A Survey of Transformer Structures and Related Optimization Techniques

李宇哲 PB21111653

摘要—Transformer[12] 架构自 2017 年提出以来，凭借其基于自注意力机制的特性，彻底革新了序列到序列的学习方法。在自然语言处理、计算机视觉等领域，Transformer 逐渐成为主流模型。

Transformer 通过多头自注意力和位置编码来捕捉序列中的依赖关系，其独特的编码器-解码器结构有效提升了模型的并行处理能力。

本文从 Transformer 模型结构，改进及其变体，优化方法三方面进行调研。首先概述了 Transformer 的结构及其关键组件，包括自注意力机制、前馈神经网络、位置编码、层归一化和残差连接。随后，介绍了一系列 Transformer 模型的变体及在计算机视觉，自然语言处理等领域的不同应用，如视觉领域的 Vit[6]，ViViT[1]，NLP 领域的 BERT[5]、GPT[2] 等。最后，本文介绍了 Transformer 的一系列优化方法，包括模型结构本身的优化，训练加速技术，硬件和内存上的优化。

***Index Terms*—Transformer, BERT, GPT, ViT, ViViT, FlashAttention, Memory Hierarchy, Parallelism**

I. Introduction

自 2017 年 Vaswani 等人提出 Transformer 架构以来，该模型凭借其独特的自注意力机制和高效的并行处理能力，在自然语言处理领域（NLP）和计算机视觉等领域取得了广泛的应用和成功。传统的序列到序列（seq2seq）模型如循环神经网络（RNN[7]）、LSTM[9] 等在处理长序列时往往面临信息遗失和计算效率低的问题，而 Transformer 通过多头注意力机制和位置编码，有效地解决了这一问题，并成为现代深度学习模型的基础。

随着 Transformer 模型的成功，研究者们提出了多种变体和优化策略，以适应不同的任务需求和计算资源限制。比如，在计算机视觉领域，Vit 和 ViViT 等模型扩展了 Transformer 的应用范围；在 NLP 领域，BERT 和 GPT 等预训练模型展示了强大的语言理解和生成能力。此外，这篇文献调研还集中于提高 Transformer 的计算效率和模型质量，如通过稀疏化等手段提高训练效率；通过 flash attention[4] 利用好现在主流的训练硬件 GPU 硬件特征来加速训练，获得更好的模型质量；通过压缩等技术，加速模型训练。

本篇调研旨在系统性的了解 Transformer 模型的基础结构、其不同领域的变体和常用的优化方法，聚焦于其模型的基础结构和 Flash Attention 这种在当今 Sora 模型中常用的优化技术。与此同时，本文也会讨论 Tranformer 的不同变体的创新之处和使用场景，最后总结了在模型性能和效率方面的优化策略和技术展望。

II. Transformer Structure

在本部分中，我们将深入探讨 Transformer 模型的关键结构组成部分。主要包括编码器-解码器结构、注意力机制、前馈神经网络（FFN）、嵌入与 softmax，以及能表示位置信息的位置编码。

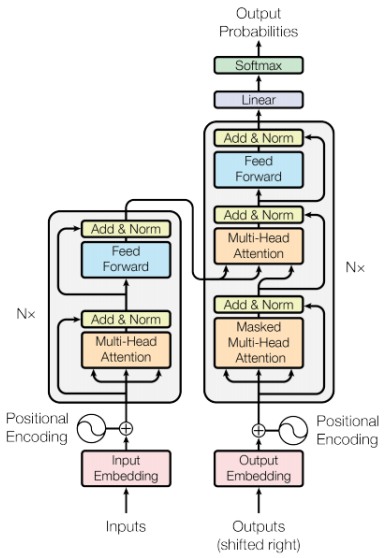

图 1. Transformer Model Structure

A. Encoder-Decoder Structure

Transformer 模型的编码器-解码器结构结构主要由两个部分组成:编码器(Encoder)和解码器(Decoder)。每个部分由多个相同的层堆叠而成，这种允许模型通过堆叠深度，从而提升表达能力。

编码器（Decoder）的任务是将输入序列 $X = (x_1, x_2, ..., x_n)$ 转换成一个中间表示，用于捕获输入序列的语义信息。在 Transformer 结构中采用了 6 层堆叠的形式，每一层都是两个子曾构成，包括一个多头注意力机制，前馈神经网络，并采用残差连接的方式 [8]。

解码器（Encoder）也是由 6 层相同的 layer 构成，包括一个 Masked 的多头注意力机制和在其后的一个多头注意力机制，经过前馈神经网络层后，通过 softmax 输出概率。

这种 Encoder-Decoder 的架构，Encoder 将会生成一个上下文向量便于后续 Decoder 阶段使用，而 transformer 中使用自注意力机制，使得输入序列中的每个元素都能与其他元素进行交互，而不仅仅依赖于固定的上下文向量，来捕获更长程的以来，并且便于并行化。

B. Attention Mechanism

注意力机制是 Transformer 的核心部分，用于捕捉序列中的依赖关系。通过动态地对输入序列的不同部分进行加权，模型能够在生成输出时选择性地关注最相关的信息。下面详细介绍其关键组成部分。

1) Scaled Dot-Product Attention: 缩放点积注意力（Scaled Dot-Product Attention）是计算注意力权重的基本方法。给定输入的查询（Query）、键（Key）和值（Value）向量，缩放点积注意力的计算过程如下：

1. 计算点积：对于每个查询 Q 和键 K，计算它们的点积，以获得注意力分数：

$$score(Q, K) = QK^T$$

2. 缩放：为了提高数值稳定性，点积结果会被缩放。具体地，将分数除以 $\sqrt{d_k}$，其中 d_k 是键向量的维度：

$$scaled\_score(Q, K) = \frac{QK^T}{\sqrt{d_k}}$$

3. 应用 Softmax：将缩放后的分数输入到 softmax 函数，以获得注意力权重：

$$\alpha = softmax(scaled\_score(Q, K))$$

4. 加权求和：最后，使用注意力权重对值向量 V 进行加权求和，生成输出：

$$Attention(Q, K, V) = \alpha V$$

这种方法能够有效捕捉输入序列中不同元素之间的关系，且通过缩放操作避免了在点积计算中可能出现的数值溢出问题。

2) Multi-Head Attention: 多头注意力机制通过多个注意力头实现，允许模型在不同的表示空间中学习更丰富的关系信息。具体过程如下：

1. 线性变换：首先，将输入的查询、键和值分别通过线性变换映射到多个不同的子空间。这是通过定义多个权重矩阵来实现的：

$$Q_i = XW_i^Q, \quad K_i = XW_i^K, \quad V_i = XW_i^V \quad (i = 1, \ldots, h)$$

其中 h 是注意力头的数量，W_i^Q, W_i^K, W_i^V 是相应的权重矩阵。

2. 应用缩放点积注意力：对每个注意力头 i 使用缩放点积注意力进行计算，得到每个头的输出：

$$head_i = Attention(Q_i, K_i, V_i)$$

3. 拼接头的输出：将所有注意力头的输出拼接起来，形成一个综合的表示：

$$concat\_heads = Concat(head_1, \ldots, head_h)$$

4. 线性变换：最后，拼接后的输出会通过另一个线性层进行映射，得到最终的输出：

$$MultiHead(Q, K, V) = concat\_headsW^O$$

其中 W^O 是输出的权重矩阵。

多头注意力机制使得模型可以并行地在不同的子空间中学习信息，这种机制提高了模型的表达能力，允许 Transformer 更加灵活地捕捉复杂的依赖关系。此外，多个注意力头的使用也有助于提升模型的泛化能力，从而在不同的任务中表现出色。从这里也可以看到，计算注意力机制的核心是 Q、K、V 三个矩阵，并涉及到矩阵乘和 reduction，element-wise 等不同的操作类型，如何对这些计算进行优化也是考虑如何加速 Transformer 计算的重中之重。

C. Feed-Forward Networks (FFN)

前馈神经网络（FFN）是每个编码器和解码器层中的关键组件，由两个线性变换和一个激活函数组成，用于提高模型的非线性表达能力。

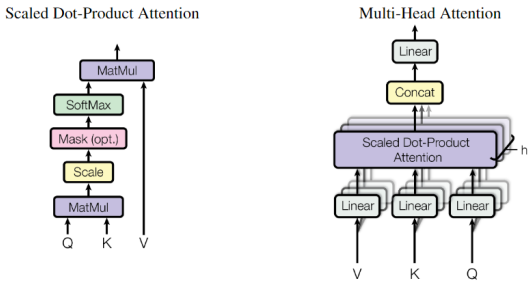

图 2. Attention Mechanism

前馈神经网络在我的理解下与 MLP[10] 本质上并没有什么区别，都是用于从输入数据提取有用的特征学习一些映射。

D. Embedding and Softmax

嵌入（Embedding）层是自然语言处理（NLP）任务中不可或缺的一部分，它将词汇表中的每个单词映射为一个固定维度的向量。这些向量通常是通过训练获得的，能够捕捉单词之间的语义关系。例如，相似意思的单词在嵌入空间中往往会靠得很近。这种映射不仅减少了输入数据的维度，还有效保留了单词的语义信息。

在实际操作中，嵌入层的输出会作为后续模型（如 Transformer）的输入，形成每个单词的上下文表示。随着模型的训练，嵌入向量会不断优化，使得语义相似的词在向量空间中更加接近，从而提高模型的性能。

而 **softmax** 函数则用于将模型的输出转化为概率分布，特别是在生成输出词汇时。具体地，**softmax** 接受模型最后一层的输出（通常是未归一化的对数几率），并将其转化为一个在 0 到 1 之间的概率值，所有概率的总和为 1。这一过程使得模型能够根据生成的概率分布选择最有可能的输出单词，从而实现序列生成、分类等任务。

softmax 在具体的诸如 pytorch 等深度学学习训练框架中，其实现用到了规约类算子，和 element-wise 的 exp 等操作，这些操作都是内存密集型的操作，如何加速这些操作也成为了如 Flash Attention 等优化方法提出的依据。

E. Positional Encoding

在 Transformer 架构中，由于模型结构的特殊性（即不使用循环神经网络或卷积神经网络），它缺乏处理输入序列中单词顺序的固有能力。为了解决这个问题，引入了 **位置编码（Positional Encoding）**，用于补偿 Transformer 缺乏序列位置信息的不足。

位置编码通过将位置信息以向量的形式添加到输入嵌入中，使模型能够理解序列的相对顺序。这些位置编码向量通常是使用正弦和余弦函数生成的，公式如下：

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

其中，pos 是单词在序列中的位置，i 是维度索引，d_{model} 是嵌入向量的维度。这种设计使得模型能够利用位置编码在不同的层之间有效捕捉位置信息。

III. Transformer in NLP

Transformer 架构自提出以来，已成为自然语言处理领域的核心模型之一。GPT 和 BERT 是两类自然语言处理领域，预训练阶段使用 Transformer 架构的典例。BERT 是一种预训练的双向 Transformer 编码器模型，而 GPT 是预训练的单向 Transformer 解码器模型，前者侧重于理解任务，使用双向的上下文结构，而后者侧重于生成任务，使用单向的上下文结构。

预训练（Pre-training）是一种深度学习的训练策略。在特定任务之前，首先使用大量无标签的数据训练模型（无监督学习），使其提取到有用的特征和模型。经过预训练的模型可以在目标任务上进行微调（fine-tuning），以提高性能。

A. BERT 与 GPT 的异同

BERT（Bidirectional Encoder Representations from Transformers）和 GPT（Generative Pre-trained Transformer）都是基于 Transformer 架构的重要模型，但在设计理念和训练任务上存在显著差异。

1) BERT: BERT 的主要创新在于其深度双向编码的能力。通过使用掩蔽语言模型（Masked Language Model, MLM）作为预训练任务，BERT 在训练时随机掩蔽输入序列中的一些单词，并训练模型预测这些掩蔽的单词。这种方式使得模型能够同时考虑上下文中左右两侧的信息，从而有效捕捉词义的多重上下文。BERT 还引入了下一句预测（Next Sentence Prediction, NSP）任务，以增强模型对句子之间关系的理解。

BERT 的基本架构是基于 Transformer 编码器堆叠而成，模型的输入由词嵌入、位置编码和段编码组成，使得模型能够处理句子对的任务（如问答和文本分类）。

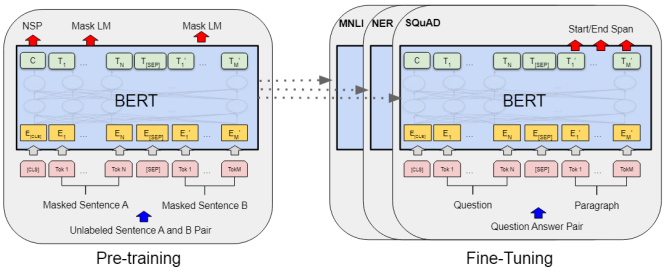

图 3. BERT

2) GPT: 与 BERT 相对，GPT 采用了单向自回归的策略。它通过预测序列中下一个单词来进行训练，使用的是前向的自注意力机制，即在计算注意力时仅考虑当前单词之前的单词。这种设计使得 GPT 在生成任务（如文本生成和对话系统）中表现出色，因为它能够根据已生成的内容逐步生成下一个词。

GPT 的基本架构是基于 Transformer 解码器堆叠而成，输入由词嵌入和位置编码组成。在训练时，GPT 通过大规模的文本数据进行预训练，学习到语言的生成模式。

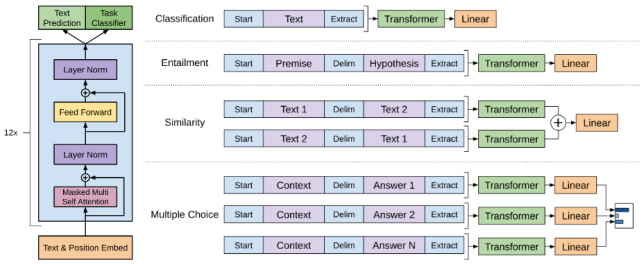

图 4. GPT

B. BERT 与 GPT 的比较

1. **结构**：- **BERT** 使用 Transformer 编码器，支持双向上下文建模。- **GPT** 使用 Transformer 解码器，基于单向自回归生成。

2. **训练目标**：- **BERT** 的训练任务包括 MLM 和 NSP，强调对上下文的全面理解。- **GPT** 的训练任务是通过下一个词预测（next token prediction），强调生成能力。

3. **应用场景**：- **BERT** 适合文本分类、问答系统等任务，需要对上下文进行深度理解。- **GPT** 更适合文本生成、对话系统等任务，能够逐步生成连贯的文本。

在如今 ChatGPT, LLaMA[11] 这种单向自回归的生成式语言模型成为自然语言处理领域训练的主流模型，其基础结构 Transformer 是我认为需要深入掌握的。

IV. Transformer in Computer Vision

Transformer 架构自提出以来，逐渐在计算机视觉（Computer Vision）领域获得广泛应用，特别是在图像分类、目标检测、图像分割等任务中展现出强大的性能.ViT 和 ViViT 是我最近在做文献调研时比较感兴趣的两篇论文，这个 part 我将主要专注于这两篇工作的调研与介绍。

A. ViT

Dosovitskiy et al. (2020) 提出了 **Vision Transformer (ViT)** 模型。该论文的核心思想是将图像视为一个序列，通过将图像分割成小块（patches）并将每个块视作一个输入的“单词”，从而将 Transformer 应用于图像识别任务。

ViT 的主要创新在于以下几点：1. **图像分块**：将输入图像划分为固定大小的小块（例如 16×16 像素），并将这些块展平以形成输入序列。

2. **位置编码**：为了保留图像中块的空间信息，ViT 在输入序列中添加位置编码，确保模型能够理解各块之间的相对位置。

3. **Transformer** 架构：通过应用标准的 Transformer 编码器结构，ViT 能够有效学习图像中的全局上下文信息，而不是局限于局部特征。

实验结果表明，ViT 在大规模数据集（如 ImageNet）上实现了与传统卷积神经网络（CNN）的竞争性能，展示了 Transformer 在视觉任务中的潜力。将输入图像处理成类似长文本序列的 idea 也打开了 Transformer 在计算机视觉领域的大门。

B. ViViT

Arnab et al. (2021) 提出了 **ViViT**，这是一种扩展的 Transformer 模型，旨在处理视频数据。ViViT 结合了 Transformer 的架构和视频序列的特性，以有效捕捉时空信息。

ViViT 的主要创新在于：1. **时空建模**：ViViT 通过将视频切分为时空块（spatio-temporal patches），处理帧序列。每个时空块包含多个时间步和空间区域，从而更好地捕捉视频中的动态变化。

2. **多种处理策略**：ViViT 提出了不同的处理方法，包括将视频帧作为单独的输入序列，或者使用交替的空间和时间注意力机制，以提高模型对视频内容的理解。

3. **适应性与扩展性**：该模型可以灵活调整，以适应不同分辨率和时长的视频输入，并在视频分类、行为识别等任务中表现出色。

实验表明，ViViT 在多个视频理解基准测试中优于传统的 CNN 方法，展示了 Transformer 架构在处理动态视频数据时的有效性。

C. 结论

Transformer 架构通过引入自注意力机制和位置编码，极大地提升了处理序列数据的能力。ViT 和 ViViT 作为代表性模型，各自通过不同的设计理念和训练目标，展示了 Transformer 在处理图像数据时的优势和灵活性。这些研究不仅推动了计算机视觉的发展，还为未来的模型设计提供了新的思路和方法。

V. Optimizations

随着 Transformer 架构在自然语言处理和计算机视觉领域的广泛应用，研究者们不断探索优化方法，以提高模型的训练和推理效率。以下是关于几个重要优化技术的总结，主要涵盖了模型结构本身的优化，比如使用稀疏注意力 [3]，开发更高效的注意力机制，通过分层等方法降低 attention 计算的时间复杂度和内存占用 [13]; 训练上的加速和针对 DNN 训练的硬件设备进行优化，比如 Flash Attention。

A. Problems in Attention Mechianism

Vaswani et al. (2017) 提出了 Transformer 模型，开创了自注意力机制在序列任务中的应用。尽管该模型展示了显著的性能提升，但自注意力机制的计算复杂度为 $O(n^2)$，其中 n 是输入序列的长度。这使得在处理长序列时，计算和内存开销迅速增加，限制了模型的可扩展性。

B. Sparse Attention

针对自注意力的计算复杂度问题，Sparse Attention 技术应运而生。Sparse Attention 旨在通过减少注意力计算的非零权重来降低复杂度，使得模型在处理长序列时更加高效。其主要思想包括以下几个方面：

1. **局部注意力**：在计算注意力时，仅关注输入序列中的一部分，例如邻近的几个元素。这种方法显著减少了每次计算所需的操作，保持了对局部上下文的关注。

2. **全局稀疏模式**：通过设计特定的稀疏模式，使得某些关键的输入元素能够在计算注意力时保持关注，而其他不太重要的元素被忽略。这种方式可以保证模型在保留重要信息的同时，降低计算复杂度。

3. **基于注意力稀疏化的机制**：一些方法使用数据驱动的方式自动学习稀疏化策略，例如通过训练期间动态调整注意力权重，使得模型在优化过程中自然地学习到最有效的注意力模式。

例如，论文《Long-Short Range Attention for Transformer》中提出了一种改进的稀疏注意力机制，结合了长距离和短距离的注意力计算，以应对长序列中的依赖关系。

C. Efficient Attention with Linear Complexity

多项工作致力于开发高效的注意力机制，以降低其时间复杂度和内存占用。例如，Linformer 通过低秩近似将注意力计算简化为线性复杂度 $O(n)$，而 Performer 利用核方法实现了类似效果。这些方法通过避免全局自注意力计算，转而采用局部注意力或近似算法，有效提高了在长序列上的处理能力。

这些研究不仅减小了内存占用，还加速了训练过程，使得 Transformer 模型能够在更大规模的数据集上进行训练，拓宽了其应用范围。这种高效注意力机制的引入，使得 Transformer 更加灵活地适应多种任务。

D. FlashAttention

在最新的研究中，FlashAttention 提供了一种新颖的自注意力实现，专门针对传统注意力机制在内存和计算效率上的瓶颈进行优化。FlashAttention 的设计理念包括以下几个关键技术：

1. **分块计算**：FlashAttention 将输入序列划分为多个小块，以减少内存占用并提高计算效率。通过在 GPU 上利用共享内存，这种分块方法显著提高了缓存的利用率，减少了数据传输延迟。

2. **避免重复计算**：FlashAttention 在计算过程中有效重用中间结果，避免了重复执行相同操作。这种方法不仅加快了计算速度，还降低了功耗，从而提高了整体效率。

3. **高效的内存管理**：该方法采用了优化的内存布局和数据流策略，以降低内存带宽的需求。通过减少内存

访问次数，FlashAttention 能够在处理长序列时保持较低的延迟和高吞吐量。

4. **支持动态序列长度**：FlashAttention 设计得足够灵活，能够处理可变长度的输入序列，这在实际应用中尤为重要，因为许多数据集中的输入序列长度不一致。

实验结果表明，FlashAttention 在多个标准基准测试中，尤其是在长序列处理时，显著优于传统的自注意力实现，提供了更高的性能和更低的内存使用。

E. 总结

通过采用 Sparse Attention , Flash Attention 等优化技术，研究者们显著提高了 Transformer 模型在长序列处理中的效率。这些优化方法不仅降低了内存占用和计算开销，还使得 Transformer 更加灵活地适应各种应用场景。随着对注意力机制优化的深入研究，新的稀疏化方法和策略将不断涌现，推动自然语言处理和计算机视觉等领域的发展。

尤其是 Flash Attention 这种方法，利用到了 gpu 的分层结构，将 memory-bound 的计算通过 gpu 的分层结构加速，通过 tiling 等手段，充分利用好 gpu 的硬件特点。不过这种方式如果要移植到其他的 DSA 结构注入 NPU 等，还需要根据不同的硬件特点，设计不同的 tiling 模型等。这种充分利用到硬件特点的优化方法无疑是 mlsys 领域的佳作。

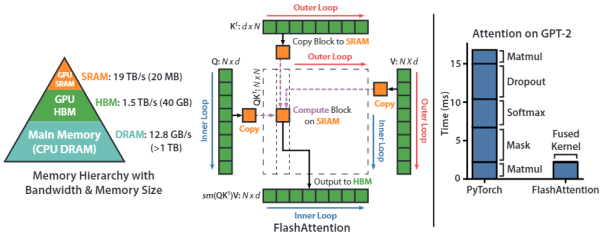

图 5. Flash Attention

VI. Conclusions

在本综述中，我们探讨了 Transformer 模型的基本结构及其在自然语言处理（NLP）和计算机视觉（CV）领域的应用。首先，我们介绍了 Transformer 的核心组件，如自注意力机制和多头注意力，这些创新使其在处理序列数据时展现出强大的性能。在 NLP 领域，我们详细分析了 BERT 和 GPT 两个模型的主要思想及其在文本理解和生成任务中的优势。BERT 通过深度双向编码捕捉上下文信息，而 GPT 采用自回归生成的方式，强调了其在文本生成中的能力。

在 CV 领域，ViT 和 ViViT 的提出标志着 Transformer 架构在视觉任务中的成功应用。ViT 将图像视为序列处理，展示了 Transformer 在图像识别任务中的潜力，而 ViViT 则通过时空建模进一步扩展了 Transformer 对视频数据的处理能力。

此外，我们还讨论了一些关键的优化技术，如 Flash Attention、Sparse Attention 和 Efficient Attention with Linear Complexity。这些方法通过降低自注意力计算的复杂性和内存占用，显著提高了 Transformer 模型的效率，使其更适用于长序列处理和大规模数据集。

综上所述，Transformer 模型及其变种在各个领域取得了显著进展，展现了强大的灵活性和适应性。随着研究的不断深入，未来将会有更多创新的方法和应用不断涌现，推动深度学习领域的发展。

参考文献

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer, 2021.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[3] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019.

[4] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[7] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[10] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[11] Hugo Touvron, Julien Vial, Ara Yeretsian, Andreas Dufter, David Dohan, Benoit Rousillon, Henry Tcheurekdjian, Michael Tschannen, David Lacroix, Jean-Philippe Goeuriot, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[12] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[13] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020.