将预训练的 Transformer 调整为 RNN

Jungo Kasai Hao Peng Yizhe Zhang Dani Yogatama Gabriel Ilharco Nikolaos Pappas Yi Mao
Weizhu Chen Noah A. Smith

摘要

在自然语言生成方面，Transformer 的表现要优于循环神经网络（如 RNN、LSTM 等）。但这也带来了巨大的计算成本，因为注意力机制的的复杂度与序列长度成二次方关系。该文提出了一种先转换再微调的的训练过程：在一个现有的预训练的 Transformer 中，将 softmax 注意力层换为一种线性复杂度的可复用选择网络，再进行微调。通过学习到的特征，该文方法提供了一种优于标准 Transformer 效率和其他循环神经网络变体准确性之间的折衷，在不重复昂贵的预训练过程的情况下提高效率。

关键词：Transformer；RNN

1 引言

在部分自然语言处理过程中，Transformer[1]模型超越了循环神经网络模型[2]的效果。特别是，Transformer 模型已经广泛应用于自回归建模，比如语言建模和机器翻译。Transformer 通过注意力机制，至关重要地利用了输入序列上特征向量之间的相互作用。然而，这在生成过程中带来了显著的计算资源和内存资源的占用。由于输出是基于前缀的增量预测，生成步骤不能随着时间步骤并行化，并且时间复杂度是序列长度的二次方。每一个生成步骤的内存消耗也会随着序列的变长而线性增长。因此长序列生成的瓶颈限制了大规模预训练 Transformer 的使用，如 GPT-3[3]、Image Transformer[4]和 DALL-E[5]。

最近的工作旨在减少自回归 Transformer 的开销。其中包括近似标准 softmax 注意力的循环替代方案[6-9]。与循环神经网络类似，这些模型通过固定大小的递归状态表示上下文，从而在生成序列长度上实现线性时间和恒定的记忆复杂度。当循环状态大小小于序列长度时，这些变体相比于 Transformer 有着实质性的速度和内存优势。然而，较小的状态大小往往会造成较差的生成质量[7]。

本文通过一种转换方法改善了效率和准确性之间的平衡：开发了一种方法，将预训练的 Transformer 转换为有效的 RNN，从而加速生成并减少内存使用，而不是从零开始训练一个循环神经网络。

2 相关工作

2.1 Tranformer

Transformer 由多头注意力、前馈网络和层归一化模块组成 (Vaswani et al., 2017)。因为它适用于并行化计算，和它本身模型的复杂程度导致它在精度和性能上都要高于之前流行的 RNN 循环神经网络，其注意力机制能够帮助当前词获取较好的上下文信息。但是在另一方面，在生成过程中，输出是增量构造的，注意力机制又成为了推理的瓶颈。

2.2 多头注意力机制

注意力模块以源向量和目标向量为输入序列。源向量用于产生键值特征，而目标向量则映射为查询向量。更正式的，用 $\{x_i^{tgt}\}_{i=1}^{N}$ 和 $\{x_j^{src}\}_{j=1}^{M}$ 表示目标和源向量，其中 x_i^{tgt}，$x_j^{src} \in \mathbb{R}^h$，h 是模型维数。

我们假设 d 维的 r 个注意力头 ($h = dr$)。对于每个头，输入向量首先通过 $W_* \in \mathbb{R}^{d\times h}$ 和 $b_* \in \mathbb{R}^d$ 的学习权重矩阵变换到 d 维的 query、key 和 value 特征向量。

$$q_i = W_q x_i^{tgt} + b_q, \tag{1a}$$

$$k_j = W_k x_j^{src} + b_k, \quad v_j = W_v x_j^{src} + b_v \tag{1b}$$

计算每个查询向量 q_i 与所有 M 个 key 向量的相似度并归一化，生成注意力系数，然后使用这些注意力系数输出 value 向量的加权平均值：

$$x_i^{out} = \sum_{j=1}^{M} \frac{sim(q_i, k_j)}{\sum_{l=1}^{M} sim(q_i, k_l)} v_j, \tag{2a}$$

$$sim(x, y) = exp(x \cdot y/\sqrt{d}). \tag{2b}$$

多头注意力对 r 个头中的每一个并行运行这个过程，并将 r 个输出向量链接起来，最终得到 h 维向量。

Transformer 生成速度开销图（图 1）描述了从输入向量出发的 Transformer 计算步骤及其时间复杂度。它总共包括两个阶段：1、特征映射: 对所有输入向量的 r 个头计算 $\{q_i\}_{i=1}^N, \{k_j\}_{j=1}^M$ 和 $\{v_j\}_{j=1}^M$。时间复杂度分别为：$\mathcal{O}(Nh^2), \mathcal{O}(Mh^2)$ 和 $\mathcal{O}(Mh^2)$。2、注意力机制：对 value 向量进行加权平均，时间复杂度为 $O(MNh)$，是序列长度 (M, N) 的二次方关系。

生成内存开销:在自回归生成中，query、key 和 value 向量在每一步消耗的空间复杂度为 $O(h), O(Mh)$ 和 $O(Mh)$。每一步的注意权重（式子 2a）跨越 M 个源位置，占用 $O(Mr)$ 空间，序列长度 M 为线性。

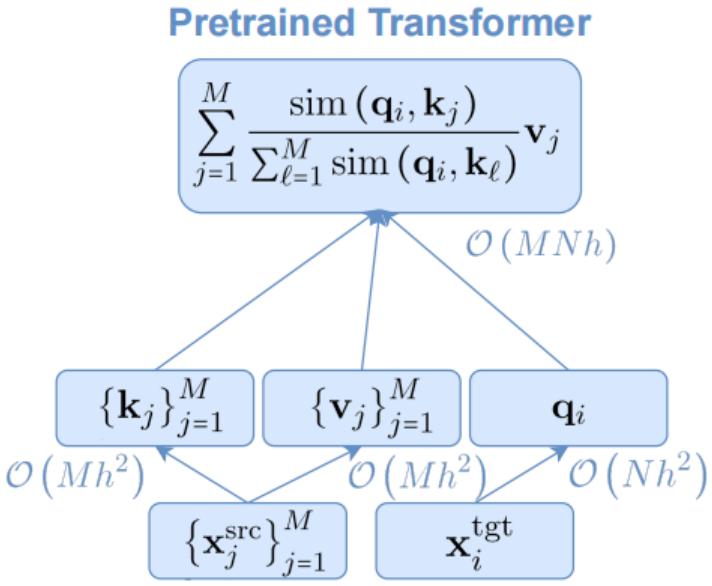

图 1: Transformer 生成速度开销图。M：源长度；N：目标长度；h：模型维度。

3 本文方法

3.1 本文方法概述

为了解决二次时间和线性空间的生成瓶颈，本文提出了 Transformer-to-RNN(T2R)，一种将预训练的 Transformer 转换为线性时间和序列长度中记忆复杂度恒定的 RNN 推理模型。T2R 遵循转换-再微调过程，修改预先训练的 Transformer 的注意力计算，并根据任务目标微调模型。

3.2 Transformer 转换为 RNN

本文首先将预训练 transformer 中的相似函数替换为

$$\widetilde{\text{sim}}(x,y) = \phi(x)\cdot\phi(y) \tag{3a}$$

$$\phi(x) = \text{relu}\left(W_\phi x + b_\phi\right). \tag{3b}$$

这里的 $\mathbf{W}_\phi \in \mathbb{R}^{k\times d}$ 和 $\mathbf{b}_\phi \in \mathbb{R}^k$ 学习单层 MLP 的参数。它们将 d 维向量映射到 k 维核特征空间。relu 激活函数确保了特征是非负的。不同的 MLP 用于不同的注意力头，因此每层总共添加 rk(d+1) 个可学习参数。然后，对这个修改后的网络中的所有参数（包括 MLP 参数）与原始任务目标进行微调。

在推理生成过程中，本文通过矩阵乘法的结合律将注意力计算重新表述为

$$\begin{aligned}\widetilde{x}_i^{\text{out}} &= \sum_{j=1}^{M}\frac{\widetilde{\text{sim}}\left(q_i,k_j\right)}{\sum_{\ell=1}^{M}\widetilde{\text{sim}}\left(q_i,k_\ell\right)}v_j \\ &= \left(\frac{\phi\left(q_i\right)\cdot\sum_{j=1}^{M}\phi\left(k_j\right)\otimes v_j}{\phi\left(q_i\right)\cdot\sum_{\ell=1}^{M}\phi\left(k_\ell\right)}\right)^\top\end{aligned} \tag{4}$$

这个公式适用于循环计算。在这个注意力机制中，每个查询只关注其前缀来预测下一个单词 $(M=i)$，先将状态定义为：

$$S_i = \sum_{j=1}^{i}\phi\left(k_j\right)\otimes v_j, \quad z_i = \sum_{j=1}^{i}\phi\left(k_j\right) \tag{5}$$

其中 $S_i, z_i \in \mathbb{R}^{k\times d}, \mathbb{R}^k$。这些状态可以循环计算：

$$S_i = S_{i-1} + \phi\left(k_i\right)v_i^\top, \quad z_i = z_{i-1} + \phi\left(k_i\right) \tag{6}$$

在序列对序列模型的自注意或交叉（编码器到解码器）注意中，S_i 和 z_i 相对于 i 是常数，只需要计算一次。给定位置 i 处的两种状态，我们可以得到输出向量：

$$\widetilde{x}_i^{\text{out}} = \left(\frac{\phi\left(q_i\right)^\top S_i}{\phi\left(q_i\right)^\top z_i}\right)^\top \tag{7}$$

这避免了关于输入序列长度的二次计算。我们还通过合并 MLP 特征映射和产生查询与键值的特征映射来加快推断。

$$\phi\left(q_i\right) = \text{relu}\left(\widetilde{W}_q x_i^{\text{tgt}} + \widetilde{b}_q\right), \tag{8a}$$

$$\phi\left(k_j\right) = \text{relu}\left(\widetilde{W}_k x_j^{\text{src}} + \widetilde{b}_k\right), \tag{8b}$$

$$\text{其中} \quad \widetilde{W}_q = W_\phi W_q, \quad \widetilde{W}_k = W_\phi W_k, \tag{8c}$$

$$\widetilde{b}_q = b_\phi + W_\phi b_q, \quad \widetilde{b}_k = b_\phi + W_\phi b_k. \tag{8d}$$

模型训练完成后，在生成前计算一次式子 8c-8d，中间特征 q_i 和 k_j 在推理过程中从不参与计算。

T2R 模型中每一步的时间复杂度如图 2 所示。与 Transformer 类似，T2R 也包含两个阶段：1、特征映射：对所有 r 个头计算 $\{\boldsymbol{\phi}\left(q_i\right)\}_{i=1}^N$，$\{\boldsymbol{\phi}\left(k_j\right)\}_{j=1}^M$ 和 $\{(v_j)\}_{j=1}^M$。时间复杂度分别为：$\mathcal{O}(Nhkr), \mathcal{O}(Mhkr)$ 和 $\mathcal{O}(Mh^2)$。2、注意力机制：RNN 状态和 r 个注意力头的输出的计算复杂度为：$\mathcal{O}(Mhk)$ 和 $\mathcal{O}(Nhk)$。

同预训练的 Transformer 相比，可以看到，如果特征大小远小于输入序列长度 $(k << M, N)$，T2R 中注意力阶段从 $\mathcal{O}(MNh)$ 到 $\mathcal{O}(hk(M+N))$ 带来了实质性的加速。

内存开销：T2R 只需要存储 RNN 状态，因此其空间复杂度为 $\mathcal{O}(hk)$，序列长度恒定。这意味着，

与 Trandformer 的 $\mathcal{O}(Mh)$ 相比，k < M 时内存占用会减少。

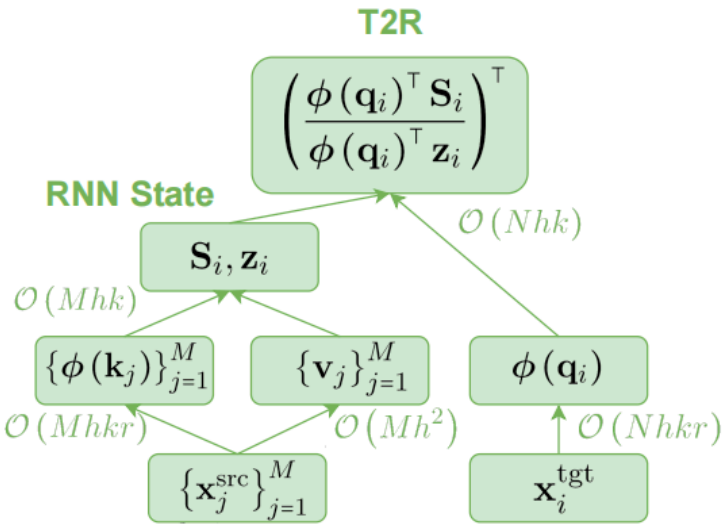

图 2: T2R 生成速度开销图。特征 $\phi(q_i)$ 和 $\phi(k_j)$ 直接由输入向量计算得到。M：源长度；N：目标长度；h：模型维度；k：特征纬度；r：注意力头数。

4 复现细节

4.1 与已有开源代码对比

本文虽然与 Transformer 和 RNN 有差异，但还是基于 Transformer 和 RNN 改进而来，因此复现代码参考了 Transformer 和 RNN 的源代码实现，再基于论文公式推导 T2R 的模型代码。

Procedure 1 Transformer

Input: seq *src*, seq *tgt*
Output: Probabilities x_i
for *j* ***in*** *lenth(src)* **do**
$k_j = W_k x_j^{src} + b_k \qquad v_j = W_v x_j^{src} + b_v$
end
for *i* ***in*** *lenth(tgt)* **do**
$q_i = W_q x_i^{tgt} + b_q$
end
for *i* ***in*** *lenth(tgt)* **do**
$x_i^{out} = \sum_{j=1}^{M} \frac{sim(q_i, k_j)}{\sum_{l=1}^{M} sim(q_i, k_l)} v_j \qquad sim(x, y) = exp(x \cdot y / \sqrt{d})$
end

4.2 实验环境搭建

本实验主要使用的 IDE 是 Pycharm，所使用的深度学习框架为 Pytorch。本实验所使用的环境依赖如下：

1、Python：3.9

2、Pytorch：1.12.0

3、transformers：4.23.1

4、jupyter notebook：6.4.8

4.3 创新点

本文的最大创新点就是提出了新的相似函数，将 Transformer 模型转换为 RNN，实现了生成加速并且从实质上减少内存的使用。T2R 伪代码如下：

Procedure 2 T2R

Input: seq *src*, seq *tgt*

Output: Probabilities x_i

for *j* ***in*** *lenth(src)* **do**

$$\phi(k_j) = \text{relu}\left(\widetilde{W}_k x_j^{\text{src}} + \widetilde{b}_k\right) \qquad \widetilde{W}_k = W_\phi W_k \qquad \widetilde{b}_k = b_\phi + W_\phi b_k$$

$$v_j = W_v x_j^{src} + b_v$$

end

for *i* ***in*** *lenth(tgt)* **do**

$$\phi(q_i) = \text{relu}\left(\widetilde{W}_q x_i^{\text{tgt}} + \widetilde{b}_q\right) \qquad \widetilde{W}_q = W_\phi W_q \qquad \widetilde{b}_q = b_\phi + W_\phi b_q$$

end

for *i* ***in*** *lenth(tgt)* **do**

$$\widetilde{x}_i^{\text{out}} = \left(\frac{\phi(q_i)^\top S_i}{\phi(q_i)^\top z_i}\right)^\top$$

$$S_i = S_{i-1} + \phi(k_i)\, v_i^\top, \quad z_i = z_{i-1} + \phi(k_i)$$

end

5 实验结果分析

首先训练一个 Transformer，训练参数：字典大小：38；文本长度：100；词向量纬度：8；Transformer 层数：2；注意力头数：2。训练 100 个 epoch 后的到损失函数如图：

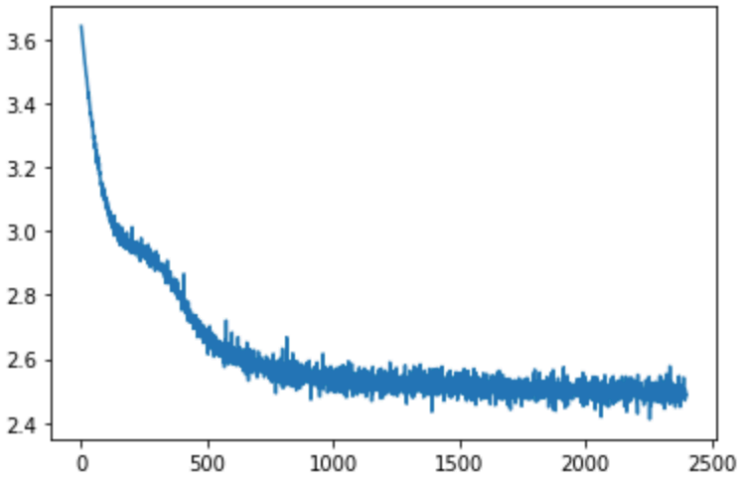

图 3: Transformer 损失函数图

然后训练 LSTM，训练参数：字典大小：38；文本长度：100；词向量纬度：8；网络层数：4。训练 250 个 epoch 后的效果才达到 Transformer 的效果，用时约为 Transformer 的 3.5 倍。

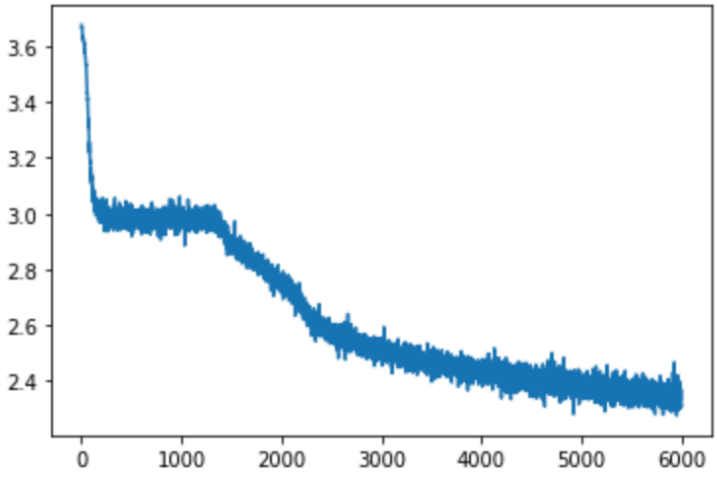

图 4: LSTM 损失函数图

最后再训练 T2R 模型，参数与 Transformer 一致，训练 100 个 epoch，模型的训练时间差不多为 Transformer 两倍，效果与 Tranformer 相似。

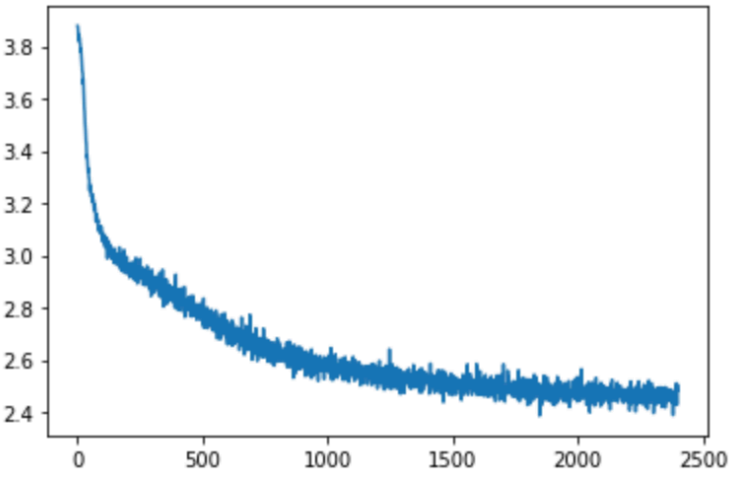

图 5: T2R 损失函数图

将预训练的两个模型分别载入，再根据论文公式微调，可以看到同样的序列长度 T2R 的推理时间比 Transformer 要短。

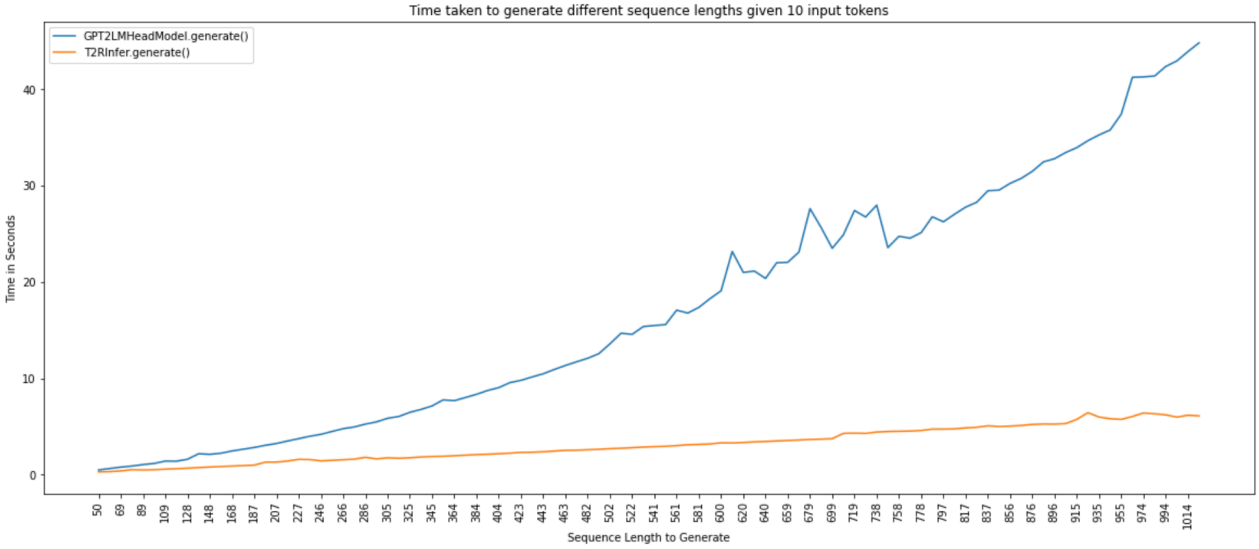

图 6: 序列长度相同时的总用时对比

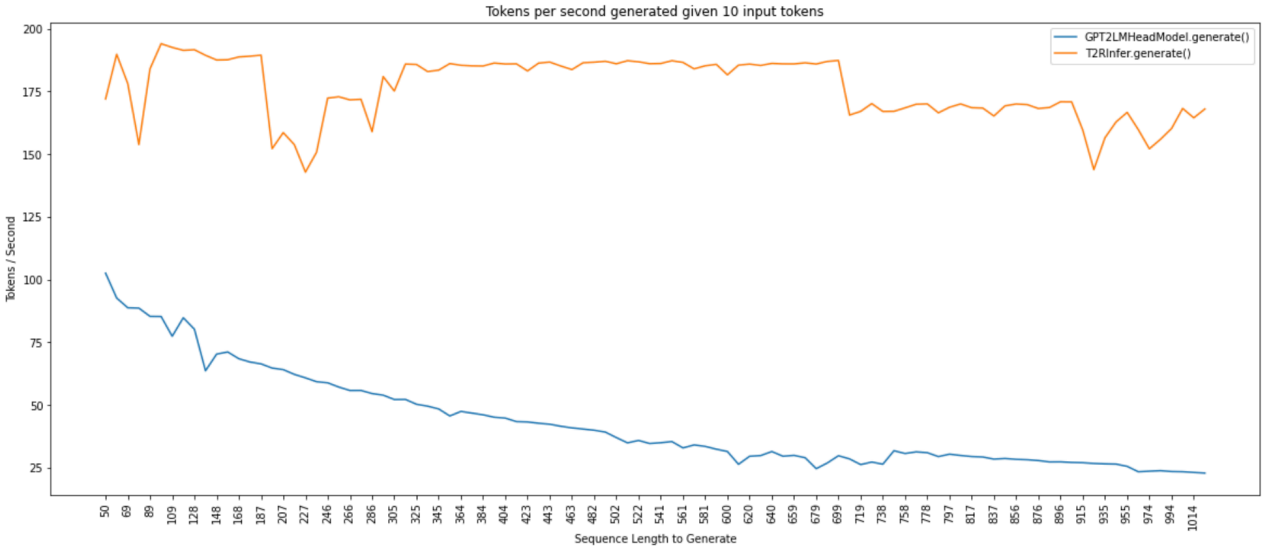

图 7: 序列长度相同时每 token 用时对比

6 总结与展望

本文提出了 T2R[10]，一种将预训练的 Transformer 转换为 RNN 的方法，减少了自回归生成的时间和内存成本。通过实验对比发现，该模型在效率与准确性之间产生了一种改进的权衡。本文的工作进一步说明了大规模的与训练模型可以被压缩为高效的推理模型，从而促进下游应用。

参考文献

[1] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[2] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[3] BROWN T, MANN B, RYDER N, et al. Language models are few-shot learners[J]. Advances in neural information processing systems, 2020, 33: 1877-1901.

[4] PARMAR N, VASWANI A, USZKOREIT J, et al. Image transformer[C]//International conference on machine learning. 2018: 4055-4064.

[5] RAMESH A, PAVLOV M, GOH G, et al. Zero-shot text-to-image generation[C]//International Conference on Machine Learning. 2021: 8821-8831.

[6] KATHAROPOULOS A, VYAS A, PAPPAS N, et al. Transformers are rnns: Fast autoregressive transformers with linear attention[C]//International Conference on Machine Learning. 2020: 5156-5165.

[7] PENG H, PAPPAS N, YOGATAMA D, et al. Random feature attention[J]. arXiv preprint arXiv:2103.02143, 2021.

[8] CHOROMANSKI K, LIKHOSHERSTOV V, DOHAN D, et al. Rethinking attention with performers [J]. arXiv preprint arXiv:2009.14794, 2020.

[9] SCHLAG I, IRIE K, SCHMIDHUBER J. Linear transformers are secretly fast weight memory systems [J]., 2021.

[10] KASAI J, PENG H, ZHANG Y, et al. Finetuning pretrained transformers into rnns[J]. arXiv preprint arXiv:2103.13076, 2021.