ODE Transformer: An Ordinary Differential Equation-Inspired Model for Sequence Generation

Bei Li\textsuperscript{1}, Quan Du\textsuperscript{1,2}, Tao Zhou\textsuperscript{1}, Yi Jing\textsuperscript{1}, Shuhan Zhou\textsuperscript{1}, Xin Zeng\textsuperscript{1}
Tong Xiao\textsuperscript{1,2}, Jingbo Zhu\textsuperscript{1,2}, Xuebo Liu\textsuperscript{2} and Min Zhang\textsuperscript{3}

\textsuperscript{1}School of Computer Science and Engineering, Northeastern University, Shenyang, China
\textsuperscript{2}NiuTrans Research, Shenyang, China
\textsuperscript{3}Institute of Computing and Intelligence, Harbin Institute of Technology, Shenzhen, China
libei_neu@outlook.com, \{xiaotong,zhujingbo\}@mail.neu.edu.cn, \{liuxuebo,zhangmin2021\}@hit.edu.cn

Abstract

Residual networks are an Euler discretization of solutions to Ordinary Differential Equations (ODE). This paper explores a deeper relationship between Transformer and numerical ODE methods. We first show that a residual block of layers in Transformer can be described as a higher-order solution to ODE. Inspired by this, we design a new architecture, ODE Transformer, which is analogous to the Runge-Kutta method that is well motivated in ODE. As a natural extension to Transformer, ODE Transformer is easy to implement and efficient to use. Experimental results on the large-scale machine translation, abstractive summarization, and grammar error correction tasks demonstrate the high genericity of ODE Transformer. It can gain large improvements in model performance over strong baselines (e.g., 30.77 and 44.11 BLEU scores on the WMT’14 English-German and English-French benchmarks) at a slight cost in inference efficiency.

1 Introduction

Residual networks have been used with a great success as a standard method of easing information flow in multi-layer neural models (He et al., 2016; Vaswani et al., 2017). Given an input $y_t$, models of this kind define the output of a layer $t$ to be:

\begin{equation}
y_{t+1} = y_t + F(y_t, \theta_t) \tag{1}\end{equation}

where $F(\cdot, \cdot)$ is the function of the layer and $\theta_t$ is its parameter. Interestingly, recent work in machine learning (Weinan, 2017; Lu et al., 2018; Haber et al., 2018; Chang et al., 2018; Ruthotto and Haber, 2019) points out that Eq. (1) is an Euler discretization of the Ordinary Differential Equation (ODE), like this:

\begin{equation}\frac{dy(t)}{dt} = F(y(t), \theta(t)) \tag{2}\end{equation}

where $y(t)$ and $\theta(t)$ are continuous with respect to $t$. In this way, we can call Eq. (1) an ODE block.

This finding offers a new way of explaining residual networks in the view of numerical algorithms. Then, one can think of a multi-layer network as applying the Euler method (i.e., Eq. (1)) to solve Eq. (2) subject to the initial conditions $y(0) = y_0$ and $\theta(0) = \theta_0$.

The solution of Eq. (2) has a sufficiently low error bound (call it a stable solution) only if $\theta(t)$ changes slow along $t$ (Haber and Ruthotto, 2017; Chen et al., 2018). But this assumption does not always hold for state-of-the-art natural language processing (NLP) systems, in which models are non-linear and over-parameterized. For example, language modeling and machine translation systems learn quite different parameters for different layers, especially when the layers are close to the model input (Vaswani et al., 2017; Dai et al., 2019). Also, truncation errors are nonnegligible for the Euler method because it is a first-order approximation to the true solution (He et al., 2019). These problems make the situation worse, when more layers are stacked and errors are propagated through the neural network. It might explain why recent

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
\textbullet & $\theta_6$ & $\theta_5$ \\
\textcircled{O} & $\theta_4$ & $\theta_3$ \\
\hline
\end{tabular}
\end{center}

6 ODE blocks with 1st-order solutions

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
\textbullet & $\theta_6$ & $\theta_3$ \\
\textcircled{O} & $\theta_2$ & $\theta_1$ \\
\hline
\end{tabular}
\end{center}

3 ODE blocks with 2nd-order solutions

Figure 1: Models with different ODE blocks.
Machine Translation (MT) systems cannot benefit from extremely deep models (Wang et al., 2019; Liu et al., 2020a; Wei et al., 2020; Li et al., 2020). This paper continues the line of research on the ODE-inspired method. The basic idea is to use a high-order method for more accurate numerical solutions to the ODE. This leads to a larger ODE block that generates a sequence of intermediate approximations to the solution. We find that the larger ODE block is sufficient to take the role of several ODE blocks with first-order solutions. The benefit is obvious: the use of fewer ODE blocks lowers the risk of introducing errors in block switching, and the high-order method reduces the approximation error in each ODE block. See Figure 1 for a comparison of different models.

Our method is parameter-efficient because \( \theta(t) \) is re-used within the same ODE block. As another “bonus”, the model can be improved by learning coefficients of different intermediate approximations in a block. We evaluate our method in strong Transformer systems, covering both the wide (and big) model and the deep model. For machine translation tasks, ODE Transformer achieves 30.77 and 44.11 BLEU scores on the WMT’14 En-De and En-Fr test sets, setting a new state-of-the-art on the WMT’14 En-Fr task. It also significantly outperforms baselines on abstractive summarization and grammar error correction tasks.

2 Transformer and ODEs

We start with a description of Transformer, followed by its relationship with ODEs. We choose Transformer for our discussion and experiments because it is one of the state-of-the-art models in recent sentence generation tasks.

2.1 Transformer

Transformer is an example of the encoder-decoder paradigm (Vaswani et al., 2017). The encoder is a stack of identical layers. Each layer consists of a self-attention block and a feedforward network (FFN) block. Both of them equip with a residual connection and a layer normalization unit. Note that the term “block” is used in many different ways. In this paper, the term refers to any neural network that is enhanced by the residual connection (occasionally call it a residual block). Following the Pre-norm architecture (Wang et al., 2019), we define a block as

\[
y_{t+1} = y_t + G(\text{LN}(y_t), \theta_t)
\]

where \( \text{LN}(\cdot) \) is the layer normalization function, and \( G(\cdot) \) is either the self-attention or feedforward network. The decoder shares a similar architecture, having an additional encoder-decoder attention block sandwiched between the self-attention and FFN blocks.

2.2 Ordinary Differential Equations

An ordinary differential equation is an equation involving a function \( y(t) \) of a variable \( t \) and its derivatives. A simple form of ODE is an equation that defines the first-order derivative of \( y(t) \), like

\[
\frac{dy(t)}{dt} = f(y(t), t)
\]

where \( f(y(t), t) \) defines a time-dependent vector field if we know its value at all points of \( y \) and all instants of time \( t \). Eq. (4) covers a broad range of problems, in that the change of a variable is determined by its current value and a time variable \( t \). This formulation also works with Pre-norm Transformer blocks. For notational simplicity, we redefine \( G(\text{LN}(y_t), \theta_t) \) as a new function \( F(y_t, \theta_t) \):

\[
F(y_t, \theta_t) = G(\text{LN}(y_t), \theta_t)
\]

We then relax \( y_t \) and \( \theta_t \) to continuous functions \( y(t) \) and \( \theta(t) \), and rewrite Eq. (3) to be:

\[
y(t + \Delta t) = y(t) + \Delta t \cdot F(y(t), \theta(t))
\]

where \( \Delta t \) is the change of \( t \), and is general called step size. Obviously, we have \( \Delta t = 1 \) in Transformer. But we can adjust step size \( \Delta t \) using a limit, and have

\[
\lim_{\Delta t \to 0} \frac{y(t + \Delta t) - y(t)}{\Delta t} = F(y(t), \theta(t))
\]

Given the fact that \( \lim_{\Delta t \to 0} \frac{y(t + \Delta t) - y(t)}{\Delta t} = \frac{dy(t)}{dt} \), Eq. (7) is an instance of Eq. (4). The only difference lies in that we introduce \( \theta(t) \) into the right-hand side of Eq. (4). Then, we say that a Pre-norm Transformer block describes an ODE. It has been found that Eq. (3) shares the same form as the Euler method of solving the ODE described in Eq. (7) (Haber and Ruthotto, 2017). This establishes a relationship between Transformer and ODEs, in that, given \( F(\cdot, \cdot) \) and learned parameters \( \{\theta_t\} \), the forward pass of a multi-block Transformer is a process of running the Euler method for several steps.

\footnote{We drop the parameter of \( \text{LN}(\cdot) \) for simplicity.}
3 The ODE Transformer

In numerical methods of ODEs, we want to ensure the precise solutions to the ODEs in a minimum number of computation steps. But the Euler method is not “precise” because it is a first-order method, and naturally with local truncation errors. The global error might be larger if we run it for a number of times. This is obviously the case for Transformer, especially when the multi-layer neural network arises a higher risk of instability in solving the ODEs (Haber and Ruthotto, 2017).

3.1 High-Order ODE Solvers

Here we use the Runge-Kutta methods for a higher order solution to ODEs (Runge, 1895; Kutta, 1901; Butcher, 1996; Ascher and Petzold, 1998). They are a classic family of iterative methods with different orders of precision. More formally, the explicit Runge-Kutta methods of an n-step solution is defined to be:

\[ y_{t+1} = y_t + \sum_{i=1}^{n} \gamma_i F_i \]  
\[ F_1 = hf(y_t, t) \]  
\[ F_i = hf(y_t + \sum_{j=1}^{i-1} \beta_{ij} F_j, t + \alpha_i h) \]  

where \( h \) is the step size and could be simply 1 in most cases. \( F_i \) is an intermediate approximation to the solution at step \( t + \alpha_i h \). \( \alpha, \beta \) and \( \gamma \) are coefficients which can be determined by the Taylor series of \( y_{t+1} \) (Butcher, 1963). Eq. (10) describes a sequence of solution approximations \( \{F_1, ..., F_n\} \) over \( n \) steps \( \{t + \alpha_1 h, ..., t + \alpha_n h\} \). These approximations are then interpolated to form the final solution, as in Eq. (8).

The Runge-Kutta methods are straightforwardly applicable to the design of a Transformer block. All we need is to replace the function \( f \) (see Eq. (10)) with the function \( F \) (see Eq. (5)). The advantage is that the function \( F \) is re-used in a block. Also, the model parameter \( \theta_L \) can be shared within the block. In this way, one can omit \( t + \alpha_i h \) in Eq. (10), and compute \( F_i \) by

\[ F_i = F(y_t + \sum_{j=1}^{i-1} \beta_{ij} F_j, \theta_i) \]  

This makes the system more parameter-efficient. As would be shown in our experiments, the high-order Runge-Kutta methods can learn strong NMT systems with significantly smaller models.

The Runge-Kutta methods are general. For example, the Euler method is a first-order instance of them. For a second-order Runge-Kutta (RK2) block, we have

\[ y_{t+1} = y_t + \frac{1}{2}(F_1 + F_2) \]  
\[ F_1 = F(y_t, \theta_i) \]  
\[ F_2 = F(y_t + \frac{1}{2}F_1, \theta_i) \]

This is also known as the improved Euler method. Likewise, we can define a fourth-order Runge-Kutta (RK4) block to be:

\[ y_{t+1} = y_t + \frac{1}{6}(F_1 + 2F_2 + 2F_3 + F_4) \]  
\[ F_1 = F(y_t, \theta_i) \]  
\[ F_2 = F(y_t + \frac{1}{2}F_1, \theta_i) \]  
\[ F_3 = F(y_t + \frac{1}{2}F_2, \theta_i) \]  
\[ F_4 = F(y_t + F_3, \theta_i) \]

See Figure 2 for a comparison of different Runge-Kutta blocks. It should be noted that the method presented here can be interpreted from the perspective of representation refinement (Greff et al., 2017). It provides a way for a function to update the function itself. For example, Universal Transformer refines the representation of the input sequence using the same function and the same parameters in a block-wise manner (Dehghani et al., 2019). Here we show that inner block refinements can be modeled with good theoretical support.

3.2 Coefficient Learning

In our preliminary experiments, the RK2 and RK4 methods yielded promising BLEU improvements when the model was shallow. But it was found that the improvements did not persist for deeper models. To figure out why this happened, let us review the Runge-Kutta methods from the angle of training.
Take the RK2 method as an example. We rewrite Eq. (12) by substituting $F_1$ and $F_2$, as follow

\[
y_{t+1} = y_t + \frac{1}{2} F(y_t, \theta_t) + \frac{1}{2} F(y_t + F(y_t, \theta_t), \theta_t)
\]  \hspace{1cm} (20)

Let $\mathcal{E}$ be the loss of training, $L$ be the number blocks of the model, and $y_L$ be the model output. The gradient of $\mathcal{E}$ at $y_t$ is

\[
\frac{\partial \mathcal{E}}{\partial y_t} = \frac{\partial \mathcal{E}}{\partial y_L} \cdot \frac{1}{2^{L-i}} \cdot \prod_{k=t}^{L-1} (1 + g_k)
\]  \hspace{1cm} (21)

where

\[
g_k = \left(1 + \frac{\partial F(y_k, \theta_k)}{\partial y_k}\right) \cdot \left(1 + \frac{\partial F(y_k + F(y_k, \theta_k), \theta_k)}{\partial y_k + F(y_k, \theta_k)}\right)
\]  \hspace{1cm} (22)

Seen from Eq. (21), $\frac{\partial \mathcal{E}}{\partial y_t}$ is proportional to the factor $\frac{1}{2^{L-i}}$. This leads to a higher risk of gradient vanishing when $L$ is larger.

The problem somehow attributes to the small coefficients of $F_i$, that is, $\gamma_1 = \gamma_2 = \frac{1}{2}$. A natural idea is to empirically set $\gamma_1 = 1$ to eliminate the product factor of less than 1 in gradient computation, although this is not theoretically grounded in standard Runge-Kutta methods. We rewrite Eq. (20) with the new coefficients, as follows

\[
y_{t+1} = y_t + F(y_t, \theta_t) + F(y_t + F(y_t, \theta_t), \theta_t)
\]  \hspace{1cm} (23)

Then, we have the gradient, like this

\[
\frac{\partial \mathcal{E}}{\partial y_{t}} = \frac{\partial \mathcal{E}}{\partial y_L} \cdot \prod_{k=t}^{L-1} g_k
\]  \hspace{1cm} (24)

This model is easy to optimize because $\frac{\partial \mathcal{E}}{\partial y_{t}}$ can be passed to lower-level blocks with no scales. Note that, the methods here are instances of parameter sharing (Dehghani et al., 2019; Lan et al., 2020). For example, in each ODE block, we use the same function $F$ with the same parameter $\theta_t$ for all intermediate steps. Setting $\gamma_i = 1$ is a further step towards this because $F_i$ is passed to the following computations with the same scale. Here we call it implicit parameter sharing.

Another way of scaling $F_i$ to further improve ODE functions is to learn the coefficients automatically on the training data. The simplest method is to initialize $\gamma_i = 1$ and independently optimize each scale. It helps the system learn the way of flowing $F_i$ in a block. Based on it, scaling $F_i$ by a weighted gate mechanism (Srivastava et al., 2015) empirically achieves the best performance (see Section 4). Take RK2-block as an instance, the concatenation of $F_1$ and $F_2$ is transformed to a scalar $(0, 1)$ through a sigmoid gate, then the block output $y_{t+1}$ is

\[
y_{t+1} = y_t + g \cdot F_1 + (1 - g) \cdot F_2
\]  \hspace{1cm} (25)

\[
g = \text{sigmoid}([F_1, F_2] \cdot W + b)
\]  \hspace{1cm} (26)

where $[,]$ denotes the concatenation operation and $W, b$ are learnable parameters. We call it RK2-block (learnable $\gamma_i$), and the architecture is shown in Figure 2 (d). This kind of formulation offers a more flexible way to decide which part contributes more and is also easy to be optimized. Moreover, we also summarize the comparison of various scaling functions in Appendix C.
| Model                              | Layers | WMT En-De | WMT En-Fr |
|-----------------------------------|--------|-----------|-----------|
|                                   | #Param | Steps     | BLEU      | #Param | Steps     | BLEU      |
| Transformer (Vaswani et al., 2017) | 6-6    | 213M 100K | 28.40     | -      | 222M 300K | 41.00     |
| MacaronNet (Lu et al., 2019)      | 6-6    | -         | 30.20     | -      | -         | -         |
| Depth growing (Wu et al., 2019)   | 8-8    | 270M 800K | 29.92     | -      | -         | 43.27     |
| Transformer-DLCL (Wang et al., 2019) | 30-6  | 137M 50K  | 28.6      | -      | -         | -         |
| Multiscale Collaborative (Wei et al., 2020) | 18-6  | 512M 300K | 30.56     | -      | -         | -         |
| ADMIN (Liu et al., 2020a)         | 60-12  | 262M 250K | 29.5      | 250K   | 43.80     | 41.8      |
| SDT (Li et al., 2020)             | 48-6   | 192M 50K  | 30.21     | 198M   | 43.28     | 41.5      |
| BERT-fused model (Zhu et al., 2020) | -     | 30.75    | -         | -      | 43.78     | -         |

Wide Models

| Model                              | Layers | WMT En-De | WMT En-Fr |
|-----------------------------------|--------|-----------|-----------|
|                                   | #Param | Steps     | BLEU      | #Param | Steps     | BLEU      |
| Residual-block                     | 6-6    | 61M 50K   | 27.89     | 26.8   | 69M 100K  | 41.05     |
| RK2-block                          | 6-6    | 61M 50K   | 28.67     | 27.5   | 69M 100K  | 42.08     |
| RK2-block (learnable $\gamma_i$) | 6-6    | 61M 50K   | 28.89     | 27.7   | 69M 100K  | 42.31     |
| RK4-block                          | 6-6    | 61M 50K   | 29.03     | 27.9   | 69M 100K  | 42.56     |
| Residual-block                     | 24-6   | 118M 50K  | 29.43     | 28.3   | 123M 100K | 42.67     |
| RK2-block                          | 24-6   | 118M 50K  | 29.85     | 28.7   | 123M 100K | 43.04     |
| RK2-block (learnable $\gamma_i$)  | 24-6   | 118M 50K  | 30.29     | 29.2   | 123M 100K | 43.48     |
| RK4-block                          | 24-6   | 118M 50K  | 29.80     | 28.8   | 123M 100K | 43.28     |

Wide Models

| Model                              | Layers | WMT En-De | WMT En-Fr |
|-----------------------------------|--------|-----------|-----------|
|                                   | #Param | Steps     | BLEU      | #Param | Steps     | BLEU      |
| Residual-block                     | 6-6    | 211M 100K | 29.21     | 28.1   | 221M 100K | 42.89     |
| RK2-block                          | 6-6    | 211M 100K | 30.11     | 29.0   | 221M 100K | 43.34     |
| RK2-block (learnable $\gamma_i$)  | 6-6    | 211M 100K | 30.53     | 29.4   | 221M 100K | 43.59     |
| RK4-block                          | 6-6    | 211M 100K | 30.39     | 29.3   | 221M 100K | 43.55     |
| Residual-block                     | 12-6   | 286M 100K | 29.91     | 28.9   | 297M 100K | 43.22     |
| RK2-block                          | 12-6   | 286M 100K | 30.58     | 29.4   | 297M 100K | 43.88     |
| RK2-block (learnable $\gamma_i$)  | 12-6   | 286M 100K | 30.77     | 29.6   | 297M 100K | 44.11     |
| RK4-block                          | 12-6   | 286M 100K | 30.55     | 29.4   | 297M 100K | 43.81     |

Table 1: Comparison with the state-of-the-arts on the WMT En-De and WMT En-Fr tasks. We both report the tokenized BLEU and SacreBLEU scores for comparison with previous work.

3.3 Efficiency Discussion

ODE Transformer is efficient to use. As we only apply the ODE design schema to the encoder side, it only brings minor impacts on the inference speed due to the autoregressive decoding schema. Another concern here is memory consumption. ODE Transformer consumes more memory than the baseline in the same depth since we need to store the intermediate approximations in the forward pass. But the additional consumption is less than that of the baseline who has the same computation cost, which is acceptable for most scenarios. We give a quantitative analysis in Section 5.

4 Experiments

We evaluated the ODE Transformer on three sequence generation tasks: machine translation, abstractive summarization and grammar error correction. The datasets we used are elaborated in the following section, and more details of experimental setups could be found in Appendix A and B.

4.1 Datasets

Machine Translation We report results on three WMT benchmarks. For the WMT’14 English-German (En-De) task, the training data consisted of approximately 4.5M tokenized sentence pairs, as in (Vaswani et al., 2017). All sentences were segmented into sequences of sub-word units (Sennrich et al., 2016) with 32K merge operations using a shared vocabulary. We selected newstest2013 as the validation data and newstest2014 as the test data. For the WMT’14 English-French (En-Fr) task, we used the dataset provided within Fairseq, i.e., 36M training sentence pairs from WMT’14. newstest2012+newstest2013 was the validation data and newstest2014 as the test data. For the WMT’14 English-Romanian (En-Ro) task, we replicated the setup of (Mehta et al., 2020), which used 600K/2K/2K sentence pairs for training, evaluation and inference, respectively.

Abstractive Summarization We also tested the models’ ability to process long sequences on the CNN-DailyMail summarization task (Nallapati et al., 2016; Hermann et al., 2015). The prepro-
Language Modeling  The truncation error analysis is conducted on the Penn Treebank (Mikolov et al., 2011), which is a widely-used language model dataset. It contains 88K, 3,370 and 3,761 sentences for training, validation and test. The vocabulary size was 10K. We set the layer depth of the language model to 1 or 2 to make a fair comparison. Assume the layer depth is 1, then the loss between the block output and the ground-truth could be regarded as the truncation error. It alleviates the influence of the error accumulation across different layers.

4.2 Experimental Results

Results of En-De and En-Fr  Table 1 compares ODE Transformer with several state-of-the-art systems. Both RK2-block and RK4-block outperform the baselines by a large margin with different model capacities. For example, RK2-block obtains a +1.00 BLEU improvement with the base configuration when the depth is 6. RK4-block yields a gain of 0.17 BLEU points on top of RK2-block. This observation empirically validates the conjecture that high-order ODE functions are more efficient.

When we switch to deep models, our method is more parameter efficient. E.g., RK2-block is comparable with a strong 48-layer system (Li et al., 2020) with half of the encoder depth. Similarly, wide models can also benefit from the enlarging layer depth (Wei et al., 2020; Li et al., 2020). RK2-block achieves BLEU scores of 30.77 and 44.11 on the En-De and the En-Fr tasks, significantly surpassing the standard Big model by 1.32 and 0.70 BLEU points. This sets a new state-of-the-art on these tasks with fewer parameters.

Results of En-Ro  Table 2 exhibits model parameters, total training steps and BLEU scores of several strong systems on the En-Ro task. Again, ODE Transformer outperforms these baselines. As stated in (Mehta et al., 2020), they trained the model up to 170 epochs and obtained a BLEU score of 34.70 through the DeLight model. However, the observation here is quite different. The validation PPL begins to increase after 20 epochs. Thus, our baseline is slightly inferior to theirs, but matches the result reported in Lin et al. (2020). ODE blocks achieve even better performance with DeLight within much less training cost. For a bigger model (line 6), it obtains a BLEU score of 35.28.

Parameter Efficiency  Table 3 summaries the results of several efficient Transformer variants, including Lite Transformer (Wu et al., 2020), DeLight (Mehta et al., 2020) and a light version of the Evolved Transformer (So et al., 2019). As expected, ODE Transformer is promising for smaller models. It is comparable in BLEU with DeLight but having 9M fewer parameters. Under the same model capacity, it outperforms Delight by 0.64 BLEU points. It may offer a new choice for deploying NMT systems on edge devices.

Results of Summarization and Correction  We also evaluated the ODE Transformer on another two sequence generation tasks. Table 4 shows that both RK2-block and RK4-block outperform the

| Model                     | Params | Epochs | BLEU |
|---------------------------|--------|--------|------|
| Transformer in Mehta et al. (2020) | 62M    | 170    | 34.30|
| DeLight (Mehta et al., 2020)     | 53M    | 170    | 34.70|
| Int Transformer' (Lin et al., 2020) | -      | -      | 32.60|
| Transformer (Our impl.)        | 69M    | 20     | 33.49|
| RK2-block (learnable $\gamma_i$) | 69M    | 20     | 34.94|
| RK2-block-Big (learnable $\gamma_i$) | 226M   | 20     | **35.28**|

Table 2: Results on the WMT En-Ro task. † indicates the related information is not reported.

| Model                     | Params | BLEU |
|---------------------------|--------|------|
| Transformer (Vaswani et al., 2017) | 62M    | 27.30|
| Evolved Transformer (So et al., 2019) | 46M    | 27.70|
| Lite Transformer' (Wu et al., 2020) | -      | 26.50|
| DeLight (Mehta et al., 2020)     | 37M    | 27.60|
| RK2-block (learnable $\gamma_i$, $H=256$, $L=28$) | 37M    | **28.24**|
| RK2-block (learnable $\gamma_i$, $H=256$, $L=18$) | 29M    | 27.84|

Table 3: The comparison of model efficiency on the WMT En-De task.
baselines by a margin. Similarly, RK4-block is superior to RK2-block when the model is shallow. More results and case studies could be found in Appendix C.

5 Analysis

Here we investigate some interesting issues. For simplicity, we call RK2-block with coefficients initialized by 1 as RK2-block-v1, and learnable coefficients (Eq. (25)) as RK2-block-v2.

Quantization of the Truncation Error In fact, we cannot obtain the “true” solution of each block output in NMT, because we mainly experimented on the encoder side. Instead, we tested our system on the language modeling task, where the perplexity between the single-layer model output and the ground truth could be regarded as the truncation error with no error propagations. Table 5 shows the perplexities on the Penn Treebank dataset (Mikolov et al., 2011). All ODE Transformer variants reduce the errors significantly. RK4-order achieves the lowest PPL on both settings. In addition, RK2-block can even obtain a lower PPL than a 2-layer residual-block. The observation here again verifies larger ODE blocks behave superior to the standard residual block.

Inference Speed and Memory Consumption Table 6 shows the comparison of inference speed and memory consumption discussed in Section 3.3. Experimental results demonstrate the proposed ODE design schema results in acceptable inference speeds. And it is also memory-friendly through the memory comparison between the baseline and the RK variants in both base and big configurations.

BLEU against Encoder Depth Figure 3 (left) depicts BLEU scores of several ODE Transformer variants and the baseline under different encoder depths. All ODE Transformer variants are significantly superior to the baseline when depth $\leq 24$. RK2-block-v2 almost achieves the best performance over all depths, especially when the model becomes deeper. Interestingly, Figure 3 confirms again that ODE Transformer is parameter efficient, e.g., a 6-layer RK2-block is comparable with the 18-layer baseline system. Another finding here is RK4-block performs well on shallow models, but it is inferior to RK2-block when the depth is going deep. This is because original coefficients may cause the optimization problem in the backward propagation in deep models (see Section 3.2). Also, Figure 3 (right) plots BLEU as a function of the model size when the hidden size is $256$. The RK2 method significantly surpasses the baseline using much fewer parameters.

Ablation Study on Different $F(\cdot, \cdot)$ As stated in Section 3, the $F(\cdot, \cdot)$ function can either be SAN, FFN or both of them (SAN+FFN). As shown in Figure 4, high-order ODE works better with FFN than SAN. An explanation might be that the FFN component has more parameters than the SAN component. The model that treats FFN and SAN as a single ODE block behaves the best.

Training and Validation Perplexity Figure 5 plots the training and validation PPL curves of RK blocks and the baseline enhanced by RPR (Shaw et al., 2018). RK2-block obtains lower training and validation PPLs in both configurations (base and wide models).

5 There are $2 \cdot d_{\text{model}} \cdot 4d_{\text{model}}$ parameters in FFN and $d_{\text{model}} \cdot 3d_{\text{model}} + d_{\text{model}} \cdot d_{\text{model}}$ in SAN.
Encoder Depth
BLEU RK2-block
RK2-block-v1
RK4-block

Figure 3: The comparison of BLEU against different encoder depth and the number of model parameters.

Figure 4: BLEU scores [%] of several $F(\cdot, \cdot)$ on the WMT En-De task.

**Visualization of the Gradient Norm** We also collect the gradient information of several well-trained systems during training. Figure 6 plots the gradient norm of RK2-block-v2, RK4-block and the standard residual-block (baseline). As we can see that Pre-Norm residual block is able to make the training stable (Wang et al., 2019). Both RK2-block-v2 and RK4-block provide richer signals due to the implicit parameter sharing among intermediate approximations. The two learning curves appear to be nearly the same, which is consistent with the results in Table 1.

**Comparison of Different ODE Design Schemas** Then, we take a comprehensive analysis of several ODE design schemas. As stated in Lu et al. (2018)’s work, several models in computer vision, such as LeapfrogNet (He et al., 2019), PolyNet (Zhang et al., 2017) and MultistepNet (Lu et al., 2018), can also be interpreted from the ODE perspective. The related ODE functions are summarized in Table 7. We re-implemented these methods using the same codebase for fair comparisons. We conducted experiments following the base configuration on the En-De task.

At the time $t$, Multistep Euler methods require previous states, e.g. $y_{t-1}$, to generate the current approximation, instead of iterative refinements based on the current-time state. So these methods are heavier than ODE Transformer. Note that DLCL (Wang et al., 2019) can also be regarded as a multistep Euler method, which is more competitive in deep Transformer. But there is just a modest improvement upon the shallow baseline. Theoretically, the Backward Euler method is slightly better than the Forward Euler method in numerical analysis, but the improvement is marginal. Note that our ODE Transformer achieves consistent BLEU improvements over the aforementioned methods. The reason is that such iterative refinements provide more efficient and effective parameter learning.

**6 Related Work**

**Deep Transformer models** Recently, deep Transformer has witnessed tremendous success in machine translation, especially on WMT news tasks (Li et al., 2019; Zhang et al., 2020; Zhou et al., 2021; Tran et al., 2021). A straightforward way is to shorten the path from upper-level layers to lower-level layers thus to alleviate the gradient vanishing or exploding problems (Bapna et al., 2018; Wang et al., 2019; Wu et al., 2019; Wei et al., 2020). For deeper models, the training cost is nonnegligible. To speed up the training, an alternative way is to train a shallow model first and progressively increase the model depth (Li et al., 2020; Dong et al., 2020). Apart from the model architecture improvements, another way of easing the optimization is to utilize carefully designed parameter initialization strategies (Zhang et al., 2019; Xu et al., 2020; Huang et al., 2020; Liu et al., 2020a). With the model capacity going larger, one can use Layer-
Drop (Fan et al., 2020) or Skipping Sublayers (Li et al., 2021) to prevent deep models from the overfitting problem. Note that ODE Transformer is orthogonal to the aforementioned methods, and we will test it on these methods in future work.

**Ordinary Differential Equations** The relationship between ResNet and ODEs was first proposed by Weinan (2017). This shows a brand-new perspective on the design of effective deep architectures. Moreover, the success of Neural ODENet (Chen et al., 2018) has attracted researchers. Some insightful architectures (Zhang et al., 2017; Lars-son et al., 2017; Lu et al., 2018; He et al., 2019; Zhu and Fu, 2018; Lu et al., 2019; Sander et al., 2021) can also be interpreted from the ODE perspective. But, in NLP, it is still rare to see studies on designing models from the ODE perspective. Zhang et al. (2021) proposed continuous self-attention models using the same merit with neural ODE. Perhaps the most relevant work with us is an (2021)’s work. They redesigned the Transformer architecture from a multi-particle dynamic system view in terms of efficiency. Unlike them, we show that the stacked first-order ODE blocks may cause error accumulation, thus hindering the model performance. We address this issue by introducing high-order blocks, and demonstrate significant performance improvements on three sequence generation tasks, which is complementary to Baier-Reinio and De Sterck (2020)’s work.

### 7 Conclusions

This paper explores the relationship between Transformer and ODEs. We propose ODE Transformer to help the model benefit from high-order ODE solutions. Experimental results on the three representative sentence generation tasks (i.e., machine translation, abstractive summarization, and grammatical error correction) show the effectiveness and efficiency of ODE Transformer. It achieves 30.77 and 44.11 BLEU scores on the WMT’14 En-De and En-Fr benchmarks, setting a new state-of-the-art result on the En-Fr. Note that our code is publicly available at https://github.com/libeineu/ODE-Transformer.

### Acknowledgments

This work was supported in part by the National Science Foundation of China (Nos. 61732005 and 61876035), the National Key R&D Project of China (No. 2019QY1801), the China HTRD Center Project (No. 2020AAA0107904) and Yunnan Provincial Major Science and Technology Special Plan Projects (Nos. 201902D08001905 and 202103AA080015). The authors would like to thank anonymous reviewers for their valuable comments. And thank Yufan Jiang for his helpful advice to improve the paper.

### References

Subhabrata Dutta an. 2021. Redesigning the transformer architecture with insights from multi-particle. ArXiv preprint, abs/2109.15142.

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

Aaron Baier-Reinio and Hans De Sterck. 2020. *N-ode transformer: A depth-adaptive variant of the transformer using neural ordinary differential equations*. ArXiv preprint, abs/2010.11358.

Ankur Bapna, Mia Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. 2018. Training deeper neural machine translation models with transparent attention. In *Proceedings of the 2018 Conference on
Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multi-layer convolutional encoder-decoder neural network for grammatical error correction. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 2811–2818. AAAI Press.

Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. 2018. Reversible architectures for arbitrarily deep residual neural networks. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 2811–2818. AAAI Press.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multi-layer convolutional encoder-decoder neural network for grammatical error correction. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 5755–5762. AAAI Press.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications, pages 22–31, Atlanta, Georgia. Association for Computational Linguistics.

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

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019. OpenReview.net.

Chengyu Dong, Liyuan Liu, Zichao Li, and Jingbo Shang. 2020. Towards adaptive residual network training: A neural-ode perspective. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pages 2616–2626. PMLR.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4098–4109, Brussels, Belgium.

Klaus Greff, Rupesh Kumar Srivastava, and Jürgen Schmidhuber. 2017. Highway and residual networks learn unrolled iterative estimation. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net.

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

Eldad Haber, Lars Ruthotto, Elliot Holtham, and Seong-Hwan Jun. 2018. Learning across scales - multiscale methods for convolution neural networks. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 3142–3148. AAAI Press.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 770–778. IEEE Computer Society.

Xiangyu He, Zitao Mo, Peisong Wang, Yang Liu, Mingyuan Yang, and Jian Cheng. 2019. Ode-inspired network design for single image super-resolution. In IEEE Conference on Computer Vision...
Abstractive text summarization using sequence-to-sequence RNNs and beyond. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pages 4475–4483. PMLR.

Wilhelm Kutta. 1901. Beitrag zur naherungsweisen Integration totaler differentialgleichungen. Z. Math. Phys., 46:435–453.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2017. FractalNet: Ultra-deep neural networks without residuals. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net.

Bei Li, Ziyang Wang, Hui Liu, Yufan Jiang, Quan Du, Tong Xiao, Huizhen Wang, and Jingbo Zhu. 2020. Shallow-to-deep training for neural machine translation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 995–1005, Online. Association for Computational Linguistics.

Yeping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Understanding and improving transformer from a multi-particle dynamic system point of view. ArXiv preprint, abs/1906.02762.

Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. 2018. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research, pages 3282–3291. PMLR.

Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2020. Delight: Very deep and light-weight transformer. ArXiv preprint, abs/2008.00623.

Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan Černocký. 2011. Empirical evaluation and combination of advanced language modeling techniques. In Twelfth annual conference of the international speech communication association.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos Santos, Çağlar Gülcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In Proceedings of the...
20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016, pages 280–290. ACL.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Carl Runge. 1895. Über die numerische auflösung von differentialgleichungen. Mathematische Annalen, 46(2):167–178.

Lars Ruthotto and Eldad Haber. 2019. Deep neural networks motivated by partial differential equations. Journal of Mathematical Imaging and Vision, 62:352–364.

Michael E. Sander, Pierre Ablin, Mathieu Blondel, and Gabriel Peyré. 2021. Momentum residual neural networks. In Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, volume 139 of Proceedings of Machine Learning Research, pages 9276–9287. PMLR.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1073–1083, Vancouver, Canada.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

David R. So, Quoc V. Le, and Chen Liang. 2019. The evolved transformer. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, pages 5877–5886. PMLR.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. ArXiv preprint, abs/1505.00387.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 198–202, Jeju Island, Korea. Association for Computational Linguistics.

Chau Tran, Shruti Bhosale, James Cross, Philipp Koehn, Sergey Edunov, and Angela Fan. 2021. Facebook AI’s WMT21 news translation task submission. In Proceedings of the Sixth Conference on Machine Translation, pages 205–215, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. 2019. Learning deep transformer models for machine translation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1810–1822, Florence, Italy. Association for Computational Linguistics.

Xiangpeng Wei, Heng Yu, Yue Hu, Yue Zhang, Rongxiang Weng, and Weihua Luo. 2020. Multiscale collaborative deep models for neural machine translation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 414–426, Online. Association for Computational Linguistics.

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

Lijun Wu, Yiren Wang, Yingce Xia, Fei Tian, Fei Gao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2019. Depth growing for neural machine translation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5558–5563, Florence, Italy. Association for Computational Linguistics.

Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. 2020. Lite transformer with long-short range attention. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Hongfei Xu, Qihui Liu, Josef van Genabith, Deyi Xiong, and Jingyi Zhang. 2020. Lipschitz constrained parameter initialization for deep transformers. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 397–402, Online. Association for Computational Linguistics.
A new dataset and method for automatically grading ESOL texts. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2019. Improving deep transformer with depth-scaled initialization and merged attention. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 898–909, Hong Kong, China. Association for Computational Linguistics.

Jing Zhang, Peng Zhang, Baiwen Kong, Junquiu Wei, Xingcheng Zhang, Zhizhong Li, Chen Change Loy, Biao Zhang, Ivan Titov, and Rico Sennrich. 2019. Incorporating BERT into neural machine translation. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Mai Zhu and Chong Fu. 2018. Convolutional neural networks combined with runge-kutta methods. CoRR, abs/1802.08831.

### A Experimental Setups

Table 8 summarizes the details of our datasets. We both present the sentences and tokens of each task. For the En-De and En-Fr tasks, the datasets used in this work could be found in Fairseq. For the En-Ro task, we used the preprocessed dataset provided by Delight. Note that we only shared the target embedding and the softmax embedding instead of a shared vocabulary between the source side and the target side. The CNN/DailyMail dataset consists of CNN stories and Daily emails. For the grammar error correction task (GEC), we conducted experiments on the CONLL dataset.

### B Training and Evaluation

**Training** As suggested in Li et al. (2020)’s work, we adopted relative positional representation (RPR) (Shaw et al., 2018) for stronger baselines. Dense connections among layers (Wang et al., 2019) are also applied for stable learning since the model is optimized with FP16 training. All experiments were trained on 8 GPUs with 4,096 tokens on each GPU. For the En-De and the En-Fr tasks, we employed the gradient accumulation strategy with a step of 2 and 8, respectively. We used the Adam optimizer (Kingma and Ba, 2015) whose hyperparameters were set to (0.9, 0.997). The hyperparameters including the learning rate, the warmup step and the total training steps of three tasks could be found in Table 8. Note that we trained Base/Deep and Big models for 50K and 100K steps on the En-De task.

We regarded merging SAN and FFN as the default ODE block. In addition, main results were the average of three times running with different random seeds.

6https://github.com/pytorch/fairseq/tree/master/examples/scaling_nmt
7https://github.com/sacmehta/delight/blob/master/README_files/nmt/wmt16_enro.md
8https://drive.google.com/uc?export=download&id=0Bwmd_VLjROrftTHk4NPf2SmdKcjQ
9https://drive.google.com/uc?export=download&id=0Bwmd_VLjROrM18dxkkVaiTY2bW
10https://www.cl.cam.ac.uk/research/nl/pea2019st
Table 8: Statistics of the datasets and hyperparameters for three sequence generation tasks. For the dataset, we report the vocabulary size, sentence numbers of training, validation and test sets. For the training, Lr denotes the peaking learning rate and Warmup denotes the warmup step of the Adam optimizer. WD denotes whether we applied word dropout. For the inference, Beam and LP denote the beam size and length penalty, respectively.

| Dataset       | Vocab | Dataset | Train | Dev  | Test  | Lr   | Warmup | Batch | Steps | WD | Beam | LP |
|---------------|-------|---------|-------|------|-------|------|--------|-------|-------|----|------|----|
| WMT'14 En-De  | 34040 | 4.5M    | 3000  | 3003 |       | 0.002| 16000  | 80K   | 50K   | ×  | 4    | 0.6|
| WMT'14 En-Fr  | 44424 | 35.7M   | 26822 | 3003 |       | 0.002| 16000  | 320K  | 100K  | ×  | 4    | 0.6|
| WMT'16 En-Ro  | 34976 | 602K    | 1999  | 1999 |       | 0.002| 8000   | 80K   | 17K   | ×  | 5    | 1.3|
| CNN/DailyMail | 32584 | 287K    | 13368 | 11490|       | 0.002| 8000   | 160K  | 50K   | ×  | 4    | 2.0|
| CONLL         | 33136 | 827K    | 5448  | 1312 |       | 0.0015| 4000   | 160K  | 15K   | ✓  | 6    | 0.6|

seeds, and we averaged the last 5/10 checkpoints for fair comparisons with previous work. The detail of Base/Deep/Wide configurations is as follows:

- **Base/Deep Model.** The hidden size of self-attention was 512, and the dimension of the inner-layer in FFN was 2,048. We used 8 heads for attention. For training, we set all dropout to 0.1 as default, including residual dropout, attention dropout, ReLU dropout. Label smoothing $\epsilon_{ls} = 0.1$ was applied to enhance the generation ability of the model. For deep models, we only enlarged the encoder depth considering the inference speed.

- **Wide (or Big) Model.** We used the same architecture as Transformer-Base but with a larger hidden layer size 1,024, more attention heads (16), and a larger feed forward inner-layer (4,096 dimensions). The residual dropout was set to 0.3 for the En-De task and 0.1 for the En-Fr task.

For the language modeling task, the hidden size was 512, and the filter size of the FFN was 2,048. We set all the dropout rates as 0.1, including the residual dropout, attention dropout and ReLU dropout. Each model was trained up to 20 epochs, and most models achieved the lowest PPL on the validation set when the epoch is 10. Then the validation PPL began to increase, though the training PPL is still declining. The warmup step was 2,000 and the batch size was 4,096. The max learning rate was set to 0.0007.

**Evaluation** For machine translation, we measured performance in terms of BLEU. Both tokenized BLEU and SacreBLEU\(^\text{11}\) scores were reported on the En-De and En-Fr tasks. Also, we reported tokenized BLEU scores on the En-Ro task. In addition, we measured Rouge-1, Rouge-2, Rouge-L for CNN/DailyMail and precision, recall, $F_{0.5}$ for CONLL. The beam size and length penalty of each task are summarized in Table 8.

C Additional Results and Analyses

**Comparison on the CNN/DailyMail Dataset**

We summarize the previous results on the CNN/DailyMail dataset (See Table 9). The performance was evaluated by ROUGE-1, ROUGE-2 and ROUGE-L, respectively. Intuitively, high-order ODE functions can significantly improve on top of the Euler method as well as several strong existing models.\(^\text{12}\) Again, RK4-block beats the baseline and RK2-block by up to 1.36 and 0.25 scores in terms of ROUGE-1, respectively.

**Comparison of Various Scaling Methods** We have emphasized the importance of automatic coefficient learning in Section 3.2. The forward pass of RK2-block can be described as $y_{t+1} = y_t + \gamma_1 \cdot F_1 + \gamma_2 \cdot F_2$, where $\gamma_1$ and $\gamma_2$ are coefficients which can be numerical suggested or learnable. Here we exhibit the comparison of various scaling methods on the WMT’14 En-De dataset, and the results are listed in Table 10. We can see that RK2-block (learnable $\gamma_i$) equips with a single sigmoid gate (line 5 in Table 10) yields best results on both shallow and deep configurations. The observation here reveals that appropriate scaling functions can further improve the RK2-block. Tanh activation even brings negative impacts on the performance, especially when the model is deep. A possible explanation is that Tanh produces a larger

\(^{11}\text{BLEU}+\text{case.mixed}+\text{numrefs.1}+\text{smooth.exp}+\text{tok.13a+version.1.2.12}\)

\(^{12}\text{We only compared models without using pre-training.}\)
Table 9: ROUGE scores of various models on the CNN/DailyMail dataset.

| Model | ROUGE-1 | ROUGE-2 | ROUGE-L |
|-------|---------|---------|---------|
| LEAD3 | 40.24   | 17.70   | 36.45   |
| NEUSUM (Zhou et al., 2018) | 41.59   | 19.01   | 37.98   |
| PGNet (See et al., 2017) | 39.53   | 17.28   | 36.38   |
| Soft Fusion (Liu et al., 2020b) | 41.00   | 18.30   | 37.90   |
| Bottom-Up Summarization (Gehrmann et al., 2018) | 41.22   | 18.68   | 38.34   |
| Residual-block | 40.47   | 17.73   | 37.29   |
| RK2-block | 41.58   | 18.57   | 38.41   |
| RK4-block | **41.83** | **18.84** | **38.68** |

Table 10: Comparison of various scaling functions on the WMT14’ En-De dataset.

| Model | $\gamma_1$ | $\gamma_2$ | 6-layer | 24-layer |
|-------|------------|------------|---------|----------|
| weight sharing | 1 | 1 | 28.51 | 29.60 |
| RK2-block | 1/2 | 1/2 | 28.67 | 29.85 |
| RK2-block ($\gamma_i = 1$) | 1 | 1 | 28.77 | 30.01 |
| RK2-block (learnable $\gamma_i = 1$) | scalar | scalar | 28.80 | 30.13 |
| RK2-block (learnable $\gamma_i$) | sigmoid | sigmoid | 28.74 | 30.06 |
| RK2-block (learnable $\gamma_i$) | sigmoid | (1 - sigmoid) | 28.86 | 30.29 |
| RK2-block (learnable $\gamma_i$) | tanh | tanh | 28.45 | 29.47 |

Case Study on the GEC Task  Table 11 summarizes several cases from the GEC task. Here, we make a comparison between the baseline and the RK4-block due to its superiority on the GEC task. We can clearly see that the proposed RK4-block delivers more accurate corrections compared with the baseline when handling subject-verb agreement (Case2), collocation (Case1, Case3), spelling (Case4) and other issues. More specifically, Figure 7 illustrates the statistics of different error types annotated by ERRANT (Bryant et al., 2017), a grammatical ERRor ANnotation Toolkit designed to automatically annotate parallel error correction data. For more details please refer to Bryant et al. (2017)’s work. With the help of ERRANT, we can carry out a detailed error type analysis. As shown in Figure 7, RK4-block corrects the input in a more similar way with the reference, though there is still a large gap between them. Limited by the model ability, the baseline sometimes even cannot generate the right corrections, e.g. R:PUNCT and M:OTHER cases.

D Comparison with Related Work

As we aforementioned, the ODE design schema somehow shares a similar merit with the weight sharing, especially when the coefficients are set to 1. This is because we reuse the same function $F$ to compute the intermediate approximation at each timestep, and it is also an effective way to apply the higher-order ODE into the Transformer architecture. Compared with weight sharing (line 1 in Table 10), ODE Transformer variants can deliver better performance within the same computation cost, demonstrating the effectiveness of ODE design schema.

Next, we make a detailed comparison between the proposed ODE Transformer and previous studies (Baier-Reinio and De Sterck, 2020; Zhu and Fu, 2018; Zhang et al., 2021) to avoid the potential misunderstandings.

Compared with RKNet  RKNet (Zhu and Fu, 2018) is mainly designed to improve the ResNet using implicit Runge-Kutta methods for vision tasks. There are some differences between ours and RKNet. (i) We mainly conduct experiments on sequence generation tasks, e.g. machine translation, abstract summarization, and grammar error correction tasks. They focused on the image clas-
What’s more, various cultures can be shown to us through social medias.

Social media sites such as Facebook have allowed us to share our pictures or even chat online with our parents while we are overseas.

On one hand, it is obviously that many advantages have been brought to our lives.

Other than that, I believe that the strong bond we have with our family is the biggest pillar of support to the carrier.

Table 11: Several examples from the GEC task. Here, source and reference denote the model input and the correction result, respectively. Green words are good corrections, while Red words are bad corrections.

Figure 7: Statistics of different error type information.
sification task. (ii) Except for the integration of ODE into the Transformer design schema, we also make an analysis on how to choose appropriate coefficients of intermediate approximations. And we bridge the relationship between the ODE design schema with the explicit weight sharing. (iii) We also offer an automatic coefficient learning method for RK2-block which delivers the best performance in different configurations.

**Compared with N-ODE** As we discussed in the related work, our work is complementary to Baier-Reinio and De Sterck (2020)’s work. We empirically demonstrate the effectiveness of integrating ODE design schema into Transformer on several sequence generation tasks. This work may shed light on the design of effective Transformer architectures from the numerical perspective and provides stronger baselines to the literature.

**Compared with CSAODE** The differences between these two works are summarized below: (i) As we emphasized above, the benchmarks we experimented on are quite different. They mainly validated the proposed CSAODE on text classification and QA tasks. (ii) The proposed CSAODE (Zhang et al., 2021) is an extension of neural ODE (cheng et al., 2018), where the motivation is quite different. They aim to effectively calculate the contiguous states of hidden features only via one-layer parameters and proposed a self-attention solver to fix the issue. While our motivation is to employ higher-order ODE solutions to reduce the truncation errors produced by each layer. On the other hand, CSAODE is still a single-layer model, and ours is a multi-layer sequence-to-sequence model. We also show the comparison of different components based on higher-order ODE solutions (See Figure 4). (iii) The single-layer model is not strong enough to solve complicated tasks, e.g. machine translation. However, when stacking several layers, we need to re-consider the error accumulation among layers, that each layer is an individual ODE solver. How to mitigate the error accumulation is the main goal in this work, which is not discussed in their work.

### E Derivations of the Equation

Let $E$ be the loss of training. $L$ be the number blocks of the model, and $y_L$ be the model output. Here, we define

$$ z_k = y_k + F(y_k, \theta_k) \quad (27) $$

Then the information flow of the RK2 method can be described as follows:

$$ y_{k+1} = y_k + \frac{1}{2}F(y_k, \theta_k) + \frac{1}{2}F(y_k + F(y_k, \theta_k), \theta_k) $$

$$ = y_k + \frac{1}{2}F(y_k, \theta_k) + \frac{1}{2}F(z_k, \theta_k) \quad (28) $$

where $\frac{\partial z_k}{\partial y_k} = 1 + \frac{\partial F(y_k, \theta_k)}{\partial y_k}$. In this way, the detail derivation of Eq. (28) is as follows:

$$ \frac{\partial y_{k+1}}{\partial y_k} = \frac{1}{2} \left(1 + \frac{\partial F(y_k, \theta_k)}{\partial y_k} + \frac{\partial F(z_k, \theta_k)}{\partial z_k} \cdot \left(1 + \frac{\partial F(y_k, \theta_k)}{\partial y_k}\right)\right) $$

$$ = \frac{1}{2} \left(1 + \left(1 + \frac{\partial F(z_k, \theta_k)}{\partial z_k}\right) \cdot \left(1 + \frac{\partial F(y_k, \theta_k)}{\partial y_k}\right)\right) \quad (29) $$

With the chain rule, the error $E$ propagates from the top layer $y_L$ to layer $y_l$ by the following formula:

$$ \frac{\partial E}{\partial y_t} = \frac{\partial E}{\partial y_l} \cdot \frac{\partial y_{L-1}}{\partial y_{L-2}} \cdots \frac{\partial y_{t+1}}{\partial y_t} \quad (30) $$

Here we have

$$ g_k = \left(1 + \frac{\partial F(y_k, \theta_k)}{\partial y_k}\right) \cdot \left(1 + \frac{\partial F(z_k, \theta_k)}{\partial z_k}\right) $$

Then, put the Eq. (30) into Eq. (29), the gradient of $E$ at $y_l$ is

$$ \frac{\partial E}{\partial y_{y_l}} = \frac{\partial E}{\partial y_L} \cdot \frac{1}{2^{L-l}} \prod_{k=l}^{L-1} \left(1 + g_k\right) \quad (31) $$

Similarly, we can easily obtain the gradient of RK2 method where $\gamma_i = 1$:

$$ \frac{\partial E}{\partial y_t} = \frac{\partial E}{\partial y_l} \cdot g_{l-1} \cdot g_{l-2} \cdots g_l $$

$$ = \frac{\partial E}{\partial y_L} \cdot \prod_{k=t}^{L-1} g_k \quad (32) $$

8351