Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention

Angelos Katharopoulos

摘要

Transformer 在很多任务如图像识别，自自然语言处理上取得了非常显著的效果，但是由于其二次复杂度，在处理非常长的序列时, 需要占用大量的计算资源，导致处理速度缓慢。为了解决这一局限性，本文将自注意力机制表示为核特征映射的线性点积，并利用矩阵乘积的结合性，将复杂度从 O(N^2) 降低至 O(N)。其中 N 表示序列的长度。本文表明，这种映射允许迭代实现，显著加速自回归 Transformer，并揭示了它们与循环神经网络的关系。Liner Transformer 实现了与普通 Transformer 相似的性能，在非常长的序列的自回归预测上，它们的速度高达 4000 倍。

关键词：Transformer；RNN；Fast Transformer

1 引言

Transformer 模型最初是由 V aswani 等人 (2017) 在神经机器翻译的背景下引入的 (Sutskever 等人，2014;Bahdanau 等人，2015 年)，并在处理自然语言 (Devlin 等人，2019 年)、音频 (Sperber 等人，2018 年) 和图像 (Parmar 等人，2019 年) 的各种任务中展示了非常好的效果。除了有充分监督的任务外，在使用自回归进行预训练时，Transformer 模型也能有效地将知识转移到监督有限或没有监督的任务中 (Radford 等人，2018;2019) 或隐藏语言建模目标 (Devlin 等人，2019; 杨勇等，2019; 宋等，2019;Liu 等人，2020)。

然而，Transformer 模型虽然带来非常好的效果，但是需要非常高的计算和内存成本。瓶颈是主要原因, 它处理 N 个输入的上下文，具有二次记忆内存和时间复杂度 O(N^2) 因此，在实践中，Transformer 的训练是缓慢的，并且他们的上下文是有限的。这破坏了时间一致性，阻碍了对长期依赖性的捕获。Dai 等人 (2019) 通过关注来自以前上下文的记忆解决了后者，尽管这是以牺牲计算效率为代价的。

最近，研究人员将注意力转移到在不牺牲效率的情况下增加上下文长度的方法上。为此，Child 等人 (2019) 引入了注意矩阵的稀疏分解，将自我注意复杂度降低到 O ($N\sqrt{N}$)．Kitaev 等人 (2020) 使用局部敏感哈希进一步将复杂度降低到 O (N log N)。这使得扩展到长序列成为可能。尽管上述模型可以有效地训练长序列，但它们不能加速自回归推理。

在本文中，引入了 Linear Transformer 模型，该模型显著减少了内存占用，并随上下文长度线性扩展。我们通过使用基于核的自我注意公式和矩阵乘积的结合律来计算自我注意权重来实现这一点。使用线性公式，可以得到线性复杂度和恒定记忆表示因果掩蔽。这揭示了 Transformer 和 rnn 之间的关系，使我们能够更快地执行数量级的自回归推理。

2 相关工作

在本节中，将概述寻求解决 Transformer 的大内存和计算需求的最相关的工作。此外，我们讨论了理论分析 Transformer 模型核心部件的方法，即自注意。最后，介绍本文使用的 Linear Transformer，旨在缓解注意力计算中的 softmax 瓶颈。

2.1 Efficient Transformers

现有的研究试图通过权重修剪 (Michel et al, 2019)、权重分解 (Lan et al, 2020)、权重量化 (Zafrir et al, 2019) 或知识蒸馏来提高变压器的记忆效率。Clark et al(2020) 提出了一种新的预训练目标，称为替换令牌检测，该目标的样本效率更高，并减少了总体计算量。Lample 等人 (2019) 使用产品键注意力以可以忽略的计算开销增加任何层的容量。

使用这些方法减少内存或计算需求可以加速训练或推理时间，但从根本上说，时间复杂度相对于序列长度仍然是二次的，这阻碍了扩展到长序列。相比之下，Linear Transformer 降低了变压器的内存和时间复杂度。

另一项研究旨在增加 Transformer 自我注意的“上下文”。上下文指的是序列中用于计算自我注意力的最大部分。Dai 等人 (2019) 介绍了 Transformer-XL，它通过在不破坏时间一致性的情况下学习固定长度上下文之外的依赖关系，实现了语言建模方面的最先进技术。但是，在内存中维护以前的上下文会带来显著的额外计算成本。相比之下，Sukhbaatar 等人 (2019) 通过学习每个注意头的最佳注意时长，显著延长了上下文长度，同时保持对内存占用和计算时间的控制。请注意，这两种方法都具有与香草模型相同的渐近复杂性。相反，本文提高了自我注意的渐近复杂性，这允许有更大的应用场景。

与本文模型更相关的是 Child 等人的作品 (2019) 和 Kitaev 等 (2020)。前者 (Child et al, 2019) 引入了注意矩阵的稀疏分解，将总体复杂度从二次元降低到 O ($N\sqrt{N}$) 用于长序列的生成建模。最近，Kitaev 等人 (2020) 提出了改革者。该方法通过使用位置敏感哈希 (LSH) 来执行更少的点积，进一步将复杂度降低到 O (N log N)。注意，为了能够使用 LSH, Reformer 将注意的键约束为与查询相同。因此，该方法不能用于解码键需要与查询不同的任务。相比之下，线性变压器对查询和键没有任何约束，而是根据序列长度线性扩展。此外，它们可以用于在自回归任务中执行推理，速度快了三个数量级，在验证难度方面实现了相当的性能。

2.2 自注意力机制

从理论角度更好地理解自我注意的努力很少。Tsai 等人 (2019) 在 Transformer 中提出了一种基于核的注意公式，该公式认为注意是在输入上应用核平滑器，核分数是输入之间的相似性。该公式提供了一种更好地理解注意成分和整合位置嵌入的方法。相比之下，本文使用核公式来加快自我注意的计算速度，降低其计算复杂度。此外，如果在查询和键上应用相似度为正的核，线性注意会正常收敛。

最近，Cordonnier 等 (2020) 提供了理论证明和经验证据，证明一个头部数量足够多的多头自注意可以表达任何卷积层。在这里，我们转而表明，用自回归目标训练的自注意层可以被视为一个递归神经网络，这一观察结果可用于显著加快自回归变压器模型的推理时间。

2.3 Linearized softmax

多年来，softmax 一直是训练具有大量类别的分类模型的瓶颈 (Goodman, 2001;Mnih, 2009)。近期工作 (Blanc, 2017;Rawat 等人，2019) 使用特征图的线性点积来近似 softmax，以通过采样加速训练。在这些工作的启发下，本文将变压器的软最大注意线性化。与此同时，Shen 等人 (2020) 探索了在图像中的目标检测任务中使用线性化注意力。此外，通过核映射，每个 Transformer 可以被视为一个循环神经网络。

3 本文方法

3.1 本文方法概述

在本节中，我们形式化了 Linear Transformer。将注意力从传统的 softmax 注意力转变为基于特征图的点积注意力，可以获得更好的时间和记忆复杂度，以及一个可以在线性时间内执行序列生成的因果模型，类似于递归神经网络。首先，在 §3.2 中，介绍了 (V aswani 等人，2017) 中介绍的 Transformer 架构的公式。随后，在 §3.3 中，介绍 Linear Transformer 的定义和实现。

3.2 Transformer

令 x^{N*F} 是维度为 F 的 N 个特征向量。一个 Transformer 相当于一个转换函数 $T : R^{N*f} \Longrightarrow R^{N*f}$, 由 L 个 transformer 层组成：$T_1(\cdot), \cdots, T_n$, 如下：

$$T_x = f_l(A_l(x) + x) \tag{1}$$

函数 $f_l(\cdot)$ 对每一个特征进行独立变换，通常采用两个小型的两层前馈网络实现，$A_l(\cdot)$ 是自我注意函数，是 Transformer 中唯一跨序列作用的部分。自我注意函数 $A_l(\cdot)$ 对每个位置计算所有位置的特征表示的加权平均值，其权重与表示之间的相似度成正比。形式上表示为，输入序列 X 由三个矩阵 $W_Q \in R_{F*D}, W_K \in R_{F*D} and W_V \in R_{F*M}$ 投影到表示为 Q，K，V。所有位置的输出表示为，$A_l(x) = v'$, 计算公式如下：

$$\begin{aligned} Q &= xW_Q, \\ K &= xW_k, \\ V &= xW_V, \\ A_l(x) = V' &= softmax(QK^T/\sqrt{D})V \end{aligned} \tag{2}$$

在上式中，对 Q,K_T 按行应用了 softmax 函数。根据常用术语，Q、K 和 V 分别被称为“查询”、“键”和“值”。公式 2 实现了一种特定形式的自我注意，称为 softmax 注意，其中相似性得分是查询和键之间的点积的指数。假设用 i 下标矩阵返回第 i 行作为向量，我们可以写出任意相似函数的广义注意方程:

$$V_i' = \frac{\sum_{j=1}^{N} sim(Q_i, K_j)V_j}{\sum_{j=1}^{N} sim(Q_i, K_j)} \tag{3}$$

3.3 Linearized Attention 定义

方程 2 中注意的定义是通用的，可以用来定义其他几个注意实现，如多项式注意或 RBF 核注意 (Tsai et al, 2019)。注意，为了让公式 3 定义注意函数，我们需要对 sim(•) 施加的唯一约束是非负的。这包括所有核 $kx, y : R_+^{2*F}$。给定一个核函数 $\phi(x)$ 后，可以将公式二改写为：

$$V_i' = \frac{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)V_j}{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)} \tag{4}$$

然后利用矩阵乘法的结合律进一步简化:

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j) V_j}{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)} \tag{5}$$

当分子写成向矢量形式时，上面的方程就更容易理解了，

$$(\phi(Q)\phi(K)^T)V = \phi Q(\phi(K)^T TV). \tag{6}$$

由式 2 可知，softmax 注意力的计算代价随 ON^2，其中 N 为序列长度。内存需求也是如此，因为必须存储完整的注意矩阵来计算与查询、键和值有关的梯度。相比之下，我们从公式 5 提出的 Linear Transformer 具有时间和内存复杂度 O(N)，因为我们可以计算一次 $\sum_{j=1}^{N} \phi(K_j) V_j^T$ 和 $\sum_{j=1}^{N} \phi K_j$，并在每个查询中重用它们。

对于我们处理较小序列的实验，我们使用了一个特征映射，核映射函数定义如下:

$$\phi x = elu(x) + 1 \tag{7}$$

其中 elu(・) 表示指数线性单位 (Clevert et al, 2015) 激活函数。使用 elu(・) 而不是 rlu((・)，以避免在 x 为负时将梯度设置为 0。这个特征映射产生了一个需要 O (NDM) 乘法和加法的注意函数。在我们的实验部分，我们展示了方程 7 的特征映射与完整 Transformer 的性能相当，同时显著减少了计算和内存需求。

4 复现细节

4.1 与已有开源代码对比

网上传统的 Transformer 和 Fast Transformer 都有开源代码,但是没有源码可以同时生成两种不同种类的 Transformer, 本文使用工厂设计模式，设计了一个 builders，可以同时生成不同种类的 Transformer。

4.2 实验环境搭建

实验环境如表 1 所示：

表 1: 实验环境

环境名称	版本
python	3.8
pytorch	1.4.0-cu
OS	Ubuntu20.4

4.3 界面分析与使用说明

本文提供了一个网页去调用对联生成模型，可以在输入框内输入上联，点击提交就会返回模型自动生成的下联。

图 1: 操作界面示意

5 实验结果分析

在本节中，我们将通过实验分析所提出的 Linear Transformer 的性能。首先，在 §5.1 中，我们根据计算成本、内存消耗和合成数据的收敛性来评估线性化注意。为了进一步展示 Linear Transformer 的有效性，我们在两个现实应用中评估了我们的模型，§4.2 中的图像识别和 §4.3 中的对联生成。我们表明，与最先进的 Transformer 架构相比，我们的模型实现了具有竞争力的性能，同时需要更少的 GPU 内存和计算。

5.1 收敛性分析

为了检验 Linear Transformer 的收敛性，我们在一个带有因果掩蔽的人工复制任务上进行训练。也就是说，Transformer 必须复制一系列类似于 Kitaev 等人的序列复制任务的符号 (2020)。我们使用输入不同长度长度的序列，比较传统 Transformer 和 Linear Transformer 的时间和内存消耗随着序列长度增加的增长速度，实验结果如图 2 所示:

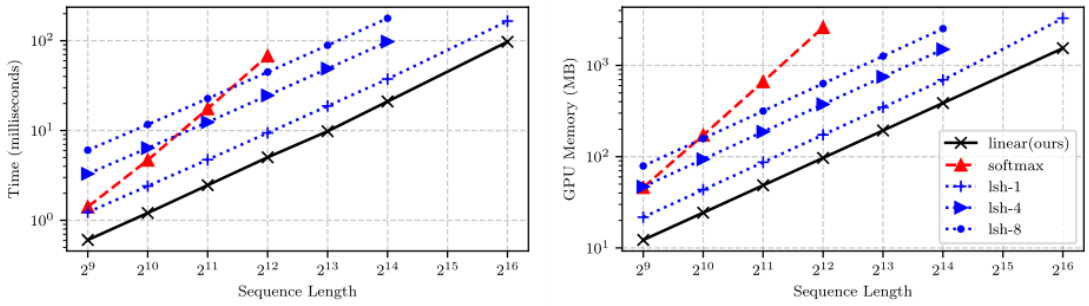

图 2: 收敛性分析图

比较重整器 (lsh-X) 的正向/向后通、软最大注意和线性注意的计算要求。与 softmax 不同的是，Linear 和 Reformer 模型随序列长度线性缩放，而 softmax 则随序列长度在内存和时间上的平方缩放。

5.2 基于 MNIST 的图像生成模型

首先，我们在广泛使用的 MNIST 数据集 (LeCun et al, 2010) 上评估了使用自回归 Transformer 的图像生成模型。本实验的结构包括 8 个注意层，每个注意层有 8 个注意头。我们将嵌入大小设置为 256，即每个头 32 个维度。我们的前馈尺寸是嵌入尺寸的 4 倍。我们用 Salimans 等人 (2017) 介绍的 10 维向量对输出进行建模。我们使用学习率为 10^{-4} 的 RAdam 优化器，训练所有模型 250 个周期。对于转换器基线，我们使用 1 和 4 轮哈希。此外，正如 Kitaev 等人 (2020) 所建议的，我们使用 64 个桶和大约 32 个元素的块。特别地，我们将 783 个长输入序列分成 27 个块，每个块包含 29 个元素。由于序列长度非常小，即只有 784 像素，为了消除由于不同批处理大小造成的差异，我们对所有方法都使用 10 批处理大小。表 1 总结了结果。我们观察到，就最终的复杂性而言，Linear Transformer 的性能与传统 Transformer 几乎相同，同时能够以超过 300 倍的速度生成图像。

表 2: 手写数字生成结果分析表

Method	Bits/dim	Images/sec
SoftMax	0.621	0.45(1x)
Linear	0.644	142.89(317x)

5.3 对联生成

本次用到的数据集为 couplet-dataset，其一共包含有 770491 条训练样本，4000 条测试样本。使用 couplet-dataset 数据集构建一个使用自回归 Transformer 的对联生成模型。本实验的结构包括 8 个注意层，每个注意层有 8 个注意头。我们将嵌入大小设置为 256，即每个头 32 个维度。我们的前馈尺寸是嵌入尺寸的 4 倍。我们使用学习率为 10^{-4} 的 RAdam 优化器，训练所有模型 20 个周期。模型训练的 loss 收敛情况如图 3 所示。模型在 1k 次 epoch 后开始收敛，并且取得了一个比较良好的 loss 值。

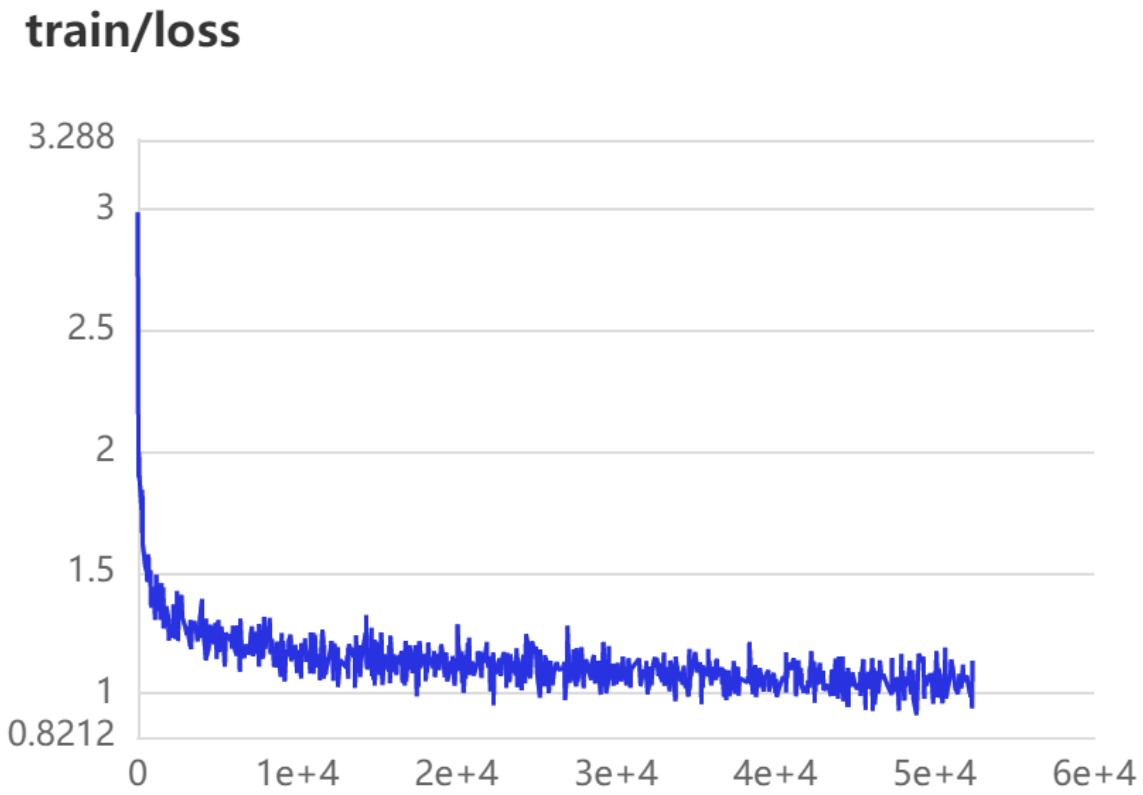

图 3: loss 收敛图

6 总结与展望

在这项工作中，我们提出了 Linear Transformer，一个模型，大大降低了原有 Transformer 的内存和计算成本。特别是，通过利用矩阵乘积的结合律，我们能够计算时间和记忆中的自我注意，它们与序列长度成线性关系。本文表明，该模型可以使用因果掩蔽，并仍然保持其线性渐近复杂性。最后，我们将 Transformer 模型表示为一个循环神经网络，这允许我们对自回归任务进行推断，速度快数千倍。

这一特性为未来在 rnn 和 transformer 中存储和检索信息的研究开辟了许多方向。另一个有待探索的研究方向与线性注意特征图的选择有关。例如，用随机傅里叶特征近似 RBF 核可以允许我们使用预训练的模型，使用 softmax 注意。