Do You Really Need Attention?

Yuzhao Chen [*]
CSE Department
University of California San Diego
San Diego, CA 92093
yuc103@ucsd.edu

Andre (Jianyou) Wang [†]
CSE Department
University of California San Diego
San Diego, CA 92093
jiw101@ucsd.edu

Yongce Li [‡]
Halıcıoğlu Data Science Institute
Department of Mathematics
University of California San Diego
San Diego, CA 92093
yol013@ucsd.edu

Xiaoyue Wang [§]
Halıcıoğlu Data Science Institute
Department of Mathematics
University of California San Diego
San Diego, CA 92093
xiw027@ucsd.edu

Abstract

Our technical report investigates suitable unparametrized or sparsely-parametrized substitutions for attention layers in natural language generation tasks such as machine translation. We demonstrate the mathematical formulations for self-attention in the encoder model, masked self-attention in the decoder model, and cross encoder-decoder attention. We show that the attention layer serves as a linear token-mixing mechanism across the positional dimension, and propose unparametrized structured transformations, including the 2D unparametrized Discrete Fourier Transform (DFT), as effective replacements for attention layers with minimal performance degradation on the WMT benchmark for machine translation tasks. Our proposed Transformer with DFT achieved 27.2% in BLEU score compared with Transformer with self-attention 28.3%, though our proposed model only utilizes 55 million parameters compared to the full Transformer model which has more than 80 million parameters. We also built an end-to-end distributed training pipeline for bilingual machine translation tasks and we further proposed a method for sparsely parameterizing Fourier transform. We demonstrated the potential for unparametrized or sparsely-parametrized substitutions for attention layers in natural language generation tasks.

1 Introduction

Since the advent of residual connection and the transformer model with multi-headed attention [14], fields in NLP entered into a revolutionary era where deeper and larger neural language models ranging from BERT, RoBERTa, GPT-2, GPT-3, Instruct GPT, all the way to recent chatGPT and GPT4 have flourished and revelled in their success of completing many NLP tasks with performances that are comparable to human or even superhuman. One questions remains: why does attention work? researchers have gained great understandings that the transformer model's feed-forward and residual architecture that does not rely on recurrent structures can be trained faster and deeper without

[*]Author contributed equally in this technical report
[†]Author contributed equally in this technical report
[‡]Author contributed equally in this technical report
[§]Author contributed equally in this technical report

encountering vanishing gradient problem, reaping the benefit of the plethora of Internet textual data. However, the specific formulation of the attention layer has been under constant scrutiny from the research community.

In this technical report, we investigate suitable unparametrized or sparsely-parametrized substitutions for attention layers for natural language generation tasks such as machine translation. We demonstrate the mathematical formulations for self-attention in the encoder model, the masked self-attention in the decoder model and the cross encoder-decoder attention that bridges between the encoder and decoder models. From the mathematical formulations, we show that the attention layer serves as a linear token-mixing mechanism across the positional (i.e. sequence length) dimension, which is the only part of the Transformer model that mixes the positional dimension, whereas the embedding layers, final projection layers and position-wise dense MLP layers all mix the embedding dimension. While the multi-headed attention layers assign adjustable (i.e. learnable) coefficients for its linear combination of tokens across the positional dimension, it has been shown by the recent FNet paper [5] that an unparametrized structured transformation (i.e. token-mixing mechanism) across the positional dimension achieve slightly diminished or even comparable performances on the GLUE [16] and LRA [9] natural language understanding benchmarks. Though a unparametrized transformation will assign unchangeable (i.e. unlearnable) coefficients for its linear combination of tokens across the positional dimension, it only diminishes the performances 3%-8% on natural language understanding tasks, suggesting the benefits of using dense MLP+Softmax layers to model learnable coefficients across positional dimension may be overrated, at least in some NLP tasks.

In contrast to natural language understanding tasks on the GLUE and LRA benchmarks that only require the encoder of the transformer model, we consider natural language generation tasks on the WMT benchmark (e.g. English to Italian machine translation task [1]) that require both the encoder and decoder of the transformer model. Specifically, we investigate the effects of replacing only the encoder-attention, and both the encoder-attention and masked decoder-attention with unparametrized as well as sparsely-parametrized token-mixing transformations that are fast, scalable and leave a small memory footprint. Similar to findings in the FNet paper, we report only slightly diminished performances on the English to Italian machine translation task [6] (BLEU 27.2% compared to BLEU 28.3%) given limited computing and data resources (4 GPU and 1.5 million data, trained for only 50 epochs).

Our main contributions in this report include:

- We proposed several unparametrized structured transformation across the positional dimension, and we found the 2D unparametrized Discrete Fourier Transform could effectively replace the encoder self-attention and decoder masked self-attention with minimal performance degradation on the WMT benchmark, especially the English to Italian translation task, which, to the best of our knowledge, is the first effort to use Fourier unparametrized layers for language generation and machine translation tasks.
- We built an end-to-end distributed training pipeline for bilingual machine translation tasks including the joint byte pair encoding (BPE) [8, 2], distributed data-loading, distributed training across multiple machines and multiple GPUs, optimized dynamic batch-size selection for full utilization of computing resources, distributed testing and inference procedures with dynamic beam search algorithm. We recycled open-sourced implementations for a fast BPE algorithm and encoder-decoder transformer architecture, where we incorporated our own unparametrized transformations in lieu of attention layers.
- We proposed a method for sparsely parameterizing Fourier transform as a parametrized general Fourier Transform that allow additional flexibility for creating learnable coefficients (i.e, weights) for token-mixing across the positional dimension with little added memory footprint. We also implemented the "causally" masked Discrete Fourier Transform across the positional dimension.

2 Related Work

[14] introduced the Transformer model in 2017 and had laid the foundation for all of our analysis and implementation. Following their work, [11],[15] and [10] have started the line of research to investigate the effectiveness of the attention sub-layer.

[13] proposed to use a MLP-Mixer in the positional dimension to replace the attention layer. [17] worked on replacing the attention layer with unparameterized Gaussian distribution but they claimed the encoder-decoder cross attention layer is critical to model's performance. [7] used fixed coefficients for the attention layer weights provided one of the attention head remains learnable. Their work as well as ours indicate the learnable self-attention weights are not necessarily required to perform many NLP tasks.

Our work follow closely with the recent FNet [5] and the Performer [3] papers, where the effectiveness of using Discrete Fourier Transform to replace the self-attention layers have been investigated and tested on natural language understanding benchmarks such as GLUE [16] and LRA [9]. Different from their work, we focused on the task of natural language generation which requires the encoder-decoder attention as well as the masked decoder self-attention.

For specific implementations of Fourier Transform, we usd the Fast Fourier Transform (FFT)[4] and [12] for our encoder self-attention, and we directly used a casually masked version of the Discrete Fourier Matrix.

3 Methods

3.1 Transformer Attention

3.1.1 Self-Attention in Encoder

For the encoder in Transformer model, the methodology for self attention is:
Assume now we have batch size = 1, the input is $[x_1 \quad x_2 \quad \cdots \quad x_n]$, where x_i is a sub-word (token) id at ith position in sentence and n is the number of sub-words in sentence. After we input through embedding layer, each x_i is embedded by size E such that $\vec{x_i} \in \mathbb{R}^E$. We pack together all the embedding of input into a matrix Q, K, and V, where $Q = \begin{bmatrix} \vec{x_1} \\ \vec{x_2} \\ \vdots \\ \vec{x_n} \end{bmatrix} \in \mathbb{R}^{n \times E}$, and $Q = K = V$. Then

$$QK^T = \begin{bmatrix} \vec{x_1} \\ \vec{x_2} \\ \vdots \\ \vec{x_n} \end{bmatrix} [\vec{x_1} \quad \vec{x_2} \quad \cdots \quad \vec{x_n}] = \begin{bmatrix} <\vec{x_1},\vec{x_1}> & <\vec{x_1},\vec{x_2}> & \cdots & <\vec{x_1},\vec{x_n}> \\ <\vec{x_2},\vec{x_1}> & <\vec{x_2},\vec{x_2}> & \cdots & <\vec{x_2},\vec{x_n}> \\ \vdots & & \ddots & \\ <\vec{x_n},\vec{x_1}> & <\vec{x_n},\vec{x_2}> & \cdots & <\vec{x_n},\vec{x_n}> \end{bmatrix} \in \mathbb{R}^{n \times n}$$

We perform softmax on each row of QK^T, where

$$\text{Softmax } (QK^T) = \begin{bmatrix} P(<\vec{x_1},\vec{x_1}>) & P(<\vec{x_1},\vec{x_2}>) & \cdots & P(<\vec{x_1},\vec{x_n}>) \\ P(<\vec{x_2},\vec{x_1}>) & P(<\vec{x_2},\vec{x_2}>) & \cdots & P(<\vec{x_2},\vec{x_n}>) \\ \vdots & & \ddots & \\ P(<\vec{x_n},\vec{x_1}>) & P(<\vec{x_n},\vec{x_2}>) & \cdots & P(<\vec{x_n},\vec{x_n}>) \end{bmatrix} \in \mathbb{R}^{n \times n}$$

where $\sum_{j=1}^{n} P(<\vec{x_i},\vec{x_j}>) = 1$ for ith row, $1 \le i \le n$.

$$\text{Softmax } (QK^T)V = \begin{bmatrix} \sum_{j=1}^{n} P(<\vec{x_1},\vec{x_j}>)\vec{x_j} \\ \vdots \\ \sum_{j=1}^{n} P(<\vec{x_n},\vec{x_j}>)\vec{x_j} \end{bmatrix} \in \mathbb{R}^{n \times E}$$

3.1.2 Masked Self-Attention in Decoder

For the decoder, the methodolody for masked attention is:
Assume now we have batch size = 1, the input is $[y_1 \quad y_2 \quad \cdots \quad y_m]$, after we input through embedding layer, each y_i is embedded by size E as $\vec{x_i} \in \mathbb{R}^E$. We pack together all the embedding of

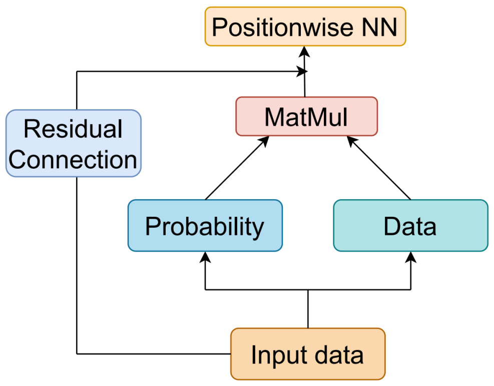

Figure 1: The structure of Transformer

input into a matrix Q, K, and V, where $Q = \begin{bmatrix} \vec{y_1} \\ \vec{y_2} \\ \vdots \\ \vec{y_m} \end{bmatrix} \in \mathbb{R}^{m \times E}$, and $Q = K = V$. Then

$$\text{Softmax }(QK^T) = \begin{bmatrix} P(<\vec{y_1}, \vec{y_1}>) & P(<\vec{y_1}, \vec{y_2}>) & \cdots & P(<\vec{y_1}, \vec{y_m}>) \\ P(<\vec{y_2}, \vec{y_1}>) & P(<\vec{y_2}, \vec{y_2}>) & \cdots & P(<\vec{y_2}, \vec{y_m}>) \\ \vdots & \ddots & & \\ P(<\vec{y_m}, \vec{y_1}>) & P(<\vec{y_m}, \vec{y_2}>) & \cdots & P(<\vec{y_m}, \vec{y_m}>) \end{bmatrix} \in \mathbb{R}^{m \times m}$$

where $\sum_{j=1}^{n} P(<\vec{x_i}, \vec{x_j}>) = 1$ for ith row, $1 \leq i \leq m$.

$$\text{Softmax }(QK^T)V = \begin{bmatrix} \sum_{j=1}^{m} P(<\vec{y_1}, \vec{y_j}>)\vec{y_j} \\ \vdots \\ \sum_{j=1}^{m} P(<\vec{y_m}, \vec{y_j}>)\vec{y_j} \end{bmatrix} \in \mathbb{R}^{m \times E}$$

For decoder, we can only know the sub-word that already comes before the current sub-word. And since $\vec{y_1}$ comes form a ”start” token such that the input to decoder is shifted right, at ith position or row, the decoder cannot know $(\vec{y_i}, \vec{y_{i+1}}, \cdots, \vec{y_m})$. Thus, we make $P(<\vec{y_i}, \vec{y_j}>) = 0$ when $j > i$ by using a lower triangular mask $M = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \ddots & & \\ 1 & 1 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{m \times m}$ and performing element-wise multiplication on mask and Softmax (QK^T) that $M \odot$ Softmax (QK^T).

3.1.3 Cross-Attention in Encoder and Decoder

For cross-attention of encoder output and decoder input that passed masked self-attention layers, the methodology is:

Assume now we have batch size = 1, the K and V from output of encoder are $K = V = \begin{bmatrix} \vec{x_1} \\ \vec{x_2} \\ \vdots \\ \vec{x_n} \end{bmatrix} \in$

$\mathbb{R}^{n\times E}$, and the Q from decoder is $Q = \begin{bmatrix} \vec{y_1} \\ \vec{y_2} \\ \vdots \\ \vec{y_m} \end{bmatrix} \in \mathbb{R}^{m\times E}$. Then

$$QK^T = \begin{bmatrix} \vec{y_1} \\ \vec{y_2} \\ \vdots \\ \vec{y_m} \end{bmatrix} [\vec{x_1} \quad \vec{x_2} \quad \cdots \quad \vec{x_n}] = \begin{bmatrix} <\vec{y_1},\vec{x_1}> & <\vec{y_1},\vec{x_2}> & \cdots & <\vec{y_1},\vec{x_n}> \\ <\vec{y_2},\vec{x_1}> & <\vec{y_2},\vec{x_2}> & \cdots & <\vec{y_2},\vec{x_n}> \\ \vdots & \ddots & & \\ <\vec{y_m},\vec{x_1}> & <\vec{y_m},\vec{x_2}> & \cdots & <\vec{y_m},\vec{x_n}> \end{bmatrix} \in \mathbb{R}^{m\times n}$$

$$\text{Softmax }(QK^T)V = \begin{bmatrix} \sum_{j=1}^{n} P(<\vec{y_1},\vec{x_j}>)\vec{x_j} \\ \vdots \\ \sum_{j=1}^{n} P(<\vec{y_m},\vec{x_j}>)\vec{x_j} \end{bmatrix} \in \mathbb{R}^{m\times E}$$

3.2 Fourier Transform

The Fourier Transform converts a function into a form that describes the frequencies present in the original function. The discrete Fourier transform (DFT) as a transformation matrix at N-point is experessed as $X = Wx$ where x is the original input and W is the DFT matrix. The transformation matrix W_N for sentence DFT in encoder, W_M for DFT in decoder, and W_E for embedding DFT are

$$W_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

$$W_M = \frac{1}{\sqrt{M}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{M-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{M-1} & \omega^{2(M-1)} & \cdots & \omega^{(M-1)(M-1)} \end{bmatrix}$$

$$W_E = \frac{1}{\sqrt{E}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{E-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{E-1} & \omega^{2(E-1)} & \cdots & \omega^{(E-1)(E-1)} \end{bmatrix}$$

where $\omega = e^{-2\pi i/N}$ is a primitive Nth root of unity satisfied that $z^n = 1$ for number z. Thus, the ω is independent of the actual value of x, it only depend on the length of x and the position in sequence.

3.2.1 Discrete Fourier Transform in Encoder

Assume now we have batch size = 1, we have input sequence as $[x_1 \quad x_2 \quad \cdots \quad x_N]$ where x_i is a sub-word (token) id at ith position in sentence and n is the number of sub-words in sentence. We perform the Fast Fourier Transform (FFT) and matrix multiplication on Encoder part. After we input through embedding layer, we have embedded input as $X = \begin{bmatrix} \vec{x_0} \\ \vec{x_1} \\ \vdots \\ \vec{x_{n-1}} \end{bmatrix} \in \mathbb{R}^{N\times E}$. We perform DFT on both dimension of embedded input by multiplying W_N and W_E on left and right sides of X.

$$\frac{1}{\sqrt{NE}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} \vec{x_0} \\ \vec{x_1} \\ \vdots \\ \vec{x_{n-1}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{E-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{E-1} & \omega^{2(E-1)} & \cdots & \omega^{(E-1)(E-1)} \end{bmatrix}$$

After the DFT, the embedded input is transformed to

$$\left\{ (\sum_{k=0}^{N-1} W_{N_{ik}} \vec{x_k}) W{E_j}^T \right\}_{ij}$$

at ith row and jth column in the embedded input.

3.2.2 Discrete Fourier Transform in Decoder

Assume now we have batch size = 1, we have input sequence as $[y_1 \quad y_2 \quad \cdots \quad y_M]$. After we input through embedding layer, we have embedded input as $Y = \begin{bmatrix} \vec{y_0} \\ \vec{y_1} \\ \vdots \\ \vec{y_{m-1}} \end{bmatrix} \in \mathbb{R}^{M \times E}$. We perform DFT on both dimension of embedded input by multiplying W_N and W_E on left and right sides of X, with W_M is masked since decoder cannot know the input after ith column for ith row. Thus we have our mask $K = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \ddots & & \\ 1 & 1 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{M \times M}$ element-wise multiply W_M, $K \odot W_M$. Then we have

$$\frac{1}{\sqrt{ME}} \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & \omega & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{M-1} & \omega^{2(M-1)} & \cdots & \omega^{(M-1)(M-1)} \end{bmatrix} \begin{bmatrix} \vec{y_0} \\ \vec{y_1} \\ \vdots \\ \vec{y_{n-1}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{E-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{E-1} & \omega^{2(E-1)} & \cdots & \omega^{(E-1)(E-1)} \end{bmatrix}$$

After the DFT, the embedded input in decoder is transformed to

$$\left\{ (\sum_{k=0}^{M-1} W_{M_{ik}} \vec{y_k}) W_{E_j}^T \right\}_{ij}$$

at ith row and jth column in the embedded input.

4 Experiment Setup

4.1 Dataset and Pre-Processing

The dataset we use is the ACL 2014 English-Italian paired translation dataset [1], which consisting of about 1.9 million sentence pairs. Sentences were encoded using byte-pair encoding [2], which has a shared source target vocabulary of about 35000 tokens. We use a fast BPE algorithm [8], an unsupervised text tokenizer for performing our byte-pair encoding. We obtained the length for each encoded sentences pairs, and by Figure2, we found out that the majority of length is below 50.

Thus, for our sentence pairs in dataset, those with length greater than 50 were truncated and the number of sentences in each range are in the Table1. Sentence pairs were first listed in ascending order by the length of English sentences, and then were divided into 10 groups based on their length. Within each group, we added `<BOS>` and `<EOS>` to the beginning and the ending of the sentences respectively, and then padded each sentence with `<PAD>` to ensure that all sentences have the same length within each group. We've also found that there are some sentence mismatching in the ACL dataset, so we filtered out some misaligned data points and we'll report this issue after we finish the whole project. As a preliminary method to deal with mismached pairs, we delete all pairs which the difference of length between English and Italian sentences are greater than 20. Then we split the dataset into train, validation, and test for each range of length in 8-1-1 ratio.

4.2 Distributed Dataloading

Since we employed a distributed training framework across multiple GPUs, we divide the training dataset equally into 4 sub training datasets after random shuffling. To prevent the distribution

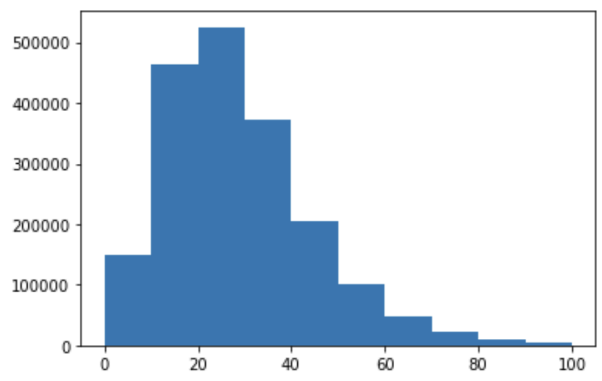

Figure 2: The distribution for the length of encoded sentence pairs

Table 1: The number of setences in each range

Range of length	Number of sentences
0-5	40315
6-10	140751
11-15	218451
16-20	265180
21-25	271525
26-30	244069
31-35	200129
36-40	153377
41-45	112679
46-50	79755
In total	1726231

of sequence length from varying across different subsets, we split each of the 10 length classes respectively into 4 sub length classes. Our model will run on different sub training datasets in parallel during forward propagation, and during the backpropagation phase, the gradients are averaged across different GPUs before being used in weight update.

4.3 Model Training

We trained our model on one machine with 4 NVIDIA RTX A5000 GPUs. We fixed the embedding size (d_E = 512) and inner hidden dimension for the dense position-wise feed forward layer (d_I =2048) for both encoder and decoder, ensuring the data matrix during forward propagation always has the shape $B \times N \times d_E$ in encoder and $B \times M \times d_E$ in decoder, where N, M are the respective input and output sequence length. The encoder and decoder layers are stacked 6 times respectively, resulting in total 55 million parameters for our model with only encoder-attention replaced, and resulting in total 49 million parameters for our model with both encoder-attention and decoder-attention replaced.

We trained our Fourier-based models for 50 epochs on the training dataset which took 48 hours and terminated it early due to resources limitation.

4.4 Dynamic Batching

We developed a dynamic batching strategy for different sequence length class to fully utilize our computing resources. $C = L \times B$ where C is a constant hyperparameter for GPU memory usage, set to $C = 15,000$. L denotes different length class, and B is minibatch size for said length class. In practice, since the target sequence is Italian, due to possible misalignment issues, sometimes length L English sequence is paired with length $L + 20$ Italian sequence. Therefore, we used $C = (L + 20) \times B$.

4.5 Optimizer and Regularization

We used the Adam optimizer and a warm-up learning rate scheduler similar to Transformer paper. For regularization, we used residual dropout with drop out rate =0.1, and label smoothing with $\epsilon = 0.1$, as suggested by the Transformer paper.

5 Results

5.1 Training and Validation Performance

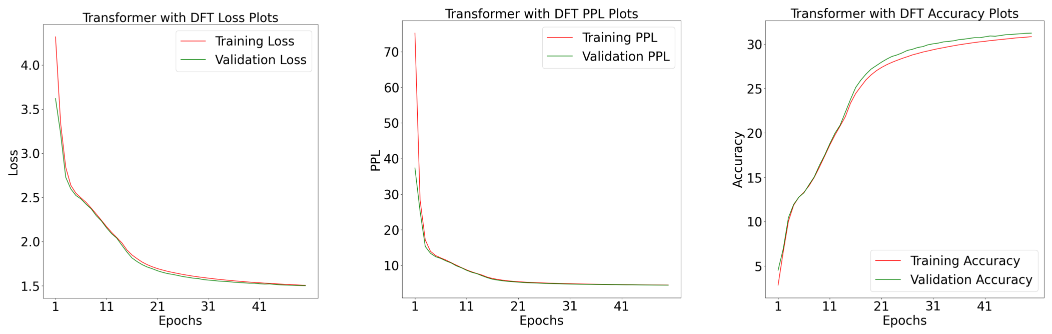

Figure 3: The loss, ppl, and accuracy for train and validation vs epochs

From Figure 3, we can see that for training and validation datasets, the cross entropy loss, the perplexity, and the next-word accuracy are always extremely close. The validation curve in next-word accuracy even outperform the training curve indicating our proposed Transformer+DFT is far from convergence. The lack of visible gaps between training and validation shows the model has not over-fitted yet and its performance would benefit from further training. Due to the limitation of computational resources at the time, we are unable to further train our proposed model, though its success from Table 2 in the machine translation task is already indicative of its potential for other language generation tasks.

5.2 BLEU Scores

Table 2: The BLEU scores for test

Range of length	BLEU1	BLEU4	BLEU	Number of sentences
0-5	3.0645	2.0586	2.3589	4027
6-10	10.064	6.1947	7.5047	14064
11-15	22.7739	13.7944	17.0454	21822
16-20	31.8099	19.3028	23.9982	26486
21-25	37.2931	22.9952	28.5155	27068
26-30	42.0169	25.9455	32.2692	24326
31-35	45.7253	28.2285	35.2254	19928
36-40	48.3839	29.6873	37.2576	15243
41-45	49.9175	30.3478	38.366	11174
46-50	50.3814	30.3498	38.6439	7886
Average	35.6272	21.8271	27.2218	172024

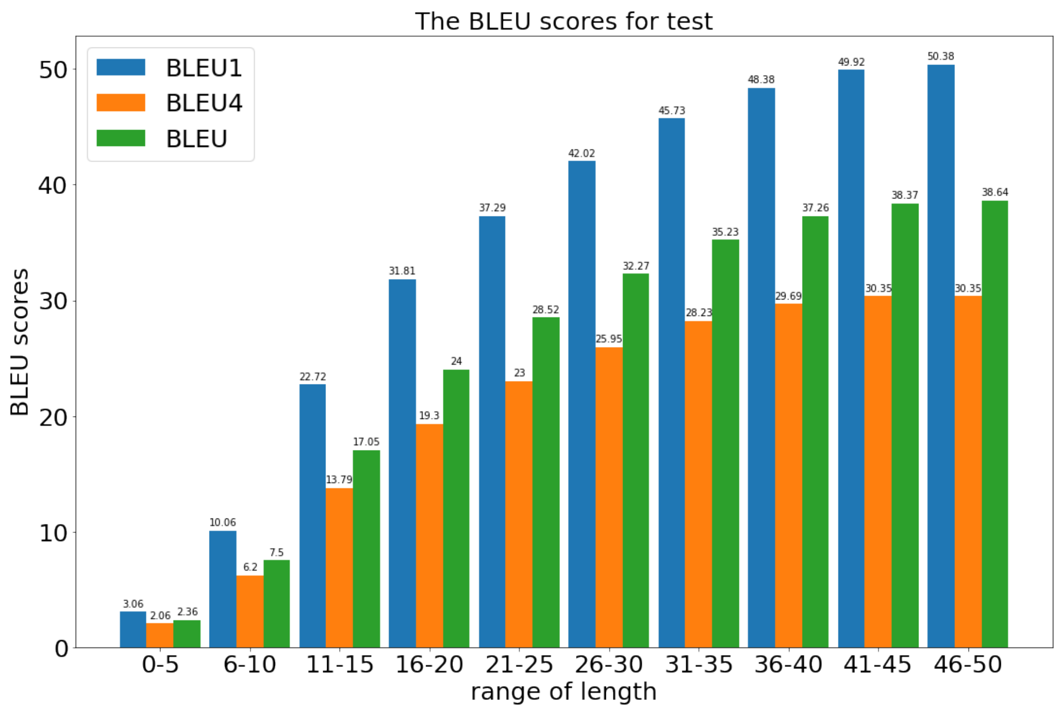

Figure 4: The distribution of BLEU1, BLEU4, and BLEU score on test for different range of length

We used a beam size of 4 during our auto-regressive inference time which also utilized distributed computing. As we can see from Table 2, the BLEU scores for different length classes can be different, especially for length class [0-5] and length class [6-10], the BLEU score can be low. This phenomenon is unsurprising, since BLEU is related to the length of the sequence: a longer sequence has a higher chance of appearing n-grams. While for shorter sequences, in many occasions the machine translation was able to recover the meaning of the original source sequence but was simply saying it in another way, which would result in low BLEU score.

5.3 BLEU Scores

Table 3: The BLEU scores for test on different models

Model	BLEU
Transformer (base model)	28.1
Transformer (big)	30.1
Transformer with F-Net	27.2

The averaged BLEU score in Table 2 for all 170k testing data is 27.2%, suggesting only minimal performance degradation from the 30.1% larger Transformer model with much more parameters.

6 Discussion

In this section, we discuss the importance and utility of having attention layers in the Transformer models. From our experimental results, we have shown that when the lower layer self-attention layers are replaced by a structured unparametrized transformation such as the Discrete Fourier Transform, this change does not impair the accuracy and quality of our proposed task (i.e., the English to Italian Machine Translation Task), as we can see in Figure 4. Since the number of n-grams could have a greater chance of being matched and found in longer sentences, it is no wonder why there is an increasing trend for the BLEU score among different sequence-length class. Many of the sentence-pairs have BLEU score over 30 which suggest decent qualities in translation tasks.

As we have noted before, unparameterized transformations do not support learnable weights (i.e. softmax probabilities as coefficients during sequence dimension mixing) and are limited in their expressibility or representational power to capture complex in-sentence or cross-sentence dynamics. Therefore, as some of the previous work [17] have shown (and we have also run some preliminary experiments on the accuracy when we replace the masked decoder self-attention with Fourier Transform), it seems that when the higher-up layers' attention layers are replaced with fixed mixing transformations, the model degrades in its ability to generate sentences. Note, the decoder layers are considered higher-up layers because they are directly below the final output prediction layer, whereas the encoder layers represent the first 6 layers in the bottom of the Transformer. Previous work [17] [5] have shown that when some of the decoder layers (especially the last 2 decoder layers) remain unchanged with the same masked self-attention and cross-attention, the performance only suffers slightly compared with a massive drop in accuracy when the cross-attention layers are replaced by fixed transform. We are also observing the fact that lower level self-attention layers have relatively static or unchanged attention patterns that are mostly unrelated with the value of the input tokens, and have more to do with the input length. Therefore, it makes intuitive sense to use structured unparameterized transform on lower levels but not on higher up levels.

We also note that [17] used Gaussian-distributed noise for fixed structured transform in contrast to the Fourier Transform investigated in our technical report and in [5]. Since other intuitions from the Fourier Transform does not align with the performance in the machine translation task, ([5] attempted to use multiple Fourier transform block within a single Transformer block but reportedly failed to increase performance). It is our opinion that any reasonably diverse and complex position-dependent mixing mechanism for the sequence dimension should serve as valid replacements for the self-attention layers in the encoder.

It is also noted that another line of research using global convolution instead of self-attention has been used in Transformer-styled models. These models can be extended to as large as 14B parameters and are shown comparable performance across many of the natural language tasks. The global convolution along the sequence dimension is a parameter-light token mixing mechanism that uses the same intuition as Fourier Transform and other unparametrized transform but also leverage their learnable convolution kernels for more representational power. These models serve as a great way when the training token length scale up to more than 10k tokens per sequence.

Lastly, regarding our experiments, because of the computational intensity and memory requirements for storing large model checkpoints on the UCSD datahub server, we were only able to train our proposed Fourier Transformer for 50 epochs before exceeding disk memory quota. But from the Figure 3, we can see the validation and training curve are closely knitted together, indicating the model is far from overfitting and far from training convergence. It is reasonable to assume even better BLEU score performance when the model is trained for more epochs (most transformer papers [14] train for over 200 epochs in total).

7 Conclusion and Future Work

7.1 Conclusion

Our model replaces the multi-head attention structure of original transformer model encoder part with a parameterless Fourier transform, which can effectively reduce the model parameters and running time while ensuring the performance of the model. It is proved that for the original transformer model, simple linear transform can be used to extract features in the shallower network architecture.

7.2 Future Work

We considers applying the same Fourier structure to the decoder's multi-head attention and exploring the effect of replacing multi-head attention with different linear transformation operations on model performance.

Another interesting line of research is to utilize the simplicity of unparameterized transformation for model distillation when we use a large language model as teacher and force a smaller language model (potentially using unparameterized transform in its attention layers) to learn its inference behavior for a variety of downstream tasks. This line of work is important because the difficulty to deploy large language models on personal devices due to their heavy memory footprint.

Furthermore, there is also the research direction in robustness verification of large language model during training and during inference. Since most in-complete bound verifiers turn the robustness certification problems as a primal problem in convex optimization, the task of dualizing it requires getting the upper and lower bound of the conjugate functions of their dual layers. The process of approximating lower and upper bounds requires using the stationarity condition in the KKT conditions where we need to explicitly calculate gradients of the conjugate functions for each layer. As it turns out, the self-attention layer's joint conjugate function does not admit an analytical solution, whereas a fixed linear layer has a simple solution. It would be interesting to run these network verification analysis on a Transformer model with attention-layers replaced by fixed structured transform and check if it improves the model's overall ability to be robust against adversarial input.

8 Authors' Contribution

It is our consensus that all authors contributed equally and roughly the same amount of time in this technical report though with different focus.

- Andre Wang worked on developing and researching the idea of using Fourier Transform and other token-mixing strategy with Transformer models. He also worked on composing this technical report with Xiaoyue Wang. In addition, he worked on collecting, curating and preprocessing the machine translation data with Xiaoyue Wang.
- Yuzhao Chen focused on implementing the distributed learning and inference pipeline with the help of Xiaoyue Wang and Andre Wang, which is a hard part of this project. He also monitored the training progress as well as caching critical checkpoints and logging training/testing statistics.
- Xiaoyue Wang worked across the entire project from data cleaning, pre-processing, to model formulation, implementation as well as the drafting of this technical report. She assisted other members and provided help in the form of pair-programming.
- Yongce Li worked on literature review with Andre Wang. He provided a detailed report of the landscape of the field and he also worked on reducing the runtime complexity of several Fourier-based algorithms. In addition, he helped in running some of the experiments and managing GPU resources.

References

[1] Ondřej Bojar et al. "Proceedings of the Ninth Workshop on Statistical Machine Translation". In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. 2014.

[2] Denny Britz et al. "Massive Exploration of Neural Machine Translation Architectures". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 1442–1451.

[3] Krzysztof Choromanski et al. *Masked Language Modeling for Proteins via Linearly Scalable Long-Context Transformers*. 2020. arXiv: `2006.03555 [cs.LG]`.

[4] James W. Cooley and John W. Tukey. "An algorithm for the machine calculation of complex Fourier series". In: *Mathematics of Computation* 19 (1965), pp. 297–301.

[5] James Lee-Thorp et al. "FNet: Mixing Tokens with Fourier Transforms". In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2022, pp. 4296–4313.

[6] Kishore Papineni et al. "Bleu: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318.

[7] Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. "Fixed Encoder Self-Attention Patterns in Transformer-Based Machine Translation". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020, pp. 556–568.

[8] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. DOI: `10.18653/v1/P16-1162`. URL: `https://aclanthology.org/P16-1162`.

[9] Yi Tay et al. "Long Range Arena: A Benchmark for Efficient Transformers". In: *ICLR 2021* abs/2011.04006 (2020).

[10] Yi Tay et al. "Synthesizer: Rethinking self-attention for transformer models". In: *International conference on machine learning*. PMLR. 2021, pp. 10183–10192.

[11] Ian Tenney, Dipanjan Das, and Ellie Pavlick. "BERT Rediscovers the Classical NLP Pipeline". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 4593–4601. DOI: `10.18653/v1/P19-1452`. URL: `https://aclanthology.org/P19-1452`.

[12] "The design and implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005), pp. 216–231.

[13] Ilya O Tolstikhin et al. "Mlp-mixer: An all-mlp architecture for vision". In: *Advances in neural information processing systems* 34 (2021), pp. 24261–24272.

[14] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[15] Jesse Vig and Yonatan Belinkov. "Analyzing the Structure of Attention in a Transformer Language Model". In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 2019, pp. 63–76.

[16] Alex Wang et al. "GLUE: A multi-task benchmark and analysis platform for natural language understanding". In: *arXiv preprint arXiv:1804.07461* (2018).

[17] Weiqiu You, Simeng Sun, and Mohit Iyyer. "Hard-Coded Gaussian Attention for Neural Machine Translation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 7689–7700.