)	25.34	25.76 / 11	25.88	25.84	26.54
En→De ($k = 7$)	26.74	26.87 / 9	26.85	26.88	27.37
En→De ($k = 9$)	27.25	27.54 / 11	27.48	27.07	27.87
En→Vi ($k = 1$)	25.14	26.14 / 5	26.01	26.12	27.03
En→Vi ($k = 3$)	27.17	28.25 / 5	26.37	28.54	29.01
En→Vi ($k = 5$)	28.29	28.44 / 9	27.97	28.61	28.91
En→Vi ($k = 7$)	28.31	28.38 / 13	28.31	28.77	29.17
En→Vi ($k = 9$)	28.39	28.39 / 9	28.31	28.70	29.06
En→Zh ($k = 1$)	14.24	19.34 / 9	17.26	18.02	19.21
En→Zh ($k = 3$)	19.90	21.66 / 11	21.64	21.36	22.18
En→Zh ($k = 5$)	21.45	23.57 / 11	23.62	22.59	23.70
En→Zh ($k = 7$)	23.23	24.95 / 11	24.32	23.15	24.35
En→Zh ($k = 9$)	23.93	24.83 / 11	24.55	23.55	24.78

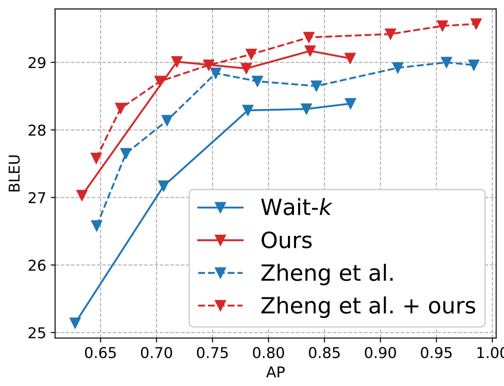
Table 1: BLEU scores on IWSLT simultaneous NMT tasks.

k	*wait-k*	*wait-k**/ best k^*	CL	MTL	Ours
1	17.07	19.83 / 9	19.41	17.59	18.63
3	22.86	23.14 / 7	22.51	22.76	23.98
5	25.52	26.09 / 7	25.51	25.66	26.77
7	27.32	27.50 / 9	26.80	26.91	27.92
9	28.05	28.05 / 9	28.20	27.82	28.67

Table 2: BLEU scores on WMT'15 En→De dataset.

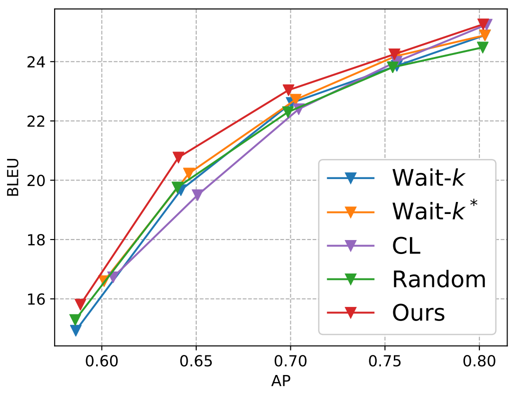


(a) Comparison of our method, WIW, WID, MILk, MMA-IL and MMA-H.

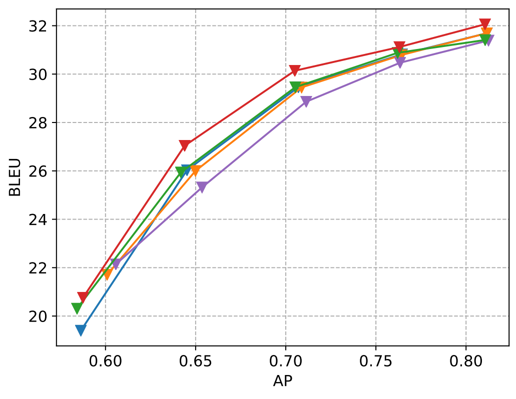


(b) Comparison and combination of our method and Zheng et al. (2020).

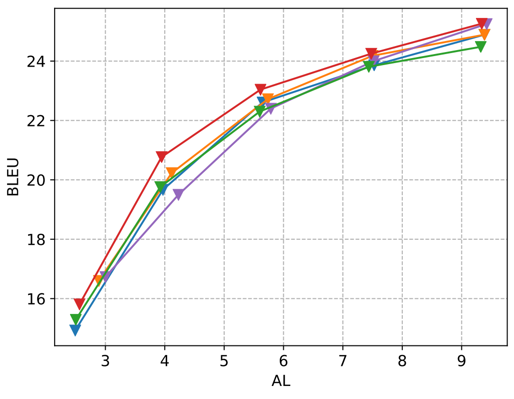
Figure 2: BLEU-AP comparison between our method and baselines on En→Vi.



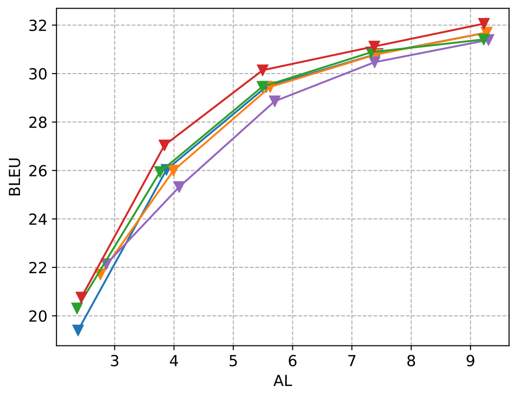
(a) BLEU-AP, WMT'14 En→De



(b) BLEU-AP, WMT'16 En→De



(c) BLEU-AL, WMT'14 En→De



(d) BLEU-AL, WMT'16 En→De

Figure 3: Translation quality against latency metrics (AP and AL) on WMT'14 and 16 English→German test sets.

important than input data information including sequence length (feature i) and data difficulty (feature ii).

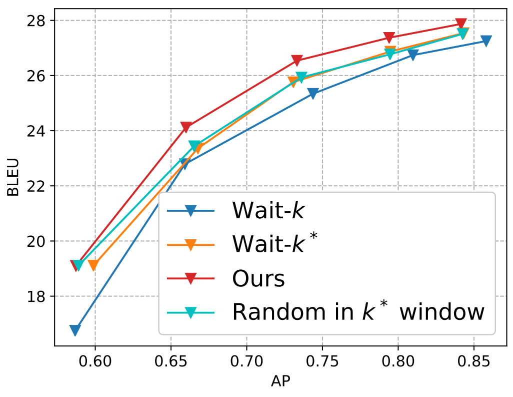
	$k = 3$	$k = 5$	$k = 7$
Ours	24.12	26.54	27.37
- (i)	23.67 (-1.87%, rank 3)	26.03 (-1.92%, rank 3)	26.92 (-1.64%, rank 4)
- (ii)	23.70 (-1.74%, rank 4)	26.04 (-1.88%, rank 4)	26.91 (-1.68%, rank 3)
- (iii)	23.57 (-2.28%, rank 1)	25.92 (-2.34%, rank 2)	26.72 (-2.37%, rank 1)
- (iv)	23.65 (-1.95%, rank 2)	25.63 (-3.43%, rank 1)	26.86 (-1.86%, rank 2)

Table 3: Ablation study for feature selection on IWSLT'14 En→De dataset.

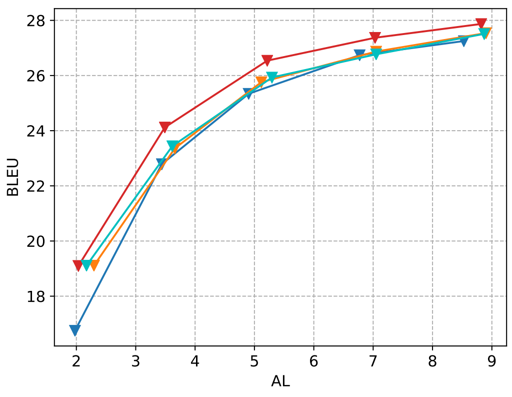
E.2. More heuristic baselines

In this section, we conduct ablation studies on two more heuristic baselines to demonstrate the effectiveness of our method.

(1) **Randomly selecting k in a window around k^***: We implement another baseline which is a combination of *wait*-k^* and MTL. After obtaining k^*, instead of sampling *wait*-m on all possible $\{1, 2, \cdots, M\}$, we sample on a smaller region around k^*. We conduct experiments on IWSLT En→De and the results are in Figure 4. We can see that this variant achieves similar results with *wait*-k^*, but is not as good as our method. This shows the importance of using an adaptive controller to guide the training.



(a) BLEU-AP, IWSLT'14 En→De



(b) BLEU-AL, IWSLT'14 En→De

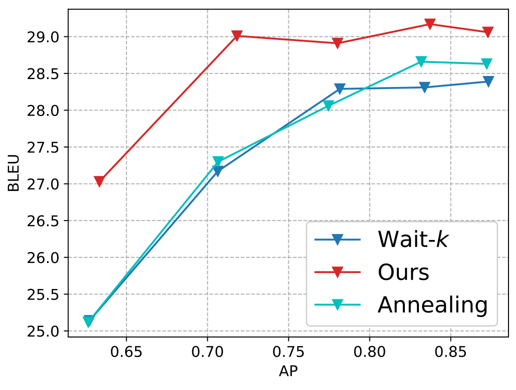
Figure 4: Randomly selecting k in a window around k^*.

(2) **A variant of CL**: Considering that the curriculum transfer learning is related to curriculum learning, we implement self-paced learning (SPL) for *wait*-k, a CL method based on loss function: within each minibatch, we remove the $\tau\%$ sentences with the largest loss, where τ is gradually decreased from 40 to 0. On IWSLT En→De, when $k = 3$, the BLEU/AL/AP for SPL are 22.87/0.66/3.47, which are worse than conventional CL (23.41/0.66/3.48).

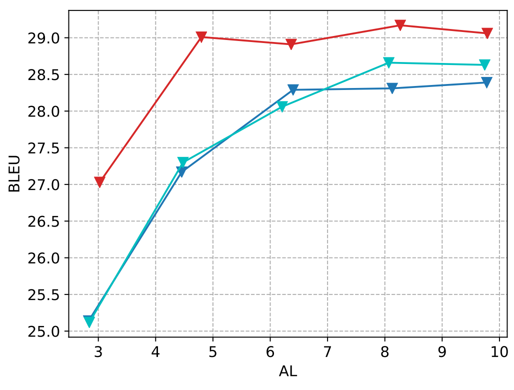
(3) **An annealing strategy**: Inspired by the fact that *wait*-k^* outperforms *wait*-k, we design a baseline where we randomly sample the waiting threshold m from a distribution $p_t(m)$ at each training step t. The distribution $p_t(m)$ linearly anneals from a uniform distribution to a distribution which prefers larger m. We expect a single annealing strategy can train reasonably good models for different inference-time k values. Suppose the minimal m value is m_{min}, the maximal m value is m_{max}. m_{min} and m_{max} are two integers and $m \in \{m_{\text{min}}, m_{\text{min}}+1, \cdots, m_{\text{max}}\}$. The total training step is denote as t_{max}. $p_t(m)$ is mathematically defined as follows:

$$\begin{aligned} p_t(m) &= (1 - \frac{t}{t_{\text{max}}}) \cdot p_{\text{init}}(m) + \frac{t}{t_{\text{max}}} \cdot p_{\text{final}}(m), \\ p_{\text{init}}(m) &= \frac{1}{m_{\text{max}} - m_{\text{min}} + 1}, \quad p_{\text{final}}(m) = \frac{m}{\sum_{i=m_{\text{min}}}^{m_{\text{max}}} i}. \end{aligned} \tag{5}$$

The results on IWSLT'15 En→Vi are shown in Figure 5, which shows this baseline brings limited improvement compared to *wait-k*. A possible reason is that this baseline cannot guarantee the best "annealing" strategy for each separate k, while our method can adaptively find the optimal strategy. Besides, as shown in Figure 5 in the main content, the learned strategies for different *wait-k* inference are pretty different.



(a) BLEU-AP, IWSLT'15 En→Vi



(b) BLEU-AL, IWSLT'15 En→Vi

Figure 5: The annealing strategy.

F. More details about stock trend forecasting

F.1. Detailed introduction of the datasets

For the Chinese A-share market dataset used in stock trend forecasting tasks, all the price and volume data are loaded from Qlib (Yang et al., 2020). In this paper, only the stocks in CSI300 are included in the experiments. CSI300 consists of the 300 largest and most liquid A-share stocks, that can reflect the overall performance of China A-share market.

F.2. Training details

For the stock trend forecasting models, we select the optimal hyper-parameters including learning rate, dropout rate, batch size and optimizer of scheduler and the sequence length L on the validation set. We repeat each experiment by using 5 random seed and report the average scores (RankIC, ICIR and MSE) to get the final result.

F.3. Supplemental results

We report the specific ICIR scores for stock trend forecasting task in Table 4. Comparing with RankIC and MSE, similar phenomenon can be observed: 1) with temporally correlated tasks, our method significantly outperforms MTL and CL on most settings; 2) using a larger temporally correlated task set does not always bring improvements; 3) our method performs better than some strong models like GAT and lightGBM (briefly, LGB).

References

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`.

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

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

	Task-1	**Task-2**	**Task-3**
GRU	0.373	0.127	0.111
MLP	0.175	0.164	0.116
LGB	0.297	0.180	0.070
GAT	0.389	0.133	0.113
	MTL		
3tasks	0.467	0.179	0.083
5tasks	0.444	0.146	0.115
10tasks	0.403	0.134	0.135
	CL		
3tasks	0.397	0.198	0.112
5tasks	0.419	0.200	0.110
10tasks	0.393	0.200	0.117
	Ours		
3tasks	0.559	0.228	0.126
5tasks	0.583	0.203	0.117
10tasks	0.532	0.222	0.160

Table 4: ICIR scores. 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 and different task sets.

2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1033. URL https://www.aclweb.org/anthology/N18-1033.

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.

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.

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.

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.

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, 2020. doi: 10.18653/v1/2020.acl-main.254. URL https://www.aclweb.org/anthology/2020.acl-main.254.