Pro-Attention: Efficient Probability Distribution Matching-Based Attention Through Feature Space Conversion

JONGSEONG BAE\textsuperscript{1}, BYUNG DO CHEON\textsuperscript{2}, AND HA YOUNG KIM\textsuperscript{2}

\textsuperscript{1}Department of Artificial Intelligence, Yonsei University, Seoul 03722, Republic of Korea
\textsuperscript{2}Graduate School of Information, Yonsei University, Seoul 03722, Republic of Korea

Corresponding author: Ha Young Kim (hayoung.kim@yonsei.ac.kr)

This work was supported in part by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) funded by the Korean Government [South Korean Ministry of Trade, Industry and Energy (MOTIE)] (Development of Smart Hybrid Envelope Systems for Zero Energy Buildings through Holistic Performance Test and Evaluation Methods and Fields Verifications) under Grant 20202020800030, and in part by the Yonsei University Research Fund under Grant 2021-22-0319.

ABSTRACT The self-attention mechanism requires a huge amount of computational cost despite its successful use in the transformer. The computational cost linearly increases in proportion to the embedding dimension size, owing to the dot product operation that calculates token similarities in vector spaces. To tackle this problem, we propose a novel efficient self-attention mechanism (Pro-attention) that computes the attention scores through distribution matching in probability space. To this end, we assume that each token has its unique probability distribution and regard each component of a token vector as a sample from the probability distribution. Then, we estimate the statistics of each token-specific probability distribution from the samples, and the token similarities are obtained by using the Kullback-Leibler Divergence. According to the time complexity analysis, the computational cost is markedly saved because the time complexity is independent of the feature dimension size. Our method produces competitive performances in machine translation and language modeling benchmarks such as IWSLT’14 De-En, WMT’14 En-De, WMT’14 En-Fr, and WikiText-103 datasets. Moreover, our model maintains the performances with considerably reduced FLOPs of the self-attention mechanism, which is up to 87\% less compared to the baseline transformer. Especially, our model improves the efficiency in a large volume of the training dataset.

INDEX TERMS Artificial intelligence, deep learning, natural language processing, model compression, self-attention mechanism, transformer.

I. INTRODUCTION

The transformer has become the predominant architecture in natural language processing (NLP) tasks while replacing traditional RNN-based architectures. Most of all, the self-attention mechanism enables parallelization so that the relationships between all token pairs in an input sequence can be computed at once [1]. The transformer architecture consists of a chain of blocks, and each block contains the self-attention module and feed-forward layers. Depending on this block-wise structure, the transformer-based networks are easily scalable with the number of blocks and the dimension size of feature representations.

The associate editor coordinating the review of this manuscript and approving it for publication was Aysegul Ucar\textsuperscript{3}.

However, the self-attention mechanism requires a huge computational cost [2]. One of the main factors is that it is quadratic to the input sequence length. To handle this problem, there have been several attempts to lighten the self-attention mechanism. For example, the attention mechanism is modified as block-wise scaling [3], long-short range attention [4], synthetic attention [5] and locality-sensitive hashing attention [2].

The cost of the self-attention also increases linearly in proportion to the feature dimension size. It is because the dot product that calculates token similarity requires more computational cost as the size of the feature dimension grows. The studies that are mentioned above showed the possibilities in lightening the self-attention, but do not handle this problem. Moreover, the embedding dimension size of the
transformer-based models such as T-5 [6], GPT-3 [7], and BiBERT [8] has increased, and the problem has been more crucial. Therefore, we are motivated to analyze this problem.

Hence, we propose a novel self-attention mechanism, Pro-attention, whose method to calculates token similarity is free from the feature dimension size. For this aim, we assume that each token in a sequence has its unique probability distribution, and each component of a token representation is regarded as a sample from the token-specific probability distribution. The statistics of each token are estimated from the samples, and the Kullback-Leibler (KL) divergence [9] is used to compute attention scores. It can be interpreted as the distribution matching process by converting the feature embedding to probability distribution.

Pro-attention just requires the statistics of the samples to compute attention scores. Once the statistics are obtained, the fixed number of operations are needed for calculating attention scores regardless of the feature dimension size. Hence, the computational cost of Pro-attention is markedly saved especially when the dimension size of the input embedding is large. This efficiency of Pro-attention increases as the embedding dimension size grows.

We evaluate our method in machine translation and language modeling tasks. Our experiment shows that our model with Pro-attention achieves competitive performance compared to the baseline transformer whereas it saves 50% to 87% of FLOPs. Our experiments also demonstrate that our model still maintains its competitive performance for sufficiently large datasets without a significant increase in the computational cost and parameters.

Our main contributions are as follows:

- We provide a novel paradigm (Pro-attention) of the self-attention mechanism. It converts the feature vector space to the probability space, and computes similarities by the probability distribution matching.
- Pro-attention has the time complexity of \(O(n^2)\), independent of the feature dimension size \(d\). It is \(d \times n^2\) less than \(O(d \cdot n^2)\) of the baseline transformer.
- In the experiment, Pro-attention can reduce at least 50% of the computational cost compared to the traditional self-attention whereas maintaining competitive performance in machine translation and language modeling benchmarks such as IWSLT’14 De-En, WMT’14 En-De, WMT’14 En-Fr, and WikiText-103.
- We observed that our Pro-attention can save more computational cost and parameters as the volume of the dataset grows.

II. RELATED WORK

Studies to increase the efficiency of the transformer can be categorized into three groups according to the existence of the attention mechanism or the dot product in a model.

A. MLP-BASED METHOD

First, the attention mechanism is discarded in some models. For instance, MLP-Mixer [10] consists of MLP blocks instead of attention blocks. MLP blocks carry out token and channel mixings. Consequently, MLP-Mixer achieved a competitive result without the attention mechanism.
Similarly, ResMLP [11] and gMLP [12] also consist of MLP instead of attention blocks. The necessity of an attention mechanism is also examined in those studies.

sMLPNet [13] is a follow-up model of above MLP-based models. MLP-Mixer could not reflect the locality and has a problem with overfitting due to its large number of parameters. As a result, there is an accuracy gap between MLP-Mixer and Vision Transformer (ViT) [14] when they are trained on a medium-scale dataset. To achieve better accuracy within a scheme of MLP-based method, sMLPNet adopts depth-wise convolution to transplant locality bias of convolutional networks and modified MLP to acquire global dependencies.

FNet [15] is another model which emphasizes the non-essentialness of the self-attention mechanism. In FNet, unparameterized Fourier transform replaces the self-attention sublayer. FNet achieves competitive performance compared to other transformer-based efficient models such as Longformer [16] and Sinkhorn transformer [17] but also shows a slight performance drop compared to bigger models such as the baseline transformer.

### B. SPARSE ATTENTION

Some models reduce the computational complexity by introducing sparse attention mechanism. In Longformer [16], for instance, the full self-attention matrix becomes sparse due to the use of attention patterns such as sliding window, dilated sliding window, and global attention. BigBird [18] is another model with sparse attention. BigBird introduces random windows and global attention. Longformer and BigBird reduce the quadratic dependency of the original attention mechanism to linear. However, the total computation per layer in those models is bigger than in models with the original attention when the input dimension is large. Hence, BigBird and Longformer are not completely upgraded models of the original transformer which can universally replace dense attention mechanisms but task-specific models for a long sequence length.

Similar to the above models, Routing Transformer [19] also adopts a sparse attention to achieve the complexity of 1.5 power of input sequence length. Routing Transformer selects sparsity patterns without computation of a full attention matrix. Lite Transformer [4] adopts long-short range attention. The long-short range attention separately captures sparse and diagonal patterns in the attention maps, and reduces model size and computation cost.

LogSparse Transformer [20] reduces the linear complexity of each attention layer to the logarithmic complexity to relieve the quadratic total complexity. To achieve this, in the self-attention of LogSparse Transformer, each cell only attends to its previous cells with an exponential step size. LogSparse Transformer also applies convolutional self-attention to manipulate queries and keys to catch more local context and enhance the overall performance.

Sparse Sinkhorn transformer [17] is another sparse efficient model. Tokens in the vanilla self-attention only attend to tokens within the same block but each token attends to tokens in the sorted block in Sparse Sinkhorn attention. Then, other tokens far away in the other block can be considered. It enables the model to compute quasi-global attention and improves memory efficiency.

Meanwhile, some models directly reduce the number of parameters. For example, DeLighT [3] lowers the
number of parameters by using block-wise scaling. Similarly, Informer [21] adopts ProbSparse-attention to reduce the number of queries used, and enhance memory and calculation efficiency.

C. ATTENTION WITHOUT THE DOT PRODUCT
Synthesizer [5] replaces the vanilla self-attention by synthetic attention. Dense and random synthetic attentions, which do not adopt the dot product, are proposed. Especially, query and key part of vanilla attention are replaced by randomly initialized matrices that does not depend on inputs. Then, it was shown that the synthesizer reduces the computational cost.

Self-Attentive Associative Memory model [22] employs a modified self-attention mechanism. The dot product is replaced by the outer product in the model. The outer product has the advantage of constructing higher-order relational representations and it increases efficiency.

Reformer [2] replaces dot product attention by locality-sensitive hashing attention. Data points that are near each other are to have similar hashing values in the locality-sensitive hashing. Moreover, reformer uses angular locality-sensitive hashing in which hash values are computed by direction components of vectors of data points and decreases the complexity of the transformer, which relates to the length of the input sequence.

Since the transformer’s base structure with query, key and value is maintained, our model can be compared to DeLight. Our model is also related to Informer and Synthesizer in the sense that the dot product is removed. However, none of the above models uses direct statistics to compute similarities. For instance, similarities in our model are calculated by KL-divergence using two statistics, the mean and standard deviation. Moreover, we provide a different point of view on the interpretation of input features. Although query and key manipulations are also conducted in LogSparse Transformer by convolutional self-attention, it does not provide a novel perspective on input feature space. On the other hand, regarding an input feature as a statistical sample in our research is a novel idea so our model has originality.

III. METHOD
A. PRO-ATTENTION MECHANISM
In this study, we no longer consider the query and key features in self-attention as vectors. We assume that each token of the query and key representation has its Gaussian probability distribution respectively, and a token embedding is a set of samples from the token-specific Gaussian distribution. Therefore, just two statistics of each token, the mean and standard deviation, are used for calculating an attention score rather than the complete token embedding. Our overall self-attention mechanism is in Fig. 1.

We adopt Gaussian distribution among other various probability distributions. It is because Gaussian distribution is a common probability distribution that requires minimum inductive biases compared to other probability distributions, and the statistics defined in Gaussian distributions are easy to be differentiated with respect to each samples.

To be more specific about our mechanism, similar to the vanilla self-attention, the query, key, and value are linearly projected from the input feature respectively. However, the query and key are semantically different from the traditional one. Each component of a token representation in vector space is considered as a sample from the token-specific Gaussian distribution. Therefore, the linear projections of the query and key features can be considered as a kind of samplers, and the dimension size of the query and key is interpreted as the samples size. By using these samples, the mean and standard deviation are computed to estimate a token-specific Gaussian distribution.

The process of computation of the query feature with the statistics is illustrated in Fig. 2. First, a word token is embedded to a query feature of $d_m$ dimension. Next, it is linearly transformed to a projected feature of $d_k$ dimension. Then, each element of the projected feature is considered as a sample from token-specific Gaussian distribution by our assumption. Statistical calculations of the mean and variance of samples are followed. Finally, we have query statistics which will be used to compute KL-divergence for the similarity. This process is equally applied to the computation of the key feature.

Fig. 3 presents the concept of feature space conversion. It shows that how are the two points interpreted differently as those are converted to probability space from vector space. From the geometric point of view, the query and key embeddings no longer points out any location in vector spaces. A point in a vector space determines a distribution in a probability space based on our assumption. Therefore, there is a need for the method to compute the similarity between two probability distributions.

Thus, we adopted the KL-divergence to compute the similarities of token pairs in the self-attention mechanism. KL-divergence quantifies the difference between two probability distributions [23]. Here, the KL-divergence compares all pairs of the $i$-th query token probability distribution ($Q_i$) and the $j$-th key token probability distribution ($K_j$). Then, KL-divergence of $Q_i$ from $K_j$, $D_{KL}(Q_i||K_j)$, can be represented as (1).

$$D_{KL}(Q_i||K_j) = H(Q_i, K_j) - H(Q_i) = \left(-\int q_i(x) \log k_j(x) dx \right)$$
where \( H \) is an entropy of given probability distribution, and \( q_i(x) \) and \( k_j(x) \) is the probability density function of \( Q_i \) and \( K_j \), respectively. We assumed that all query and key token embeddings follow their Gaussian distributions. Thus, only the mean and standard deviation of each token are used to compute the KL-divergence. Then, we derive (2) with these two statistics [24].

\[
D_{KL}(Q_i||K_j) = -\int q_i(x) \log k_j(x) dx + \int q_i(x) \log q_i(x) dx
\]

\[
= \frac{1}{2} \log(2\pi \sigma_{Q_i}^2) \sigma_{K_j}^2 + (\mu_{Q_i} - \mu_{K_j})^2
\]

\[
- \frac{1}{2} \left( 1 + \log(2\pi \sigma_{Q_i}^2) \right)
\]

\[
= \log \frac{\sigma_{K_j}}{\sigma_{Q_i}} + \frac{\sigma_{Q_i}^2 + (\mu_{Q_i} - \mu_{K_j})^2}{2\sigma_{K_j}^2} - \frac{1}{2}
\]

(1)

where \( \mu_{Q_i}, \sigma_{Q_i}, \mu_{K_j} \) and \( \sigma_{K_j} \) are the mean and standard deviation of \( i \)-th query and \( j \)-th key token representations.

The main reason why we adopted KL-divergence is as follows. KL-divergence is not commutative. That is, \( D_{KL}(Q_i||K_j) \neq D_{KL}(K_j||Q_i) \) in general. It can be a problem because the operation of measuring distance should be commutative. Hence, other commutative operations such as Jensen-Shannon divergence [25] can be used. However, we experimentally identified that the non-commutativity of KL-divergence hardly affects the model performance. Moreover, KL-divergence has the advantage of cost-efficiency over other statistical divergences, as far as we know.

We change the sign of the KL-divergence so it returns a bigger value for more similar input pairs as the dot product does. The changed sign of KL-divergence does not affect the result because the softmax function computes the output value from the relative distribution of input values rather than size differences between input values. Consequently, we calculate the negative KL-divergence between all pairs of query and key token probability distributions in our self-attention mechanism.

As shown in Fig. 1, a constant scaling term is applied in our mechanism. A scaling term in the attention mechanism plays an important role because it controls the distribution of the output values of the softmax function. In the traditional self-attention mechanism, the output of the dot product is divided by the square root of the feature dimension size \( d \) because the range of its expected value increases with the feature dimension size. Hence, the scaling term varies with \( d \). Whereas a fixed scaling term is used in our mechanism. It is because the probability distributions of tokens hardly vary with the sufficient feature dimension size due to central limit theorem [26] if certain amount of the samples are satisfied.

Meanwhile, the dot product and KL-divergence have different codomain. The set of all real numbers is the codomain of the operation of the dot product whereas KL-divergence only provides positive real numbers in a short-range. Therefore, to match the scale of final outputs of the negative KL-divergence to that of the dot product, the scaling term is set to a large number that is greater than 1. Finding the optimal value of the constant scaling term will be discussed in the experiment section with quantitative analyses.

In summary, our Pro-attention linearly project the query and key feature to convert them to the probability space. Each component of the projected feature means a sample from the token-specific Gaussian distribution, then the mean and standard deviation of this feature are computed. Obtained statistics are used to calculate the token similarities through the negative KL-divergence.

In this way, the computational cost can be significantly reduced with large feature dimension size, because our Pro-attention requires a fixed number of statistics of the query and key representations consistently regardless of its dimension size. Although there is an additional step of calculating the statistics, it does not burden computational resources because the cost of calculating the statistics is just linear to the input sequence length.

### B. ANALYSIS OF TIME COMPLEXITY

A comparison of the time complexity with other efficient self-attention mechanisms in terms of Big-\( O \) notation is presented in Table 1. The number of operations of Pro-attention is \( 10n^2 + 20nd \), whose first term is for calculating attention scores and the second term is for calculating the statistics of query and key features. Thus, the time complexity of ours is \( O(n^2) \). Here, \( d \) is excluded from the Big-\( O \) complexity, different from other models that have the \( d \) term. It is because the number of statistics for the calculation of the similarity score is independent of \( d \) so only ten fixed operations are needed for each calculation of the negative KL-divergence.

Compared to the baseline transformer and Synthesizer [5], which have the time complexity of \( O(d \cdot n^2) \), Pro-attention takes \( 1/d \) times of the computational cost of those. Compared to the Reformer [2] or the Informer [21], whose time complexity is \( O(d \cdot n \log n) \), Pro-attention takes lower computational cost only if \( n \) and \( d \) have certain values. In strict asymptotic analysis, the time complexity of Reformer and Informer is smaller than that of Pro-attention because \( n^2/(n \log n) \) diverges when \( n \) goes to infinity.

| Model            | Time complexity |
|------------------|-----------------|
| Transformer      | \( O(d \cdot n^2) \) |
| Synthesizer      | \( O(d \cdot n^2) \) |
| Reformer         | \( O(d \cdot n \log n) \) |
| Informer         | \( O(d \cdot n \log n) \) |
| Pro-attention (Ours) | \( O(n^2) \) |

**TABLE 1. Comparison of the time complexity of the self-attention mechanisms [27].** \( n \) is the input sequence length and \( d \) denotes the dimension size of the feature.
TABLE 2. Comparison of performances and FLOPs in machine translation task. † indicate that performances and number of parameters of the model are reported in the original paper. FLOPs are computed in a encoder self-attention module on a batch of each test dataset in person. Encoder self-attention module contains the process from the query, key, value linear projection to the output linear projection without masking process.

| Models               | BLEU | FLOPs | # Params |
|----------------------|------|-------|---------|
| Transformer          | 34.4 | 3.18 B | 49 M    |
| Transformer (Our imp.) | 34.4 | 3.18 B | 65 M    |
| DeLightT † [3]       | 35.3 | 1.55 B | 30 M    |
| Reformer † [2]       | -    | 2.38 B | -       |
| Lite Transformer † [4] | 33.6 | 0.95 B | 8.9 M   |
| Ours (dₐ=8)          | 31.9 | 0.40 B | 43 M    |
| Ours (dₐ=32, Base)   | 34.1 | 1.60 B | 46 M    |
| Ours (dₐ=128)        | 33.9 | 6.35 B | 55 M    |

TABLE 3. Comparison of performances and FLOPs in language modeling task. † indicate that performances of the model are reported in the original paper. FLOPs are computed in a decoder self-attention module on a batch of test dataset in person. Decoder self-attention module contains the process from the query, key, value linear projection.

| Models               | WIKITEXT-103 | FLOPs |
|----------------------|--------------|-------|
| Transformer † [1]    | 18.7         | 77.31 B|
| Transformer (Our imp.) | 18.7 | 77.31 B|
| DeLightT † [3]       | 24.1         | -     |
| Reformer † [2]       | -            | -     |
| Lite Transformer † [4] | 22.2 | -     |
| Ours (dₐ=32, Base)   | 21.3         | 38.66 B|

However, \( d \cdot \log n \) is larger than \( n^2 \) where \( n \) and \( d \) have values in practice. To illustrate this, considering \( d \) of 192 and \( n \) of 512 in BiBERT that is one of the state-of-the-art models in machine translation task, over a half of the computational cost is saved compared to the Reformer or the Informer. As \( d \) grows, Pro-attention is far more efficient.

IV. EXPERIMENT

A. DATASETS

We evaluate our method on three machine translation benchmarks and one language modeling benchmark. We followed the setup of [3]. The Byte-Pair Encoding (BPE) [28] is applied in common for the tokenization.

- **IWSLT’14 German-English (De-En)** [29] is a comparatively small machine translation dataset consisting of 160K/7K/7K sentence pairs for training, validation, and test respectively. The dataset uses a shared vocabulary of 10K BPE tokens.
- **WMT’14 English-German (En-De)** [30] dataset consists of 4.5 million sentence pairs, which are split into 3.9M/39K/3K for training, validation, and test. The dataset uses a shared vocabulary of 44K BPE tokens.
- **WMT’14 English-French (En-Fr)** [30] is a large machine translation benchmark that consists of 36M/27K/3K sentence pairs for training, validation, and test. Same with WMT’14 En-De, a joint BPE vocabulary size of 44K is used.
- **WikiText-103** [31] is a language modeling dataset that consists of 103M/217K/245K tokens for training, validation, and test. The dataset has a word-level vocabulary of about 260K tokens.

B. EXPERIMENTAL SETUP

In case of machine translation task, we followed the setup of [1]. For IWSLT’14, a dropout rate of 0.3 is applied and a batch size of 4K max tokens is used. We trained a model on a single NVIDIA RTX A5000 GPU for 50K iterations. For WMT’14 En-De, and WMT’14 En-Fr, 0.1 is used for the dropout rate, and 8 NVIDIA RTX A5000 GPUs are used for 100K training iterations. We replaced all the self-attention modules of the transformer encoder and decoder layers with our Pro-attention modules. For language modeling task, we followed the setup of [32]. Same with machine translation task, we replaced all the self-attention modules of the transformer decoder layers with our Pro-attention modules. The dimension size of the query and key embeddings referred as the sample size \( d_s \) is set of 32, and the scaling constant \( c \) is set of 32 in base model.

C. RESULTS

1) MACHINE TRANSLATION

The overall performance of our model is comparable with that of the baseline transformer as summarized in Table 2. Our base model consistently achieves competitive performances in each dataset. The performance in the smaller sample size setting drops on IWSLT’14 De-En dataset, but it maintains a comparable level on bigger datasets. In the largest sample size setting, the higher performance is reported on WMT’14 En-Fr dataset. Our base model is the most competitive in terms of the general efficiency of the mechanism.

To demonstrate the high efficiency of our Pro-attention mechanism, we also provide the comparison of the FLOPs for a self-attention module of our mechanism with those of other efficient models including the baseline transformer. We calculated the FLOPs of the self-attention module of an encoder on a batch of each test dataset. Our base model with Pro-attention spends 50% of the computational resources that the vanilla self-attention module requires. It is almost the same as that of DeLightT which is less than those of other efficient models in the table. In addition, our small model requires the least FLOPs which is only 13% of that of the vanilla transformer.
J. Bae et al.: Pro-Attention: Efficient Probability Distribution Matching-Based Attention

FIGURE 4. Comparison of the visualizations of the attention weight matrices. (a) is the results of traditional self-attention mechanism and (b) is those of our mechanism. Two rows from the top are the results from one sequence example, and two rows from the bottom are those of the other sequence example. The scale of the matrices are 0 to 1 and the higher brightness is related to the higher attention score.

TABLE 4. Result with various sample size setting. For WMT’14 En-De and En-Fr, training batch size of 32K setting is applied to this result. Tendency of the performance change in various sample size setting is irregular, and larger sample size does not guarantee better performance. However, as the volume of the dataset grows, the gap of the performance between various sample settings decreases. In other words, when the dataset is large enough, our method can obtain cost-efficiency by reducing sample size whereas maintaining the performance.

| Sample size $d_s$ | IWSLT 14 De-En | WMT’14 En-De | WMT’14 En-Fr |
|-------------------|----------------|--------------|--------------|
| 8                 | 31.85          | 25.33        | 38.33        |
| 16                | 33.86          | 25.14        | 38.45        |
| 32                | 34.10          | 25.62        | 38.57        |
| 64                | 34.11          | 25.58        | 38.38        |
| 128               | 33.83          | 25.93        | 38.56        |

Standard deviation 0.86 0.27 0.09

2) LANGUAGE MODELING
Table 3 shows the result of performances and FLOPs on WikiText-103 dataset. Our base model shows better performance compared to other efficient models. Although it is less than that of the baseline transformer, our base model has the advantage of efficiency. Similar to the result in the machine translation task, FLOPs of our base model is half of that of the vanilla transformer. FLOPs are calculated in the same way as computation in the machine translation task.

3) ABLATION STUDY OF THE SAMPLE SIZE
We conducted ablation studies on machine translation datasets. First, the performances according to the sample size of the query and key are reported in Table 4. In general, it is believed that increasing the dimension size of features enhances the model performance. However, our results show that larger sample size is not compulsory for better performance. We found that the sample size of 32 gives the best performance on average in three benchmark datasets. The bottom line of Table 4 represents the standard deviations of performances according to sample size. Overall, the performances are narrowly distributed. It means that there is no big difference between the performances for small and large sample sizes. The result for WMT’14 En-Fr is remarkable for the exceptionally small standard deviation. It represents that our model works properly even for a very small size sample such as a sample size of 8. We think that as the scale of the dataset gets bigger, a token probability distribution can be trained sophisticatedly with only small samples through numerous interactions with more other tokens.

4) ABLATION STUDY OF THE SCALING CONSTANT
The effect of the scaling term on the performance is investigated in Table 5. The optimal scaling constant varies with datasets. However, the bottom line of the table shows that the performance does not much vary with scaling terms. Similar to the case of the sample sizes, the bigger the training dataset is, the less the scaling constant affects the performance. It seems that a scaling term can be ignored when the training dataset is large enough.

| Scaling constant $c$ | IWSLT 14 De-En | WMT’14 En-De | WMT’14 En-Fr |
|----------------------|----------------|--------------|--------------|
| No scaling           | 33.53          | 25.06        | 38.54        |
| 8                    | 33.77          | 25.65        | 38.40        |
| 16                   | 33.71          | 25.72        | 38.21        |
| 20                   | 34.25          | 25.51        | 38.32        |
| 24                   | 33.98          | 25.73        | 38.36        |
| 32                   | 34.10          | 25.62        | 38.57        |

Standard deviation 0.24 0.23 0.12

5) SIMILARITY DISTRIBUTION COMPARISON
Fig. 4 shows examples of the visualization of the attention weight matrices of the vanilla self-attention module and our Pro-attention mechanism. We investigated the results of 6 encoder self-attention modules, and these are samples from the last encoder self-attention module on WMT’14 En-Fr.
test dataset. 8 matrices are obtained per example because the self-attention module consists of 8 heads. Overall patterns of the attention matrices are similar in both of them. Although some cases of Pro-attention look more dense compared to those of the traditional self-attention, the results generally present that Pro-attention can reflect the relationships of the tokens sufficiently.

6) AN EXAMPLE OF TOKEN-SPECIFIC PROBABILITY DISTRIBUTIONS

Fig. 5 shows an example of token-specific probability distributions and Negative KL-divergence from WMT’14 En-De test dataset. Rows and columns are the query and key tokens respectively. \( \mathcal{N} \) notation means that the token follows Gaussian probability distribution, whose mean and variance are the reported numbers. For example, the query token Not follows the Gaussian distribution whose mean and variance are -5.14 and 7.20. It shows that all tokens in a sentence follow their unique Gaussian distribution. In addition, the scaled (the scaling constant of 32) matrix of Negative KL-divergence reflects the semantic similarities well in our method.

7) PERFORMANCE COMPARISON WITH OTHER DISTANCE CALCULATION METHODS

For feasibility, we observe the performances when common distance calculation methods such as the Euclidean distance and cosine similarity are applied for the token similarity computation. It is presented in Table 6. We evaluate the performances on IWSLT’14 De-En dataset. All attention mechanism including cross-attention in the transformer architecture are simply replaced with each method. No scaling term is applied. As you can see, the Negative KL-divergence shows the highest performance over other methods. Especially, the performance significantly drops when the Euclidean distance is applied.

8) TRANSLATION EXAMPLE ANALYSIS

Fig. 6 shows translation examples of the baseline transformer and our model on WMT’14 En-De test dataset. First, (a) shows the strength of our method that precisely performs word-level translation. The bolded words ‘frühen’ and ‘Pubertät’ are matched with the words ‘early’ and ‘puberty’, and

![Figure 5](image_url)

**FIGURE 5.** An example of token-specific probability distributions and Negative KL-divergence from WMT’14 En-De test dataset.

**TABLE 6.** Performance comparison with other distance calculation methods for the token similarity computation.

| Calculation Method | BLEU |
|--------------------|------|
| - Euclidean distance | 13.78 |
| - Cosine similarity | 25.35 |
| - KL-divergence | **31.78** |
respects. As we can see, our model completely translates the phrase, however, the baseline transformer translates this to ‘Frühpubertät’ that means ‘early puberty’ in a word. Although the word ‘each’ and ‘puberty’ are individually embedded tokens, it recognizes the phrase as a word. This result demonstrates that our method successfully reflects the token-level probability distribution.

On the other hand, (b) shows the shortcoming of our method. The bolded word ‘fun’ in the source sentence is translated to the word ‘Spaß’ in the target sentence. However, our model translates it to ‘Spielspaß’, which is similar to the word ‘Spaß’ morphologically and semantically. We infer that this mistranslation is caused by the information loss that occurred in the feature space conversion process. In our method, the vector representations are converted to probability spaces by simply calculating the statistics of the features. As we assumed that all elements of a vector are regarded as the samples from a probability distribution, however, it precisely causes the information loss that the dimension information of features vanishes. Hence, the probability distributions of morphologically similar synonyms are sometimes indistinguishable. This result suggests that more advanced feature space conversion methods are needed to prevent this information loss.

V. CONCLUSION

In this study, we propose a novel efficient self-attention mechanism, Pro-attention, that calculates token similarities by matching the probability distributions of each token. Pro-attention is thought to be the cornerstone of the application of the probabilistic method to the self-attention mechanism. Pro-attention just requires $O(n^2)$ of time complexity which is $d$ times less than that of the traditional self-attention mechanism, and it is experimentally shown that the FLOPs can be reduced up to 87% compared to that of the traditional one. whereas saving computational cost, Pro-attention even shows competitive performances in three machine translation and one language modeling benchmarks. Especially, our results presents that the computational cost can be saved more as the volume of the dataset gets larger. We expect that additional follow-up studies are conducted to develop our distribution matching method. Investigating feature space conversion for more precise application and developing an architecture that is more suitable for probability distribution matching are suggested.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., vol. 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran, 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
[2] N. Kitaev, L. Kaiser, and A. Levskaya, “Reformer: The efficient transformer,” 2020, arXiv:2001.04451.
[3] S. Mehta, M. Ghazvininejad, S. Iyer, L. Zettlemoyer, and H. Hajishirzi, “Delight: Deep and light-weight transformer,” 2020, arXiv:2008.06023.
[4] Z. Wu, Z. Liu, L. Lin, Y. Liu, and S. Han, “Lite transformer with long-short range attention,” 2020, arXiv:2004.11886.
[5] Y. Tay, D. Bahri, D. Metzler, D.-C. Juan, Z. Zhao, and C. Zheng, “Synthesizer: Rethinking self-attention for transformer models,” in Proc. Int. Mach. Learn. Conf., 2021, pp. 10183–10192.
[6] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” J. Mach. Learn. Res., vol. 21, no. 140, pp. 1–67, 2020.
[7] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, and A. Askell, “Language models are few-shot learners,” in Proc. Adv. Neu. Inf. Process. Sys., vol. 33, 2020, pp. 1877–1901.
[8] H. Xu, B. Van Durme, and K. Murray, “BERT, mBERT, or BiBERT? A study on contextualized embeddings for neural machine translation,” 2021, arXiv:2109.04588.
[9] S. Kullback and R. A. Leibler, “On information and sufficiency,” Ann. Math. Statist., vol. 22, no. 1, pp. 79–86, 1951.
[10] I. Tolstikhin, “MLP-mixer: An all-MLP architecture for vision,” in Proc. 35th Conf. Neural Inf. Process. Syst., vol. 34, 2021, pp. 24261–24272.
[11] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, G. Izacard, A. Joulin, G. Synnaeve, J. Verbeek, and H. Jegou, “ResMLP: Feedforward networks for image classification with data-efficient training,” 2021, arXiv:2105.03404.
[12] H. Liu, Z. Dai, D. So, and Q. V. Le, “Pay attention to MLPs,” in Proc. Int. Conf. Neural Inf. Process. Syst., vol. 34, 2021, pp. 9204–9215.
[13] C. Tang, Y. Zhao, G. Wang, C. Luo, W. Xie, and W. Zeng, “Sparse MLP for image recognition: Is self-attention really necessary?” in Proc. AAAI Conf. Artif. Intell., vol. 36, no. 2, 2022, pp. 2344–2351.
