Efficient k-Nearest-Neighbor Machine Translation with Dynamic Retrieval

Yan Gao[1,2*], **Zhiwei Cao**[1,2*], **Zhongjian Miao**[1,2], **Baosong Yang**[3], **Shiyu Liu**[2], **Min Zhang**[4], **Jinsong Su**[1,2†]
[1]School of Informatics, Xiamen University, China
[2]Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan, Ministry of Culture and Tourism, China
[3]Alibaba Group, China
[4]Institute of Computer Science and Technology, Soochow University, China
gaoyan@stu.xmu.edu.cn lines1@stu.xmu.edu.cn jssu@xmu.edu.cn

Abstract

To achieve non-parametric NMT domain adaptation, k-Nearest-Neighbor Machine Translation (kNN-MT) constructs an external datastore to store domain-specific translation knowledge, which derives a kNN distribution to interpolate the prediction distribution of the NMT model via a linear interpolation coefficient λ. Despite its success, kNN retrieval at each timestep leads to substantial time overhead. To address this issue, dominant studies resort to kNN-MT with adaptive retrieval (kNN-MT-AR), which dynamically estimates λ and skips kNN retrieval if λ is less than a fixed threshold. Unfortunately, kNN-MT-AR does not yield satisfactory results. In this paper, we first conduct a preliminary study to reveal two key limitations of kNN-MT-AR: 1) the optimization gap leads to inaccurate estimation of λ for determining kNN retrieval skipping, and 2) using a fixed threshold fails to accommodate the dynamic demands for kNN retrieval at different timesteps. To mitigate these limitations, we then propose kNN-MT with dynamic retrieval (kNN-MT-DR) that significantly extends vanilla kNN-MT in two aspects. Firstly, we equip kNN-MT with a MLP-based classifier for determining whether to skip kNN retrieval at each timestep. Particularly, we explore several carefully-designed scalar features to fully exert the potential of the classifier. Secondly, we propose a timestep-aware threshold adjustment method to dynamically generate the threshold, which further improves the efficiency of our model. Experimental results on the widely-used datasets demonstrate the effectiveness and generality of our model.[1]

1 Introduction

As an effective paradigm for non-parametric domain adaptation, k-Nearest-Neighbor Machine Translation (kNN-MT) (Khandelwal et al., 2020) derives from k-Nearest-Neighbor Language Model (kNN-LM) (Khandelwal et al., 2019) and has garnered much attention recently (Zheng et al., 2021; Wang et al., 2022; Cao et al., 2023; Zhu et al., 2023b). Typically, kNN-MT introduces translation knowledge stored in an external datastore to enhance the NMT model, which can conveniently achieve non-parametric domain adaptation by changing external datastores.

In kNN-MT, a datastore containing key-value pairs is first constructed with an off-the-shelf NMT model, where the key is the decoder representation and the value corresponds to its target token. During translation, the current decoder representation is used as a query to retrieve k nearest pairs from the datastore, where retrieved values are converted into a probability distribution. Finally, via a linear interpolation coefficient λ, this distribution is used to adjust the prediction distribution of the NMT model. In spite of success, retrieving at each timestep incurs substantial time overhead, which becomes considerable as the datastore expands.

To address this drawback, researchers have proposed two categories of approaches: 1) *datastore compression* that improves retrieval efficiency by reducing the size of datastores (Martins et al., 2022a; Meng et al., 2022; Wang et al., 2022; Dai et al., 2023; Zhu et al., 2023a; Deguchi et al., 2023); 2) *retrieval reduction* that skips some kNN retrieval to speed up decoding. In this regard, the most representative work is kNN-MT with adaptive retrieval (kNN-MT-AR) (Martins et al., 2022a) that skips kNN retrieval when the coefficient λ is less than a fixed threshold α. However, kNN-MT-AR does not achieve desired results as reported in (Martins et al., 2022a).

In this work, we mainly focus on the studies of *retrieval reduction*, which is compatible with the other type of studies. To this end, we first re-implement kNN-MT-AR (Martins et al., 2022a) and conduct a preliminary study to analyze its limi-

*These authors contributed equally.
†Corresponding author.
[1]Our code is available at https://github.com/DeepLearnXMU/knn-mt-dr.

Findings of the Association for Computational Linguistics ACL 2024, pages 7990–8001
August 11-16, 2024

tations. Through in-depth analyses, we show that 1) the optimization gap leads to inaccurate estimation of λ for determining kNN retrieval skipping; 2) with the increase in timesteps, the demand for kNN retrieval diminishes, which proves challenging for the fixed threshold α to handle effectively.

To overcome the above defects, we then significantly extend the vanilla kNN-MT into kNN-MT with dynamic retrieval (kNN-MT-DR), which accelerates the model decoding in two aspects. Concretely, instead of relying on the interpolation coefficient λ, we introduce a MLP-based classifier to explicitly determine whether to skip kNN retrieval as a binary classification task. Particularly, instead of using the decoder representation as the input of the classifier, we explore several carefully-designed scalar features to fully exert the potential of the classifier. Besides, we propose a timestep-aware threshold adjustment method to dynamically generate the threshold, so as to further improve the efficiency of our model.

To summarize, main contributions of our work include the following four aspects:

- Through in-depth analyses, we conclude two defects of kNN-MT-AR: the optimization gap leads to inaccurate estimation of λ for kNN retrieval skipping, and a fixed threshold is unable to effectively handle the varying demands of kNN retrieval at different timesteps.
- We propose to equip kNN-MT with an explicit classifier to determine whether to skip kNN retrieval, where carefully-designed features enable our model to achieve a better balance between model acceleration and performance.
- We propose a timestep-aware threshold adjustment method to further improve the efficiency of our model.
- Empirical evaluations on the multi-domain datasets validate the effectiveness of our model, as well as its compatibility with datastore compression methods.

2 Related Work

Datastore Compression. In this aspect, the size of the datastore for kNN retrieval is decreased to make retrieval efficient. For example, Martins et al. (2022a) compress the datastore by greedily merging neighboring pairs that share the same values, and applying PCA algorithm (Wold et al., 1987) to reduce the dimension of stored keys. Meanwhile, Zhu et al. (2023a) prune the datastore based on the concept of local correctness, while Wang et al. (2022) presents a cluster-based compact network to condense the dimension of stored keys, coupled with a cluster-based pruning strategy to discard redundant pairs. Additionally, some studies opt for dynamically adopting more compact datastores. For instance, for each token in the input sentence, Meng et al. (2022) identify the relevant parallel sentences that contain this token and then collect corresponding word-aligned target tokens to construct a smaller datastore. Subsequently, Dai et al. (2023) conduct sentence-level retrieval and dynamically construct a compact datastore for each input sentence. With the same motivation, Deguchi et al. (2023) suggest retrieving target tokens from a subset of neighbor sentences related to the input sentence, where a look-up table based distance computation method is used to expedite retrieval.

Retrieval Reduction. In this regard, some kNN retrieval is reduced to decrease time overhead for retrieval. For instance, Martins et al. (2022b) adopt chunk-wise kNN retrieval rather than timestep-wise one, and Martins et al. (2022a) explore two approaches to reduce the frequency of kNN retrieval operations: 1) one introduces a caching mechanism to speed up decoding, where the cache mainly contains retrieved pairs from previous timesteps, and skip kNN retrieval if the distance between the query and any cached key is less than a predefined threshold; 2) the other proposes to conduct kNN retrieval when the interpolation coefficient λ is less than a predefined threshold α, which, however, does not achieve satisfactory results.

Our work mainly focuses on the second type of studies mentioned above. We first conduct a preliminary study to in-depth analyze two limitations of the λ-based kNN retrieval skipping. To address these limitations, we introduce a classifier to explicitly determine whether to skip kNN retrieval as a classification task. Notably, almost concurrently with our work, Shi et al. (2023) also use a classifier to speed up model decoding, sharing a similar motivation with ours. However, our work not only achieves better results, but also significantly differs from theirs in the following three aspects:

First, we explore several carefully-designed scalar features as the input for the classifier, which are crucial for achieving better performance. Second, when training the classifier, we adopt more reasonable criteria to construct training samples.

To be specific, in addition to skipping retrieval when the target token ranks the 1st position in the NMT prediction distribution, we believe that the model should also skip when the target token can not be obtained through kNN retrieval. Finally, based on the observation that the demand for kNN retrieval diminishes as timesteps increase, we propose a timestep-aware threshold method to further improve the efficiency of our model.

3 Preliminary Study

3.1 Background

Typically, given an off-the-shelf NMT model f_θ, a vanilla kNN-MT model is constructed through the following two stages:

Datastore Construction. At this stage, all parallel sentence pairs in the training corpus $\mathcal{C}$=$\{(\boldsymbol{x}, \boldsymbol{y})\}$ are first fed into the NMT model f_θ in a teacher-forcing manner (Williams and Zipser, 1989). At each timestep *t*, the decoder representation h_t and its corresponding target token y_t are collected to form a key-value pair, which is then added to the key-value datastore $\mathcal{D}$=$\{(h_t, y_t) \mid \forall y_t \in \boldsymbol{y}, (\boldsymbol{x}, \boldsymbol{y})\}$, where h_t=$f_\theta(\boldsymbol{x}, \boldsymbol{y}_{<t})$.

Translating with Retrieved Pairs. During inference, the datastore is used to assist the NMT model. Specifically, the decoder representation $\hat{h}_t$ is used as a query to retrieve k pairs $\mathcal{N}_t$=$\{(h_i, y_i)\}_{i=1}^{k}$ from $\mathcal{D}$, which are then converted into a probability distribution over the vocabulary, abbreviated as *kNN distribution*:

$$p_{k\mathrm{NN}}(\hat{y}_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) \propto \sum_{(h_i,y_i)\in\mathcal{N}_t} \mathbb{1}_{(\hat{y}_t=y_i)} \exp(\frac{-d(h_i, \hat{h}_t)}{\tau}), \quad (1)$$

where $\mathbb{1}_{(*)}$ is an indicator function, $d(h_i, \hat{h}_t)$ measures the Euclidean distance between the query $\hat{h}_t$ and the key h_i, and τ is a predefined temperature. Finally, kNN-MT interpolates $p_{k\mathrm{NN}}$ with the prediction distribution p_{NMT} of the NMT model as a final translation distribution:

$$p(\hat{y}_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) = \lambda p_{k\mathrm{NN}} + (1-\lambda) p_{\mathrm{NMT}}, \quad (2)$$

where λ denotes a predefined interpolation coefficient tuned on the validation set.

kNN-MT with Adaptive Retrieval Obviously, the retrieval of kNN-MT at each timestep incurs significant time overhead. To address this limitation, Martins et al. (2022a) follow (He et al., 2021) to explore kNN-MT with adaptive retrieval (kNN-MT-AR). Unlike the vanilla kNN-MT, they dynamically estimate the interpolation coefficient λ using a light MLP network, which takes several neural and count-based features as the input. Then, they not only interpolate the kNN and NMT prediction distributions with λ, but also skip kNN retrieval when λ is less than a fixed threshold α. During training, they minimize the cross-entropy (CE) loss over the interpolated translation distribution.

Unfortunately, extensive results on several commonly-used datasets indicate that kNN-MT-AR does not achieve satisfactory results.

α	IT	Koran	Law	Medical	Subtitles
0.25	0.27	0.14	0.04	0.12	0.50
0.50	0.50	0.54	0.26	0.43	0.60
0.75	0.51	0.59	0.40	0.42	0.60

Table 1: F1 scores of the λ-based kNN retrieval skipping of kNN-MT-AR (Martins et al., 2022a) on the test sets.

3.2 Limitations of kNN-MT-AR.

In this subsection, we conduct a preliminary study to explore the limitations of kNN-MT-AR. We strictly follow the settings of (Martins et al., 2022a) to re-implement their kNN-MT-AR, and then conduct two groups of experiments on the commonly-used multi-domain datasets released by Aharoni and Goldberg (2020).

As reported by Martins et al. (2022a), dynamically determining whether to skip kNN retrieval based on λ leads to significant performance degradation. In the first group of experiments, to further provide evidence of this conclusion, we perform decoding on the test sets in a teacher-forcing manner and analyze the F1 scores of λ-based kNN retrieval skipping. As shown in Table 1, F1 scores remain relatively low no matter which thresholds and datasets are used.

For the above results, we believe that there are two reasons leading to the inaccurate estimation of λ, which in turn makes λ unsuitable for deciding whether to skip kNN retrieval.

In addition to lacking the information of kNN distribution for λ estimation[2], we believe that **the optimization objective of minimizing the CE loss over the translation distribution may be unsuitable to train an accurate λ estimator for deter-**

mining kNN retrieval skipping. To verify this claim, we consider whether to skip kNN retrieval as a standard binary classification task and use a binary CE loss to train a classifier for λ estimation. Note that this classifier is also based on MLP and contains the same input as kNN-MT-AR. To avoid description confusion, we denote the λ trained by kNN-MT-AR and the above binary CE loss as Tran-λ and Bina-λ, respectively. Then, we calculate the average absolute value of the difference between Bina-λ and Tran-λ at all timesteps. The statistical results show that the average difference is 0.1495, and 29.12% of timesteps exhibit a difference exceeding 0.2. These findings indicate significant differences between Bina-λ and Tran-λ.

In the second group of experiments, we conduct experiments with vanilla kNN-MT on the validation sets to explore the impact of kNN retrieval during different timestep intervals. Specifically, we limit the model to only perform kNN retrieval in specific timestep intervals, where each interval starts from 0 and increases by 5 timesteps in length, and we only use instances with a translation length no less than the interval's right endpoint. From Figure 1, we observe that with the increase in timesteps, the performance gain caused by kNN retrieval gradually decreases across all datasets. This observation reveals that **the demand for kNN retrieval varies at different timesteps, which can not be handled well by the fixed threshold α in kNN-MT-AR.**

In summary, the above two defects seriously limit the practicality of kNN-MT-AR. Therefore, it is of great significance to explore more effective skipping kNN retrieval methods for kNN-MT.

4 Our Model

In this section, we significantly extend kNN-MT into kNN-MT-DR in the following two aspects.

4.1 Classifier for Determining kNN Retrieval Skipping

Unlike kNN-MT-AR leveraging the interpolation coefficient λ for determining whether to skip kNN retrieval, we directly equip kNN-MT with a binary classifier to determine whether to skip at each timestep. This classifier comprises a two-layer MLP network with ReLU activation. At timestep

[2]Due to the consideration of model efficiency, kNN-MT-AR do not exploit the kNN retrieval information to estimate λ, which has been shown to be effective in previous studies (Zheng et al., 2021; Jiang et al., 2022).

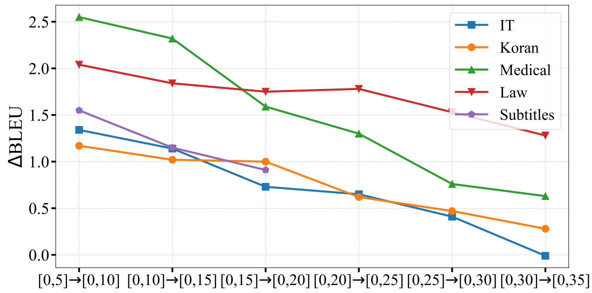

Figure 1: The changes of BLEU improvements between adjacent intervals. [0,5] means that kNN-MT only conducts retrieval when timestep ranges from 0 to 5. We only display the results for the first three BLEU improvements between adjacent intervals on the Subtitles, since the ratio of examples with length >= 25 is only about 1.35%.

t, we conduct kNN retrieval only if the prediction probability of the classifier on conducting kNN retrieval exceeds a timestep-aware threshold α_t, otherwise we will directly skip kNN retrieval. In the following, we will discuss the classifier, which involves the construction of training samples, input features, and the training objective.

Construction of Training Samples To train the classifier, one crucial step is to construct training samples. In this regard, within the exploration of kNN-LM, He et al. (2021) propose to construct training examples with two distinct labels, namely, "conducting retrieval" and "skipping retrieval", by comparing the prediction probabilities of kNN and NMT distributions on the target token y_t: when $p_{\text{kNN}}(y_t)$ is greater than $p_{\text{NMT}}(y_t)$, then kNN retrieval should be conducted, otherwise it can be skipped. However, such a criterion still leads to a lot of redundant kNN retrieval. For example, when the target token y_t has the highest probability in the NMT prediction distribution, there is no need to perform kNN retrieval, even if $p_{\text{kNN}}(y_t) \geq p_{\text{NMT}}(y_t)$. Taking the IT validation set as an example, 69.8% of timesteps satisfy $p_{\text{kNN}}(y_t) \geq p_{\text{NMT}}(y_t)$, among which 77.9% of the timesteps have y_t ranking the 1st position in the NMT prediction distribution. Based on the above analysis, we traverse the parallel sentence pairs in the validation set, and collect various information at each timestep to construct training samples according to the following criteria:

- kNN retrieval should be skipped if one of the two conditions is satisfied: 1) y_t ranks the 1st position in the NMT prediction distribution, and 2) y_t does not appear in the pairs obtained via kNN retrieval. Obviously, kNN retrieval yields no benefit in both conditions.

- kNN retrieval should be conducted if y_t is not the top-1 token in the NMT prediction distribution and it occurs in the kNN retrieval pairs. In this situation, conducting kNN distribution has the potential to improve translation.

Input Features. Unlike kNN-MT-AR, which uses the decoder representation and vectors mapped by other scalar features as the input, we consider several carefully-designed scalar features as the input for the classifier directly. By doing so, we reduce the input dimension, achieving effective training and enabling efficient inference. Here, we give detailed descriptions to these features:

- $p_{\text{NMT}}(\hat{y}_t)$: the probability of the top-1 predicted token $\hat{y}_t$ in the NMT prediction distribution. The higher the prediction confidence of the NMT model, the more likely the $\hat{y}_t$ to be the correct one. In this situation, kNN retrieval is more likely to be skipped.
- $\|\hat{h}_t\|_2$: the L_2 norm of current decoder representation. Inspired by (Liu et al., 2020), we use the vector norm of the decoder representation $\hat{h}_t$ to measure the translation difficulty at current timestep: the larger $\|\hat{h}_t\|_2$, the more difficult the translation is.
- $\max(attn)$: the maximal weight of the cross-attention in the last layer of the decoder during current decoding timestep. A large weight means that the NMT model is relatively certain about which source token to be translated. In this case, the translation difficulty is often relatively low.

Finally, these features are concatenated and normalized with batch normalization (Ioffe and Szegedy, 2015) before being the input for the classifier.

Classifier Training. To achieve efficient domain adaptation for NMT, we fix the parameters of NMT model and only update those of classifier during training. Following He et al. (2021), we select 90% of the validation set to train the classifier, and use the remaining 10% for validation. Then, according to the above criterion, we construct training samples with different labels at each timestep to train our classifier. Considering the significant imbalance between two classes of training samples[3], we adopt Focal Loss (Lin et al., 2017) to train our classifier as follows:

$$\mathcal{L}(p_c) = -\alpha_c(1 - p_c)^{\gamma} log(p_c), \tag{3}$$

where c=0/1 denotes the label of skipping/conducting kNN retrieval, p_c is the prediction probability of the classifier on the label c, α_c is a weighting factor controlling the balance between different kinds of samples, and γ is a hyper-parameter adjusting the impacts of loss functions of easy and hard samples (Lin et al., 2017).

4.2 Timestep-aware Threshold Adjustment

As analyzed in Section 3.2, the benefit of kNN retrieval diminishes with the increase in timesteps, indicating that using the fixed threshold α is not the most reasonable choice. To deal with this issue, we propose a timestep-aware threshold adjustment method to accommodate the varied demands of kNN retrieval. Formally, we heuristically define a dynamic threshold function specific to the timestep:

$$\alpha_t = \alpha_{\text{min}} + \text{clip}(\frac{t}{T}; 0, 1)^2 \times (0.5 - \alpha_{\text{min}}) \tag{4}$$

where $\text{clip}(x; a, b)$ clamp x within the range of $[a, b]$, t is the decoding timestep, α_{min} is the lower limit of threshold, and T is the average length of sentences in the validation set. Apparently, with the increase of t, α_t will gradually increase until it reaches 0.5.

5 Experiments

5.1 Setup

Datasets. We conduct experiments on the multi-domains dataset released by Aharoni and Goldberg (2020). The dataset comprises German-English parallel corpora across five domains: Koran, IT, Medical, Law, and Subtitles, the detailed statistics can be found in Appendix A. We employ Byte Pair Encoding (Sennrich et al., 2016) to split words into subwords. Finally, we use two metrics to evaluate the translation quality: SacreBLEU[4] (Post, 2018) and COMET[5] (Rei et al., 2020).

[3]Through data analysis, we find that only 16.8% of training samples require kNN retrieval in the IT validation set.

[4]https://github.com/mjpost/sacrebleu

[5]https://github.com/unbabel/COMET

Model	IT	Koran	Law	Medical	Subtitles	Average
Base NMT	38.35 / 82.74	16.26 / 72.04	45.48 / 85.66	39.99 / 83.13	29.27 / 79.76	33.87 / 80.67
Vanilla kNN-MT	45.83 / 85.19	20.37 / 72.30	61.16 / 87.46	54.22 / 84.73	31.28 / 80.13	42.57 / 81.96
kNN-MT-AR(α=0.25)	43.20 / 84.57	19.57 / 72.27	59.89 / 87.57	53.12 / **84.97**	30.46 / 80.04	41.25 / **81.88**
kNN-MT-AR(α=0.50)	41.19 / 84.05	17.23 / 72.25	58.83 / 87.50	51.22 / 84.69	29.45 / 79.87	39.58 / 81.67
kNN-MT-AR(α=0.75)	39.05 / 83.30	16.40 / 72.09	51.11 / 86.65	45.14 / 84.08	29.30 / 79.82	36.20 / 81.19
Faster kNN-MT	44.25 / 84.59	18.82 / 72.07	58.97 / 87.36	51.02 / 84.45	30.76 / 80.04	40.76 / 81.70
SK-MT$_1$	46.11 / 84.39	17.13 / 72.16	60.43 / 87.46	53.98 / 84.22	28.63 / 77.52	41.26 / 81.15
SK-MT$_2$	**46.28 / 85.41**	18.18 / 72.17	**61.55 / 87.68**	**55.42** / 84.90	28.14 / 78.28	**41.91** / 81.69
Ours	45.48 / 84.60	**20.34 / 72.40**	60.10 / 87.39	51.97 / 84.36	**31.24 / 80.14**	41.83 / 81.78

Table 2: BLEU / COMET scores of various models on the multi-domain test sets.

Model Configuration. We develop our model with kNN-BOX[6] (Zhu et al., 2023c) and use Faiss (Johnson et al., 2019) to build the datastore and search nearest neighbors. To ensure fair comparisons, we adopt the same settings as the previous study (Khandelwal et al., 2020). Concretely, we set the number of retrieved pairs to 8, the temperature τ to 100 for Koran and 10 for the other datasets, and λ to 0.7 for IT, Subtitles, 0.8 for the other datasets. We use a two-layer MLP network with ReLU activation (Agarap, 2018) to construct our classifier, of which hidden size is set to 32 because it is not sensitive in our model. Besides, we set the hyper-parameter α_{min} to 0.45 for Koran, Subtitles, 0.4 for the other datasets.[7]

Baselines. Our baselines include:

- **Base NMT** (Ng et al., 2019). Following Khandelwal et al. (2020), we use the WMT'19 German-English news translation task winner as the base NMT model.
- **Vanilla kNN-MT** (Khandelwal et al., 2020). It serves as a baseline, upon which we develop our model.
- **kNN-MT-AR** (Martins et al., 2022a). It performs retrieval only when the interpolation coefficient λ is less than a predefined threshold α. Note that it is our most important baseline. Particularly, we report the performance of kNN-MT-AR with α set to 0.25, 0.50, and 0.75, respectively.
- **Faster kNN-MT** (Shi et al., 2023). It is a concurrent work with ours, where a two-layer MLP network takes decoder representation as the input to determine whether to skip kNN retrieval at each timestep.
- **SK-MT** (Dai et al., 2023). It dynamically constructs a compact datastore by conducting sentence-level retrieval for each input sentence. Specially, we report the performance of SK-MT$_1$ with $m = 2$, $k = 1$ and SK-MT$_2$ with $m = 16$, $k = 2$.

5.2 Main Results

To comprehensively evaluate various models, we report their translation quality and decoding speed.

Translation Quality. Table 2 presents BLEU and COMET scores of various models on the multi-domain test sets. We observe that both kNN-MT-AR and Faster kNN-MT suffer from significant performance declines compared to Vanilla kNN-MT, echoing with the results reported in previous studies (Martins et al., 2022a; Shi et al., 2023). In contrast, our model exhibits the least performance degradation. Specifically, our model achieves average BLEU and COMET scores of 41.83 and 81.78 points, with only 0.74 and 0.18 points lower than those of Vanilla kNN-MT, respectively. Although SK-MT$_2$ performs better than our model, experiments in Section 5.4 find that it is not compatible with Adaptive kNN-MT, while our model significantly outperforms SK-MT$_2$ when using Adaptive kNN-MT as the base model.

Decoding Speed. Model efficiency is a crucial performance indicator for kNN-MT. As implemented in previous studies (Zheng et al., 2021; Deguchi et al., 2023), we try different batch sizes: 1, 16, 32, 64 and 128, and then report the model efficiency using "*#Tok/Sec*": the number of translation tokens generated by the model per second.

Experimental results are listed in Table 3. We have the following interesting findings: First, regardless of the batch size used, our model is more

[6] https://github.com/NJUNLP/knn-box

[7] The details of tuning α_{min} are reported in Appendix C.

Model	IT	Koran	Law	Medical	Subtitles
Batch Size = 128					
Base NMT	3270.84	3912.95	3690.85	3152.59	4004.40
Vanilla kNN-MT	2584.31	3287.24	2300.23	2363.00	478.99
kNN-MT-AR	2724.76	3069.38	2241.93	2382.52	886.16
Faster kNN-MT	2912.67	**3609.53**	2923.79	**2676.11**	999.57
SK-MT$_1$	524.65	537.06	533.52	560.14	264.30
SK-MT$_2$	385.95	408.21	423.16	428.63	236.42
Ours	**2944.38**	3522.49	**2933.76**	2605.12	**1002.13**
Batch Size = 64					
Base NMT	3150.95	3730.90	3607.41	3111.54	3377.17
Vanilla kNN-MT	2506.85	2945.54	2252.18	2329.36	445.88
kNN-MT-AR	2789.59	2678.89	2125.88	2323.60	794.04
Faster kNN-MT	2783.62	3124.68	2726.92	**2592.75**	898.26
SK-MT$_1$	518.82	525.16	524.28	547.08	258.02
SK-MT$_2$	381.87	396.00	411.91	420.05	224.41
Ours	**2798.39**	**3132.26**	**2755.02**	2575.40	**901.33**
Batch Size = 32					
Base NMT	2559.84	2933.82	2995.43	2688.93	2635.05
Vanilla kNN-MT	2001.80	2360.50	1908.76	1955.65	408.54
kNN-MT-AR	2067.55	1792.74	1925.76	1694.34	676.28
Faster kNN-MT	**2131.48**	**2432.76**	2225.68	**2047.19**	735.26
SK-MT$_1$	486.17	500.06	494.30	523.16	247.17
SK-MT$_2$	360.76	374.32	392.97	400.63	203.85
Ours	2117.94	2392.60	**2226.63**	2031.51	**737.85**
Batch Size = 16					
Base NMT	1577.03	1878.36	1959.55	1737.23	1686.02
Vanilla kNN-MT	1378.65	1429.78	1318.55	1366.35	340.96
kNN-MT-AR	1369.49	1437.82	1244.21	1323.07	506.17
Faster kNN-MT	1396.32	1451.95	1455.46	**1406.26**	538.65
SK-MT$_1$	410.57	409.76	431.25	440.31	220.65
SK-MT$_2$	318.62	340.16	354.70	355.37	176.17
Ours	**1441.66**	**1487.54**	**1472.04**	1395.21	**546.22**
Batch Size = 1					
Base NMT	159.24	168.84	173.22	171.12	159.04
Vanilla kNN-MT	136.19	139.02	142.91	138.31	42.75
kNN-MT-AR	127.23	130.35	127.93	128.09	57.98
Faster kNN-MT	139.54	**140.85**	147.18	**140.68**	58.46
SK-MT$_1$	89.76	103.97	96.42	92.52	35.26
SK-MT$_2$	84.10	97.01	89.82	85.72	32.68
Ours	**139.84**	140.18	**147.44**	139.84	**58.62**

Table 3: Decoding speed (#Tok/Sec↑) of various models using different batch sizes on the multi-domain test sets. Here, we only display the decoding speed of kNN-MT-AR(α=0.25), since kNN-MT-AR(α=0.5) and kNN-MT-AR(α=0.75) exhibit significant performance degradation, as reported in Table 2. All results are evaluated on an NVIDIA RTX A6000 GPU.

Model	BLEU
Faster kNN-MT	44.25
Ours	45.48
Our Criteria⇒Conventional Criteria	43.90
Dynamic Threshold⇒Fixed Threshold	44.28
Focal Loss⇒Weighted CE Loss	44.79
w/o $p_{\text{NMT}}(\hat{y}_t)$	44.62
w/o $\|\hat{h}_t\|_2$	45.01
w/o max(*Attn*)	45.12

Table 4: Ablation studies on the IT test set.

efficient than both Vanilla kNN-MT, kNN-MT-AR(α=0.25), SK-MT$_1$ and SK-MT$_2$.

Second, as the batch size increases, the efficiency advantage of our model becomes more apparent. On most datasets, we find that the acceleration ratios of our model with large batch sizes (64 or 128) are significantly higher than those with small batch sizes (1 or 16). Finally, with the increase of the datastore size, the efficiency advantage of our model also becomes more significant. As analysed in Appendix A, the datastore in Subtitles contains the maximum number of pairs while the datastore in Koran is the smallest. Correspondingly, our model has the most significant acceleration effect on the Subtitles dataset, while the acceleration effect on the Koran dataset is the least significant.

Based on the above experimental results, we believe that compared with baselines, ours can achieve better balance between model performance degradation and acceleration.

5.3 Ablation Studies

Following previous studies (Zheng et al., 2021; Jiang et al., 2022), we compare our model with its variants on the IT test set. As shown in Table 4, we consider the following variants:

- *Our Criteria⇒Conventional Criteria.* As mentioned in Section 4.1, we adopt new criteria to determine whether kNN retrieval in training samples can be skipped. To verify the effectiveness of our criteria, we compare our criteria with the conventional criteria as mentioned in He et al. (2021): the kNN retrieval should be conducted if $p_{\text{kNN}}(y_t) \geq p_{\text{NMT}}(y_t)$, otherwise it can be skipped. We first report the proportion changes between two labels of training samples on the IT dataset. Using the conventional criteria, the proportion of training samples labeled as skipping retrieval is about 30.2%, which is significantly smaller than the proportion 83.2% in our criteria. Obviously, more kNN retrieval can be skipped with our criteria. Second, we focus on the change of model performance. From Line 2, we observe that the conventional criteria leads to a significant performance degeneration, which strongly reveals the effectiveness of our critera.
- *Dynamic Threshold⇒Fixed Threshold.* We replace the proposed dynamic threshold α_t mentioned in Section 4.2 with the originally-used fixed threshold α=0.5 in this variant. As shown in Line 3, we observe that removing the dynamic threshold leads to a performance

Model	IT	Koran	Law	Medical	Subtitles	Average
SK-MT$_1$	46.11 / 84.39	17.13 / 72.16	60.43 / 87.46	53.98 / 84.22	28.63 / 77.52	41.26 / 81.15
SK-MT$_2$	46.28 / 85.41	18.18 / 72.17	61.55 / 87.68	55.42 / 84.90	28.14 / 78.28	41.91 / 81.69
Adaptive kNN-MT	47.26 / 85.99	20.15 / 73.22	62.68 / 88.07	56.49 / 85.25	31.49 / 80.25	43.61 / 82.56
+ kNN-MT-AR(α=0.25)	44.34 / 84.92	**20.19** / 72.40	**61.86** / 87.66	**55.46** / 84.76	30.64 / 79.92	42.50 / 81.93
+ kNN-MT-AR(α=0.50)	41.34 / 84.51	17.04 / 72.05	59.71 / 87.37	52.33 / 84.59	29.37 / 79.83	39.96 / 81.67
+ kNN-MT-AR(α=0.75)	39.22 / 83.69	16.48 / 72.06	51.28 / 86.60	45.23 / 84.08	29.30 / 79.81	36.30 / 81.25
+ Faster kNN-MT	45.38 / 85.43	19.04 / 72.98	59.95 / 87.73	53.09 / 84.91	30.63 / 80.06	41.62 / 82.22
+ Ours	**46.94 / 85.46**	20.05 / **73.26**	61.17 / **87.75**	54.58 / **84.98**	**31.35 / 80.38**	**42.82 / 82.37**

Table 5: BLEU / COMET scores of various models based on Adaptive kNN-MT.

Model	IT	Koran	Law	Medical	Subtitles
SK-MT$_1$	524.65	537.06	533.52	560.14	264.30
SK-MT$_2$	385.95	408.21	423.16	428.63	236.42
Adaptive kNN-MT	2583.92	3320.01	2292.75	2368.51	484.62
+ kNN-MT-AR(α=0.25)	2646.95	3098.34	2191.50	2235.01	873.98
+ Faster kNN-MT	2923.62	**3665.24**	**2901.53**	**2733.55**	952.27
+ Ours	**2971.77**	3569.44	2883.89	2712.45	**1075.36**

Table 6: Decoding speed (#Tok/Sec↑) of various models based on Adaptive kNN-MT. Note that we also omit the results of kNN-MT-AR(α=0.5) and kNN-MT-AR(α=0.75). Here, we set the batch size as 128.

Model	IT	Koran	Law	Medical	Subtitles	Average
PLAC	46.81 / 85.65	20.51 / 73.21	62.89 / 88.01	56.05 / 85.16	31.59 / 80.36	43.57 / 82.48
+ Ours	46.83 / 85.40	20.36 / 73.25	61.66 / 87.82	54.82 / 85.01	31.28 / 80.29	42.99 / 82.35
PCK	47.27 / 86.43	19.93 / 72.96	62.91 / 88.03	56.46 / 85.15	31.69 / 80.53	43.65 / 82.62
+ Ours	46.85 / 85.97	19.99 / 73.24	61.98 / 88.05	55.34 / 85.11	31.20 / 80.44	43.07 / 82.56

Table 7: BLEU / COMET scores of PLAC (Zhu et al., 2023a) and PCK (Wang et al., 2022), alongside these integrated with ours.

Model	IT	Koran	Law	Medical	Subtitles
PLAC	2684.36	3398.53	2433.44	2383.00	749.49
+Ours	3027.95	3596.20	3025.14	2713.74	1461.30
PCK	2873.40	3535.19	2673.76	2617.73	979.52
+Ours	3072.21	3588.76	3009.64	2720.04	1801.97

Table 8: Decoding speed (#Tok/Sec↑) of PLAC (Zhu et al., 2023a) and PCK (Wang et al., 2022), alongside these integrated with ours. Here, we set the batch size as 128.

decline, demonstrating the effectiveness of our threshold adjustment method.

- *Focal Loss⇒Weighted CE Loss.* To make a fair comparison, we follow Shi et al. (2023) to adopt a weighted CE loss, which sets γ as 0 in Equation 3. Back to Table 4, we find that this variant is inferior to our model in terms of translation quality. However, it still surpasses Faster kNN-MT with a large margin, confirming the significant advantage of our model in translation quality.
- *w/o Input Features.* To verify the benefit of our carefully-designed features, we thoroughly construct several variants, each of which discards one kind of feature to train the classifier. As shown in Lines 6-8, all variants exhibit performance drops with varying degrees. Thus, we confirm all features are useful for our classifier.

5.4 Experiments on Adaptive kNN-MT

Adaptive kNN-MT (Zheng et al., 2021) is a widely-used variant of kNN-MT and significantly outperforms Vanilla kNN-MT in terms of performance. It introduces a meta-k network, a two-layer MLP incorporating distances and counts of all kNN retrieval pairs, to dynamically estimate λ. Our model can also utilize Adaptive kNN-MT as the base models. When using Adaptive kNN-MT as the base model, we dynamically estimate λ solely for timesteps considered to conduct kNN retrieval. Additionally, we explore the performance of Adaptive kNN-MT as the base model for kNN-MT-AR. To

Model	IT	Koran	Law	Medical	Subtitles
Vanilla kNN-MT	45.72 / 467.21	19.38 / 534.79	61.22 / 456.88	54.11 / 501.02	31.62 / 515.47
kNN-MT-AR(α=0.25)	43.56 / 569.73	19.10 / 598.69	59.42 / 533.70	51.20 / 530.95	30.78 / 634.12
Faster kNN-MT	43.79 / 762.51	17.82 / **1108.25**	58.82 / **1155.10**	50.51 / **1076.66**	30.71 / 1048.35
SK-MT$_1$	45.36 / 306.53	16.24 / 236.57	60.21 / 310.59	53.78 / 346.35	26.87 / 265.72
SK-MT$_2$	**45.51** / 258.14	17.12 / 184.36	**60.62** / 277.27	**55.10** / 277.27	28.40 / 214.33
Ours	45.24 / **886.54**	**19.17** / 880.50	60.23 / 949.79	52.59 / 1040.25	**31.12** / **1078.92**

Table 9: BLEU↑ and #Tok/Sec↑ of various models on the all-domain datastore.

ensure fairness, we employ the λ of kNN-MT-AR to determine whether to skip kNN retrieval, and interpolate using the λ of Adaptive kNN-MT.

We also report the translation quality and decoding speed, as shown in the Table 5 and Table 6, respectively. Our model also demonstrate the least performance decline and achieve the most efficient decoding speed. Although Faster kNN-MT demonstrates comparable decoding speeds to ours, our model achieves superior performance.

5.5 Compatibility with Datastore Compression Methods

In this group of experiments, we choose PLAC (Zhu et al., 2023a) and PCK (Wang et al., 2022) as the basic models for our compatibility experiment, both of which are derived from Adaptive kNN-MT. Typically, PLAC prunes the datastore by eliminating pairs with high knowledge margin values, while PCK introduces a cluster-based compact network to condense the dimension of stored keys and utilizes a cluster-based pruning strategy to discard redundant pairs.

Tables 7 and 8 report the translation quality and decoding speed, respectively. We can observe that our model can further improve the efficiency of these two models, with slight drops in translation quality. Thus, we confirm that ours is also compatible with both PLAC and PCK.

5.6 All-Domains Datastore Experiment

To provide more evidences for the efficiency of our model, we follow Khandelwal et al. (2020) to conduct the experiment on the all domains datastore. We report the BLEU scores and decoding speed as shown in Table 9. Although SK-MT$_2$ significantly outperforms ours in the medical domain, it exhibits a significant slowdown in decoding speed across all domains. In contrast, our model achieves the best balance between translation quality and decoding speed.

5.7 Evaluation on Other Languages

In order to further validate the generality of our model, we adopt the same settings as the previous study (Zhu et al., 2023a) to perform experiments on Chinese-to-English translation using the Laws and Thesis domains from the UM dataset(Tian et al., 2014). As reported in Table 10, it is observable that ours achieves a more efficient decoding speed with almost no loss in performance.

Model	Laws	Thesis
Base NMT	14.48 / 5578.24	12.23 / 5985.98
Adaptive kNN-MT	31.61 / 3142.54	15.96 / 3389.67
kNN-MT-AR(α=0.25)	27.66 / 3233.05	13.54 / 3555.49
Faster kNN-MT	27.86 / **3619.14**	13.45 / 3882.05
SK-MT$_1$	27.02 / 604.16	15.18 / 589.71
SK-MT$_2$	27.21 / 547.20	15.33 / 564.37
Ours	**31.72** / 3457.92	**15.83** / **3989.80**

Table 10: BLEU↑ and #Tok/Sec↑ of various models on the UM dataset.

6 Conclusion and Future Work

In this work, we first in-depth analyze the limitations of kNN-MT-AR, and then significantly extend the vanilla kNN-MT to kNN-MT-DR in two aspects. First, we equip the model with a classifier to determine whether to skip kNN retrieval, where several carefully-designed scalar features are exploited to exert the potential of the classifier. Second, we propose a timestep-aware threshold adjustment method to further refine kNN retrieval skipping. Extensive experiments and analyses verify the effectiveness of our model.

Inspired by (Li et al., 2023), we will further improve our model by incorporating more source-side information into our classifier. Besides, we aim to generalize our model to kNN-LM (Khandelwal et al., 2019) and multilingual scenario (Stap and Monz, 2023), so as to validate its generalizability.

Limitations

As our model integrates an additional classifier, there is an associated increase in time consumption. Notably, as the size of the datastore decreases, the time overhead for kNN retrieval diminishes and classifier-related time cost becomes more apparent, which results in a less pronounced acceleration in decoding. Besides, the experiments of decoding speed are evaluated solely on a single computer, while the time overhead of kNN retrieval may differ across different hardware, yielding varied acceleration results.

Acknowledgements

The project was supported by National Natural Science Foundation of China (No. 62036004, No. 62276219), and the Public Technology Service Platform Project of Xiamen (No. 3502Z20231043). We also thank the reviewers for their insightful comments.

References

Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proc. of ACL*.

Zhiwei Cao, Baosong Yang, Huan Lin, Suhang Wu, Xiangpeng Wei, Dayiheng Liu, Jun Xie, Min Zhang, and Jinsong Su. 2023. Bridging the domain gaps in context representations for k-nearest neighbor neural machine translation. In *Proc. of ACL*.

Yuhan Dai, Zhirui Zhang, Qiuzhi Liu, Qu Cui, Weihua Li, Yichao Du, and Tong Xu. 2023. Simple and scalable nearest neighbor machine translation. In *Proc. of ICLR*.

Hiroyuki Deguchi, Taro Watanabe, Yusuke Matsui, Masao Utiyama, Hideki Tanaka, and Eiichiro Sumita. 2023. Subset retrieval nearest neighbor machine translation. In *Proc. of ACL*.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021. Efficient nearest neighbor language models. In *Proc. of EMNLP*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of ICML*.

Hui Jiang, Ziyao Lu, Fandong Meng, Chulun Zhou, Jie Zhou, Degen Huang, and Jinsong Su. 2022. Towards robust k-nearest-neighbor machine translation. In *Proc. of EMNLP*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Trans. on Big Data*.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Nearest neighbor machine translation. In *Proc. of ICLR*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. In *Proc. of ICLR*.

Xuanhong Li, Peng Li, and Po Hu. 2023. Revisiting source context in nearest neighbor machine translation. In *Proc. of EMNLP*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proc. of ICCV*.

Xuebo Liu, Houtim Lai, Derek F. Wong, and Lidia S. Chao. 2020. Norm-based curriculum learning for neural machine translation. In *Proc. of ACL*.

Pedro Martins, Zita Marinho, and André FT Martins. 2022a. Efficient machine translation domain adaptation. In *Proc. of the 1st Workshop on Semiparametric Methods in NLP: Decoupling Logic from Knowledge*.

Pedro Henrique Martins, Zita Marinho, and André FT Martins. 2022b. Chunk-based nearest neighbor machine translation. In *Proc. of EMNLP*.

Yuxian Meng, Xiaoya Li, Xiayu Zheng, Fei Wu, Xiaofei Sun, Tianwei Zhang, and Jiwei Li. 2022. Fast nearest neighbor machine translation. In *Proc. of ACL Findings*.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook FAIR's WMT19 news translation task submission. In *Proc. of MT*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proc. of MT*.

Ricardo Rei, Craig Stewart, Ana C. Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proc. of EMNLP*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL*.

Xiangyu Shi, Yunlong Liang, Jinan Xu, and Yufeng Chen. 2023. Towards faster k-nearest-neighbor machine translation. *arXiv preprint arXiv:2312.07419*.

David Stap and Christof Monz. 2023. Multilingual k-nearest-neighbor machine translation. In *Proc. of EMNLP*.

Liang Tian, Derek F Wong, Lidia S Chao, Paulo Quaresma, Francisco Oliveira, and Lu Yi. 2014. Um-corpus: A large english-chinese parallel corpus for statistical machine translation. In *Proc. of LREC*.

Dexin Wang, Kai Fan, Boxing Chen, and Deyi Xiong. 2022. Efficient cluster-based k-nearest-neighbor machine translation. In *Proc. of ACL*.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*.

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*.

Xin Zheng, Zhirui Zhang, Junliang Guo, Shujian Huang, Boxing Chen, Weihua Luo, and Jiajun Chen. 2021. Adaptive nearest neighbor machine translation. In *Proc. of ACL*.

Wenhao Zhu, Shujian Huang, Yunzhe Lv, Xin Zheng, and Jiajun Chen. 2023a. What knowledge is needed? towards explainable memory for knn-mt domain adaptation. In *Proc. of ACL Findings*.

Wenhao Zhu, Jingjing Xu, Shujian Huang, Lingpeng Kong, and Jiajun Chen. 2023b. Ink: Injecting knn knowledge in nearest neighbor machine translation. In *Proc. of ACL*.

Wenhao Zhu, Qianfeng Zhao, Yunzhe Lv, Shujian Huang, Siheng Zhao, Sizhe Liu, and Jiajun Chen. 2023c. knn-box: A unified framework for nearest neighbor generation. *arXiv preprint arXiv:2302.13574*.

A Dataset Statistics

The number of parallel sentence pairs in different datasets and the sizes of the constructed datastores are shown in Table 11.

Dataset	IT	Koran	Law	Medical	Subtitles
Train	223K	18K	467K	248K	14.46M
Valid	2K	2K	2K	2K	2K
Test	2K	2K	2K	2K	2K
Size	3.6M	0.5M	19.1M	6.9M	180.7M

Table 11: The statistics of datasets in different domains. We also list the size of the datastore, which is the number of stored pairs.

B Effect of Datastore Size

As analyzed in Section 5.2, our speed advantage becomes more significant with the increase of datastore size. To further verify this, we construct datastores of varying sizes by randomly deleting pairs from the original datastore, and employ the pruned datastores for kNN retrieval. The results of decoding speed on the Subtitles dataset are reported in Figure 2. As expected, we observe that our model consistently surpasses kNN-MT, regardless of the datastore size. Furthermore, the efficiency advantage of our model over kNN-MT becomes more evident with the increase of datastore size. These results further confirm that the pronounced speed advantage of our model as the datastore expands.

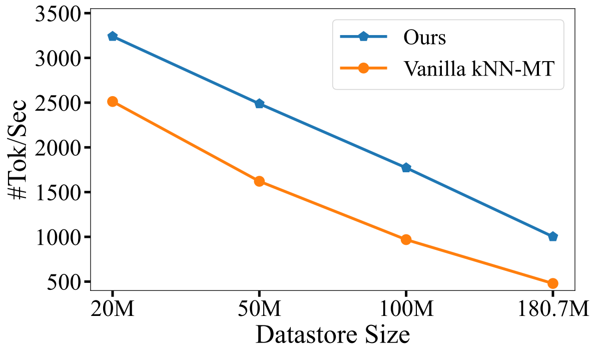

Figure 2: Decoding speed(#Tok/Sec↑) of Vanilla kNNMT and ours. Here, we set the batch size as 128.

C Hyper-Parameter Tuning

The performance and efficiency of our model is significantly impacted by the hyper-parameter α_{min}, and we tune α_{min} among the subset of $\{0.45, 0.40, 0.35\}$ on the validation set.

We report the BLEU scores and #Tok/Sec, as shown in Table 12. As α_{min} decreases, the increment in BLEU scores gradually diminishes, while the drop in decoding speed becomes more pronounced. So we set the hyper-parameter α_{min} to 0.45 for Koran, Subtitles, and 0.40 for other datasets to achieve a balance between performance and efficiency.

Note that as the validation set is utilized in training the classifier network, there exists a potential risk of overfitting when tuning α_{min}, which may result in a suboptimal selection of α_{min}.

Datasets	0.45	0.40	0.35
IT	42.03 / 2978.73	42.30 / 2940.88	42.23 / 2878.88
Koran	19.53 / 3452.23	19.50 / 3415.35	19.58 / 3408.09
Law	58.66 / 3137.26	59.20 / 3097.68	59.31 / 3001.18
Medical	51.45 / 3155.22	51.75 / 3069.02	51.86 / 2989.31
Subtitles	32.05 / 1027.21	32.13 / 898.61	32.09 / 771.34

Table 12: BLEU↑ and #Tok/Sec↑ of our model on the multi-domain validation sets with different α_{min}. Here, we set the batch size as 128.

D Compatibility with INK

INK (Zhu et al., 2023b) achieves excellent performance by performing parameter-efficient fine-

Model	IT	Koran	Law	Medical
INK	49.06 / 2842.24	22.35 / 3401.44	63.51 / 2922.82	57.41 / 2687.21
INK with Robust kNN-MT	49.97 / 1489.68	20.90 / 1839.12	65.41 / 1053.55	58.30 / 1391.53
+ Ours	49.72 / 2065.52	21.40 / 2243.05	65.17 / 1734.07	57.98 / 1788.20

Table 13: BLEU↑ and #Tok/Sec↑ of models on the multi-domain test sets. We are unable to provide the results on the Subtitles domain, since INK needs to fine-tune the base NMT model and reconstructs the datastore at each epoch, which is extremely time-consuming on the Subtitles domain.

tuning on the base NMT model using domain-specific data through knowledge distillation, and its variant equipped with Robust kNNMT (Jiang et al., 2022) achieves the state-of-the-art performance, we conduct compatibility experiments on this variant with our model and report the BLEU scores and decoding speed as shown in Table 13. We can observe that our model can improve the efficiency with only a slight drop in translation quality. Thus, we confirm that our model is compatible with INK.