Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention

Angelos Katharopoulos [1 2] **Apoorv Vyas** [1 2] **Nikolaos Pappas** [3] **François Fleuret** [2 4 *]

Abstract

transformer在几个任务中取得了卓越的性能，但由于它们二次复杂度，相对于输入长度，它们是对于非常长的序列来说，速度慢得令人望而却步。为了解决这个限制，我们将自注意力表示为核特征图和的线性点积利用矩阵乘积的相合性来约简复杂度从$\mathcal{O}\left(N^2\right)$到$\mathcal{O}\left(N\right)$，其中$N$是序列长度。这种表述允许迭代实现极大地加速了自回归transformer，并揭示了它们与循环神经网络的关系。线性转换器 实现与普通transformer相似的性能，最高可达4000倍更快地进行超长序列的自回归预测。

1. 简介

Transformer模型最初由Vaswani et al. (2017) in引入神经机器翻译的语境(Sutskever et al., 2014; Bahdanau et al., 2015)和有在处理自然的各种任务中展示了令人印象深刻的结果语言(Devlin et al., 2019)，音频(Sperber et al., 2018)和图像(Parmar et al., 2019)。除了有充分监督的任务外，transformer 是否能有效地将知识转移到有限或没有的任务中用自回归进行预训练时的监督(Radford et al., 2018; 2019)或掩码语言建模目标(Devlin et al., 2019; Yang et al., 2019; Song et al., 2019; Liu et al., 2020)。

然而，这些好处通常伴随着非常高的计算和内存成本。该瓶颈主要是由于自注意力的全局感受野进行处理造成的N具有二次记忆和时间复杂度的输入的上下文$\mathcal{O}\left(N^2\right)$。因此，在实践中，transformer的训练速度很慢上下文是有限的。这破坏了时间一致性，并妨碍了捕获长期依赖关系。Dai et al. (2019) 通过关注来自以前背景的记忆来解决后者，尽管在计算效率的开销。

[1]Idiap Research Institute, Switzerland [2]EPFL, Switzerland [3]University of Washington, Seattle, USA [4]University of Geneva, Switzerland. [*]Work done at Idiap. Correspondence to: Angelos Katharopoulos <firstname.lastname@idiap.ch>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

最近，研究人员将他们的注意力转移到增加不牺牲效率的上下文长度。为此目的，Child et al. (2019)介绍了稀疏分解的注意力矩阵将self-attention复杂度降低到$\mathcal{O}\left(N\sqrt{N}\right)$。Kitaev et al. (2020)进一步将复杂性降低到$\mathcal{O}\left(N \log N\right)$ 使用局部敏感哈希。这使得扩展到长序列有可能。尽管上述模型可以有效地进行训练大序列，它们不会加速自回归推理。

本文介绍了*linear transformer*模型显著减少了内存占用，并与之线性扩展上下文长度。我们通过使用基于核的公式来实现这一点利用自注意力和关联度的矩阵乘积来计算自注意力权重(§3.2)。使用线性函数公式化，用线性复杂度和常数表示因果掩蔽记忆(§3.3)。这揭示了它们之间的关系transformer和rnn，使我们能够进行自回归推理快了几个数量级(§3.4)。

对图像生成和自动语音识别的评估表明*linear transformer*可以达到该性能水平transformer的速度提高了3个数量级推理。

2. 相关工作

在本节中，我们提供了寻求的最相关工作的概述解决transformer的大内存和计算需求。此外，还讨论了核的理论分析方法transformer模型的组成部分，即自注意力。最后，我们提出另一种工作旨在缓解网络中的softmax瓶颈注意力计算。

2.1. 高效变压器

现有工作试图通过权重来提高transformer的记忆效率剪枝(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。Clark et al. (2020)提出一个新的预训练目标称为替换token检测，它包含更多的样本高效且减少了整体计算量。Lample et al. (2019)使用产品关键注意增加任何一层的容量，可忽略不计计算开销。

使用这些方法减少内存或计算需求会导致训练或推理时间加快，但从根本上说，时间复杂度是相对于序列长度来说仍然是二次的，这阻碍了缩放长序列。相比之下，所提出方法减少了两种内存以及transformer的时间复杂度(§3.2)和经验(§4.1)。

另一项研究旨在增加自我注意力的“上下文”在《变形金刚》中。Context指的是序列的最大部分用于计算自注意力。Dai et al. (2019)介绍Transformer-XL通过学习实现了最先进的语言建模依赖超出固定长度的上下文而不破坏时态连贯性。然而，在内存中维护先前的上下文会引入显著的额外计算成本。相比之下，Sukhbaatar et al. (2019)显著地扩展了上下文长度学习每个注意力头的最佳注意力广度，同时保持控制内存占用和计算时间。注意，两者这些方法与普通模型具有相同的渐近复杂性。在相比之下，提高了自注意力的渐近复杂性允许我们使用更大的上下文。

与我们的模型更相关的是Child et al. (2019)和的作品Kitaev et al. (2020)。前(Child et al., 2019)介绍注意力矩阵的稀疏分解降低了整体复杂度从quadratic到$\mathcal{O}\left(N\sqrt{N}\right)$用于长序列的生成建模。最近，Kitaev et al. (2020)提出了Reformer。这种方法通过使用局部敏感进一步降低复杂性到$\mathcal{O}\left(N \log N\right)$ hashing (LSH)以减少点积运算。注意，为了能够使用LSH, Reformer约束键，以获得注意力，使其相同查询。因此，此方法不能用于以下情况的解码任务键必须与查询不同。相比之下，*linear transformers* 对查询和键没有任何限制，可以线性扩展相对于序列长度。此外，它们还可以用于执行在自回归任务中的推理速度快三个数量级，实现在验证困惑度方面的可比较性能。

2.2. 理解Self-Attention

很少有人努力更好地理解自注意力理论视角。Tsai et al. (2019)提出了一种基于核的transformer中注意力的表述，将注意力视为应用一个在输入上更平滑的核，核分数是相似度输入之间。这种表述提供了一种更好的理解注意力的方式组件并集成位置嵌入。相反，我们使用核公式以加快自注意力的计算并降低its 计算复杂度。此外，我们观察到，如果一个核为正数对查询和关键字进行相似度评分，线性注意力正态收敛。

最近，Cordonnier et al. (2020)提供了理论证明和经验证据表明，多头自注意力具有足够的数量头可以表达任何卷积层。这里，我们将a 用自回归目标训练的自注意力层可以被视为循环神经网络和这种观察可以显著地用于加快自回归transformer模型的推理时间。

2.3. 线性化softmax

多年来，softmax一直是训练分类器的瓶颈具有大量类别的模型(Goodman, 2001; Morin & Bengio, 2005; Mnih & Hinton, 2009)。最近的工作(Blanc & Rendle, 2017; Rawat et al., 2019)，用的线性点积近似softmax 通过采样进行特征映射以加快训练速度。受到这些启发有效，我们在transformer中对softmax注意力进行线性化。同时地这项工作，Shen et al. (2020)探索了线性化注意力的使用图像中的目标检测任务。相比之下，我们不仅线性化了注意力计算，还开发了一个自回归transformer模型线性复杂度和恒定内存，用于推理和训练。通过核的透镜，每个transformer都可以被视为循环神经网络。

3. 线性变压器

在本节中，我们正式提出我们的建议*linear transformer*。我们现在将注意力从传统的*softmax*注意力转移到a 基于特征图的点积注意力可以提高时间和记忆力复杂性以及可以在中进行序列生成的因果模型线性时间，类似于循环神经网络。

首先，在§3.1中，我们介绍了一种配方Transformer架构介绍在(Vaswani et al., 2017)。随后，在§3.2和§3.3我们的建议*linear transformer*最后，在§3.4我们重写Transformer是一个循环神经网络。

3.1. transformer

设$x \in \mathbb{R}^{N\times F}$表示一个$N$维度特征向量序列$F$。transformer是一个函数$T : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times F}$由$L$ transformer层$T_1(\cdot), \ldots, T_L(\cdot)$的组成定义如下:

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

函数$f_l(\cdot)$独立地转换每个特征通常用一个小型的两层前馈网络实现。$A_l(\cdot)$是自我注意力功能，是transformer唯一一起作用的部分序列。

自注意力函数$A_l(\cdot)$为每个位置计算a a的所有其他位置特征表示的加权平均值权重与表示之间的相似度分数成比例。形式上，输入序列x由三个矩阵$W_Q \in \mathbb{R}^{F\times D}$、$W_K \in \mathbb{R}^{F\times D}$和$W_V \in \mathbb{R}^{F\times M}$投影到对应表示$Q$、$K$、$V$。所有元素的输出位置$A_l(x) = V'$的计算方法如下:

$$\begin{aligned} Q &= xW_Q, \\ K &= xW_K, \\ V &= xW_V, \\ A_l(x) = V' &= \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right)V. \end{aligned} \tag{2}$$

注意，在前面的等式中，softmax函数是逐行应用的QK^T。以下是常用的术语:Q, K和V分别作为“queries ‘，“keys ‘和“values ‘。

方程2实现了一种特殊形式的self-attention Softmax attention，其中相似性分数是点的指数查询和键之间的乘积。给定一个矩阵的下标i 返回i -th行作为向量，我们可以写一个泛化注意对于任何相似度函数，

$$V'_i = \frac{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right) V_j}{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right)}. \tag{3}$$

方程3等价于方程2 用$\text{sim}\left(q, k\right) = \exp\left(\frac{q^T k}{\sqrt{D}}\right)$替换相似度函数。

3.2. 线性注意

方程2中的注意力定义是通用的，可以是用于定义其他几个注意力实现，如多项式注意或RBF核注意(Tsai et al., 2019)。请注意我们只需要对sim $(\cdot)$施加约束，以便方程3定义一个注意力函数，即为非负的。这包括所有的内核$k(x, y) : \mathbb{R}^{2\times F} \rightarrow \mathbb{R}_+$。

给定这样一个具有特征表示的内核$\phi(x)$我们可以重写方程2如下:

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

然后利用的结合律进一步化简矩阵乘法

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

当分子写成时，上面的方程更容易遵循矢量形式如下:

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

注意，特征映射$\phi(\cdot)$是逐行应用到的矩阵Q和K。

从方程2可以明显看出，的计算成本Softmax注意力以$\mathcal{O}\left(N^2\right)$进行缩放，其中$N$表示序列长度。对于内存需求也是如此，因为需要充分的注意力必须存储矩阵来计算关于查询的梯度，键和值。相比之下，我们从方程5提出的*linear transformer*具有时间和内存复杂性$\mathcal{O}(N)$因为我们可以一次性计算$\sum_{j=1}^{N} \phi(K_j) V_j^T$和$\sum_{j=1}^{N} \phi(K_j)$，并为每个重复使用它们查询

3.2.1. 特征图和计算成本

对于softmax attention，总成本为乘法和添加内容的规模为$\mathcal{O}\left(N^2 \max(D, M)\right)$，其中$D$是查询和键的维度，$M$是维度价值观。相反，对于线性注意力，我们首先计算特征维度地图C。随后，计算新值需要$\mathcal{O}(NCM)$加法和乘法。

前面的分析没有考虑内核和的选择Feature功能。请注意，特征函数对应于指数核是无限维的，使得线性化精确的softmax注意力不可行。另一方面，多项式核函数，例如，具有精确的有限维特征映射，并已被证明同样适用于指数核或RBF核(Tsai et al., 2019)。线性化多项式的计算开销2级变压器是$\mathcal{O}\left(ND^2M\right)$。这使得计算复杂性有利时$N > D^2$。请注意，这在实践中是正确的我们希望能够处理具有数万个元素的序列。

对于处理较小序列的实验，我们采用了特征映射得到一个正的相似度函数，定义如下:

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

其中elu$(\cdot)$表示指数线性单位(Clevert et al., 2015)激活功能。我们更喜欢elu$(\cdot)$ relu$(\cdot)$避免在x为负时将梯度设置为0。这个特征图产生了一个需要$\mathcal{O}(NDM)$的注意力功能乘法和加法。在我们的实验部分，我们展示了方程7的特征图表现与完整的transformer相当，同时显著降低了计算和内存需求。

3.3. 因果掩蔽

transformer架构可用于有效地训练自回归模型通过掩盖注意力计算，使i -th位置只能受一个位置的影响j当且仅当$j \leq i$，即a 位置不受后续位置的影响。形式上，这是因果掩蔽变化方程3如下:

$$V_i' = \frac{\sum_{j=1}^{i} \text{sim}(Q_i, K_j) V_j}{\sum_{j=1}^{i} \text{sim}(Q_i, K_j)}. \tag{8}$$

按照§3.2的推理，我们将其线性化掩盖注意力如下所述，

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j)}. \tag{9}$$

通过对S_i和Z_i的介绍，

$$S_i = \sum_{j=1}^{i} \phi(K_j) V_j^T, \tag{10}$$

$$Z_i = \sum_{j=1}^{i} \phi(K_j), \tag{11}$$

我们可以把方程9简化为

$$V_i' = \frac{\phi(Q_i)^T S_i}{\phi(Q_i)^T Z_i}. \tag{12}$$

请注意，S_i和Z_i可以从S_{i-1}和Z_{i-1}计算常数时间，因此使线性transformer的计算复杂性因果掩蔽与序列长度呈线性关系。

3.3.1. 梯度计算

一个简单的方程12的实现，在任何深度学习框架，需要存储所有中间值S_i以便计算梯度。这增加了$\max(D, M)$倍的内存消耗;从而阻碍了因果线性注意的适用性更长的序列或更深的模型。为了解决这个问题，我们导出了中分子的梯度方程9作为累加和。这使得我们可以计算中因果线性注意力的前向和后向传递线性时间和常数存储器。详细的推导如下: 在补充材料中提供。

给定分子$\bar{V}_i$和标量损失函数的梯度关于分子$\nabla_{\bar{V}_i}\mathcal{L}$，我们推导$\nabla_{\phi(Q_i)}\mathcal{L}, \nabla_{\phi(K_i)}\mathcal{L}$和$\nabla_{V_i}\mathcal{L}$如下:

$$\nabla_{\phi(Q_i)}\mathcal{L} = \nabla_{\bar{V}_i}\mathcal{L}\left(\sum_{j=1}^{i}\phi\left(K_j\right)V_j^T\right)^T, \tag{13}$$

$$\nabla_{\phi(K_i)}\mathcal{L} = \left(\sum_{j=i}^{N}\phi\left(Q_j\right)\left(\nabla_{\bar{V}_j}\mathcal{L}\right)^T\right)V_i\,, \tag{14}$$

$$\nabla_{V_i}\mathcal{L} = \left(\sum_{j=i}^{N}\phi\left(Q_j\right)\left(\nabla_{\bar{V}_j}\mathcal{L}\right)^T\right)^T\phi\left(K_i\right). \tag{15}$$

方程中的累加和项9，13 - 15在线性时间内计算，并且需要相对于序列长度而言，内存是恒定的。这将导致对于给定的C维度的特征图，具有计算复杂度$\mathcal{O}\left(NCM\right)$和内存$\mathcal{O}\left(N\max\left(C,M\right)\right)$的算法。伪代码给出了分子的前向和后向传递的实现算法1。

3.3.2. 训练和推理

当训练自回归transformer模型时，需要完整的基本事实序列是可用的。这使得分层并行成为可能公式1的$f_l(\cdot)$和注意力计算。因此，与递归神经网络相比，transformer的训练效率更高网络。另一方面，在推断期间，timestep i的输出是timestep的输入$i+1$。这使得自回归模型不可能并行化。此外，transformer的每时间步成本不是恒定的; 相反，它会按当前序列长度的平方进行缩放，因为必须计算之前所有时间步的注意力。

我们提出的*linear transformer*模型结合了两者的优点。当涉及到训练时，计算可以并行化并充分利用gpu或其他加速器的优势。说到推理，成本对于我们的模型来说，每次预测的时间和内存是常数。这意味着我们可以简单地将$\phi\left(K_j\right)V_j^T$矩阵存储为内部状态并更新它吗在每个时间步，就像循环神经网络一样。这就导致了推理比其他transformer模型快数千倍。

3.4. transformer是rnn

在文献中，transformer模型被认为是一种基本的递归神经网络的不同方法。然而，从因果关系来看掩膜配方在§3.3和讨论在上一节，很明显，任何transformer层都与因果相关掩码可以写成一个模型，给定输入，修改内部状态，然后预测输出，即循环神经网络(RNN)。注意，与通用transformer (Dehghani et al., 2018)，我们考虑递归关于时间而不是深度。

在下面的方程中，我们形式化方程的transformer层1作为循环神经网络。生成的RNN 有两个隐藏状态，即注意力记忆s和Normalizer内存z。在中使用下标来表示时间步长递归式。

$$s_0 = 0, \tag{16}$$

$$z_0 = 0, \tag{17}$$

$$s_i = s_{i-1} + \phi\left(x_iW_K\right)\left(x_iW_V\right)^T, \tag{18}$$

$$z_i = z_{i-1} + \phi\left(x_iW_K\right), \tag{19}$$

$$y_i = f_l\left(\frac{\phi\left(x_iW_Q\right)^T s_i}{\phi\left(x_iW_Q\right)^T z_i} + x_i\right). \tag{20}$$

在上述等式中，x_i表示i -th输入，y_i表示i -th输入特定transformer层的输出。注意，我们的公式没有对特征函数施加任何约束，它就可以用于表示任何*transformer*模型，理论上甚至是使用softmax的模型注意。这个公式是更好地理解的第一步transformer和流行的递归网络之间的关系(Hochreiter & Schmidhuber, 1997)以及用于存储和检索的过程信息。

4. 实验

Algorithm 1 基于因果掩蔽的线性transformer

function forward($\phi(Q)$, $\phi(K)$, V):
  $V' \leftarrow 0, S \leftarrow 0$
  for $i = 1, \ldots, N$ **do**
    $S \leftarrow S + \phi\left(K_i\right)V_i^T$ equation 10
    $\bar{V}_i \leftarrow \phi\left(Q_i\right)S$
  end
  return $\bar{V}$
end
function backward($\phi(Q)$, $\phi(K)$, V, G):
  /* G is the gradient of the loss with respect to the output of forward */
  $S \leftarrow 0, \nabla_{\phi(Q)}\mathcal{L} \leftarrow 0$
  for $i = 1, \ldots, N$ **do**
    $S \leftarrow S + \phi\left(K_i\right)V_i^T$
    $\nabla_{\phi(Q_i)}\mathcal{L} \leftarrow G_iS^T$ equation 13
  end
  $S \leftarrow 0, \nabla_{\phi(K)}\mathcal{L} \leftarrow 0, \nabla_V\mathcal{L} \leftarrow 0$
  for $i = N, \ldots, 1$ **do**
    $S \leftarrow S + \phi\left(Q_i\right)G_i^T$
    $\nabla_{V_i}\mathcal{L} \leftarrow S^T\phi\left(K_i\right)$ equation 15
    $\nabla_{\phi(K_i)}\mathcal{L} \leftarrow SV_i$ equation 14
  end
  return $\nabla_{\phi(Q)}\mathcal{L}, \nabla_{\phi(K)}\mathcal{L}, \nabla_V\mathcal{L}$
end

在本节中，我们通过实验分析了所提算法的性能*linear transformer*。最初，在§4.1，我们评估在计算成本、内存消耗和合成数据的收敛。以进一步展示*linear transformers*，我们在两个真实世界的应用程序上评估了我们的模型，§4.2中的图像生成和自动语音

识别§4.3。该模型实现了具有竞争力的性能最先进的transformer架构，同时所需的资源大大减少GPU内存和计算。

在我们的实验中，我们将我们的模型与两个基线进行比较transformer与softmax attention和Reformer (Kitaev et al., 2020)，后者是最先进的加速transformer架构。为了改革者，我们使用PyTorch重新实现已发布的代码对于完整的转换器，我们使用默认的PyTorch实现。注意，对于Reformer，我们不使用可逆层。这不会影响结果，因为我们只测量内存消耗尊重self attention层。在所有实验中，我们使用**Softmax**(Vaswani et al., 2017)参考标准transformer 建筑，线性为我们提出的*linear transformers*和**lsh-X**用于改革者(Kitaev et al., 2020)，其中*X*表示哈希轮。

为了训练*linear transformers*，我们使用方程的特征图7。我们的PyTorch (Paszke et al., 2019)代码的文档和相关示例可以在`https://linear-transformers.com/`上找到。常数记忆梯度计算方程13 - 15是大约200行CUDA代码实现。

4.1. 综合任务

4.1.1. 收敛性分析

为了检查*linear transformers*的收敛特性，我们在an上进行训练具有因果掩蔽的人工复制任务。换句话说，transformer必须进行复制类似于一系列符号的序列复制任务Kitaev et al. (2020)。我们使用一个最大长度为128的序列，其中包含10个不同的符号，符号之间用a分隔专用分隔符符号。对于这三种方法，我们训练一个4层transformer有8个注意力头，batch大小为64，使用的是RAdam 优化器(Liu et al., 2019)的学习率10^{-3}这是在3000次更新后减少到10^{-4}。图2描绘相对于梯度步数的损失。我们观察到Linear平滑收敛，比LSH具有更低的损失没有哈希带来的噪声。特别是，它到达了与softmax损失相同。

4.1.2. 内存和计算需求

在本节中，我们将比较transformer的计算能力以及内存需求。我们计算a的注意力和梯度具有不同序列长度的合成输入$N \in \{2^9, 2^{10}, \ldots, 2^{16}\}$并测量分配的峰值GPU内存和每个所需的时间变压器的变化。我们将批量大小与序列长度和成反比报告每批样品的时间和内存。

每种方法都根据适合GPU的最大序列长度进行评估记忆。对于此基准测试，我们使用具有11GB内存的NVidia GTX 1080 Ti。这导致softmax和的最大序列长度为4096个元素lsh-4和lsh-8为16384。正如预期的那样，softmax以二次曲线的方式扩展相对于序列长度。我们的方法更快，需要的内存更少比每个配置的基线都要高，如图所示1。重构器和线性注意力量表与序列长度成线性关系。注意，虽然渐进的复杂性对于Reformer是$\mathcal{O}(N \log N)$, $\log N$足够小，不影响计算时间。

Method	Bits/dim	Images/sec	
Softmax	0.621	0.45	(1×)
LSH-1	0.745	0.68	(1.5×)
LSH-4	0.676	0.27	(0.6×)
Linear (ours)	0.644	**142.8**	**(317×)**

Table 1: MNIST图像的自回归图像生成比较。我们的线性变压器的比特/亮度几乎与全变压器相同Softmax注意力，但吞吐量高300倍以上图像生成。实验的全部细节见下文§4.2.1。

4.2. 图像生成

transformer 在有条件或无条件的任务上表现出了巨大的成果自回归生成(Radford et al., 2019; Child et al., 2019)，然而，由于任务本身的原因，从transformer中采样很慢顺序和内存随序列长度的平方而变化。在本节中，我们训练因果掩码transformer来预测图像逐像素。我们以比特为单位实现的性能每个维度的注意力与*softmax*相当，同时能够生成图像速度超过**1000**倍，并且内存保持不变从第一个像素到最后一个像素的每个图像。我们向读者推荐我们的补充比较在训练进化，质量生成的图像和时间来生成单个图像。此外，我们还与缓存键和值的更快的softmax transformer 相比在推理过程中，与PyTorch实现相反。

4.2.1. MNIST

首先，用自回归在图像生成方面评估了该模型广泛使用的MNIST数据集(LeCun et al., 2010)上的transformer。The 本实验的架构由8个attention层和8个attention组成正面各朝上。我们将嵌入大小设置为256，即每个头32维。我们的前馈尺寸是嵌入尺寸的4倍。我们用引入的10个物流的混合物对输出进行建模Salimans et al. (2017)。我们使用带有学习率的RAdam优化器10^{-4}并训练所有模型250个epoch。对于改革者的基线，我们使用1轮和4轮哈希。此外，正如在Kitaev et al. (2020)，我们使用64个桶和大约32个块元素。特别地，我们将783个长输入序列划分为27个块每个数组有29个元素。由于序列长度比较小，即只有784像素，为了消除因批量大小不同而产生的差异，我们使用了一个批量所有方法的大小为10。

表1总结了结果。我们观察到线性在最终的困惑度方面，transformer取得了几乎相同的性能，作为softmax transformer，同时能够生成图像超过300次更快。这是由于我们的模型的低内存需求而实现的能够用单个GPU同时生成10 000张MNIST图像。在特别是，内存相对于序列长度是恒定的，因为唯一需要存储在像素之间的是s_i和z_i 如公式18和19所描述的值。在另一方面，softmax和Reformer都需要随着序列的长度。

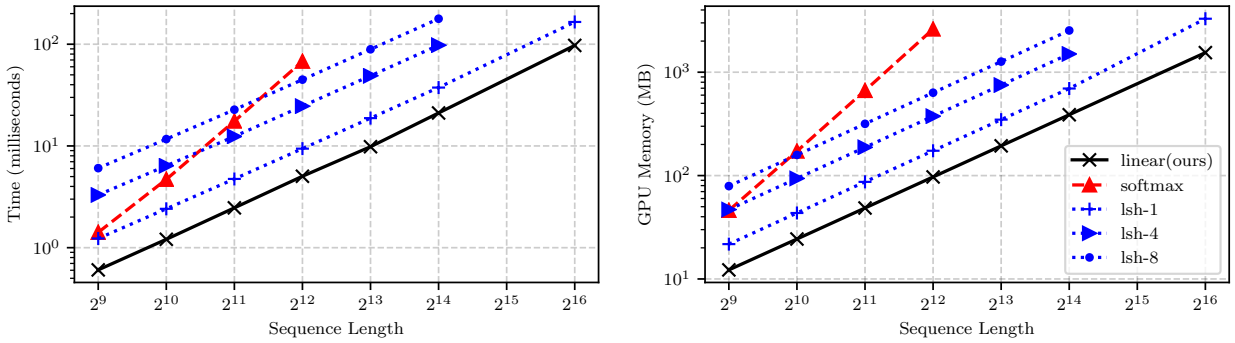


Figure 1: 比较for的前向/后向传递的计算需求Reformer (lsh-X)，softmax注意力和线性注意力。线性Reformer模型随着序列长度线性扩展，不像Softmax与序列长度的平方成比例在记忆和时间里。实验的完整细节可以在§4.1。

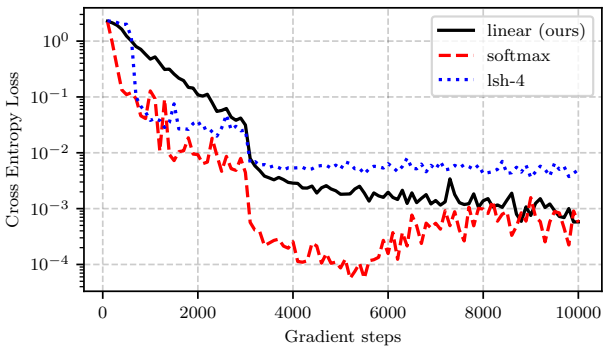


Figure 2: *softmax*、*linear*和*Reformer*专注于序列复制任务。线性稳定收敛并达到相同的终点性能与softmax相同。实验的细节已经公布§4.1。

Method	Bits/dim	Images/sec	
Softmax	3.47	0.004	(1×)
LSH-1	3.39	0.015	(3.75×)
LSH-4	3.51	0.005	(1.25×)
Linear (ours)	3.40	**17.85**	**(4,462×)**

Table 2: 我们在单个GPU上训练了一周的自回归transformer生成CIFAR-10图像。我们的线性transformer完成了3次比softmax更多的epoch，导致更好的困惑度。我们的模型生成图像4000 ×比基线快。The 实验的完整细节请见§4.2.2。

来自MNIST模型的图像补全和无条件样本可以在图3。我们观察到线性transformer产生非常有说服力的样本，具有清晰的边界和无噪声。在关于图像补全，我们还观察到transformer学会了使用有效地体现了与原始图像相同的笔画风格和宽度在很长的时间距离上。注意，当达到的困惑度更大时与所有模型不同的是，我们没有观察到两者之间的定性差异不同模型生成的样本。

4.2.2. CIFAR-10

线性公式的好处随着序列长度的增加而增加。为了展示这一点，我们训练了16层transformer来生成CIFAR-10图片(Krizhevsky et al., 2009)。对于每一层我们使用相同的和前面的实验一样配置。对于Reformer，我们再次使用64 桶和83个块，37个元素，大约是32，如纸张。由于序列长度几乎是的4倍在之前的实验中，完整的transformer只能在批量大小为1的情况下使用在我们可用的最大的GPU中，即具有24GB内存的NVidia P40 记忆。对于线性transformer和reformer，我们都使用批量大小4。所有模型训练7天。我们报告的结果是每维比特数和图像生成吞吐量见表2。请注意，虽然这个实验的重点不是最终的复杂度，很明显，随着序列长度的增长，速度很快transformer模型在每GPU小时的效率越来越高，实现了比慢速的同伴得分更高。

因为生成单个像素的内存和时间是对于Reformer和softmax attention，像素数量的增加线性transformer的吞吐量更明显。特别是，对于**softmax transformer**生成的每个图像，我们的方法可以生成**4460**张图像。图像补全和无条件样本从我们的模型可以看到图4。我们观察到该模型生成具有空间一致性的图像，可以补全图像令人信服，而不显著阻碍图像的识别类别。例如，在图4b中，所有图像都有成功完成狗的鼻子(第一排)或挡风玻璃的卡车(最后一排)。

4.3. 自动语音识别

为了表明所提出方法也可以用于非自回归任务，我们评估线性transformer在端到端自动语音中的性能基于连接时序分类(CTC)损失的识别(Graves et al., 2006)。在这种设置中，我们预测一个分布每个输入帧的音素以非自回归的方式。我们用80 小时WSJ数据集(Paul & Baker, 1992)的40维梅尔尺度没有时间差异作为特征的滤波器组。数据集包含平均帧数为800，最大序列长度为2400 框架。对于这项任务，

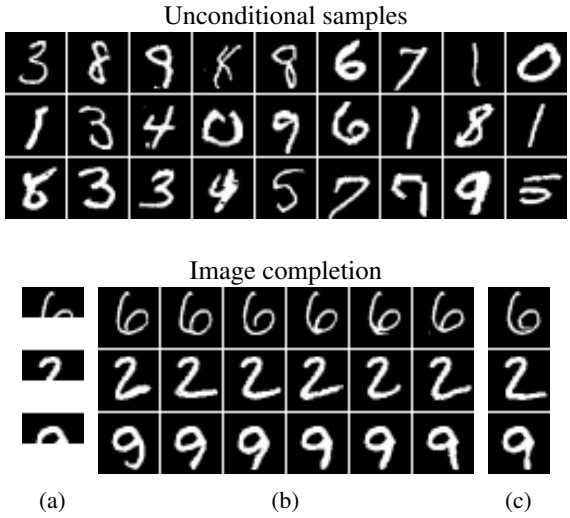


Figure 3: 由生成的无条件样本和图像补全MNIST的方法。(a)描述被遮挡的原始图像，(b (c)原件。我们的模型达到了位/维度对softmax，同时具有更多吞吐量提高**300**倍，生成**142** 图片/秒。详情请参见§4.2.1。

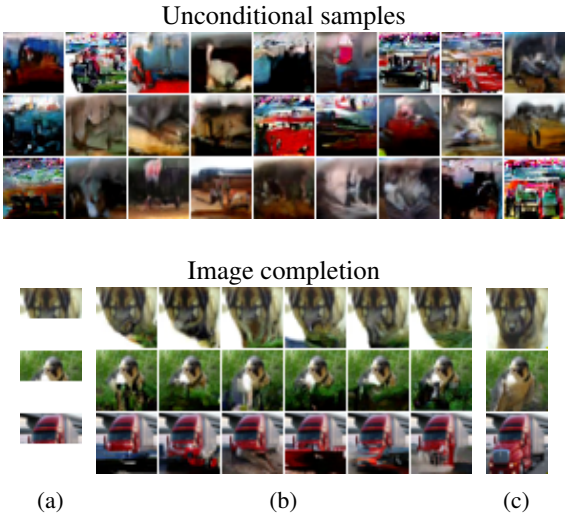


Figure 4: 由生成的无条件样本和图像补全CIFAR-10的方法。(a)描述被遮挡的原始图像，(b (c)原件。随着序列长度的增长与softmax相比，线性transformer更高效注意。我们的模型达到了**4000**多次 更高的吞吐量，每秒生成**17.85**张图像。详情请参见§4.2.2。

Method	Validation PER	Time/epoch (s)
Bi-LSTM	10.94	1047
Softmax	5.12	2711
LSH-4	9.33	2250
Linear (ours)	8.08	**824**

Table 3: 基于WSJ的自动语音识别性能比较数据集。结果以音素错误率的形式给出(PER)和每个epoch的训练时间。我们的模型优于LSTM 并进行改革，同时更快地进行培训和评估。详情实验可以在§4.3找到。

还与双向LSTM进行了比较(Hochreiter & Schmidhuber, 1997)与3层隐藏大小320。我们用Adam 优化器(Kingma & Ba, 2014)的学习率10^{-3}这是减少了验证错误停止减少。对于transformer模型，我们使用9层6个头，与的嵌入维度相同图像实验。作为优化器，我们使用具有初始学习率的RAdam 10^{-4}，当验证错误停止减少时，除以2。

所有模型都根据音素错误率(PER)和训练时间进行评估纪元。我们观察到，线性优于循环网络基线和Reformer在性能和速度方面都有很大的提升，如表3。请注意，softmax transformer，实现了较低的电话与所有基线相比，错误率较高，但速度明显较慢。在具体来说，*linear transformer*比3×更快。我们提供训练演化图在补充。

5. 结论

在这项工作中，我们提出了*linear transformer*，一个显著的模型减少了原始transformer的内存和计算成本。在特别地，通过利用矩阵乘积的结合性，我们能够在时间和内存上计算线性扩展的自注意力相对于序列长度。该模型可以用于因果关系掩码，仍然保持其线性渐近复杂度。最后，我们将transformer模型表示为循环神经网络，这允许我们对自回归任务执行推理的速度快了数千倍。

这一特性为未来的研究开辟了许多方向rnn和transformer中的信息存储和检索。另一个有待探索的研究方向与特征图的选择有关线性注意力。例如，用random近似RBF核傅里叶特征允许我们使用用softmax预训练的模型注意。

致谢

Angelos Katharopoulos是由瑞士国家科学基金会资助的授权编号为FNS-30209“ISUL”和FNS-30224“CORTI”。Apoorv Vyas是由瑞士国家科学基金会资助，资助编号为FNS-30213 shissm。尼古拉·帕帕斯得到了瑞士国家科学研究所的支持基金资助编号P400P2__183911“UNISON”。

References

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 5th International Conference on Learning Representations*, San Diego, CA, USA, 2015.

Blanc, G. and Rendle, S. Adaptive sampled softmax with kernel based sampling. *arXiv preprint arXiv:1712.00527*, 2017.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015.

Cordonnier, J.-B., Loukas, A., and Jaggi, M. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*, 2020.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, Ł. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Goodman, J. Classes for fast maximum entropy training. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, volume 1, pp. 561–564. IEEE, 2001.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Lample, G., Sablayrolles, A., Ranzato, M. A., Denoyer, L., and Jegou, H. Large memory layers with product keys. In Wallach, H., Larochelle, H., Beygelzimer, A., dÁlché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8546–8557. Curran Associates, Inc., 2019.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. 2010.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. RoBERTa: A robustly optimized BERT pretraining approach, 2020.

Michel, P., Levy, O., and Neubig, G. Are sixteen heads really better than one? In Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 14014–14024. Curran Associates, Inc., 2019.

Mnih, A. and Hinton, G. E. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pp. 1081–1088, 2009.

Morin, F. and Bengio, Y. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pp. 246–252. Citeseer, 2005.

Parmar, N., Ramachandran, P., Vaswani, A., Bello, I., Levskaya, A., and Shlens, J. Stand-alone self-attention in vision models. In Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 68–80. Curran Associates, Inc., 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Paul, D. B. and Baker, J. M. The design for the wall street journal-based csr corpus. In *Proceedings of the workshop on Speech and Natural Language*, pp. 357–362. Association for Computational Linguistics, 1992.

Radford, A., Narasimhan, K., Salimans, T., , and Sutskever, I. Improving language understanding by generative pre-training. In *OpenAI report*, 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Rawat, A. S., Chen, J., Yu, F. X. X., Suresh, A. T., and Kumar, S. Sampled softmax with random fourier features. In *Advances in Neural Information Processing Systems*, pp. 13834–13844, 2019.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Shen, Z., Zhang, M., Zhao, H., Yi, S., and Li, H. Efficient attention: Attention with linear complexities. *arXiv preprint arXiv:1812.01243*, 2020.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. MASS: Masked sequence to sequence pre-training for language generation. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5926–5936, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Sperber, M., Niehues, J., Neubig, G., Stüker, S., and Waibel, A. Self-attentional acoustic models. In *19th Annual Conference of the International Speech Communication Association (InterSpeech 2018)*, Hyderabad, India, September 2018.

Sukhbaatar, S., Grave, E., Bojanowski, P., and Joulin, A. Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 331–335, Florence, Italy, July 2019. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pp. 3104–3112. Curran Associates, Inc., 2014.

Tsai, Y.-H. H., Bai, S., Yamada, M., Morency, L.-P., and Salakhutdinov, R. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. 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)*, pp. 4343–4352, Hong Kong, China, November 2019. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NIPS*, 2017.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019.

Zafrir, O., Boudoukh, G., Izsak, P., and Wasserblat, M. Q8BERT: quantized 8bit BERT. *CoRR*, abs/1910.06188, 2019.

Supplementary Material for Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention

A. 梯度微分

在我们补充材料的第一节中，我们详细推导了因果掩码线性transformer的梯度，并表明它们可以以线性时间和常数内存计算。特别地，我们推导标量损失相对于下列分子的梯度方程，

$$V_i' = \frac{\phi\left(Q_i\right)^T \sum_{j=1}^{i} \phi\left(K_j\right) V_j^T}{\phi\left(Q_i\right)^T \sum_{j=1}^{i} \phi\left(K_j\right)}. \tag{21}$$

关于分母和分数的梯度是有效的由autograd处理。在不失一般性的情况下，我们可以假设Q和K 已经包含了$\phi(\cdot)$映射的向量，因此给出了分子

$$\bar{V}_i = Q_i^T \sum_{j=1}^{i} K_j V_j^T, \tag{22}$$

$\nabla_{\bar{V}}\mathcal{L}$我们试图计算$\nabla_Q\mathcal{L}$，$\nabla_K\mathcal{L}$和$\nabla_V\mathcal{L}$。请注意$Q \in \mathbb{R}^{N\times D}$、$K \in \mathbb{R}^{N\times D}$和$V \in \mathbb{R}^{N\times M}$。为了推导梯度，我们首先，不使用向量表示法表示上面的单元素方程，

$$\bar{V}_{ie} = \sum_{d=1}^{D} Q_{id} \sum_{j=1}^{i} K_{jd} V_{je} = \sum_{d=1}^{D} \sum_{j=1}^{i} Q_{id} K_{jd} V_{je}. \tag{23}$$

随后，我们可以通过取偏导数来推导Q的梯度派生任何Q_{lt}，如下

$$\frac{\partial \mathcal{L}}{\partial Q_{lt}} = \sum_{e=1}^{M} \frac{\partial \mathcal{L}}{\partial \bar{V}_{le}} \frac{\partial \bar{V}_{le}}{\partial Q_{lt}} = \sum_{e=1}^{M} \frac{\partial \mathcal{L}}{\partial \bar{V}_{le}} \left(\sum_{j=1}^{l} K_{jt} V_{je} \right). \tag{24}$$

如果我们将上面的方程写成梯度的矩阵乘积，它就变成了，

$$\nabla_{Q_i}\mathcal{L} = \nabla_{\bar{V}_i}\mathcal{L} \left(\sum_{j=1}^{i} K_j V_j^T \right)^T, \tag{25}$$

从主要论文证明方程13。在方程24中，我们使用事实上Q_{lt}只影响$\bar{V}_l$，因此我们不需要对i求和来计算梯度。然而，对于K和V来说，这并不是病例。特别是K_j影响所有$\bar{V}_i$其中$i \geq j$。因此，我们可以将损失对K_{lt}的偏导数写成:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial K_{lt}} &= \sum_{e=1}^{M} \sum_{i=l}^{N} \frac{\partial \mathcal{L}}{\partial \bar{V}_{ie}} \frac{\partial \bar{V}_{ie}}{\partial K_{lt}} = \sum_{e=1}^{M} \sum_{i=l}^{N} \frac{\partial \mathcal{L}}{\partial \bar{V}_{ie}} \frac{\partial \left(\sum_{d=1}^{D} \sum_{j=1}^{i} Q_{id} K_{jd} V_{je} \right)}{\partial K_{lt}} \\ &= \sum_{e=1}^{M} \sum_{i=l}^{N} \frac{\partial \mathcal{L}}{\partial \bar{V}_{ie}} Q_{it} V_{le}. \end{aligned} \tag{26}$$

至于Q，我们现在可以将梯度写成矢量形式，

$$\nabla_{K_i}\mathcal{L} = \left(\sum_{j=i}^{N} Q_j \left(\nabla_{\bar{V}_j}\mathcal{L} \right)^T \right) V_i, \tag{27}$$

Method	Bits/dim	Images/sec	
Softmax	0.621	0.45	(1×)
Stateful-softmax	0.621	7.56	(16.8×)
LSH-1	0.745	0.68	(1.5×)
LSH-4	0.676	0.27	(0.6×)
Linear (ours)	0.644	**142.8**	**(317×)**

(a) MNIST上的图像生成

Method	Bits/dim	Images/sec	
Softmax	3.47	0.004	(1×)
Stateful-softmax	3.47	0.32	(80×)
LSH-1	3.39	0.015	(3.75×)
LSH-4	3.51	0.005	(1.25×)
Linear (ours)	3.40	**17.85**	**(4,462×)**

(b) CIFAR-10上的图像生成

Table 4: MNIST自回归图像生成吞吐量比较以及CIFAR-10图像。实验可以在§4.2中找到主纸。对于statefulsoftmax，我们保存键和值重用它们来预测下一个元素。详细描述这个额外的基线可以在§C.1。

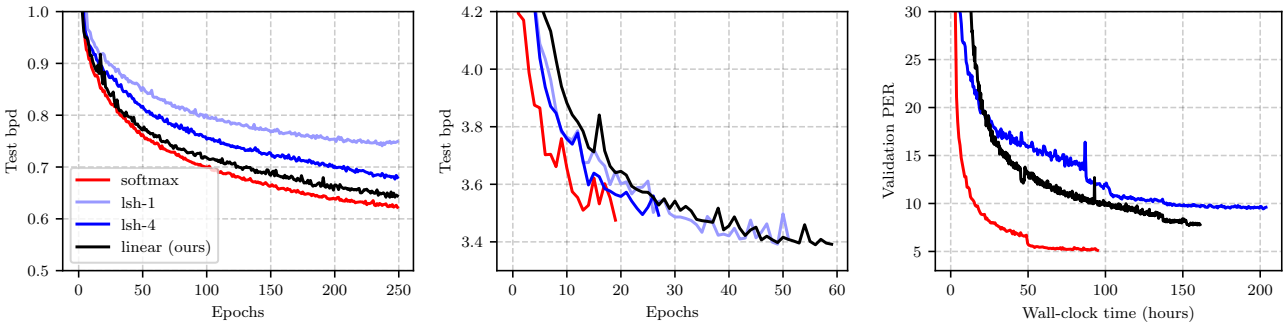
从论文中证明方程14。按照同样的推理，我们可以计算损失关于V_{lt}的偏导数并证明公式15。注意梯度的累积和矩阵就Q和K来说有相同的大小，不过在里面算一个向前方向(从1到N求和)类似于向前传递和Other是向后计算的(从N到1求和)，类似于在rnn中完成的时间反向传播。

B. 训练进化

在图5中，我们展示了所有transformer的训练演变我们实验中的模型。MNIST实验(图5a) 我们对所有方法进行250个epoch的训练。序列长度足够小，所以所有方法的训练时间都没有显著差异。我们观察该方法与softmax注意力收敛速度相当，表现优于softmax 值得注意的是，这两个reformer变体。

另一方面，对于CIFAR-10(图5b)，我们训练所有方法为固定的时间，即7天。我们观察到*lsh-1*和线性完成的纪元明显多于softmax和lsh-4和实现更好的性能。预计这一差距将进一步扩大增加序列长度。

最后，在我们的最后一个自动语音识别实验中(图. 5c)，softmax的性能明显优于Reformer和收敛性是线性的。请注意，线性每轮快3× 这意味着它完成了大约4倍于softmax。尽管softmax注意力在此任务中更好，但我们观察到*linear transformers*在收敛性和性能方面都明显优于Reformer 最后的表演。



(a) mnist (b) cifar-10 (c) 语音识别

Figure 5: 为所有实验训练transformer进化。它可以可以观察到*linear transformers*的收敛速度始终比Reformer和在自回归实验中与之相当softmax。对于MNIST，所有方法都训练了250个epoch 对于CIFAR，我们训练了7天。在语音识别中实验所有方法都经过训练达到收敛。详情实验可在§4.2.1, §4.2.2和§4.3中找到主要论文。

C. 图像生成吞吐量讨论

C.1. 有状态的**softmax**注意

在主论文§4.2中，我们报告了图像生成吞吐量，并与**softmax transformer**和**lsh**进行了比较。在本节中，我们创建另一个基线，记为**statefulsoftmax**，它实现了softmax自回归转换器作为一个循环模型。也就是说，所有的键和值都会被保存并传递在预测序列的下一个元素时，再次将其映射到模型。国家在这个循环模型中，键和值的集合是有大小的与序列长度成正比。这和我们的有质的不同提出的模型具有固定维度的状态，并计算i -th 给定前一个状态具有固定的计算成本，无论i。

表4总结了结果。我们观察到statefulsoftmax是比普通的transformer快得多。然而，它的复杂性是对于序列长度来说仍然是二次的，我们的公式更多比CIFAR-10快50 ×。此外，我们想指出为Reformer实现类似的有状态注意力并不是一项简单的任务每次有新输入时，都需要执行排序和分块操作提供。

C.2. 均衡批量大小

在前几节中，我们评估了所有transformer变体的吞吐量用于自回归图像生成任务。然而，另一个重要的需要考虑的因素是延迟，即产生a所需的总时间单幅图像。为此，我们使用批量大小为1并测量时间所有方法都需要生成单个图像。除了运行推断在GPU上，我们还评估了在CPU上所需的时间。结果详见表5。

Method	Seconds (CPU)		Seconds (GPU)	
Softmax	72.6	(13.2×)	10.2	(1.4×)
Stateful-softmax	7.4	(1.3×)	10.4	(1.42×)
LSH-1	46.0	(8.3×)	19.2	(2.6×)
LSH-4	112.0	(20×)	55.8	(7.6×)
Linear (ours)	**5.5**	(1×)	**7.3**	(1×)

(a) MNIST上的图像生成

Method	Seconds (CPU)		Seconds (GPU)	
Softmax	8651.4	(191.8×)	300.1	(4.9×)
Stateful-softmax	71.9	(1.6×)	70.4	(1.14×)
LSH-1	2318.9	(51.4×)	221.6	(3.6×)
LSH-4	5263.7	(116.7×)	683.9	(11.1×)
Linear (ours)	**45.1**	(1×)	**61.3**	(1×)

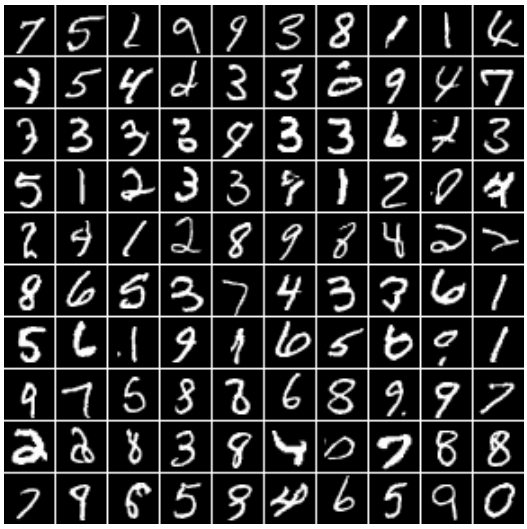
(b) CIFAR-10上的图像生成

Table 5: 生成单幅图像所需时间的比较MNIST和CIFAR-10上的自回归transformer。我们全都跑了方法在CPU和GPU上的批处理大小为1，并报告总时间以秒为单位。对于表中的所有数字，较低为更好。

所有方法都未充分利用GPU，并取得了显著的效果比表4中显示的图像生成吞吐量更小。所提出的线性transformer比所有方法和在特别是，它几乎比softmax transformer快6.6 × 在CIFAR-10上生成图像。注意我们的线性自回归transformer是在CPU上比在GPU上更快的唯一方法每一个案例。这是因为计算RNN所具有的注意力这一事实如此低的成本使得主要的计算瓶颈成为不可避免的对序列进行外循环。

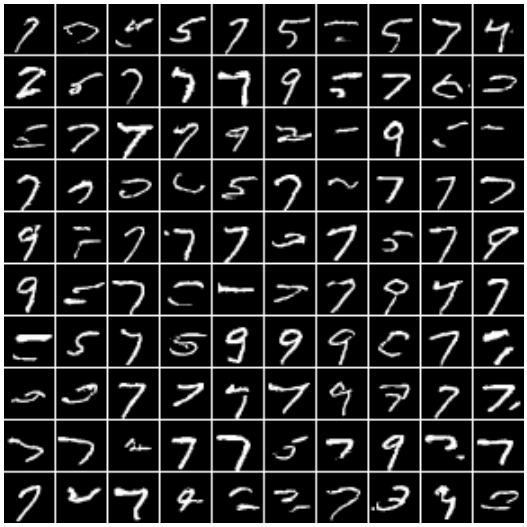
D. 图像生成的定性结果

在本节中，我们为图像生成提供定性的结果实验。由于所有模型的复杂度大致相同，因此预期，质量差异不显著。一个相当有趣的但是，Reformer模型提供的数据要少得多无条件样本的变异。此外，我们观察到那个图像完成任务比无条件生成要容易得多模型的表现明显更好。



(a) Softmax

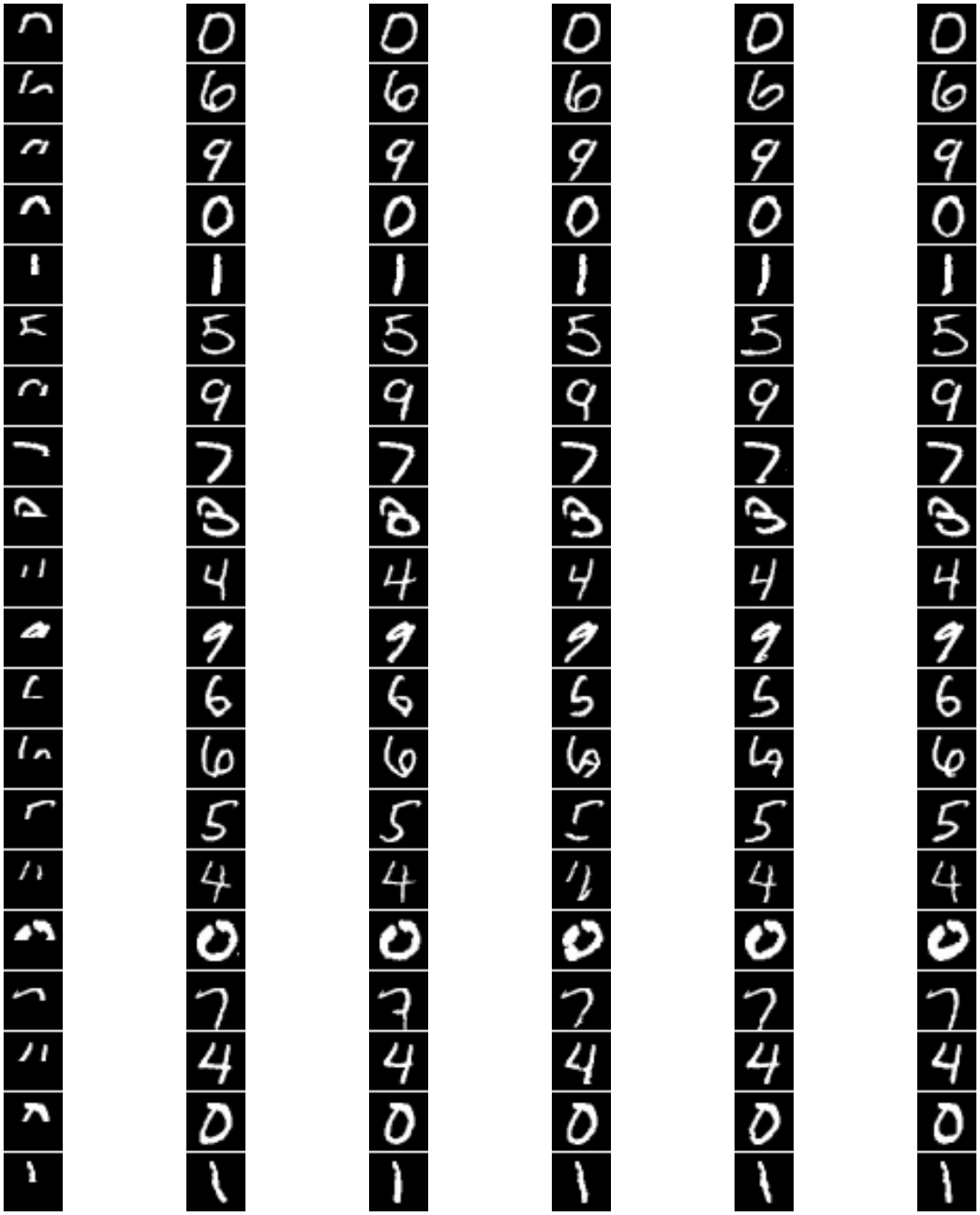
(b) 线性(我们的)



(c) lsh-1

(d) lsh-4

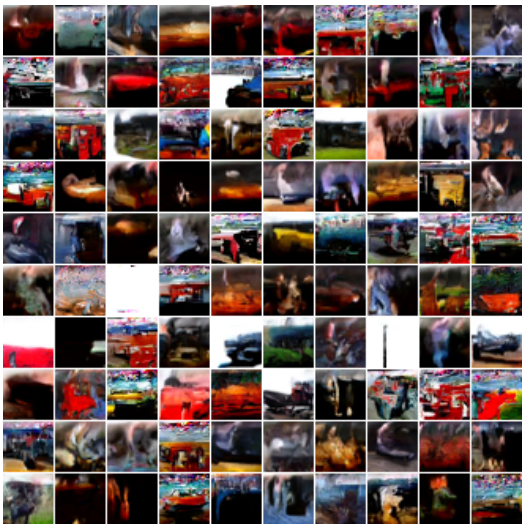
Figure 6: 训练的transformer模型的无条件样本mnist。见正文§4.2.1。



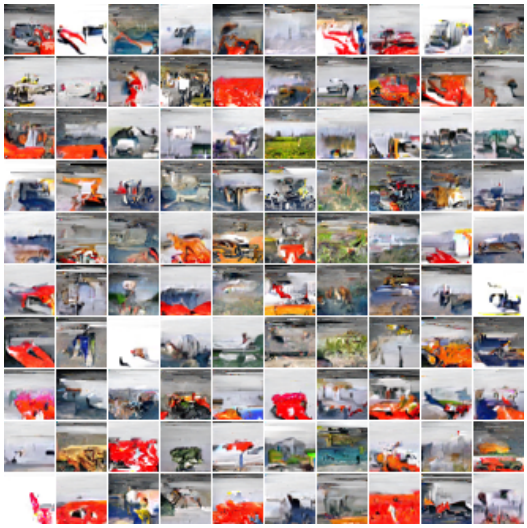
(a) 闭塞的 (b) Softmax (c) 线性(我们的) (d) lsh-1 (e) lsh-4 (f) 原

Figure 7: 所有训练模型的MNIST数字补全。见正文§4.2.1。

(a) Softmax



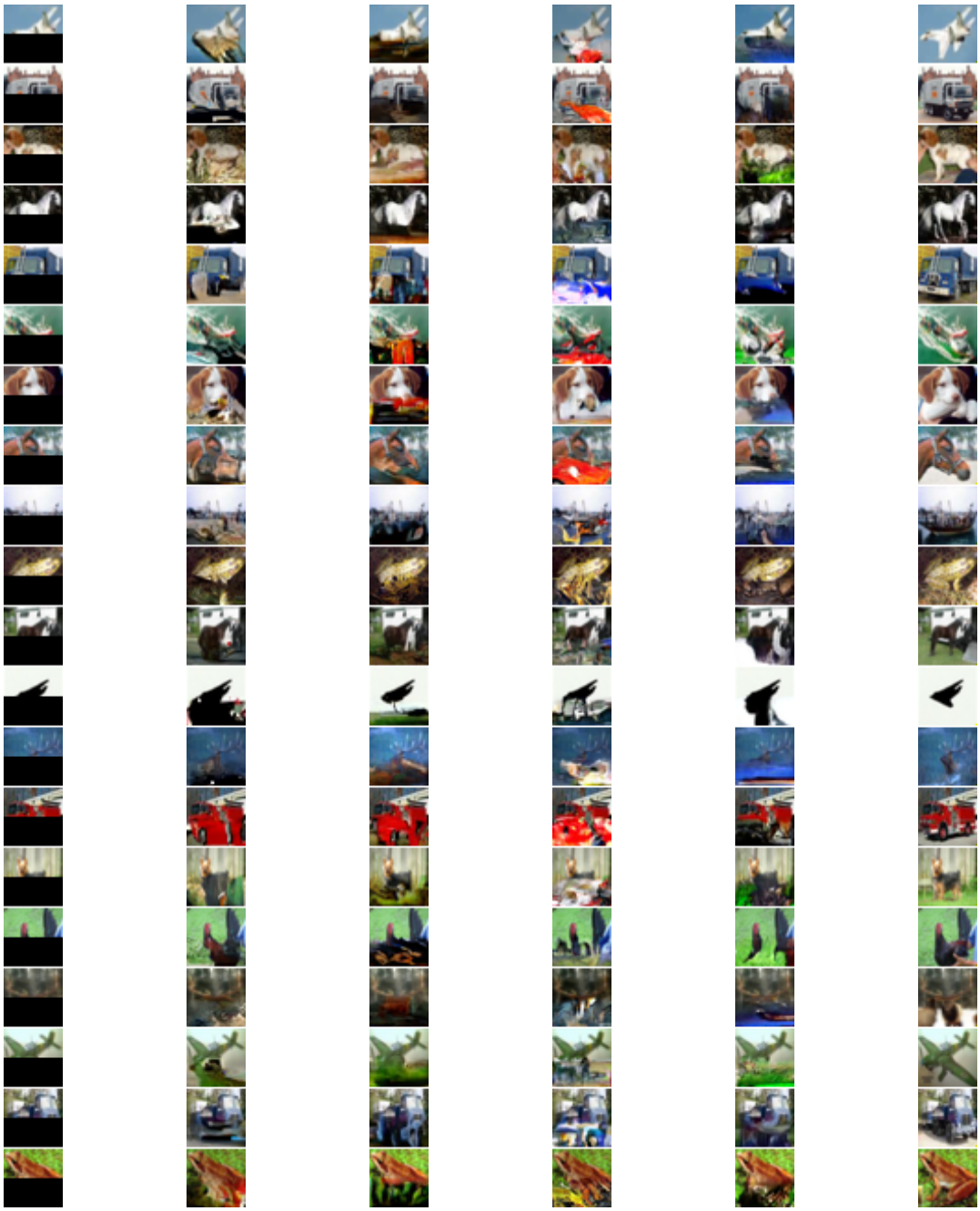
(b) 线性(我们的)



(c) lsh-1

(d) lsh-4

Figure 8: 训练的transformer模型的无条件样本cifar-10。见正文§4.2.2。



(a) 闭塞的 (b) Softmax (c) 线性(我们的) (d) lsh-1 (e) lsh-4 (f) 原

Figure 9: 所有训练过的transformer模型的CIFAR-10图像补全。见正文§4.2.2。