高效transformer综述

Yi Tay YITAY@GOOGLE.COM
Google Research

Mostafa Dehghani DEHGHANI@GOOGLE.COM
Google Research, Brain team

Dara Bahri DBAHRI@GOOGLE.COM
Google Research

Donald Metzler METZLER@GOOGLE.COM
Google Research

Editor: Preprint, Version 2, Updated Mar 2022

Abstract

Transformer模型架构在语言、视觉和强化学习等一系列领域的有效性，最近引起了极大的兴趣。例如，在自然语言处理领域，transformer已经成为现代深度学习堆栈中不可或缺的重要组成部分。最近，许多令人眼花缭乱的"*X-former*"模型被提出——Reformer、Linformer、Performer、Longfor它们改进了原始的Transformer架构，其中许多围绕计算和内存效率进行了改进。为了帮助热心的研究人员在这一热潮中导航，本文对最近的效率味的" X-former "模型进行了大量和深思熟虑的选择，为多个领域的现有工作和模型提供了有组织和全面的概述。

Keywords: Deep Learning, Natural Language Processing, Transformer Models, Attention Models, Neural Networks

1. 简介

transformer (Vaswani et al., 2017)是现代深度学习领域中一股强大的力量。transformer无处不在，并在许多领域产生了巨大的影响，如语言理解(Devlin et al., 2018; Brown et al., 2020; Raffel et al., 2019)和图像处理(Parmar et al., 2018; Carion et al., 2020)。因此，在过去几年中，致力于对模型进行根本改进的大量研究是很自然的(Dehghani et al., 2018; So et al., 2019; Ahmed et al., 2017)。这种巨大的兴趣也刺激了对该模型更有效变体的研究(Kitaev et al., 2020; Roy et al., 2020; Beltagy et al., 2020; Katharopoulos et al., 2020; Tay et al., 2020b; Wang et al., 2020c; Rae et al., 2020; Choromanski et al., 2020b; Dai et al., 2020; Correia et al., 2019; Sukhbaatar et al., 2019a; Vyas et al., 2020)。

最近提出的Transformer模型变体数量激增，研究人员和从业人员可能会发现，跟上创新的速度具有挑战性。截至本文撰写和本文初稿(约2020年8月)，仅在过去6个月里就提出了近12个新的以效率为重点的模型。因此，对现有文献的调查不仅对社区有益，而且非常及时。

自注意力机制是Transformer模型的一个关键定义特征。该机制可以被视为类似图的归纳偏差，用基于相关性的池化操作将序列中的所有标记连接起来。自注意力的一个众所周知的问题是二次时间和内存复杂性，在许多情况下会阻碍模型的可扩展性。最近提出了大量的模型变体来解决这个问题。本文将这类模型命名为"高效*transformer* "。

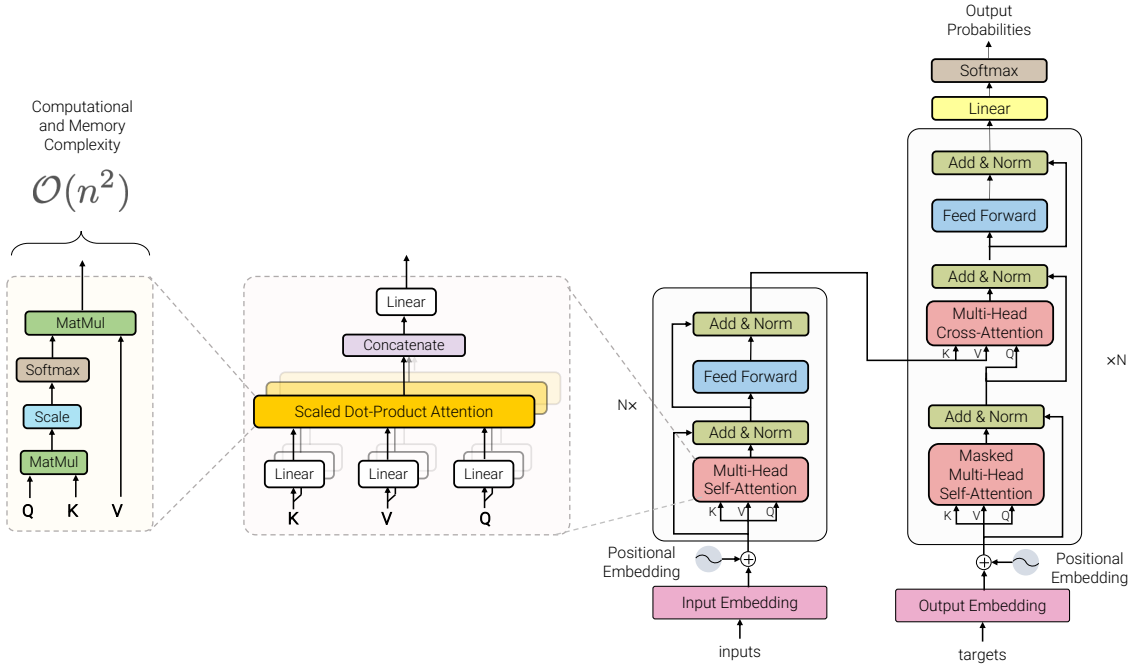


Figure 1: 标准变压器的结构(Vaswani et al., 2017)

模型的效率可以用多种方式来解释。它可能与模型的内存占用有关，当模型运行在加速器上的内存有限时，这一点很重要。效率也可能指的是计算成本，例如在训练和推理期间的FLOPs数量。特别是，对于设备上的应用程序，模型通常必须在高度受限的计算预算内运行。在本综述中，我们提到了transformer在内存和计算方面的效率。我们对这些模型应用于大输入时的表现特别感兴趣。

高效的自注意力模型在长序列建模应用中至关重要。例如，文档、图像和视频通常都是由相对较多的像素或标记组成的。因此，处理长序列的效率对transformer的广泛采用至关重要。

本综述旨在对这类模型的最新进展进行全面概述。本文主要感兴趣的是建模进展和架构创新，以提高transformer的一般效率，包括但不限于解决自注意力机制的二次复杂度问题，或通过池化和/或稀疏性等手段降低计算成本。我们还简要讨论了一般性的改进和其他效率改进，如参数共享。

本文提出一种高效Transformer模型的分类法，以其技术创新和主要用例为特征。回顾了在语言和视觉领域都有应用的Transformer模型，试图巩固整个频谱的文献。本文还提供了许多这些模型的详细介绍，并绘制它们之间的联系。

更新版本的作者说明(2021年12月) 该手稿于2021年12月进行了一轮修订(在第一稿撰写后大约1年零4个月)。主要的变化包括增加我们的讨论，以更好地反映当前时间点的研究状态(新模型，新范式)，并准确反映该研究领域的当前元趋势。在论文的末尾有一个回顾部分。附录中有关于我们迁移到V2版本时发生的有意义的变化日志。

更新版本的作者说明(2022年3月) 我们想在Jan发布更新到arxiv，但忘记了。我们在3月通过添加较新的SOTA稀疏模型(如ST-MoE-32B (Zoph et al., 2022))再次对其进行了轻微修订。

2. transformer背景知识

本节提供完善的Transformer架构的概述(Vaswani et al., 2017)。Transformer是由将Transformer块堆叠在一起形成的多层架构。

Transformer模块的特点是多头自注意力机制，位置前馈网络，层归一化(Ba et al., 2016)模块和残差连接器。Transformer模型的输入通常是形状为$\mathbb{R}^B \times \mathbb{R}^N$的张量，其中$B$是批大小，$N$是序列长度。

输入首先通过一个嵌入层，该层将每个one-hot token表示转换为d_{model}维度的嵌入，即$\mathbb{R}^B \times \mathbb{R}^N \times \mathbb{R}^{d_{model}}$。然后，新的张量与位置编码相加组成，并通过多头自注意力模块。位置编码可以采用正弦输入的形式(参见(Vaswani et al., 2017))或可训练的嵌入。

多头自注意力模块的输入和输出通过残差连接器和一层归一化层连接。然后，多头自注意力模块的输出被传递到一个双层前馈网络，该网络的输入/输出以残差方式类似地与层归一化连接。具有层范数的子层残差连接器表示为:

$$X = \text{LayerNorm}(F_S(X)) + X$$

其中F_S是子层模块，它是多头自注意力或位置前馈层。

2.1 多头自注意力

Transformer模型利用了多头自注意力机制。该机制背后的关键思想是，序列中的每个元素都要学会从序列中的其他标记中收集信息。单个表头的操作定义如下:

$$A_h = \text{Softmax}(\alpha Q_h K_h^\top) V_h,$$

其中X是$\mathbb{R}^{N \times d}$中的一个矩阵，α是一个缩放因子，通常设置为$\frac{1}{\sqrt{d}}$, $Q_h = X\boldsymbol{W}_q, K_h = X\boldsymbol{W}_k$和$V_h = X\boldsymbol{W}_v$是应用于输入序列的时间维度上的线性变换，$\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v \in \mathbb{R}^{d \times \frac{d}{H}}$是用于将输入$X$投影到$d$维度的输出张量的查询、键和值投影的权重矩阵(参数)。N_H是正面的个数。Softmax按行应用。

head $A_1 \cdots A_H$的输出连接在一起并传递到一个密集层。因此，输出Y可以表示为$Y = \boldsymbol{W}_o[A_1 \cdots A_H]$，其中$\boldsymbol{W}_o$是输出线性投影。请注意，$A$的计算通常以并行方式完成，即考虑$\mathbb{R}^B \times \mathbb{R}^N \times \mathbb{R}^H \times \mathbb{R}^{\frac{d}{H}}$的张量并并行计算所有头部的线性变换。

注意力矩阵$A = QK^\top$主要负责学习序列中标记之间的对齐分数。在这个公式中，取查询中的每个元素/标记(Q)和键(K)之间的点积。这推动了自注意力中的自对齐过程，令牌学习从彼此收集。

2.2 位置前馈层

然后，自注意力模块的输出被传递到具有ReLU激活的双层前馈网络。这个前馈层独立地作用于每个位置。表达式如下:

$$F_2(ReLU(F_1(X_A)))$$

其中F_1和F_2是$Wx + b$形式的前馈函数。

2.3 把它们放在一起

每个变压器块可以表示为:

$$X_A = \text{LayerNorm}(\text{MultiheadAttention}(X, X)) + X$$
$$X_B = \text{LayerNorm}(\text{PositionFFN}(X_A)) + X_A$$

其中X为变压器块的输入，X_B为变压器块的输出。请注意，MultiheadAttention()函数接受两个参数张量，一个用于查询，另一个用于键值。如果第一个参数和第二个参数是相同的输入张量，则这是MultiheadSelfAttention机制。

2.4 关于transformer的计算成本

transformer的计算成本来自多个因素。首先，计算注意力矩阵所需的内存和计算复杂度在输入序列长度上是二次的，即$N \times N$。特别是$QK^\top$矩阵乘法运算本身就消耗N^2时间和内存。这限制了自注意力模型在需要处理长序列的应用中的总体效用。内存限制往往更适用于训练(由于梯度更新)，通常对推理的影响较小(没有梯度更新)。自注意力的二次成本在训练和推理中都影响速度[1]。自注意力机制的计算成本部分地贡献了Transformer的整体计算成本。相当多的计算仍然来自每个Transformer块的两层前馈层(大约一半的计算时间和/或FLOPs)。FFN的复杂度与序列长度成线性关系，但通常仍然是昂贵的。因此，最近的大部分工作都探索了稀疏性(Lepikhin et al., 2020; Fedus et al., 2021)作为一种不产生计算成本的扩展FFN的方法。高效的注意力和高效的模型通常是正交的——尽管一些高效的注意力方法明确地旨在减少序列长度(Dai et al., 2020)，因此也节省了这两方面的计算成本。效率和计算成本通常是一件复杂的事情，我们建议读者阅读(Dehghani et al., 2021)以了解更多关于权衡、复杂性等细节。

2.5 变压器方式

注意Transformer块使用方式的差异是很重要的。transformer主要可以以三种方式使用，即:(1)仅用于编码器(例如，用于分类)，(2)仅用于解码器(例如，用于语言建模)，以及(3)编码器-解码器(例如，用于机器翻译)。在编码器-解码器模式下，通常有多个多头自注意力模块，包括编码器和解码器中的标准自注意力，以及编码器-解码器交叉注意力，允许解码器利用编码器的信息。这影响了自注意力机制的设计。在编码器模式中，没有限制或约束自注意力机制必须是因果的，即仅依赖于现在和过去的token。在编码器-解码器设置中，解码器中使用的自注意力(即跨解码位置)必须是因果的，因为每个自回归解码步骤只能依赖于之前的token，而编码器中使用的自注意力则不必。对于许多高效的自注意力设计来说，满足这一要求可能具有挑战性。

Transformer模型的使用模式通常取决于目标应用程序。给定一个输入序列，该序列通常通过一个编码器堆栈传递。在这个阶段，可能有太多的选择。对于多类分类，具有Softmax输出的线性层通常将序列表示投影到类的数量。在BERT (Devlin et al., 2018)的情况下，这是一个*[CLS]*标记，作为前缀附加到序列的开始。最近的工作还探索了分类的编码器-解码器架构的使用，如T5 (Raffel et al., 2019)。纯解码器模型通常用于生成，并使用语言建模目标(预测下一个token)进行训练。由于损失的性质，这些模型通常优于开放式的一代(Brown et al., 2020)。只有解码器的模型需要是因果的，并且需要应用上三角掩码来

1. We would like to emphasize that complexity does not always translate to real world throughput or latency. A model of linear complexity can be slower than a model with quadratic complexity in practice.

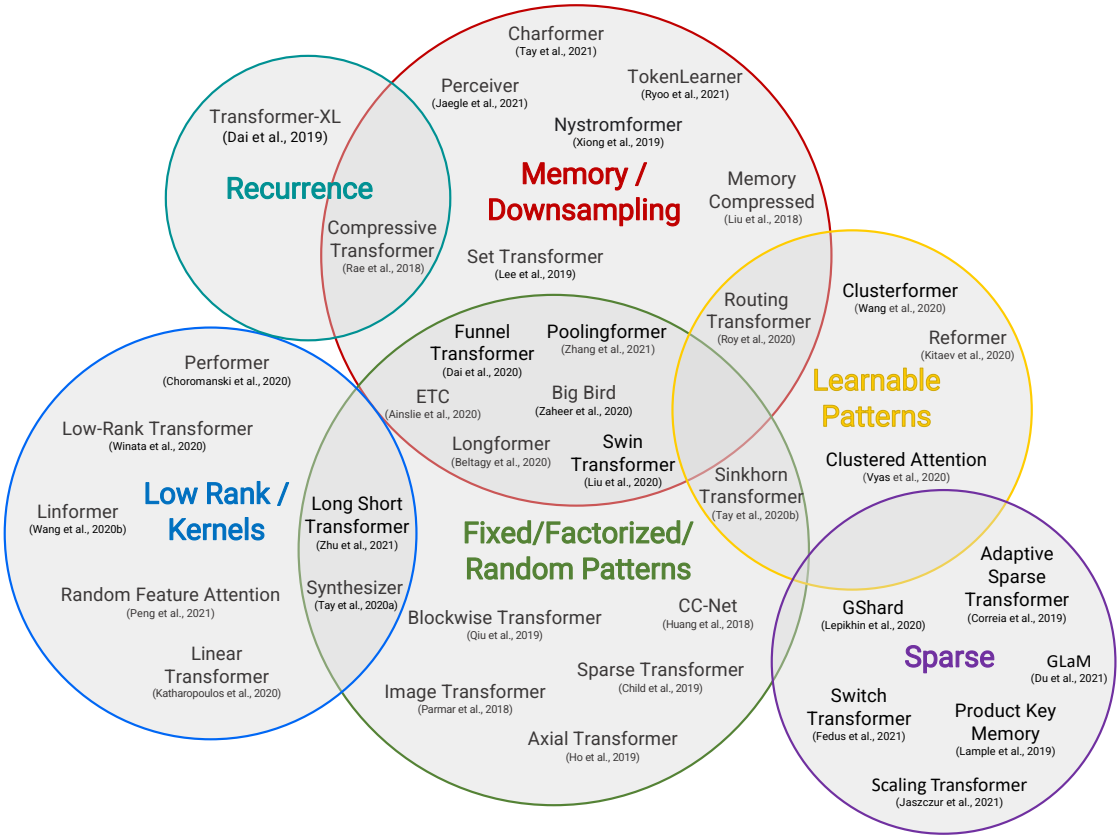


Figure 2: 高效Transformer架构的分类。

防止token窥视未来。有兴趣的读者请访问(Raffel et al., 2019)，以获得各种Transformer模式的更详细描述。

2.6 应用程序

transformer具有广泛的应用，从语言到视觉，语音和强化学习。它最初是在自然语言处理中序列到序列机器翻译的背景下被引入的。其次，考虑到预训练模型(如BERT (Devlin et al., 2018))的同时发展，transformer的大多数应用都是在语言的上下文中。因此，对这一系列高效transformer的许多早期改进都集中在语言处理应用程序(Beltagy et al., 2020; Ainslie et al., 2020)上。由于历史原因，这篇调查论文稍微倾向于语言。然而，也值得注意的是，在我们的调查中考虑的大量论文也考虑了需要序列处理器的多模态应用。例如Roy et al. (2020); Choromanski et al. (2020b); Tay et al. (2020b); Child et al. (2019)考虑图像或其他模态(如蛋白质)上的生成建模任务。

3. 高效Transformer模型综述

在本节中，我们对高效的Transformer模型进行高层次的概述。首先介绍不同模型的特征。表1列出了迄今为止发布的高效Transformer，图2展示了几个关键高效Transformer模型的图形化概述。

Model / Paper	Complexity	Decode	Class
Memory Compressed (Liu et al., 2018)	$\mathcal{O}(N_c^2)$	✓	FP+M
Image Transformer (Parmar et al., 2018)	$\mathcal{O}(N.m)$	✓	FP
Set Transformer (Lee et al., 2019)	$\mathcal{O}(kN)$	✗	M
Transformer-XL (Dai et al., 2019)	$\mathcal{O}(N^2)$	✓	RC
Sparse Transformer (Child et al., 2019)	$\mathcal{O}(N\sqrt{N})$	✓	FP
Reformer (Kitaev et al., 2020)	$\mathcal{O}(N \log N)$	✓	LP
Routing Transformer (Roy et al., 2020)	$\mathcal{O}(N\sqrt{N})$	✓	LP
Axial Transformer (Ho et al., 2019)	$\mathcal{O}(N\sqrt{N})$	✓	FP
Compressive Transformer (Rae et al., 2020)	$\mathcal{O}(N^2)$	✓	RC
Sinkhorn Transformer (Tay et al., 2020b)	$\mathcal{O}(B^2)$	✓	LP
Longformer (Beltagy et al., 2020)	$\mathcal{O}(n(k+m))$	✓	FP+M
ETC (Ainslie et al., 2020)	$\mathcal{O}(N_g^2 + NN_g)$	✗	FP+M
Synthesizer (Tay et al., 2020a)	$\mathcal{O}(N^2)$	✓	LR+LP
Performer (Choromanski et al., 2020a)	$\mathcal{O}(N)$	✓	KR
Funnel Transformer (Dai et al., 2020)	$\mathcal{O}(N^2)$	✓	FP+DS
Linformer (Wang et al., 2020c)	$\mathcal{O}(N)$	✗	LR
Linear Transformers (Katharopoulos et al., 2020)	$\mathcal{O}(N)$	✓	KR
Big Bird (Zaheer et al., 2020)	$\mathcal{O}(N)$	✗	FP+M
Random Feature Attention (Peng et al., 2021)	$\mathcal{O}(N)$	✓	KR
Long Short Transformers (Zhu et al., 2021)	$\mathcal{O}(kN)$	✓	FP + LR
Poolingformer (Zhang et al., 2021)	$\mathcal{O}(N)$	✗	FP+M
Nyströmformer (Xiong et al., 2021b)	$\mathcal{O}(kN)$	✗	M+DS
Perceiver (Jaegle et al., 2021)	$\mathcal{O}(kN)$	✓	M+DS
Clusterformer (Wang et al., 2020b)	$\mathcal{O}(N \log N)$	✗	LP
Luna (Ma et al., 2021)	$\mathcal{O}(kN)$	✓	M
TokenLearner (Ryoo et al., 2021)	$\mathcal{O}(k^2)$	✗	DS
Adaptive Sparse Transformer (Correia et al., 2019)	$\mathcal{O}(N^2)$	✓	Sparse
Product Key Memory (Lample et al., 2019)	$\mathcal{O}(N^2)$	✓	Sparse
Switch Transformer (Fedus et al., 2021)	$\mathcal{O}(N^2)$	✓	Sparse
ST-MoE (Zoph et al., 2022)	$\mathcal{O}(N^2)$	✓	Sparse
GShard (Lepikhin et al., 2020)	$\mathcal{O}(N^2)$	✓	Sparse
Scaling Transformers (Jaszczur et al., 2021)	$\mathcal{O}(N^2)$	✓	Sparse
GLaM (Du et al., 2021)	$\mathcal{O}(N^2)$	✓	Sparse

Table 1: 高效Transformer模型总结。第一部分中的模型主要是有效的注意力方法。下一节中的模型一般指稀疏模型。类缩写包括:FP =固定模式或固定模式的组合，M =记忆，LP =可学习模式，LR =低秩，KR =核RC =递归，DS =下采样。此外，N通常指序列长度，B是局部窗口(或块)大小。N_g和N_c分别表示全局模型记忆长度和卷积压缩序列长度。

3.1 高效transformer的分类

本节概述了高效Transformer模型的一般分类，其特征是它们的核心技术和主要用例。虽然大多数这些模型的主要目标是通过自注意力机制来提高记忆复杂度，但也包括了提高Transformer架构总体效率的方法。

- **固定模式(FP)**——对自注意力的最早修改只是通过将视野限制为固定的、预定义的模式(如局部窗口和固定步长的块模式)来稀疏化注意力矩阵。

  - **分块模式**这种技术在实践中最简单的例子是分块(或分块)范式，它通过将输入序列分块为固定块来考虑局部感受野的块。这样做的模型示例包括分块(Qiu et al., 2019)和/或局部注意力(Parmar et al., 2018)。使用$B << N$将输入序列分块，将复杂性从N^2降低到B^2(块大小)，显著降低了成本。这些分块或分块方法是许多更复杂模型的基础。
  - **跨步模式**另一种方法是考虑跨步注意模式，即只以固定的间隔出席。稀疏Transformer (Child et al., 2019)和/或Longformer (Beltagy et al., 2020)等模型采用大步或"扩张"窗口。
  - **压缩模式**——这里的另一种攻击是使用一些池化操作符对序列长度进行下采样，使其成为一种固定模式的形式。例如，压缩注意力(Liu et al., 2018)使用跨步卷积来有效地减少序列长度。

- **模式组合(CP)**——组合[2]方法的关键思想是通过组合两个或更多不同的访问模式来提高覆盖率。例如，稀疏Transformer (Child et al., 2019)通过将其一半的头部分配给每个模式来结合大步注意力和局部注意力。类似地，axis Transformer (Ho et al., 2019)应用给定高维张量作为输入的自注意力计算序列，每个沿着输入张量的单个轴。从本质上讲，模式的组合以与固定模式相同的方式降低了内存复杂度。然而，不同的是，多个模式的聚合和组合提高了自注意力机制的整体覆盖率。

- **可学习的模式(LP)** -对固定的、预先确定的模式的扩展是可学习的模式。不出所料，使用可学习模式的模型旨在以数据驱动的方式学习访问模式。学习模式的一个关键特征是确定标记相关性的概念，然后将标记分配给桶或集群(Vyas et al., 2020; Wang et al., 2020b)。值得注意的是，Reformer (Kitaev et al., 2020)引入了一种基于哈希的相似性度量，以有效地将token聚集到块中。类似地，路由Transformer (Roy et al., 2020)采用在线k——意味着对令牌进行聚类。同时，Sinkhorn排序网络(Tay et al., 2020b)通过学习对输入序列的块进行排序，暴露了注意力权重的稀疏性。在所有这些模型中，相似度函数与网络的其余部分一起进行端到端的训练。可学习模式的关键思想仍然是利用固定模式(分块模式)。然而，这类方法学会了对输入标记进行排序/聚类——在保持固定模式方法的效率优势的同时，启用更优的序列全局视图。

- **神经记忆**——另一种突出的方法是利用可学习的边记忆模块，可以一次访问多个标记。一种常见的形式是全局神经[3]记忆，它能够访问整个序列。全局标记作为一种模型记忆形式，它学习从输入序列标记中收集信息。这是在Set Transformers (Lee et al., 2019)中首次引入的诱导点方法。这些参数通常被解释为'memory '，并用作未来处理的临时上下文形式。这可以被认为是一种参数关注的形式(Sukhbaatar et al., 2019b)。全局内存标记也用于ETC (Ainslie et al., 2020)和Longformer (Beltagy et al., 2020)。在有限的神经记忆(或诱导点)下，我们能够对输入序列进行初步的类似池化的操作，以压缩输入序列——这是设计高效的自注意力模块时可以使用的一种巧妙的技巧。

- **低秩方法**——另一种新兴技术是通过利用自注意力矩阵的低秩近似来提高效率。其关键思想是在$N \times N$矩阵中假设低秩结构。Linformer (Wang et al., 2020c)是这种技术的一个经典示例，因为它将键和值的长度维度投射到低维表示($N \rightarrow k$)。很容

2. 我们注意到，这也经常被称为因子分解方法，例如，在Child et al. (2019)中。我们决定将这类模型称为组合方法，因为(1)它更适合这些模型的实际功能;(2)避免与矩阵分解或低秩方法混淆。
3. 我们在这里使用术语neural来指模型中经常表现的类表示记忆。

易看出，低秩方法改善了自注意力的记忆复杂度问题，因为$N \times N$矩阵现在被分解为$N \times k$。

- **核**——最近另一种提高transformer效率的流行方法是通过核化来查看注意力机制。内核的使用(Katharopoulos et al., 2020; Choromanski et al., 2020a)使自注意力机制的巧妙数学重写成为可能，以避免显式计算$N \times N$矩阵。由于核是注意力矩阵的一种近似形式，它们也可以被视为一种低秩方法(Choromanski et al., 2020a)。该领域最近工作的例子包括表演者、线性transformer和随机特征注意力(RFA, (Peng et al., 2021))

- **递归**——对分块方法的自然扩展是通过递归连接这些块。Transformer-XL (Dai et al., 2019)提出了一种连接多个段和块的段级递归机制。在某种意义上，这些模型可以被视为固定模式模型。然而，由于它与其他块/局部方法的偏差，我们决定创建自己的类别。

- **下采样**-另一种降低计算成本的流行方法是降低序列的分辨率，从而降低相应的计算成本。这类模型的例子包括Perceiver (Jaegle et al., 2021), Funnel Transformers (Dai et al., 2020), Swin Transformer (Liu et al., 2021b)和Charformer (Tay et al., 2021c)模型。值得注意的是，这类模型还可能与利用记忆标记作为模型的模型存在某种形式的重叠，如Set Transformer，也可以被视为一种下采样形式，尽管在注意力机制中。最近的Nyströmformer (Xiong et al., 2021b)，从表面上看，似乎是一种低等级或基于内核的方法。然而，它实际上是一种下采样方法，其中"地标"只是基于池化进行跨越——与设置Transformer、漏斗Transformer或感知器的精神类似。

- **稀疏模型和条件计算**——虽然不是专门针对注意力模块，但稀疏模型稀疏地激活了参数的子集，这通常会提高参数与FLOPs的比率。这类模型的例子包括开关转换器(Fedus et al., 2021), ST-MoE (Zoph et al., 2022), GShard (Lepikhin et al., 2020)，产品关键内存层(Lample et al., 2019)。在我们研究的模型范围内，稀疏模型通常在自适应的基础上运行，其中稀疏性通常是学习的(通过专家混合机制)。在这种情况下，我们还可以认为注意力权重的稀疏化属于这种范式。因此，我们认为注意力与固定模式或学习模式有密切联系。然而，我们认为，基于稀疏高效的整个研究方向(Roller et al., 2021; Lewis et al., 2021; Lepikhin et al., 2020; Du et al., 2021)的出现，应该保证一个新的高效transformer类别。

这些桶是不同高效Transformer模型的广泛表征。实际上，这些桶之间没有明显的界限，因为模型可能由多种技术创新组成。例如，路由Transformer (Roy et al., 2020)中的k -均值聚类也可以解释为全局模型记忆方法的一种形式，因为可以将质心视为参数化的模型记忆。然而，在Reformer中，聚类用于学习注意力权重的稀疏模式。此外，池(Liu et al., 2018)也可以解释为模型内存机制的一种形式。我们还注意到，最近的xformer模型(约2021年12月)已经开始采用某种形式的两阶段注意力机制。很多时候，这些注意力机制明确地结合了上述的一种或多种形式，例如，局部窗口和池化中的记忆(Zhang et al., 2021)，或利用固定窗口的低秩注意力的长短transformer (Zhu et al., 2021)(例如，局部注意力与类似linformer的归纳偏差的结合)。

3.2 高效Transformer模型的详细介绍

本节深入探讨了几个关键的高效Transformer模型的细节，讨论了它们的优缺点和独特的讨论点。这里的目标不是详尽地介绍所有这些模型，而是涵盖一个具有代表性的模型样本。

本节结构 我们首先讨论局部和固定模式模型，如内存压缩Transformer (Liu et al., 2018)和图像Transformer (Parmar et al., 2018)。然后讨论集合transformer (Lee et al., 2019)，一种利用全局模型内存的早期方法。接下来，我们继续介绍利用模式组合的模型，如稀疏transformer (Child et al., 2019), CCNet (Huang et al., 2019)和轴向transformer (Ho et al., 2019)。接下来，我们讨论Longformer (Beltagy et al., 2020)等(Ainslie et al., 2020)，作为基于内存的稀疏Transformer方法的示例。然后，我们将详细介绍包含可学习模式(LP)的模型，如路由transformer (Roy et al., 2020)、Reformer (Kitaev et al., 2020)和Sinkhorn transformer (Tay et al., 2020b)。之后，介绍了Linformer (Wang et al., 2020c)和synthesizer (Tay et al., 2020a)，这些模型可以被认为是低秩因子分解方法。然后讨论基于核方法的模型，如Performer (Choromanski et al., 2020a)和线性transformer (Katharopoulos et al., 2020)。接下来，我们讨论了基于分段递归的模型，如Transformer-XL (Dai et al., 2019)和Transformers (Rae et al., 2020)。讨论了稀疏模型族，主要利用专家混合(MoE)类型架构和条件计算来实现计算效率。本节的逻辑流程是松散地按时间顺序组织的，而不是按类别组织的(递归性或稀疏性等更正交的方法除外)。我们相信这在教学上是有帮助的。

3.2.1 存储压缩变压器

内存压缩Transformer (Liu et al., 2018)是修改Transformer以更好地处理较长序列的早期尝试之一。记忆压缩transformer引入的修改有两个方面:定位注意力广度和使用记忆压缩注意力。

局部注意广度 在transformer中处理长序列的一个简单解决方案是将注意力广度限制在局部邻域。Liu et al. (2018)提出将输入序列划分为长度相似的块，以便在每个块内独立计算自注意力。这使每个块的注意力成本保持不变，因此激活的数量与输入长度呈线性增长。

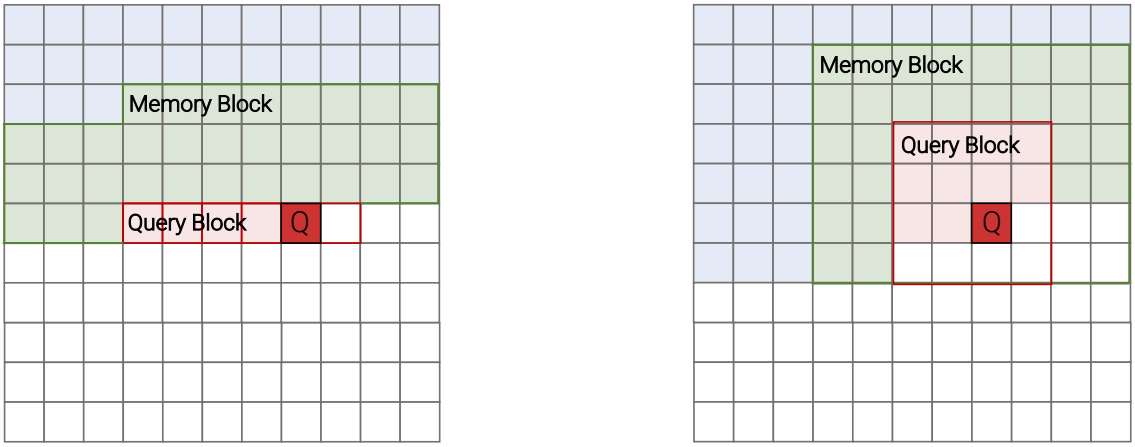
记忆压缩注意力 内存压缩注意力背后的思想是使用大步卷积减少键和值的数量，而查询保持不变。这导致了注意力矩阵大小的减少，以及基于压缩因子的注意力计算的减少，压缩因子取决于内核大小和卷积的步长。记忆压缩注意力让模型在整个输入序列中全局交换信息，而不是局部注意力。

计算和存储复杂度 对于b大小的块，每个块中的自注意力的计算和内存成本是$\mathcal{O}(b^2)$。假设有n/b块，局部注意力的计算和内存成本是$\mathcal{O}(b.n)$。对于内存压缩的注意力，应用内核大小和步长为k的卷积，注意力机制的计算和内存成本降低到$\mathcal{O}(n \cdot n/k)$。

3.2.2 图像转换器

图像Transformer (Parmar et al., 2018)，受卷积神经网络的启发，将自注意力的感受野限制为仅局部邻域。这有助于模型扩展以处理更大的批量大小，同时保持可能性损失可处理。除了效率之外，采用局部性的概念也是处理图像的一个理想的归纳偏差。Image Transformer提供了编码器-解码器架构，其中编码器为输入中的每个像素通道生成上下文表示，解码器在每个时间步长自回归地为每个像素生成一个通道。

局部注意广度 将感受野限制在局部邻域(Parmar et al., 2018, 2019)解决了在大输入上运行全局自注意力的计算和内存成本问题，但改变每个查询位置的邻域将禁止将自注意力的计算打包为两个矩阵乘法。为了避免这种情况，Image Transformer建议将输入划分为“查询块”及其相关的“内存块”，其中对于单个查询块的所有查询，模型都会处理相同的内存



(a) 1-维局部注意力　　(b) 2-维局部注意

Figure 3: 图像Transformer对二维输入的注意力广度。

块。查询块及其关联记忆块邻域的选择有两种不同的方案:一维局部注意力和二维局部注意力。在这里，我们在解码器的情况下简要解释这些方案。

对于一维局部注意力，图像按栅格顺序扁平化[4]并划分为长度为l_q的非重叠查询块Q，对于每个查询块，从Q中的相同像素以及在查询像素之前生成的固定数量的像素l_m构建内存块M。在二维局部注意力中，像素按栅格顺序生成。针对二维局部注意力，将图像划分为多个长度为$l_q = w_q \times h_q$的不重叠矩形查询块。内存块将查询块扩展到顶部、左侧h_m和w_m像素以及右侧w_m像素，即$l_m = (w_q \times q_h) + 2 \times (h_m + w_m)$。查询像素可以处理所有其他像素。在二维局部注意力中，图像中的像素是一个接一个的查询块生成的。生成的块是按栅格顺序排列的，每个块内生成的像素也是如此。

计算和存储复杂度　在Image Transformer中，注意力矩阵的形状为$l_q \times m$，其中l_q是为查询块选择的长度，M是内存块的长度(实际上是$l_q + l_m$)。考虑到内存块不重叠，我们必须计算$n \times l_q$注意力矩阵。因此，Image Transformer的内存和计算复杂性是$\mathcal{O}(n \cdot m)$。

限制条件　Image Transformer通常将注意力机制中的上下文限制在局部邻域内，可以以失去全局感受野为代价减少内存和计算成本。这可能是一个需要全局信息来解决任务的问题。此外，局部注意力相对于区域长度具有二次复杂度，因此在性能和计算复杂度之间的权衡中引入了一个额外的超参数。

3.2.3 成套变压器

Set Transformer (Lee et al., 2019)为集合输入问题调整了Transformer模型，即输入是一组特征，输出是该集合的某个函数的问题(因此对输入特征的排列或排序是不变的)。Set Transformer利用注意力来捕获输入集元素之间的交互。此外，应用从稀疏高斯过程文献中诱导点的思想，在输入集的大小上将注意力的复杂性从二次型降低到线性型。

4. 给定一个由像素组成的网格的2D图像，从左到右逐行水平扫描像素，创建一个栅格顺序。

涉及对象集合的问题通常具有排列不变性:无论对象在集合中的顺序如何，集合的目标值都是相同的。Zaheer et al. (2017)证明了所有置换不变函数都可以表示为下列函数形式:

$$\text{network}\left(\{x_1,\ldots,x_N\}\right)=\rho\left(\text{pool}\left(\{\phi(x_1),\ldots,\phi(x_N)\}\right)\right),$$

其中池化函数pool是一个简单的求和，ϕ和ρ是连续函数。这种形式可以解释为编码器ϕ和解码器$\rho\left(\text{pool}(\cdot)\right)$的组合。虽然这种形式是置换不变函数空间中的通用近似器，但尚不清楚此类模型在实践中适合任务的程度。Set Transformer提出了一个可以被视为编码器和池化解码器的解决方案，但与上面给出的形式不同，编码器和解码器可以单独处理输入元素，池化函数是参数化的。

注意块 该模型引入了以下结构:"多头注意力块" (MAB)、"集合注意力块" (SAB)、"诱导集合注意力块" (ISAB)和"多头注意力池化" (PMA)。它们定义如下。

$$\begin{aligned}\mathbf{MAB}(\mathbf{X},\mathbf{Y}) &:= \text{LayerNorm}\left(H+\text{rFF}(H)\right),\\ H &:= \text{LayerNorm}\left(X+\text{MultiheadAttention}(X,Y)\right),\\ \mathbf{SAB}(\mathbf{X}) &:= \text{MAB}(X,X),\\ \mathbf{ISAB_m}(\mathbf{X}) &:= \text{MAB}\left(X,\text{MAB}(I_m,X)\right).\\ \mathbf{PMA_k}(\mathbf{X}) &:= \text{MAB}\left(S_k,\text{rFF}(X)\right).\end{aligned}$$

在这里，$X\in\mathbb{R}^{N\times d}$表示$Nd$按行堆叠的维度输入/输出，rFF是一个参数化的前馈层，它分别对其输入矩阵的每一行进行操作。$I_m\in\mathbb{R}^{m\times d}$表示$m$可训练$d$维的诱导点“，而$S_k\in\mathbb{R}^{k\times d}$表示$k$可训练$d$维的种子向量"($k$设置为1，除非需要$k>1$相关的输出)。Set Transformer的编码器只是N层的SAB或ISAB(在实践中N通常设置为2)，而其解码器由:

$$\mathbf{Decoder}(\mathbf{X}):=\text{rFF}\left(\text{SAB}\left(\text{PMA}_k(X)\right)\right).$$

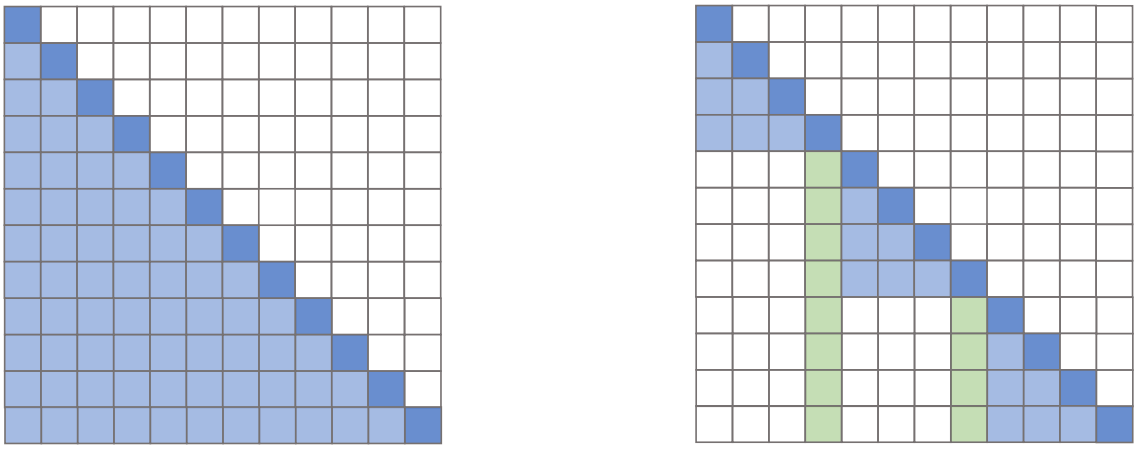
很容易看出ISAB和SAB都是置换等变的——换句话说，如果输入以某种方式置换，那么块的相应输出也以完全相同的方式置换。同时，池化层PMA具有排列不变性。由于函数组合，即分层，保留了这些属性，Set Transformer编码器-解码器组合是置换不变的。

效率 我们可以将在每个ISAB层中学习的m诱导点I_m理解为一种静态模型记忆的形式。除了将自参与SAB层的$\mathcal{O}(Nn^2)$复杂性降低到$\mathcal{O}(Nmn)$之外，当输入集很大时，这一减少尤其有价值，诱导点还有效地编码了一些有助于解释其输入的全局结构。例如，在平摊聚类问题中，人们试图学习将输入点集映射到集合内点的聚类中心，学习到的诱导点可以适当分布，以便编码器可以通过它们与诱导点的接近程度有效地比较查询元素。

在池化层PMA_k中使用的可训练的k种子S_k可以以类似的方式视为静态模型内存，降低了体系结构的内存和运行时复杂性。

3.2.4 稀疏变压器

稀疏Transformer (Child et al., 2019)提出了一个简单的初始尝试，以降低标准自注意力机制的二次复杂度。其关键思想是通过仅在稀疏数量的q_i,k_j对上计算注意力，将密集注意力矩阵减少为稀疏版本。稀疏Transformer采用由步长和局部邻域定义的固定注意力模式。计算被分解，其中局部和步幅模式在头部之间被分割。



(a) 变压器　　(b) 稀疏变压器

Figure 4: transformer中密集自注意力和稀疏固定注意力的注意力矩阵模式说明。右图中的蓝色代表局部自注意力，绿色代表稀疏注意力的大步分量。

局部注意力头　**稀**疏Transformer中有一半的头部致力于局部注意力。

$$\hat{A}_{ij} = \begin{cases} Q_i(K)_j^\top), & \text{if } \lfloor j/N \rfloor = \lfloor i/N \rfloor \\ 0 & \text{otherwise} \end{cases}$$

其中A_{ij}为q_i, k_j的注意力权重，$\lfloor\ \rfloor$为地板操作。在本例中，我们只计算$\lfloor j/N \rfloor = \lfloor i/N \rfloor$(在同一块内)的注意力。

大步注意头　**另**外一半的头是固定的跨步模式。具体来说，

$$\hat{A}_{ij} = \begin{cases} Q_i(K)_j^\top), & \text{if } (i-j) \mod N = 0 \\ 0 & \text{otherwise} \end{cases}$$

稀疏注意力分解的最终结果如图4所示。我们感兴趣的是(Yun et al., 2020)，以了解有关稀疏注意力机制的表现力的其他理论分析。

参数和存储复杂度　**自**注意力机制中的修改不会改变模型的参数成本，因为模型仍然保留原始Transformer模型的Q, K, V转换。注意力层的记忆复杂度从$\mathcal{O}(n^2)$降低到$\mathcal{O}(n \log n)$。

限制条件　**稀**疏Transformer的实现需要自定义GPU内核来实现矩阵乘法的特定块稀疏变体，而在tpu等其他硬件上无法轻松实现。

3.2.5 轴向变压器

axis Transformer (Ho et al., 2019; Weissenborn et al., 2019)在自注意力机制的简单而有效的设置中使用因子分解来处理组织为多维张量的大型输入。轴向Transformer没有将注意力应用于输入的扁平版本，而是简单地应用多个注意力，每个注意力沿着输入张量的单个轴。事实上，每个注意力都是沿着一个特定轴混合信息，同时保持其他轴上的信息独立。由于任何单轴的长度通常比元素的总数小得多，轴向Transformer大大节省了计算和内存。

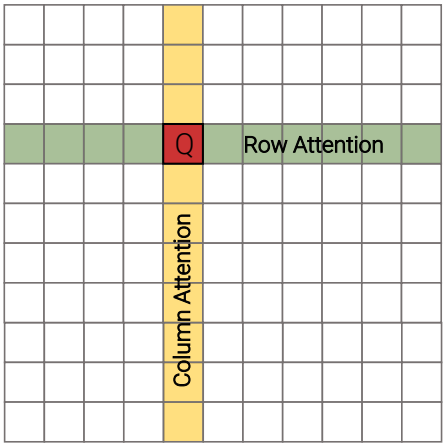


Figure 5: 轴向Transformer对二维输入的注意广度。

axis Transformer提供了编码器-解码器架构。对于解码，为了能够实现因果掩码，axis Transformer将轴向注意力与移位操作相结合。例如，对于二维张量模型，像素是按栅格顺序生成的，为此，模型首先通过未掩码的行和未掩码的列注意力对所有像素进行编码。然后，对于每一行，模型应用未掩码的行和掩码的列注意力来集成先前采样的行。最后，该模型将编码表示向上移动，以确保条件信息满足因果关系，并运行掩码行注意力以在图像中采样新行。

与Sparse Transformer等类似方法相比，axis Transformer的优势在于，虽然它提供了全局感受野，但易于实现，不需要自定义内核来高效实现。

计算和存储复杂度 在内存和计算复杂度方面，在N大小的正方形图像上，axis Transformer在$\mathcal{O}(n\sqrt{n})$中执行注意力计算，比正常的自注意力节省$\mathcal{O}(\sqrt{n})$。例如，在N像素的正方形图像上，组织在$b \times b$网格中，轴向Transformer运行长度为b的b注意力序列，这是复杂的$\mathcal{O}(b.b^2)$。在更一般的情况下，对于d维形状的张量$N = N^{1/d} \times \ldots \times N^{1/d}$，axis Transformer比标准的自注意力节省了$\mathcal{O}(N^{(d-1)/d})$因子的资源。

3.2.6 长形机

Longformer (Beltagy et al., 2020)是Sparse Transformer的一个变体。与稀疏Transformer相比，它的关键区别是“扩张的滑动窗口”，可以在不牺牲稀疏性的情况下实现更好的长程覆盖。这是通过在注意模式中存在间隙来增加感受野来实现的。随着模型深入，Longformer还逐渐增加感受野，用较低的层来建模局部模式，用较高的层来建模全局模式。

全球关注 对于分类任务，Longformer采用可以访问所有输入序列的全局内存标记。

参数和存储复杂度 模型的复杂性从$\mathcal{O}(n^2)$降低到$\mathcal{O}(nk)$，其中k是窗口的大小。使用全局注意力机制时，Longformer会创建另一组查询键值对这种全局注意力的预测，使注意力层的参数成本增加一倍。

3.2.7 扩展的变压器结构(ETC)

ETC模型(Ainslie et al., 2020)是稀疏Transformer家族中的另一种变体。引入了一种新的全局-局部注意力机制。这种新的注意力机制有四个组成部分，即(1)全局到全局(g2g)、

全局到局部(g2l)、局部到全局(l2g)和局部到局部(l2l)。除了模型的原始输入外，ETC引入了n_g辅助标记作为原始输入序列的前缀。这些代币被视为全局代币，参与global-to- $*$和$*$ to-global的关注。local-to-local组件充当具有固定半径k的局部注意力。总的来说，ETC在引入全局辅助标记的方式上与Longformer非常相似。这些标记是可训练的参数，可以解释为一种跨序列池化的模型内存形式，以收集全局序列信息。

内存和参数复杂度 ETC模型的内存复杂度是$\mathcal{O}(n_g^2 + n_g N)$，其中$n_g$是全局标记的数量，$N$是输入序列长度。

限制条件 直观地说，很容易观察到ETC不能用于自回归解码。这是因为由于全局的关注，我们无法计算因果掩码。

3.2.8 大鸟

BigBird模型(Zaheer et al., 2020)是另一个用于建模较长的序列的Transformer，主要构建在ETC之上(Ainslie et al., 2020)。大鸟模型由几个关键组件组成，即(1)全局标记，(2)随机注意力(查询关注随机键)，和(3)固定模式(局部滑动窗口)。

全球关注 基本上，使用全局模型内存的想法可以一直追溯到Longformer/ETC和Set Transformer模型。值得注意的是，Big Bird中的全局模型内存被扩展为包含序列中的标记，而不是简单的参数化模型内存。作者将其称为"内部*transformer*构造*(ITC)*"，其中选择一个索引子集作为全局token。这可以解释为一种基于模型内存的方法。

滑动窗口注意力 窗口注意力在早期的基于局部的注意力模型(图像Transformer、压缩注意力和/或稀疏Transformer)中首次提出。在BigBird中，每个查询处理左侧的$w/2$标记和右侧的$w/2$标记。这对应于固定模式(FP)方法。

随机注意 最后，每个查询处理r随机密钥。这种模式是固定的。

内存和参数复杂度 自注意力的记忆复杂度是线性的，即$\mathcal{O}(n)$。BigBird模型没有引入Transformer模型之外的新参数。

限制条件 与ETC类似，BigBird模型不能用于自回归解码。因此，将其限定为仅用于编码器的模型。

3.2.9 路由转换器

路由Transformer (Roy et al., 2020)是一种基于内容的稀疏注意力机制。提出了一种基于聚类的注意力机制，以数据驱动的方式学习注意力稀疏性。第一步是将Q和K投射到尺寸$n \times d$的路由矩阵R中。

$$R = QW_R + KW_R \tag{1}$$

其中W_R是一个$d \times d$正交投影矩阵。

k-均值聚类 R矩阵通过一系列参数化的聚类质心$u_1, u_2 \cdots c_k$进行k -均值聚类。k -表示路由Transformer以在线方式进行训练。为了确保每个簇中有相似数量的token，该模型初始化$\sqrt{n}$簇，计算每个token与簇质心的距离，并为每个质心取相等的top- k。由于聚类中心是可训练的参数，这也让人想起(Sukhbaatar et al., 2019b)提出的全注意力层。

路由策略 **路**由策略定义如下:

$$X_i' = \sum_{j \in C_i, j \leq i} A_{ij} V_j \tag{2}$$

其中C_i是向量R_i所分配的簇。换句话说，i处的令牌只处理同一集群中的令牌。

内存和参数复杂度 **路**由Transformer在聚类机制中引入了额外的参数，即$k \times d$质心向量和W_r投影矩阵。内存复杂度为$\mathcal{O}(n^{1.5})$。

3.2.10 重整器

Reformer (Kitaev et al., 2020)是另一种基于局部敏感哈希(LSH)的高效注意力模型。Reformer还引入了可逆Transformer层，这有助于进一步减少内存占用。

LSH注意力 LSH attention引入了查询和键之间的参数共享。它使用基于随机投影的散列函数将查询键散列到桶中。其关键思想是邻近的向量应该获得相似的哈希值，而距离远的向量则不应该，因此被称为"局部敏感"。为了执行哈希，首先引入一个随机矩阵$R \in \mathbb{R}^{k \times b/2}$。接下来，散列函数定义如下:

$$h(x) = \arg\max([xR; -xR]) \tag{3}$$

其中[;]是两个向量的拼接。对于所有查询，当且仅当查询和键散列匹配时才计算注意力，即$h(q_i) = h(k_j)$。换句话说，如果查询和键属于同一个哈希桶，则会计算它们的注意力。为了保持因果屏蔽，Reformer为每个查询和键分配并维护一个位置索引。因此，它能够比较每个查询键是否自回归有效。

基于LSH Attention的内存效率 LSH attention背后的关键思想是将令牌分类到桶中，然后以分块的方式逐桶处理它们。为此，查询首先按桶号排序，然后在同一桶内按顺序排序。在计算过程中，令牌只关注同一个桶本身的块和前一个块。分块和排序桶技术有助于提高重构模型的整体效率。

参数和存储复杂度 Reformer的内存复杂度是$\mathcal{O}(n \log n)$。在参数开销方面，Reformer实现了查询和键的共享，使QKV变换的开销降低了1 / 3。随机投影不是可训练参数，因此不会产生参数开销。总的来说，Reformer的参数比普通transformer少。Reformer中的可逆层还通过允许从下一层重建激活值来减少训练期间的内存消耗。这减少了内存成本，因为这消除了在反向传播期间存储所有层的激活值的需要。

3.2.11 Sinkhorn变压器

本节介绍稀疏Sinkhorn Transformer (Tay et al., 2020b)。Sinkhorn Transformer属于学习模式家族。该模型是一个分块/块模型，通过以块的方式重新排序输入键和值，然后应用基于局部块的注意力来学习稀疏模式。

$$A_{ij} = \begin{cases} (Q_i \psi_S(K)_j^\top), & \text{if} \lfloor j/N \rfloor = \lfloor i/N \rfloor \\ 0 & \text{otherwise} \end{cases}$$

其中ψ_S对序列长度维度应用排序操作符。

分拣网络 **排**序操作符由元排序网络参数化。设X为维度$N \times d$的输入序列。

$$\psi_S(X) = \phi_S(F_S(\textsc{BlockSum}(X)))\ \textsc{BlockShape}(X) \tag{4}$$

其中$F_S(.)$是一个参数化函数，如具有ReLU激活的两层前馈网络。$F_S(.)$的输出是$n_B \times n_B$的张量。BlockSum函数学习局部块的和嵌入。BlockShape函数将输入张量重塑为$\mathbb{R}^{N \times d} \rightarrow \mathbb{R}^{n_B \times b \times d}$。在这里，我们注意到$N = n_B \times b$，其中$b$是块的大小，$n_B$是总块的数量。

沉角分选 ϕ 是Sinkhorn平衡算子(Sinkhorn, 1964; Adams and Zemel, 2011)，它将$n_B \times n_B$矩阵转换为软置换矩阵。具体来说，在F_SBlockSum(X)的矩阵输出上应用了一系列的行和列归一化。为简洁起见，我们不会深入讨论该操作的细节。更多细节请访问Adams and Zemel (2011); Tay et al. (2020b)。

参数和存储复杂度 Sinkhorn Transformer的内存复杂度是$\mathcal{O}(b^2)$，其中b是块大小，$b = \frac{N}{N_b}$。额外的参数成本来自于元排序网络$F_S(.)$。因此，当使用两层ReLU网络作为排序网络时，附加参数的数量为$2d^2$。

3.2.12 Linformer

Linformer (Wang et al., 2020c)是一个基于低秩自注意力思想的高效Transformer。

长度维度上的低秩投影 Linformer使用额外的投影层将$N \times d$维度的键和值投影到$k \times d$维度。注意，这是对长度维度的缩减，而不是对键和值维度的缩减。这个罐子给定新投影的键(K')和值(V')，QK'矩阵现在是$(N \times k)$维度，而不是$(N \times N)$。注意力矩阵Softmax(QK')与$V' \in \mathbb{R}^{k \times d}$相乘，得到维度为$N \times d$的输出张量。在某种程度上，Linformer让人想起深度卷积(Kaiser et al., 2017)。在长度维度上的投影导致在单个变换中序列信息(维度方面)混合。因此，在计算注意力分数时，保持因果掩盖和/或防止混合过去和未来信息是很重要的。Linformer(对于每个注意头)的公式可以表示为:

$$Softmax(\frac{1}{\sqrt{d_k}} XW_i^Q (E_i XW_i^K)) \cdot F_i XW_i^V \tag{5}$$

其中$W^{Q,K,V}$是X到查询的默认线性转换(根据vanilla Transformer)，E_i, F_i是键和值到$k \times d$张量的额外$k \times N$投影。

参数和存储复杂度 Linformer的内存复杂度是$\mathcal{O}(n)$。由于额外的$N \times k$长度投影，Linformer的参数成本只有最小。如果k足够小，则所产生的参数成本可以忽略不计。

3.2.13 表演者

Performer (Choromanski et al., 2020a,b)模型的特点是其广义注意力机制和随机核的使用。

广义注意 **广**义的注意力将Q_i, K_j与一个核函数K纠缠在一起。Performer中的注意力矩阵通过以下方式计算:

$$A = [g(Q_i^\top) K(Q_i^\top K_j^\top) h(K_j^\top)] \tag{6}$$

其中$K(.)$是一个将$d \times d$映射到标量值$\mathbb{R}$的内核函数，g, h是将d映射到标量值$\mathbb{R}$的函数。

基于正交随机特征的快速注意力机制(FAVOR) 上述计算仍然是二次复杂度。因此，Performer利用近似技巧来避免存储和计算$N \times N$注意力矩阵。它利用正交随机特征(ORF)来做到这一点。Performer的最终注意力输出Y描述如下:

$$Y = \hat{D}^{-1}(Q'((K')^\top V)) \tag{7}$$

其中$\hat{D} = \mathrm{diag}(Q'((K')^\top 1_N))$、$Q' = D_Q\phi(Q^\top)^\top$和$K' = D_K\phi(K^\top)^\top$。请注意$D_Q = g(Q_i^\top), D_K = h(K_i^\top)$。函数$\phi(x)$定义为:

$$\phi(X) = \frac{c}{\sqrt{M}} f(Wx + b)^\top \tag{8}$$

其中$c > 0$是一个常数，$W \in \mathbb{R}^{M\times d}$是一个随机特征矩阵，$M$是这个矩阵的维数，它控制随机特征的数量。我们可以看到，我们没有明确计算$A = QK^\top$，因此避免了支付N^2成本。有兴趣的读者可以访问(Choromanski et al., 2020a)获取严谨的理论分析和更多细节。

参数/内存复杂度和计算成本 双向偏好算法的复杂性是$\mathcal{O}(Md + Nd + MN)$，其中$M$是随机特征的维度。值得注意的是，单向变化不能以有效的线性时间方式进行因果掩盖。因此，在训练过程中，在并行化训练期间，在自回归任务上运行基于核的注意力的单向(因果)实现可能比普通Transformer慢几倍，因为需要进行从左到右的传递(即扫描操作)，这与递归神经网络类似。由于许多自回归任务是通过并行化和教师强迫进行训练的，这使得生成任务上的训练Performer非常慢。为了使KV被有效地因果掩盖，人们必须在每个时间步显示$d \times d$ KV矩阵-恢复一个二次复杂度模型。我们认为这是一个错综复杂的问题，它突出了高效的内存复杂性在实践中可能并不等于更快或更有效的模型。本文强调，这只发生在自回归训练期间。然而，增量解码的推理时间将受益于速度的提高。

3.2.14 线性变压器

Linear Transformer (Katharopoulos et al., 2020)通过使用基于核的自注意力公式和矩阵乘积的结合性，将自注意力的复杂性从二次提高到线性。此外，它通过因果掩盖(用于自回归解码)将注意力降低到线性时间、恒定记忆的递归神经网络(RNN)。该模型已被证明可以将推理速度提高三个数量级，而不会损失太多的预测性能。除了核函数之外，线性transformer类似于表演者，因此也有相同的缺点(在自回归教师强制设置的训练期间，无法跨时间维度并行化)。

该方法基于一个简单但强大的观察，即位置i的查询Q_i的累积值V_i'可以写成:

$$V_i' = \frac{\sum_{j=1}^{p} \mathrm{sim}(Q_i, K_j)V_j}{\sum_{j=1}^{p} \mathrm{sim}(Q_i, K_j)}.$$

在这里，$p = N$是完整的，揭示了注意力和$p = i$在因果掩盖的情况下。现在，在平时的softmax注意，$\mathrm{sim}(q, k) = \exp\left(\frac{q^T k}{\sqrt{d}}\right)$。然而，Linear Transformer将相似度表示为核函数。也就是$\mathrm{sim}(q, k) := \phi(q)^T\phi(k)$，其中$\phi$是一个可能是高维的特征图。有了这个选择，我

们可以将V_i'重写为:

$$V_i' = \frac{\phi(Q_i)^T S_p}{\phi(Q_i)^T Z_p},$$
$$S_p := \sum_{j=1}^{p} \phi(K_j) V_j^T,$$
$$Z_p := \sum_{j=1}^{p} \phi(K_j).$$

为了揭示注意力，由于$p = N$我们只需要计算S_N和Z_N一次，并且我们在每个位置重用它们$0 \leq i \leq N$。对于因果注意，S_i 's和Z_i 's可以视为RNN的状态，通过以下递归关系进行更新:

$$S_i = S_{i-1} + \phi(K_i) V_i^T,$$
$$Z_i = Z_{i-1} + \phi(K_i)$$

具有初始条件$S_0 = Z_0 = 0$。如果键、查询和值的维度都是d，计算ϕ的成本是$\mathcal{O}(c)$，那么Linear Transformer的整体运行时复杂度是$\mathcal{O}(Ncd)$。作者选择

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

其中elu$(\cdot)$为指数线性单位(Clevert et al., 2015)。有了这种特征图的选择，$c = d$和模型的端到端复杂度是$\mathcal{O}(Nd^2)$。

3.2.15 合成器

合成器模型(Tay et al., 2020a)是研究和调查自注意力机制中条件作用的真正重要性的一次尝试，也是无条件token混合的第一次尝试。在Tay et al. (2020a)中，作者研究了一个合成的自注意力模块，其中注意力权重是近似的，而不是通过成对点积计算。合成器仅与高效transformer隐式相关，可以更多地视为mlp混合器(Tolstikhin et al., 2021)。然而，分解后的变体可以被认为是低秩高效的Transformer模型。

密集合成器 在密集合成器中，每个令牌x_i使用两层非线性前馈网络投影到长度为N的向量。注意力矩阵A的计算如下:

$$A = W_2(\sigma_R(W_1(X) + b)) + b \tag{9}$$

其中$X \in \mathbb{R}^{N \times d}$是输入序列，$W_2 \in \mathbb{R}^{d \times N}, W_1 \in \mathbb{R}^{d \times d}$, σ_R是ReLU激活函数。给定A，合成稠密函数的输出计算如下:

$$Y = \text{Softmax}(A)G(X). \tag{10}$$

其中$G(X)$是另一个参数化函数$\mathbb{R}^{N \times d} \to \mathbb{R}^{N \times d}$。

随机合成器 合成器模型的另一个变体为A使用随机矩阵。在这种情况下，输出可以表示为:

$$Y = \text{Softmax}(R)G(X). \tag{11}$$

其中$R \in \mathbb{R}^{N \times N}$是一个可训练和/或不可训练的矩阵。在Tay et al. (2020a)中，作者表明随机合成器取得了有竞争力的性能。

因式变异 密集和随机合成器还带有因子分解变体，考虑注意力矩阵的低秩结构。For分解随机合成器可以写成:

$$Y = \text{Softmax}(R_1 R_2^\top) G(X). \tag{12}$$

其中$R_1, R_2 \in \mathbb{R}^{N \times k}$。另一方面，稠密合成器可以被分解为:

$$A = H_B(B) * H_C(C) \text{ where } B, C = F_B(X_i), F_C(X_i), \tag{13}$$

其中$F_B(.)$项目到b维度，$F_C(.)$项目X_i到c维度$c \times b = N$。H_B, H_C分别是tile和repeat函数。

参数和存储复杂度 对于采用不可训练R的随机合成器，不需要在这一层存储N^2激活。对于可训练的随机合成器，存储复杂度和参数复杂度仍然是N^2。但是，不需要计算N^2点积，大大降低了计算成本。因子分解随机合成器将参数开销降低到$2(N \times k)$。

3.2.16 变压器- xl

Transformer-XL模型(Dai et al., 2019)依赖于基于分段的递归。基于段的递归可以被认为是与讨论的其他技术的正交方法，因为它没有显式稀疏化密集的自注意力矩阵。相反，它用循环机制连接相邻的块。

分段递归 Transformer-XL中的循环机制描述如下:

$$\tilde{\boldsymbol{h}}_{\tau+1}^{n-1} = [\text{SG}(\boldsymbol{h}_{\tau}^{n-1}) \odot \boldsymbol{h}_{\tau+1}^{n-1}] \tag{14}$$

$$q_{\tau+1}^n, k_{\tau+1}^n, v_{\tau+1}^n = \boldsymbol{h}_{\tau+1}^{n-1} \boldsymbol{W}_q^\top \, , \, \tilde{\boldsymbol{h}}_{\tau+1}^{n-1} \boldsymbol{W}_k^\top \, , \, \tilde{\boldsymbol{h}}_{\tau+1}^{n-1} \boldsymbol{W}_v^\top \tag{15}$$

$$\boldsymbol{h}_{\tau+1}^n = \text{Transformer}(q_{\tau+1}^n, k_{\tau+1}^n, v_{\tau+1}^n) \tag{16}$$

其中SG()是停止梯度函数，$\odot$是沿长度维度的两个序列的连接。值得注意的是，键和值是以前一个序列长度$\tilde{\boldsymbol{h}}_{\tau+1}^{n-1}$为条件的，而不是$\boldsymbol{h}_{\tau+1}^{n-1}$

相对位置编码 Transformer-XL引入了新的相对位置编码。在该方案中，绝对位置编码不添加到内容嵌入中。相反，它们只在计算注意力权重时被考虑，其中可以用相对位置编码替换。由于相对位置编码与模型的效率没有直接关系，我们请感兴趣的读者访问Dai et al. (2019)以了解更多细节。

3.2.17 压缩变压器

压缩transformer (Rae et al., 2020)是Transformer-XL模型的自然扩展。压缩Transformer背后的关键思想是维护过去段激活的细粒度记忆。这与Transformer-XL不同，后者在跨段移动时丢弃过去的激活。

模型存储器 压缩Transformer的特点是具有双模型存储系统-初级模型存储和次级压缩模型存储。它维护一个带有n_m内存插槽和n_{cm}压缩内存插槽的模型内存。每当模型接受新的输入段时，主模型内存中最古老的n_s激活将移动到应用压缩函数的压缩模型内存中。

压缩 这些记忆用各种压缩函数进行压缩，如(1)均值/最大池化(2)一维卷积，(3)扩张卷积，和(4)最常用的(例如，按注意力的使用情况排序)。

记忆重构 为了更好地保留长序列中的记忆，压缩Transformer实现了一种自编码损失，该损失学习从其压缩版本中重建原始记忆，即$L^{ae} = ||\text{old_mem} - g(\text{new_cm}^{(i)})||$，其中$g(.) : \mathbb{R}^{\frac{n_s}{c} \times d} \rightarrow \mathbb{R}^{n_s \times d}$是一个参数化函数。第二次注意力重建是一种有损重建，试图重建模型记忆上的注意力，而不是模型记忆本身的无损重建。

3.2.18 稀疏模型

在本节中，我们将介绍稀疏模型族。稀疏模型通常通过稀疏地激活一部分参数或激活来实现高参数到FLOP的比率。值得注意的是，虽然本综述范围内的大多数工作涉及高效的注意力，但稀疏模型的范围超出了注意力模块，通常更频繁地应用于前馈层(Lepikhin et al., 2020; Fedus et al., 2021)。在本节中，我们讨论稀疏模型的主要变体，即基于专家混合的稀疏模型，包括GShard (Lepikhin et al., 2020)、Switch Transformer (Fedus et al., 2021)和GLaM (Du et al., 2021)等模型。

专家混合 MoE背后的关键思想是将令牌x_i路由到由路由函数确定的一组选定的专家。路由函数通常使用softmax函数比专家计算线性组合，可以解释为一种门控机制。然后为每个标记选择top-k门值x_i，该层的最终输出由所选择的top-k专家的线性组合决定。除了某些实现细节之外，这个MoE层对于许多MoE体系结构来说仍然是基础的。例如，Switch使用top-1路由策略，而GShard使用分组级别的top-2门控。

4. 讨论

本节将介绍这类高效模型的研究现状。

4.1 论评价

虽然该领域充斥着新的Transformer模型，但没有一种简单的方法来并排比较这些模型。许多研究论文选择了自己的基准来展示所提出模型的能力。这还与不同的超参数设置(如模型大小和配置)相结合，这可能很难正确地归因性能提升的原因。此外，一些论文将这与预训练(Devlin et al., 2018)混为一谈，这使得区分这些不同模型的相对性能更加困难。人们应该考虑使用哪种基本的高效变压器块仍然是一个谜。

一方面，有多个模型专注于生成式建模，展示了所提出的Transformer单元在序列自回归建模方面的能力。为此，稀疏transformer (Child et al., 2019)，自适应transformer (Correia et al., 2019)，路由transformer (Roy et al., 2020)和改革者(Kitaev et al., 2020)主要专注于生成式建模任务。这些基准通常涉及在wikitext (Merity et al., 2017)和/或ImageNet (Deng et al., 2009) / CIFAR (Krizhevsky et al., 2009)等数据集上进行语言建模和/或逐像素图像生成。使用基于段的递归的模型，如Transformer-XL和Transformers，也专注于长程语言建模任务，如PG-19。

一方面，模型集合主要专注于纯编码任务，如问答、阅读理解和GLUE基准的选择。例如，ETC模型(Ainslie et al., 2020)只在问答基准上运行实验，如NaturalQuestions (Kwiatkowski et al., 2019)或TriviaQA (Joshi et al., 2017)。另一方面，Linformer (Wang et al., 2020c)专注于GLUE (Wang et al., 2018)基准的子集。这种划分是非常自然和直观的，因为像ETC和Linformer这样的模型不能以自回归的方式使用。这加剧了将这些纯编码器模型与其他模型进行比较的挑战。

有一些模型关注两者的平衡。Longformer (Beltagy et al., 2020)试图通过在生成建模和纯编码器任务上运行基准来平衡这一点。Sinkhorn Transformer (Tay et al., 2020b)在生成建模任务和仅编码任务上进行了比较。

此外，还值得注意的是，尽管Seq2Seq机器翻译(MT)是推广Transformer模型的问题之一，但这些高效的Transformer模型并没有很多在MT任务上进行评估。这可能是因为机器翻译中的序列长度不够长，不足以保证这些模型的使用。

虽然生成模型、胶水任务和/或问答似乎是许多这些任务采用的共同评估基准，但有几个小众基准是少数孤立的论文选择评估的。首先，Performer模型(Choromanski et al.,

2020a)评估了蛋白质上的掩码语言建模，偏离了与其他有效的Transformer模型的严重正面比较。Linear Transformer (Katharopoulos et al., 2020)还对语音识别进行了评估，这在这组论文中是一个罕见的基准。

最近试图统一对高效transformer的评估，即Long Range Arena，即LRA, (Tay et al., 2021a)，在远程建模任务中对10种不同的xformer变体进行基准测试。值得注意的是，LRA是为评估仅编码器模式下的transformer而设计的，不考虑需要因果掩盖的生成(或自回归任务)。

4.2 论模型设计趋势

当将广泛的分类与这些模型引入的时间线相匹配时，我们能够看到社区在设计高效Transformer模型方面的趋势。这一领域的早期工作主要集中在更直观和简单的方法，如固定模式。为此，这一领域的大多数早期工作是基于块/局部模式，如图像Transformer (Parmar et al., 2018)、压缩注意力(Liu et al., 2018)、分块Transformer (Qiu et al., 2019)或稀疏Transformer中的局部窗口(Child et al., 2019)。

分解各种固定模式的范式最早是在Child et al. (2019)和CCNet (Huang et al., 2019)中提出的。大约在同一时间，我们开始从Set Transformer (Lee et al., 2019)中的诱导点方法或Star Transformer (Guo et al., 2019a)模型中的全局节点中观察到基于模型内存的方法的早期痕迹。

我们观察到下一波模型以可学习的稀疏模式的形式出现。Reformer (Kitaev et al., 2020)和Routing Transformers (Roy et al., 2020)在某种意义上非常相似，它们是在执行注意力之前学习聚类/存储令牌的模型。关键的区别在于最终的方法，Reformer使用散列函数，而Routing Transformer使用在线k -方法进行集群分配。与此同时，Sinkhorn transformer (Tay et al., 2020b)也基于排序的思想，尽管是在块级别。这三个模型在很大程度上遵循了重新安排序列以有效计算注意力分数的类似范式。

接下来，观察几个主要基于稀疏Transformer范式的扩展。ETC (Ainslie et al., 2020)和Longformer (Beltagy et al., 2020)模型是非常相似的想法，从根本上来说是稀疏的Transformer扩展。这些模型包含了全局模型记忆(global model memory)的概念，这让人想起Set Transformer的诱导点方法或Star Transformer的全局模型记忆。Longformer的工作中也提出了对步幅的修改，例如使用膨胀窗口。

最近一波我们已经看到的模型是基于低秩近似或核方法的模型，例如低秩Transformer (Winata et al., 2020), Linformer (Wang et al., 2020c), Performer (Choromanski et al., 2020a)和/或线性Transformer (Katharopoulos et al., 2020)等模型。尽管由于评估状态和研究的高度并行性，目前还不清楚这种低秩或内核范式实际上是否比可学习模式(LP)或基于模型内存的高效Transformer模型更好。

最近，有更多的模型提出了一种双管齐下或两步注意力机制，结合了来自不同技术的模型。长短Transformer (Zhu et al., 2021)是一种结合固定模式注意力机制的Linformer动态形式。另一方面，像Poolingformer这样的模型也通过让人想起基于记忆的方法和局部注意力的技术显式地构建了一个两级注意力机制。Scatter Brain是一项新工作(Chen et al., 2021)试图将稀疏(固定模式)注意力与低秩注意力统一起来。Luna还提出了两种阶段注意力机制(Ma et al., 2021)

另一方面，重要的是要注意，基于递归的模型(Transformer-XL和Transformers)似乎是正交运行的，不能直接与其他模型相比。我们还观察到，稀疏模型(Lepikhin et al., 2020; Fedus et al., 2021)不仅适用于注意力模块，最近也正在出现并变得更受欢迎，并在最近几个月显示出相当大的成功(Du et al., 2021)。

4.3 浅谈正交效率的努力

本文主要关注(1)自注意力模块的计算和记忆复杂性，以及(2)稀疏性和自适应计算，简要总结了几种正交努力，这些努力也可能有助于Transformer模型的效率、可扩展性和整体可用性。

- **权重共享**——共享Transformer模型的参数将有助于减小总体模型大小。通用transformer (Dehghani et al., 2018)在各层之间连接注意力和过渡权重。类似地，Albert (Lan et al., 2019)跨层共享相同的参数。另一方面，四元数Transformer (Tay et al., 2019)提出了一种受Hamilton产品启发的权重共享方案，该方案局部共享线性变换层中的组件。

- **量化/混合精度**——学习混合精度模型有可能提高内存成本。Q-BERT (Shen et al., 2020)是一个将Transformer模型量化到超低精度的模型。同时，混合精度训练(Ott et al., 2019)是一种非常流行的技术，以减少训练transformer的内存成本。Fan et al. (2020)将量化感知训练应用于Transformer模型。

- **推理时间效率和网络剪枝**——多个研究方向探索在推理时提高Transformer的效率。一个典型的例子就是网络模型。一个例子是在推理过程中修剪注意力头(Voita et al., 2019; Michel et al., 2019)。这已经表明对下游任务的性能影响最小。另一方面，Lagunas et al. (2021)提出了一种"块"修剪方法，可以使Transformer更快2.4倍，而在语言任务的预测性能方面几乎没有损失。另一项工作涉及推理期间的快速退出，如果模型对其预测有信心，则允许我们退出计算(Schuster et al., 2021)。

- **知识蒸馏**- 知识蒸馏(KD) (Hinton et al., 2015)是一种有用的技术，可以将学到的知识从较大的教师模型迁移到较小的学生模型。较小的模型可以有效地部署到生产中。已经有许多提取大型Transformer模型的尝试。例如，DistilBERT (Sanh et al., 2019)，特定任务蒸馏(Tang et al., 2019)和TinyBERT (Jiao et al., 2019)。

- **神经架构搜索(NAS)** - 寻找更高效的Transformer架构也是一种常见策略。Guo et al. (2019b)提出的神经架构Transformer (NAT)，通过删除冗余操作，使用NAS来搜索更紧凑和有效的Transformer。Wang et al. (2020a)提出了HAT(硬件感知transformer)，一种利用NAS并使用硬件效率反馈作为奖励信号的方法。

- **任务适配器**——这一研究方向主要关注在T任务上微调大型Transformer的问题，并旨在跨各种任务重用参数。其关键思想是，任务适配器(Houlsby et al., 2019)支持跨任务重用参数，并在生产中重用服务T模型的需求——从而节省整体参数。已经提出了数量适中的模型，如PALS (Stickland and Murray, 2019), MAD-X (Pfeiffer et al., 2020)和HyperGrid (Tay et al., 2020c)。

- **可选架构**——**在设计Transformer替代**方案方面已经付出了相当大的努力。在考虑的许多替代方案中，一个突出的新兴研究路线属于MLP Mixers家族(Tolstikhin et al., 2021)。不同的混合操作已被提出，如G-MLP (Liu et al., 2021a), FNet (Lee-Thorp et al., 2021)。合成器(Tay et al., 2020a)，虽然通常被称为一种有效的注意力方法，但也是混频器工作的早期表现，因为随机矩阵同样作为一种无条件的混频器操作。最近基于结构化状态空间(Gu et al., 2021)的一项有希望的工作也在远程建模方面展示了非常有希望的结果。最后，卷积模型通常比transformer更有效，因为卷积核在输入标记周围的固定的小局部邻域上运行。Tay et al. (2021b)表明，当进行预训练时，这些更有效的卷积模型有时可以匹配Transformer模型的预测性能。

4.4 回顾过去的一年和未来的研究方向

随着我们对这项调查的V2版本的及时更新(2021年12月更新)，我们对该领域在过去一年左右的时间里如何发展提出了回顾性的想法。从上次更新开始，不可否认的是出现了更多的xformer变体，为普通transformer提供了更有效的替代方案。

值得注意的是，这些例子包括Nyströmformer (Xiong et al., 2021b), Perceiver (Jaegle et al., 2021), RFA (Peng et al., 2021), Luna (Ma et al., 2021)和Long Short Transformer (Zhu et al., 2021)。在本文发表时，也涌现出了其他值得注意的模型，并差一点被包含在第一版中(例如，漏斗Transformer (Dai et al., 2020))。在所有新的xformer变体中，值得注意的是，大多数变体都没有偏离第一个版本中提出的基本概念。我们的分类法和分类或多或少足够广泛，可以捕捉到许多这些模型，因为它们使用了现有工作中已经存在的基本思想，因此可以适当地分类。许多工作可以被认为是现有技术的显式组合(两阶段或两个方法类的组合)或对现有方法的改进(Linformer的低秩投影的动态表述或线性transformer的更好的核)。尽管许多现有的"内存"模型利用了一种形式的下采样来实现速度和效率的提高，但我们建立了一种新的"下采样"分类，以更好地反映这种新出现的趋势(Dai et al., 2020; Jaegle et al., 2021; Tay et al., 2021c; Ryoo et al., 2021)。

在过去的一年里，很明显，大量的研究投资被投入到使二次注意力在复杂性，有时是内存方面可扩展。在这个关头，最好考虑一下线性时间注意力的实际需求。即使在语言和视觉领域，许多应用程序仍然由具有二次注意力的vanilla transformer主导，这些*xformer*变体都没有成为事实上的标准。对于这一现象，可能有多种角度的解释。首先，线性注意力(例如Performer)模型很难在常见基准上具有竞争力，正如多个来源所指出的(Xiong et al., 2021a; Anonymous, 2021b)。

值得注意的是，除了简单的设置或特定的领域和问题外，它们直到最近才与常见的范式(如预训练和微调)进行过战斗测试。与此同时，基于固定和/或学习模式的局部注意力模型，如稀疏transformer (Child et al., 2019), Longformer (Beltagy et al., 2020)等(Ainslie et al., 2020)或BigBird (Zaheer et al., 2020)，已经看到了更合理的使用，特别是在长上下文问答领域。然而，在ETC (Ainslie et al., 2020)(通过拥有如此多不同的注意力方向，大大增加了代码复杂性)、Swin Transformer (Liu et al., 2021b)或Longformer (Beltagy et al., 2020)(需要自定义CUDA内核，从而使其在tpu等硬件上令人难以接受)等方法的高内在实现复杂性，可能是这些模型尚未发现自己是一个良好、易于使用的现成Transformer替代品的原因。

正如(Rabe and Staats, 2021)所指出的，对于需要调整序列长度和内存需要时间的应用程序，"只是顺序处理它"可能就足够了，即使这可能不像找到理论近似那样令人满意。与此同时，(Xiong et al., 2021a)表明，如果处理得当，本地注意力可能是一个非常难以超越的基准。

关于大量高效注意力模型的一个值得注意的事实是，"效率"一词超载。一个常见的误解是，高效的注意力模型总是意味着Transformer是快速的。事实是，由于创新限制，许多有效的注意力模型可能会使模型慢得多。此外，如果序列长度较短，许多线性注意力模型根本观察不到任何速度或记忆增益。它们中的许多都有非常痛苦的要求来实现因果掩盖(或TPU打包)(Choromanski et al., 2020b; Peng et al., 2021; Wang et al., 2020c)，并且经常必须在本质上权衡吞吐量和线性复杂度。另一方面，有些模型根本无法打包或因果掩码。关于这种效率误称的更多注释和讨论可以在本文(Dehghani et al., 2021)中找到，我们鼓励读者也仔细阅读。

这次更新还将基于注意力的高效xformer模型的原始范围扩展到稀疏模型，即使它们不一定针对注意力模块。我们相信，鉴于最近有希望的迹象，稀疏模型是本文范围的必要

补充(Fedus et al., 2021; Du et al., 2021; Zoph et al., 2022)。需要特别注意的是，在过去的一年(在正交方向部分)中，其他体系结构所做的工作。Mixer类型架构(Tolstikhin et al., 2021)在计算机视觉方面引起了一些兴趣，但似乎在语言(Anonymous, 2021a)上表现不佳。同时，基于结构化状态空间(如S4 (Gu et al., 2021))的替代模型解决了Long Range Arena基准(Tay et al., 2021a)中最难的Path-X任务。看到S4这样的模型在大规模和预训练条件下的表现应该是令人兴奋的。

随着这一年的结束，我们反思社区取得的惊人进展，我们开始思考高效变压器的未来以及理想的transformer模型应该是什么样子的。本文认为，理想的xformer应该有希望处理二次记忆问题，同时保持普适性(例如，在大多数任务中表现良好，而不仅仅是在长程任务中)。理想的xformer不应该在速度和内存之间进行权衡，也不应该牺牲tpu封装和/或因果屏蔽的能力。理想情况下，它应该是简单的，而不是使用严格的硬编码或过度的工程，即，它应该是优雅的和可扩展性好。理想情况下，效率将直接考虑到下一代transformer，而不是总是有一个可以用于长上下文任务的副变体。虽然我们不能明确指出xformer的任何变种是解决transformer效率问题的决定性变种，但考虑到进步的速度，我们乐观地认为*the*真正的xformer最终会出现。接下来的问题是，新的xformer是否仍然是Transformer。

5. 结论

本文综述了关于高效Transformer模型的文献，特别是与自注意力模块的二次复杂度有关的文献。我们提供了这些新模型中采用的核心技术的分类法和高级抽象。对现有的基于技术的模型进行了特征描述，并对几个高效的Transformer模型进行了全面的介绍。最后，讨论了这些模型的评价概况和设计趋势。最后，我们简要讨论了其他可能提高Transformer模型效率的并行正交工作。**注:**本调查可能每两年或每年修订一次。请随时将反馈发送到我们的电子邮件地址。虽然我们可能不会回复所有邮件，但我们肯定会阅读它们。我们也欢迎匿名反馈到`https://forms.gle/kqjmhSDEQrmL4Egk6`。

Acknowledgments

The authors would like to send the numerous authors who send us feedback via email. We tried our best to incorporate most of the suggestions as we sat fit. We also thank Tamas Sarlos for feedback on this manuscript.

英寸

References

Ryan Prescott Adams and Richard S Zemel. Ranking via sinkhorn propagation. *arXiv preprint arXiv:1106.1925*, 2011.

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

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Philip Pham, Anirudh Ravula, and Sumit Sanghai. Etc: Encoding long and structured data in transformers. *Proceedings of EMNLP*, 2020.

Anonymous. Remixers: A mixer-transformer architecture with compositional operators for natural language understanding. *ACL RR 2021 September Submission*, 2021a.

Anonymous. Scaling laws vs model architectures: How does inductive bias influence scaling? an extensive empirical study on language tasks. *ACL Rolling Review, September*, 2021b.

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

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *Proceedings of EMNLP*, 2020.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020.

Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Jared Davis, Tamas Sarlos, David Belanger, Lucy Colwell, and Adrian Weller. Masked language modeling for proteins via linearly scalable long-context transformers. *Proceedings of ICLR*, 2020a.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *Proceedings of ICLR*, 2020b.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *Proceedings of ICLR 2016*, 2015.

Gonçalo M Correia, Vlad Niculae, and André FT Martins. Adaptively sparse transformers. *Proceedings of EMNLP*, 2019.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *Proceedings of ACL*, 2019.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Proceedings of NeurIPS*, 2020.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *Proceedings of ICLR*, 2018.

Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. The efficiency misnomer. *arXiv preprint arXiv:2110.12894*, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL*, 2018.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathy Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. Glam: Efficient scaling of language models with mixture-of-experts, 2021.

Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Remi Gribonval, Herve Jegou, and Armand Joulin. Training with quantization noise for extreme fixed-point compression. *arXiv preprint arXiv:2004.07320*, 2020.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *Proceedings of NeurIPS*, 2021.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. Star-transformer. *Proceedings of NAACL*, 2019a.

Yong Guo, Yin Zheng, Mingkui Tan, Qi Chen, Jian Chen, Peilin Zhao, and Junzhou Huang. Nat: Neural architecture transformer for accurate and compact architectures. In *Advances in Neural Information Processing Systems*, pages 737–748, 2019b.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. *Proceedings of ICML*, 2019.

Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 603–612, 2019.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Łukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. Depthwise separable convolutions for neural machine translation. *Proceedings of ICLR*, 2017.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. *arXiv preprint arXiv:2006.16236*, 2020.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rkgNKkHtvB`.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit,

Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M Rush. Block pruning for faster transformers. *Proceedings of EMNLP 2021*, 2021.

Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Large memory layers with product keys. *arXiv preprint arXiv:1907.05242*, 2019.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *Proceedings of ICLR*, 2019.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753, 2019.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*, 2021.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. *arXiv preprint arXiv:2103.16716*, 2021.

Hanxiao Liu, Zihang Dai, David R So, and Quoc V Le. Pay attention to mlps. *Proceedings of NeurIPS*, 2021a.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *Proceedings of ICLR*, 2018.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021b.

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. Luna: Linear unified nested attention. In *Proceedings of NeurIPS 2021*, 2021.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *Proceedings of ICLR*, 2017.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Proceedings of NeurIPS*, 2019.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*, 2019.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *Proceedings of ICML 2018*, 2018.

Niki Parmar, Prajit Ramachandran, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *Advances in Neural Information Processing Systems*, pages 68–80, 2019.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. *Proceedings of ICLR*, 2021.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. *Proceedings of EMNLP*, 2020.

Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972*, 2019.

Markus N Rabe and Charles Staats. Self-attention does not need o (n^ 2) memory. *arXiv preprint arXiv:2112.05682*, 2021.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SylKikSYDH`.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research, 2020*, 2019.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. Hash layers for large sparse models. *arXiv preprint arXiv:2106.04426*, 2021.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Proceedings of TACL*, 2020.

Michael S. Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Tal Schuster, Adam Fisch, Tommi Jaakkola, and Regina Barzilay. Consistent accelerated inference via confident adaptive transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4962–4979, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.emnlp-main.406.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. 2020.

Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879, 1964.

David R So, Chen Liang, and Quoc V Le. The evolved transformer. *Proceedings of ICML*, 2019.

Asa Cooper Stickland and Iain Murray. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. *Proceedings of ICML*, 2019.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. *arXiv preprint arXiv:1905.07799*, 2019a.

Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*, 2019b.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*, 2019.

Yi Tay, Aston Zhang, Luu Anh Tuan, Jinfeng Rao, Shuai Zhang, Shuohang Wang, Jie Fu, and Siu Cheung Hui. Lightweight and efficient neural natural language processing with quaternion networks. *Proceedings of ACL*, 2019.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention in transformer models. *Proceedings of ICML, 2021*, 2020a.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. *Proceedings of ICML*, 2020b.

Yi Tay, Zhe Zhao, Dara Bahri, Donald Metzler, and Da-Cheng Juan. Hypergrid: Efficient multi-task transformers with grid-wise decomposable hyper projections. *Proceedings of ICLR*, 2020c.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *Proceedings of ICLR*, 2021a.

Yi Tay, Mostafa Dehghani, Jai Gupta, Dara Bahri, Vamsi Aribandi, Zhen Qin, and Donald Metzler. Are pre-trained convolutions better than pre-trained transformers? *arXiv preprint arXiv:2105.03322*, 2021b.

Yi Tay, Vinh Q Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. Charformer: Fast character transformers via gradient-based subword tokenization. *arXiv preprint arXiv:2106.12672*, 2021c.

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Peter Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580. URL `https://aclanthology.org/P19-1580`.

Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. *Proceedings of NeurIPS*, 2020.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL `https://aclanthology.org/W18-5446`.

Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. Hat: Hardware-aware transformers for efficient natural language processing. *arXiv preprint arXiv:2005.14187*, 2020a.

Shuohang Wang, Luowei Zhou, Zhe Gan, Yen-Chun Chen, Yuwei Fang, Siqi Sun, Yu Cheng, and Jingjing Liu. Cluster-former: Clustering-based sparse transformer for long-range dependency encoding. *Proceedings of ACL-IJCNLP (Findings)*, 2020b.

Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020c.

Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. *Proceedings of ICLR*, 2019.

Genta Indra Winata, Samuel Cahyawijaya, Zhaojiang Lin, Zihan Liu, and Pascale Fung. Lightweight and efficient end-to-end speech recognition using low-rank transformer. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6144–6148. IEEE, 2020.

Wenhan Xiong, Barlas Oğuz, Anchit Gupta, Xilun Chen, Diana Liskovich, Omer Levy, Wen-tau Yih, and Yashar Mehdad. Simple local attentions remain competitive for long-context tasks. *arXiv preprint arXiv:2112.07210*, 2021a.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nystr\" omformer: A nystr\" om-based algorithm for approximating self-attention. *Proceedings of AAAI*, 2021b.

Chulhee Yun, Yin-Wen Chang, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. $o(n)$ connections are expressive enough: Universal approximability of sparse transformers. *Proceedings of NeurIPS*, 2020.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Proceedings of NeurIPS*, 2020.

Hang Zhang, Yeyun Gong, Yelong Shen, Weisheng Li, Jiancheng Lv, Nan Duan, and Weizhu Chen. Poolingformer: Long document modeling with pooling attention. *Proceedings of ICML*, 2021.

Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. Long-short transformer: Efficient transformers for language and vision. *Advances in Neural Information Processing Systems*, 34, 2021.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. Designing effective sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.