TUFORMER: DATA-DRIVEN DESIGN OF EXPRESSIVE TRANSFORMERS WITH LEARNABLE WEIGHT SHARING ACROSS HEADS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers are neural network architectures that achieve remarkable performance in many areas. However, the core component of Transformers, multi-head self-attention (MHSA), is mainly derived from heuristics, and the interactions across its components are not well understood. To address the problem, we first introduce a mathematically rigorous and yet intuitive *tensor diagram* representation of MHSA. Guided by tensor diagram representations, we formulate a design space where we can analyze the expressive power of the network structure, providing new directions and possibilities for enhanced performance. We then propose a novel design, namely *Tunable Transformer* (Tuformers), with a guaranteed higher expressive power than MHSA by allowing learnable weight sharing across heads. Unlike vanilla Transformer models, where the number of heads is a pre-defined fixed constant, Tuformer's structure is data-driven, and the number of heads is trainable. Training of Tuformers could be made very efficient as it allows initialization from existing pre-trained Transformer models. We test Tuformers on various tasks across multiple domains and show competitive results under a wide range of model sizes.

1 INTRODUCTION

Transformer models are first introduced by Vaswani et al. (2017) in the context of sequence modelling. They have demonstrated impressive results on a wide variety of tasks in many fields, such as language model pre-training (Sanh et al., 2019), speech recognition (Moritz et al., 2020), image classification (Dosovitskiy et al., 2020), and generation (Parmar et al., 2018). The core component of Transformer models is the *multi-head self-attention* (MHSA) which is extended from the standard attention mechanism (Bahdanau et al., 2014). Each attention head in MHSA has a global receptive field, i.e., each token's representation is updated by attending to all other tokens, and H attention heads are computed in parallel and concatenated together.

The current MHSA design is mainly derived from empirical studies or heuristics, leaving some unresolved challenges. **(1) Lack of solid theoretical understanding.** The theory behind MHSA is only starting to catch up with practice. The role of the components and their interactions in MHSA are not well understood rigorously from a theoretical perspective, which may lead to inefficiencies in the design. For example, Michel et al. (2019) show that most attention heads can be removed in the testing phase without much performance compromise, while Cordonnier et al. (2020) find empirical evidence of redundancy in key/query projections and propose a re-parameterization scheme. These works focus more on practical solutions, leading to questions about whether theoretical patterns exist in these designs. **(2) The number of heads is not trainable.** Intuitively, the heads in multi-heads are expected to capture different context information through each head. However, the number of heads is fixed in training. Thus, although we could tune the hyper-parameter, an exhaustive search would be time-consuming or not practical for large-scale tasks. **(3) Hard to analyze the expressive power.** Analyzing the expressive power (as will be defined in Definition 2) of a neural network, i.e., proving that some architectures are more expressive than others, is a non-trivial and challenging task. Such analysis has been done in Convolutional Neural Networks (CNNs) (LeCun et al., 1995; Cohen et al., 2016), Recurrent Neural Networks (RNNs) (Mikolov et al., 2010; Khrulkov et al., 2018),but no such work exists for interpreting MHSA or guiding the structural design.

In response to the above challenges, we first **interpret** MHSA from a tensor representation perspective using the intuitive graphical tool, the *tensor diagram*. Current prevalent descriptions of MHSA use flow charts to convey high-level intuitions, which could cause ambiguities. Therefore, it is inevitable to pair those flow charts with mathematical formulas to understand the mechanism precisely. However, these two separated descriptions create difficulties for interpretations and inspections of the operations implemented in MHSA. To address this issue, we propose a graphical representation of MHSA, which is both semantically intuitive and mathematically rigorous. Specifically, we modify and extend the vanilla *tensor diagrams* (Penrose, 1971), which conveniently allow for rigorous graphical representation of multi-linear operations between multi-dimensional arrays (i.e., higher-order tensors) to represent nonlinear operations.

We also **propose a novel data-driven structure** that is guaranteed to have higher expressive power. Inspired from the tensor diagram representation of MHSA, we obtain a novel design, namely *Tunable-Head Self-Attention (THSA)*, which is a re-parameterization of the weight matrices to allow learnable weight sharing across heads. Transformers with THSA, named *Tunable Transformers (Tuformers)*, have several advantages compared against vanilla Transformers: **(1)** A guaranteed higher expressive power. We prove that MHSA is a special case of THSA. **(2)** The number of heads is trainable. The concept of the number of heads in THSA generalizes to the stable rank of the core matrix, allowing data-driven implicit training. **(3)** Tuformers allow initialization from pre-trained Transformers such as BERT (Devlin et al., 2019) and its variants.

We experiment Tuformers with several tasks across multiple domains, from language modeling, machine translation to image generation under a wide range of model sizes. We demonstrate competitive results not only on Tuformers but also in cases where Tuformers are initialized with pre-trained Transformers for other downstream tasks and when combined with Linear Transformer (Katharopoulos et al., 2020) on the image generation task.

Using tensor diagram representation, we project the current design of MHSA into its tensor form, which renders a holistic view of the weights in MHSA for better interpretation of the information flow and exchange among the components of MHSA. More importantly, we **formulate a design space** for MHSA that allows improving the expressive power of the network by finding better tensor representations of the weights through tensor representation theory in followup works.

Summary of Contributions:
(1) We propose a mathematically rigorous and semantically intuitive *tensor diagram* representation of the multi-head self-attention, introducing a new tool to the ML community for future studies on interpretation and improvements of Transformers.
(2) We propose a novel design of the MHSA, *Tunable-Head Self-Attention (THSA)*, resulting in *Tuformers*, which has a theoretical guaranteed improvement in expressive power.
(3) We provide a constructive re-parameterization scheme from vanilla Transformers to *Tuformers* so that *Tuformers* can be easily used in fine-tuning tasks with pre-trained models and efficient Transformers such as Linear Transformer (Katharopoulos et al., 2020) and Performer (Choromanski et al., 2020) with the state-of-the-art computation and memory complexities.
(4) We formulate a design space for attention unit design and a framework for analyzing their expressive power using tensor representation theory under the intuitive guidance of tensor diagrams.

2 Tensor Diagram Representation

Notations. We use lower case letters (e.g., $\boldsymbol{v}$) to denote vectors, upper case letters (e.g., $\boldsymbol{M}$) to denote matrices, and curly letters (e.g., $\boldsymbol{\mathcal{T}}$) to denote general tensors. For a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times \cdots I_M}$, we refer to the number of indices as *order*, each individual index as *mode*, and the length of one mode as *dimension*. For instance, $\boldsymbol{\mathcal{T}}$ is an Mth order tensor that has dimension I_m at its mth mode. We reserve superscripts to distinguish similar arrays (e.g., $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V$ are query/key/value weight matrices), and subscripts to index the elements in an array (e.g., $\boldsymbol{W}_{ij}$ is the $(i,j)^{\text{th}}$ element of $\boldsymbol{W}$). We use colon **:** to slice an array (e.g., $\boldsymbol{W}_{i,:}$ denotes the ith row of $\boldsymbol{W}$).

We propose to use *tensor diagrams* (Penrose, 1971), a commonly used rigorous/precise and intuitive graphical representation for multi-linear operations among higher-order arrays (i.e., tensors), to represent multi-head self-attention (MHSA). Since MHSA consists of multilinear operations and a nonlinear softmax function, we will introduce tensor diagrams and our novel design extending tensor diagrams to denote the composition of nonlinear and multilinear operations in MHSA.

2.1 TENSOR DIAGRAM BASICS

Arrays denoted as nodes with legs. An array is represented as a node with leg(s) in a tensor diagram as shown in Figure 1. We denote the *order* (the number of dimensions) of the array by the number of legs extending from the node. Each labeled leg represents one *mode* of a tensor. Every mode of the tensor needs to be uniquely labeled. We usually use the dimension of the mode as the label (i.e., an associated positive integer written on top of each leg). The legs do not need to be straight lines, and their orientations do not matter. Matrices $\boldsymbol{M} \in \mathbb{R}^{A\times B}$ and $\boldsymbol{M}^\top \in \mathbb{R}^{B\times A}$ can be represented via the same tensor diagram as long as the $\boldsymbol{M}$ node has two legs (to denote that it is a matrix, wherever the legs extend to), labeled as A and B (to denote its size).

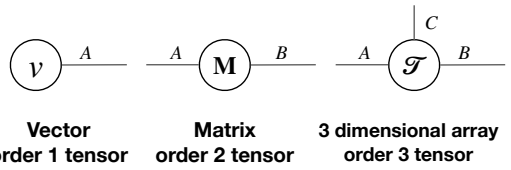
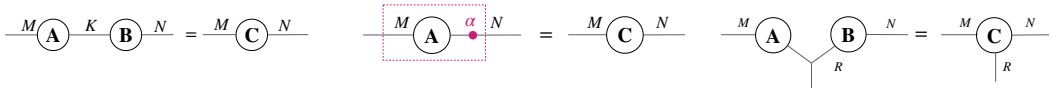


Figure 1: Arrays in tensor diagram. A vector is denoted as a node with 1 leg, a matrix as a node with 2 legs and an N-dimensional array as a node with N legs.

Operations in tensor diagrams. There are three types of operations in the calculation of MHSA:, *contraction*, *softmax*, and *batch multiplication*, as shown in Figure 2 and explained in its caption.



$$\sum_{k=1}^{K} \boldsymbol{A}_{mk}\boldsymbol{B}_{kn} = \boldsymbol{C}_{mn} \qquad \frac{\exp(\alpha \cdot \boldsymbol{A}_{mn})}{\sum_{n'=1}^{N} \exp(\alpha \cdot \boldsymbol{A}_{mn'})} = \boldsymbol{C}_{mn} \qquad \boldsymbol{A}_{mr}\boldsymbol{B}_{nr} = \boldsymbol{\mathcal{C}}_{nmr}$$

Figure 2: Tensor diagrams for atomic operations. (1) (left) Contraction is an operation that generalizes the matrix multiplication. It sums element-wise products on a mode in object $\boldsymbol{A}$ and a corresponding mode (with the same dimension) in object $\boldsymbol{B}$ (i.e., along a leg in node $\boldsymbol{A}$ and a corresponding leg in node $\boldsymbol{B}$). In the tensor diagram, multiplying two matrices (or higher-order tensors with more than 2 legs) corresponds to *"gluing" their corresponding legs (on a certain mode)*. **(2) (middle) Softmax** is an element-wise exponential function normalized along a certain mode. We propose to denote the α*-scaled softmax* function $\mathsf{softmax}(\alpha\boldsymbol{A})$ on $\boldsymbol{A}$ as *a dotted box with a labeled filled ball* (to distinguish itself from tensor objects, i.e., nodes which are blank circles) attached to one leg. **(3) (right) Batch multiplication** is an element-wise product along the connected legs.

Evaluation of tensor diagrams: (1) Evaluation order. We can evaluate a tensor diagram in any pair-wise order except for the nodes in the softmax box. Since the softmax function is nonlinear, we must first evaluate the nodes in the softmax box with arbitrary order before those outside the box. **(2) Reading a tensor diagram.** We can easily identify the output shape by the dangling edges. For instant, in Section 2.1, there are three dangling legs M, N, R. Thus the output is a 3rd order tensor with dimensions M, N, R. Note that the softmax function does not change the shape of the tensor.

Advantages of tensor diagrams: (1) Tensor diagram is *orientation invariant*, meaning that we can represent $\boldsymbol{X}$ and $\boldsymbol{X}^\top$ using the same tensor diagram. Thus in multi-head, we obtain a universal graphical representation regardless of whether we represent an embedding of each token in the sequence as rows or columns of the input embedding matrix $\boldsymbol{X}$. **(2)** Multi-linear operations are concisely represented and interpreted. **(3)** The representation is both precise in math and intuitive, and the information flow is clear, making analysis of network structure more accessible. In addition, with the labels of the legs, we can read the model complexity explicitly.

We include a comprehensive introduction to tensor diagram representation in Appendix A.

2.2 TENSOR DIAGRAM REPRESENTATION OF MULTI-HEAD SELF-ATTENTION

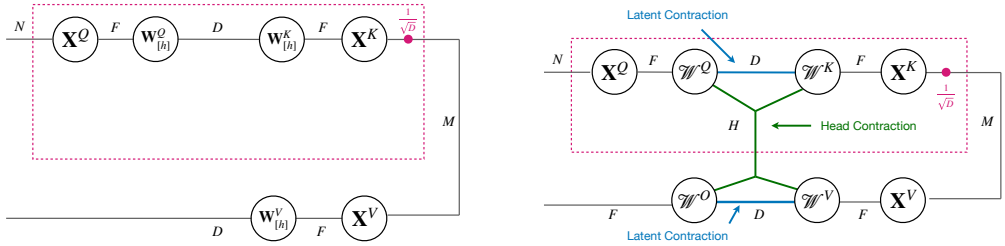
The core of Transformer models is a *multi-head self-attention* (MHSA) module which allows the model to jointly attend to information from different representation sub-spaces at different positions (Vaswani et al., 2017). To distinguish inputs for query, key, value matrices, we use $\boldsymbol{X}^Q \in \mathbb{R}^{N\times F}$, $\boldsymbol{X}^K, \boldsymbol{X}^V \in \mathbb{R}^{M\times F}$ respectively. The MHSA module outputs a matrix $\boldsymbol{M} \in \mathbb{R}^{N\times F}$ as

$$\boldsymbol{Q}_{[h]} = \boldsymbol{X}^Q \boldsymbol{W}^Q_{[h]};\ \boldsymbol{K}_{[h]} = \boldsymbol{X}^K \boldsymbol{W}^K_{[h]};\ \boldsymbol{V}_{[h]} = \boldsymbol{X}^V \boldsymbol{W}^V_{[h]}, \tag{1a}$$

$$\mathbf{head}_{[h]} = \mathsf{softmax}\left(\boldsymbol{Q}_{[h]} \boldsymbol{K}^\top_{[h]} / \sqrt{D}\right) \boldsymbol{V}_{[h]}, \tag{1b}$$

$$\boldsymbol{M} = \left[\mathbf{head}_{[1]},\ \mathbf{head}_{[2]},\ \cdots,\ \mathbf{head}_{[H]}\right] \boldsymbol{W}^O. \tag{1c}$$

In the above equations, we see how MHSA computes its output in three *steps*:
(1a) Linear transformation on input embedding to get latent features. For each h, the query/key/value matrices $\boldsymbol{Q}_{[h]}$, $\boldsymbol{K}_{[h]}$, $\boldsymbol{V}_{[h]}$ are linear transformations, parameterized by $\boldsymbol{W}^Q_{[h]}$, $\boldsymbol{W}^K_{[h]}$, and $\boldsymbol{W}^V_{[h]} \in \mathbb{R}^{F \times D}$ respectively, of the input embedding.
(1b) Scaled dot-product attention. Each $\mathbf{head}_{[h]} \in \mathbb{R}^{N \times D}$ computes a scaled dot-product between a latent feature matrix $\boldsymbol{Q}_{[h]} \in \mathbb{R}^{N \times D}$ (i.e., a *query* matrix) and another latent feature matrix $\boldsymbol{K}_{[h]} \in \mathbb{R}^{M \times D}$ (i.e., a *key* matrix) along D, and applies a softmax function to obtain the weights to be multiplied with a third latent feature matrix $\boldsymbol{V}_{[h]} \in \mathbb{R}^{M \times D}$ (i.e., a *value* matrix).
(1c) Concatenation and contraction for MHSA. After concatenation of the H heads, a linear transformation of the concatenated heads, parameterized by $\boldsymbol{W}^O \in \mathbb{R}^{HD \times F}$, is implemented to get the outcome $\boldsymbol{M}$ of the MHSA as a $\mathbb{R}^{N \times F}$ object.



(a) Single-Head Self-Attention

(b) Multi-head Self-Attention

Figure 3: Figure 3a shows the tensor diagram of a single attention head. Figure 3b shows MHSA. See Appendix B for more details and the proof.

Figure 3a shows how Single-Head Self-Attention (SHSA) is represented in tensor diagram according to *Step (1b)*. Then we will see how to extend SHSA to MHSA as in *Step (1c)* (Figure 3b).

Multi-head self-attention in tensor diagram. The *concatenation* of H matrices $\{\mathbf{head}_{[h]}\}_{h=1}^H$ can be viewed as constructing a three dimensional array with the third mode being the number of heads, i.e., three-legged nodes in tensor diagram notation. This way, the weight matrices $\{\boldsymbol{W}^Q_{[h]}\}_{h=1}^H$ are stacked to be an order 3 tensor $\boldsymbol{\mathcal{W}}^Q$. It is the same with other weight matrices. Since in MHSA, the attention calculation among key, query and value matrices are done separately within each head, batch multiplication along mode H is a proper way to denote these "head-wise" operations. Finally, we connect the resultant tensor with $\boldsymbol{\mathcal{W}}^O$ to obtain MHSA as in Figure 3b. See Appendix B for more details and the proof.

3 Tuformers: Data-Driven Design with Higher Expressive Power

3.1 Motivation: Interpretation of MHSA

In MHSA, $\boldsymbol{Q}_{[h]}$ and $\boldsymbol{K}_{[h]}$ can only contribute to $\mathbf{head}_{[h]}, \forall h$. In order to characterize the information flow in MHSA, we stack the weight matrices $\boldsymbol{W}^Q_{[h]}, \forall h$ together (similarly for $\boldsymbol{W}^K_{[h]}$ and $\boldsymbol{W}^V_{[h]}$) and introduce a matrix $\boldsymbol{C}$ to interpret MHSA. Guided by its tensor diagram representation in Figure 3b, $\boldsymbol{C}$ is clearly characterized as in Proposition 1.

Proposition 1 (Interpretation of MHSA)**.** *With weight matrices stacked as* $\overline{\boldsymbol{W}}^Q := [\boldsymbol{W}^Q_{[1]}, \cdots, \boldsymbol{W}^Q_{[H]}], \overline{\boldsymbol{W}}^K := [\boldsymbol{W}^K_{[1]}, \cdots, \boldsymbol{W}^K_{[H]}], \overline{\boldsymbol{W}}^V := [\boldsymbol{W}^V_{[1]}, \cdots, \boldsymbol{W}^V_{[H]}] \in \mathbb{R}^{F \times DH}$*, and a core matrix* $\boldsymbol{C} := \boldsymbol{I}_H \otimes (\mathbf{1}_D \mathbf{1}_D^\top) \in \mathbb{R}^{DH \times DH}$ *(i.e., a Kronecker product of an all-ones matrix* $\mathbf{1}_D \mathbf{1}_D^\top \in \mathbb{R}^{D \times D}$ *and an identity matrix* $\boldsymbol{I}_H \in \mathbb{R}^{H \times H}$*), MHSA is equivalent to*

$$\overline{\boldsymbol{Q}} = \boldsymbol{X}^Q \overline{\boldsymbol{W}}^Q;\ \overline{\boldsymbol{K}} = \boldsymbol{X}^K \overline{\boldsymbol{W}}^K;\ \overline{\boldsymbol{V}} = \boldsymbol{X}^V \overline{\boldsymbol{W}}^V, \tag{2a}$$

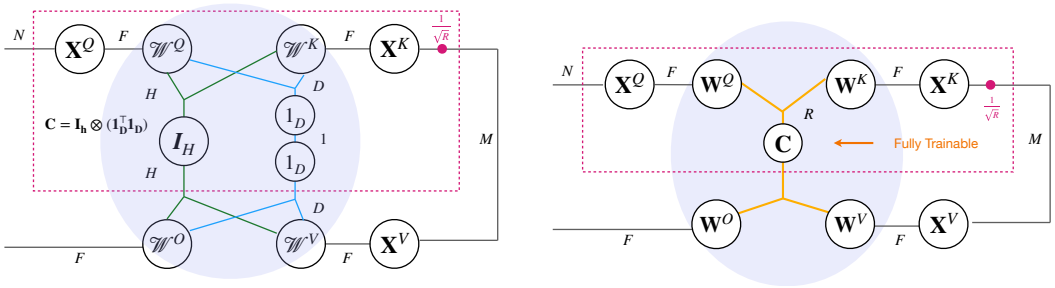
$$\overline{\mathsf{head}}_r = \mathsf{softmax}\left(\sum_{s=1}^{DH} \boldsymbol{C}_{rs}\, \overline{\boldsymbol{Q}}_s \overline{\boldsymbol{K}}_s^\top / \sqrt{D}\right) \overline{\boldsymbol{V}}_r, \tag{2b}$$

$$\boldsymbol{M} = \left[\overline{\mathsf{head}}_1, \overline{\mathsf{head}}_2,\ \cdots,\ \overline{\mathsf{head}}_{DH}\right] \boldsymbol{W}^O. \tag{2c}$$

In tensor diagram notation, MHSA is equivalent to Figure 4a. Here, we use $\overline{\boldsymbol{Q}}_r, \overline{\boldsymbol{K}}_r, \overline{\boldsymbol{V}}_r \in \mathbb{R}^F$, $\overline{\mathsf{head}}_r \in \mathbb{R}^N$ *to denote the* r^{th} *column of the corresponding matrices, and* $\boldsymbol{C}_{rs}$ *to denote the* $(r,s)^{\text{th}}$ *element of* $\boldsymbol{C}$.

Proposition 1 is proved in Appendix D.1. Note that $\{\overline{\mathsf{head}}_1, \cdots, \overline{\mathsf{head}}_D\}$ correspond to $\mathsf{head}_{[1]}$, $\{\overline{\mathsf{head}}_{D+1}, \cdots, \overline{\mathsf{head}}_{2D}\}$ correspond to $\mathsf{head}_{[2]}$ and so forth. The intuition of interpreting MHSA using Proposition 1 comes from the tensor diagram graphical representation of MHSA structure. From the tensor diagram of MHSA as shown in Figure 3b, we see two types of contractions: latent contractions along D and head contractions along H. The former corresponds to $(\mathbf{1}_D \mathbf{1}_D^\top)$ part of $\boldsymbol{C}$ and the latter corresponds to $\boldsymbol{I}_H$ part of $\boldsymbol{C}$ as shown in Figure 4a.

Motivation for an improved design: from MHSA (a preset C) to THSA (a learnable C). In MHSA, with the preset specific $\boldsymbol{C} = \boldsymbol{I}_H \otimes (\mathbf{1}_D \mathbf{1}_D^\top)$, it restricts the possible weight sharing. In other words, $\boldsymbol{Q}_{[h]}$ and $\boldsymbol{K}_{[h]}$ can only contribute to $\mathsf{head}_{[h]}, \forall h$. However, there is no motivation why such a preset and fixed $\boldsymbol{C}$ is desirable or why we do not allow $\boldsymbol{Q}_{[h]}$ or $\boldsymbol{K}_{[h]}$ to contribute to other heads. Our motivation for an improved design is simple: rather than presetting the core matrix as $\boldsymbol{C} = \boldsymbol{I}_H \otimes (\mathbf{1}_D \mathbf{1}_D^\top)$, we allow a data-driven learnable $\boldsymbol{C}$, and we call such a design *Tunable-head self-attention (THSA)*. Now with the learnable $\boldsymbol{C}$ in THSA, we allow $\boldsymbol{Q}_{[h]}$ and $\boldsymbol{K}_{[h]}$ to contribute to other heads $\mathsf{head}_{[h']}$, where $h' \neq h$. In other words, THSA allows weight sharing across heads.



(a) Multi-head Self-Attention (MHSA) **(b)** Tunable-Head Self-Attention (THSA)

Figure 4: Multi-Head Self-Attention as a special case of Tucker-Head Self-Attention.

3.2 A Novel Design: Tunable-Head Self-Attention (THSA)

As shown in Figure 4b, a Tunable-head self-attention (THSA) module has the same input and output domains as an MHSA, i.e., it takes the input embedding $\boldsymbol{X}^Q \in \mathbb{R}^{N \times F}$ and $\boldsymbol{X}^K, \boldsymbol{X}^V \in \mathbb{R}^{M \times F}$ as inputs and returns a matrix $\boldsymbol{T} \in \mathbb{R}^{N \times F}$ as output. A THSA, with five weight matrices $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V \in \mathbb{R}^{F \times R}$, $\boldsymbol{W}^O \in \mathbb{R}^{R \times F}$, and $\boldsymbol{C} \in \mathbb{R}^{R \times R}$, is mathematically described as

$$\boldsymbol{Q} = \boldsymbol{X}^Q \boldsymbol{W}^Q;\ \boldsymbol{K} = \boldsymbol{X}^K \boldsymbol{W}^K;\ \boldsymbol{V} = \boldsymbol{X}^V \boldsymbol{W}^V, \tag{3a}$$

$$\boldsymbol{\Omega}_r = \mathsf{softmax}\left(\sum_{s=1}^{R} \boldsymbol{C}_{rs}\, \boldsymbol{Q}_s \boldsymbol{K}_s^\top / \sqrt{R}\right) \boldsymbol{V}_r, \tag{3b}$$

$$\boldsymbol{T} = \boldsymbol{\Omega} \boldsymbol{W}^O = [\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2,\ \cdots,\ \boldsymbol{\Omega}_R]\, \boldsymbol{W}^O. \tag{3c}$$

The intermediate result $\boldsymbol{\Omega}_r \in \mathbb{R}^N$ is the rth column of the matrix $\boldsymbol{\Omega}$. In MHSA, the softmax function is scaled, which normalizes its input by $\sqrt{D}$, i.e., the square root of the latent dimension. The scaling ensures that the distribution of the attention matrix approximates a standard Gaussian distribution. To achieve a similar effect, in THSA, we change the constant to $\sqrt{R}$ accordingly.

THSA generalizes MHSA. THSA has learnable $\boldsymbol{C}$. **(1)** If $\boldsymbol{C} = \mathbf{1}_D \mathbf{1}_D^\top$, i.e., an $D \times D$ all-ones matrix, the THSA module reduces to an MHSA with a single head and latent dimension D (i.e.,

a single-head self-attention with latent dimension D). **(2)** If $\boldsymbol{C} = \boldsymbol{I}_H$, i.e., an $H \times H$ identity matrix, the THSA reduces to a MHSA with H heads and latent dimension 1 (i.e., a heads-only self-attention with H heads). **(3)** If $\boldsymbol{C} = \boldsymbol{I}_H \otimes (\mathbf{1}_D \mathbf{1}_D^\top)$, i.e., a Kronecker product of an all-ones matrix $\mathbf{1}_D \mathbf{1}_D^\top \in \mathbb{R}^{D \times D}$ and an identity matrix $\boldsymbol{I}_H \in \mathbb{R}^{H \times H}$, the THSA reduces to an MHSA with H heads and latent dimension D, as illustrated in Figure 4a. **(4)** We call a Transformer with THSA module as Tunable Transformers (Tuformers). The tensor diagram representation of THSA is given in Figure 4b. It has an learnable core matrix $\boldsymbol{C}$ compared to MHSA in Figure 4a which has a preset core matrix.

Tuformers can be initialized with Pre-trained Transformers. Although different in the structure, Tuformers can always be initialized with pre-trained Transformer models through re-parameterization. MHSA is updated by weight matrices $\{\boldsymbol{W}_{[h]}^K\}_{h=1}^H$, $\{\boldsymbol{W}_{[h]}^Q\}_{h=1}^H$, $\{\boldsymbol{W}_{[h]}^V\}_{h=1}^H$ and $\boldsymbol{W}^O$ while in THSA we have $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, $\boldsymbol{W}^V$, $\boldsymbol{W}^O$ and $\boldsymbol{C}$. To initialize THSA with MHSA, we follow the scheme proposed in Proposition 1. $\boldsymbol{C}$ is the kronecker product between a $D \times D$ all-one matrix and an $H \times H$ identity matrix, where D is the latent dimension and H is the number of heads in MHSA. $\boldsymbol{W}^O$ remains the same as that of MHSA. As for $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$ and $\boldsymbol{W}^V \in \mathbb{R}^{F \times R}$, we concatenate $\{\boldsymbol{W}_{[h]}^K\}_{h=1}^H$, $\{\boldsymbol{W}_{[h]}^Q\}_{h=1}^H$, $\{\boldsymbol{W}_{[h]}^V\}_{h=1}^H \in \mathbb{R}^{F \times D}$ along mode D.

The notion of heads in THSA. Due to the learnable $\boldsymbol{C}$ and data-dependent weight sharing mechanism, the notion of heads in THSA is not as obvious as in MHSA. Although we have an upper bound R of the number of heads, which is achieved when the learned $\boldsymbol{C}$ decides not to share any weights across heads, the actual effective number of heads depends on $\boldsymbol{C}$. As a result, we define the number of heads in THSA as the stable rank of the core matrix $\boldsymbol{C}$, which is $\sum_i \sigma_i^2 \big/ \max_i \sigma_i$, where σ_i is the singular value of $\boldsymbol{C}$. We use the stable rank because it is largely unaffected by tiny singular values.

3.3 Expressive Power of Tuformers

We first formally describe how to compare the expressive power between two models.

Definition 2 (Expressive Power)**.** *Suppose we have two function classes $\mathbb{F}$, $\mathbb{G}$ with the same source domain $\mathcal{X}$ and target domain $\mathcal{Y}$, i.e., each function $f \in \mathbb{F}$ (or $g \in \mathbb{G}$) is a mapping from $\mathcal{X}$ to $\mathcal{Y}$. We say $\mathbb{G}$ is more expressive than $\mathbb{F}$ if $G \supseteq F$: for any $f \in \mathcal{F}$, there exists $g \in G$ such that $g(x) = f(x), \forall x \in \mathcal{X}$. Furthermore, we say $\mathbb{G}$ is strictly more expressive than $\mathbb{F}$ if $\mathbb{G} \supset \mathbb{F}$: besides $G \supseteq F$, there exists $g \in \mathbb{G}$ such that given any f there exists $x \in \mathcal{X}$ and $g(x) \neq f(x)$.*

According to the definition, expressive power is a partial order set. Therefore, expressive power of different models is usually not comparable as two sets might have overlap and non-overlap regions. However, when one is a superset or subset of the other, we can unambiguously compare their expressive power. In the following theorem, we formally show that a Tunable-head self-attention (THSA) module has a higher expressive than multi-head self-attention (MHSA) if the rank R in THSA is equal to the product of the number of heads H and the latent dimension D in MHSA.

Theorem 3 (THSA is more expressive than MHSA)**.** *Given $R = DH$ and other hyper-parameters being the same, a Tunable-head self-attention (THSA) module with rank R is more expressive than a multi-head self-attention (MHSA) module with H heads and latent dimension D.*

Theorem 3, proved in Appendix D.2, shows that any function realizable by MHSA can be realized by THSA.

A Design Space. With the tensor diagram of MHSA in Section 2.2, it is clear that if we take a holistic view of the weight tensors $\boldsymbol{\mathcal{W}}^Q, \boldsymbol{\mathcal{W}}^K, \boldsymbol{\mathcal{W}}^V, \boldsymbol{\mathcal{W}}^O \in \mathbb{R}^{H \times D \times F}$ and consider them as components of an order 4 tensor of size $F \times F \times F \times F$ (the "super node" represented as the shaded ellipse in Figure 4a, although this order 4 tensor is not a tensor that is free to have any forms because of the limitation brought by the softmax box), then we can use the tensor representation theory to "re-parameterize" the 4th order tensor. This reveals a design space for Transformers. Our THSA serves as a specific design of the "super node", with a proved higher expressive power than MHSA. Such a design space makes it possible for optimal structure search using tensor representation theories and can inspire better designs and deeper understandings of MHSA and Transformers.

4 EXPERIMENTS

In this section, we show that THSA (used in Tuformers) universally outperforms MHSA (used in Transformers) under a wide range of model scales in diverse tasks, including natural language processing, automatic speech recognition and image generation.

We also demonstrate that THSA is simple to use in practice. First, THSA is a plug-and-play module that can be used in existing Transformer architectures. Second, THSA can be initialized from pre-trained MHSA modules (to avoid expensive training from scratch). For instance, fine-tuning Tuformers initialized from pre-trained BERT and its variants further improves their performance. Finally, THSA can be combined with other kernel-based efficient Transformers to obtain linear computational and memory complexities regarding the sequence length.

Datasets, tasks and evaluation metrics. We evaluate Tuformers on 7 datasets for 5 tasks: word-level Language Modeling (LM) on Penn Treebank (PTB) (Marcus et al., 1993), Neural Machine Translation (NMT) on WMT16 'English-German' (Sennrich et al., 2016), Automatic Speech Recognition (ASR) on LibriSpeech (Panayotov et al., 2015), Natural Language Inference (NLI) on MNLI (Williams et al., 2018) and QNLI (Wang et al., 2018) and Image Generation on CIFAR10 (Krizhevsky, 2009) and MNIST (Deng, 2012) datasets.

Baselines. We compare the performance of MHSA module with THSA module in the following 5 backbone architectures: **(1)** a vanilla Transformer model (Vaswani et al., 2017), **(2)** an TransformerXL model (Dai et al., 2019), **(3)** a BERT-large model (Devlin et al., 2019), **(4)** a RoBERTa-large model (Liu et al., 2019), and **(5)** an ALBERT model (Lan et al., 2019). In all baselines, we adopt all default settings used in the original paper (hyperparameters specified in Appendix E), and only replace/re-parameterize the MHSA to THSA.

Performance Results of Tuformers.
(1) THSA consistently outperforms MHSA on diverse datasets for a variety of tasks under different model scales. In LM, NMT and ASR tasks, we compare THSA against the MHSA models under different model scales. MHSA is parameterized by four weight matrices $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, $\boldsymbol{W}^V$ and $\boldsymbol{W}^O$ whereas THSA adds an additional core matrix $\boldsymbol{C}$ of the same size. Adopting the size of the four original weight matrices in MHSA to THSA incurs an 25% increase in the number of parameters. For a fair comparison, we first compare MHSA against THSA with the same number of parameters (thus with smaller query/key/value/output weight matrices). To get a full picture, we also compare MHSA against THSA with the same sized four original weight matrices. As shown in Figure 5, under the same number of parameters, our THSA (**blue** curve) universally outperforms MHSA (**red** curve) under all model scales. By increasing the number of parameters of the attention units by 25% [1] (i.e., adding the core matrix $\boldsymbol{C}$ while keeping the size of the query/key/value/output weight matrices the same), our THSA (**purple** curve) further improves the performance universally under all model scales.

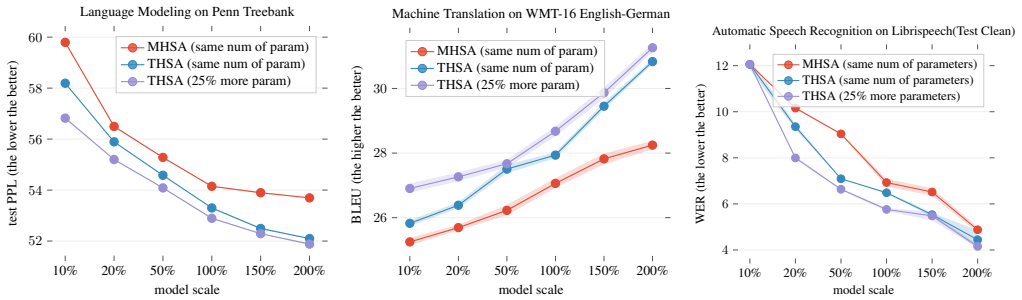


Figure 5: Performance comparison of THSA and MHSA in **(left)** language modeling on Penn Treebank dataset on TransformerXL, **(middle)** neural machine translation on WMT16 English-German dataset and **(right)** automatic speech recognition on Librispeech on Transformers. Under the same number of parameters, our THSA universally outperforms MHSA under all model sizes. By increasing the number of parameters of the attention units parameter by 25% (i.e., adding the core matrix $\boldsymbol{C}$ while keeping the size of the query/key/value weight matrices the same), our THSA further improves the performance universally under all model scales. The error bars displayed in shades are generated using 10 random runs in LM and NMT and 3 random runs in ASR.

[1] As will be discussed later, the overall model size is not drastically increased.

(2) Tuformers initialized with pre-trained strong models further improve the performance. In NLI task, we initialize Tuformers with pre-trained BERT-large, RoBERTa-large and ALBERT models, and perform the same fine-tuning tasks. As shown in Table 1 **(Left)**, THSA is compatible with existing pre-trained Transformer models and there is no need to train Tuformers from scratch: fine-tuning THSA initialized from pre-trained BERT and its variants further improves their performance.

(3) Combing THSA with kernels to obtain linear computation and memory complexities in sequence length. Although an effective design with guaranteed higher expressive power, Tuformers come with an extra memory overhead when calculating values within the softmax. In Tuformers, if done naively, the memory overhead is MNR where N, M denote the sequence length and $R = DH$ is the rank of the core tensor, incurring a D times larger memory overhead than vanilla Transformers in calculating softmax. Combining Tuformers with some state-of-the-art kernel-based efficient Transformers such as Linear Transformers using ELU (Katharopoulos et al., 2020) and polynomial (Tsai et al., 2019) kernels, we reduce the computation and memory complexities of Tuformers to be linear in the sequence length. Kernel-based Transformers apply kernel methods to MHSA to remove the nonlinearity constrain in the softmax function, obtaining linear computation and memory complexities. For example, in Linear Transformer ($N = M$), the space complexity is $N \times D \times H$. Applying the same idea to THSA leads to the same space complexity. We experiment on image generation tasks on CIFAR10 (Krizhevsky, 2009) and MNIST (Deng, 2012) datasets as in Table 1**(Right)** and show that Tuformers obtain strong performance and linear space complexity when combined with kernels.

Table 1: (Left) THSA initialized with pre-trained models. Comparison of accuracies obtained for language inference task on MNLI and QNLI datasets, the higher the better. Results show that Tuformers can be conveniently initialized with pre-trained models for downstream tasks, and fine-tuning obtains impressive performance. **(Right) THSA +kernels.** Comparison of bits per dimension (BPD) for image generation task on MNIST and CIFAR-10 datasets, the lower the better. Results show that Tuformers can be extended to image generation tasks and improve the performance of other efficient designs with linear computational and memory complexities.

Models	MNLI	QNLI
BERT (Devlin et al., 2019)	84.1%	92.1%
BERT + THSA	**84.9%**	**92.8%**
RoBERTa (Liu et al., 2019)	89.4%	94.3%
RoBERTa + THSA	**90.2%**	**94.7%**
ALBERT (Lan et al., 2019)	90.8%	95.3%
ALBERT + THSA	**91.7%**	**96.1%**

Models	MNIST	CIFAR 10
ELU kernel (Katharopoulos et al., 2020)	0.72	3.51
ELU kernel + THSA	**0.65**	**3.42**
Polynomial kernel (Tsai et al., 2019)	0.64	3.47
Polynomial kernel THSA	**0.57**	**3.37**

Parameter efficiency of Tuformers. THSA introduces an additional trainable core matrix compared with MHSA. However, THSA does not increase the complexity in big O notation compared with MHSA. Moreover, in Transformer models, attention modules are not the only source of parameters and a 25% increase in MHSA often does not increase the total number of parameters significantly. For instance, if we replace MHSA with THSA by keeping the size of the query/key/value/output weight matrices to be the same, a Transformer encoder with $1.3M$ parameters will be transferred to a Tuformer with $1.6M$ parameters. Finally, we empirically verify that the core matrix design still maintains parameter efficiency. In Figure 6, we benchmark the training time of Tuformers in multiple tasks and show that THSA with 25% more number of parameters (**blue** curve) is slightly slower to train compared with MHSA (**red** curve) but significantly improves the performance (as shown in Figure 5). In addition, as shown previously in Figure 5**(middle)**, using 10% parameters of a vanilla Transformer, a Tuformer obtains a 25.8 BLEU score whereas the original vanilla Transformer has a BLEU score of 27.1 (95% of the performance), demonstrating competitive parameter efficiency.

Ablation study: check-pointing to alleviate memory overhead without kernels. Although combining THSA with kernel-based efficient Transformers relieves the memory overhead as shown above, we investigate other solutions to alleviate the D times larger memory overhead compared with vanilla Transformers in calculating softmax when not using kernels. Specifically, we use check-pointing to eliminate the memory overhead in THSA. Since no intermediate results are saved in the forward pass using check-pointing, we recalculate the intermediate results of the softmax box in the back-propagation, which introduces some computational overhead. Figure 6 (**purple** curve) shows the per iteration training time overhead induced by check-pointing. From Figure 6 we see that applying check-pointing to Tuformers leads to only a slight increase in training time while relieving the memory overhead. We can reduce or even avoid the repetitive calculations by developing a proper

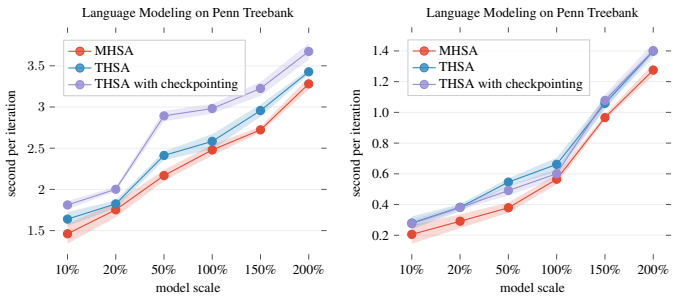


Figure 6: **(Left)** Training and **(Right)** inference run-time comparison between MHSA, THSA with the same number of parameters and THSA with the same number of parameters with checkpointing. The batch size is 256. The comparison of the NMT task, when THSA has the same sized query/key/value/outputmatrices as MHSA,the can be found in E.

backward module, although such development is beyond the scope of this paper and deferred for future works.

5 Related Works

Analysis of MHSA. Given the popularity of Transformers in a wide range of domains, a line of works focuses on understanding and improving MHSA. Voita et al. (2019) proposed a practical scheme to prune less informative heads. Michel et al. (2019) also showed that most attention heads are removable in testing without much performance compromise. Cordonnier et al. (2020) found empirical evidence of redundancy in key/query projections and proposed a re-parameterization scheme. These works bring valuable insights into MHSA, but a consistent theoretical analysis for these findings is missing. Our work differs from these works in that we focus on the theoretical understanding of MHSA in terms of expressive power, and we provide theoretical guarantees for a novel design, Tuformers, with higher expressive power and data-driven trainable heads.

Tensor theories and the expressive power of neural networks. There are a long line of works applying tensor methods in modern machine learning. Most of them focus on neural network compression using the tensor decomposition methods as (Ma et al., 2019; Su et al., 2018; Rabanser et al., 2017). There are also works analyzing the expressive power in neural networks using tensor representation theories (Cohen et al., 2016; Khrulkov et al., 2018). Cordonnier et al. (2019) used expressive power in comparing the relationship between the multi-head self-attention and convolutional layers. Yun et al. (2019) established that Transformer models are universal approximators of continuous permutation equivariant sequence-to-sequence functions with compact support. In our work, we represent MHSA in tensor diagram and form a design space where we compare the expressive power of different tensor structures among the weight matrices in MHSA.

Efficient Transformers. MHSA has a global receptive field, i.e., each token's representation is updated by attending to all other tokens, therefore incurring a quadratic memory and computation complexities concerning the sequence length. An extensive line of works focusing on reliving such the dependencies on the sequence length, such as Performer (Choromanski et al., 2020), Reformer (Kitaev et al., 2020), Linformer (Wang et al., 2020) and Linear Transformer (Katharopoulos et al., 2020). Our proposed model, Tuformers, can be incorporated into some kernel-based efficient Transformers, which applies kernel-based methods to MHSA to remove the nonlinearity constraint brought by the softmax function, achieving the same state-of-the-art linear computation and memory complexities in the sequence length.

6 Conclusion

This paper introduces a mathematically rigorous yet intuitive tensor diagram representation of MHSA, formulates a design space where we can analyze the expressive power of MHSA and its variants and proposes Tuformers, a novel model design with a guaranteed higher expressive power. Furthermore, Tuformers have a data-driven structure where heads can be trained implicitly and initialized with pre-trained Transformer models. The introduction of the tensor diagram representations and the theory presented in this paper provide new tools and open new directions for future research searching for expressive and efficient architectures.

ETHICS STATEMENT

Transformers are powerful neural network structures that achieve remarkable results in many domains. Understanding the expressive power of such neural networks is desirable but difficult. In this paper, we introduce tensor diagram representation of MHSA (the core component of Transformers) and formulate a design space where we can analyze the expressive power from a tensor perspective. Guided by the tensor diagram representation, we propose a plug-and-play THSA module to replace MHSA with THSA, obtaining Tuformers. Tuformers are theoretically guaranteed to have high expressive power than Transformers.

Our work benefits the Machine Learning community in following points:

(1) We introduce tensor diagram representation to the ML community as an analytical tool for understanding expressive power of neural networks. Equipping Transformer models with such a precise and yet intuitive tool, we are able to close the gap between the currently used simple but intuitive flowcharts and the precise but complex code/mathematical formulation.

(2) By tensorizing MHSA, we provide new possibilities for network analysis from a tensor representation perspective. We formulate a design space for MHSA, which can foster future studies for architecture design.

(3) We explore the theory behind MHSA and propose Tuformers with THSA. The theoretical framework we provide can lead to future explorations and developments of expressive and yet efficient models.

(4) Our proposed Tuformers are energy and resource efficient. (a) Tuformers can be initialized with pre-trained strong models such as BERT, RoBERTa and ALBERT models, drastically reducing the training cost for new downstream tasks. (b) THSA is a plug-and-play module that can be paired with ELU or polynomial kernels to obtain linear computation and memory efficiency in sequence length.

REPRODUCIBILITY STATEMENT

For theoretical results, full details are given in Appendix D.

For experimental results, firstly, all the datasets used in this paper are public and contain no sensitive information. Then a detailed description of the datasets, tasks, the hyper-parameters and how the are chosen, and train/test data splitting are in Appendix E. The main code can be found in the supplementary materials.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2020.

Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on learning theory*, pp. 698–728. PMLR, 2016.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*, 2019.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. Multi-head attention: Collaborate instead of concatenate. *arXiv preprint arXiv:2006.16362*, 2020.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, 2019.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 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)*, pp. 4171–4186, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Valentin Khrulkov, Alexander Novikov, and Ivan Oseledets. Expressive power of recurrent neural networks. In *International Conference on Learning Representations*, 2018.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Jerry Ma and Denis Yarats. On the adequacy of untuned warmup for adaptive optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. A tensorized transformer for language modeling. *Advances in Neural Information Processing Systems*, 32:2232–2242, 2019.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in Neural Information Processing Systems*, 32:14014–14024, 2019.

Tomas Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

Niko Moritz, Takaaki Hori, and Jonathan Le. Streaming automatic speech recognition with the transformer model. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6074–6078. IEEE, 2020.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 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)*, pp. 48–53, 2019.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206–5210. IEEE, 2015.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pp. 4055–4064. PMLR, 2018.

Roger Penrose. Applications of negative dimensional tensors. *Combinatorial mathematics and its applications*, 1:221–244, 1971.

Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann. Introduction to tensor decompositions and their applications in machine learning. *arXiv preprint arXiv:1711.10781*, 2017.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pp. 371–376, 2016.

Jiahao Su, Jingling Li, Bobby Bhattacharjee, and Furong Huang. Tensorial neural networks: Generalization of neural networks and application to model compression. *arXiv preprint arXiv:1805.10352*, 2018.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/N18-1101.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2019.

Appendix of Tuformer: Data-driven Design of Expressive Transformers with Learnable Weight Sharing Across Heads

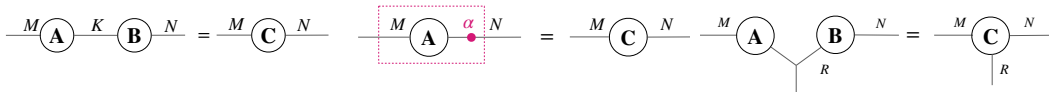
A Supplementary Material for Tensor Diagrams

In this section, we provide a comprehensive introduction to the tensor diagram representation.

A.1 Components of Tensor Diagram: Arrays as Nodes with Leg(s)

Arrays denoted as Nodes with Legs. An array is represented as a node with leg(s) in a tensor diagram as shown in Figure 1. The *order* (the number of dimensions) of the array is denoted by the number of legs extending from the node. The legs do not need to be straight lines ,and the orientation does not matter. For example, matrices $\boldsymbol{A}$ and $\boldsymbol{A}^\top$ are equivalent in this notation. Each labeled leg represents one *mode* of a tensor. Every mode of the tensor needs to be uniquely labeled. We usually use the dimension of the mode as the label (i.e., an associated positive integer written on top of each leg). If multiple modes of the same tensor have the same length, we use subscripts to differentiate them.

A.2 Operations in Tensor Diagram



$$\sum_{k=1}^{K} \boldsymbol{A}_{mk}\boldsymbol{B}_{kn} = \boldsymbol{C}_{mn} \qquad \frac{\exp(\alpha \cdot \boldsymbol{A}_{mn})}{\sum_{n'=1}^{N} \exp(\alpha \cdot \boldsymbol{A}_{mn'})} = \boldsymbol{C}_{mn} \qquad \boldsymbol{A}_{nr}\boldsymbol{B}_{mr} = \boldsymbol{\mathcal{C}}_{nmr}$$

Figure 7: (1) Contraction as shown in Appendix A.2 is an operation that generalizes the matrix multiplication. It does summation on element-wise products along a mode in object 1 and a corresponding mode (with the same dimension) in object 2 (i.e., in tensor diagram language, along a leg in node 1 and a corresponding leg in node 2). **In tensor diagram, multiplying two matrices (or higher-order tensors with more than 2 legs) corresponds to "gluing" their corresponding legs (along a certain mode). (2) Softmax** as shown in Appendix A.2 is an element-wise exponential function normalized along a certain mode. Tensor diagram has a convention to denote the contraction, but not the softmax. We propose to denote the α**-scaled softmax** function $\mathsf{softmax}(\alpha\boldsymbol{A})$ on $\boldsymbol{A}$ as **a dotted box with a labeled filled ball** (to distinguish itself from tensor objects, i.e., nodes which are blank circles) attached to one leg. **(3) Batch Multiplication**, as shown in appendix A.2, is an elementwise product along the connected legs.

There are three types of operations:*contraction*, *softmax* and *batch multiplication*. *Contraction* is an operation that does summation on element-wise products along a mode in object 1 and a corresponding mode (with the same dimension) in object 2 (i.e., in tensor diagram language, along a leg in node 1 and a corresponding leg in node 2), whereas *softmax* is an element-wise exponential function normalized along a certain mode. Tensor diagram has a convention to denote the contraction, but not the softmax. We will first introduce contraction in tensor diagram notation and then propose our design of softmax in tensor diagram language.

Contractions denoted as edges connecting the node legs. In tensor diagram, multiplying two matrices (or higher-order tensors with more than 2 legs) corresponds to "gluing" their corresponding legs (along a certain mode) as shown in Appendix A.2, and it is called *tensor contraction*. Since the representation of arrays in the tensor diagram is orientation invariant, contractions are also orientation invariant. This is especially useful in representing the self-attention unit. In some cases, the data object are represented as row vectors while in some other scenarios they are denoted as column vectors. If we use mathematical formula, we need to take care of the ordering of the matrix multiplication because $\boldsymbol{A}^\top\boldsymbol{B}, \boldsymbol{A}\boldsymbol{B}^\top, \boldsymbol{B}\boldsymbol{A}^\top, \boldsymbol{B}^\top\boldsymbol{A}$ and more are all different. In contrast, tensor diagram provides a universal graphical representation as long as the corresponding legs are connected correctly.

Nonlinear activation denoted as dotted box with a labelled filled ball attached. We propose to denote the α-scaled softmax function $\text{softmax}(\alpha \boldsymbol{A})$ on $\boldsymbol{A}$ as a dotted box with a labeled filled ball (to distinguish itself from tensor objects, i.e., nodes which are blank circles) attached to one leg. The label right above the filled ball denote the scaling parameter α, and the leg that the ball is attached to indicate the mode where normalization is implemented along in the softmax operation. As shown in Appendix A.2, let $\boldsymbol{A}$ denote the resultant of some operations (for example $\boldsymbol{Q}_{[h]}\boldsymbol{K}_{[h]}^\top$), the $\frac{1}{\sqrt{D}}$-scaled softmax on $\boldsymbol{A}$ which normalize along leg N is depicted. Softmax does not change the shape of the input. That is why we introduce a dotted box, which does not change the shape either, to denote it.

Batch Multiplication Given $\boldsymbol{A}$ and $\boldsymbol{B}$, the batch multiplication is an element-wise product along the connected legs. The resultant tensor $\boldsymbol{C}$ is obtained by merging connected nodes, maintaining the dangling legs M, N and R. Since in MHSA, the attention calculation among key, query and value matrices are done separately within each head, batch multiplication is a proper way to denote these "head-wise" operations.

A.3 Advantages of Tensor Diagram

Tensor diagram representations enjoy a few advantages compared with using flow charts together with math formulas.

(1) The notation is orientation invariant. The input embedding matrix $\boldsymbol{X}$ can be represented as either $\mathbb{R}^{N\times F}$ or $\mathbb{R}^{F\times N}$ with rows or columns being the embedding of each token in the sequence respectively. Although we take the former convention in our mathematical notations in Equation B.1, there are papers that use the latter, creating discrepancies between notations. However, in tensor diagram, since the node legs are orientation invariant, we obtain a universal graphical representation of multi-head self-attention, irrespective of how the input embedding matrix is represented, as shown in Figure 8c.

(2) The representation is both precise in math and intuitive, making analysis easier. The tensor diagram itself is mathematically rigorous.

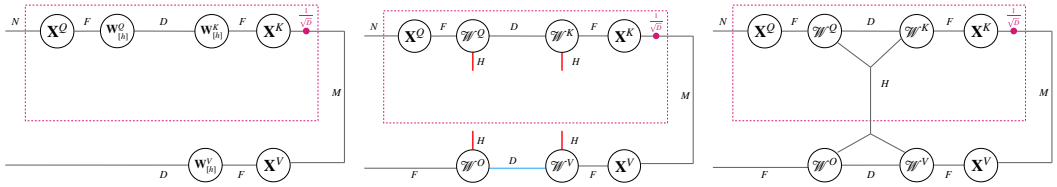
With the labels of the legs, the model complexity (number of parameters) is explicitly displayed, requiring no supplementary information such as the size of the parameters $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$ and $\boldsymbol{W}^V$ as in mathematical formulas. In addition,the resultant from an operation or a sequence of operations (i.e., a connected sub-graph of the entire tensor diagram graph) can be treated as "merging" any connected nodes in the sub-graph. This is particularly convenient to obtain the size of the resultant: any the dangling leg becomes a leg of the resultant.

(3) Multi-linear operations are concisely represented and interpreted. Using tensor diagram, we can concisely represent the multi-head self-attention rigorously in one diagram as shown in Figure 8c. More important, the multi-linear interaction between the latent features $\boldsymbol{Q}_{[h]}$, $\boldsymbol{K}_{[h]}$ and $\boldsymbol{V}_{[h]}$, which essentially attribute to the multi-linear interaction between the weight tensors $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$ and $\boldsymbol{W}^B$, is clearly illustrated in the tensor diagram. This illustration, which is not achieved by any existing representations, is crucial for our principled understanding of the multi-head self-attention.

(4) The information flow is clear. For example, in Figure 8a there is no direct information pass from $\boldsymbol{W}_{[h]}^K$, $\boldsymbol{W}_{[h]}^Q$ to $\boldsymbol{W}_{[h]}^V$ while in Figure 8c, such exchange is done through edge Ds and Hs. It is such observation (there are multiple paths that connect the weight parameter matrices) that leads to our idea of comparing expressive power between different structures.

B SUPPLEMENTARY MATERIAL FOR MULTI-HEAD SELF-ATTENTION

In the section, we prove the equivalence between the tensor diagram and the equations for *multi-head self-attention* (MHSA). To establish such an equivalence, we will need to show that both tensor diagram and the matrix equations lead to the same result in element-wise notation.



(a) Single-Head Self-Attention **(b)** Single-Head to Multi-Head **(c)** Multi-head self-attention

Figure 8: Tensor diagrams from single-head self-attention to multi-head self-attention. Figure 8a is the tensor diagram representation of a single-head self-attention. Figure 8c is the tensor-diagram representation of a multi-head self-attention.

For convenience of reference, we recap both representations in Equation (B.1) and Figure 8.

$$\boldsymbol{Q}_{[h]} = \boldsymbol{X}^Q \boldsymbol{W}^Q_{[h]};\ \boldsymbol{K}_{[h]} = \boldsymbol{X}^K \boldsymbol{W}^K_{[h]};\ \boldsymbol{V}_{[h]} = \boldsymbol{X}^V \boldsymbol{W}^V_{[h]}. \tag{B.1a}$$

$$\mathbf{head}_{[h]} = \mathsf{softmax}\left(\boldsymbol{Q}_{[h]} \boldsymbol{K}^\top_{[h]} / \sqrt{D}\right) \boldsymbol{V}_{[h]}, \tag{B.1b}$$

$$\boldsymbol{M} = \left[\mathbf{head}_{[1]}, \mathbf{head}_{[2]},\ \cdots,\ \mathbf{head}_{[H]}\right] \boldsymbol{W}^O, \tag{B.1c}$$

Note that the query weight tensor $\boldsymbol{\mathcal{W}}^Q$ comes from a concatenation of H query weight matrices $\{\boldsymbol{W}^Q_{[h]}\}_{h=1}^H$ such that $\boldsymbol{\mathcal{W}}^Q_{h,:,:} = \boldsymbol{W}^Q_{[h]}$. Similarly, $\boldsymbol{\mathcal{W}}^K$, $\boldsymbol{\mathcal{W}}^V$ are concatenated from $\{\boldsymbol{W}^K_{[h]}\}_{h=1}^H$, $\{\boldsymbol{W}_V[h]\}_{h=1}^H$ such that $\boldsymbol{\mathcal{W}}^K_{h,:,:} = \boldsymbol{W}^K_{[h]}, \boldsymbol{\mathcal{W}}^V_{h,:,:} = \boldsymbol{W}^V_{[h]}$. Furthermore, we denote $\boldsymbol{\mathcal{Q}}$, $\boldsymbol{\mathcal{K}}$, $\boldsymbol{\mathcal{V}}$ as the reshaped tensors from $\{\boldsymbol{Q}_{[h]}\}_{h=1}^H$, $\{\boldsymbol{K}_{[h]}\}_{h=1}^H$, $\{\boldsymbol{V}_{[h]}\}_{h=1}^H$. Finally, $\boldsymbol{\mathcal{W}}^O$ is reshaped from $\boldsymbol{W}^O$.

To differentiate between equations and tensor diagrams in the proof, we add an over-line to every object in the tensor diagram throughout this section.

B.1 SINGLE-HEAD SELF-ATTENTION (SHSA)

We will prove that the tensor diagram in Figure 8a is equivalent to Equations (B.1a) and (B.1b).

Proof. For simplicity, we omit the subscript $[h]$ in this subsection.

We first show that a contraction of the nodes within the softmax box is equivalent to $\boldsymbol{Q}\boldsymbol{K}^\top$. Let $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{K}^\top = \boldsymbol{X}^Q \boldsymbol{W}^Q (\boldsymbol{W}^K \boldsymbol{X}^K)^\top$, where $\boldsymbol{X}^Q \in \mathbb{R}^{N\times F}$, $\boldsymbol{X}^K \in \mathbb{R}^{M\times F}$, $\boldsymbol{W}^Q, \boldsymbol{W}^K \in \mathbb{R}^{F\times D}$. We use subscripts in the element-wise notation to differentiate edges when multiple edges with same dimension appear in the same equation.

$$\boldsymbol{Q}_{nd} = \sum_{f_1=1}^{F} \boldsymbol{X}^Q_{nf_1} \boldsymbol{W}^Q_{f_1 d}, \tag{B.2a}$$

$$\boldsymbol{K}_{md} = \sum_{f_2=1}^{F} \boldsymbol{X}^K_{mf_2} \boldsymbol{W}^K_{f_2 d}, \tag{B.2b}$$

$$\boldsymbol{A}_{nm} = \sum_{d=1}^{D} \boldsymbol{Q}_{nd} \boldsymbol{K}_{dm} = \sum_{d=1}^{D} \sum_{f_1=1}^{F} \sum_{f_2=1}^{F} \boldsymbol{X}^Q_{nf_1} \boldsymbol{W}^Q_{f_1 d} \boldsymbol{W}^K_{f_2 d} \boldsymbol{X}^K_{mf_2}. \tag{B.2c}$$

Denote be the result of the subset of nodes in the softmax box as $\boldsymbol{A}$.

$$\overline{\boldsymbol{A}}_{nm} = \sum_{d=1}^{D} \sum_{f_1=1}^{F} \sum_{f_2=1}^{F} \overline{\boldsymbol{X}}^Q_{nf_1} \overline{\boldsymbol{W}}^Q_{f_1 d} \overline{\boldsymbol{W}}^K_{f_2 d} \overline{\boldsymbol{X}}^K_{mf_2}. \tag{B.3}$$

Comparing Equations (B.2c) and (B.3), we have $\boldsymbol{A} = \overline{\boldsymbol{A}}$. Let $\boldsymbol{B} = \mathsf{softmax}(\boldsymbol{A})$ and $\overline{\boldsymbol{B}} = \mathsf{softmax}(\overline{\boldsymbol{A}})$, we further have $\boldsymbol{B} = \mathsf{softmax}(\boldsymbol{A}) = \mathsf{softmax}(\overline{\boldsymbol{A}}) = \overline{\boldsymbol{B}}$.

Let $\overline{\boldsymbol{M}}$ be the result of the tensor diagram and $\boldsymbol{M}$ be the matrix representation of the single-head attention. We have

$$\boldsymbol{M}_{nd} = \sum_{m=1}^{N} \sum_{f=1}^{F} \boldsymbol{B}_{nm} \boldsymbol{X}^{V}_{mf} \boldsymbol{W}^{V}_{fd}. \tag{B.4}$$

$$\overline{\boldsymbol{M}}_{nd} = \sum_{m=1}^{N} \sum_{f=1}^{F} \overline{\boldsymbol{B}}_{nm} \overline{\boldsymbol{X}}^{V}_{mf} \overline{\boldsymbol{W}}^{V}_{fd}. \tag{B.5}$$

Comparing Equations (B.4) and (B.5), we have $\boldsymbol{C}$ is equivalent to $\overline{\boldsymbol{C}}$, which completes the proof. □

B.2 Multi-head Self-Attention (MHSA)

We now prove that tensor diagram in Figure 8c is equivalent to Equation (1c).

Proof. Let H be the number of heads. Let $\overline{\boldsymbol{\mathcal{A}}} \in \mathbb{R}^{H \times N \times M}$ be the result of a contraction of the nodes within the softmax box. From the last section, we know that $\overline{\boldsymbol{\mathcal{A}}}_{h,:,:} = \boldsymbol{A}_{[h]} = \boldsymbol{Q}_{[h]} \boldsymbol{K}^{\top}_{[h]}$. Applying the softmax function $\overline{\boldsymbol{\mathcal{B}}} = \mathsf{softmax}(\overline{\boldsymbol{\mathcal{A}}})$, we have $\overline{\boldsymbol{\mathcal{B}}}_{h,:,:} = \mathsf{softmax}(\boldsymbol{Q}_{[h]} \boldsymbol{K}^{\top}_{[h]})$.

Let $\overline{\boldsymbol{T}}$ be result of the tensor diagram and $\boldsymbol{T}$ be that of Equation (B.1c).

$$\boldsymbol{T}_{nf_4} = \sum_{m,h,d_1,d_2} \mathsf{softmax}(\boldsymbol{Q}_{[h],nd_1} \boldsymbol{Q}_{[h],md_1}) \boldsymbol{V}_{[h],md_2} \boldsymbol{W}^{O}_{[h],d_2 f_4}. \tag{B.6}$$

$$\overline{\boldsymbol{T}}_{n1f_4} = \sum_{m,h,d_1,d_2} \mathsf{softmax}(\overline{\boldsymbol{\mathcal{Q}}}_{hnd_1} \overline{\boldsymbol{\mathcal{K}}}_{hmd_1}) \overline{\boldsymbol{\mathcal{V}}}_{hmd_2} \overline{\boldsymbol{\mathcal{W}}}^{O}_{hmd_2}. \tag{B.7}$$

Comparing $\overline{\boldsymbol{T}}$ and $\boldsymbol{T}$, we complete the proof.

□

C Closer Investigation of MHSA and its Comparison to THSA

Rethinking MHSA. There are two types contractions in MHSA as shown in Figure 4a: *Head-Contraction*, a global contraction among all weights $\boldsymbol{\mathcal{W}}^Q$, $\boldsymbol{\mathcal{W}}^K$, $\boldsymbol{\mathcal{W}}^V$ and $\boldsymbol{\mathcal{W}}^O$ along mode H, and *Latent-Contraction*, a local contraction between $\boldsymbol{\mathcal{W}}^Q$ and $\boldsymbol{\mathcal{W}}^K$ as well as between $\boldsymbol{\mathcal{W}}^V$ and $\boldsymbol{\mathcal{W}}^O$ both along mode D. If we remove the Head-Contraction and maintain the Latent-Contraction only, multi-head is reduced to single-head with an additional contraction with the weight matrix $\boldsymbol{\mathcal{W}}^O$. To understand the role of the two types of contractions, we propose ablation studies by removing one of them and study the changes in the structure and the expressive power. It turns out that in both situation, *MHSA falls into a special case of THSA*.

Repeated Proposition 1 with two special cases. *Given $R = DH$ and other hyperparameters being the same, THSA module includes MHSA as a special case. Specifically,*
(1) *If $\boldsymbol{C} = \mathbf{1}_D \mathbf{1}_D^{\top}$, i.e., an $D \times D$ all-ones matrix, the THSA module reduces to an MHSA with a single head and latent dimension D (i.e., a single-head self-attention with latent dimension D).*
(2) *If $\boldsymbol{C} = \boldsymbol{I}_H$, i.e., an $H \times H$ identity matrix, the THSA reduces to a MHSA with H heads and latent dimension 1 (i.e., a heads-only self-attention with H heads).*
(3) *If $\boldsymbol{C} = \boldsymbol{I}_H \otimes (\mathbf{1}_D \mathbf{1}_D^{\top})$, i.e., a Kronecker product of an all-ones matrix $\mathbf{1}_D \mathbf{1}_D^{\top} \in \mathbb{R}^{D \times D}$ and an identity matrix $\boldsymbol{I}_H \in \mathbb{R}^{H \times H}$, the THSA reduces to an MHSA with H heads and latent dimension D.*

The proof of the above propositions is given in Appendix D.

Definition 4 (Stable Rank)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{N \times M}$, the stable rank of $\boldsymbol{A}$ of is defined as $\sum_i \sigma_i^2 / \max_i \sigma_i$ where σ_i is the singular value of $\boldsymbol{A}$.*

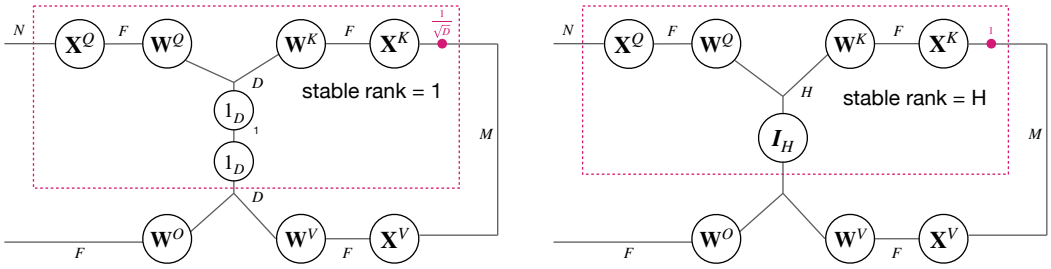


Figure 9: (Left) THSA includes a single-head self-attention. In this case, the stable rank of the core tensor $\boldsymbol{C}$ is 1. It is interesting to observe that this is a single-head self-attention and this structure can be obtained by removing the head contraction from Figure 4a. **(Right)** THSA includes a MHSA with H heads and latent dimension $= 1$. In this case, the stable rank equals to the rank of the identical matrix, which is H, corresponding to the number of heads of MHSA. Also, the structure on the right can be obtained by removing the latent contraction edges D from Figure 4a.

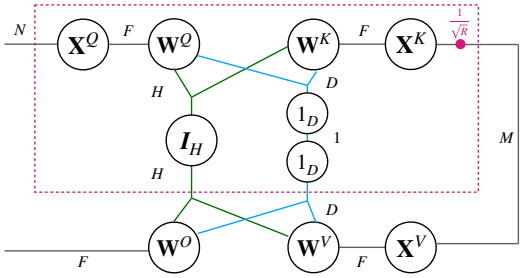


Figure 10: This figure shows how to initialize THSA with MHSA. MHSA is a special case of THSA when the core tensor $\boldsymbol{C}$ takes the Kronecker product of a $D \times D$ all-one matrix and a $H \times H$ identical matrix. Note that in this case, the number of heads H in MHSA is equivalent to the stable rank of $\boldsymbol{C}$.

The notion of heads in THSA. The concept of the number of heads in THSA generalizes to the stable rank of the core matrix, allowing a data-driven implicit training. The number of heads H in MHSA corresponds to the *stable rank* of the core matrix $\boldsymbol{C}$, defined as $\sum_i \sigma_i^2 / \max_i \sigma_i$ where σ_i is the singular value. It is a useful surrogate for the rank because it is largely unaffected by tiny singular values.

As shown in Appendix C when $\boldsymbol{C}$ is constructed by the outer product of two vectors, the stable rank of $\boldsymbol{C}$ is 1, while in this case, the whole structure is equivalent to a single-head self-attention. When $\boldsymbol{C}$ is an identical matrix as shown in appendix C, the stable rank of $\boldsymbol{C}$ equals to the rank of the identical matrix, which is H, corresponding to the number of heads of MHSA. Also, the structure on the right can be obtained by removing the latent contraction edges D from Figure 4a. When $\boldsymbol{C}$ is the Kronecker product of the all-one matrix and the identical matrix, which corresponds to MHSA, the stable rank of $\boldsymbol{C}$ is H, and this number does not change in the training because that $\boldsymbol{C}$ is a fixed matrix in MHSA.

D Supplementary Material for Tuformer and Proofs

In this section, we will prove Proposition 1 and Theorem 3 in Section 3. In both theorems, we assume the rank in *Tunable-head self-attention* (THSA) equals to the product of the number of heads H and the latent dimension D in *multi-head self-attention* (MHSA), i.e., $R = DH$.

For convenience of reference, we recap the mathematical expressions and tensor diagrams for both multi-head self-attention (MHSA) and Tunable-head self-attention (THSA) in the following.

Multi-head Self-Attention (MHSA). An MHSA module has its learnable parameters as H sets of (query, key, value) weight matrices $\{\boldsymbol{W}^Q_{[h]}, \boldsymbol{W}^K_{[h]}, \boldsymbol{W}^V_{[h]}\}_{h=1}^H$ (with $\boldsymbol{W}^Q_{[h]}, \boldsymbol{W}^K_{[h]}, \boldsymbol{W}^V_{[h]} \in \mathbb{R}^{F\times D}$ for each h) and an output weight matrix $\boldsymbol{W}^O \in \mathbb{R}^{HD\times F}$. The module maps three input matrices $\boldsymbol{X}^Q \in \mathbb{R}^{N\times F}$, $\boldsymbol{X}^K, \boldsymbol{X}^V \in \mathbb{R}^{M\times F}$ into an output matrix $\boldsymbol{M} \in \mathbb{R}^{N\times F}$.

$$\boldsymbol{Q}_{[h]} = \boldsymbol{X}^Q \boldsymbol{W}^Q_{[h]};\ \boldsymbol{K}_{[h]} = \boldsymbol{X}^K \boldsymbol{W}^K_{[h]};\ \boldsymbol{V}_{[h]} = \boldsymbol{X}^V \boldsymbol{W}^V_{[h]}, \tag{D.1a}$$

$$\mathbf{head}_{[h]} = \mathsf{softmax}\left(\boldsymbol{Q}_{[h]}\boldsymbol{K}_{[h]}^\top\right)\boldsymbol{V}_{[h]}, \tag{D.1b}$$

$$\boldsymbol{M} = \left[\mathbf{head}_{[1]},\ \mathbf{head}_{[2]},\ \cdots,\ \mathbf{head}_{[H]}\right]\boldsymbol{W}^O. \tag{D.1c}$$

In Equation (D.1a), $\boldsymbol{Q}_{[h]} \in \mathbb{R}^{N\times D}$, $\boldsymbol{K}_{[h]}, \boldsymbol{V}_{[h]} \in \mathbb{R}^{M\times D}$ are query, key, value matrices respectively. In Equation (D.1b), we omit the scaling factor $1/\sqrt{D}$ for the $\mathsf{softmax}$ function (the scalar can be merged into the learnable parameters), and $\mathbf{head}_{[h]} \in \mathbb{R}^{N\times D}$ is the resulted matrix for head h.

Tunable-Head Self-Attention. A THSA module is parameterized by five matrices: a query weight matrix $\overline{\boldsymbol{W}}^Q \in \mathbb{R}^{F\times R}$, a key weight matrix $\overline{\boldsymbol{W}}^K \in \mathbb{R}^{F\times R}$, a value weight matrix $\overline{\boldsymbol{W}}^V \in \mathbb{R}^{F\times R}$, an output weight matrix $\overline{\boldsymbol{W}}^O \in \mathbb{R}^{R\times F}$, and an additional core matrix $\boldsymbol{C} \in \mathbb{R}^{R\times R}$. The module has the same input and output domains as in MHSA, i.e., it takes three matrices $\boldsymbol{X}^Q \in \mathbb{R}^{N\times F}$, $\boldsymbol{X}^K, \boldsymbol{X}^V \in \mathbb{R}^{M\times F}$ as inputs and returns a matrix $\boldsymbol{T} \in \mathbb{R}^{N\times F}$ as output.

$$\overline{\boldsymbol{Q}} = \boldsymbol{X}^Q \overline{\boldsymbol{W}}^Q;\ \overline{\boldsymbol{K}} = \boldsymbol{X}^K \overline{\boldsymbol{W}}^K;\ \overline{\boldsymbol{V}} = \boldsymbol{X}^V \overline{\boldsymbol{W}}^V, \tag{D.2a}$$

$$\overline{\mathbf{head}}_r = \mathsf{softmax}\left(\sum_{s=1}^{R} \boldsymbol{C}_{rs}\, \overline{\boldsymbol{Q}}_s \overline{\boldsymbol{K}}_s^\top\right)\overline{\boldsymbol{V}}_r, \tag{D.2b}$$

$$\boldsymbol{T} = \overline{\mathbf{head}}\ \overline{\boldsymbol{W}}^O = \left[\overline{\mathbf{head}}_1,\ \overline{\mathbf{head}}_2,\ \cdots,\ \overline{\mathbf{head}}_R\right]. \tag{D.2c}$$

To distinguish THSA from MHSA, we use over-scored symbols whenever needed. In Equation (D.2a), $\overline{\boldsymbol{Q}} \in \mathbb{R}^{N\times R}$, $\overline{\boldsymbol{K}}, \overline{\boldsymbol{V}} \in \mathbb{R}^{M\times R}$ are query, key, value matrices respectively. In Equation (D.2b), we again omit the scaling factor $1/\sqrt{R}$ in the $\mathsf{softmax}$ function. Equation (D.2b) leads to a head matrix $\overline{\mathbf{head}} \in \mathbb{R}^{F\times N}$, where $\overline{\mathbf{head}}_r \in \mathbb{R}^F$ denotes the rth column of the matrix $\overline{\mathbf{head}}$.

D.1 Proof of Proposition 1

Proof of Proposition 1. We constructively prove that THSA reduces to three cases of MHSA if the query/key/value weight matrices in THSA concatenate the corresponding matrices in MHSA:

$$\overline{\boldsymbol{W}}^Q = \left[\boldsymbol{W}^Q_{[1]},\ \boldsymbol{W}^Q_{[2]},\ \cdots,\ \boldsymbol{W}^Q_{[H]}\right], \tag{D.3a}$$

$$\overline{\boldsymbol{W}}^K = \left[\boldsymbol{W}^K_{[1]},\ \boldsymbol{W}^K_{[2]},\ \cdots,\ \boldsymbol{W}^K_{[H]}\right], \tag{D.3b}$$

$$\overline{\boldsymbol{W}}^V = \left[\boldsymbol{W}^V_{[1]},\ \boldsymbol{W}^V_{[2]},\ \cdots,\ \boldsymbol{W}^V_{[H]}\right]. \tag{D.3c}$$

In addition, we set $\overline{\boldsymbol{W}}^O = \boldsymbol{W}^O$. We have $\overline{\boldsymbol{W}}^Q, \overline{\boldsymbol{W}}^K, \overline{\boldsymbol{W}}^V \in R^{F\times R}$, and $\overline{\boldsymbol{W}}^O \in \mathbb{R}^{R\times F}$.

(1) For $\boldsymbol{C} = \mathbf{1}_R\mathbf{1}_R^\top$, i.e. an $R\times R$ all-ones matrix, we aim to prove that the THSA module reduces to a single-head self-attention with $H = 1$ and $D = R$. Since there is only one head in the module, we have $\overline{\boldsymbol{W}}^Q = \boldsymbol{W}^Q_{[1]}$, $\overline{\boldsymbol{W}}^K = \boldsymbol{W}^K_{[1]}$, and $\overline{\boldsymbol{W}}^V = \boldsymbol{W}^V_{[1]}$. As an immediate result,

$$\overline{\boldsymbol{Q}} = \boldsymbol{X}^Q \overline{\boldsymbol{W}}^Q = \boldsymbol{X}^Q \boldsymbol{W}^Q_{[1]} = \boldsymbol{Q}_{[1]}, \tag{D.4a}$$

$$\overline{\boldsymbol{K}} = \boldsymbol{X}^K \overline{\boldsymbol{W}}^K = \boldsymbol{X}^K \boldsymbol{W}^K_{[1]} = \boldsymbol{K}_{[1]}, \tag{D.4b}$$

$$\overline{\boldsymbol{V}} = \boldsymbol{X}^V \overline{\boldsymbol{W}}^V = \boldsymbol{X}^V \boldsymbol{W}^V_{[1]} = \boldsymbol{V}_{[1]}. \tag{D.4c}$$

Since $\boldsymbol{C}$ is an all-ones matrix, i.e., $\boldsymbol{C}_{rs} = 1, \forall r, s$, we rewrite Equation (D.2b) as:

$$\overline{\mathbf{head}}_r = \mathsf{softmax}\left(\sum_{s=1}^{R} \overline{\boldsymbol{Q}}_s \overline{\boldsymbol{K}}_s^\top\right)\overline{\boldsymbol{V}}_r = \mathsf{softmax}\left(\overline{\boldsymbol{Q}}\ \overline{\boldsymbol{K}}^\top\right)\overline{\boldsymbol{V}}_r. \tag{D.5}$$

Notice that the equation holds for each column r, we further write all R equations jointly as:

$$\overline{\textbf{head}} = \textsf{softmax}\left(\overline{\boldsymbol{Q}}\,\overline{\boldsymbol{K}}^\top\right)\overline{\boldsymbol{V}} = \textsf{softmax}\left(\boldsymbol{Q}_{[1]}\boldsymbol{K}_{[1]}^\top\right)\boldsymbol{V}_{[1]} = \textbf{head}_{[1]}. \tag{D.6}$$

Finally, we express the output matrix as:

$$\boldsymbol{T} = \overline{\textbf{head}}\,\overline{\boldsymbol{W}}^O = \textbf{head}_{[1]}\boldsymbol{W}^O = \boldsymbol{M}, \tag{D.7}$$

which concludes the reduction for the special case **(1)**.

(2) For $\boldsymbol{C} = \boldsymbol{I}_R$, an $R \times R$ identity matrix, we aim to prove that the THSA module reduces to a heads-only self-attention with $H = R$ and $D = 1$. Since the latent dimension $D = 1$, i.e., each $\boldsymbol{W}_{[h]}^Q$ (or $\boldsymbol{W}_{[h]}^K, \boldsymbol{W}_{[h]}^V$) is a vector, we have $\overline{\boldsymbol{W}}_r^Q = \boldsymbol{W}_{[r]}^Q$, $\overline{\boldsymbol{W}}_r^K = \boldsymbol{W}_{[r]}^K$, and $\overline{\boldsymbol{W}}_r^V = \boldsymbol{W}_{[r]}^V$. Therefore,

$$\overline{\boldsymbol{Q}}_r = \boldsymbol{X}^Q\overline{\boldsymbol{W}}_r^Q = \boldsymbol{X}^Q\boldsymbol{W}_{[r]}^Q = \boldsymbol{Q}_{[r]}, \tag{D.8a}$$

$$\overline{\boldsymbol{K}}_r = \boldsymbol{X}^K\overline{\boldsymbol{W}}_r^K = \boldsymbol{X}^K\boldsymbol{W}_{[r]}^K = \boldsymbol{K}_{[r]}, \tag{D.8b}$$

$$\overline{\boldsymbol{V}}_r = \boldsymbol{X}^V\overline{\boldsymbol{W}}_r^V = \boldsymbol{X}^V\boldsymbol{W}_{[r]}^V = \boldsymbol{V}_{[r]}. \tag{D.8c}$$

Since $\boldsymbol{C}$ is an identity matrix, i.e., $\boldsymbol{C}_{rr} = 1$ and $\boldsymbol{C}_{rs} = 0, \forall r \neq s$, we rewrite Equation (D.2b) as:

$$\overline{\textbf{head}}_r = \textsf{softmax}\left(\overline{\boldsymbol{Q}}_r\overline{\boldsymbol{K}}_r^\top\right)\overline{\boldsymbol{V}}_r = \textsf{softmax}\left(\boldsymbol{Q}_{[r]}\boldsymbol{K}_{[r]}^\top\right)\boldsymbol{V}_{[r]} = \textbf{head}_{[r]}. \tag{D.9}$$

Finally, we express the output matrix as:

$$\boldsymbol{T} = \overline{\textbf{head}}\,\overline{\boldsymbol{W}}^O = \left[\textbf{head}_{[1]},\ \textbf{head}_{[2]},\ \cdots;\ \textbf{head}_{[R]}\right]\boldsymbol{W}^O = \boldsymbol{M}, \tag{D.10}$$

which concludes the reduction for the special case **(2)**.

(3) For $\overline{\boldsymbol{C}} = \boldsymbol{I}_H \otimes (\boldsymbol{1}_D\boldsymbol{1}_D^\top)$, i.e., a Kronecker product between an identity matrix $\boldsymbol{I}_H \in \mathbb{R}^{H\times H}$ and an all-ones matrix $\boldsymbol{1}_D\boldsymbol{1}_D^\top \in \mathbb{R}^{D\times D}$, we aim to prove that the THSA module reduces to a multi-head self-attention (MHSA) module with H heads and latent dimension D. In this general case, we have

$$\overline{\boldsymbol{Q}} = \boldsymbol{X}^Q\overline{\boldsymbol{W}}^Q = \boldsymbol{X}^Q\left[\boldsymbol{W}_{[1]}^Q,\ \boldsymbol{W}_{[2]}^Q,\ \cdots,\ \boldsymbol{W}_{[H]}^Q\right] = \left[\boldsymbol{Q}_{[1]},\ \boldsymbol{Q}_{[2]},\ \cdots,\ \boldsymbol{Q}_{[H]}\right] \tag{D.11}$$

$$\overline{\boldsymbol{K}} = \boldsymbol{X}^K\overline{\boldsymbol{W}}^K = \boldsymbol{X}^K\left[\boldsymbol{W}_{[1]}^K,\ \boldsymbol{W}_{[2]}^K,\ \cdots,\ \boldsymbol{W}_{[H]}^K\right] = \left[\boldsymbol{K}_{[1]},\ \boldsymbol{K}_{[2]},\ \cdots,\ \boldsymbol{K}_{[H]}\right] \tag{D.12}$$

$$\overline{\boldsymbol{V}} = \boldsymbol{X}^V\overline{\boldsymbol{W}}^V = \boldsymbol{X}^V\left[\boldsymbol{W}_{[1]}^V,\ \boldsymbol{W}_{[2]}^V,\ \cdots,\ \boldsymbol{W}_{[H]}^V\right] = \left[\boldsymbol{V}_{[1]},\ \boldsymbol{V}_{[2]},\ \cdots,\ \boldsymbol{V}_{[H]}\right] \tag{D.13}$$

Notice that $\boldsymbol{C}$ is a block diagonal matrix $\textsf{blkdiag}(\boldsymbol{1}_D\boldsymbol{1}_D^\top, \cdots, \boldsymbol{1}_D\boldsymbol{1}_D^\top)$, where each block is an all-ones matrix. Therefore we can rewrite Equation (D.2b) for $r = (h-1)D + d$ as:

$$\overline{\textbf{head}}_{(h-1)D+d} = \textsf{softmax}\left(\sum_{s=1}^{R}\mathbb{1}_{(h-1)D<s\leq hD}\left(\overline{\boldsymbol{Q}}_s\overline{\boldsymbol{K}}_s^\top\right)\right)\overline{\boldsymbol{V}}_r \tag{D.14}$$

$$= \textsf{softmax}\left(\sum_{d=1}^{D}\boldsymbol{Q}_{[h],d}\boldsymbol{K}_{[h],d}^\top\right)\boldsymbol{V}_{[h],d} \tag{D.15}$$

$$= \textsf{softmax}\left(\boldsymbol{Q}_{[h]}\boldsymbol{K}_{[h]}^\top\right)\boldsymbol{V}_{[h],d}. \tag{D.16}$$

Since Equation (D.16) holds for each column d, we write all D equations jointly as:

$$\left[\overline{\textbf{head}}_{(h-1)D+d},\ \cdots,\ \overline{\textbf{head}}_{hD}\right] = \textsf{softmax}\left(\boldsymbol{Q}_{[h]}\boldsymbol{K}_{[h]}^\top\right)\boldsymbol{V}_{[h]} = \textbf{head}_{[h]}. \tag{D.17}$$

Again, since Equation (D.17) holds for each block h, we combine all H equations as:

$$\overline{\textbf{head}} = \left[\textbf{head}_{[1]},\ \textbf{head}_{[2]},\ \cdots,\ \textbf{head}_{[H]}\right]. \tag{D.18}$$

Finally, we express the output matrix as:

$$\boldsymbol{T} = \overline{\textbf{head}}\,\overline{\boldsymbol{W}}^O = \left[\textbf{head}_{[1]},\ \textbf{head}_{[2]},\ \cdots,\ \textbf{head}_{[H]}\right]\boldsymbol{W}^O = \boldsymbol{M}, \tag{D.19}$$

which concludes the reduction for the case **(3)**. □

D.2 PROOF OF THEOREM 3

Proof of Theorem 3. We have shown that THSA reduces to MHSA when $\boldsymbol{C}$ takes specific forms. This shows that THSA is more expressive than MHSA — any mapping by an MHSA module can also be realized by a THSA module.

To prove that THSA is *strictly* more expressive than MHSA, we need find a mapping by a THSA module that can not be realized by an MHSA module. It suffices to show that the matrix $\boldsymbol{M}$ is rank-deficient, while the matrix $\boldsymbol{T}$ can be full-rank. We assume $MH < \min(N, F)$.

For convenience, we divide the matrix $\boldsymbol{W}^O$ for MHSA into H blocks.

$$\boldsymbol{W}^O = \left[\boldsymbol{W}_{[1]}^{O\top}, \boldsymbol{W}_{[2]}^{O\top}, \cdots, \boldsymbol{W}_{[H]}^{O} \right]^\top, \tag{D.20}$$

where $\boldsymbol{W}_{[h]}^O \in \mathbb{R}^{D\times F}$ is the matrix for the hth head. We now rewrite the output matrix $\boldsymbol{M}$ as:

$$\boldsymbol{M} = \sum_{h=1}^{H} \underbrace{\mathsf{softmax}\left(\boldsymbol{Q}_{[h]}\boldsymbol{K}_{[h]}^\top\right)}_{\mathbb{R}^{N\times M}} \underbrace{\left(\boldsymbol{V}_{[h]}\boldsymbol{W}_{[h]}^O\right)}_{\mathbb{R}^{M\times F}}. \tag{D.21}$$

Since the matrix inside summation is a product of two matrices of size $\mathbb{R}^{N\times M}$ and $\mathbb{R}^{M\times F}$, its rank is at most M. Use the property $\mathsf{rank}(\boldsymbol{A}+\boldsymbol{B}) \le \mathsf{rank}(\boldsymbol{A}) + \mathsf{rank}(\boldsymbol{B})$, we have $\mathsf{rank}(\boldsymbol{M}) \le MH < \min(N, F)$. Similarly, we divide the matrix $\overline{\boldsymbol{W}}^O$ for THSA into R rows and rewrite the output matrix $\boldsymbol{T}$ as:

$$\boldsymbol{T} = \sum_{r=1}^{R} \underbrace{\mathsf{softmax}\left(\sum_{s=1}^{R} \boldsymbol{C}_{rs}\, \overline{\boldsymbol{Q}}_{:,r}\overline{\boldsymbol{K}}_{:,r}^\top\right)}_{\mathbb{R}^{N\times M}} \underbrace{\left(\overline{\boldsymbol{V}}_{:,r}\overline{\boldsymbol{W}}_{r,:}^O\right)}_{\mathbb{R}^{M\times F}}. \tag{D.22}$$

Using the same argument, we have $\mathsf{rank}(\boldsymbol{T}) \le MR$. Since $H < R$ (in fact, $R = DH$), we can always find an example such that $\mathsf{rank}(\boldsymbol{T}) > \mathsf{rank}(\boldsymbol{M})$. This completes the proof. □

E SUPPLEMENTARY MATERIAL FOR EXPERIMENTS

All the experiments are run on computing nodes with 4 NVIDIA GeForce GPUs.

(1) Language Modeling on Penn Treebank. Language modeling is the task of computing the probability of a sentence or a sequence of words. The model performance is measured by *per-word Perplexity (PPL)* which is the lower the better.

Penn Treebank (Marcus et al., 1993) is under the LDC User Agreement for Non-Members. We adopt the default settings as in the vanilla Transformer models(Vaswani et al., 2017). Specifically, we have N (length of the sequence) to be 2048, F (length of the initial embedding) to be 512, 8 number of heads, 6 layers encoder-decoder structure, D (latent feature dimension) to be 64. We use Adam optimizer and have 4000 warm-up steps. We also use label smoothing(0.1). The model is trained for 50 epochs.

(2) Neural Machine Translation on WMT16. The goal of NMT is to generate a corresponding sequence in one language given a sequence in another. The performance is measured by *BLEU* scores. (the higher, the better).

WMT16 English-German dataset (Sennrich et al., 2016) is under the MIT license. In this experiment, we use a Transformer model with 8 layers and we have $F = 512$ and $R = 1024$. The learning rate is $1e^{-5}$ and 4000 warmups. We also set label-smooting to be 0.1. To speed up the experiment, we use the mix-precision trick with an optimization level being O1.

(3) Image Generation on MNIST and CIFAR-10. The image generation task is to predict an image pixel by pixel. The performance is evaluated by *bits per dimension (BPD)*. The image generation task is chosen to demonstrate, firstly, that Tuformers work beyond the language domain, and secondly, Tuformers can improve other efficient designs.

CIFAR10 (Krizhevsky, 2009) dataset is under the MIT license (MIT), MNIST(LeCun et al., 2010) is under the Creative Commons Attribution-Share Alike 3.0 license. The reason why we choose to evaluate Tuformer in image generation tasks on these two datasets is that we want to show the compatibility of our model to other efficient designs. Katharopoulos et al. (2020) introduces a linear transformer that has a state-of-the-art complexity. As a result, we evaluate our design on the same experiments as they do in their paper. We adopt basically the same settings except for GPU resources. Concretely, we use a 8 layer transformer model with 8 heads. The embedding size F is 256 an D is 32. N is set to be 1024. We also use "RAdam" (Ma & Yarats, 2021) optimizer for a state-of-the-art result. The model is trained for 250 epochs for MNIST dataset and 50 epochs for CIFAR10 dataset. The batch size is 10 in MNIST training and 4 in CIFAR training. We additionally use he mix-precision trick with an optimization level set to be O1 to speed up the training.

(4) Automatic Speech Recognition on LibriSpeech Dataset ASR consists of transcribing audio speech segments into text. The performance is measured by *Word Error Rate (WER)*. (the lower, the better). We adopt the default settings as in Fairseq (Ott et al., 2019), except that we replace MHSA with THSA.

(5) Natural Language Inference on MNLI and QNLI. NLI is a task of determining whether the given "hypothesis" and "premise" logically follow (entailment) or unfollow (contradiction) or are undetermined (neutral) to each other. The results are evaluated by accuracy. Using MNLI dataset, the model tries to predict whether sentence A entails or contradicts B while in QNLI dataset, the model is trained to answer whether sentence B contains answers to the question in sentence A. We use a batch size of 32 for 10 epochs, with a learning rate $1e^{-5}$.

Additional Results

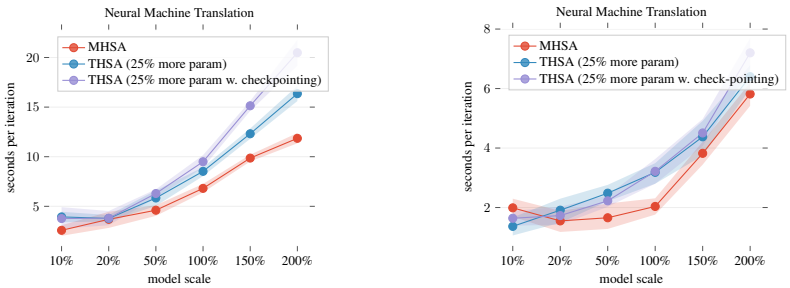


Figure 11: (Left) Training and **(Right)** inference run-time comparison between MHSA, THSA with same sized query/key/value/output matrices as MHSA with an additional core matrix (thus 25% more number of parameters in the attention units), and THSA 25% more parameters with check-pointing.