SHAPE: Shifted Absolute Position Embedding for Transformers

Shun Kiyono[♠,♡] **Sosuke Kobayashi**[♡,♢] **Jun Suzuki**[♡,♠] **Kentaro Inui**[♡,♠]
[♠] RIKEN [♡] Tohoku University [♢] Preferred Networks, Inc.
shun.kiyono@riken.jp, sosk@preferred.jp,
{jun.suzuki, inui}@tohoku.ac.jp

Abstract

Position representation is crucial for building position-aware representations in Transformers. Existing position representations suffer from a lack of generalization to test data with unseen lengths or high computational cost. We investigate shifted absolute position embedding (SHAPE) to address both issues. The basic idea of SHAPE is to achieve *shift invariance*, which is a key property of recent successful position representations, by randomly shifting absolute positions during training. We demonstrate that SHAPE is empirically comparable to its counterpart while being simpler and faster[1].

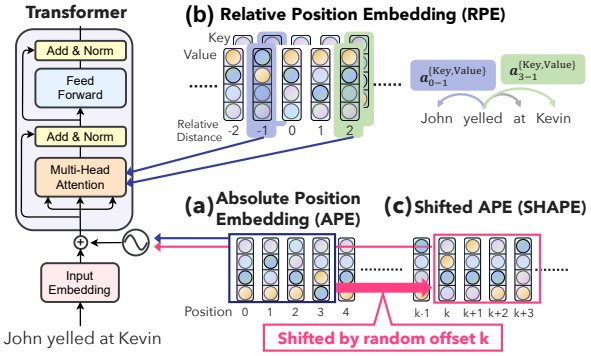


Figure 1: Overview of position representations. (a) APE and (c) SHAPE consider *absolute* positions in the *input layer*, whereas (b) RPE considers the *relative* position of a given token pair in the *self-attention mechanism*.

1 Introduction

Position representation plays a critical role in self-attention-based encoder-decoder models (Transformers) (Vaswani et al., 2017), enabling the self-attention to recognize the order of input sequences. Position representations have two categories (Dufter et al., 2021): absolute position embedding (APE) (Gehring et al., 2017; Vaswani et al., 2017) and relative position embedding (RPE) (Shaw et al., 2018). With APE, each position is represented by a unique embedding, which is added to inputs. RPE represents the position based on the relative distance between two tokens in the self-attention mechanism.

RPE outperforms APE on sequence-to-sequence tasks (Narang et al., 2021; Neishi and Yoshinaga, 2019) due to *extrapolation*, i.e., the ability to generalize to sequences that are longer than those observed during training (Newman et al., 2020). Wang et al. (2021) reported that one of the key properties contributing to RPE's superior performance is *shift invariance*[2], the property of a function to not change its output even if its input is shifted. However, unlike APE, RPE's formulation strongly depends on the self-attention mechanism. This motivated us to explore a way to incorporate the benefit of shift invariance in APE.

A promising approach to achieving shift invariance while using absolute positions is to randomly shift positions during training. A similar idea can be seen in several contexts, e.g., computer vision (Goodfellow et al., 2016) and question-answering in NLP (Geva et al., 2020). APE is no exception; a random shift should force Transformer to capture the relative positional information from absolute positions. However, the effectiveness of a random shift for incorporating shift invariance in APE is yet to be demonstrated. Thus, we formulate APE with a random shift as a variant of position representation, namely, *Shifted Absolute Position Embedding* (*SHAPE*; Figure 1c), and conduct a thorough investigation. In our experiments, we first confirm that Transformer with SHAPE learns to be shift-invariant. We then demonstrate that SHAPE achieves a performance comparable to RPE in machine translation. Finally, we reveal that Transformer equipped with shift invariance shows not only better extrapolation ability but also better *interpolation* ability, i.e., it can better predict rare words at positions observed during the training.

[1]The code is available at https://github.com/butsugiri/shape.

[2]*Shift invariance* is also known as *translation invariance*.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3309–3321
November 7–11, 2021.

2 Position Representations

Figure 1 gives an overview of the position representations compared in this paper. We denote a source sequence $\boldsymbol{X}$ as a sequence of I tokens, namely, $\boldsymbol{X} = (x_1, \ldots, x_I)$. Similarly, let $\boldsymbol{Y}$ represent a target sequence of J tokens $\boldsymbol{Y} = (y_1, \ldots, y_J)$.

2.1 Absolute Position Embedding (APE)

APE provides each position with a unique embedding (Figure 1a). Transformer with APE computes the input representation as the sum of the word embedding and the position embedding for each token $x_i \in \boldsymbol{X}$ and $y_j \in \boldsymbol{Y}$.

Sinusoidal positional encoding (Vaswani et al., 2017) is a deterministic function of the position and the *de facto* standard APE for Transformer[3]. Specifically, for the i-th token, the m-th element of position embedding $\mathrm{PE}(i, m)$ is defined as

$$\mathrm{PE}(i,m) = \begin{cases} \sin\left(\frac{i}{10000^{\frac{2m}{D}}}\right) & m \text{ is even} \\ \cos\left(\frac{i}{10000^{\frac{2m}{D}}}\right) & m \text{ is odd} \end{cases}, \quad (1)$$

where D denotes the model dimension.

2.2 Relative Position Embedding (RPE)

RPE (Shaw et al., 2018) incorporates position information by considering the relative distance between two tokens in the self-attention mechanism (Figure 1b). For example, Shaw et al. (2018) represent the relative distance between the i-th and j-th tokens with relative position embeddings $\boldsymbol{a}^{\mathrm{Key}}_{i-j}, \boldsymbol{a}^{\mathrm{Value}}_{i-j} \in \mathbb{R}^D$. These embeddings are then added to key and value representations, respectively.

RPE outperforms APE on out-of-distribution data in terms of sequence length owing to its innate *shift invariance* (Rosendahl et al., 2019; Neishi and Yoshinaga, 2019; Narang et al., 2021; Wang et al., 2021). However, the self-attention mechanism of RPE involves more computation than that of APE[4]. In addition, more importantly, RPE requires the modification of the architecture, while APE does not. Specifically, RPE strongly depends on the self-attention mechanism; thus, it is not necessarily compatible with studies that attempt to replace the self-attention with a more lightweight alternative (Kitaev et al., 2020; Choromanski et al., 2021; Tay et al., 2020).

RPE, which was originally proposed by Shaw et al. (2018), has many variants in the literature (Dai et al., 2019; Raffel et al., 2020; Huang et al., 2020; Wang et al., 2021; Wu et al., 2021). They aim to improve the empirical performance or the computational speed compared with the original RPE. However, the original RPE is still a strong method in terms of the performance. Narang et al. (2021) conducted a thorough comparison on multiple sequence-to-sequence tasks and reported that the performance of the original RPE is comparable to or sometimes better than its variants. Thus, we exclusively use the original RPE in our experiments.

2.3 Shifted Absolute Position Embedding (SHAPE)

Given the drawbacks of RPE, we investigate SHAPE (Figure 1c) as a way to equip Transformer with shift invariance without any architecture modification or computational overhead on APE. During training, SHAPE shifts every position index of APE by a random offset. This prevents the model from using absolute positions to learn the task and instead encourages the use of relative positions, which we expect to eventually lead to the learning of shift invariance.

Let k represent an offset drawn from a discrete uniform distribution $\mathcal{U}\{0, K\}$ for each sequence and for every iteration during training, where $K \in \mathbb{N}$ is the maximum shift. SHAPE only replaces $\mathrm{PE}(i, m)$ of APE in Equation 1 with

$$\mathrm{PE}(i + k, m). \quad (2)$$

We independently sample k for the source and target sequence. SHAPE can thus be incorporated into any model using APE with virtually no computational overhead since only the input is modified. Note that SHAPE is equivalent to the original APE if we set $K = 0$; in fact, we set $K = 0$ during inference. Thus, SHAPE can be seen as a natural extension to incorporate shift invariance in APE.

SHAPE can be interpreted in multiple viewpoints. For example, SHAPE can be seen as a regularizer that prevents Transformer from overfitting to the absolute position; such overfitting is undesirable not only for extrapolation (Neishi and Yoshinaga, 2019) but also for APE with length

[3] Learned position embedding (Gehring et al., 2017) is yet another variant of APE; however, we exclusively focus on sinusoidal positional encoding as its performance is comparable (Vaswani et al., 2017).

[4] Narang et al. (2021) reported that Transformer with RPE is up to 25% slower than that with APE.

constraints (Takase and Okazaki, 2019; Oka et al., 2020, 2021). In addition, SHAPE can be seen as a data augmentation method because the randomly sampled k shifts each instance into different subspaces during training.

3 Experiments

Using machine translation benchmark data, we first confirmed that Transformer trained with SHAPE learns shift invariance (Section 3.2). Then, we compared SHAPE with APE and RPE to investigate its effectiveness (Section 3.3).

3.1 Experimental Configuration

Dataset We used the WMT 2016 English-German dataset for training and followed Ott et al. (2018) for tokenization and subword segmentation (Sennrich et al., 2016). We used newstest2010-2013 and newstest2014-2016 as the validation and test sets, respectively.

Our experiments consist of the following three distinct dataset settings:

(i) VANILLA: Identical to previous studies (Vaswani et al., 2017; Ott et al., 2018).

(ii) EXTRAPOLATE: Shift-invariant models are typically evaluated in terms of extrapolation ability (Wang et al., 2021; Newman et al., 2020). We replicated the settings of Neishi and Yoshinaga (2019); the training set excludes pairs whose source or target sequence exceeds 50 subwords, while the validation and test sets are identical to VANILLA.

(iii) INTERPOLATE: We also evaluate the models from the viewpoint of *interpolation*, which we define as the ability to generate tokens whose lengths are seen during training. Specifically, we evaluate interpolation using long sequences since, first, the generation of long sequences is an important research topic in NLP (Zaheer et al., 2020; Maruf et al., 2021) and second, in datasets with long sequences, the position distribution of each token becomes increasingly sparse. In other words, tokens in the validation and test sets become unlikely to be observed in the training set at corresponding positions; we expect that shift invariance is crucial for addressing such position sparsity.

In this study, we artificially generate a long sequence by simply concatenating independent sentences in parallel corpus. Specifically, given ten neighboring sentences of VANILLA, i.e., $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{10}$ and $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_{10}$, we concatenate each sentence with a unique token $\langle sep \rangle$. We also apply the same operation to the validation and test sets.

	Original	Swapped	Performance Drop
APE	28.81	20.74	8.07
SHAPE	28.51	27.06	1.45

Table 1: BLEU score on the sub-sampled *training* data of INTERPOLATE (10,000 pairs). In **Original** and **Swapped**, the order of input sequence is $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{10}$ and $\boldsymbol{X}_2, \ldots, \boldsymbol{X}_{10}, \boldsymbol{X}_1$, respectively.

Evaluation We evaluate the performance with sacreBLEU (Post, 2018). Throughout the experiment, we apply the moses detokenizer to the system output and then compute the *detokenized* BLEU[5].

Models We adopt *transformer-base* (Vaswani et al., 2017) with APE, SHAPE, or RPE, respectively. Our implementations are based on OpenNMT-py (Klein et al., 2017). Unless otherwise stated, we use a fixed value ($K = 500$) for the maximum shift of SHAPE to demonstrate that SHAPE is robust against the choice of K. We set the relative distance limit in RPE to 16 following Shaw et al. (2018) and Neishi and Yoshinaga (2019)[6].

3.2 Experiment 1: Shift Invariance

We confirmed that SHAPE learns shift invariance by comparing APE and SHAPE trained on INTERPOLATE.

Quantitative Evaluation: BLEU on Training Data We first evaluated if the model is robust to the order of sentences in each sequence. We used the sub-sampled training data (10k pairs) of INTERPOLATE to eliminate the effect of unseen sentences; in this way, we can isolate the effect of sentence order. Given a sequence in the original order (**Original**), $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{10}$, we generated a *swapped* sequence (**Swapped**) by moving the first sentence to the end, i.e., $\boldsymbol{X}_2, \ldots, \boldsymbol{X}_{10}, \boldsymbol{X}_1$. The model then generates two sequences $\boldsymbol{Y}'_1, \ldots, \boldsymbol{Y}'_{10}$ and $\boldsymbol{Y}'_2, \ldots, \boldsymbol{Y}'_{10}, \boldsymbol{Y}'_1$. Finally, we evaluated the BLEU score of $\boldsymbol{Y}'_1$. The result is shown in Table 1. Here, SHAPE has a much smaller performance drop than APE when evaluated on different sentence ordering. This result indicates the shift invariance property of SHAPE.

Qualitative Evaluation: Similarities of Representations We also qualitatively confirmed the

[5]Details of datasets and evaluation are in Appendix A.

[6]See Appendix B for a list of hyperparameters.

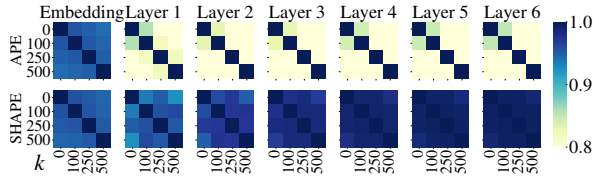


Figure 2: Cosine similarities of the encoder hidden states with different offsets $k \in \{0, 100, 250, 500\}$. Only the representation of SHAPE is invariant with k.

Dataset	Model	Valid	Test	Speed
VANILLA	APE†	23.61	30.46	x1.00
	RPE†	23.67	30.54	x0.91
	SHAPE†	23.63	30.49	x1.01
EXTRAPOLATE	APE	22.18	29.22	x1.00
	RPE	22.97	29.86	x0.91
	SHAPE	22.96	29.80	x0.99
INTERPOLATE	APE	31.40	38.23	x1.00
	RPE*	-	-	-
	SHAPE	32.50	39.09	x0.99

Table 2: BLEU scores on newstest2010-2016. **Valid** is the average of newstest2010-2013. **Test** is the average of newstest2014-2016. The scores for individual newstests are available in Appendix D. †: the values are averages of five distinct trials with five different random seeds. ∗: not available as the implementation was very slow. **Speed** is the relative speed to APE (larger is faster).

shift invariance as shown in Figure 2. The figure illustrates how the offset k changes the encoder representations of trained models APE and SHAPE. Given the two models and an input sequence $\boldsymbol{X}$, we computed the encoder hidden states of the given input sequence for each $k \in \{0, 100, 250, 500\}$. For each position i, we computed the cosine similarity (sim) of the hidden states from two offsets, i.e., $\boldsymbol{h}_i^{k_1}, \boldsymbol{h}_i^{k_2} \in \mathbb{R}^D$, and computed its average across the positions as

$$\frac{1}{I}\sum_{i=1}^{I} \mathrm{sim}(\boldsymbol{h}_i^{k_1}, \boldsymbol{h}_i^{k_2}). \quad (3)$$

As shown in Figure 2, SHAPE builds a shift-invariant representation; regardless of the offset k, the cosine similarity is almost always 1.0. Such invariance is nontrivial because the similarity of APE does not show similar characteristics[7].

3.3 Experiment 2: Performance Comparison

We compared the overall performance of position representations on the validation and test sets as

[7]Additional figures are available in Appendix C.

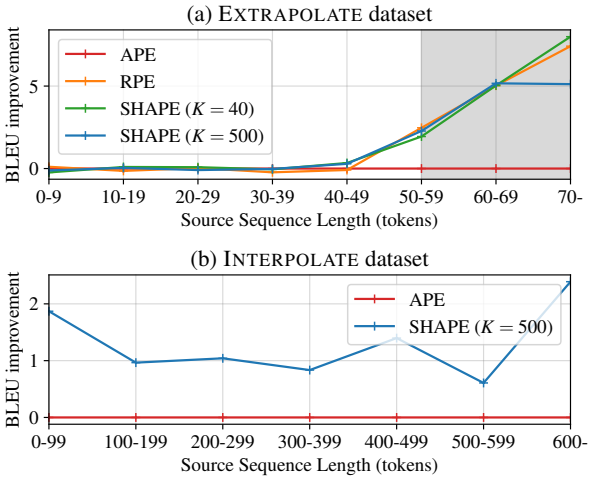


Figure 3: BLEU score improvement from APE on validation and test sets with respect to the source sequence length. The gray color means no training data.

shown in Table 2. Figure 3 shows the BLEU improvement of RPE and SHAPE from APE with respect to the source sequence length[8].

On **VANILLA**, the three models show comparable results. APE being comparable to RPE is inconsistent with the result reported by Shaw et al. (2018); we assume that this is due to a difference in implementation. In fact, Narang et al. (2021) have recently reported that improvements in Transformer often do not transfer across implementations.

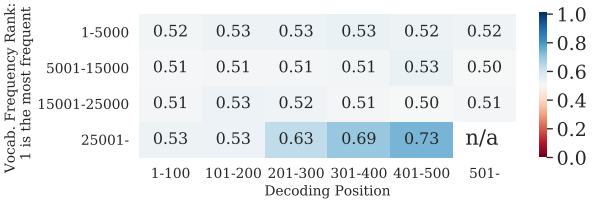
On **EXTRAPOLATE**, RPE (29.86) outperforms APE (29.22) by approximately 0.6 BLEU points on the test set; this is consistent with the result reported by Neishi and Yoshinaga (2019). Moreover, SHAPE achieves comparable test performance to RPE (29.80). According to Figure 3a, both RPE and SHAPE have improved extrapolation ability, i.e., better BLEU scores on sequences longer than those observed during training. In addition, Figure 3a shows the performance of SHAPE with the maximum shift $K = 40$ that was chosen on the basis of the BLEU score for the validation set. This model outperforms RPE, achieving BLEU scores of 23.12 and 29.86 on the validation and test sets, respectively. These results indicate that SHAPE can be a better alternative to RPE.

On **INTERPOLATE**, we were unable to train RPE because its training was prohibitively slow[9].

[8]The same graph with absolute BLEU is in Appendix D.

[9]A single gradient step of RPE took about 5 seconds, which was 20 times longer than that of APE and SHAPE. We assume that the RPE implementation available in OpenNMT-py has difficulty in dealing with long sequences.

(a) EXTRAPOLATE dataset



(b) INTERPOLATE dataset

Figure 4: Tokenwise analysis on gold references: the value in each cell represents the ratio that SHAPE assigns a higher score to a gold token than APE.

Similarly to EXTRAPOLATE, SHAPE (39.09) outperforms APE (38.23) on the test set. Figure 3b shows that SHAPE consistently outperformed APE for every sequence length. From this result, we find that the shift invariance also improves the *interpolation ability* of Transformer.

4 Analysis

This section provides a deeper analysis of how the model with translation invariance improves the performance. We hereinafter exclusively focus on APE and SHAPE because SHAPE achieves comparable performance to RPE, and we were unable to train RPE on the INTERPOLATE dataset as explained in footnote 9.

As discussed in Section 3.3, Figure 3 demonstrated that SHAPE outperformed APE in terms of BLEU score. However, BLEU evaluates two concepts simultaneously, that is, the token precision via n-gram matching and the output length via the brevity penalty (Papineni et al., 2002). Thus, the actual source of improvement remains unclear. We hereby exclusively analyzed the precision of token prediction. Specifically, we computed tokenwise scores assigned for gold references, and we then compared them across the models; given a sequence pair $(\boldsymbol{X}, \boldsymbol{Y})$ and a trained model, we computed a score (i.e., log probability) s_j for each token y_j in a teacher-forcing manner. Here, a higher score to gold token means better model performance. We used the validation set for comparison.

Figure 4 shows the ratio that SHAPE assigns a higher score to a gold token than APE, compared across for each position of the decoder.

Better extrapolation means better token precision Figure 4a shows that SHAPE outperforms APE, especially in the right part of the heat map. This area corresponds to sequences longer than those observed during training. This result indicates that better extrapolation in terms of BLEU score means better token precision.

Interpolation is particularly effective for rare tokens As shown in Figure 4b, SHAPE consistently outperforms APE and the performance gap is especially significant in the low-frequency region (bottom part). This indicates that SHAPE predicts rare words better than APE. One plausible explanation for this observation is that SHAPE carries out data augmentation in the sense that in each epoch, the same sequence pair is assigned a different position depending on the offset k. Rare words typically have sparse position distributions in training data and thus benefit from the extra position assignment during training.

5 Conclusion

We investigated SHAPE, a simple variant of APE with shift invariance. We demonstrated that SHAPE is empirically comparable to RPE yet imposes almost no computational overhead on APE. Our analysis revealed that SHAPE is effective at extrapolation to unseen lengths and interpolating rare words. SHAPE can be incorporated into the existing codebase with a few lines of code and no risk of a performance drop from APE; thus, we expect SHAPE to be used as a drop-in replacement for APE and RPE.

Acknowledgements

We thank the anonymous reviewers for their insightful comments. We thank Sho Takase for valuable discussions. We thank Ana Brassard, Benjamin Heinzerling, Reina Akama, and Yuta Matsumoto for their valuable feedback. The work of Jun Suzuki was supported by JST Moonshot R&D Grant Number JPMJMS2011 (fundamental research) and JSPS KAKENHI Grant Number 19H04162 (empirical evaluation).

References

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Łukasz Kaiser, et al. 2021. Rethinking Attention

with Performers. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 2978–2988.

Michael Denkowski and Graham Neubig. 2017. Stronger Baselines for Trustable Results in Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27.

Philipp Dufter, Martin Schmitt, and Hinrich Schütze. 2021. Position Information in Transformers: An Overview. *arXiv preprint arXiv:2102.11090*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, pages 1243–1252.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting Numerical Reasoning Skills into Language Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 946–958.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*, chapter 7.4. MIT Press. `http://www.deeplearningbook.org`.

Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. 2020. Improve Transformer Models with Better Relative Position Embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3327–3335.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proceedings of ACL 2017, System Demonstrations (ACL 2017)*, pages 67–72.

Sameen Maruf, Fahimeh Saleh, and Gholamreza Haffari. 2021. A Survey on Document-Level Neural Machine Translation: Methods and Evaluation. *ACM Computing Survey*, 54(2).

Sharan Narang, Hyung Won Chung, Yi Tay, William Fedus, Thibault Fevry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam Shazeer, Zhenzhong Lan, et al. 2021. Do Transformer Modifications Transfer Across Implementations and Applications? *arXiv preprint arXiv:2102.11972*.

Masato Neishi and Naoki Yoshinaga. 2019. On the Relation between Position Information and Sentence Length in Neural Machine Translation. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL 2019)*, pages 328–338.

Benjamin Newman, John Hewitt, Percy Liang, and Christopher D. Manning. 2020. The EOS Decision and Length Extrapolation. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP (BlackboxNLP 2020)*, pages 276–291.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook FAIR's WMT19 News Translation Task Submission. In *Proceedings of the Fourth Conference on Machine Translation (WMT 2019)*, pages 314–319.

Yui Oka, Katsuki Chousa, Katsuhito Sudoh, and Satoshi Nakamura. 2020. Incorporating Noisy Length Constraints into Transformer with Length-aware Positional Encodings. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING 2020)*, pages 3580–3585.

Yui Oka, Katsuhito Sudoh, and Satoshi Nakamura. 2021. Using Perturbed Length-aware Positional Encoding for Non-autoregressive Neural Machine Translation. *arXiv preprint arXiv:2107.13689*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation (WMT 2018)*, pages 1–9.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 311–318.

Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers (WMT 2018)*, pages 186–191.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research (JMLR)*, 21(140):1–67.

Jan Rosendahl, Viet Anh Khoa Tran, Weiyue Wang, and Hermann Ney. 2019. Analysis of Positional Encodings for Neural Machine Translation. In *Proceedings of 16th International Workshop on Spoken Language Translation 2019 (IWSLT 2019)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1715–1725.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers) (NAACL 2018)*, pages 464–468.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 2818–2826.

Sho Takase and Naoaki Okazaki. 2019. Positional encoding to control output sequence length. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3999–4004, Minneapolis, Minnesota. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient Transformers: A Survey. *arXiv preprint arXiv:2009.06732*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 5998–6008.

Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. 2021. On Position Embeddings in BERT. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

Chuhan Wu, Fangzhao Wu, and Yongfeng Huang. 2021. DA-Transformer: Distance-aware Transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2021)*, pages 2059–2068.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 17283–17297.

A Summary of Datasets

We summarized the statistics, preprocessing, and evaluation metrics of datasets used in our experiment in Table 3. The length statistics are in Figure 5.

B Hyperparameters

We present the list of hyperparameters used in our experiments in Table 4. Hyperparameters for training Transformer follow the recipe available in the official documentation page of OpenNMT-py[10].

C Similarities of Representations

In Section 3.2, we presented Figure 2 to qualitatively demonstrate that the representation of SHAPE is shift-invariant. We present ten additional figures that we created from ten additional instances in Figure 6. The characteristic of the figures are consistent with that observed in Figure 2; the representation of SHAPE is shift-invariant, whereas the representation of APE is not.

D Detailed BLEU Scores

We report the BLEU score on each of newstest2010-2016 in Table 5[11][12]. In addition, we report the performance of APE, RPE, and SHAPE with respect to the source sequence lengths in Figure 7.

E Learning Curve of Each Model

We present the learning curve of each model (APE, RPE, SHAPE) trained on different datasets (VANILLA, EXTRAPOLATE, INTERPOLATE). Figures 8 and 9 present the validation perplexity against the number of gradient steps and wall clock, respectively. From these figures, we made the following observations:

First, according to Figure 8, **the speed of convergence is similar across the models in terms of the number of gradient steps**. In other words, in our experiment (Section 3), we never compare the models whose degree of convergence is different.

Second, Figure 9 demonstrates that **RPE requires more time to complete the training than APE and SHAPE do.** As explained in Section 2.2, RPE causes the computational overhead because it needs to compute attention for relative position embeddings. The amount of time required to complete the training is presented in Table 6.

F Sanity Check of the Baseline Performance

Building a strong baseline is essential for trustable results (Denkowski and Neubig, 2017). To confirm that our baseline model (i.e., Transformer with APE) trained using OpenNMT-py (Klein et al., 2017) is strong enough, we compared its performance with that trained on Fairseq (Ott et al., 2019). Fairseq is another state-of-the-art framework used by winning teams of WMT shared task (Ng et al., 2019). For training on Fairseq, we used the official recipe available in the documentation[13]. The result is presented in Table 7. Here, the results are the average of five distinct trials with different random seeds. From the table, we can confirm that both models can achieve comparable results.

[10] https://opennmt.net/OpenNMT-py/FAQ.html#how-do-i-use-the-transformer-model

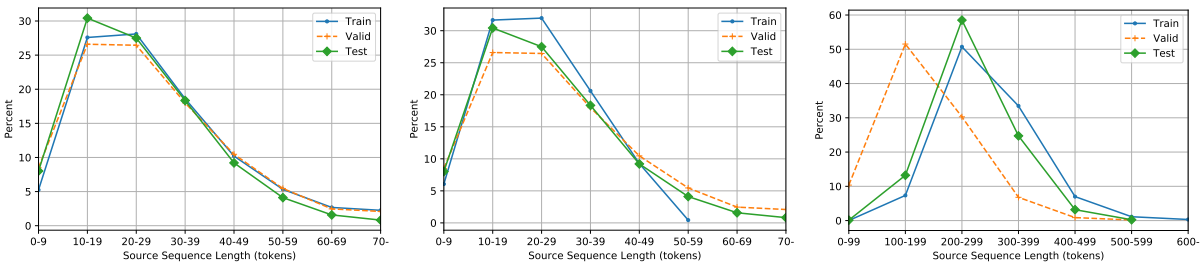
[11] SacreBLEU hash of VANILLA and EXTRAPOLATE is: BLEU+case.mixed+lang.en-de+numrefs.1+smooth.exp+test.wmt{10,11,12,13,14/full,15,16}+tok.13a+version.1.5.0.

[12] SacreBLEU hash of INTERPOLATE is BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a+version.1.5.0.

[13] https://github.com/pytorch/fairseq/tree/master/examples/scaling_nmt

Dataset Name	Training Data	# of Sent. Pairs in Training Data	Validation	Test	Evaluation Metric
VANILLA	WMT 2016 English-German	4.5M	newstest2010-2013	newsetst2014-2016	detokenized BLEU via sacreBLEU
EXTRAPOLATE	WMT 2016 English-German. We removed sequence pairs if the length of the source or target sentence exceeds 50 subwords.	3.9M	newstest2010-2013	newsetst2014-2016	detokenized BLEU via sacreBLEU
INTERPOLATE	WMT 2016 English-German. Given neighboring ten sentence of VANILLA, i.e., $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{10}$ and $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_{10}$, we concatenate each sentence with a special token $\langle sep \rangle$.	450K	newstest2010-2013. We concatenated sentences as in training data.	newstest2014-2016. We concatenated sentences as in training data.	detokenized BLEU via sacreBLEU

Table 3: Summary of statistics, preprocessing, and evaluation metric of datasets used in our experiment.

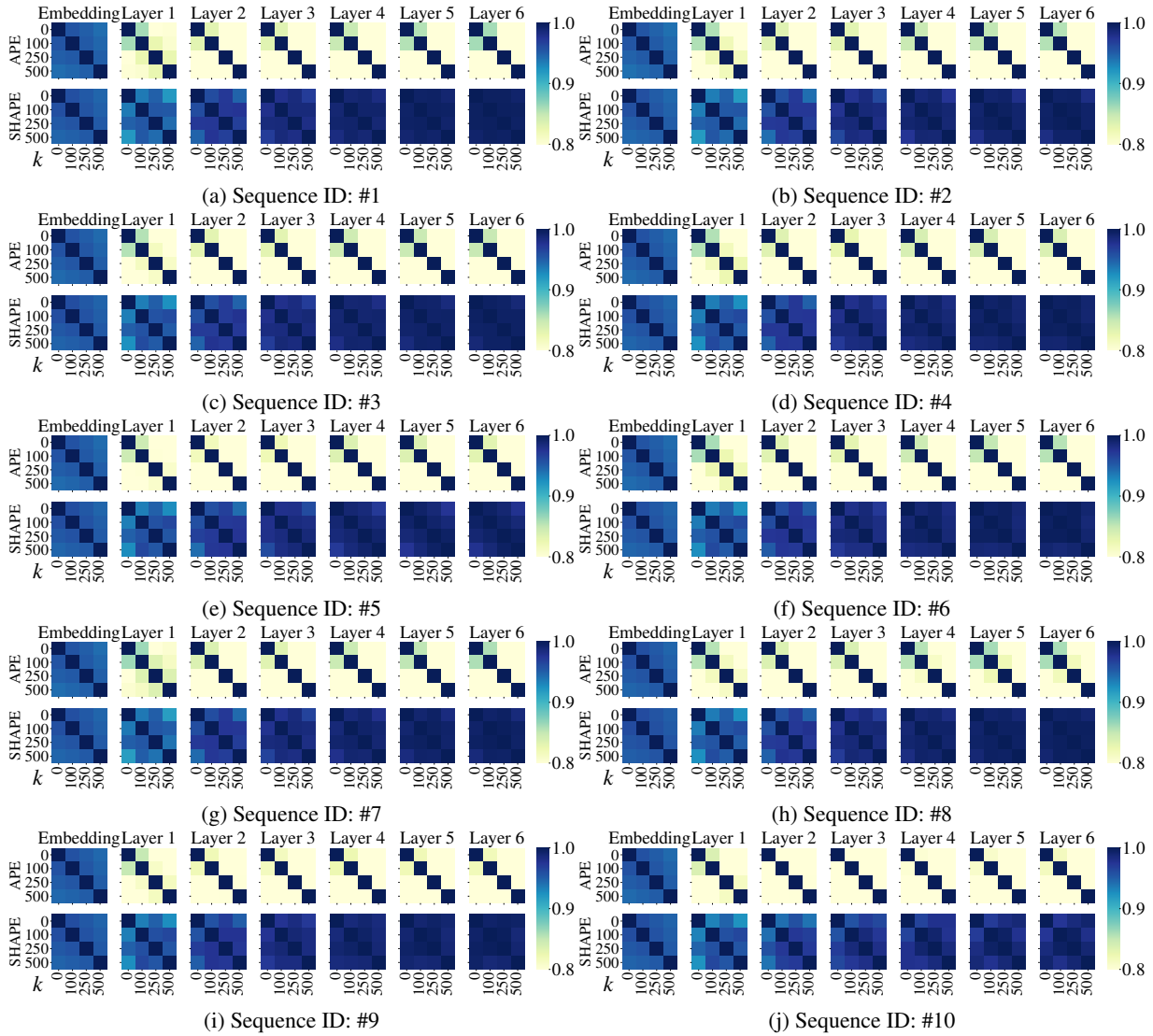


(a) VANILLA dataset (b) EXTRAPOLATE dataset (c) INTERPOLATE dataset

Figure 5: Distribution of source sequence length of each dataset.

Configurations	Selected Value
Encoder-Decoder Architecture	*transformer-base* (Vaswani et al., 2017)
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$)
Learning Rate Schedule	"Noam" scheduler described in (Vaswani et al., 2017)
Warmup Steps	8,000
Learning Rate Scaling Factor†	2
Dropout	0.1
Gradient Clipping	None
Beam Search Width	4
Label Smoothing	$\epsilon_{ls} = 0.1$ (Szegedy et al., 2016)
Mini-batch Size	112k tokens
Number of Gradient Steps	200,000
Averaging	Save checkpoint for every 5,000 steps and take an average of last 10 checkpoints
Maximum Offset K (for SHAPE)	We set $K = 500$ for the most of the experiments. We manually tuned K on validation BLEU for EXTRAPOLATE from following range: {10, 20, 30, 40, 100, 500}, and report the score of $K = 40$ in addition to $K = 500$. We used a single random seed for the tuning.
Relative Distance Limit (for RPE)	16 following (Neishi and Yoshinaga, 2019)
GPU Hardware Used	DGX-1 and DGX-2

Table 4: List of hyperparameters. †: this corresponds to "learning rate" variable defined in OpenNMT-py framework.



(a) Sequence ID: #1 (b) Sequence ID: #2

(c) Sequence ID: #3 (d) Sequence ID: #4

(e) Sequence ID: #5 (f) Sequence ID: #6

(g) Sequence ID: #7 (h) Sequence ID: #8

(i) Sequence ID: #9 (j) Sequence ID: #10

Figure 6: Cosine similarities of encoder hidden states with different offsets $k \in \{0, 100, 250, 500\}$. Only the representation of SHAPE is invariant with k.

Model	2010	2011	2012	2013	2014	2015	2016	Average	Speed
Dataset: VANILLA									
APE†	24.22	21.98	22.20	26.06	26.95	29.98	34.46	26.55	x1.00
RPE†	24.29	22.05	22.22	26.13	27.00	30.00	34.61	26.61	x0.91
SHAPE†	24.18	22.01	22.23	26.08	26.89	30.12	34.48	26.57	x1.01
Dataset: EXTRAPOLATE									
APE	22.69	20.36	20.72	24.94	26.24	28.79	32.62	25.19	x1.00
RPE	23.46	21.19	21.69	25.54	26.80	29.43	33.34	25.92	x0.91
SHAPE	23.60	21.24	21.53	25.45	26.54	29.22	33.63	25.89	x0.99
Dataset: INTERPOLATE ‡									
APE	31.41	29.71	29.79	34.69	35.36	38.00	41.32	34.33	x1.00
RPE*	-	-	-	-	-	-	-	-	-
SHAPE	32.71	30.77	30.96	35.54	35.72	39.18	42.37	35.32	x0.99

Table 5: BLEU scores on newstest2010-2016. **Average** column shows the macro average of all newstests. †: the values are averages of five distinct trials with five different random seeds. ∗: not available as the implementation was very slow. **Speed** is the relative speed to APE (larger is faster).

Model	Dataset	Hardware	Training Time (sec)	Number of Parameters
APE	VANILLA	DGX-1	97,073	61M
RPE	VANILLA	DGX-1	107,089	61M
SHAPE	VANILLA	DGX-1	96,439	61M
APE	EXTRAPOLATE	DGX-1	101,469	61M
RPE	EXTRAPOLATE	DGX-1	111,246	61M
SHAPE	EXTRAPOLATE	DGX-1	102,535	61M
APE	INTERPOLATE	DGX-2	69,148	61M
SHAPE	INTERPOLATE	DGX-2	69,529	61M

Table 6: Training time required to complete 200,000 gradient steps. RPE requires more time than APE and SHAPE do. Figure 9 illustrates the corresponding learning curve.

Model	Implementation	2010	2011	2012	2013	2014	2015	2016	Average
APE	Fairseq	24.24	22.10	22.40	26.38	27.11	29.58	34.34	26.59
APE	OpenNMT-py	24.22	21.98	22.20	26.06	26.95	29.98	34.46	26.55

Table 7: BLEU score on newstest2010-2016. We report average result of five distinct trials with different random seeds.

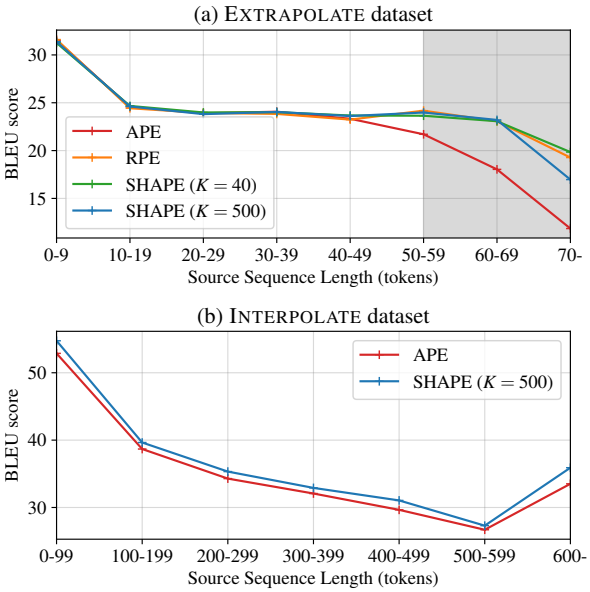
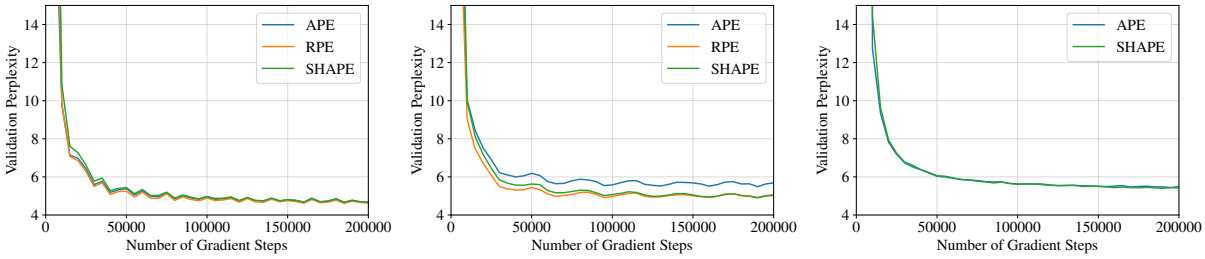
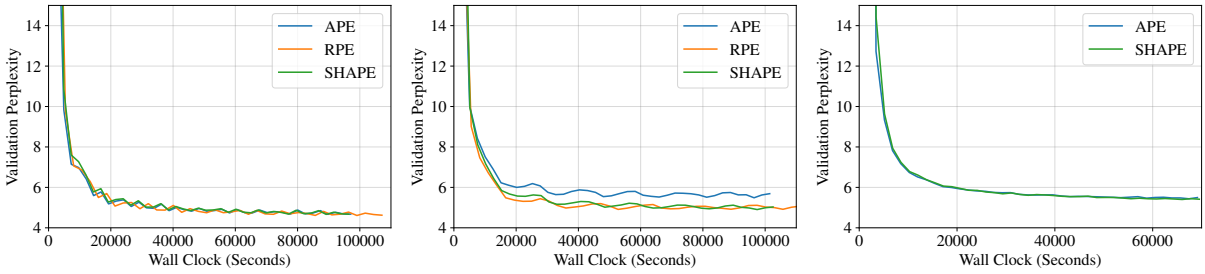


Figure 7: BLEU score on validation and test sets with respect to the source sequence length. The gray color means no training data.



(a) VANILLA dataset (b) EXTRAPOLATE dataset (c) INTERPOLATE dataset

Figure 8: Learning curves for each position representation and dataset. We compare the speed of convergence in terms of **number of gradient steps**.



(a) VANILLA dataset (b) EXTRAPOLATE dataset (c) INTERPOLATE dataset

Figure 9: Learning curves for each position representation and dataset. We compare the speed of convergence in terms of **wall clock**.