Understanding and Improving Transformer From a Multi-Particle Dynamic System Point of View

Yiping Lu[1,*] **Zhuohan Li**[1,*] **Di He**[1] **Zhiqing Sun**[1]
Bin Dong[1] **Tao Qin**[2] **Liwei Wang**[1] **Tie-Yan Liu**[2]
[1]Peking University [2]Microsoft Research
{luyiping9712,lizhouhan,di_he,zhiqing}@pku.edu.cn
dongbin@math.pku.edu.cn wanglw@pku.edu.cn
{taoqin, tyliu}@microsoft.com

Abstract

The Transformer architecture is widely used in natural language processing. Despite its success, the design principle of the Transformer remains elusive. In this paper, we provide a novel perspective towards understanding the architecture: we show that the Transformer can be mathematically interpreted as a *numerical Ordinary Differential Equation (ODE) solver for a convection-diffusion equation in a multi-particle dynamic system*. In particular, how words in a sentence are abstracted into contexts by passing through the layers of the Transformer can be interpreted as approximating multiple particles' movement in the space using the Lie-Trotter splitting scheme and the Euler's method. Given this ODE's perspective, the rich literature of numerical analysis can be brought to guide us in designing effective structures beyond the Transformer. As an example, we propose to replace the Lie-Trotter splitting scheme by the Strang–Marchuk splitting scheme, a scheme that is more commonly used and with much lower local truncation errors. The Strang–Marchuk splitting scheme suggests that the self-attention and position-wise feed-forward network (FFN) sub-layers should not be treated equally. Instead, in each layer, two position-wise FFN sub-layers should be used, and the self-attention sub-layer is placed in between. This leads to a brand new architecture. Such an FFN-attention-FFN layer is "Macaron-like", and thus we call the network with this new architecture the *Macaron* Net. Through extensive experiments, we show that the Macaron Net is superior to the Transformer on both supervised and unsupervised learning tasks. The reproducible code can be found at `https://github.com/zhuohan123/macaron-net`

1 Introduction

The Transformer is one of the most commonly used neural network architectures in natural language processing. Variants of the Transformer have achieved state-of-the-art performance in many tasks including language modeling [1, 2] and machine translation [3, 4, 5]. Transformer-based unsupervised pre-trained models also show impressive performance in many downstream tasks [6, 7, 8].

The Transformer architecture is mainly built by stacking layers, each of which consists of two sub-layers with residual connections: the self-attention sub-layer and the position-wise feed-forward network (FFN) sub-layer. For a given sentence, the self-attention sub-layer considers the semantics and dependencies of words at different positions and uses that information to capture the internal structure and representations of the sentence. The position-wise FFN sub-layer is applied to each

[*]Equal contribution. Works done while interning at Microsoft Research Asia.

Preprint. Under review.

position separately and identically to encode context at each position into higher-level representations. Although the Transformer has demonstrated promising results in many tasks, its design principle is not fully understood, and thus the strength of the architecture is not fully exploited. As far as we know, there is little work studying the foundation of the Transformer or different design choices.

In this paper, we provide a novel perspective towards understanding the architecture. In particular, we are the first to show that the Transformer architecture is inherently related to Multi-Particle Dynamic System (MPDS) in physics. MPDS is a well-established research field which aims at modeling how a collection of particles move in the space using differential equations [9]. In MPDS, the behavior of each particle is usually modeled by two factors separately. The first factor is the convection which concerns the mechanism of each particle regardless of other particles in the system, and the second factor is the diffusion which models the movement of the particle resulting from other particles in the system.

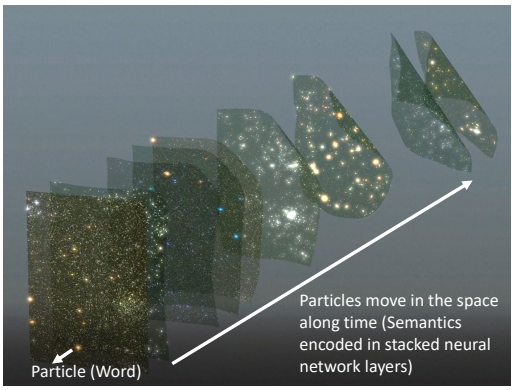

Figure 1: Physical interpretation of Transformer.

Inspired by the relationship between the ODE and neural networks [10, 11], we first show that the Transformer layers can be naturally interpreted as a numerical ODE solver for a first-order convection-diffusion equation in MPDS. To be more specific, the self-attention sub-layer, which transforms the semantics at one position by attending over all other positions, corresponds to the diffusion term; The position-wise FFN sub-layer, which is applied to each position separately and identically, corresponds to the convection term. The number of stacked layers in the Transformer corresponds to the time dimension in ODE. In this way, the stack of self-attention sub-layers and position-wise FFN sub-layers with residual connections can be viewed as solving the ODE problem numerically using the Lie–Trotter splitting scheme [12] and the Euler's method [13]. By this interpretation, we have a novel understanding of learning contextual representations of a sentence using the Transformer: the feature (a.k.a, embedding) of words in a sequence can be considered as the initial positions of a collection of particles, and the latent representations abstracted in stacked Transformer layers can be viewed as the location of particles moving in a high-dimensional space at different time points.

Such an interpretation not only provides a new perspective on the Transformer but also inspires us to design new structures by leveraging the rich literature of numerical analysis. The Lie–Trotter splitting scheme is simple but not accurate and often leads to high approximation error [12]. The Strang–Marchuk splitting scheme [14] is developed to reduce the approximation error by a simple modification to the Lie-Trotter splitting scheme and is theoretically more accurate. Mapped to neural network design, the Strang–Marchuk splitting scheme suggests that there should be three sub-layers: two position-wise feed-forward sub-layers with half-step residual connections and one self-attention sub-layer placed in between with a full-step residual connection. By doing so, the stacked layers will be more accurate from the ODE's perspective and will lead to better performance in deep learning. As the FFN-attention-FFN layer is "Macaron-like", we call it *Macaron layer* and call the network composed of Macaron layers the *Macaron Net*.

We conduct extensive experiments on both supervised and unsupervised learning tasks. For each task, we replace Transformer layers by Macaron layers and keep the number of parameters to be the same. Experiments show that the Macaron Net can achieve higher accuracy than the Transformer on all tasks which, in a way, is consistent with the ODE theory.

2 Background

2.1 Relationship Between Neural Networks and ODE

Recently, there are extensive studies to bridge deep neural networks with ordinary differential equations [15, 10, 16, 11, 17, 18, 19]. We here present a brief introduction to such a relationship and discuss how previous works borrow powerful tools from numerical analysis to help deep neural network design.

A first-order ODE problem is usually defined as to solve the equation (i.e., calculate $x(t)$ for any t) which satisfies the following first-order derivative and the initial condition:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x, t), \quad x(t_0) = w, \tag{1}$$

in which $x(t) \in \mathbb{R}^d$ for all $t \geq t_0$. ODEs usually have physical interpretations. For example, $x(t)$ can be considered as the location of a particle moving in the d-dimensional space and the first order time derivative can be considered as the velocity of the particle.

Usually there is no analytic solution to Eqn (1) and the problem has to be solved numerically. The simplest numerical ODE solver is the Euler's method [13]. The Euler's method discretizes the time derivative $\frac{\mathrm{d}x(t)}{\mathrm{d}t}$ by its first-order approximation $\frac{x(t_2)-x(t_1)}{t_2-t_1} \approx f(x(t_1), t_1)$. By doing so, for the fixed time horizon $T = t_0 + \gamma L$, we can estimate $x(T)$ from $x_0 \doteq x(t_0)$ by sequentially estimating $x_{l+1} \doteq x(t_{l+1})$ using

$$x_{l+1} = x_l + \gamma f(x_l, t_l) \tag{2}$$

where $l = 0, \cdots, L-1$, $t_l = t_0 + \gamma l$ is the time point corresponds to x_l, and $\gamma = (T - t_0)/L$ is the step size. As we can see, this is mathematically equivalent to the ResNet architecture [10, 11]: The function $\gamma f(x_l, t_l)$ can be considered as a neural-network block, and the second argument t_l in the function indicates the set of parameters in the l-th layer. The simple temporal discretization by Euler's method naturally leads to the residual connection.

Observing such a strong relationship, researchers use ODE theory to explain and improve the neural network architectures mainly designed for computer vision tasks. [10, 11] show any parametric ODE solver can be viewed as a deep residual network (probably with infinite layers), and the parameters in the ODE can be optimized through backpropagation. Recent works discover that new neural networks inspired by sophisticated numerical ODE solvers can lead to better performance. For example, [20] uses a high-precision Runge-Kutta method to design a neural network, and the new architecture achieves higher accuracy. [16] uses a leap-frog method to construct a reversible neural network. [21, 22] try to understand recurrent neural networks from the ODE's perspective, and [23] uses non-local differential equations to model non-local neural networks.

2.2 Transformer

The Transformer architecture is usually developed by stacking Transformer layers [3, 7]. A Transformer layer operates on a sequence of vectors and outputs a new sequence of the same shape. The computation inside a layer is decomposed into two steps: the vectors first pass through a (multi-head) self-attention sub-layer and the output will be further put into a position-wise feed-forward network sub-layer. Residual connection [24] and layer normalization [25] are employed for both sub-layers. The visualization of a Transformer layer is shown in Figure 2(a) and the two sub-layers are defined as below.

Self-attention sub-layer The attention mechanism can be formulated as querying a dictionary with key-value pairs [3], e.g., $\text{Attention}(Q, K, V) = \text{softmax}(QK^T/\sqrt{d_{model}}) \cdot V$, where d_{model} is the dimensionality of the hidden representations and Q (Query), K (Key), V (Value) are specified as the hidden representations of the previous layer in the so-called *self-attention* sub-layers in the Transformer architecture. The multi-head variant of attention allows the model to jointly attend to information from different representation subspaces, and is defined as

$$\text{Multi-head}(Q, K, V) = \text{Concat}(\text{head}_1, \cdots, \text{head}_H)W^O, \tag{3}$$

$$\text{head}_k = \text{Attention}(QW_k^Q, KW_k^K, VW_k^V), \tag{4}$$

where $W_k^Q \in \mathbb{R}^{d_{model} \times d_K}, W_k^K \in \mathbb{R}^{d_{model} \times d_K}, W_k^V \in \mathbb{R}^{d_{model} \times d_V}$, and $W^O \in \mathbb{R}^{Hd_V \times d_{model}}$ are project parameter matrices, H is the number of heads, and d_K and d_V are the dimensionalities of Key and Value.

Position-wise FFN sub-layer In addition to the self-attention sub-layer, each Transformer layer also contains a fully connected feed-forward network, which is applied to each position separately and identically. This feed-forward network consists of two linear transformations with an activation

function σ in between. Specially, given vectors $h_1, \ldots, h_n$, a position-wise FFN sub-layer transforms each h_i as $\text{FFN}(h_i) = \sigma(h_i W_1 + b_1)W_2 + b_2$, where W_1, W_2, b_1 and b_2 are parameters.

In this paper, we take the first attempt to provide an understanding of the feature extraction process in natural language processing from the ODE's viewpoint. As discussed in Section 2.1, several works interpret the standard ResNet using the ODE theory. However, we found this interpretation cannot be directly applied to the Transformer architecture. First, different from vision applications whose size of the input (e.g., an image) is usually predefined and fixed, the input (e.g., a sentence) in natural language processing is always of variable length, which makes the single-particle ODE formulation used in previous works not applicable. Second, the Transformer layer contains very distinct sub-layers. The self-attention sub-layer takes the information from all positions as input while the position-wise feed-forward layer is applied to each position separately. How to interpret these heterogeneous components by ODE is also not covered by previous works [23, 11].

3 Reformulate Transformer Layers as an ODE Solver for Multi-Particle Dynamic System

In this section, we first introduce the general form of differential equations in MPDS and then reformulate the stacked Transformer layers to show they form a numerical ODE solver for a specific problem. After that, we use advanced methods in the ODE theory to design new architectures.

3.1 Multi-Particle ODE and Its Numerical Solver

Understanding the dynamics of multiple particles' movements in space is one of the important problems in physics, especially in fluid mechanics and astrophysics [9]. The behavior of each particle is usually modeled by two factors: The first factor concerns about the mechanism of its movement regardless of other particles, e.g., caused by an external force outside of the system, which is usually referred to as the convection; The second factor concerns about the movement resulting from other particles, which is usually referred to as the diffusion. Mathematically, assume there are n particles in d-dimensional space. Denote $x_i(t) \in \mathbb{R}^d$ as the location of i-th particle at time t. The dynamics of particle i can be formulated as

$$\begin{aligned} \frac{\mathrm{d}x_i(t)}{\mathrm{d}t} &= F(x_i(t), [x_1(t), \cdots, x_n(t)], t) + G(x_i(t), t), \\ x_i(t_0) &= w_i, \quad i = 1, \ldots, n. \end{aligned} \tag{5}$$

Function $F(x_i(t), [x_1(t), \cdots, x_n(t)], t)$ represents the diffusion term which characterizes the interaction between the particles. $G(x, t)$ is a function which takes a location x and time t as input and represents the convection term.

Splitting schemes As we can see, there are two coupled terms in the right-hand side of Eqn (5) describing different physical phenomena. Numerical methods of directly solving such ODEs can be complicated. The splitting method is a prevailing way of solving such coupled differential equations that can be decomposed into a sum of differential operators [26]. Furthermore, splitting convection from diffusion is quite standard for many convection-diffusion equations [27, 12]. The Lie–Trotter splitting scheme [12] is the simplest splitting method. It splits the right-hand side of Eqn (5) into function $F(\cdot)$ and $G(\cdot)$ and solves the individual dynamics alternatively. More precisely, to compute $x_i(t + \gamma)$ from $x_i(t)$, the Lie–Trotter splitting scheme with the Euler's method reads as

$$\tilde{x}_i(t) = x_i(t) + \gamma F(x_i(t), [x_1(t), x_2(t), \cdots, x_n(t)], t), \tag{6}$$

$$x_i(t + \gamma) = \tilde{x}_i(t) + \gamma G(\tilde{x}_i(t), t). \tag{7}$$

From time t to time $t + \gamma$, the Lie–Trotter splitting method first solves the ODE with respect to $F(\cdot)$ and acquire an intermediate location $\tilde{x}_i(t)$. Then, starting from $\tilde{x}_i(t)$, it solves the second ODE with respect to $G(\cdot)$ to obtain $x_i(t + \gamma)$.

3.2 Physical interpretation of the Transformer

We reformulate the two sub-layers of the Transformer in order to match its form with the ODE described above. Denote $x_l = (x_{l,1}, \ldots, x_{l,n})$ as the input to the l-th Transformer layer, where n is the sequence length and $x_{l,i}$ is a real-valued vector in $\mathbb{R}^d$ for any i.

Reformulation of the self-attention sub-layer Denote $\tilde{x}_{l,i}$ as the output of the (multi-head) self-attention sub-layer at position i with residual connections. The computation of $\tilde{x}_{l,i}$ can be written as

$$\tilde{x}_{l,i} = x_{l,i} + \text{Concat}\left(\text{head}_1, ..., \text{head}_H\right) W^{O,l}, \tag{8}$$

$$\text{where head}_k = \sum_{j=1}^{n} \alpha_{ij}^{(k)} [x_{l,j} W_k^{V,l}] = \sum_{j=1}^{n} \left(\frac{\exp(e_{ij}^{(k)})}{\sum_{q=1}^{n} \exp(e_{iq}^{(k)})} \right) [x_{l,j} W_k^{V,l}], \tag{9}$$

and $e_{ij}^{(k)}$ is computed as the dot product of input $x_{l,i}$ and $x_{l,j}$ with linear projection matrices $W_k^{Q,l}$ and $W_k^{K,l}$, i.e., $e_{ij}^{(k)} = d_{model}^{-1/2} \cdot (x_{l,i} W_k^{Q,l})(x_{l,j} W_k^{K,l})^T$. Considering $\alpha_{ij}^{(k)}$ as a normalized value of the pair-wise dot product $e_{ij}^{(k)}$ over j, we can generally reformulate Eqn (8) as

$$\tilde{x}_{l,i} = x_{l,i} + \text{MultiHeadAtt}_{W_{att}^l}(x_{l,i}, [x_{l,1}, x_{l,2}, \cdots, x_{l,n}]), \tag{10}$$

where W_{att}^l denotes all trainable parameters in the l-th self-attention sub-layer.

Reformulation of the position-wise FFN sub-layer Next, $\tilde{x}_{l,i}$ is put into the position-wise FFN sub-layer with residual connections and output $x_{l+1,i}$. The computation of $x_{l+1,i}$ can be written as

$$x_{l+1,i} = \tilde{x}_{l,i} + \text{FFN}_{W_{ffn}^l}(\tilde{x}_{l,i}), \tag{11}$$

where W_{ffn}^l denotes all trainable parameters in the l-th position-wise FFN sub-layer.

Reformulation of Transformer layers Combining Eqn (10) and (11), we reformulate the Transformer layers[2] as

$$\tilde{x}_{l,i} = x_{l,i} + \text{MultiHeadAtt}_{W_{att}^l}(x_{l,i}, [x_{l,1}, x_{l,2}, \cdots, x_{l,n}]), \tag{12}$$

$$x_{l+1,i} = \tilde{x}_{l,i} + \text{FFN}_{W_{ffn}^l}(\tilde{x}_{l,i}). \tag{13}$$

We can see that the Transformer layers (Eqn (12-13)) resemble the multi-particle ODE solver in Section 3.1 (Eqn (6-7)). Indeed, we can formally establish the link between the ODE solver with splitting scheme and stacked Transformer layers as below.

Claim 1. *Define* $\gamma F^*(x_{l,i}, [x_{l,1}, \cdots, x_{l,n}], t_l) = \textit{MultiHeadAtt}_{W_{att}^l}(x_{l,i}, [x_{l,1}, \cdots, x_{l,n}])$ *and* $\gamma G^*(x_{l,i}, t_l) = \textit{FFN}_{W_{ffn}^l}(x_{l,i})$. *The Transformer can be viewed as a numerical ODE solver using Lie–Trotter splitting scheme and the Euler's method (with time step* γ*) for Eqn (5) with* F^* *and* G^*.

The above observation grants a physical interpretation of natural language processing and provides a new perspective on the Transformer architecture. First, this perspective provides a unified view of the heterogeneous components in the Transformer. The self-attention sub-layer is viewed as a diffusion term which characterizes the particle interactions while the position-wise feed-forward networks is viewed as a convection term. The two terms together naturally form the convection-diffusion equation in physics. Second, this interpretation advances our understanding of the latent representations of language through the Transformer. Viewing the feature (a.k.a., embedding) of words in a sequence as the initial position of particles, we can interpret the latent representations of the sentence abstracted by the Transformer as particles moving in a high-dimensional space as demonstrated in Figure 1 [29].

3.3 Improving Transformer Via Strang–Marchuk Splitting Scheme

In the previous subsection, we have successfully mapped the Transformer architecture to a numerical ODE solver for MPDS. However, we would like to point out that one of the key components in this ODE solver, the Lie-Trotter splitting scheme, is the simplest one but has relatively high errors. In this subsection, we incorporate one of the most popular and widely used splitting scheme [12], the Strang–Marchuk splitting scheme, into the design of the neural networks.

[2] Layer normalization is sometimes applied to the sub-layers but recent work [28] shows that the normalization trick is not essential and can be removed. One can still readily check that the reformulation (Eqn (12) and (13)) still holds with layer normalization.

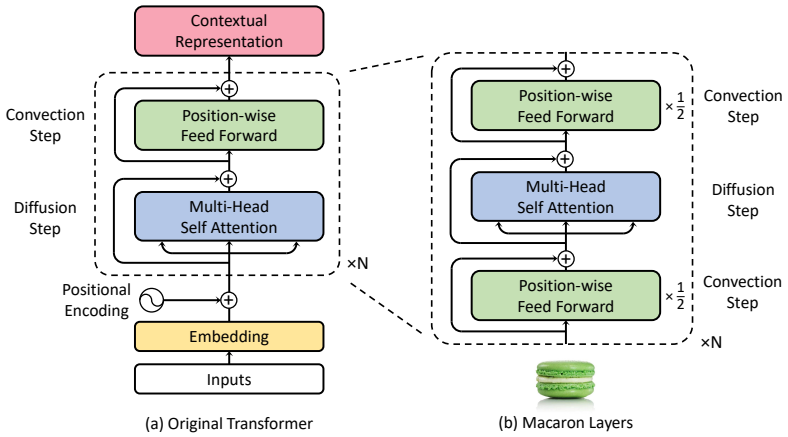

Figure 2: The Transformer and our Macaron architectures.

The Lie-Trotter splitting scheme solves the dynamics of $F(\cdot)$ and $G(\cdot)$ alternatively and exclusively in that order. This inevitably brings bias and leads to higher local truncation errors [12]. To mitigate the bias, we use a simple modification to the Lie-Trotter splitting scheme by dividing the one-step numerical solver for $G(\cdot)$ into two *half-steps*: we put one half-step before solving $F(\cdot)$ and put the other half-step after solving $F(\cdot)$. This modified splitting scheme is known as the Strang–Marchuk splitting scheme [14]. Mathematically, to compute $x_i(t+\gamma)$ from $x_i(t)$, the Strang–Marchuk splitting scheme reads as

$$\tilde{x}_i(t) = x_i(t) + \frac{\gamma}{2} G(x_i(t), t), \tag{14}$$

$$\hat{x}_i(t) = \tilde{x}_i(t) + \gamma F(\tilde{x}_i(t), [\tilde{x}_1(t), \tilde{x}_2(t), \cdots, \tilde{x}_n(t)], t), \tag{15}$$

$$x_i(t+\gamma) = \hat{x}_i(t) + \frac{\gamma}{2} G\left(\hat{x}_i(t), t + \frac{\gamma}{2}\right). \tag{16}$$

The Strang–Marchuk splitting scheme enjoys higher-order accuracy than the Lie-Trotter splitting scheme [30] in terms of the *local truncation error* [13], which measures the per-step distance between the true solution and the approximated solution using numerical schemes. Mathematically, for a differential equation $\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x, t)$ and a numerical scheme $\mathcal{A}$, the local truncation error of numerical scheme $\mathcal{A}$ is defined as $\tau = x(t+\gamma) - \mathcal{A}(x(t), \gamma)$. For example, when $\mathcal{A}$ is the Euler's method, $\tau_{Euler} = x(t+\gamma) - x(t) - \gamma f(x(t), t)$. The order of local truncation error of the two schemes has been studied in [30], as shown in the following theorem.

Theorem 1. *[30] The local truncation error of the Lie-Trotter splitting scheme is* ***second-order*** *($O(\gamma^2)$) and the local truncation error of the Strang–Marchuk splitting scheme is* ***third-order*** *($O(\gamma^3)$).*

For completeness, we provide the formal theorem with proof in Appendix A. We can see from Eqn (14-16) that the Strang-Marchuk splitting scheme uses a three-step process to solve the ODE. Mapped to neural network design, the Strang–Marchuk splitting scheme (together with the Euler's method) suggests there should also be three sub-layers instead of the two sub-layers in Transformer. By replacing function γF and γG by MultiHeadAtt and FFN, we have

$$\tilde{x}_{l,i} = x_{l,i} + \frac{1}{2}\mathrm{FFN}_{W_{ffn}^{l,down}}(x_{l,i}), \tag{17}$$

$$\hat{x}_{l,i} = \tilde{x}_{l,i} + \mathrm{MultiHeadAtt}_{W_{att}^{l}}(\tilde{x}_{l,i}, [\tilde{x}_{l,1}, \tilde{x}_{l,2}, \cdots, \tilde{x}_{l,n}]), \tag{18}$$

$$x_{l+1,i} = \hat{x}_{l,i} + \frac{1}{2}\mathrm{FFN}_{W_{ffn}^{l,up}}(\hat{x}_{l,i}). \tag{19}$$

From Eqn (17-19), we can see that the new layer composes of three sub-layers. Each hidden vector at different positions will first pass through the first position-wise FFN sub-layer with a half-step residual connection ("$\frac{1}{2}$" in Eqn (17)), and then the output vectors will be feed into a self-attention sub-layer. In the last step, the vectors outputted from the self-attention sub-layer will be put into the second position-wise FFN sub-layer with a half-step residual connection. Since the FFN-attention-FFN structure is "Macaron"-like, we call the layer as *Macaron* layer and call the network using

Table 1: Translation performance (BLEU) on IWSLT14 De-En and WMT14 En-De testsets.

Method	IWSLT14 De-En `small`	WMT14 En-De `base`	WMT14 En-De `big`
Transformer [3]	34.4	27.3	28.4
Weighted Transformer [31]	/	28.4	28.9
Relative Transformer [32]	/	26.8	29.2
Universal Transformer [4]	/	**28.9**	/
Scaling NMT [33]	/	/	29.3
Dynamic Conv [34]	35.2	/	29.7
Macaron Net	**35.4**	**28.9**	**30.2**

Macaron layers as Macaron Net, as shown in Figure 2(b). Previous works [10, 20] have successfully demonstrated that the neural network architectures inspired from higher-order accurate numerical ODE solvers will lead to better results in deep learning and we believe the Macaron Net can achieve better performance on practical natural language processing applications than the Transformer.

4 Experiments

We test our proposed Macaron architectures in both supervised and unsupervised learning setting. For supervised learning setting, we use IWLST14 and WMT14 machine translation datasets. For unsupervised learning setting, we pretrain the model using the same method as in [7] and test the learned model over a set of downstream tasks. Due to space limitations, we put dataset description, model description, hyperparameter configuration into Appendix B.

4.1 Experiment Settings

Machine Translation Machine translation is an important application for natural language processing [3]. We evaluate our methods on two widely used public datasets: IWSLT14 German-to-English (De-En) and WMT14 English-to-German (En-De) dataset.

For the WMT14 dataset, the basic configurations of the Transformer architecture are the `base` and the `big` settings [3]. Both of them consist of a 6-layer encoder and 6-layer decoder. The size of the hidden nodes and embeddings are set to 512 for `base` and 1024 for `big`. The number of heads are 8 for `base` and 16 for `big`. Since the IWSLT14 dataset is much smaller than the WMT14 dataset, the `small` setting is usually used, whose size of hidden states and embeddings is set to 512 and the number of heads is set to 4. For all settings, the dimensionality of the inner-layer of the position-wise FFN is four times of the dimensionality of the hidden states.

For each setting (`base`, `big` and `small`), we replace all Transformer layers by the Macaron layers[3] and obtain the `base`, `big` and `small` Macaron, each of which contains two position-wise feed-forward sub-layers in a layer. To make a fair comparison, we set the dimensionality of the inner-layer of the two FFN sub-layers in the Macaron layers to two times of the dimensionality of the hidden states. By doing this, the `base`, `big` and `small` Macaron have the same number of parameters as the `base`, `big` and `small` Transformer respectively.

Unsupervised Pretraining BERT [7] is the current state-of-the-art pre-trained contextual representation model based on a multi-layer Transformer encoder architecture and trained by masked language modeling and next-sentence prediction tasks. We compare our proposed Macaron Net with the `base` setting from the original BERT paper [7], which consists of 12 Transformer layers. The size of hidden states and embeddings are set to 768, and the number of attention heads is set to 12. Similarly, we replace the Transformer layers in BERT `base` by the Macaron layers and reduce the

[3]The translation model is based on the encoder-decoder framework. In the Transformer, the decoder layer has a third sub-layer which performs multi-head attention over the output of the encoder stack (encoder-decoder-attention) and a mask to prevent positions from attending to subsequent positions. In our implementation of Macaron decoder, we also use masks and split the FFN into two sub-layers and thus our decoder layer is (FFN, self-attention, encoder-decoder-attention, and FFN).

Table 2: Test results on the GLUE benchmark (except WNLI).

Method	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m/mm	QNLI	RTE	**GLUE**
Existing systems									
ELMo [8]	33.6	90.4	84.4/78.0	74.2/72.3	63.1/84.3	74.1/74.5	79.8	58.9	70.0
OpenAI GPT [35]	47.2	93.1	87.7/83.7	85.3/84.8	70.1/88.1	80.7/80.6	87.2	69.1	76.9
BERT `base` [7]	52.1	93.5	**88.9/84.8**	87.1/85.8	**71.2/89.2**	84.6/83.4	90.5	66.4	78.3
Our systems									
BERT `base` (ours)	52.8	92.8	87.3/83.0	81.2/80.0	70.2/88.4	84.4/83.7	90.4	64.9	77.4
Macaron Net `base`	**57.6**	**94.0**	88.4/84.4	**87.5/86.3**	70.8/89.0	**85.4/84.5**	**91.6**	**70.5**	**79.7**

dimensionality of the inner-layer of the two FFN sub-layers by half, and thus we keep the number of parameters of our Macaron `base` same as BERT `base`.

4.2 Experiment Results

Machine Translation We use BLEU [36] as the evaluation measure for machine translation. Following common practice, we use tokenized case-sensitive BLEU and case-insensitive BLEU for WMT14 En-De and IWSLT14 De-En respectively.

The results for machine translation are shown in Table 1. For the IWSLT14 dataset, our Macaron `small` outperforms the Transformer `small` by 1.0 point in terms of BLEU. For the WMT14 dataset, our Macaron `base` outperforms its Transformer counterpart by 1.6 BLEU points. Furthermore, the performance of our Macaron `base` model is even better than that of the Transformer `big` model reported in [3], but with much less number of parameters. Our Macaron `big` outperforms the Transformer `big` by 1.8 BLEU points. Comparing with other concurrent works, the improvements in our proposed method are still significant.

Unsupervised Pretraining Following [7], we evaluate all models by fine-tuning them on 8 downstream tasks in the General Language Understanding Evaluation (GLUE) benchmark [37], including CoLA [38], SST-2 [39], MRPC [40], STS-B [41], QQP [42], MNLI [43], QNLI [44], and RTE [45]. More details about individual tasks and their evaluation metrics can be found in Appendix B.3 and [37, 7]. To fine-tune the models, we follow the hyperparameter search space in [7] for all downstream tasks, including different batch sizes, learning rates, and numbers of epochs.

The GLUE results are presented in Table 2. We present the results of two BERT `base` models. One is from [7], and the other is trained using our own data. Due to that some pieces of data used in [7] are no longer freely distributed, the two numbers are slightly different. We can see from the table, our proposed Macaron Net `base` outperforms all baselines in terms of the general GLUE score. Specifically, given the same dataset, our proposed model outperforms our trained BERT `base` model in all tasks. Even comparing with the BERT `base` model in [7], our model performs better in 6 out of 8 tasks and achieves close performance in the rest 2 tasks. Comparing our results with original BERT model, we can see that given a fixed number of parameters and a fixed number of attention sub-layers, stacking more FFN sub-layers can get better performance on downstream tasks.

As a summary, the improvement in both machine translation and GLUE tasks well aligns with the ODE theory and our proposed architecture performs better than the Transformer in real practice.

5 Conclusion and Future Work

In this paper, we interpret the Transformer as a numerical ODE solver for a convection-diffusion equation in a multi-particle dynamic system. Specifically, how words in a sentence are abstracted into contexts by passing through the layers of the Transformer can be interpreted as approximating multiple particles' movement in the space using the Lie-Trotter splitting scheme and the Euler's method. By replacing the Lie-Trotter splitting scheme with the more advanced Strang–Marchuk splitting scheme, we obtain a new architecture. The improvements in real applications show the effectiveness of the new model and are consistent with the ODE theory. In the future, we will explore deeper connections between the ODE theory and the Transformer models and use the ODE theory to improve the individual components in the Transformer architecture such as attention modules.

Acknowledgements

We would like to thank Shuonan Wu for helpful discussions and Linyuan Gong for the help on unsupervised pretraining experiments.

References

[1] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[2] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. *arXiv preprint arXiv:1808.04444*, 2018.

[3] 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*, pages 5998–6008, 2017.

[4] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[5] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*, 2018.

[6] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[9] Forest Ray Moulton. *An introduction to celestial mechanics*. Courier Corporation, 2012.

[10] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *arXiv preprint arXiv:1710.10121*, 2017.

[11] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6572–6583, 2018.

[12] Juergen Geiser. *Decomposition methods for differential equations: theory and applications*. CRC Press, 2009.

[13] Uri M Ascher and Linda R Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*, volume 61. Siam, 1998.

[14] Gilbert Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968.

[15] E Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

[16] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017.

[17] Xiaoshuai Zhang, Yiping Lu, Jiaying Liu, and Bin Dong. Dynamically unfolding recurrent restorer: A moving endpoint control method for image restoration. In *International Conference on Learning Representations*, 2019.

[18] Sho Sonoda and Noboru Murata. Transport analysis of infinitely deep neural network. *The Journal of Machine Learning Research*, 20(1):31–82, 2019.

[19] Matthew Thorpe and Yves van Gennip. Deep limits of residual neural networks. *arXiv preprint arXiv:1810.11741*, 2018.

[20] Mai Zhu, Bo Chang, and Chong Fu. Convolutional neural networks combined with runge-kutta methods. *arXiv preprint arXiv:1802.08831*, 2018.

[21] Qianli Liao and Tomaso Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*, 2016.

[22] Bo Chang, Minmin Chen, Eldad Haber, and Ed H Chi. Antisymmetricrnn: A dynamical system view on recurrent neural networks. *arXiv preprint arXiv:1902.09689*, 2019.

[23] Yunzhe Tao, Qi Sun, Qiang Du, and Wei Liu. Nonlocal neural networks, nonlocal diffusion and nonlocal modeling. In *Advances in Neural Information Processing Systems*, pages 496–506, 2018.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[25] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[26] Robert I McLachlan and G Reinout W Quispel. Splitting methods. *Acta Numerica*, 11:341–434, 2002.

[27] Roland Glowinski, Stanley J Osher, and Wotao Yin. *Splitting methods in communication, imaging, science, and engineering*. Springer, 2017.

[28] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

[29] Wenxin Zhang. The bonfire of time. In Tragoidia, editor, *Iris Long*, 2019.

[30] Alexander V Bobylev and Taku Ohwada. The error of the splitting scheme for solving evolutionary equations. *Applied Mathematics Letters*, 14(1):45–48, 2001.

[31] Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. Weighted transformer network for machine translation. *arXiv preprint arXiv:1711.02132*, 2017.

[32] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[33] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. *arXiv preprint arXiv:1806.00187*, 2018.

[34] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019.

[35] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

[36] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[37] 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. 2019. In the Proceedings of ICLR.

[38] Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *arXiv preprint 1805.12471*, 2018.

[39] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP.*, pages 1631–1642, 2013.

[40] William B Dolan and Chris. Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the International Workshop on Paraphrasing.*, 2005.

[41] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[42] Z. Chen, H. Zhang, X. Zhang, and L. Zhao. Quora question pairs. 2018.

[43] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL-HLT.*, 2018.

[44] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy. Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP.*, pages 2383–2392. Association for Computational Linguistics, 2016.

[45] Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo. Magnini. The fifth PASCAL recognizing textual entailment challenge. 2009.

[46] Carmen Chicone. Ordinary differential equations by vladimir i. arnold. *Siam Review*, 49(2):335–336, 2007.

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

[48] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[49] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional Sequence to Sequence Learning. In *Proc. of ICML*, 2017.

[50] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *arXiv preprint arXiv:1506.06724*, 2015.

[51] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *ACL*, 2007.

[52] Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[53] Hector J Levesque, Ernest Davis, and Leora. Morgenstern. The Winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning.*, volume 46, page 47, 2011.

Appendix

A Proof of the Theorem

Theorem 2. *([30]) We denote the true solution of equation* $\frac{\mathrm{d}x}{\mathrm{d}t} = F(x), x(0) = y_0$[4] *at time* t *as* $x(t) = S_F^t(y_0)$*. Simliarily, we can define* S_G^t *and* S_{F+G}^t*. The local truncation error at* $t = 0$ *of the Lie-Trotter splitting scheme is* $S_{F+G}^\gamma(y_0) - S_G^\gamma[S_F^\gamma(y_0)] = \frac{\gamma^2}{2}\{F'(y_0)(y_0)G(y_0) - G'(y_0)(y_0)F(y_0)\} + O(\gamma^3)$ *which is* ***second-order****. The local truncation error at* $t = 0$ *of the Strang–Marchuk splitting scheme* $S_G^{\gamma/2}\{S_F^\gamma[S_G^{\gamma/2}(y_0)]\}$ *is* $S_{F+G}^\gamma(y_0) - S_G^{\gamma/2}\{S_F^\gamma[S_G^{\gamma/2}(y_0)]\} = O(\gamma^3)$ *which is* ***third-order****.*

Proof. Since

$$S_F^\gamma(y_0) = y_0 + \gamma F(y_0) + \frac{\gamma^2}{2}F'(y_0)F(y_0) + O(\gamma^3),$$
$$S_G^\gamma(y_0) = y_0 + \gamma G(y_0) + \frac{\gamma^2}{2}G'(y_0)G(y_0) + O(\gamma^3),$$

we have

$$S_G^\gamma[S_F^\gamma(y_0)] = S_F^\gamma(y_0) + \gamma G(S_F^\gamma(y_0)) + \frac{\gamma^2}{2}G'(S_F^\gamma(y_0))G(S_F^\gamma(y_0)) + O(\gamma^3).$$

At the same time, we have

$$G(y_0 + \gamma F(y_0) + O(\gamma^2)) = G(y_0) + \gamma G'(y_0)F(y_0) + O(\gamma^2),$$
$$G'(S_F^\gamma(y_0))G(S_F^\gamma(y_0)) = G'(y_0)G(y_0) + O(\gamma).$$

Combine the estimations we have

$$\begin{aligned} S_G^\gamma[S_F^\gamma(y_0)] &= y_0 + \gamma[F(y_0) + G(y_0)] \\ &\quad + \frac{\gamma^2}{2}[G'(y_0)G(y_0) + F'(y_0)F(y_0) + 2G'(y_0)F(y_0)] + O(\gamma^3). \end{aligned}$$

As a result, we can estimate the local truncation error of Lie-Trotter splitting scheme as

$$\begin{aligned} & S_{F+G}^\gamma(y_0) - S_G^\gamma[S_F^\gamma(y_0)] \\ &= y_0 + \gamma(F(y_0) + G(y_0)) + \frac{\gamma^2}{2}(F'(y_0) + G'(y_0))(F(y_0) + G(y_0)) + O(\gamma^3) \\ &\quad - (y_0 + \gamma[F(y_0) + G(y_0)] + \frac{\gamma^2}{2}[G'(y_0)G(y_0) + F'(y_0)F(y_0) + 2G'(y_0)F(y_0)] + O(\gamma^3)) \\ &= \frac{\gamma^2}{2}\{F'(y_0)G(y_0) - G'(y_0)F(y_0)\} + O(\gamma^3). \end{aligned}$$

To estimate the Strang–Marchuk splitting scheme's local truncation error, we rewrite the Strang–Marchuk splitting scheme as

$$S_G^{\gamma/2}\{S_F^\gamma[S_G^{\gamma/2}(y_0)]\} = S_G^{\gamma/2}\{S_F^{\gamma/2}\{S_F^{\gamma/2}[S_G^{\gamma/2}(y_0)]\}\}.$$

From the previous estimation of Lie–Trotter splitting scheme we have

$$S_{F+G}^{\gamma/2}(y_0) - S_G^\gamma[S_F^\gamma(y_0)] = \frac{\gamma^2}{8}\{F'(y_0)G(y_0) - G'(y_0)F(y_0)\} + O(\gamma^3),$$
$$S_{F+G}^{\gamma/2}(y_0) - S_F^\gamma[S_G^\gamma(y_0)] = \frac{\gamma^2}{8}\{G'(y_0)F(y_0) - F'(y_0)G(y_0)\} + O(\gamma^3).$$

[4] Since a time-dependent ODE can be formulated as a time-independent ODE by introducing an auxiliary variable [46], the theorem here developed for time-independent ODEs can also be applied to time-dependent ODEs without loss of generality.

Combine the two estimations, we have

$$
\begin{aligned}
S_G^{\gamma/2}\{S_F^{\gamma/2}\{S_F^{\gamma/2}[S_G^{\gamma/2}(y_0)]\}\} &= S_{F+G}^{\gamma}(y_0) + \frac{\gamma^2}{8}\{F'(y_0)G(y_0) - G'(y_0)F(y_0)\} \\
&\quad + \frac{\gamma^2}{8}\{G'(y_0)F(y_0) - F'(y_0)G(y_0)\} + O(\gamma^3) \\
&= S_{F+G}^{\gamma}(y_0) + O(\gamma^3).
\end{aligned}
$$

□

B Experiment Settings

B.1 Machine Translation

Dataset The training/validation/test sets of the IWSLT14 dataset contain about 153K/7K/7K sentence pairs, respectively. We use a vocabulary of 10K tokens based on a joint source and target byte pair encoding (BPE) [47]. For WMT14 dataset, we replicate the setup of [3], which contains 4.5M training parallel sentence pairs. Newstest2014 is used as the test set, and Newstest2013 is used as the validation set. The 37K vocabulary for WMT14 is based on a joint source and target BPE factorization.

Model For the WMT14 dataset, the basic configurations of the Transformer architecture are the `base` and the `big` settings [3]. Both of them consist of a 6-layer encoder and 6-layer decoder. The size of the hidden nodes and embeddings are set to 512 for `base` and 1024 for `big`. The number of heads are 8 for `base` and 16 for `big`. Since the IWSLT14 dataset is much smaller than the WMT14 dataset, the `small` setting is usually used, whose size of hidden states and embeddings is set to 512 and the number of heads is set to 4. For all settings, the dimensionality of the inner-layer of the position-wise FFN is four times of the dimensionality of the hidden states.

For each setting (`base`, `big` and `small`), we replace all Transformer layers by the Macaron layers and obtain the `base`, `big` and `small` Macaron, each of which contains two position-wise feed-forward sub-layers in a layer. The translation model is based on the encoder-decoder framework. In the Transformer, the decoder layer has a third sub-layer which performs multi-head attention over the output of the encoder stack (encoder-decoder-attention) and a mask to prevent positions from attending to subsequent positions. In our implementation of Macaron decoder, we also use masks and split the FFN into two sub-layers and thus our decoder layer is (FFN, self-attention, encoder-decoder-attention and FFN).

To make a fair comparison, we set the dimensionality of the inner-layer of the two FFN sub-layers in the Macaron layers to two times of the dimensionality of the hidden states. By doing this, the `base`, `big` and `small` Macaron have the same number of parameters as the `base`, `big` and `small` Transformer respectively.

Optimizer and training We use the Adam optimizer and follow the optimizer setting and learning rate schedule in [3]. For the `big` setting, we enlarge the batch size and learning rate as suggested in [33] to accelerate training. We employ label smoothing of value $\epsilon_{ls} = 0.1$ [48] in all experiments. Models for WMT14/IWSLT14 are trained on 4/1 NVIDIA P40 GPUs respectively. Our code is based on the open-sourced `fairseq` [49] code base in PyTorch toolkit.

Evaluation We use BLEU[5] [36] as the evaluation measure for machine translation. Following common practice, we use tokenized case-sensitive BLEU and case-insensitive BLEU for WMT14 En-De and IWSLT14 De-En respectively. During inference, we use beam search with beam size 4 and length penalty 0.6 for WMT14, and beam size 5 and length penalty 1.0 for IWSLT14, following [3].

[5] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

B.2 Unsupervised Pretraining

Pre-training dataset We follow [7] to use English Wikipedia corpus and BookCorpus for pre-training. As the dataset BookCorpus [50] is no longer freely distributed. We follow the suggestions from [7] to crawl and collect BookCorpus[6] on our own. The concatenation of two datasets includes roughly 3.4B words in total, which is comparable with the data corpus used in [7]. We first segment documents into sentences with Spacy;[7] Then, we normalize, lower-case, and tokenize texts using Moses[51] and apply BPE[47]. We randomly split documents into one training set and one validation set. The training-validation ratio for pre-training is 199:1.

Model We compare our proposed Macaron Net with the `base` setting from the original BERT paper [7], which consists of 12 Transformer layers. The size of hidden states and embeddings are set to 768, and the number of attention heads is set to 12. Similarly, we replace the Transformer layers in BERT `base` by the Macaron layers and reduce the dimensionality of the inner-layer of the two FFN sub-layers by half, and thus we keep the number of parameters of our Macaron `base` as the same as BERT `base`.

Optimizer and training We follow [7] to use two tasks to pretrain our model. One task is masked language modeling, which masks some percentage of the input tokens at random, and then requires the model to predict those masked tokens. Another task is next sentence prediction, which requires the model to predict whether two sentences in a given sentence pair are consecutive. We use the Adam optimizer and and follow the optimizer setting and learning rate schedule in [7] and trained the model on 4 NVIDIA P40 GPUs.

B.3 GLUE Dataset

We provide a brief description of the tasks in the GLUE benchmark [37] and our fine-tuning process on the GLUE datasets.

CoLA The Corpus of Linguistic Acceptability [38] consists of English acceptability judgments drawn from books and journal articles on linguistic theory. The task is to predict whether an example is a grammatical English sentence. The performance is evaluated by Matthews correlation coefficient [52].

SST-2 The Stanford Sentiment Treebank [39] consists of sentences from movie reviews and human annotations of their sentiment. The task is to predict the sentiment of a given sentence (positive/negative). The performance is evaluated by the test accuracy.

MRPC The Microsoft Research Paraphrase Corpus [40] is a corpus of sentence pairs automatically extracted from online news sources, with human annotations for whether the sentences in the pair are semantically equivalent, and the task is to predict the equivalence. The performance is evaluated by both the test accuracy and the test F1.

STS-B The Semantic Textual Similarity Benchmark [41] is a collection of sentence pairs drawn from news headlines, video and image captions, and natural language inference data. Each pair is human-annotated with a similarity score from 1 to 5; the task is to predict these scores. The performance is evaluated by Pearson and Spearman correlation coefficients.

QQP The Quora Question Pairs[8] [42] dataset is a collection of question pairs from the community question-answering website Quora. The task is to determine whether a pair of questions are semantically equivalent. The performance is evaluated by both the test accuracy and the test F1.

MNLI The Multi-Genre Natural Language Inference Corpus [43] is a crowdsourced collection of sentence pairs with textual entailment annotations. Given a premise sentence and a hypothesis sentence, the task is to predict whether the premise entails the hypothesis (*entailment*), contradicts

[6] `https://www.smashwords.com/`

[7] `https://spacy.io`

[8] `https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs`

the hypothesis (*contradiction*), or neither (*neutral*). The performance is evaluated by the test accuracy on both *matched* (in-domain) and *mismatched* (cross-domain) sections of the test data.

QNLI The Question-answering NLI dataset is converted from the Stanford Question Answering Dataset (SQuAD) [44] to a classification task. The performance is evaluated by the test accuracy.

RTE The Recognizing Textual Entailment (RTE) datasets come from a series of annual textual entailment challenges [45]. The task is to predict whether sentences in a sentence pair are entailment. The performance is evaluated by the test accuracy.

WNLI The Winograd Schema Challenge [53] is a reading comprehension task in which a system must read a sentence with a pronoun and select the referent of that pronoun from a list of choices. We follow [7] to skip this task in our experiments, because few previous works do better than predicting the majority class for this task.

Fine-tuning on GLUE tasks To fine-tune the models, following [7], we search the optimization hyperparmaters in a search space including different batch sizes (16/32), learning rates (5e-3/3e-5), number of epochs (3/4/5), and a set of different random seeds. We select the model for testing according to their performance on the development set.

Test data Note that the GLUE dataset distribution does not include the Test labels, and we only made a single GLUE evaluation server submission[9] for each of our models.

[9] `https://gluebenchmark.com`