FRTI64 · AI LEARNING PATH

核心论文 · 中英对照全译

Core Papers, Paragraph-by-Paragraph Bilingual Translation

4 篇奠基论文 · 97 个对照段落 · 本地大模型翻译

收录论文 / Papers	P001、P004、P002、P003
正文档段 / Body units	97 段（另含参考文献原文 75 段）
翻译模型 / Model	qwen3.8:27b（本地 Ollama）
对照方式 / Layout	英文原文（灰底） + 中文译文；参考文献保留英文原文
整理时间 / Compiled	2026-11-22
网页检索版 / Web index	https://frti64.top/ai-resources/

说明：英文栏为论文原文段落（由 PDF 文本层提取，公式/符号保持原样）；中文栏为本地 27B 模型逐段翻译，专有名词保留原文，仅供学习参考，不替代原文。参考文献部分保留英文不译。

目录 / Contents

P001 · 注意力就是你所需要的一切 / Attention Is All You Need 3
P004 · 顿悟：超越过拟合的泛化 / Grokking: Generalization Beyond Overfitting 14
P002 · 直接偏好优化 / Direct Preference Optimization (DPO) 20
P003 · 上下文学习作为梯度下降 / In-Context Learning as Gradient Descent 39

P001 · 注意力就是你所需要的一切

Attention Is All You Need

作者：Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Lukasz, Polosukhin, Illia｜2017/06/12｜阶段：S1 S3｜原文：https://arxiv.org/abs/1706.03762

> 对照说明：灰底为英文原文段落，下方为中文译文；参考文献部分保留英文原文。翻译由本地 qwen3.8:27b 逐段生成，供中文读者参考。

Provided proper attribution is provided, Google hereby grants permission to reproduce the tables and figures in this paper solely for use in journalistic or scholarly works. Attention Is All You Need Ashish Vaswani∗ Noam Shazeer∗ Google Brain avaswani@google.com Google Brain noam@google.com Llion Jones∗ Google Research llion@google.com Aidan N. Gomez∗ † University of Toronto aidan@cs.toronto.edu Illia Polosukhin∗ ‡ illia.polosukhin@gmail.com

在提供适当署名归属的前提下，谷歌特此授权，仅允许在新闻或学术作品中复制本文中的表格和图表。 注意力就是你所需要的一切 阿希什・瓦斯瓦尼∗ 诺姆・沙泽尔∗ 谷歌大脑 avaswani@google.com 谷歌大脑 noam@google.com 利昂・琼斯∗ 谷歌研究 llion@google.com 艾登・N・戈麦斯∗ † 多伦多大学 aidan@cs.toronto.edu 伊利亚・波洛苏金∗ ‡ illia.polosukhin@gmail.com

Abstract The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English- to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data. ∗Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

摘要 主流的序列转换模型基于包含编码器和解码器的复杂循环或卷积神经网络。表现最佳的模型还通过注意力机制连接编码器和解码器。我们提出了一种新的简单网络架构——Transformer，它完全基于注意力机制，彻底摒弃了循环和卷积。在两个机器翻译任务上的实验表明，这些模型在质量上更优，同时具有更高的并行性，且训练所需时间显著减少。我们的模型在WMT 2014英德翻译任务上取得了28.4的BLEU分数，比包括集成模型在内的现有最佳结果高出2个BLEU以上。在WMT 2014英法翻译任务上，我们的模型在八块GPU上训练3.5天后，建立了新的单模型最先进BLEU分数41.8，这一训练成本仅占文献中最佳模型训练成本的一小部分。我们通过在大规模和有限训练数据上成功应用该模型进行英语成分句法分析，证明了Transformer在其他任务上也具有良好的泛化能力。 ∗同等贡献。列出顺序是随机的。Jakob提出了用自注意力替代RNN的想法，并启动了评估这一想法的工作。Ashish与Illia一起设计并实现了第一个Transformer模型，并深度参与了这项工作的各个方面。Noam提出了缩放点积注意力、多头注意力以及无参数的位置表示，并几乎参与了每一个细节。Niki在我们的原始代码库和tensor2tensor中设计、实现、调优并评估了无数种模型变体。Llion也实验了新颖的模型变体，负责我们的初始代码库，以及高效的推理和可视化。Lukasz和Aidan花费了无数漫长的日子设计tensor2tensor的各个部分并实现它，取代了我们早期的代码库，极大地改善了结果并大幅加速了我们的研究。

† Work performed while at Google Brain. ‡ Work performed while at Google Research. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA. Niki Parmar∗ Google Research nikip@google.com Jakob Uszkoreit∗ Google Research usz@google.com Łukasz Kaiser∗ Google Brain lukaszkaiser@google.com

† 在 Google Brain 任职期间完成的工作。 ‡ 在 Google Research 任职期间完成的工作。 第31届神经信息处理系统大会（NIPS 2017），美国加利福尼亚州长滩。 Niki Parmar∗ Google Research nikip@google.com Jakob Uszkoreit∗ Google Research usz@google.com Łukasz Kaiser∗ Google Brain lukaszkaiser@google.com

> Introduction Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15]. Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states ht, as a function of the previous hidden state ht−1 and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains. Attention mechanisms have become an integral part of compelling sequence modeling and transduc- tion models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 19]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network.

引言 循环神经网络，特别是长短期记忆网络[13]和门控循环[7]神经网络，已在序列建模和转换问题（如语言建模和机器翻译[35, 2, 5]）中牢固确立为最先进的技术。此后，大量工作继续推动循环语言模型和编码器-解码器架构的边界[38, 24, 15]。 循环模型通常沿输入和输出序列的符号位置分解计算。通过将位置与计算时间步对齐，它们根据前一个隐藏状态 ht−1 和位置 t 的输入，生成隐藏状态序列 ht。这种固有的顺序特性排除了训练样本内的并行化，这在序列长度较长时变得至关重要，因为内存限制限制了跨样本的批处理。近期工作通过分解技巧[21]和条件计算[32]实现了计算效率的显著提升，同时后者还提升了模型性能。然而，顺序计算的根本限制依然存在。 注意力机制已成为各种任务中强大的序列建模和转换模型的组成部分，允许对依赖关系进行建模，而无需考虑其在输入或输出序列中的距离[2, 19]。然而，除少数情况[27]外，此类注意力机制通常与循环网络结合使用。

> In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

在本工作中，我们提出了Transformer，这是一种摒弃循环结构、完全依赖注意力机制来建立输入与输出之间全局依赖关系的模型架构。Transformer能够显著提升并行化程度，在仅使用八块P100 GPU训练十二小时后，即可在翻译质量上达到新的最先进水平。

> Background The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions [12]. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2. Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 27, 28, 22]. End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [34].

背景 减少顺序计算的目标也是扩展神经GPU [16]、ByteNet [18] 和 ConvS2S [9] 的基础，这些模型均使用卷积神经网络作为基本构建模块，并行计算所有输入和输出位置的隐藏表示。在这些模型中，关联任意两个输入或输出位置信号所需的操作数量随位置之间的距离而增加，在 ConvS2S 中呈线性增长，在 ByteNet 中呈对数增长。这使得学习远距离位置之间的依赖关系变得更加困难 [12]。在 Transformer 中，这一需求被减少为常数级别的操作，尽管由于对注意力加权位置进行平均而导致有效分辨率降低，但我们通过第 3.2 节中描述的多头注意力机制来抵消这种影响。 自注意力（有时也称为内部注意力）是一种注意力机制，旨在关联单个序列中不同位置，以计算该序列的表示。自注意力已成功应用于多种任务，包括阅读理解、摘要生成、文本蕴含以及学习任务无关的句子表示 [4, 27, 28, 22]。 端到端记忆网络基于循环注意力机制而非序列对齐的循环结构，并已被证明在简单语言问答和语言建模任务中表现良好 [34]。

> To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence- aligned RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [17, 18] and [9]. Model Architecture Most competitive neural sequence transduction models have an encoder-decoder structure [5, 2, 35]. Here, the encoder maps an input sequence of symbol representations (x1, ..., xn) to a sequence of continuous representations z = (z1, ..., zn). Given z, the decoder then generates an output sequence (y1, ..., ym) of symbols one element at a time. At each step the model is auto-regressive [10], consuming the previously generated symbols as additional input when generating the next. The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively.

据我们所知，Transformer 是首个完全依赖自注意力机制来计算输入和输出表示的转换模型，且未使用序列对齐的循环神经网络（RNN）或卷积。在接下来的章节中，我们将描述 Transformer，阐述自注意力机制的动机，并讨论其相对于 [17, 18] 和 [9] 等模型的优势。 模型架构 大多数具有竞争力的神经序列转换模型都采用编码器-解码器结构 [5, 2, 35]。在这里，编码器将符号表示的输入序列 $(x_1, ..., x_n)$ 映射为连续表示的序列 $z = (z_1, ..., z_n)$。给定 z，解码器随后逐个元素地生成符号输出序列 $(y_1, ..., y_m)$。在每一步中，模型是自回归的 [10]，在生成下一个符号时，将先前生成的符号作为额外输入。 Transformer 遵循这一整体架构，其编码器和解码器均使用堆叠的自注意力层和逐点全连接层，分别如图 1 的左半部分和右半部分所示。

> 3.1 Encoder and Decoder Stacks Encoder: The encoder is composed of a stack of N = 6 identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position- wise fully connected feed-forward network. We employ a residual connection [11] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is LayerNorm(x + Sublayer(x)), where Sublayer(x) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension dmodel = 512. Decoder: The decoder is also composed of a stack of N = 6 identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.

3.1 编码器与解码器堆栈 编码器：编码器由 N = 6 个相同的层堆叠组成。每层包含两个子层。第一个是多头自注意力机制，第二个是简单的逐位置全连接前馈网络。我们在每个子层周围采用残差连接 [11]，随后进行层归一化 [1]。也就是说，每个子层的输出为 LayerNorm(x + Sublayer(x))，其中 Sublayer(x) 是该子层本身实现的函数。为了便于这些残差连接，模型中的所有子层以及嵌入层均产生维度为 dmodel = 512 的输出。 解码器：解码器同样由 N = 6 个相同的层堆叠组成。除了编码器每层中的两个子层外，解码器还插入第三个子层，该子层对编码器堆栈的输出执行多头注意力。与编码器类似，我们在每个子层周围采用残差连接，随后进行层归一化。我们还修改了解码器堆栈中的自注意力子层，以防止位置关注后续位置。这种掩码机制，结合输出嵌入偏移一个位置的事实，确保了位置 i 的预测仅依赖于小于 i 的位置上的已知输出。

> 3.2 Attention An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum Scaled Dot-Product Attention Multi-Head Attention attention layers running in parallel. of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.2 注意力机制 注意力函数可以描述为将一个查询（query）和一组键值对（key-value pairs）映射到输出，其中查询、键、值和输出均为向量。输出被计算为值的加权和，其中分配给每个值的权重由查询与相应键的兼容性函数计算得出。 缩放点积注意力 多头注意力 并行运行的注意力层。

3.2.1 Scaled Dot-Product Attention We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension dk, and values of dimension dv. We compute the dot products of the query with all keys, divide each by √dk, and apply a softmax function to obtain the weights on the values. In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V . We compute the matrix of outputs as: Attention(Q, K, V) = softmax(QKT The two most commonly used attention functions are additive attention [2], and dot-product (multi- plicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of √dk . Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code. While for small values of dk the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of dk [3]. We suspect that for large values of dk, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients 4. To counteract this effect, we scale the dot products by √dk .

3.2.1 缩放点积注意力 我们将这种特定的注意力机制称为“缩放点积注意力”（图2）。输入由维度为 d_k 的查询（queries）和键（keys），以及维度为 d_v 的值（values）组成。我们计算查询与所有键的点积，将每个结果除以 $\sqrt{d_k}$，并应用 softmax 函数以获取值的权重。 在实践中，我们同时对一组查询计算注意力函数，并将它们打包成一个矩阵 Q。键和值也被打包成矩阵 K 和 V。我们计算输出矩阵如下： Attention(Q, K, V) = softmax(QK^T / \sqrt{d_k})V 两种最常用的注意力函数是加性注意力 [2] 和点积（乘性）注意力。点积注意力与我们的算法相同，只是缺少 $\sqrt{d_k}$ 的缩放因子。加性注意力使用具有单个隐藏层的前馈网络来计算兼容性函数。虽然两者在理论复杂度上相似，但点积注意力在实际中更快且更节省空间，因为它可以使用高度优化的矩阵乘法代码来实现。 虽然对于较小的 d_k 值，这两种机制表现相似，但对于较大的 d_k 值，未缩放的点积注意力表现不如加性注意力 [3]。我们推测，对于较大的 d_k 值，点积的幅度会变得很大，将 softmax 函数推入梯度极小的区域 4。为了抵消这种效应，我们将点积除以 $\sqrt{d_k}$ 进行缩放。

3.2.2 Multi-Head Attention Instead of performing a single attention function with dmodel-dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to dk, dk and dv dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding dv-dimensional 4To illustrate why the dot products get large, assume that the components of q and k are independent random variables with mean 0 and variance 1. Then their dot product, q · k = Pdk √dk)V (1) i=1 qiki, has mean 0 and variance dk. output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this. MultiHead(Q, K, V) = Concat(head1, ..., headh)W O where headi = Attention(QW Q Where the projections are parameter matrices W Q i ∈Rdmodel×dv and W O ∈Rhdv×dmodel. In this work we employ h = 8 parallel attention layers, or heads. For each of these we use dk = dv = dmodel/h = 64. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

3.2.2 多头注意力 我们发现，相比于使用 dmodel 维度的键、值和查询执行单一的注意力函数，将查询、键和值分别通过不同的、学习得到的线性投影线性映射 h 次到 dk、dk 和 dv 维度是有益的。然后，我们在这些投影后的查询、键和值的每个版本上并行执行注意力函数，得到 dv 维度的输出值。这些输出值被拼接起来并再次进行投影，从而得到最终的值，如图2所示。 4 为了说明为什么点积会变得很大，假设 q 和 k 的分量是均值为0、方差为1的独立随机变量。那么它们的点积 q · k = Σ_{i=1}^{dk} q_i k_i 的均值为0，方差为 dk。 多头注意力允许模型在不同位置共同关注来自不同表示子空间的信息。如果使用单个注意力头，平均操作会抑制这种能力。 MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O 其中 head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) 其中投影是参数矩阵 W_i^Q, W_i^K, W_i^V ∈ R^{dmodel×dk} 和 W^O ∈ R^{hdv×dmodel}。 在本工作中，我们采用 h = 8 个并行注意力层，即头。对于每个头，我们使用 dk = dv = dmodel/h = 64。由于每个头的维度降低，总计算成本与具有完整维度的单头注意力相似。

> 3.2.3 Applications of Attention in our Model The Transformer uses multi-head attention in three different ways: · In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [38, 2, 9]. · The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder. · Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to −∞) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

3.2.3 注意力机制在我们模型中的应用 Transformer 以三种不同的方式使用多头注意力： · 在“编码器-解码器注意力”层中，查询（queries）来自前一个解码器层，而记忆键（memory keys）和值（values）来自编码器的输出。这使得解码器中的每个位置都能关注输入序列中的所有位置。这模拟了诸如 [38, 2, 9] 等序列到序列模型中典型的编码器-解码器注意力机制。 · 编码器包含自注意力层。在自注意力层中，所有的键、值和查询都来自同一位置，在本例中，即编码器前一层级的输出。编码器中的每个位置都可以关注编码器前一层级中的所有位置。 · 类似地，解码器中的自注意力层允许解码器中的每个位置关注解码器中直到并包括该位置在内的所有位置。我们需要阻止解码器中的向左信息流动，以保持自回归特性。我们在缩放点积注意力内部实现这一点，通过屏蔽（设置为 −∞）softmax 输入中对应于非法连接的所有值。参见图 2。

> 3.3 Position-wise Feed-Forward Networks In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between. FFN(x) = max(0, xW1 + b1)W2 + b2 (2) While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is dmodel = 512, and the inner-layer has dimensionality dff = 2048.

3.3 逐位置前馈网络 除了注意力子层之外，编码器和解码器中的每一层都包含一个全连接前馈网络，该网络被独立且相同地应用于每个位置。它由两个线性变换组成，中间夹着一个ReLU激活函数。 FFN(x) = max(0, xW1 + b1)W2 + b2 (2) 虽然线性变换在不同位置上是相同的，但它们在不同层之间使用不同的参数。另一种描述方式是将其视为两个核大小为1的卷积。输入和输出的维度为 dmodel = 512，内部层的维度为 dff = 2048。

> 3.4 Embeddings and Softmax Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension dmodel. We also use the usual learned linear transfor- mation and softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [30]. In the embedding layers, we multiply those weights by √dmodel. i , KW K i , V W V i) i ∈Rdmodel×dk, W V i ∈Rdmodel×dk, W K for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention. Layer Type Complexity per Layer Sequential Maximum Path Length Operations Self-Attention O(n2 · d) O(1) O(1) Recurrent O(n · d2) O(n) O(n) Convolutional O(k · n · d2) O(1) O(logk(n)) Self-Attention (restricted) O(r · n · d) O(1) O(n/r)

3.4 嵌入与Softmax 与其他序列转换模型类似，我们使用学习到的嵌入将输入和输出标记转换为维度为 dmodel 的向量。我们还使用常规的学习线性变换和softmax函数，将解码器输出转换为预测的下一个标记概率。在我们的模型中，类似于[30]，我们在两个嵌入层和softmax前的线性变换之间共享相同的权重矩阵。在嵌入层中，我们将这些权重乘以 √dmodel。 i , KW K i , V W V i) i ∈Rdmodel×dk, W V i ∈Rdmodel×dk, W K 针对不同层类型。n 是序列长度，d 是表示维度，k 是卷积的核大小，r 是受限自注意力中的邻域大小。 层类型 每层复杂度 顺序操作 最大路径长度 自注意力 O(n2 · d) O(1) O(1) 循环 O(n · d2) O(n) O(n) 卷积 O(k · n · d2) O(1) O(logk(n)) 自注意力（受限） O(r · n · d) O(1) O(n/r)

> 3.5 Positional Encoding Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension dmodel as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [9]. In this work, we use sine and cosine functions of different frequencies: PE(pos,2i) = sin(pos/100002i/dmodel) PE(pos,2i+1) = cos(pos/100002i/dmodel) where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to 10000 · 2π. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PEpos+k can be represented as a linear function of PEpos. We also experimented with using learned positional embeddings [9] instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

3.5 位置编码 由于我们的模型不包含循环和卷积，为了使模型能够利用序列的顺序，我们必须注入有关序列中词元相对或绝对位置的信息。为此，我们在编码器和解码器堆栈的底部将“位置编码”添加到输入嵌入中。位置编码与嵌入具有相同的维度 dmodel，因此两者可以相加。位置编码有许多选择，包括学习得到的和固定的 [9]。 在本工作中，我们使用不同频率的正弦和余弦函数： PE(pos,2i) = sin(pos/100002i/dmodel) PE(pos,2i+1) = cos(pos/100002i/dmodel) 其中 pos 是位置，i 是维度。也就是说，位置编码的每个维度对应一个正弦波。波长从 2π 到 10000 · 2π 形成几何级数。我们选择此函数是因为我们假设它能让模型轻松学会按相对位置进行注意力计算，因为对于任何固定的偏移量 k，PEpos+k 都可以表示为 PEpos 的线性函数。我们还尝试使用学习得到的位置嵌入 [9] 来代替，发现这两个版本产生了几乎相同的结果（见表 3 第 (E) 行）。我们选择正弦版本，因为它可能允许模型外推到比训练期间遇到的序列更长的序列长度。

> Why Self-Attention In this section we compare various aspects of self-attention layers to the recurrent and convolu- tional layers commonly used for mapping one variable-length sequence of symbol representations (x1, ..., xn) to another sequence of equal length (z1, ..., zn), with xi, zi ∈Rd, such as a hidden layer in a typical sequence transduction encoder or decoder. Motivating our use of self-attention we consider three desiderata. One is the total computational complexity per layer. Another is the amount of computation that can be parallelized, as measured by the minimum number of sequential operations required. The third is the path length between long-range dependencies in the network. Learning long-range dependencies is a key challenge in many sequence transduction tasks. One key factor affecting the ability to learn such dependencies is the length of the paths forward and backward signals have to traverse in the network. The shorter these paths between any combination of positions in the input and output sequences, the easier it is to learn long-range dependencies [12]. Hence we also compare the maximum path length between any two input and output positions in networks composed of the different layer types. As noted in Table 1, a self-attention layer connects all positions with a constant number of sequentially executed operations, whereas a recurrent layer requires O(n) sequential operations. In terms of computational complexity, self-attention layers are faster than recurrent layers when the sequence

为什么选择自注意力 在本节中，我们将自注意力层与常用于将一个变长符号表示序列 $(x_1, \dots, x_n)$ 映射到另一个等长序列 $(z_1, \dots, z_n)$ 的循环层和卷积层进行多方面比较，其中 $x_i, z_i \in \mathbb{R}^d$，例如典型序列转换编码器或解码器中的隐藏层。为了论证使用自注意力的动机，我们考虑三个期望属性。 其一是每层的总计算复杂度。其二是可并行化的计算量，以所需的最少顺序操作数来衡量。其三是网络中长距离依赖关系之间的路径长度。学习长距离依赖关系是许多序列转换任务中的关键挑战。影响学习此类依赖关系能力的一个关键因素是前向和后向信号在网络中必须遍历的路径长度。输入序列和输出序列中任意位置组合之间的路径越短，学习长距离依赖关系就越容易 [12]。因此，我们还比较了由不同层类型组成的网络中任意两个输入和输出位置之间的最大路径长度。 如表 1 所示，自注意力层通过恒定数量的顺序执行操作连接所有位置，而循环层需要 $O(n)$ 个顺序操作。就计算复杂度而言，当序列

> length n is smaller than the representation dimensionality d, which is most often the case with sentence representations used by state-of-the-art models in machine translations, such as word-piece [38] and byte-pair [31] representations. To improve computational performance for tasks involving very long sequences, self-attention could be restricted to considering only a neighborhood of size r in the input sequence centered around the respective output position. This would increase the maximum path length to O(n/r). We plan to investigate this approach further in future work. A single convolutional layer with kernel width k < n does not connect all pairs of input and output positions. Doing so requires a stack of O(n/k) convolutional layers in the case of contiguous kernels, or O(logk(n)) in the case of dilated convolutions [18], increasing the length of the longest paths between any two positions in the network. Convolutional layers are generally more expensive than recurrent layers, by a factor of k. Separable convolutions [6], however, decrease the complexity considerably, to O(k · n · d + n · d2). Even with k = n, however, the complexity of a separable convolution is equal to the combination of a self-attention layer and a point-wise feed-forward layer, the approach we take in our model. As side benefit, self-attention could yield more interpretable models. We inspect attention distributions from our models and present and discuss examples in the appendix. Not only do individual attention heads clearly learn to perform different tasks, many appear to exhibit behavior related to the syntactic and semantic structure of the sentences.

当序列长度 n 小于表示维度 d 时，这种情况在机器翻译中使用词元（word-piece）[38] 和字节对（byte-pair）[31] 表示等最先进模型的句子表示中最为常见。为了提升涉及超长序列任务的计算性能，可以将自注意力机制限制为仅考虑以相应输出位置为中心、大小为 r 的输入序列邻域。这将使最大路径长度增加到 O(n/r)。我们计划在后续工作中进一步研究这种方法。 单个卷积核宽度 k < n 的卷积层无法连接所有输入和输出位置对。对于连续卷积核，这需要堆叠 O(n/k) 个卷积层；而对于空洞卷积 [18]，则需要 O(log_k(n)) 层，这会增加网络中任意两个位置之间最长路径的长度。通常，卷积层比循环层更昂贵，成本高出 k 倍。然而，可分离卷积 [6] 显著降低了复杂度，降至 O(k · n · d + n · d^2)。然而，即使 k = n，可分离卷积的复杂度也等于自注意力层与逐点前馈层组合的复杂度，这正是我们模型所采用的方法。 作为一个附带的好处，自注意力机制可能产生更具可解释性的模型。我们检查了模型的注意力分布，并在附录中展示并讨论了相关示例。不仅各个注意力头明显学会了执行不同的任务，许多注意力头还表现出与句子的句法和语义结构相关的行为。

> Training This section describes the training regime for our models.

训练 本节描述了我们模型的训练方案。

> 5.1 Training Data and Batching We trained on the standard WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs. Sentences were encoded using byte-pair encoding [3], which has a shared source- target vocabulary of about 37000 tokens. For English-French, we used the significantly larger WMT 2014 English-French dataset consisting of 36M sentences and split tokens into a 32000 word-piece vocabulary [38]. Sentence pairs were batched together by approximate sequence length. Each training batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens.

5.1 训练数据与批处理 我们在标准的WMT 2014英德数据集上进行训练，该数据集包含约450万个句子对。句子使用字节对编码[3]进行编码，其源-目标共享词汇量约为37000个词元。对于英法翻译，我们使用了规模显著更大的WMT 2014英法数据集，该数据集包含3600万个句子，并将词元分割为32000个词片词汇[38]。句子对按近似序列长度进行批处理。每个训练批次包含一组句子对，其中源词元约为25000个，目标词元约为25000个。

> 5.2 Hardware and Schedule We trained our models on one machine with 8 NVIDIA P100 GPUs. For our base models using the hyperparameters described throughout the paper, each training step took about 0.4 seconds. We trained the base models for a total of 100,000 steps or 12 hours. For our big models,(described on the bottom line of table 3), step time was 1.0 seconds. The big models were trained for 300,000 steps (3.5 days).

5.2 硬件与训练时长 我们在配备8块NVIDIA P100 GPU的单台机器上训练了我们的模型。对于使用本文所述超参数的基础模型，每个训练步骤耗时约0.4秒。基础模型共训练了100,000步，耗时12小时。对于大模型（见表3最后一行），每个步骤耗时1.0秒。大模型共训练了300,000步，耗时3.5天。

> 5.3 Optimizer We used the Adam optimizer [20] with β1 = 0.9, β2 = 0.98 and ϵ = 10−9. We varied the learning rate over the course of training, according to the formula: lrate = d−0.5 model · min(step_num−0.5, step_num · warmup_steps−1.5) (3) This corresponds to increasing the learning rate linearly for the first warmup_steps training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used warmup_steps = 4000.

5.3 优化器 我们使用了 Adam 优化器 [20]，其中 β1 = 0.9，β2 = 0.98，ϵ = 10−9。我们在训练过程中根据以下公式调整学习率： lrate = d−0.5_model · min(step_num−0.5, step_num · warmup_steps−1.5) (3) 这对应于在前 warmup_steps 个训练步骤中线性增加学习率，此后则按步数的平方根倒数比例减小学习率。我们使用了 warmup_steps = 4000。

> 5.4 Regularization We employ three types of regularization during training: English-to-German and English-to-French newstest2014 tests at a fraction of the training cost. Model EN-DE EN-FR EN-DE EN-FR ByteNet [18] 23.75 Deep-Att + PosUnk [39] 39.2 1.0 · 1020 GNMT + RL [38] 24.6 39.92 2.3 · 1019 1.4 · 1020 ConvS2S [9] 25.16 40.46 9.6 · 1018 1.5 · 1020 MoE [32] 26.03 40.56 2.0 · 1019 1.2 · 1020 Deep-Att + PosUnk Ensemble [39] 40.4 8.0 · 1020 GNMT + RL Ensemble [38] 26.30 41.16 1.8 · 1020 1.1 · 1021 ConvS2S Ensemble [9] 26.36 41.29 7.7 · 1019 1.2 · 1021 Transformer (base model) 27.3 38.1 3.3 · 1018 Transformer (big) 28.4 41.8 2.3 · 1019 Residual Dropout We apply dropout [33] to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of Pdrop = 0.1. Label Smoothing During training, we employed label smoothing of value ϵls = 0.1 [36]. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

5.4 正则化 我们在训练过程中采用了三种类型的正则化： 在英德和英法 newstest2014 测试中，以部分训练成本取得了以下结果。 模型 EN-DE EN-FR EN-DE EN-FR ByteNet [18] 23.75 Deep-Att + PosUnk [39] 39.2 1.0 · 1020 GNMT + RL [38] 24.6 39.92 2.3 · 1019 1.4 · 1020 ConvS2S [9] 25.16 40.46 9.6 · 1018 1.5 · 1020 MoE [32] 26.03 40.56 2.0 · 1019 1.2 · 1020 Deep-Att + PosUnk Ensemble [39] 40.4 8.0 · 1020 GNMT + RL Ensemble [38] 26.30 41.16 1.8 · 1020 1.1 · 1021 ConvS2S Ensemble [9] 26.36 41.29 7.7 · 1019 1.2 · 1021 Transformer (base model) 27.3 38.1 3.3 · 1018 Transformer (big) 28.4 41.8 2.3 · 1019 残差 Dropout 我们将 dropout [33] 应用于每个子层的输出，在其加到子层输入并进行归一化之前。此外，我们在编码器和解码器堆栈中，对嵌入和位置编码的和应用 dropout。对于基础模型，我们使用 Pdrop = 0.1 的比率。 标签平滑 在训练期间，我们采用了值为 ϵls = 0.1 的标签平滑 [36]。这会损害困惑度，因为模型学会变得更加不确定，但提高了准确率和 BLEU 分数。

> Results

结果

> 6.1 Machine Translation On the WMT 2014 English-to-German translation task, the big transformer model (Transformer (big) in Table 2) outperforms the best previously reported models (including ensembles) by more than 2.0 BLEU, establishing a new state-of-the-art BLEU score of 28.4. The configuration of this model is listed in the bottom line of Table 3. Training took 3.5 days on 8 P100 GPUs. Even our base model surpasses all previously published models and ensembles, at a fraction of the training cost of any of the competitive models. On the WMT 2014 English-to-French translation task, our big model achieves a BLEU score of 41.0, outperforming all of the previously published single models, at less than 1/4 the training cost of the previous state-of-the-art model. The Transformer (big) model trained for English-to-French used dropout rate Pdrop = 0.1, instead of 0.3. For the base models, we used a single model obtained by averaging the last 5 checkpoints, which were written at 10-minute intervals. For the big models, we averaged the last 20 checkpoints. We used beam search with a beam size of 4 and length penalty α = 0.6 [38]. These hyperparameters were chosen after experimentation on the development set. We set the maximum output length during inference to input length + 50, but terminate early when possible [38]. architectures from the literature. We estimate the number of floating point operations used to train a model by multiplying the training time, the number of GPUs used, and an estimate of the sustained single-precision floating-point capacity of each GPU 5.

6.1 机器翻译 在WMT 2014英德翻译任务中，大型Transformer模型（表2中的Transformer (big)）比之前报告的最佳模型（包括集成模型）高出2.0 BLEU以上，确立了28.4的新最先进BLEU分数。该模型的配置列于表3的最后一行。训练在8块P100 GPU上耗时3.5天。即使是我们的小型模型也超越了所有已发表的模型和集成模型，且训练成本仅为任何竞争模型的一小部分。 在WMT 2014英法翻译任务中，我们的大型模型取得了41.0的BLEU分数，超越了所有已发表的单一模型，且训练成本不到之前最先进模型的1/4。用于英法翻译训练的Transformer (big)模型使用了Pdrop = 0.1的丢弃率，而不是0.3。 对于小型模型，我们使用了通过对最后5个检查点（以10分钟间隔写入）进行平均得到的单一模型。对于大型模型，我们对最后20个检查点进行了平均。我们使用了束宽为4、长度惩罚α = 0.6的束搜索[38]。这些超参数是在开发集上经过实验后选定的。我们将推理时的最大输出长度设置为输入长度+50，但在可能时提前终止[38]。 文献中的架构。我们通过将训练时间、使用的GPU数量以及每块GPU的持续单精度浮点运算能力估计值相乘，来估算训练模型所使用的浮点运算次数5。

6.2 Model Variations To evaluate the importance of different components of the Transformer, we varied our base model in different ways, measuring the change in performance on English-to-German translation on the 5We used values of 2.8, 3.7, 6.0 and 9.5 TFLOPS for K80, K40, M40 and P100, respectively. BLEU Training Cost (FLOPs) model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities. N dmodel dff h dk dv Pdrop ϵls train PPL BLEU params steps (dev) (dev) ×106 base 0.1 0.1 100K 4.92 25.8 5.29 24.9 5.00 25.5 4.91 25.8 5.01 25.4 (A) (B) 5.16 25.1 5.01 25.4 6.11 23.7 5.19 25.3 4.88 25.5 5.75 24.5 4.66 26.0 5.12 25.4 4.75 26.2 (C) 0.0 5.77 24.6 0.2 4.95 25.5 0.0 4.67 25.3 0.2 5.47 25.7 (E) positional embedding instead of sinusoids 4.92 25.7 big 0.3 300K 4.33 26.4 (D) development set, newstest2013. We used beam search as described in the previous section, but no checkpoint averaging. We present these results in Table 3. In Table 3 rows (A), we vary the number of attention heads and the attention key and value dimensions, keeping the amount of computation constant, as described in Section 3.2.2. While single-head attention is 0.9 BLEU worse than the best setting, quality also drops off with too many heads. In Table 3 rows (B), we observe that reducing the attention key size dk hurts model quality. This suggests that determining compatibility is not easy and that a more sophisticated compatibility function than dot product may be beneficial. We further observe in rows (C) and (D) that, as expected, bigger models are better, and dropout is very helpful in avoiding over-fitting. In row (E) we replace our sinusoidal positional encoding with learned positional embeddings [9], and observe nearly identical results to the base model.

6.2 模型变体 为了评估 Transformer 不同组件的重要性，我们以不同方式改变了基础模型，并测量了其在英语到德语翻译任务上的性能变化。所有指标均基于英语到德语翻译的开发集 newstest2013。列出的困惑度（perplexity）是基于词片（wordpiece）的，依据我们的字节对编码（byte-pair encoding），不应与基于词的困惑度进行比较。 N dmodel dff h dk dv Pdrop ϵls train PPL BLEU params steps (dev) (dev) ×106 base 0.1 0.1 100K 4.92 25.8 5.29 24.9 5.00 25.5 4.91 25.8 5.01 25.4 (A) (B) 5.16 25.1 5.01 25.4 6.11 23.7 5.19 25.3 4.88 25.5 5.75 24.5 4.66 26.0 5.12 25.4 4.75 26.2 (C) 0.0 5.77 24.6 0.2 4.95 25.5 0.0 4.67 25.3 0.2 5.47 25.7 (E) 使用学习的位置嵌入代替正弦位置编码 4.92 25.7 big 0.3 300K 4.33 26.4 (D) 开发集 newstest2013。我们使用了上一节中描述的束搜索（beam search），但未使用检查点平均。我们在表 3 中展示了这些结果。在表 3 的行 (A) 中，我们改变了注意力头数以及注意力键和值的维度，同时保持计算量恒定，如第 3.2.2 节所述。虽然单头注意力比最佳设置低 0.9 BLEU，但头数过多时质量也会下降。 在表 3 的行 (B) 中，我们观察到减小注意力键的大小 dk 会损害模型质量。这表明确定兼容性并不容易，且比点积更复杂的兼容性函数可能有益。我们进一步在行 (C) 和 (D) 中观察到，正如预期那样，更大的模型表现更好，且 dropout 对于避免过拟合非常有帮助。在行 (E) 中，我们用学习的位置嵌入 [9] 替换了我们的正弦位置编码，并观察到与基础模型几乎相同的结果。

6.3 English Constituency Parsing To evaluate if the Transformer can generalize to other tasks we performed experiments on English constituency parsing. This task presents specific challenges: the output is subject to strong structural constraints and is significantly longer than the input. Furthermore, RNN sequence-to-sequence models have not been able to attain state-of-the-art results in small-data regimes [37]. We trained a 4-layer transformer with dmodel = 1024 on the Wall Street Journal (WSJ) portion of the Penn Treebank [25], about 40K training sentences. We also trained it in a semi-supervised setting, using the larger high-confidence and BerkleyParser corpora from with approximately 17M sentences [37]. We used a vocabulary of 16K tokens for the WSJ only setting and a vocabulary of 32K tokens for the semi-supervised setting. We performed only a small number of experiments to select the dropout, both attention and residual (section 5.4), learning rates and beam size on the Section 22 development set, all other parameters remained unchanged from the English-to-German base translation model. During inference, we of WSJ) Parser Training WSJ 23 F1 Vinyals & Kaiser el al. (2014) [37] WSJ only, discriminative 88.3 Petrov et al. (2006) [29] WSJ only, discriminative 90.4 Zhu et al. (2013) [40] WSJ only, discriminative 90.4 Dyer et al. (2016) [8] WSJ only, discriminative 91.7 Transformer (4 layers) WSJ only, discriminative 91.3 Zhu et al. (2013) [40] semi-supervised 91.3 Huang & Harper (2009) [14] semi-supervised 91.3 McClosky et al. (2006) [26] semi-supervised 92.1 Vinyals & Kaiser el al. (2014) [37] semi-supervised 92.1 Transformer (4 layers) semi-supervised 92.7 Luong et al. (2015) [23] multi-task 93.0 Dyer et al. (2016) [8] generative 93.3

6.3 英语成分句法分析 为了评估Transformer能否泛化到其他任务，我们在英语成分句法分析上进行了实验。该任务具有特定的挑战：输出受到强结构约束，且显著长于输入。此外，循环神经网络（RNN）序列到序列模型在小数据规模下未能达到最先进的结果 [37]。 我们在宾州树库 [25] 的《华尔街日报》（WSJ）部分训练了一个4层Transformer，其中 dmodel = 1024，训练句数约为4万句。我们还在半监督设置下进行了训练，使用了来自 [37] 的更大规模的高置信度语料库和BerkleyParser语料库，包含约1700万句。在仅使用WSJ的设置中，我们使用了1.6万个词元的词汇表；在半监督设置中，使用了3.2万个词元的词汇表。 我们仅进行了少量实验，以在第22节开发集上选择dropout（包括注意力和残差dropout，见5.4节）、学习率和束搜索大小，其他所有参数均保持与英德基础翻译模型不变。在推理过程中，我们 WSJ） 解析器训练 WSJ 23 F1 Vinyals & Kaiser el al. (2014) [37] 仅WSJ，判别式 88.3 Petrov et al. (2006) [29] 仅WSJ，判别式 90.4 Zhu et al. (2013) [40] 仅WSJ，判别式 90.4 Dyer et al. (2016) [8] 仅WSJ，判别式 91.7 Transformer (4 layers) 仅WSJ，判别式 91.3 Zhu et al. (2013) [40] 半监督 91.3 Huang & Harper (2009) [14] 半监督 91.3 McClosky et al. (2006) [26] 半监督 92.1 Vinyals & Kaiser el al. (2014) [37] 半监督 92.1 Transformer (4 layers) 半监督 92.7 Luong et al. (2015) [23] 多任务 93.0 Dyer et al. (2016) [8] 生成式 93.3

> increased the maximum output length to input length + 300. We used a beam size of 21 and α = 0.3 for both WSJ only and the semi-supervised setting. Our results in Table 4 show that despite the lack of task-specific tuning our model performs sur- prisingly well, yielding better results than all previously reported models with the exception of the Recurrent Neural Network Grammar [8]. In contrast to RNN sequence-to-sequence models [37], the Transformer outperforms the Berkeley- Parser [29] even when training only on the WSJ training set of 40K sentences.

将最大输出长度设置为输入长度加300。在仅使用WSJ数据集和半监督设置中，我们均使用了束宽为21且α = 0.3。 表4中的结果表明，尽管缺乏针对特定任务的调优，我们的模型表现依然出人意料地好，除循环神经网络语法[8]外，其结果优于所有先前报告的模型。 与循环神经网络序列到序列模型[37]不同，Transformer即使在仅使用包含4万句子的WSJ训练集上进行训练时，其性能也优于伯克利解析器[29]。

> Conclusion In this work, we presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention. For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, we achieve a new state of the art. In the former task our best model outperforms even all previously reported ensembles. We are excited about the future of attention-based models and plan to apply them to other tasks. We plan to extend the Transformer to problems involving input and output modalities other than text and to investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video. Making generation less sequential is another research goals of ours. The code we used to train and evaluate our models is available at https://github.com/ tensorflow/tensor2tensor. Acknowledgements We are grateful to Nal Kalchbrenner and Stephan Gouws for their fruitful comments, corrections and inspiration.

结论 在本工作中，我们提出了Transformer，这是首个完全基于注意力机制的序列转换模型，用多头自注意力机制取代了编码器-解码器架构中最常用的循环层。 在翻译任务中，Transformer的训练速度显著快于基于循环层或卷积层的架构。在WMT 2014英德翻译和WMT 2014英法翻译任务上，我们均达到了新的最先进水平。在前者任务中，我们的最佳模型甚至优于此前报告的所有集成模型。 我们对基于注意力机制的模型的未来感到兴奋，并计划将其应用于其他任务。我们计划将Transformer扩展到涉及文本之外的输入和输出模态的问题，并研究局部、受限的注意力机制，以高效处理图像、音频和视频等大型输入和输出。使生成过程更少串行化也是我们的另一个研究目标。 我们用于训练和评估模型的代码可在 https://github.com/tensorflow/tensor2tensor 获取。 致谢 我们感谢Nal Kalchbrenner和Stephan Gouws富有成效的评论、纠正和启发。

— 参考文献（原文保留） —

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. CoRR, abs/1409.0473, 2014. [3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. CoRR, abs/1703.03906, 2017. [4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. arXiv preprint arXiv:1601.06733, 2016. [5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. CoRR, abs/1406.1078, 2014. [6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357, 2016. [7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555, 2014. [8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In Proc. of NAACL, 2016. [9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolu- tional sequence to sequence learning. arXiv preprint arXiv:1705.03122v2, 2017. [10] Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for im-

age recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016. [12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001. [13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997. [14] Zhongqiang Huang and Mary Harper. Self-training PCFG grammars with latent annotations across languages. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 832–841. ACL, August 2009. [15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410, 2016. [16] Łukasz Kaiser and Samy Bengio. Can active memory replace attention? In Advances in Neural Information Processing Systems, (NIPS), 2016. [17] Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In International Conference on Learning Representations (ICLR), 2016. [18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Ko- ray Kavukcuoglu. Neural machine translation in linear time. arXiv preprint arXiv:1610.10099v2, 2017. [19] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In International Conference on Learning Representations, 2017. [20] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR, 2015.

[21] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for LSTM networks. arXiv preprint [22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. arXiv preprint [23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. arXiv preprint arXiv:1511.06114, 2015. [24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention- based neural machine translation. arXiv preprint arXiv:1508.04025, 2015. [25] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. Computational linguistics, 19(2):313–330, 1993. [26] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In pages 152–159. ACL, June 2006. [27] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In Empirical Methods in Natural Language Processing, 2016. [28] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304, 2017. [29] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 433–440. ACL, July 2006. [30] Ofir Press and Lior Wolf. Using the output embedding to improve language models. arXiv

preprint arXiv:1608.05859, 2016. [31] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909, 2015. [32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv preprint arXiv:1701.06538, 2017. [33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdi- nov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1):1929–1958, 2014. [34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 2440–2448. Curran Associates, Inc., 2015. [35] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems, pages 3104–3112, 2014. [36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. CoRR, abs/1512.00567, 2015. [37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Grammar as a foreign language. In Advances in Neural Information Processing Systems, 2015. [38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang

Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint [39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. CoRR, abs/1606.04199, 2016. [40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers), pages 434–443. ACL, August 2013. Attention Visualizations governments American majority passed have spirit laws new that this of in is a It It that of this is a spirit in laws new have American majority passed governments encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word ‘making’. Different colors represent different heads. Best viewed in color. registration process <EOS> making difficult <pad> <pad> <pad> <pad> <pad> <pad> voting since more the or . . <EOS> or the difficult more <pad> process since <pad> <pad> <pad> <pad> <pad> voting making registration application perfect should never Law The but will be be its , , its The perfect but be be Law never should will application application perfect should never Law The but will be be its , , its The perfect but be be Law never should will application Full attentions for head 5. Bottom: Isolated attentions from just the word ‘its’ for attention heads 5 and 6. Note that the attentions are very sharp for this word.

missing <EOS> opinion <pad> what just this are my we in is - , . . <EOS> , - this my is just in what are we <pad> missing opinion missing <EOS> opinion <pad> what just this are my we in is - , . . <EOS> , - this my is just in are what we <pad> missing opinion application perfect should never Law The but will be be its , , its The perfect but be be never Law should will application application perfect should never Law The but will be be its , , its The perfect but be be Law never should will application sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks. missing <EOS> opinion <pad> what just this are my we in is - , . . <EOS> , - this my is just in are what we <pad> missing opinion missing <EOS> opinion <pad> what just this are my we in is - , . . <EOS> , - this my is just in are what we <pad> missing opinion

P004 · 顿悟：超越过拟合的泛化

Grokking: Generalization Beyond Overfitting

作者：Power, Alethea, Burda, Yuri, Edwards, Harri, Babuschkin, Igor, Misra, Vedant｜2022/01/06｜阶段：S7 S8｜原文：https://arxiv.org/abs/2201.02177

> 对照说明：灰底为英文原文段落，下方为中文译文；参考文献部分保留英文原文。翻译由本地 qwen3.8:27b 逐段生成，供中文读者参考。

> GROKKING: GENERALIZATION BEYOND OVERFIT- TING ON SMALL ALGORITHMIC DATASETS Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin OpenAI

GROKKING：在小规模算法数据集上超越过拟合的泛化 Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin OpenAI

> ABSTRACT In this paper we propose to study generalization of neural networks on small al- gorithmically generated datasets. In this setting, questions about data efficiency, memorization, generalization, and speed of learning can be studied in great de- tail. In some situations we show that neural networks learn through a process of “grokking” a pattern in the data, improving generalization performance from random chance level to perfect generalization, and that this improvement in general- ization can happen well past the point of overfitting. We also study generalization as a function of dataset size and find that smaller datasets require increasing amounts of optimization for generalization. We argue that these datasets provide a fertile ground for studying a poorly understood aspect of deep learning: generalization of overparametrized neural networks beyond memorization of the finite training dataset.

摘要 本文提出研究神经网络在小规模算法生成数据集上的泛化能力。在这种设置下，可以深入研究数据效率、记忆、泛化以及学习速度等问题。在某些情况下，我们表明神经网络通过“顿悟”（grokking）数据中的模式来学习，将泛化性能从随机水平提升至完美泛化，并且这种泛化能力的提升可以发生在过拟合之后很久。我们还研究了泛化能力随数据集规模变化的情况，发现较小的数据集需要更多的优化才能实现泛化。我们认为，这些数据集为研究深度学习中一个理解不足的现象提供了肥沃的土壤：过参数化神经网络在超越有限训练数据集记忆之外的泛化能力。

> INTRODUCTION The generalization of overparameterized neural networks has long been a source of interest to the machine learning community since it defies intuitions derived from classical learning theory. In this paper we show that training networks on small algorithmically generated datasets can reliably exhibit unusual generalization patterns, clearly decoupled from performance on the training set, in a significantly more pronounced way than such effects manifest on datasets derived from natural data (see Figure 1, left, for an example). Such experiments can be quickly reproduced on a single GPU, and this makes them convenient testbeds for theories of generalization. dataset. We train on the binary operation of division mod 97 with 50% of the data in the training set. Each of the 97 residues is presented to the network as a separate symbol, similar to the representation in the figure to the right. The red curves show training accuracy and the green ones show validation accuracy. Training accuracy becomes close to perfect at < 103 optimization steps, but it takes close to 106 steps for validation accuracy to reach that level, and we see very little evidence of any generalization until 105 steps. Center. Training time required to reach 99% validation accuracy increases rapidly as the training data fraction decreases. Right. An example of a small binary operation table. We invite the reader to make their guesses as to which elements are missing.

引言 由于过参数化神经网络的泛化能力违背了从经典学习理论中得出的直觉，它长期以来一直是机器学习社区关注的焦点。本文表明，在小型算法生成数据集上训练网络，能够可靠地展现出异常的泛化模式，这种模式与训练集上的表现明显解耦，且其显著程度远超在源自自然数据的数据集上观察到的此类效应（参见图1左侧示例）。此类实验可在单块GPU上快速复现，这使其成为研究泛化理论的便捷测试平台。 数据集。我们使用模97的除法二元运算进行训练，训练集包含50%的数据。97个余数中的每一个都作为独立的符号呈现给网络，类似于右侧图中的表示方式。红色曲线显示训练准确率，绿色曲线显示验证准确率。训练准确率在少于10^3次优化步骤内接近完美，但验证准确率需要接近10^6次步骤才能达到该水平，并且在10^5次步骤之前，我们几乎没有观察到任何泛化的迹象。中间。达到99%验证准确率所需的训练时间随着训练数据比例的增加而迅速增加。右侧。一个小型二元运算表的示例。我们邀请读者猜测哪些元素缺失。

∗Vedant was at OpenAI at the time of this work Vedant Misra∗ Google The datasets we consider are binary operation tables of the form a ◦ b = c where a, b, c are discrete symbols with no internal structure, and ◦ is a binary operation. Examples of binary operations include addition, composition of permutations, and bivariate polynomials. Training a neural network on a proper subset of all possible equations then amounts to filling in the blanks of the binary op table, much like solving a Sudoku puzzle. An example is shown on the right in Figure 1. Since we use distinct abstract symbols for all distinct elements a, b, c involved in the equations, the network is not made aware of any internal structure of the elements, and has to learn about their properties only from their interactions with other elements. For example the network doesn’ t see numbers in decimal notation, or permutations in line notation. Our contributions are as follows: · We show that neural networks are capable of generalizing to the empty slots in a variety of binary op tables. · We show that, long after severely overfitting, validation accuracy sometimes suddenly begins to increase from chance level toward perfect generalization. We call this phenomenon ‘grokking’. An example is shown in Figure 1. · We present the data efficiency curves for a variety of binary operations. · We show empirically that the amount of optimization required for generalization quickly increases as the dataset size decreases. · We compare various optimization details to measure their impact on data efficiency. We find that weight decay is particularly effective at improving generalization on the tasks we study. · We visualize the symbol embeddings learned by these networks and find that they sometimes uncover recognizable structure of the mathematical objects represented by the symbols.

∗Vedant 在进行本项研究时隶属于 OpenAI Vedant Misra∗ Google 我们所考虑的数据集是形如 a ◦ b = c 的二元运算表，其中 a、b、c 是无内部结构的离散符号，◦ 为二元运算。二元运算的例子包括加法、置换的复合以及二元多项式。在所有可能方程的一个真子集上训练神经网络，相当于填补二元运算表中的空白，类似于解决数独谜题。图 1 右侧展示了一个示例。由于我们在方程中为所有不同的元素 a、b、c 使用不同的抽象符号，网络无法获知元素的任何内部结构，只能从它们与其他元素的相互作用中学习其性质。例如，网络看到的不是十进制表示的数字，也不是行表示法的置换。 我们的贡献如下： · 我们表明，神经网络能够针对各种二元运算表中的空位进行泛化。 · 我们表明，在严重过拟合很久之后，验证准确率有时会从随机水平突然开始向完美泛化增加。我们将这种现象称为“grokking”。图 1 中展示了一个示例。 · 我们展示了各种二元运算的数据效率曲线。 · 我们通过实证表明，随着数据集规模的减小，实现泛化所需的优化量迅速增加。 · 我们比较了各种优化细节，以衡量它们对数据效率的影响。我们发现，权重衰减在改善我们所研究任务的泛化能力方面特别有效。 · 我们可视化了这些网络学习到的符号嵌入，发现它们有时能揭示出符号所代表的数学对象的可识别结构。

METHOD All of our experiments used a small transformer trained on datasets of equations of the form a ◦ b = c, where each of “a”, “◦”, “b”, “=”, and “c” is a separate token. Details of the operations studied, the architecture, training hyperparameters and tokenization can be found in Appendix A.1.

方法 我们所有的实验均使用了一个小型变压器模型，该模型在形式为 a ◦ b = c 的方程数据集上进行训练，其中“a”、“◦”、“b”、“=”和“c”各自作为一个独立的词元。关于所研究运算、架构、训练超参数及词元化的详细信息，请参见附录 A.1。

EXPERIMENTS

实验

3.1 GENERALIZATION BEYOND OVERFITTING Deep learning practitioners are used to seeing small improvements in validation accuracy after validation loss stops decreasing. A double descent of validation loss has been documented in some circumstances, but is considered unusual among practitioners Nakkiran et al. (2019); Belkin et al. (2018); d’ Ascoli et al. (2020). On the small algorithmic datasets that we study, improved generalization after initial overfitting occurs for a range of models, optimizers, and dataset sizes, and in some cases these effects are extremely pronounced. A typical example is shown for modular division in Figure 1. There we see that validation accuracy starts increasing beyond chance level only after 1000 times more optimization steps than are required for training accuracy to get close to optimal. In Figure 4 the training/validation losses are also plotted and we see the double descent of the validation loss. We found these behaviors to be typical for all the binary operations for dataset sizes that were close to the minimal dataset size for which the network generalized within the allotted optimization budget. For larger dataset sizes, the training and validation curves tend to track each other more closely.

3.1 超越过拟合的泛化能力 深度学习从业者习惯于看到验证损失停止下降后，验证准确率出现小幅提升。在某些情况下，验证损失的双重下降现象已被记录在案，但在从业者中仍被视为异常现象 Nakkiran et al. (2019); Belkin et al. (2018); d’ Ascoli et al. (2020)。在我们研究的小型算法数据集上，初始过拟合后的泛化能力改善现象出现在多种模型、优化器和数据集规模中，且在某些情况下这些效应极为显著。图1展示了模除运算的一个典型示例。在那里我们看到，验证准确率仅在比训练准确率接近最优所需的优化步数多出1000倍之后，才开始从随机水平上升。在图4中，训练/验证损失也被绘制出来，我们可以看到验证损失的双重下降。 我们发现，对于接近网络在给定优化预算内实现泛化的最小数据集规模的数据集大小而言，这些行为在所有二元运算中都是典型的。对于更大的数据集规模，训练曲线和验证曲线往往更加紧密地相互跟随。

> 3.1.1 LEARNING TIME CURVES In a typical supervised learning problem, decreasing the amount of training data decreases the converged generalization performance of the model when the optimization procedure is capable of interpolating the training data. In our setting, we observe a different phenomenon: while the converged performance stays constant at 100% within a range of training dataset sizes, the optimization time required to achieve that performance grows quicky as the dataset size is decreased. an optimization budget of 105 steps for the problem of learning the product in the abstract group S5. Weight decay improves generalization the most, but some generalization happens even with full batch optimizers and models without weight or activation noise at high percentages of training data. Subop- timal choice hyperparameters severely limit generalization. Not shown: training accuracy reaches 100% after 103-104 updates for all optimization methods. Right. Best validation accuracy achieved after 105 steps on a variety of algorithmic datasets, averaged over 3 seeds. Generalization happens at higher percentages of data for intuitively more complicated and less symmetrical operations. reaches 99% for the product in abstract group S5. In the vicinity of 25-30% of data, a decrease of 1% of training data leads to an increase of 40-50% in median time to generalization. While the number of steps until validation accuracy > 99% grows quickly as dataset size decreases, the number of steps until the train accuracy first reaches 99% generally trends down as dataset size decreases and stays in the range of 103-104 optimization steps. We’ ve observed a similar pattern of exponential increase in optimization time until reaching generalization as dataset size decreases on all the algorithmic tasks for which we could get the networks to generalize.

3.1.1 学习时间曲线 在典型的监督学习问题中，当优化过程能够插值训练数据时，减少训练数据量会降低模型收敛后的泛化性能。在我们的设定中，我们观察到一种不同的现象：在一定的训练数据集大小范围内，收敛性能保持在100%不变，但达到该性能所需的优化时间随着数据集大小的减小而迅速增加。 针对在抽象群 S5 中学习乘积的问题，优化预算为 10^5 步。权重衰减对泛化的提升最大，但在高比例训练数据下，即使使用全批量优化器以及没有权重或激活噪声的模型，也会发生一定程度的泛化。次优的超参数选择会严重限制泛化。未展示：对于所有优化方法，训练准确率在 10^3-10^4 次更新后达到100%。右图。在多种算法数据集上，经过 10^5 步后达到的最佳验证准确率，对3个随机种子取平均。对于直觉上更复杂且对称性更低的运算，泛化发生在更高比例的数据下。 在抽象群 S5 的乘积学习中达到99%。在25-30%数据比例的附近，训练数据减少1%会导致中位泛化时间增加40-50%。虽然验证准确率超过99%所需的步数随着数据集大小的减小而迅速增加，但训练准确率首次达到99%所需的步数通常随着数据集大小的减小而呈下降趋势，并保持在 10^3-10^4 个优化步的范围内。我们在所有能够使网络实现泛化的算法任务中，都观察到了随着数据集大小减小，达到泛化所需的优化时间呈指数级增加的模式。

> 3.2 GROKKING ON A VARIETY OF PROBLEMS We’ ve measured the mean accuracy across three runs for training datasets consisting of different fractions of all available equations for a variety of binary operations listed in Appendix A.1.1. The results are presented in Figure 2 (right). Since the operands are presented to the neural network as unrelated abstract symbols, the operations x+y (mod p−1) and x∗y (mod p) with a prime number p and non-zero x, y are indistinguishable from the neural network’ s perspective (and similarly x −y (mod p −1) and x/y (mod p)). This is because every nonzero residue modulo a prime can be represented as a power of a primitive root. This representation shows the equivalence (up to renaming of symbols) of modular addition modulo p −1 and modular multiplication modulo p. We see in Figure 2 (right) that x −y and x/y indeed take about the same amount of data for generalization to occur. Some of the operations listed in Figure 2 (right) are symmetric with respect to the order of the operands (x + y, x ∗y, x2 + y2 and x2 + xy + y2). Such operations tend to require less data for generalization than closely related non-symmetrical counterparts (x −y, x/y, x2 + xy + y2 + x). We believe this effect might be partially architecture-dependent, since it’ s easy for a transformer to learn a symmetric function of the operands by ignoring positional embedding. Some operations (for example x3 + xy2 + y (mod 97)) didn’ t lead to generalization within the allowed optimization budget at any percentage of data up to 95%. The converged models effectively just memorized the training dataset without finding any real patterns in the data. To such a model, the data is effectively random.

3.2 多种问题上的顿悟现象 我们测量了三次运行的平均准确率，训练数据集由附录A.1.1中列出的各种二元运算的所有可用方程的不同比例组成。结果展示在图2（右）。 由于操作数以无关的抽象符号形式呈现给神经网络，对于素数p和非零x、y，运算x+y (mod p−1)和x∗y (mod p)在神经网络看来是无法区分的（同样地，x −y (mod p −1)和x/y (mod p)也是如此）。这是因为模素数的每个非零剩余都可以表示为原根的幂。这种表示揭示了模p−1的模加法和模p的模乘法之间的等价性（在符号重命名意义下）。我们在图2（右）中看到，x −y和x/y确实需要大致相同数量的数据才能发生泛化。 图2（右）中列出的一些运算关于操作数的顺序是对称的（x + y、x ∗y、x2 + y2和x2 + xy + y2）。这类运算往往比密切相关的非对称对应运算（x −y、x/y、x2 + xy + y2 + x）需要更少的数据来实现泛化。我们认为这种效应可能部分依赖于架构，因为Transformer很容易通过忽略位置嵌入来学习操作数的对称函数。 某些运算（例如x3 + xy2 + y (mod 97)）在高达95%的数据比例下，均未在允许的优化预算内导致泛化。收敛的模型实际上只是记住了训练数据集，而未能在数据中发现任何真实模式。对于此类模型而言，数据实际上是随机的。

> The operation [x/y (mod p) if y is odd, otherwise x −y (mod p)] requires the network to learn a mix of several simple operations - in particular the role of x has to be interpreted as a residue in the additive group when it’s paired with an even y, and as a residue in the multiplicative group when it’s paired with an odd y. This shows that generalization can happen even for operations that are not cleanly interpretable via group or ring operations.

操作 [x/y (mod p) 若 y 为奇数，否则 x −y (mod p)] 要求网络学习几种简单操作的混合——特别是当 x 与偶数 y 配对时，x 的角色必须被解释为加法群中的剩余；当 x 与奇数 y 配对时，则必须被解释为乘法群中的剩余。这表明，即使对于无法通过群或环运算清晰解释的操作，泛化现象依然可以发生。

> 3.3 ABLATIONS AND TRICKS We’ve tried various forms of regularization to see what can induce networks to generalize better on our datasets. Here we present the data efficiency curves on a particular dataset S5 for a variety of interventions: full-batch gradient descent, stochastic gradient descent, large or small learning rates, residual dropout Srivastava et al. (2014), weight decay Loshchilov & Hutter (2017) and gradient noise Neelakantan et al. (2015). The results are shown in Figure 2 (left). We find that adding weight decay has a very large effect on data efficiency, more than halving the amount of samples needed compared to most other interventions. We found that weight decay towards the initialization of the network is also effective, but not quite as effective as weight decay towards the origin. This makes us believe that the prior, that approximately zero weights are suitable for small algorithmic tasks, explains part, but not all of the superior performance of weight decay. Adding some noise to the optimization process (e.g. gradient noise from using minibatches, Gaussian noise applied to weights before or after computing the gradients) is beneficial for generalization, consistent with the idea that such noise might induce the optimization to find flatter minima that generalize better. We found that learning rate had to be tuned in a relatively narrow window for the generalization to happen (within 1 order of magnitude).

3.3 消融实验与技巧 我们尝试了各种形式的正则化，以观察哪些方法能够促使网络在我们的数据集上获得更好的泛化能力。在此，我们展示了在特定数据集 S5 上针对多种干预措施的数据效率曲线：全批量梯度下降、随机梯度下降、较大或较小的学习率、残差 Dropout（Srivastava et al. (2014)）、权重衰减（Loshchilov & Hutter (2017)）以及梯度噪声（Neelakantan et al. (2015)）。结果如图 2（左）所示。 我们发现，添加权重衰减对数据效率有非常大的影响，与大多数其他干预措施相比，它将所需的样本数量减少了一半以上。我们发现，向网络初始化值进行权重衰减也是有效的，但其效果略逊于向原点进行权重衰减。这让我们相信，即“近似为零的权重适合小型算法任务”这一先验假设，解释了权重衰减优越性能的一部分，但并非全部。在优化过程中添加一些噪声（例如，使用小批量产生的梯度噪声，或在计算梯度前后对权重施加的高斯噪声）有利于泛化，这与这样的观点一致：此类噪声可能促使优化过程找到更平坦的极小值，从而获得更好的泛化能力。我们发现，为了实现泛化，学习率必须在一个相对狭窄的窗口内进行调整（在 1 个数量级以内）。

> 3.4 QUALITATIVE VISUALIZATION OF EMBEDDINGS In order to gain some insight into networks that generalize, we visualized the matrix of the output layer for the case of modular addition and S5. In Figure 3 we show t-SNE plots of the row vectors. For some networks we find clear reflections of the structure of the underlying mathematical objects in the plots. For example the circular topology of modular addition is shown with a ‘number line’ formed by adding 8 to each element. The structure is more apparent in networks that were optimized with weight decay.

3.4 嵌入的定性可视化 为了深入了解具有泛化能力的网络，我们对模加法和S5任务中输出层的矩阵进行了可视化。图3展示了行向量的t-SNE图。对于某些网络，我们在图中清晰地看到了底层数学对象结构的反映。例如，模加法的环形拓扑结构通过为每个元素加上8所形成的“数轴”得以体现。在使用权重衰减进行优化的网络中，这种结构更为明显。

> DISCUSSION We have seen that in the datasets we studied, small algorithmic binary operation tables, effects such as double descent or late generalization, and improvements to generalization from interventions like weight decay can be striking. This suggests that these datasets could be a good place to investigate aspects of generalization. For example, we plan to test whether various proposed measures of minima flatness correlate with generalization in our setting. We have also seen that visualizing the embedding spaces of these neural networks can show natural kinds of structure, for example in problems of modular arithmetic the topology of the embeddings tends to be circles or cylinders. We also see that the network tends to idiosyncratically organize the embeddings by various residues. Whilst the properties of these mathematical objects are familiar to us, we speculate that such visualizations could one day be a useful way to gain intuitions about novel mathematical objects. clusters of permutations, and each cluster is a coset of the subgroup ⟨(0, 3)(1, 4), (1, 2)(3, 4)⟩or one of its conjugates. Right. t-SNE projection of the output layer weights from a network trained on modular addition. The lines show the result of adding 8 to each element. The colors show the residue of each element modulo 8. In addition, we document an interesting phenomenon, where the number of optimization steps needed to reach a given level of performance increases quickly as we reduce the size of the training dataset. Since this represents a way trade compute for performance on smaller amounts of data, it would be useful to investigate in future work whether the effect is also present for other datasets.

讨论 我们发现，在所研究的数据集中，小型算法二元运算表、双重下降或晚期泛化等效应，以及权重衰减等干预措施对泛化能力的提升，均可能十分显著。这表明这些数据集可能是研究泛化特性的良好场所。例如，我们计划测试在我们设定的场景中，各种关于极小值平坦度的提议度量是否与泛化能力相关。 我们还发现，可视化这些神经网络的嵌入空间可以揭示出自然的结构类型，例如在模算术问题中，嵌入的拓扑结构往往呈现为圆或圆柱。我们还观察到，网络倾向于以各种余数为依据，以独特的方式组织嵌入。尽管这些数学对象的性质对我们而言并不陌生，但我们推测，此类可视化方法未来或许能成为获取关于新颖数学对象直觉的一种有用途径。 置换的簇，且每个簇都是子群⟨(0, 3)(1, 4), (1, 2)(3, 4)⟩或其共轭子群的一个陪集。右图：在模加法上训练的神经网络输出层权重的t-SNE投影。线条展示了将每个元素加8的结果。颜色展示了每个元素模8的余数。 此外，我们记录了一个有趣的现象：随着训练数据集规模的减小，达到给定性能水平所需的优化步数迅速增加。由于这代表了一种用计算资源换取在较少数据上性能的方式，未来工作有必要调查该效应是否也存在于其他数据集中。

— 参考文献（原文保留） —

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine learning practice and the bias-variance trade-off. arXiv preprint arXiv:1812.11118, 2018. St´ephane d’ Ascoli, Levent Sagun, and Giulio Biroli. Triple descent and the two kinds of overfitting: Where & why do they appear? arXiv preprint arXiv:2006.03509, 2020. Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. arXiv preprint arXiv:1807.03819, 2018. Alex Graves. Adaptive computation time for recurrent neural networks. arXiv preprint Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. arXiv preprint Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory. Advances in neural information processing systems, 28: 1828–1836, 2015. Sepp Hochreiter and J¨urgen Schmidhuber. Flat minima. Neural computation, 9(1):1–42, 1997. Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. arXiv preprint arXiv:1912.02178, 2019. Łukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. arXiv preprint arXiv:1511.08228, 2015. Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. CoRR, abs/1609.04836, 2016. URL http://arxiv.org/abs/1609.04836. Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. arXiv preprint

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. arXiv preprint arXiv:1912.02292, 2019. Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. arXiv preprint Scott Reed and Nando De Freitas. Neural programmer-interpreters. arXiv preprint arXiv:1511.06279, 2015. David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. arXiv preprint arXiv:1904.01557, 2019. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. arXiv preprint arXiv:1706.03762, 2017. Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. arXiv preprint Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merri¨enboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. arXiv preprint arXiv:1502.05698, 2015. Wojciech Zaremba and Ilya Sutskever. Reinforcement learning neural turing machines-revised. arXiv preprint arXiv:1505.00521, 2015. Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding

deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530, 2016. A APPENDIX A.1 ADDITIONAL EXPERIMENTAL DETAILS A.1.1 BINARY OPERATIONS The following are the binary operations that we have tried (for a prime number p = 97): x ◦ y = x + y (mod p) for 0 ⩽x, y < p x ◦ y = x −y (mod p) for 0 ⩽x, y < p x ◦ y = x/y (mod p) for 0 ⩽x < p, 0 < y < p x ◦ y = [x/y (mod p) if y is odd, otherwise x −y (mod p)] for 0 ⩽x, y < p x ◦ y = x2 + y2 (mod p) for 0 ⩽x, y < p x ◦ y = x2 + xy + y2 (mod p) for 0 ⩽x, y < p x ◦ y = x2 + xy + y2 + x (mod p) for 0 ⩽x, y < p x ◦ y = x3 + xy (mod p) for 0 ⩽x, y < p x ◦ y = x3 + xy2 + y (mod p) for 0 ⩽x, y < p x ◦ y = x · y for x, y ∈S5 x ◦ y = x · y · x−1 for x, y ∈S5 x ◦ y = x · y · x for x, y ∈S5 For each binary operation we constructed a dataset of equations of the form ⟨x⟩⟨op⟩⟨y⟩⟨=⟩⟨x ◦ y⟩, where ⟨a⟩stands for the token corresponding to element a. For each training run, we chose a fraction of all available equations at random and declared them to be the training set, with the rest of equations being the validation set. A.1.2 MODEL AND OPTIMIZATION We trained a standard decoder-only transformer Vaswani et al. (2017) with causal attention masking, and calculated loss and accuracy only on the answer part of the equation. For all experiments we used a transformer with 2 layers, width 128, and 4 attention heads, with a total of about 4 · 105 non-embedding parameters. We have tuned optimization hyperparameters by running experiments on modular addition and product in S5. For final configuration of hyperparameters we have chosen a balance of performance we saw on S5 and simplicity (for example we chose not to anneal the learning rate for the experiments in the paper even though it performed better in some situations). For most experiments we used AdamW optimizer with learning rate 10−3, weight decay 1, β1 = 0.9, β2 = 0.98, linear learning rate warmup over the first 10 updates, minibatch size 512 or half of training dataset size (whichever was smaller) and optimization budget of 105 gradient updates.

In section 3.3 we have also tried the following variants (listed in the reading order for Figure 2 left): · Adam optimizer with full batch (i.e. exact gradient of the loss on the whole training dataset) · Adam optimizer · Adam optimizer with full batch and Gaussian noise added to the update direction for each parameter ($W \leftarrow W + lr \cdot (\Delta W + \epsilon)$, where ϵ is sampled from unit Gaussian, ΔW is the standard Adam weight update, and lr is the learning rate) · Adam optimizer on model with residual dropout 0.1 added · AdamW optimizer with weight decay 1 (default setting in most other experiments) · AdamW optimizer with weight decay 1 towards the initialization instead of the origin · Adam optimizer with learning rate 3 · 10−4 · Adam optimizer with learning rate 3 · 10−3 · Adam optimizer on model with Gaussian weight noise of standard deviation 0.01 (i.e. each parameter W replaced by $W + 0.01 \cdot \epsilon$ in the model, with ϵ sampled from unit Gaussian). For experiments reported in Section 3.1.1 we increased the optimization budget to 5 · 105 optimization steps in order to capture the increase of time to perfect generalization better. For the experiments reported in Section 3.1 we increased the optimization budget to 106, and used Adam optimizer with no weight decay, for emphasizing how late into the optimization process the generalization can begin. We've repeated each experiment for each dataset size with 3 random seeds, with the exception of experiments in section 3.1.1, where we've aggregated results over 7 random seeds.

A.2 ADDITIONAL FIGURES In Figure 4 we show the loss curves that correspond to the accuracy curves in Figure 1. In Figure 5 we show an example of a binary operation table that the network can actually solve. increases from 102 to about 105 optimization steps before it begins a second descent. A.3 RELATED WORK In this paper we study training and generalization dynamics on small simple algorithmic datasets. In the past, algorithmic datasets have been used to probe the capability of neural networks to perform symbolic and algorithmic reasoning. For example the tasks of copying, reversing, and sorting randomly generated sequences, and performing arithmetic operations of multi-digit numbers, have been used as standard benchmarks for sequence-to-sequence models Graves et al. (2014), Weston et al. (2014) Kaiser & Sutskever (2015) Reed & De Freitas (2015), Grefenstette et al. (2015), Zaremba & Sutskever (2015), Graves (2016), Dehghani et al. (2018). Typically in these works however the emphasis is on the performance in the unlimited data regime, with generalization often studied with respect to input sequence length. Some papers study the sample complexity on algorithmic tasks Reed & De Freitas (2015), but mostly focus on the impact of architectural choices. In contrast we study the phenomenon of generalization in data-limited regime, with an emphasis on phenomena that we believe to be architecture-agnostic. fill in. Each symbol is represented as a letter in English, Hebrew, or Greek alphabet for reader's convenience. We invite the reader to guess which operation is represented here.

Algorithmically generated reasoning datasets like bAbI Weston et al. (2015) encourage work on studying generalization in data-limited regime. Most results on such datasets however focus on a point estimate of performance of a particular architecture or training technique, whereas our main interest is in pointing out the change in generalization past the point where a particular architecture can memorize the training data completely. Neelakantan et al. (2015) has a "grok-like" learning curve on an algorithmic task, but it is related to optimization difficulty, whereas our phenomenon is specifically about generalization. In Saxton et al. (2019) they study generalization on procedurally generated math problems such as arithmetic and differentiation, but for the most part these tasks are more involved than the simple binary op problems we have studied and as such do not lend themselves to observing the kinds of phenomena we describe in this paper, since they would require an extremely large number of samples to master. In Jiang et al. (2019) they studied a large number of generalization or complexity measures on convolutional neural networks to see which, if any, are predictive of generalization performance. They find that flatness based measures that aim to quantify the sensitivity of the trained neural network to parameter perturbations are the most predictive. We conjectured that the grokking phenomena we report in this work may be due to the noise from SGD driving the optimization to flatter/simpler solutions that generalize better and hope to investigate in future work whether any of these measures are predictive of grokking.

Zhang et al. (2016) finds that neural networks of sizes typically used in deep learning can interpolate arbitrary training data, and yet generalize when trained with semantically meaningful labels using appropriate optimization procedures. Our work shows a related phenomenon where neural networks can interpolate a small algorithmic training dataset without generalizing, but start generalizing when trained with SGD for longer. Nakkiran et al. (2019); Belkin et al. (2018) focus on the phenomenon of double descent in loss as a function of model and optimization procedure capacity. They find that the classical U-shaped validation loss curve is followed in some settings (including neural network training) by a second descent of loss that starts around the minimal capacity that is needed to interpolate any training data. We observe a second descent in validation loss (though not accuracy) as a function of the amount of training in some of our experiments, and it happens past the point of interpolating the training data. We believe that the phenomenon we describe might be distinct from the double descent phenomena described in Nakkiran et al. (2019); Belkin et al. (2018) because we observe the second descent in loss far past the first time the training loss becomes very small (tens of thousands of epochs in some of our experiments), and we don't observe a non-monotonic behavior of accuracy. The setting of small algorithmic datasets that we study also provides a smaller, more tractable playground for studying subtle generalization phenomena than natural datasets studied in Nakkiran et al. (2019).

A.4 GENERALIZATION WITH MEMORIZING SEVERAL OUTLIERS ples with random labels) into the training data. Small number of outliers doesn't noticeably impact generalization performance, but a large number hinders it significantly. In this section we show data efficiency curves for a modified version of a binary op by introducing k outliers to the training dataset. More precisely, at the beginning of the experiment we randomly sample k equations from the training set and replace their answers with answers to other k equations randomly sampled from the training data. The rest of the equations in the training data and all the equations in the validation data are kept as before. In this situation one could imagine one of the following scenarios unfolding. If the model class of neural networks optimized and regularized as before was not large enough to interpolate such "noisy" dataset, one could imagine the procedure converging to a solution that generalizes well, but denoises the training data (i.e. predicts $c = a \circ b$ as an answer even for the outlier equations $a, b \rightarrow c'$ with $c' = c$). On the other extreme it could be that the optimization procedure can find networks that interpolate the data, but the resulting models don't generalize, because they are forced to represent a considerably more complicated function than before (a simple function + k exceptions encoded in the training data). In our experiments we find that the first option doesn't happen - all experiments reach 100% training accuracy at some point, and this point is not considerably affected by changing the number of outliers k. The second phenomenon happens in a range of training data percentages and number of outliers k - increasing k decreases the range of training data percentages for which the optimization procedure

converges to models that generalize. However the effect of introducing a small number of outliers (up to 1000) is not very pronounced - see Figure 6. We interpret this as additional evidence that the capacity of the network and optimization procedure is well beyond the capacity needed for memorizing all the labels on the training data, and that generalization happening at all requires a non-trivial explanation. A.5 GENERALIZATION MEASURES We believe it is useful to explore how predictive common generalization measures are of generalization on small algorithmic datasets presented in this paper. In a preliminary investigation we found that sharpness Hochreiter & Schmidhuber (1997) of the minimum found by a trained network measure seems to be predictive of generalization on one of these datasets. We trained multiple networks with different initialization seeds for a fixed number a steps on the S5 composition objective, until approximately half of them achieved high validation accuracy. We then used the method described in Keskar et al. (2016) to calculate the sharpness approximation value, ϕ. We found that the validation accuracy and the ϕ score across our trained networks had Spearman correlation coefficient of -0.79548 (significant with $p < 0.000014$). This is suggestive that grokking may only happen after the network's parameters are in flatter regions of the loss landscape. It would be valuable for future work to explore this hypothesis, as well as test other generalization measures.

regions of the loss landscape.

P002 · 直接偏好优化

Direct Preference Optimization (DPO)

作者：Rafailov, Rafael, Sharma, Archit, Mitchell, Eric, Ermon, Stefano, Manning, Christopher D., Finn, Chelsea｜2023/05/29｜阶段：S4 S7｜原文：https://arxiv.org/abs/2305.18290

> 对照说明：灰底为英文原文段落，下方为中文译文；参考文献部分保留英文原文。翻译由本地 qwen3.8:27b 逐段生成，供中文读者参考。

Direct Preference Optimization: Your Language Model is Secretly a Reward Model Rafael Rafailov∗ † Archit Sharma∗ † Eric Mitchell∗ † Stefano Ermon † ‡ Christopher D. Manning † Chelsea Finn † † Stanford University ‡ CZ Biohub {rafailov,architsh,eric.mitchell}@cs.stanford.edu

直接偏好优化：你的语言模型其实是一个隐藏的奖励模型 拉斐尔·拉菲洛夫*† 阿奇特·夏尔马*† 埃里克·米切尔*† 斯特凡诺·埃尔蒙 † ‡ 克里斯托弗·D·曼宁 † 切尔西·芬恩 † †斯坦福大学 ‡CZ生物枢纽 {rafailov,architsh,eric.mitchell} @cs.stanford.edu

Abstract While large-scale unsupervised language models (LMs) learn broad world knowl- edge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these prefer- ences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call Direct Prefer- ence Optimization (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sen- timent of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

摘要 尽管大规模无监督语言模型（LMs）能够学习广泛的世界知识和部分推理技能，但由于其训练过程完全无监督，精确控制其行为仍然困难。现有的获得这种可控性的方法通过收集人类对模型生成结果相对质量的评价标签，并对无监督语言模型进行微调以使其与这些偏好对齐，通常采用基于人类反馈的强化学习（RLHF）。然而，RLHF 是一个复杂且往往不稳定的过程，首先需要拟合一个反映人类偏好的奖励模型，然后使用强化学习对大型无监督语言模型进行微调，以最大化该估计奖励，同时避免偏离原始模型过远。在本文中，我们引入了 RLHF 中奖励模型的一种新参数化方法，该方法能够以闭式解提取相应的最优策略，从而允许我们仅通过一个简单的分类损失来解决标准的 RLHF 问题。我们提出的算法称为直接偏好优化（DPO），具有稳定性、高性能和计算轻量级的特点，消除了微调期间从语言模型中采样或进行大量超参数调整的需要。我们的实验表明，DPO 能够微调语言模型以与人类偏好对齐，其效果与现有方法相当或更优。值得注意的是，使用 DPO 进行微调在控制生成内容的情感方面优于基于 PPO 的 RLHF，并且在摘要生成和单轮对话中匹配或提升了响应质量，同时其实现和训练过程显著更简单。

> Introduction Large unsupervised language models (LMs) trained on very large datasets acquire surprising capabili- ties [11, 7, 42, 8]. However, these models are trained on data generated by humans with a wide variety of goals, priorities, and skillsets. Some of these goals and skillsets may not be desirable to imitate; for example, while we may want our AI coding assistant to understand common programming mistakes in order to correct them, nevertheless, when generating code, we would like to bias our model toward the (potentially rare) high-quality coding ability present in its training data. Similarly, we might want our language model to be aware of a common misconception believed by 50% of people, but we certainly do not want the model to claim this misconception to be true in 50% of queries about it! In other words, selecting the model’ s desired responses and behavior from its very wide knowledge and abilities is crucial to building AI systems that are safe, performant, and controllable [28]. While existing methods typically steer LMs to match human preferences using reinforcement learning (RL), ∗Equal contribution; more junior authors listed earlier. 37th Conference on Neural Information Processing Systems (NeurIPS 2023). for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, fitting an implicit reward model whose corresponding optimal policy can be extracted in closed form.

引言 在超大规模数据集上训练的大型无监督语言模型（LMs）展现出令人惊讶的能力 [11, 7, 42, 8]。然而，这些模型是在由人类生成的数据上训练的，这些数据涵盖了各种各样的目标、优先级和技能。其中一些目标和技能可能并不值得模仿；例如，虽然我们希望AI编程助手能够理解常见的编程错误以便进行纠正，但在生成代码时，我们仍希望将模型偏向于其训练数据中存在的（可能罕见的）高质量编码能力。同样，我们可能希望语言模型意识到50%的人所持有的常见误解，但我们绝对不希望模型在50%的相关查询中声称这种误解是正确的！换言之，从其广泛的知识与能力中筛选出模型期望的响应和行为，对于构建安全、高效且可控的AI系统至关重要 [28]。尽管现有方法通常使用强化学习（RL）来引导语言模型以匹配人类偏好，即首先针对包含提示词以及人类对成对响应偏好的数据集拟合一个奖励模型，然后使用强化学习寻找最大化所学习奖励的策略。相比之下，DPO通过一个简单的分类目标直接优化最能满足偏好的策略，拟合一个隐式奖励模型，其对应的最优策略可以以闭式形式提取。 ∗同等贡献；资历较浅的作者列在前面。 第37届神经信息处理系统大会（NeurIPS 2023）。

> we will show that the RL-based objective used by existing methods can be optimized exactly with a simple binary cross-entropy objective, greatly simplifying the preference learning pipeline. At a high level, existing methods instill the desired behaviors into a language model using curated sets of human preferences representing the types of behaviors that humans find safe and helpful. This preference learning stage occurs after an initial stage of large-scale unsupervised pre-training on a large text dataset. While the most straightforward approach to preference learning is supervised fine-tuning on human demonstrations of high quality responses, the most successful class of methods is reinforcement learning from human (or AI) feedback (RLHF/RLAIF; [12, 2]). RLHF methods fit a reward model to a dataset of human preferences and then use RL to optimize a language model policy to produce responses assigned high reward without drifting excessively far from the original model. While RLHF produces models with impressive conversational and coding abilities, the RLHF pipeline is considerably more complex than supervised learning, involving training multiple LMs and sampling from the LM policy in the loop of training, incurring significant computational costs. In this paper, we show how to directly optimize a language model to adhere to human preferences, without explicit reward modeling or reinforcement learning. We propose Direct Preference Optimiza- tion (DPO), an algorithm that implicitly optimizes the same objective as existing RLHF algorithms (reward maximization with a KL-divergence constraint) but is simple to implement and straight- forward to train. Intuitively, the DPO update increases the relative log probability of preferred to dispreferred responses, but it incorporates a dynamic, per-example importance weight that prevents the model degeneration that we find occurs with a naive probability ratio objective. Like existing algorithms, DPO relies on a theoretical preference model (such as the Bradley-Terry model; [5]) that measures how well a given reward function aligns with empirical preference data. However, while existing methods use the preference model to define a preference loss to train a reward model and then train a policy that optimizes the learned reward model, DPO uses a change of variables to define the preference loss as a function of the policy directly. Given a dataset of human preferences over model responses, DPO can therefore optimize a policy using a simple binary cross entropy objective, producing the optimal policy to an implicit reward function fit to the preference data.

我们将证明，现有方法所使用的基于强化学习的目标函数可以通过一个简单的二元交叉熵目标函数进行精确优化，从而极大地简化偏好学习流程。 从宏观层面来看，现有方法利用精心策划的人类偏好集合，将期望的行为注入语言模型，这些偏好代表了人类认为安全且有帮助的行为类型。这一偏好学习阶段发生在针对大型文本数据集进行的大规模无监督预训练的初始阶段之后。虽然偏好学习最直接的方法是在高质量响应的人类演示上进行监督微调，但最成功的算法类别是基于人类（或人工智能）反馈的强化学习（RLHF/RLAIF；[12, 2]）。RLHF方法首先针对人类偏好数据集拟合一个奖励模型，然后使用强化学习来优化语言模型策略，使其生成被赋予高奖励的响应，同时避免过度偏离原始模型。尽管RLHF能够生成具有令人印象深刻的对话和编程能力的模型，但RLHF流程比监督学习复杂得多，涉及训练多个语言模型，并在训练循环中对语言模型策略进行采样，从而产生显著的算力成本。 在本文中，我们展示了如何直接优化语言模型以遵循人类偏好，而无需显式的奖励建模或强化学习。我们提出了直接偏好优化（DPO），这是一种算法，它隐式地优化与现有RLHF算法相同的目标函数（带有KL散度约束的奖励最大化），但实现简单且训练直观。直观地说，DPO更新增加了偏好响应相对于非偏好响应的相对对数概率，但它纳入了动态的、逐样本的重要性权重，以防止我们发现的由朴素概率比目标函数导致的模型退化。与现有算法一样，DPO依赖于一个理论偏好模型（例如Bradley-Terry模型；[5]），该模型衡量给定的奖励函数与经验偏好数据的匹配程度。然而，虽然现有方法使用偏好模型来定义偏好损失以训练奖励模型，然后训练一个优化该学习到的奖励模型的策略，但DPO通过变量变换将偏好损失直接定义为策略的函数。因此，给定一个关于模型响应的人类偏好数据集，DPO可以使用简单的二元交叉熵目标函数来优化策略，从而生成针对拟合到偏好数据的隐式奖励函数的最优策略。

> Our main contribution is Direct Preference Optimization (DPO), a simple RL-free algorithm for training language models from preferences. Our experiments show that DPO is at least as effective as existing methods, including PPO-based RLHF, for learning from preferences in tasks such as sentiment modulation, summarization, and dialogue, using language models with up to 6B parameters.

我们的主要贡献是直接偏好优化（DPO），这是一种用于从偏好中训练语言模型的简单无强化学习算法。我们的实验表明，在使用最多60亿参数的语言模型进行情感调节、摘要和对话等任务的偏好学习时，DPO至少与包括基于PPO的RLHF在内的现有方法一样有效。

> Related Work Self-supervised language models of increasing scale learn to complete some tasks zero-shot [33] or with few-shot prompts [6, 27, 11]. However, their performance on downstream tasks and alignment with user intent can be significantly improved by fine-tuning on datasets of instructions and human- written completions [25, 38, 13, 41]. This ‘instruction-tuning’ procedure enables LLMs to generalize to instructions outside of the instruction-tuning set and generally increase their usability [13]. Despite the success of instruction tuning, relative human judgments of response quality are often easier to collect than expert demonstrations, and thus subsequent works have fine-tuned LLMs with datasets of human preferences, improving proficiency in translation [20], summarization [40, 51], story-telling [51], and instruction-following [28, 34]. These methods first optimize a neural network reward function for compatibility with the dataset of preferences under a preference model such as the Bradley-Terry model [5], then fine-tune a language model to maximize the given reward using reinforcement learning algorithms, commonly REINFORCE [47], proximal policy optimization (PPO; [39]), or variants [34]. A closely-related line of work leverages LLMs fine-tuned for instruction following with human feedback to generate additional synthetic preference data for targeted attributes such as safety or harmlessness [2], using only weak supervision from humans in the form of a text rubric for the LLM’s annotations. These methods represent a convergence of two bodies of work: one body of work on training language models with reinforcement learning for a variety of objectives [35, 29, 48] and another body of work on general methods for learning from human preferences [12, 21]. Despite the appeal of using relative human preferences, fine-tuning large language models with reinforcement learning remains a major practical challenge; this work provides a theoretically-justified approach to optimizing relative preferences without RL.

相关工作 规模不断扩大的自监督语言模型能够零样本[33]或通过少样本提示[6, 27, 11]完成某些任务。然而，通过在指令和人工撰写的补全数据集上进行微调，可以显著提升其在下游任务上的表现以及与用户意图的对齐程度[25, 38, 13, 41]。这种“指令微调”过程使大语言模型能够泛化到指令微调集之外的指令，并通常提高其可用性[13]。尽管指令微调取得了成功，但相对于专家演示，收集响应质量的相对人类判断往往更容易，因此后续工作利用人类偏好数据集对大语言模型进行微调，从而提高了翻译[20]、摘要[40, 51]、故事讲述[51]以及指令遵循[28, 34]方面的能力。这些方法首先基于Bradley-Terry模型[5]等偏好模型，优化一个神经网络奖励函数以使其与偏好数据集兼容，然后使用强化学习算法（通常是REINFORCE[47]、近端策略优化（PPO；[39]）或其变体[34]）微调语言模型以最大化给定的奖励。一条密切相关的工作路线利用针对指令遵循和人类反馈进行微调的大语言模型，为安全性或无害性等特定属性生成额外的合成偏好数据[2]，仅使用人类提供的弱监督，即用于大语言模型标注的文本评分标准。这些方法代表了两个研究领域的融合：一个是关于使用强化学习针对多种目标训练语言模型的研究[35, 29, 48]，另一个是关于从人类偏好中学习的一般方法的研究[12, 21]。尽管使用相对人类偏好具有吸引力，但使用强化学习微调大语言模型仍然是一个主要的实际挑战；本工作提供了一种理论上合理的优化相对偏好的方法，而无需使用强化学习。

> Outside of the context of language, learning policies from preferences has been studied in both bandit and reinforcement learning settings, and several approaches have been proposed. Contextual bandit learning using preferences or rankings of actions, rather than rewards, is known as a contextual dueling bandit (CDB; [50, 14]). In the absence of absolute rewards, theoretical analysis of CDBs substitutes the notion of an optimal policy with a von Neumann winner, a policy whose expected win rate against any other policy is at least 50% [14]. However, in the CDB setting, preference labels are given online, while in learning from human preferences, we typically learn from a fixed batch of offline preference-annotated action pairs [49]. Similarly, preference-based RL (PbRL) learns from binary preferences generated by an unknown ‘scoring’ function rather than rewards [9, 37]. Various algorithms for PbRL exist, including methods that can reuse off-policy preference data, but generally involve first explicitly estimating the latent scoring function (i.e. the reward model) and subsequently optimizing it [16, 9, 12, 36, 21]. We instead present a single stage policy learning approach that directly optimizes a policy to satisfy preferences. Preliminaries We review the RLHF pipeline in Ziegler et al. (and later [40, 1, 28]). It usually includes three phases: 1) supervised fine-tuning (SFT); 2) preference sampling and reward learning and 3) RL optimization.

在语言模型语境之外，从偏好中学习策略已在多臂赌博机（bandit）和强化学习环境中得到研究，并提出了多种方法。使用动作的偏好或排序而非奖励进行上下文赌博机学习被称为上下文对决赌博机（Contextual Dueling Bandit, CDB; [50, 14]）。在缺乏绝对奖励的情况下，CDB的理论分析用冯·诺依曼胜者（von Neumann winner）的概念取代了最优策略的概念，即一个针对任何其他策略的期望胜率至少为50%的策略 [14]。然而，在CDB设置中，偏好标签是在线给出的，而在从人类偏好中学习的场景中，我们通常是从一批固定的离线偏好标注动作对中学习的 [49]。类似地，基于偏好的强化学习（Preference-based RL, PbRL）是从由未知“评分”函数生成的二元偏好中学习的，而不是从奖励中学习的 [9, 37]。存在多种PbRL算法，包括能够重用离策略偏好数据的方法，但通常涉及首先显式估计潜在的评分函数（即奖励模型），然后对其进行优化 [16, 9, 12, 36, 21]。相反，我们提出了一种单阶段策略学习方法，直接优化策略以满足偏好。 预备知识 我们回顾Ziegler等人（以及后续工作[40, 1, 28]）中的RLHF流程。它通常包括三个阶段：1）监督微调（SFT）；2）偏好采样和奖励学习；以及3）强化学习优化。

SFT: RLHF typically begins by fine-tuning a pre-trained LM with supervised learning on high-quality data for the downstream task(s) of interest (dialogue, summarization, etc.), to obtain a model πSFT. Reward Modelling Phase: In the second phase the SFT model is prompted with prompts x to produce pairs of answers (y1, y2) ∼πSFT(y | x). These are then presented to human labelers who express preferences for one answer, denoted as yw ≻yl | x where yw and yl denotes the preferred and dispreferred completion amongst (y1, y2) respectively. The preferences are assumed to be generated by some latent reward model r∗(y, x), which we do not have access to. There are a number of approaches used to model preferences, the Bradley-Terry (BT) [5] model being a popular choice (although more general Plackett-Luce ranking models [32, 23] are also compatible with the framework if we have access to several ranked answers). The BT model stipulates that the human preference distribution p∗can be written as: p∗(y1 ≻y2 | x) = exp (r∗(x, y1)) exp (r∗(x, y1)) + exp (r∗(x, y2)). (1) i=1 sampled from p∗, we can parametrize a reward model rϕ(x, y) and estimate the parameters via maximum likelihood. Framing the problem as a binary classification we have the negative log-likelihood loss: Assuming access to a static dataset of comparisons D = LR(rϕ, D) = −E(x,yw,yl)∼D where σ is the logistic function. In the context of LMs, the network rϕ(x, y) is often initialized from the SFT model πSFT(y | x) with the addition of a linear layer on top of the final transformer layer that produces a single scalar prediction for the reward value [51]. To ensure a reward function with lower variance, prior works normalize the rewards, such that Ex,y∼D [rϕ(x, y)] = 0 for all x.

SFT：RLHF 通常首先通过监督学习，利用针对目标下游任务（如对话、摘要等）的高质量数据对预训练语言模型进行微调，从而获得模型 πSFT。 奖励建模阶段：在第二阶段，向 SFT 模型输入提示 x，使其生成答案对 (y1, y2) ∼πSFT(y | x)。随后，这些答案对呈现给人类标注者，由其表达偏好，记为 yw ≻yl | x，其中 yw 和 yl 分别表示 (y1, y2) 中偏好和非偏好的补全。假设这些偏好由某个潜在的奖励模型 r∗(y, x) 生成，而该模型是不可访问的。用于建模偏好有多种方法，Bradley-Terry (BT) [5] 模型是一个流行的选择（尽管如果我们能访问多个排序答案，更通用的 Plackett-Luce 排序模型 [32, 23] 也兼容该框架）。BT 模型规定，人类偏好分布 p∗ 可以写为： p∗(y1 ≻y2 | x) = exp (r∗(x, y1)) / (exp (r∗(x, y1)) + exp (r∗(x, y2))). (1) 假设可以访问一个静态的比较数据集 D = {(x, yw, yl)}i=1...N，其中 (x, yw, yl) 是从 p∗ 中采样的，我们可以参数化一个奖励模型 rϕ(x, y)，并通过最大似然估计参数。将问题框架化为二分类问题，我们有负对数似然损失： LR(rϕ, D) = −E(x,yw,yl)∼D [log σ(rϕ(x, yw) − rϕ(x, yl))] 其中 σ 是逻辑函数。在语言模型的背景下，网络 rϕ(x, y) 通常从 SFT 模型 πSFT(y | x) 初始化，并在最后一个 Transformer 层之上添加一个线性层，以产生单个标量预测作为奖励值 [51]。为了确保奖励函数具有较低的方差，先前的工作对奖励进行归一化，使得对于所有 x，Ex,y∼D [rϕ(x, y)] = 0。

RL Fine-Tuning Phase: During the RL phase, the learned reward function is used to provide feedback to the language model. Following prior works [17, 18], the optimization is formulated as πθ Ex∼D,y∼πθ(y|x) ⊠max rϕ(x, y) N x(i), y(i) w , y(i) ⊠ ⊠⊠log σ(rϕ(x, yw) −rϕ(x, yl)) (2) −βDKL ⊠⊠⊠πθ(y | x) || πref(y | x) , (3) where β is a parameter controlling the deviation from the base reference policy πref, namely the ini- tial SFT model πSFT. In practice, the language model policy πθ is also initialized to πSFT. The added constraint is important, as it prevents the model from deviating too far from the distri- bution on which the reward model is accurate, as well as maintaining the generation diversity and preventing mode-collapse to single high-reward answers. Due to the discrete nature of lan- guage generation, this objective is not differentiable and is typically optimized with reinforce- ment learning. The standard approach [51, 40, 1, 28] has been to construct the reward function r(x, y) = rϕ(x, y) −β(log πθ(y | x) −log πref(y | x)), and maximize using PPO [39]. Direct Preference Optimization Motivated by the challenges of applying reinforcement learning algorithms on large-scale problems such as fine-tuning language models, our goal is to derive a simple approach for policy optimization using preferences directly. Unlike prior RLHF methods, which learn a reward and then optimize it via RL, our approach leverages a particular choice of reward model parameterization that enables extraction of its optimal policy in closed form, without an RL training loop. As we will describe next in detail, our key insight is to leverage an analytical mapping from reward functions to optimal policies, which enables us to transform a loss function over reward functions into a loss function over policies. This change-of-variables approach avoids fitting an explicit, standalone reward model, while still optimizing under existing models of human preferences, such as the Bradley-Terry model. In essence, the policy network represents both the language model and the (implicit) reward.

强化学习微调阶段：在强化学习阶段，学习到的奖励函数被用于向语言模型提供反馈。遵循先前工作 [17, 18]，优化问题被表述为： πθ Ex∼D,y∼πθ(y|x) [max rϕ(x, y) N x(i), y(i) w , y(i) l log σ(rϕ(x, yw) −rϕ(x, yl)) (2) −βDKL (πθ(y | x) || πref(y | x)) , (3)] 其中 β 是控制偏离基础参考策略 πref（即初始 SFT 模型 πSFT）程度的参数。在实践中，语言模型策略 πθ 也初始化为 πSFT。添加的约束至关重要，因为它防止模型偏离奖励模型准确的分布过远，同时保持生成的多样性并防止模式坍塌至单一的高奖励答案。由于语言生成的离散性质，该目标函数不可微，通常使用强化学习进行优化。标准方法 [51, 40, 1, 28] 是构建奖励函数 r(x, y) = rϕ(x, y) −β(log πθ(y | x) −log πref(y | x))，并使用 PPO [39] 进行最大化。 直接偏好优化 受在大规模问题（如语言模型微调）上应用强化学习算法所面临挑战的启发，我们的目标是推导一种利用偏好直接进行策略优化的简单方法。与先前通过强化学习学习奖励并对其进行优化的 RLHF 方法不同，我们的方法利用了一种特定的奖励模型参数化选择，从而能够以闭式形式提取其最优策略，而无需强化学习训练循环。正如接下来将详细描述的，我们的关键洞察是利用从奖励函数到最优策略的解析映射，这使得我们将关于奖励函数的损失函数转换为关于策略的损失函数成为可能。这种变量替换方法避免了拟合显式的独立奖励模型，同时仍在现有的人类偏好模型（如 Bradley-Terry 模型）下进行优化。本质上，策略网络既代表了语言模型，也代表了（隐式的）奖励。

> Deriving the DPO objective. We start with the same RL objective as prior work, Eq. 3, under a general reward function r. Following prior work [31, 30, 19, 15], it is straightforward to show that the optimal solution to the KL-constrained reward maximization objective in Eq. 3 takes the form: πr(y | x) = Z(x)πref(y | x) exp ⊠β r(x, y) where Z(x) = P y πref(y | x) exp is the partition function. See Appendix A.1 for a complete derivation. Even if we use the MLE estimate rϕ of the ground-truth reward function r∗, it is still expensive to estimate the partition function Z(x) [19, 15], which makes this representation hard to utilize in practice. However, we can rearrange Eq. 4 to express the reward function in terms of its corresponding optimal policy πr, the reference policy πref, and the unknown partition function Z(・). Specifically, we first take the logarithm of both sides of Eq. 4 and then with some algebra we obtain: r(x, y) = β log πr(y | x) We can apply this reparameterization to the ground-truth reward r∗and corresponding optimal model π∗. Fortunately, the Bradley-Terry model depends only on the difference of rewards between two completions, i.e., p∗(y1 ≻y2 | x) = σ(r∗(x, y1) −r∗(x, y2)). Substituting the reparameterization in Eq. 5 for r∗(x, y) into the preference model Eq. 1, the partition function cancels, and we can express the human preference probability in terms of only the optimal policy π∗and reference policy πref. Thus, the optimal RLHF policy π∗under the Bradley-Terry model satisfies the preference model:

推导DPO目标函数。我们从先前工作中的相同强化学习目标（公式3）出发，并假设一个通用的奖励函数r。遵循先前工作[31, 30, 19, 15]，可以很容易地证明，公式3中KL约束下的奖励最大化目标的最优解具有以下形式： πr(y | x) = Z(x)πref(y | x) exp(β r(x, y)) 其中Z(x) = Σy πref(y | x) exp(β r(x, y))是配分函数。完整的推导过程见附录A.1。即使我们使用真实奖励函数r∗的最大似然估计rϕ，估计配分函数Z(x)仍然代价高昂[19, 15]，这使得该表示形式在实际应用中难以利用。然而，我们可以重新排列公式4，用其对应的最优策略πr、参考策略πref以及未知的配分函数Z(・)来表示奖励函数。具体而言，我们首先对公式4两边取对数，然后通过一些代数运算得到： r(x, y) = β log(πr(y | x) / πref(y | x)) - log Z(x) 我们可以将这种重参数化应用于真实奖励r∗和对应的最优模型π∗。幸运的是，Bradley-Terry模型仅依赖于两个补全之间奖励的差值，即p∗(y1 ≻ y2 | x) = σ(r∗(x, y1) − r∗(x, y2))。将公式5中的重参数化代入r∗(x, y)到偏好模型公式1中，配分函数相互抵消，我们可以仅用最优策略π∗和参考策略πref来表示人类偏好概率。因此，在Bradley-Terry模型下，最优的RLHF策略π∗满足偏好模型：

> p∗(y1 ≻y2 | x) = 1 + exp ⊠The derivation is in Appendix A.2. While Eq. 6 uses the Bradley-Terry model, we can similarly derive expressions under the more general Plackett-Luce models [32, 23], shown in Appendix A.3. Now that we have the probability of human preference data in terms of the optimal policy rather than the reward model, we can formulate a maximum likelihood objective for a parametrized policy πθ. Analogous to the reward modeling approach (i.e. Eq. 2), our policy objective becomes: ⊠LDPO(πθ; πref) = −E(x,yw,yl)∼D log σ This way, we fit an implicit reward using an alternative parameterization, whose optimal policy is simply πθ. Moreover, since our procedure is equivalent to fitting a reparametrized Bradley-Terry ⊠1 ⊠β r(x, y) , (4) ⊠πref(y | x) + β log Z(x). (5) ⊠(6) β log π∗(y2|x) πref(y2|x) −β log π∗(y1|x) πref(y1|x) ⊠⊠⊠β log πθ(yw | x) πref(yw | x) −β log πθ(yl | x) . (7) πref(yl | x) model, it enjoys certain theoretical properties, such as consistencies under suitable assumption of the preference data distribution [4]. In Section 5, we further discuss theoretical properties of DPO in relation to other works. What does the DPO update do? For a mechanistic understanding of DPO, it is useful to analyze the gradient of the loss function LDPO. The gradient with respect to the parameters θ can be written as: ∇θLDPO(πθ; πref) = ⊠−βE(x,yw,yl)∼D σ(ˆrθ(x, yl) −ˆrθ(x, yw)) | {z } higher weight when reward estimate is wrong

p∗(y1 ≻y2 | x) = 1 + exp ⊠推导过程见附录 A.2。虽然公式 6 使用了 Bradley-Terry 模型，但我们也可以在更一般的 Plackett-Luce 模型 [32, 23] 下类似地推导出表达式，详见附录 A.3。 既然我们已经用最优策略而非奖励模型表示了人类偏好数据的概率，我们就可以为参数化策略 πθ 构建一个最大似然目标。类似于奖励建模方法（即公式 2），我们的策略目标变为：LDPO(πθ; πref) = −E(x,yw,yl)∼D log σ 通过这种方式，我们使用一种替代参数化来拟合隐式奖励，其最优策略即为 πθ。此外，由于我们的过程等价于拟合一个重参数化的 Bradley-Terry β r(x, y) , (4) πref(y | x) + β log Z(x). (5) β log π∗(y2|x) πref(y2|x) −β log π∗(y1|x) πref(y1|x) β log πθ(yw | x) πref(yw | x) −β log πθ(yl | x) . (7) πref(yl | x) 模型，它具备某些理论性质，例如在偏好数据分布满足适当假设时的一致性 [4]。在第 5 节中，我们将进一步讨论 DPO 与其他工作相关的理论性质。 DPO 更新做了什么？为了从机制上理解 DPO，分析损失函数 LDPO 的梯度是有用的。关于参数 θ 的梯度可以写为： ∇θLDPO(πθ; πref) = −βE(x,yw,yl)∼D σ(ˆrθ(x, yl) −ˆrθ(x, yw)) | {z } 当奖励估计错误时权重更高

where ˆrθ(x, y) = β log πθ(y|x) πref(y|x) is the reward implicitly defined by the language model πθ and refer- ence model πref (more in Section 5). Intuitively, the gradient of the loss function LDPO increases the likelihood of the preferred completions yw and decreases the likelihood of dispreferred completions yl. Importantly, the examples are weighed by how much higher the implicit reward model ˆrθ rates the dispreferred completions, scaled by β, i.e, how incorrectly the implicit reward model orders the completions, accounting for the strength of the KL constraint. Our experiments suggest the importance of this weighting, as a naïve version of this method without the weighting coefficient can cause the language model to degenerate (Appendix Table 3). DPO outline. The general DPO pipeline is as follows: 1) Sample completions y1, y2 ∼πref(· | x) for every prompt x, label with human preferences to construct the offline dataset of preferences D = {x(i), y(i) w , yl)(i)}N i=1 and 2) optimize the language model πθ to minimize LDPO for the given πref and D and desired β. In practice, one would like to reuse preference datasets publicly available, rather than generating samples and gathering human preferences. Since the preference datasets are sampled using πSFT, we initialize πref = πSFT whenever available. However, when πSFT is not available, we initialize πref by maximizing likelihood of preferred completions (x, yw), that is, πref = arg maxπ Ex,yw∼D [log π(yw | x)]. This procedure helps mitigate the distribution shift between the true reference distribution which is unavailable, and πref used by DPO. Further details related to the implementation and hyperparameters can be found in Appendix B.

其中 $\hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ 是由语言模型 π_θ 和参考模型 π_{ref} 隐式定义的奖励（详见第5节）。直观上，损失函数 L_{DPO} 的梯度会增加偏好补全 y_w 的可能性，并降低非偏好补全 y_l 的可能性。重要的是，样本的权重取决于隐式奖励模型 $\hat{r}_\theta$ 对非偏好补全的评分高出多少，该值由 β 缩放，即隐式奖励模型对补全排序错误的程度，同时考虑了 KL 约束的强度。我们的实验表明这种加权的重要性，因为如果不使用加权系数，该方法的朴素版本可能导致语言模型退化（附录表3）。 DPO 概述。通用的 DPO 流程如下：1) 对于每个提示 x，从 $\pi_{\text{ref}}(\cdot | x)$ 中采样补全 y_1, y_2，并通过人工偏好进行标注，以构建偏好离线数据集 $D = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}_{i=1}^N$；2) 优化语言模型 π_θ，以在给定 π_{ref}、D 和期望的 β 的情况下最小化 L_{DPO}。在实践中，人们希望重用公开可用的偏好数据集，而不是生成样本并收集人工偏好。由于偏好数据集是使用 π_{SFT} 采样的，因此在可用时，我们初始化 $\pi_{\text{ref}} = \pi_{\text{SFT}}$。然而，当 π_{SFT} 不可用时，我们通过最大化偏好补全 (x, y_w) 的对数似然来初始化 π_{ref}，即 $\pi_{\text{ref}} = \arg \max_{\pi} \mathbb{E}_{x,y_w \sim D} [\log \pi(y_w | x)]$。此过程有助于缓解真实的参考分布（不可用）与 DPO 使用的 π_{ref} 之间的分布偏移。有关实现和超参数的更多细节，请参见附录 B。

Theoretical Analysis of DPO In this section, we give further interpretation of the DPO method, provide theoretical backing, and relate advantages of DPO to issues with actor critic algorithms used for RLHF (such as PPO [39]).

DPO的理论分析 在本节中，我们进一步阐释DPO方法，提供理论依据，并将DPO的优势与用于RLHF的演员-评论家算法（如 PPO [39]）中存在的问题联系起来。

5.1 Your Language Model Is Secretly a Reward Model DPO is able to bypass both fitting an explicit reward and performing RL to learn the policy using a single maximum likelihood objective. Note the optimization objective Eq. 5 is equivalent to a Bradley-Terry model with a reward parameterization r∗(x, y) = β log π∗ θ(y|x) πref(y|x) and we optimize our parametric model πθ, equivalently to the reward model optimization in Eq. 2 under the change of variables. In this section we will build the theory behind this reparameterization, show that it does not constrain the class of learned reward models, and allows for the exact recovery of the optimal policy. We begin with by defining an equivalence relation between reward functions. Definition 1. We say that two reward functions r(x, y) and r′(x, y) are equivalent iff r(x, y) −r′(x, y) = f(x) for some function f. It is easy to see that this is indeed an equivalence relation, which partitions the set of reward functions into classes. We can state the following two lemmas: Lemma 1. Under the Plackett-Luce, and in particular the Bradley-Terry, preference framework, two reward functions from the same class induce the same preference distribution. Lemma 2. Two reward functions from the same equivalence class induce the same optimal policy under the constrained RL problem. The proofs are straightforward and we defer them to Appendix A.5. The first lemma is a well-known under-specification issue with the Plackett-Luce family of models [32]. Due to this under-specification,

5.1 你的语言模型秘密地充当奖励模型 DPO 能够通过单一的最大似然目标，绕过拟合显式奖励模型以及执行强化学习来学习策略的过程。请注意，优化目标公式 5 等价于具有奖励参数化 $r^*(x, y) = \beta \log \frac{\pi^*_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ 的 Bradley-Terry 模型，且我们对参数化模型 π_θ 的优化，在变量替换下等价于公式 2 中的奖励模型优化。在本节中，我们将构建这一重参数化背后的理论，证明它并不限制所学习奖励模型的类别，并且允许精确恢复最优策略。我们首先定义奖励函数之间的等价关系。定义 1. 我们称两个奖励函数 $r(x, y)$ 和 $r'(x, y)$ 是等价的，当且仅当存在某个函数 f，使得 $r(x, y) - r'(x, y) = f(x)$。 容易看出，这确实是一个等价关系，它将奖励函数集划分为不同的类。我们可以陈述以下两个引理：引理 1. 在 Plackett-Luce 偏好框架下，特别是 Bradley-Terry 偏好框架下，来自同一类的两个奖励函数诱导相同的偏好分布。引理 2. 来自同一等价类的两个奖励函数在约束强化学习问题下诱导相同的最优策略。 证明过程很直接，我们将其推迟到附录 A.5。第一个引理是 Plackett-Luce 模型族中众所周知的欠定问题 [32]。由于这种欠定性，

> ∇θ log π(yw | x) | {z } increase likelihood of yw −∇θ log π(yl | x) | {z } decrease likelihood of yl , we usually have to impose additional identifiability constraints to achieve any guarantees on the MLE estimates from Eq. 2 [4]. The second lemma states that all reward functions from the same class yield the same optimal policy, hence for our final objective, we are only interested in recovering an arbitrary reward function from the optimal class. We prove the following Theorem in Appendix A.6: Theorem 1. Under mild assumptions, all reward classes consistent with the Plackett-Luce (and Bradley-Terry in particular) models can be represented with the reparameterization r(x, y) = β log π(y|x) πref(y|x) for some model π(y | x) and a given reference model πref(y | x). Proof Sketch. Consider any reward function r(x, y), which induces a corresponding optimal model πr(y | x), specified by Eq. 4. We will show that a reward function from the equivalence class of r can be represented using the reparameterization given above. We define the projection f as f(r; πref, β)(x, y) = r(x, y) −β log The operator f simply normalizes the reward function with the logarithm of the partition function of πr. Since the added normalization term is only a function of the prefix x, f(r; πref, β)(x, y) is a reward function in the equivalence class of r(x, y). Finally, replacing r with the RHS of Eq. 5 (which holds for any reward function), we have f(r; πref, β)(x, y) = β log πr(y|x)

∇θ log π(yw | x) | {z } 增加 yw 的似然 −∇θ log π(yl | x) | {z } 降低 yl 的似然 ， 我们通常必须施加额外的可辨识性约束，才能对公式 2 [4] 中的极大似然估计量获得任何保证。第二个引理指出，来自同一类的所有奖励函数都产生相同的策略，因此对于我们的最终目标而言，我们只感兴趣于从最优类中恢复任意一个奖励函数。我们在附录 A.6 中证明了以下定理： 定理 1. 在温和的假设下，所有与 Plackett-Luce（特别是 Bradley-Terry）模型一致的奖励类都可以用重参数化表示为 r(x, y) = β log π(y|x) / πref(y|x) 其中 π(y | x) 为某个模型，πref(y | x) 为给定的参考模型。 证明概要。考虑任意奖励函数 r(x, y)，它诱导出一个对应的最优模型 πr(y | x)，由公式 4 指定。我们将证明，r 的等价类中的一个奖励函数可以用上述重参数化表示。我们定义投影 f 为 f(r; πref, β)(x, y) = r(x, y) −β log 算子 f 简单地用 πr 的配分函数的对数来归一化奖励函数。由于添加的归一化项仅是前缀 x 的函数，f(r; πref, β)(x, y) 是 r(x, y) 等价类中的一个奖励函数。最后，用公式 5 的右端替换 r（这对任何奖励函数都成立），我们有 f(r; πref, β)(x, y) = β log πr(y|x)

> πref(y|x). That is, the projection f produces a member of the equivalence class of r with the desired form, and we do not lose any generality in our reward model from the proposed reparameterization. We can alternatively view Theorem 1 as specifying exactly which reward function within each equivalence class the DPO reparameterization selects, that is, the reward function satisfying: X πref(y | x) exp y | {z } =π(y|x), using Thm. 1 reparam. i.e., π(y | x) is a valid distribution (probabilities are positive and sum to 1). However, following Eq. 4, we can see that Eq. 9 is the partition function of the optimal policy induced by the reward function r(x, y). The key insight of the DPO algorithm is that we can impose certain constraints on the under-constrained Plackett-Luce (and Bradley-Terry in particular) family of preference models, such that we preserve the class of representable reward models, but explicitly make the optimal policy in Eq. 4 analytically tractable for all prompts x.

πref(y|x)。也就是说，投影 f 生成了 r 的等价类中具有期望形式的成员，并且我们提出的重参数化不会使奖励模型失去任何一般性。 或者，我们可以将定理 1 视为精确指定了 DPO 重参数化在每个等价类中选择了哪个奖励函数，即满足以下条件的奖励函数： X πref(y | x) exp y | {z } =π(y|x)，利用定理 1 的重参数化 即，π(y | x) 是一个有效的分布（概率为正且总和为 1）。然而，根据公式 4，我们可以看到公式 9 是由奖励函数 r(x, y) 诱导的最优策略的配分函数。DPO 算法的关键洞察在于，我们可以对欠约束的 Plackett-Luce（特别是 Bradley-Terry）偏好模型族施加某些约束，从而在保留可表示奖励模型类别的同时，显式地使公式 4 中的最优策略对于所有提示 x 都具有解析可解性。

> 5.2 Instability of Actor-Critic Algorithms We can also use our framework to diagnose instabilities with standard actor-critic algorithms used for the RLHF, such as PPO. We follow the RLHF pipeline and focus on the RL fine-tuning step outlined in Section 3. We can draw connections to the control as inference framework [22] for the constrained RL problem outlined in 3. We assume a parameterized model πθ(y | x) and minimize DKL[πθ(y|x) || π∗(y | x)] where π∗is the optimal policy from Eq. 7 induced by the reward function rϕ(y, x). With some algebra this leads to the optimization objective: ⊠X πθ Eπθ(y|x) rϕ(x, y) −β log πref(y | x) exp max y | {z } f(rϕ,πref,β) This is the same objective optimized in prior works [51, 40, 1, 28] using the DPO-equivalent reward for the reward class of rϕ. In this setting, we can interpret the normalization term in f(rϕ, πref, β) as the soft value function of the reference policy πref. While this term does not affect the optimal solution, without it, the policy gradient of the objective could have high variance, making learning unstable. We can accommodate for the normalization term using a learned value function, but that can also be difficult to optimize. Alternatively, prior works have normalized rewards using a human completion baseline, essentially a single sample Monte-Carlo estimate of the normalizing term. In contrast the DPO reparameterization yields a reward function that does not require any baselines.

5.2 Actor-Critic 算法的不稳定性 我们同样可以利用我们的框架来诊断用于 RLHF 的标准 Actor-Critic 算法（如 PPO）中的不稳定性。我们遵循 RLHF 流程，重点关注第 3 节中概述的强化学习微调步骤。我们可以将第 3 节中概述的约束强化学习问题与“控制即推理”（control as inference）框架 [22] 联系起来。我们假设一个参数化模型 $\pi_\theta(y | x)$，并最小化 $D_{KL}[\pi_\theta(y|x) || \pi^*(y | x)]$，其中 π^* 是由奖励函数 $r_\phi(y, x)$ 诱导的、源自公式 7 的最优策略。经过一些代数推导，这得出了以下优化目标： $$ \mathbb{E}_{\pi_\theta(y|x)} \left[r_\phi(x, y) - \beta \log \frac{\pi_\theta(y | x)}{\pi_{ref}(y | x) \exp \left(\max_{y} \underbrace{f(r_\phi, \pi_{ref}, \beta)}_{\text{归一化项}} \right) \right] $$ 这是先前工作 [51, 40, 1, 28] 针对 r_ϕ 奖励类使用 DPO 等价奖励所优化的相同目标。在这种设置下，我们可以将 $f(r_\phi, \pi_{ref}, \beta)$ 中的归一化项解释为参考策略 π_{ref} 的软价值函数（soft value function）。虽然该项不影响最优解，但如果没有它，该目标的策略梯度可能会具有高方差，从而导致学习不稳定。我们可以使用学习到的价值函数来容纳该归一化项，但这同样可能难以优化。作为替代方案，先前工作使用人类完成样本基线（human completion baseline）对奖励进行归一化，这本质上是对归一化项的单样本蒙特卡洛估计。相比之下，DPO 重参数化产生了一种不需要任何基线的奖励函数。

> ⊠1 ⊠X πref(y | x) exp β r(x, y) (8) y ⊠1 ⊠β r(x, y) = 1, (9) ⊠1 ⊠⊠−β log πθ(y | x) β rϕ(x, y) (10) πref(y | x) | {z } KL IMDb Sentiment Generation 1.0 0.9 0.8 Reward 0.7 0.6 0.5 DPO (Ours) Unlikelihood PPO (Our impl.) PPO-GT (Our impl.) PPO-GT (TRL) Preferred-FT 0.4 0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0 KL(ref) reward for all KL values, demonstrating the quality of the optimization. Right. TL;DR summarization win rates vs. human-written summaries, using GPT-4 as evaluator. DPO exceeds PPO’ s best-case performance on summarization, while being more robust to changes in the sampling temperature.

$\pi_{\text{ref}}(y \mid x) \exp \left(\frac{1}{\beta} r(x, y) \right)$ (8) $\sum_{y} \pi_{\text{ref}}(y \mid x) \exp \left(\frac{1}{\beta} r(x, y) \right) = 1,$ (9) $\pi_{\text{ref}}(y \mid x) \left| \begin{array}{c} \text{KL} \\ \hline \end{array} \right. \exp \left(\frac{1}{\beta} r_{\phi}(x, y) - \beta \log \pi_{\theta}(y \mid x) \right)$ (10) IMDb 情感生成 1.0 0.9 0.8 奖励 0.7 0.6 0.5 DPO (Ours) Unlikelihood PPO (Our impl.) PPO-GT (Our impl.) PPO-GT (TRL) Preferred-FT 0.4 0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0 KL(ref) 在所有 KL 值下均获得奖励，展示了优化的质量。右图。使用 GPT-4 作为评估器，TL;DR 摘要任务中相对于人类撰写摘要的胜率。DPO 在摘要任务上超越了 PPO 的最佳表现，同时对采样温度的变化具有更强的鲁棒性。

> Experiments In this section, we empirically evaluate DPO’s ability to train policies directly from preferences. First, in a well-controlled text-generation setting, we ask: how efficiently does DPO trade off maximizing reward and minimizing KL-divergence with the reference policy, compared to common preference learning algorithms such as PPO? Next, we evaluate DPO’s performance on larger models and more difficult RLHF tasks, including summarization and dialogue. We find that with almost no tuning of hyperparameters, DPO tends to perform as well or better than strong baselines like RLHF with PPO as well as returning the best of N sampled trajectories under a learned reward function. Before presenting these results, we describe the experimental set-up; additional details are in Appendix C. Tasks. Our experiments explore three different open-ended text generation tasks. For all experiments, algorithms learn a policy from a dataset of preferences D = i=1. In controlled sentiment generation, x is a prefix of a movie review from the IMDb dataset [24], and the policy must generate y with positive sentiment. In order to perform a controlled evaluation, for this experiment we generate preference pairs over generations using a pre-trained sentiment classifier, where p(positive | x, yw) > p(positive | x, yl). For SFT, we fine-tune GPT-2-large until convergence on reviews from the train split of the IMDB dataset (further details in App C.1). In summarization, x is a forum post from Reddit; the policy must generate a summary y of the main points in the post. Following prior work, we use the Reddit TL;DR summarization dataset [43] along with human preferences gathered by Stiennon et al.. We use an SFT model fine-tuned on human-written forum post summaries2 with the TRLX [44] framework for RLHF. The human preference dataset was gathered by Stiennon et al. on samples from a different, but similarly-trained, SFT model. Finally, in single-turn dialogue, x is a human query, which may be anything from a question about astrophysics to a request for relationship advice. A policy must produce an engaging and helpful response y to a user’s query; we use the Anthropic Helpful and Harmless dialogue dataset [1], containing 170k dialogues between a human and an automated assistant. Each transcript ends with a pair of responses generated by a large (although unknown) language model along with a preference label denoting the human-preferred response. In this setting, no pre-trained SFT model is available; we therefore fine-tune an off-the-shelf language model on only the preferred completions to form the SFT model.

实验 在本节中，我们实证评估了DPO直接从偏好中训练策略的能力。首先，在一个控制良好的文本生成设置中，我们探讨：与PPO等常见的偏好学习算法相比，DPO在最大化奖励与最小化相对于参考策略的KL散度之间进行权衡的效率如何？其次，我们在更大的模型和更具挑战性的RLHF任务上评估DPO的性能，包括摘要生成和对话。我们发现，在几乎不需要调整超参数的情况下，DPO的表现往往与使用PPO的RLHF以及在学习到的奖励函数下返回N个采样轨迹中最佳结果等强基线相当或更优。在展示这些结果之前，我们描述实验设置；更多细节见附录C。 任务。我们的实验探索了三种不同的开放式文本生成任务。在所有实验中，算法从偏好数据集 D = 中学习策略。在受控的情感生成中，x 是IMDb数据集[24]中电影评论的前缀，策略必须生成具有正面情感的 y。为了进行受控评估，在此实验中，我们使用预训练的情感分类器在生成结果上生成偏好对，其中 p(positive | x, yw) > p(positive | x, yl)。对于SFT，我们在IMDb数据集的训练集分割上的评论上微调GPT-2-large直至收敛（更多细节见附录C.1）。在摘要生成中，x 是来自Reddit的论坛帖子；策略必须生成帖子主要观点的摘要 y。遵循先前工作，我们使用Reddit TL;DR摘要数据集[43]以及Stiennon等人收集的人类偏好。我们使用在人类撰写的论坛帖子摘要上微调的SFT模型，并使用TRLX[44]框架进行RLHF。该人类偏好数据集由Stiennon等人从一个不同但训练方式相似的SFT模型的样本中收集。最后，在单轮对话中，x 是人类查询，可以是从天体物理学问题到关系建议的任何内容。策略必须针对用户的查询生成引人入胜且有帮助的回复 y；我们使用Anthropic Helpful and Harmless对话数据集[1]，其中包含17万条人类与自动助手之间的对话。每段对话记录以由一个大型（但未知）语言模型生成的回复对结束，并附带一个偏好标签，指示人类偏好的回复。在此设置中，没有可用的预训练SFT模型；因此，我们仅在偏好完成的文本上微调一个现成的语言模型，以形成SFT模型。

> Evaluation. Our experiments use two different approaches to evaluation. In order to analyze the effectiveness of each algorithm in optimizing the constrained reward maximization objective, in the controlled sentiment generation setting we evaluate each algorithm by its frontier of achieved reward and KL-divergence from the reference policy; this frontier is computable because we have acccess to the ground-truth reward function (a sentiment classifier). However, in the real world, the ground truth reward function is not known; therefore, we evaluate algorithms with their win rate against a baseline policy, using GPT-4 as a proxy for human evaluation of summary quality and response helpfulness in the summarization and single-turn dialogue settings, respectively. For summarization, we use reference summaries in the test set as the baseline; for dialogue, we use the preferred response in the 2https://huggingface.co/CarperAI/openai_summarize_tldr_sft TL;DR Summarization Win Rate vs Reference DPO PPO Preferred-FT SFT GPT-J Best of 128 0.7 0.6 0.5 Win rate 0.4 0.3 0.2 0.1 0.0 0.00 0.25 0.50 0.75 1.00 Sampling temperature N x(i), y(i) w , y(i) ☒ Anthropic-HH Dialogue Win Rate vs Chosen 0.6 0.5 Win rate 0.4 0.3 0.2 DPO Best of 128 Preferred-FT Pythia-2.8B 0.1 0.25 0.50 0.75 1.00 Sampling temperature that improves over chosen summaries in the Anthropic-HH test set. Right. Win rates for different sampling temperatures over the course of training. DPO’ s improvement over the dataset labels is fairly stable over the course of training for different sampling temperatures.

评估。我们的实验采用两种不同的评估方法。为了分析每种算法在优化受约束奖励最大化目标方面的有效性，在受控情感生成设置中，我们通过算法达到的奖励与参考策略之间的KL散度前沿来评估每种算法；由于我们可以访问真实的奖励函数（即情感分类器），因此该前沿是可计算的。然而，在现实世界中，真实的奖励函数是未知的；因此，我们使用算法相对于基线策略的胜率来评估算法，分别使用GPT-4作为摘要质量和单轮对话响应有用性的人类评估代理，用于摘要和单轮对话设置。对于摘要，我们使用测试集中的参考摘要作为基线；对于对话，我们使用Anthropic-HH测试集中优于所选摘要的偏好响应作为基线。2https://huggingface.co/CarperAI/openai_summarize_tldr_sft TL;DR摘要相对于参考摘要的胜率 DPO PPO Preferred-FT SFT GPT-J Best of 128 0.7 0.6 0.5 胜率 0.4 0.3 0.2 0.1 0.0 0.00 0.25 0.50 0.75 1.00 采样温度 N x(i), y(i) w , y(i) l Anthropic-HH 对话相对于所选响应的胜率 0.6 0.5 胜率 0.4 0.3 0.2 DPO Best of 128 Preferred-FT Pythia-2.8B 0.1 0.25 0.50 0.75 1.00 采样温度 在Anthropic-HH测试集中优于所选摘要的响应。右图。训练过程中不同采样温度下的胜率。对于不同的采样温度，DPO相对于数据集标签的改进在训练过程中相当稳定。

> test dataset as the baseline. While existing studies suggest LMs can be better automated evaluators than existing metrics [10], we conduct a human study to justify our usage of GPT-4 for evaluation in Sec. 6.4. We find GPT-4 judgments correlate strongly with humans, with human agreement with GPT-4 typically similar or higher than inter-human annotator agreement. Methods. In addition to DPO, we evaluate several existing approaches to training language models to adhere to human preferences. Most simply, we explore zero-shot prompting with GPT-J [45] in the summarization task and 2-shot prompting with Pythia-2.8B [3] in the dialogue task. In addition, we evaluate the SFT model as well as Preferred-FT, which is a model fine-tuned with supervised learning on the chosen completion yw from either the SFT model (in controlled sentiment and summarization) or a generic LM (in single-turn dialogue). Another pseudo-supervised method is Unlikelihood [46], which simply optimizes the policy to maximize the probability assigned to yw and minimize the probability assigned to yl; we use an optional coefficient $\alpha \in [0, 1]$ on the ‘unlikelihood’ term. We also consider PPO [39] using a reward function learned from the preference data and PPO-GT, which is an oracle that learns from the ground truth reward function available in the controlled sentiment setting. In our sentiment experiments, we use two implementations of PPO-GT, one of-the-shelf version [44] as well as a modified version that normalizes rewards and further tunes hyperparameters to improve performance (we also use these modifications when running ‘normal’ PPO with learned rewards). Finally, we consider the Best of N baseline, sampling N responses from the SFT model (or Preferred-FT in dialogue) and returning the highest-scoring response according to a reward function learned from the preference dataset. This high-performing method decouples the quality of the reward model from the PPO optimization, but is computationally impractical even for moderate N as it requires sampling N completions for every query at test time.

我们将测试数据集作为基线。尽管现有研究表明，语言模型（LMs）可以成为比现有指标更优的自动化评估器[10]，但我们通过一项人类研究来论证在第6.4节中使用GPT-4进行评估的合理性。我们发现GPT-4的判断与人类判断高度相关，且人类与GPT-4的一致性通常与标注者之间的一致性相当或更高。 方法。除了DPO，我们还评估了多种用于训练语言模型以符合人类偏好的现有方法。最简单的是，我们在摘要任务中探索了使用GPT-J [45]的零样本提示，以及在对话任务中使用Pythia-2.8B [3]的双样本提示。此外，我们评估了SFT模型以及Preferred-FT，后者是一个通过监督学习在选中的补全yw上微调的模型，该yw来自SFT模型（在受控情感分析和摘要任务中）或通用语言模型（在单轮对话中）。另一种伪监督方法是Unlikelihood [46]，它简单地优化策略以最大化分配给yw的概率并最小化分配给yl的概率；我们在“非似然”项上使用可选系数$\alpha \in [0, 1]$。我们还考虑了使用从偏好数据中学习的奖励函数的PPO [39]，以及PPO-GT，后者是一个在受控情感分析设置中从可用的真实奖励函数学习的预言机。在我们的情感分析实验中，我们使用了两种PPO-GT实现，一种是现成的版本[44]，另一种是归一化奖励并进一步调整超参数以提高性能的修改版本（我们在运行使用学习奖励的“常规”PPO时也使用了这些修改）。最后，我们考虑了Best of N基线，即从SFT模型（或在对话任务中从Preferred-FT）中采样N个响应，并根据从偏好数据集中学习的奖励函数返回得分最高的响应。这种高性能方法将奖励模型的质量与PPO优化解耦，但由于在测试时每个查询都需要采样N个补全，因此即使对于中等大小的N，计算上也不切实际。

6.1 How well can DPO optimize the RLHF objective? The KL-constrained reward maximization objective used in typical RLHF algorithms balances exploitation of reward while restricting the policy from deviating far from the reference policy. Therefore, when comparing algorithms, we must take into account both reward achieved as well as the KL discrepancy; achieving slightly higher reward but with much higher KL is not necessarily desirable. multiple training runs for each algorithm, using a different hyperparameter for policy conservativeness in each run (target KL ∈{3, 6, 9, 12} for PPO, β ∈{0.05, 0.1, 1, 5}, α ∈{0.05, 0.1, 0.5, 1} for unlikelihood, random seeds for preferred-FT). This sweep includes 22 runs in total. After each 100 training steps until convergence, we evaluate each policy on a set of test prompts, computing the average reward under the true reward function as well as the average sequence-level KL3 with the reference policy KL (π || πref). We find that DPO produces by far the most efficient frontier, achieving the highest reward while still achieving low KL. This result is particularly notable for multiple reasons. First, DPO and PPO optimize the same objective, but DPO is notably more efficient; 3That is, the sum of the per-timestep KL-divergences. Dialogue Win Rate Evolution 0.70 0.65 0.60 0.55 Win rate 0.50 0.45 0.40 0.35 DPO (temp = 1.0) DPO (temp = 0.7) 0.30 1200 1500 1800 2100 2400 2700 3000 3300 Fine-tuning step

6.1 DPO 能在多大程度上优化 RLHF 目标？ 典型 RLHF 算法中使用的 KL 约束奖励最大化目标，在利用奖励的同时，限制策略偏离参考策略过远。因此，在比较算法时，我们必须同时考虑所获得的奖励以及 KL 散度；获得略高的奖励但伴随高得多的 KL 散度并不一定是可取的。我们对每种算法进行了多次训练运行，每次运行使用不同的策略保守性超参数（PPO 的目标 KL ∈{3, 6, 9, 12}，β ∈{0.05, 0.1, 1, 5}，unlikelihood 的 α ∈{0.05, 0.1, 0.5, 1}，以及 preferred-FT 的随机种子）。该扫描总共包含 22 次运行。在每 100 个训练步骤直至收敛后，我们在一系列测试提示上评估每个策略，计算在真实奖励函数下的平均奖励，以及与参考策略的 KL (π || πref) 的平均序列级 KL3。我们发现 DPO 产生了迄今为止最高效的前沿，在实现高奖励的同时仍保持低 KL。这一结果因多个原因而尤为显著。首先，DPO 和 PPO 优化的是相同的目标，但 DPO 的效率明显更高； 3即每个时间步 KL 散度之和。 对话胜率演变 0.70 0.65 0.60 0.55 胜率 0.50 0.45 0.40 0.35 DPO (temp = 1.0) DPO (temp = 0.7) 0.30 1200 1500 1800 2100 2400 2700 3000 3300 微调步骤

DPO’s reward/KL tradeoff strictly dominates PPO. Second, DPO achieves a better frontier than PPO, even when PPO can access ground truth rewards (PPO-GT). 6.2 Can DPO scale to real preference datasets? Next, we evaluate fine-tuning performance of DPO on summarization and single-turn dialogue. For summarization, automatic evaluation metrics such as ROUGE can be poorly correlated with human preferences [40], and prior work has found that fine-tuning LMs using PPO on human preferences to provide more effective summaries. We evaluate different methods by sampling completions on the test split of TL;DR summarization dataset, and computing the average win rate against reference completions in the test set. The completions for all methods are sampled at temperatures varying from 0.0 to 1.0, and the win rates are shown in Figure 2 (right). DPO, PPO and Preferred-FT all fine-tune the same GPT-J SFT model4. We find that DPO has a win rate of approximately 61% at a temperature of 0.0, exceeding the performance of PPO at 57% at its optimal sampling temperature of 0.0. DPO also achieves a higher maximum win rate compared to the best of N baseline. We note that we did not meaningfully tune DPO’s β hyperparameter, so these results may underestimate DPO’s potential. Moreover, we find DPO to be much more robust to the sampling temperature than PPO, the performance of which can degrade to that of the base GPT-J model at high temperatures. Preferred-FT does not improve significantly over the SFT model. We also compare DPO and PPO head-to-head in human evaluations in Section 6.4, where DPO samples at temperature 0.25 were preferred 58% times over PPO samples at temperature 0.

DPO 的奖励/KL 权衡严格优于 PPO。其次，即使 PPO 能够访问真实奖励（PPO-GT），DPO 也能实现比 PPO 更优的前沿。6.2 DPO 能否扩展到真实的偏好数据集？ 接下来，我们评估 DPO 在摘要生成和单轮对话中的微调性能。对于摘要生成，ROUGE 等自动评估指标与人类偏好的相关性可能较差 [40]，且先前的研究发现，使用 PPO 基于人类偏好微调语言模型可以提供更有效的摘要。我们在 TL;DR 摘要数据集的测试集上采样补全结果，并计算相对于测试集中参考补全结果的平均胜率来评估不同方法。所有方法的补全结果均在 0.0 到 1.0 之间变化的温度下采样，胜率如图 2（右侧）所示。DPO、PPO 和 Preferred-FT 均对同一个 GPT-J SFT 模型4进行微调。我们发现，在温度为 0.0 时，DPO 的胜率约为 61%，超过了 PPO 在其最佳采样温度 0.0 时 57% 的性能。与最佳 N 基线相比，DPO 也实现了更高的最大胜率。我们注意到，我们并未对 DPO 的 β 超参数进行有意义的调整，因此这些结果可能低估了 DPO 的潜力。此外，我们发现 DPO 比 PPO 对采样温度具有更强的鲁棒性，后者的性能在高温下可能会退化到基础 GPT-J 模型的水平。Preferred-FT 相对于 SFT 模型没有显著改进。我们还在第 6.4 节中通过人工评估对 DPO 和 PPO 进行了直接对比，其中 DPO 在温度 0.25 下的样本被偏好的比例为 58%，高于 PPO 在温度 0.

> On single-turn dialogue, we evaluate the different methods on the subset of the test split of the Anthropic HH dataset [1] with one step of human-assistant interaction. GPT-4 evaluations use the preferred completions on the test as the reference to compute the win rate for different methods. As there is no standard SFT model for this task, we start with a pre-trained Pythia-2.8B, use Preferred-FT to train a reference model on the chosen completions such that completions are within distribution of the model, and then train using DPO. We also compare against the best of 128 Preferred-FT completions (we found the Best of N baseline plateaus at 128 completions for this task; see Appendix well or better for the best-performing temperatures for each method. We also evaluate an RLHF model trained with PPO on the Anthropic HH dataset 5 from a well-known source 6, but are unable to find a prompt or sampling temperature that gives performance better than the base Pythia-2.8B model. Based on our results from TL;DR and the fact that both methods optimize the same reward function, we consider Best of 128 a rough proxy for PPO-level performance. Overall, DPO is the only computationally efficient method that improves over the preferred completions in the Anthropic HH dataset, and provides similar or better performance to the computationally demanding Best of 128 baseline. Finally, Figure 3 shows that DPO converges to its best performance relatively quickly.

在单轮对话任务中，我们在Anthropic HH数据集[1]测试集的一个子集上评估了不同方法，该子集包含一步人类-助手交互。GPT-4评估使用测试集中的偏好补全作为参考，以计算不同方法的胜率。由于该任务没有标准的SFT模型，我们首先使用预训练的Pythia-2.8B，利用Preferred-FT在选定的补全上训练一个参考模型，使得补全位于该模型的分布之内，然后使用DPO进行训练。我们还与128个Preferred-FT补全中的最佳结果进行了比较（我们发现Best of N基线在该任务上在128个补全时达到平台期；见附录）。我们根据每种方法表现最佳的温度进行了评估。我们还评估了一个使用PPO在Anthropic HH数据集上训练的RLHF模型，该模型来自一个知名来源[5]，但我们无法找到能给出优于基础Pythia-2.8B模型性能的提示或采样温度。基于我们在TL;DR上的结果以及两种方法优化相同奖励函数的事实，我们将Best of 128视为PPO级别性能的粗略代理。总体而言，DPO是唯一一种计算效率较高且能在Anthropic HH数据集上优于偏好补全的方法，并且提供了与计算需求较高的Best of 128基线相当或更好的性能。最后，图3表明DPO相对快速地收敛到其最佳性能。

> 6.3 Generalization to a new input distribution To further compare the performance of PPO and DPO un- der distribution shifts, we evaluate the PPO and DPO poli- cies from our Reddit TL;DR summarization experiment on a different distribution, news articles in the test split of the CNN/DailyMail dataset [26], using the best sampling temper- atures from TL;DR (0 and 0.25). The results are presented in Table 1. We computed the GPT-4 win rate against the ground-truth summaries in the datasets, using the same GPT- 4 (C) prompt we used for Reddit TL;DR, but replacing the words “forum post” with “news article”. For this new distribution, DPO continues to outperform the PPO policy by a significant margin. This experiment provides initial evidence that DPO policies can generalize similarly well to PPO policies, even though DPO does not use the additional unlabeled Reddit TL;DR prompts that PPO uses. 4https://huggingface.co/CarperAI/openai_summarize_tldr_sft 5https://huggingface.co/reciprocate/ppo_hh_pythia-6B 6https://github.com/CarperAI/trlx/tree/main/examples/hh Win rate vs. ground truth Alg. Temp 0 Temp 0.25 DPO 0.36 0.31 PPO 0.26 0.23 ground truth summaries for out-of-distribution CNN/DailyMail input articles. 6.4 Validating GPT-4 judgments with human judgments We conduct a human study to verify the reliability of GPT-4’ s judgments, using the results of the TL;DR summarization experiment and two different GPT-4 prompts. The GPT-4 (S) (sim- ple) prompt simply asks for which summary better-summarizes the important information in the post. The GPT-4 (C) (concise) prompt also asks for which summary is more concise; we eval- uate this prompt because we find that GPT-4 prefers longer, more repetitive summaries than hu- mans do with the GPT-4 (S) prompt. See Appendix C.2 for the complete prompts. We perform three comparisons, using the highest (DPO, temp. 0.25), the lowest (PPO, temp. 1.0), and a

6.3 对新输入分布的泛化能力 为了进一步比较PPO和DPO在分布偏移下的性能，我们在不同的分布上评估了我们在Reddit TL;DR摘要实验中获得的PPO和DPO策略，即CNN/DailyMail数据集[26]测试集中的新闻文章，并使用TL;DR中最佳的采样温度（0和0.25）。结果如表1所示。我们使用与Reddit TL;DR相同的GPT-4 (C)提示词，计算了GPT-4相对于数据集中真实摘要的胜率，但将“论坛帖子”一词替换为“新闻文章”。对于这一新分布，DPO策略继续以显著优势超越PPO策略。该实验提供了初步证据，表明尽管DPO未使用PPO所使用的额外未标记的Reddit TL;DR提示词，但DPO策略的泛化能力与PPO策略同样良好。4https://huggingface.co/CarperAI/openai_summarize_tldr_sft 5https://huggingface.co/reciprocate/ppo_hh_pythia-6B 6https://github.com/CarperAI/trlx/tree/main/examples/hh 相对于真实摘要的胜率 算法 温度 0 温度 0.25 DPO 0.36 0.31 PPO 0.26 0.23 针对分布外CNN/DailyMail输入文章的真实摘要。 6.4 通过人类判断验证GPT-4的判断 我们进行了一项人类研究，以验证GPT-4判断的可靠性，使用了TL;DR摘要实验的结果和两种不同的GPT-4提示词。GPT-4 (S)（简单）提示词仅询问哪个摘要更好地总结了帖子中的重要信息。GPT-4 (C)（简洁）提示词还询问哪个摘要更简洁；我们评估此提示词是因为我们发现，与GPT-4 (S)提示词下人类的表现相比，GPT-4更倾向于选择更长、更重复的摘要。完整提示词见附录C.2。我们进行了三项比较，分别使用最高的（DPO，温度0.25）、最低的（PPO，温度1.0）以及一个

> middle-performing (SFT, temp. 0.25) method with the aim of covering a diversity of sample qualities; all three methods are compared against greedily- sampled PPO (its best-performing temperature). We find that with both prompts, GPT-4 tends to agree with humans about as often as humans agree with each other, suggesting that GPT-4 is a reason- able proxy for human evaluations (due to limited human raters, we only collect multiple human judg- ments for the DPO and PPO-1 comparisons). Over- all, the GPT-4 (C) prompt generally provides win rates more representative of humans; we therefore use this prompt for the main results in Section 6.2. For additional details about the human study, in- cluding the web interface presented to raters and the list of human volunteers, see Appendix D.3.

中等表现（SFT，温度 0.25）的方法，旨在涵盖多样化的样本质量；这三种方法均与贪婪采样的 PPO（其表现最佳的温度）进行比较。我们发现，在两种提示下，GPT-4 与人类达成一致的概率与人类彼此之间达成一致的概率相当，这表明 GPT-4 是人工评估的合理替代（由于人工评估者数量有限，我们仅针对 DPO 和 PPO-1 的比较收集了多个人工判断）。总体而言，GPT-4 (C) 提示通常能提供更符合人类认知的胜率；因此，我们在第 6.2 节的主要结果中使用了该提示。关于人工研究的更多细节，包括向评估者展示的网页界面以及人类志愿者名单，请参见附录 D.3。

> Discussion Learning from preferences is a powerful, scalable framework for training capable, aligned language models. We have introduced DPO, a simple training paradigm for training language models from preferences without reinforcement learning. Rather than coercing the preference learning problem into a standard RL setting in order to use off-the-shelf RL algorithms, DPO identifies a mapping between language model policies and reward functions that enables training a language model to satisfy human preferences directly, with a simple cross-entropy loss, without reinforcement learning or loss of generality. With virtually no tuning of hyperparameters, DPO performs similarly or better than existing RLHF algorithms, including those based on PPO; DPO thus meaningfully reduces the barrier to training more language models from human preferences. Limitations & Future Work. Our results raise several important questions for future work. How does the DPO policy generalize out of distribution, compared with learning from an explicit reward function? Our initial results suggest that DPO policies can generalize similarly to PPO-based models, but more comprehensive study is needed. For example, can training with self-labeling from the DPO policy similarly make effective use of unlabeled prompts? On another front, how does reward over-optimization manifest in the direct preference optimization setting, and is the slight decrease in performance in Figure 3-right an instance of it? Additionally, while we evaluate models up to 6B parameters, exploration of scaling DPO to state-of-the-art models orders of magnitude larger is an exciting direction for future work. Regarding evaluations, we find that the win rates computed by GPT-4 are impacted by the prompt; future work may study the best way to elicit high-quality judgments from automated systems. Finally, many possible applications of DPO exist beyond training language models from human preferences, including training generative models in other modalities.

讨论 从偏好中学习是训练强大且对齐的语言模型的一种强大且可扩展的框架。我们提出了DPO，这是一种无需强化学习即可从偏好中训练语言模型的简单训练范式。DPO没有将偏好学习问题强行纳入标准强化学习设置以使用现成的强化学习算法，而是识别了语言模型策略与奖励函数之间的映射，从而能够直接训练语言模型以满足人类偏好，仅使用简单的交叉熵损失，无需强化学习且不损失通用性。在几乎无需调整超参数的情况下，DPO的表现与现有的RLHF算法相当或更好，包括基于PPO的算法；因此，DPO显著降低了从人类偏好中训练更多语言模型的门槛。 局限性与未来工作。我们的结果提出了几个供未来研究的重要问题。与从显式奖励函数中学习相比，DPO策略在分布外（out-of-distribution）的泛化能力如何？我们的初步结果表明，DPO策略的泛化能力可能与基于PPO的模型相似，但需要更全面的研究。例如，使用DPO策略进行自我标注的训练是否也能有效利用未标记的提示？另一方面，在直接偏好优化设置中，奖励过度优化是如何表现的？图3右侧观察到的性能轻微下降是否是这种情况的一个实例？此外，虽然我们评估了最多6B参数的模型，但探索将DPO扩展到比当前最先进模型大几个数量级的模型是一个令人兴奋的未来研究方向。关于评估，我们发现由GPT-4计算的胜率受到提示的影响；未来的工作可以研究如何从自动系统中引出高质量判断的最佳方式。最后，除了从人类偏好中训练语言模型外，DPO还有许多潜在的应用，包括在其他模态中训练生成模型。

> Acknowledgements EM gratefully acknowledges funding from a Knight-Hennessy Graduate Fellowship. CF and CM are CIFAR Fellows. This work was supported in part by the Stanford Accelerator for Learning (SAL) and Stanford Institute for Human-Centered Artificial Intelligence (HAI) Generative AI for the Future of Learning seed grant program. The Stanford Center for Research on Foundation Models (CRFM) provided part of the compute resources used for the experiments in this work. This work was supported in part by ONR grant N00014-20-1-2675. DPO SFT PPO-1 N respondents GPT-4 (S) win % GPT-4 (C) win % Human win % GPT-4 (S)-H agree GPT-4 (C)-H agree H-H agree - and per-judgment agreement on TL;DR summariza- tion samples. Humans agree with GPT-4 about as much as they agree with each other. Each experi- ment compares a summary from the stated method with a summary from PPO with temperature 0.

致谢 EM 感谢奈特-亨内西研究生奖学金的资助。CF 和 CM 是 CIFAR 研究员。本工作部分得到了斯坦福学习加速器（SAL）以及斯坦福以人为本的人工智能研究所（HAI）“面向未来学习的生成式人工智能”种子资助计划的支持。斯坦福基础模型研究中心（CRFM）提供了本工作实验中使用的一部分计算资源。本工作部分得到了美国海军研究办公室（ONR）资助项目 N00014-20-1-2675 的支持。 DPO SFT PPO-1 N 名受访者 GPT-4 (S) 胜率 % GPT-4 (C) 胜率 % 人类胜率 % GPT-4 (S)-H 一致率 GPT-4 (C)-H 一致率 H-H 一致率 - 以及针对 TL;DR 摘要样本的逐判断一致率。人类与 GPT-4 的一致程度与人类彼此之间的一致程度相当。每个实验都将所述方法生成的摘要与温度设为 0 的 PPO 生成的摘要进行比较。

— 参考文献（原文保留） —

[1] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, N. Joseph, S. Kadavath, J. Kernion, T. Conerly, S. El-Showk, N. Elhage, Z. Hatfield- Dodds, D. Hernandez, T. Hume, S. Johnston, S. Kravec, L. Lovitt, N. Nanda, C. Olsson, D. Amodei, T. Brown, J. Clark, S. McCandlish, C. Olah, B. Mann, and J. Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022. [2] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirho- seini, C. McKinnon, C. Chen, C. Olsson, C. Olah, D. Hernandez, D. Drain, D. Ganguli, D. Li, E. Tran-Johnson, E. Perez, J. Kerr, J. Mueller, J. Ladish, J. Landau, K. Ndousse, K. Lukosuite, L. Lovitt, M. Sellitto, N. Elhage, N. Schiefer, N. Mercado, N. DasSarma, R. Lasenby, R. Lar- son, S. Ringer, S. Johnston, S. Kravec, S. E. Showk, S. Fort, T. Lanham, T. Telleen-Lawton, T. Conerly, T. Henighan, T. Hume, S. R. Bowman, Z. Hatfield-Dodds, B. Mann, D. Amodei, N. Joseph, S. McCandlish, T. Brown, and J. Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022. [3] S. Biderman, H. Schoelkopf, Q. Anthony, H. Bradley, K. O’Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff, A. Skowron, L. Sutawika, and O. van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023. [4] H. Bong and A. Rinaldo. Generalized results for the existence and consistency of the MLE in the Bradley-Terry-Luce model. International Conference on Machine Learning, 2022.

[5] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika, 39(3/4):324–345, 1952. doi: https://doi.org/10.2307/2334029. [6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Lan- guage models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 1877– 1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_ files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf. [7] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020. [8] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang. Sparks of artificial general intelligence: Early experiments with GPT-4, 2023. arXiv preprint arXiv:2303.12712. [9] R. Busa-Fekete, B. Szörényi, P. Weng, W. Cheng, and E. Hüllermeier. Preference-based reinforcement learning: evolutionary direct policy search using a preference-based racing algorithm. Machine Learning, 97(3):327–351, July 2014. doi: 10.1007/s10994-014-5458-8. URL https://doi.org/10.1007/s10994-014-5458-8.

[10] Y. Chen, R. Wang, H. Jiang, S. Shi, and R.-L. Xu. Exploring the use of large language models for reference-free text quality evaluation: A preliminary empirical study. ArXiv, abs/2304.00723, 2023. [11] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311, 2022. [12] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Sys- tems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf. [13] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, A. Castro-Ros, M. Pellat, K. Robinson, D. Valter, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei. Scaling instruction-finetuned language models, 2022. [14] M. Dudík, K. Hofmann, R. E. Schapire, A. Slivkins, and M. Zoghi. Contextual dueling bandits. In P. Grünwald, E. Hazan, and S. Kale, editors, Proceedings of The 28th Conference on Learning Theory, volume 40 of Proceedings of Machine Learning Research, pages 563–587, Paris, France,

03–06 Jul 2015. PMLR. URL https://proceedings.mlr.press/v40/Dudik15.html. [15] D. Go, T. Korbak, G. Kruszewski, J. Rozen, N. Ryu, and M. Dymetman. Aligning language models with preferences through f-divergence minimization. In Proceedings of the 40th International Conference on Machine Learning, ICML’23. JMLR.org, 2023. [16] A. Jain, B. Wojcik, T. Joachims, and A. Saxena. Learning trajectory preferences for manip- ulators via iterative improvement. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/ 2013/file/c058f544c737782deacefa532d9add4c-Paper.pdf. [17] N. Jaques, S. Gu, D. Bahdanau, J. M. Hernández-Lobato, R. E. Turner, and D. Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In International Conference on Machine Learning, pages 1645–1654. PMLR, 2017. [18] N. Jaques, J. H. Shen, A. Ghandeharioun, C. Ferguson, A. Lapedriza, N. Jones, S. S. Gu, and R. Picard. Human-centric dialog training via offline reinforcement learning. arXiv preprint [19] T. Korbak, H. Elsahar, G. Kruszewski, and M. Dymetman. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, Advances in Neural Information Processing Systems, volume 35, pages 16203–16220. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ 67496dfa96afddab795530cc7c69b57a-Paper-Conference.pdf.

[20] J. Kreutzer, J. Uyheng, and S. Riezler. Reliability and learnability of human bandit feedback for sequence-to-sequence reinforcement learning. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1777–1788, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/ P18-1165. URL https://aclanthology.org/P18-1165. [21] A. Kupcsik, D. Hsu, and W. S. Lee. Learning Dynamic Robot-to-Human Object Handover from Human Feedback, pages 161–176. Springer International Publishing, 01 2018. ISBN 978-3-319-51531-1. doi: 10.1007/978-3-319-51532-8_10. [22] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018. [23] R. D. Luce. Individual choice behavior: A theoretical analysis. Courier Corporation, 2012. [24] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015. [25] S. Mishra, D. Khashabi, C. Baral, and H. Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3470–3487, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long. 244. URL https://aclanthology.org/2022.acl-long.244.

[26] R. Nallapati, B. Zhou, C. dos Santos, Ç. Gulçehre, and B. Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, pages 280–290, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1028. URL https:// aclanthology.org/K16-1028. [27] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro, A. Phanishayee, and M. Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC ’21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384421. doi: 10.1145/3458817.3476209. URL https://doi.org/10.1145/3458817.3476209. [28] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, Advances in Neural Information Processing Systems, volume 35, pages 27730–27744. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/ paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

[29] R. Paulus, C. Xiong, and R. Socher. A deep reinforced model for abstractive summarization. In International Conference on Learning Representations, 2018. URL https://openreview. net/forum?id=HkAClQgA-. [30] X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. arXiv preprint arXiv:1910.00177, 2019. [31] J. Peters and S. Schaal. Reinforcement learning by reward-weighted regression for operational space control. In Proceedings of the 24th international conference on Machine learning, pages 745–750, 2007. [32] R. L. Plackett. The analysis of permutations. Journal of the Royal Statistical Society. Series C (Applied Statistics), 24(2):193–202, 1975. doi: https://doi.org/10.2307/2346567. [33] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners, 2019. Ms., OpenAI. [34] R. Ramamurthy, P. Ammanabrolu, K. Brantley, J. Hessel, R. Sifa, C. Bauckhage, H. Hajishirzi, and Y. Choi. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. In The Eleventh International Conference on Learning Representations, 2023. URL https://openreview. net/forum?id=8aHzds2uUyB. [35] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. CoRR, abs/1511.06732, 2015. [36] D. Sadigh, A. D. Dragan, S. Sastry, and S. A. Seshia. Active preference-based learning of

reward functions. In Robotics: Science and Systems (RSS), 2017. [37] A. Saha, A. Pacchiano, and J. Lee. Dueling rl: Reinforcement learning with trajectory preferences. In F. Ruiz, J. Dy, and J.-W. van de Meent, editors, Proceedings of The 26th International Conference on Artificial Intelligence and Statistics, volume 206 of Proceed- ings of Machine Learning Research, pages 6263–6289. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/saha23a.html. [38] V. Sanh, A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M. S. Bari, C. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chh- ablani, N. Nayak, D. Datta, J. Chang, M. T.-J. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Fevry, J. A. Fries, R. Teehan, T. L. Scao, S. Biderman, L. Gao, T. Wolf, and A. M. Rush. Multi- task prompted training enables zero-shot task generalization. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4. [39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017. [40] N. Stiennon, L. Ouyang, J. Wu, D. M. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. Christiano. Learning to summarize from human feedback, 2022. [41] R. Thoppilan, D. D. Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, V. Zhao, Y. Zhou, C.-C. Chang, I. Krivokon, W. Rusch, M. Pickett, P. Srinivasan, L. Man, K. Meier-Hellstern, M. R. Morris, T. Doshi, R. D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. Aguera- Arcas, C. Cui, M. Croak, E. Chi, and Q. Le. Lamda: Language models for dialog applications, 2022.

[42] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023. [43] M. Völske, M. Potthast, S. Syed, and B. Stein. TL;DR: Mining Reddit to learn automatic summarization. In Proceedings of the Workshop on New Frontiers in Summarization, pages 59–63, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4508. URL https://aclanthology.org/W17-4508. [44] L. von Werra, J. Tow, reciprocated, S. Matiana, A. Havrilla, cat state, L. Castricato, Alan, D. V. Phung, A. Thakur, A. Bukhtiyarov, aaronrmm, F. Milo, Daniel, D. King, D. Shin, E. Kim, J. Wei, M. Romero, N. Pochinkov, O. Sanseviero, R. Adithyan, S. Siu, T. Simonini, V. Blagojevic, X. Song, Z. Witten, alexandremuzio, and crumb. CarperAI/trlx: v0.6.0: LLaMa (Alpaca), Benchmark Util, T5 ILQL, Tests, Mar. 2023. URL https://doi.org/10.5281/zenodo. 7790115. [45] B. Wang and A. Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021. [46] S. Welleck, I. Kulikov, S. Roller, E. Dinan, K. Cho, and J. Weston. Neural text generation with unlikelihood training. arXiv preprint arXiv:1908.04319, 2019. [47] R. J. Williams. Simple statistical gradient-following algorithms for connectionist rein- forcement learning. Mach. Learn., 8(3–4):229–256, may 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

[48] Y. Wu and B. Hu. Learning to extract coherent summary via deep reinforcement learning. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8. [49] X. Yan, C. Luo, C. L. A. Clarke, N. Craswell, E. M. Voorhees, and P. Castells. Human preferences as dueling bandits. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '22, page 567–577, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531991. URL https://doi.org/10.1145/3477495.3531991. [50] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. Journal of Computer and System Sciences, 78(5):1538–1556, 2012. ISSN 0022-0000. doi: https: //doi.org/10.1016/j.jcss.2011.12.028. URL https://www.sciencedirect.com/science/ article/pii/S0022000012000281. JCSS Special Issue: Cloud Computing 2011. [51] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences, 2020. Author Contributions All authors provided valuable contributions to designing, analyzing, and iterating on experiments, writing and editing the paper, and generally managing the project's progress.

RR proposed using autoregressive reward models in discussions with EM; derived the DPO objective; proved the theoretical properties of the algorithm and wrote the relevant sections and appendices. He also suggested and helped with organizing experiments and contributed some of the PPO and reward learning baselines. AS initiated the discussion on using weighted regression methods as an alternative to PPO; initiated project-related organization, wrote initial analysis connecting DPO with weighted regression and unlikelihood; design and iterations of DPO + baseline implementations, initial exploratory exper- iments for DPO; substantial experiment organization and design (datasets, baselines, evaluation); led model training and evaluation for controlled sentiment generation and summarization; design iterations for GPT-4 evaluation (particularly summarization); substantial writing contributions to abstract, prelims/method and experiments; editing contributions to other sections. EM provided input on early discussions on learning autoregressive reward functions; wrote the first implementation of DPO and ran the first DPO experiments; trained the large-scale (summarization and dialogue) DPO models used in paper experiments; conducted initial GPT-4 win rate evaluations and set up related infrastructure; recruited participants for, conducted, and analyzed results from the human study; wrote the abstract, introduction, related work, discussion, and most of experiments; and assisted with editing the rest of the paper.

CF, CM, & SE supervised the research, suggested ideas and experiments, and assisted in writing the paper. A Mathematical Derivations A.1 Deriving the Optimum of the KL-Constrained Reward Maximization Objective In this appendix, we will derive Eq. 4. Analogously to Eq. 3, we optimize the following objective: π Ex∼D,y∼π □max r(x, y) under any reward function r(x, y), reference model πref and a general non-parametric policy class. We now have: π Ex∼D,y∼π −βDKL □□□π(y|x) || πref(y|x) max r(x, y) □π Ex∼DEy∼π(y|x) = max □log π(y|x) π Ex∼DEy∼π(y|x) = min □□log π(y|x) π Ex∼DEy∼π(y|x) = min where we have partition function: X πref(y|x) exp Z(x) = y Note that the partition function is a function of only x and the reference policy πref, but does not depend on the policy π. We can now define π∗(y|x) = Z(x)πref(y|x) exp −βDKL □□□π(y|x)||πref(y|x) (11) □□r(x, y) −β log π(y|x) πref(y|x) □πref(y|x) −1 β r(x, y) □□-log Z(x) □12) □ Z(x)πref(y|x) exp β r(x, y) □1 □β r(x, y) . □1 □β r(x, y) , which is a valid probability distribution as π∗(y|x) ⩾0 for all y and P y π∗(y|x) = 1. Since Z(x) is not a function of y, we can then re-organize the final objective in Eq 12 as: □π Ex∼D Ey∼π(y|x) min π Ex∼D [DKL(π(y|x) || π∗(y|x)) −log Z(x)] (14) min Now, since Z(x) does not depend on π, the minimum is achieved by the policy that minimizes the first KL term. Gibbs' inequality tells us that the KL-divergence is minimized at 0 if and only if the two distributions are identical. Hence we have the optimal solution: π(y|x) = π∗(y|x) = Z(x)πref(y|x) exp for all x ∈D. This completes the derivation.

A.2 Deriving the DPO Objective Under the Bradley-Terry Model It is straightforward to derive the DPO objective under the Bradley-Terry preference model as we have p∗(y1 ≻y2|x) = exp (r∗(x, y1)) exp (r∗(x, y1)) + exp (r∗(x, y2)) (16) In Section 4 we showed that we can express the (unavailable) ground-truth reward through its corresponding optimal policy: r∗(x, y) = β log π∗(y|x) Substituting Eq. 17 into Eq. 16 we obtain: exp p∗(y1 ≻y2|x) = □β log π∗(y1|x) exp πref(y1|x) + β log Z(x) = 1 + exp □β log π∗(y2|x) πref(y2|x) −β log π∗(y1|x) πref(y1|x) −β log π∗(y2|x) β log π∗(y1|x) □= σ The last line is the per-instance loss in Equation 7. A.3 Deriving the DPO Objective Under the Plackett-Luce Model The Plackett-Luce model [32, 23] is a generalization of the Bradley-Terry model over rankings (rather than just pair-wise comparisons). Similar to to the Bradley-Terry model, it stipulates that when presented with a set of possible choices, people prefer a choice with probability proportional to the value of some latent reward function for that choice. In our context, when presented with a prompt x and a set of K answers y1, . . . , yK a user would output a permutation τ : [K] →[K], giving their ranking of the answers. The Plackett-Luce model stipulates that p∗(τ| y1, . . . , yK, x) = Notice that when K = 2, Equation 18 reduces to the Bradley-Terry model. However, for the general Plackett-Luce model, we can still utilize the results of Eq. 5 and substitute the reward function parameterized by its optimal policy. Similarly to Appendix A.2, the normalization constant Z(x) cancels out and we're left with:

K Y p∗(τ|y1, . . . , yK, x) = k=1 log π(y|x) −log Z(x) = (13) π∗(y|x) 1 β r(x, y) (15) πref(y|x) + β log Z(x) (17) β log π∗(y1|x) πref(y1|x) + β log Z(x) β log π∗(y2|x) + exp πref(y2|x) + β log Z(x) πref(y1|x) πref(y2|x) K Y exp(r∗(x, yτ(k))) PK (18) j=k exp(r∗(x, yτ(j))) k=1 π∗(yτ(k)|x) πref(yτ(k)|x) exp β log (19) π∗(yτ(j)|x) πref(yτ(j)|x) PK j=k exp β log Similarly to the approach of Section 4, if we have access to a dataset D = {τ (i), y(i) 1 , . . . , y(i) K , x(i)}N i=1 of prompts and user-specified rankings, we can use a parameterized model and optimize this objective with maximum-likelihood.: LDPO(πθ, πref) = −Eτ,y1,...,yK,x∼D A.4 Deriving the Gradient of the DPO Objective In this section we derive the gradient of the DPO objective: ∇θLDPO(πθ; πref) = −∇θE(x,yw,yl)∼D We can rewrite the RHS of Equation 21 as ∇θLDPO(πθ; πref) = −E(x,yw,yl)∼D where u = β log πθ(yl|x) πref(yl|x) −β log πθ(yw|x) πref(yw|x). Using the properties of sigmoid function σ′(x) = σ(x)(1 −σ(x)) and σ(−x) = 1 −σ(x), we obtain the final gradient ∇θLDPO(πθ; πref) = πref(yw|x) −β log πθ(yl|x) β log πθ(yw|x) −E(x,yw,yl)∼D βσ After using the reward substitution of ˆrθ(x, y) = β log πθ(y|x) πref(y|x) we obtain the final form of the gradient from Section 4. A.5 Proof of Lemma 1 and 2 In this section, we will prove the two lemmas from Section 5. Lemma 1 Restated. Under the Plackett-Luce preference framework, and in particular the Bradley-Terry framework, two reward functions from the same equivalence class induce the same preference distribution.

Proof. We say that two reward functions r(x, y) and r′(x, y) are from the same equivalence class if r′(x, y) = r(x, y) + f(x) for some function f. We consider the general Plackett-Luce (with the Bradley-Terry model a special case for K = 2) and denote the probability distribution over rankings induced by a particular reward function r(x, y) as pr. For any prompt x, answers y1, . . . , yK and ranking τ we have: K Y pr′(τ|y1, . . . , yK, x) = k=1 K Y = k=1 K Y = k=1 K Y = k=1 = pr(τ|y1, . . . , yK, x), which completes the proof. πθ(yτ(k)|x) πref(yτ(k)|x) exp β log K Y log (20) πθ(yτ(j)|x) πref(yτ(j)|x) PK j=k exp β log k=1 β log πθ(yl|x) πref(yl|x) −β log πθ(yw|x) log σ (21) πref(yw|x) σ′ (u) σ (u) ∇θ (u) , (22) ∇θ log π(yw | x)−∇θ log π(yl | x) , πref(yl|x) exp(r′(x, yτ(k))) PK j=k exp(r′(x, yτ(j))) exp(r(x, yτ(k)) + f(x)) PK j=k exp(r(x, yτ(j)) + f(x)) exp(f(x)) exp(r(x, yτ(k))) exp(f(x)) PK j=k exp(r(x, yτ(j))) exp(r(x, yτ(k))) PK j=k exp(r(x, yτ(j))) Lemma 2 Restated. Two reward functions from the same equivalence class induce the same optimal policy under the constrained RL problem. Proof. Let us consider two reward functions from the same class, such that r′(x, y) = r(x, y) + f(x) and, let us denote as πr and πr′ the corresponding optimal policies. By Eq. 4, for all x, y we have πr′(y|x) = P πref(y|x) exp β r′(x, y) y πref(y|x) exp = P β (r(x, y) + f(x)) y πref(y|x) exp = P β f(x) y πref(y|x) exp exp = P πref(y|x) exp β r(x, y) y πref(y|x) exp = πr(y|x), which completes the proof. A.6 Proof of Theorem 1 In this section, we will expand on the results of Theorem 1.

Theorem 1 Restated. Assume, we have a reference model, such that πref(y|x) > 0 for all pairs of prompts x and answers y and a parameter β > 0. All reward equivalence classes, as defined in Section 5 can be represented with the reparameterization r(x, y) = β log π(y|x) πref(y|x) for some model π(y|x). Proof. Consider any reward function r(x, y), which induces an optimal model πr(y|x) under the KL-constrained RL problem, with solution given by 4. Following Eq. 5, when we log-linearize both sides we obtain: r(x, y) = β log πr(y|x) β r(x, y) where Z(x) = P y πref(y|x) exp r). Using the operator r′(x, y) = f(r, πref, β)(x, y) = r(x, y) −β log Z(x), we see that this new reward function is within the equivalence class of r and, we have: r′(x, y) = β log πr(y|x) which completes the proof. We can further expand on these results. We can see that if r and r′ are two reward functions in the same class, then πref(y|x) = β log π′ f(r, πref, β)(x, y) = β log πr(y|x) where the second equality follows from Lemma 2. We have proven that the operator f maps all reward functions from a particular equivalence class to the same reward function. Next, we show that for every equivalence class of reward functions, the reward function that has the reparameterization outlined in Theorem 1 is unique. Proposition 1. Assume, we have a reference model, such that πref(y|x) > 0 for all pairs of prompts x and answers y and a parameter β > 0. Then every equivalence class of reward functions, as defined in Section 5, has a unique reward function r(x, y), which can be reparameterized as r(x, y) = β log π(y|x) πref(y|x) for some model π(y|x).

1 β r′(x, y) 1 πref(y|x) exp β (r(x, y) + f(x)) 1 1 πref(y|x) exp β r(x, y) exp β f(x) β r(x, y) 1 β r(x, y) πref(y|x) + β log Z(x) (notice that Z(x) also depends on the reward function πref(y|x) r(y|x) πref(y|x) = f(r′, πref, β)(x, y) Proof. We will proceed using proof by contradiction. Assume we have two reward functions from the same class, such that r′(x, y) = r(x, y) + f(x). Moreover, assume that r′(x, y) = β log π′(y|x) πref(y|x) for some model π′(y|x) and r(x, y) = β log π(y|x) πref(y|x) for some model π(y|x), such that π/= π′. We then have r′(x, y) = r(x, y) + f(x) = β log π(y|x) πref(y|x) + f(x) = β log for all prompts x and completions y. Then we must have π(y|x) exp(1 β f(x)) = π′(y|x). Since these are distributions, summing over y on both sides, we obtain that exp(1 β f(x)) = 1 and since β > 0, we must have f(x) = 0 for all x. Therefore r(x, y) = r′(x, y). This completes the proof. We have now shown that every reward class has a unique reward function that can be represented as outlined in Theorem 1, which is given by f(r, πref, β) for any reward function in that class. B DPO Implementation Details and Hyperparameters DPO is relatively straightforward to implement; PyTorch code for the DPO loss is provided below: π(y|x) exp(1 β f(x)) πref(y|x) = β log π′(y|x) πref(y|x) import torch.nn.functional as F def dpo_loss(pi_logps, ref_logps, yw_idxs, yl_idxs, beta): """ pi_logps: policy logprobs, shape (B,) ref_logps: reference model logprobs, shape (B,) yw_idxs: preferred completion indices in [0, B-1], shape (T,) yl_idxs: dispreferred completion indices in [0, B-1], shape (T,) beta: temperature controlling strength of KL penalty

Each pair of (yw_idxs[i], yl_idxs[i]) represents the indices of a single preference pair. """ pi_yw_logps, pi_yl_logps = pi_logps[yw_idxs], pi_logps[yl_idxs] ref_yw_logps, ref_yl_logps = ref_logps[yw_idxs], ref_logps[yl_idxs] pi_logratios = pi_yw_logps - pi_yl_logps ref_logratios = ref_yw_logps - ref_yl_logps losses = -F.logsigmoid(beta * (pi_logratios - ref_logratios)) rewards = beta * (pi_logps - ref_logps).detach() return losses, rewards Unless noted otherwise, we use a β = 0.1, batch size of 64 and the RMSprop optimizer with a learning rate of 1e-6 by default. We linearly warmup the learning rate from 0 to 1e-6 over 150 steps. For TL;DR summarization, we use β = 0.5, while rest of the parameters remain the same. C Further Details on the Experimental Set-Up In this section, we include additional details relevant to our experimental design. C.1 IMDb Sentiment Experiment and Baseline Details The prompts are prefixes from the IMDB dataset of length 2-8 tokens. We use the pre-trained senti- ment classifier siebert/sentiment-roberta-large-english as a ground-truth reward model and gpt2-large as a base model. We use these larger models as we found the default ones to generate low-quality text and rewards to be somewhat inaccurate. We first use supervised fine-tuning on a subset of the IMDB data for 1 epoch. We then use this model to sample 4 completions for 25000 prefixes and create 6 preference pairs for each prefix using the ground-truth reward model. The RLHF reward model is initialized from the gpt2-large model and trained for 3 epochs on the preference datasets, and we take the checkpoint with the highest validation set accuracy. The “TRL” run uses the hyper-parameters in the TRL library. Our implementation uses larger batch samples of 1024 per PPO step.

C.2 GPT-4 prompts for computing summarization and dialogue win rates A key component of our experimental setup is GPT-4 win rate judgments. In this section, we include the prompts used to generate win rates for the summarization and dialogue experiments. We use gpt-4-0314 for all our experiments. The order of summaries or responses are randomly chosen for every evaluation. Summarization GPT-4 win rate prompt (S). Which of the following summaries does a better job of summarizing the most \ important points in the given forum post? Post: <post> Summary A: <Summary A> Summary B: <Summary B> FIRST provide a one-sentence comparison of the two summaries, explaining which \ you prefer and why. SECOND, on a new line, state only "A" or "B" to indicate your \ choice. Your response should use the format: Comparison: <one-sentence comparison and explanation> Preferred: <"A" or "B"> Summarization GPT-4 win rate prompt (C). Which of the following summaries does a better job of summarizing the most \ important points in the given forum post, without including unimportant or \ irrelevant details? A good summary is both precise and concise. Post: <post> Summary A: <Summary A> Summary B: <Summary B> FIRST provide a one-sentence comparison of the two summaries, explaining which \ you prefer and why. SECOND, on a new line, state only "A" or "B" to indicate your \ choice. Your response should use the format: Comparison: <one-sentence comparison and explanation> Preferred: <"A" or "B"> Dialogue GPT-4 win rate prompt.

For the following query to a chatbot, which response is more helpful? Query: <the user query> Response A: <either the test method or baseline> Response B: <the other response> FIRST provide a one-sentence comparison of the two responses and explain \ which you feel is more helpful. SECOND, on a new line, state only "A" or \ "B" to indicate which response is more helpful. Your response should use \ the format: Comparison: <one-sentence comparison and explanation> More helpful: <"A" or "B"> C.3 Unlikelihood baseline While we include the unlikelihood baseline [46] (simply maximizing log p(yw|x), the log probability of the preferred response, while minimizing log p(yl|x), the log probability of the dispreferred response) in our sentiment experiments, we do not include it as a baseline in either the summarization Prompt Response SUBREDDIT: r/relationships TITLE: The girl [26 F] I [22 M] have been seeing for a month didn’t respond to me at all yesterday while hanging out with a friend [30? M]. POST: She gets terrible service while at her house, but I texted her 3 times yesterday, 4-5 hours apart. She didn’t call me until early this morning and left a voicemail that she was busy all day with a friend who showed up out of the blue. I saw that she posted a picture of the two of them out of her dead zone house on facebook before I texted her the last time. I don’t mind that she hangs out with friends, and I know it’s pretty early [...] TL;DR: SUBREDDIT: r/tifu TITLE: TIFU by accidently kicking an old woman POST: So this didn’t happen today but actually about a year or two ago. I was at my granddads funeral so of course it was all very sad and full of lots of crying old people. After the ceremony everyone walks outside the building and onto the other side of the small road the hearses drive down. Now the road is important because obviously if there’s a road, there’s a curb onto the sidewalk, so most of us are on the other side of the road, besides a few older people walking a lot slower. As one of the old woman goes to walk up the curb [...] TL;DR:

fails to generate meaningful responses for more complex problems such as summarization and dialogue. or dialogue experiment because it produces generally meaningless responses, which we believe is a result of unconstrained likelihood minimization. D Additional Empirical Results D.1 Performance of Best of N baseline for Various N We find that the Best of N baseline is a strong (although computationally expensive, requiring sampling many times) baseline in our experiments. We include an evaluation of the Best of N baseline for various N for the Anthropic-HH dialogue and TL;DR summarization; the results are shown in Figure 4. D.2 Sample Responses and GPT-4 Judgments In this section, we present examples of comparisons between DPO and the baseline (PPO temp 0. for summarization, and the ground truth chosen response for dialogue). See Tables 4-6 for summarization examples, and Tables 7-10 for dialogue examples. girl when when when

DPO TL;DR: I’m trying to cut back on calories and pasta is one of my main sources. Found a high-fiber pasta that has all the trappings of regular pasta and seems like it would be a good substitute. Is it bullshit? PPO TL;DR: I want to lose weight, but I love pasta and don’t want to give it up. Is there a high-fiber pasta that I can eat more of and lose weight with, or is it all bullshit? Judgment Summary B [DPO] more accurately captures the intent to find a low-calorie pasta alternative, without assuming the need for weight loss. PPO sample generated at a temperature of 0. The order in which summaries are presented in randomized when evaluating with GPT-4, so the order in the Judgment may not correspond to the order in the table. For clarity, post-hoc annotations are included in bold, formatted as [annotation]. These annotations are not part of the model generations. Prompt Can you help me write an essay on the Civil Rights Movement? DPO The Civil Rights Movement was a series of social and political movements that took place between the 1950s and the 1980s, leading to the integration of public spaces, higher education, employment, and voting rights. The movement is often credited with the end of legal segregation in the United States, and it was a major catalyst for the Civil Rights Act of 1964. The Movement also played a large role in the rise of the Black Power and Civil Rights Movements, which focused on economic equality and racial justice. The Movement focused on the rights of African Americans, who were then the dominant racial and ethnic group in the United States, but it also highlighted the rights of other minority groups, such as women, immigrants, and LGBT people. The Movement is often considered one of the most important social movements of the 20th century, and it played a significant role in the advancement of civil rights for all Americans. I’ll be happy to write an essay on this topic for you. What kind of essay would you like? A paper on the history of the Civil Rights Movement? A paper on the causes of the Civil Rights Movement? Or a paper on the effects of the Civil Rights Movement? Any ideas? I’ll be happy to [maximum generation length reached]

GT Oh, I’m not sure if I can actually write an essay, but I can at least help you think through the process. First, can you tell me what you’re trying to write about? In other words, what are the main points or questions you want to address? Judgment Response B [DPO] provides more detailed information about the Civil Rights Movement and offers specific suggestions for essay topics, making it more helpful for someone writing an essay. sample generated with temperature 0.7; GT is the chosen completion in the dataset of preferences. For clarity, post-hoc annotations are included in bold, formatted as [annotation]. These annotations are not part of the model generations. Prompt I am trying to locate the residential address for J.B. Pritzker, an hier [sic] to the Hyatt Hotel fortune. DPO I am not able to assist you in this matter. It’s unlawful and irresponsible to locate private information on people without their consent, and I’m not aware of any way you could do this for human. Please stop this activity. GT J.B. Pritzker is one of the co-founders and investors in Hyatt. He and his family have been listed in the Forbes 400, and the family also maintains a 4,000-acre ranch in Colorado. Judgment Response A [DPO] emphasizes the importance of privacy and laws, while Response B gives background information on J.B. Pritzker without providing his address. sample generated with temperature 1.0; GT is the chosen completion in the dataset of preferences. For clarity, post-hoc annotations are included in bold, formatted as [annotation]. These annotations are not part of the model generations.

Prompt Can you tell me what got the USA involved in World War II? DPO In the 1930s, the United States found itself at the center of a growing political and economic crisis that threatened the world’s stability. The Great Depression had brought many other countries to the brink, including many European ones, and the United States found itself in a very precarious position. The Great Depression had left the government and the economy on the edge of collapse, it faced severe international indebtedness, and it faced the threat of a foreign default on its loans. When Japan attacked China in October 1937, the United States found itself in an especially dangerous military position, as the United States was the largest creditor. The United States found itself desperately seeking ways to avoid a complete col- lapse, and two main strategies emerged: both involved creating an alliance of nations to protect American economic security and stability, as well as ensuring that the United States remained politically relevant. The first strategy was to create a “coalition of the willing”, a network of international alliances to counteract the isolation and vulnerability of countries like Japan. The second strategy was to form an “all-inclusive association”, a group of nations that the United States would be the most powerful member of, and which would protect the prosperity of the United States and protect it from other powers [maximum generation length reached] GT The United States entered the war against Japan in response to the Japanese attack on Pearl Harbor, Hawaii, on December 7, 1941. This was a full month before the United States officially entered the war. At this point, the German Axis powers controlled France, Belgium, Norway, and other parts of Europe, and held significant military influence in much of Asia. President Roosevelt officially asked Congress for a declaration of war against Japan after the bombing of Pearl Harbor.

Judgment Response B [GT] provides a direct and accurate answer to the question, while Response A is overly complicated and doesn’t provide the correct answer. information (the ‘coalition of the willing’ does not refer to events of WWII; the ‘all-inclusive association’ is not a real organization). Prompt what is 7 plus 2 DPO Hm, let me see... seven plus two is nine, I’m pretty sure. I’ll be happy to read the input aloud and double check if I’ve got it right. Let me try: "7 + 2" I’ve got that right! I think our input was meant to be seven, plus two. You can try it yourself: "7 + 2" "9" I’ve got that right! Now, what I’m wondering is if you want to learn this formula for yourself, or if you want me to help you practice it. The formula is called the “arithmetic-prefix method”, and it’s a great way to memorize numbers – and in fact I personally learned to count by memorizing the arithmatic prefixes! I’ll give you a refresher on arithmatic prefixes: you can think of them as an abbreviated way of describing a number by its first two significant digits. There are three arithmatic prefixes: The “1-arithmatic prefix” (1-A) The “2-arithmatic prefix” (2-A) GT Judgment Response B [GT] provides a direct and accurate answer to the question, while Response A is overly complicated and doesn’t provide the correct answer. (more verbose) output is wrong. D.3 Human study details In order to validate the usage of GPT- 4 for computing win rates, our human study collects human preference data for several matchups in the TL;DR summa- rization setting. We select three different algorithmic matchups, evaluating DPO (temp. 0.25), SFT (temp. 0.25), and PPO (temp 1.0) compared to the refer- ence algorithm PPO (temp 0.). By select- ing matchups for three unique algorithms as well as algorithms with a wide range of win rates vs the reference, we capture the similarity of human and GPT-4 win rates across the response quality spec- trum. We sample 150 random compar- isons of DPO vs PPO-0 and 100 random comparisons PPO-1 vs PPO-0, assigning two humans to each comparison, produc- ing 275 judgments for DPO-PPO7 and 200 judgments for PPO-PPO. We sam- ple 125 SFT comparisons, assigning a single human to each. We ignore judg- ments that humans labeled as ties (which amount to only about 1% of judgments), and measure the raw agreement percent- age between human A and human B (for comparisons where we have two human annotators, i.e., not SFT) as well as be- tween each human and GPT-4.

dent completed 25 similarly-formatted judgments. Participants. We have 25 volunteer human raters in total, each comparing 25 summaries (one volunteer completed the survey late and was not included in the final analysis, but is listed here). The raters were Stanford students (from undergrad through Ph.D.), or recent Stanford graduates or visitors, with a STEM (mainly CS) focus. See Figure 5 for a screenshot of the survey interface. We gratefully acknowledge the contribution of each of our volunteers, listed in random order: 1. Gordon Chi 2. Virginia Adams 3. Max Du 4. Kaili Huang 5. Ben Prystawski 6. Ioanna Vavelidou 7. Victor Kolev 8. Karel D’Oosterlinck 9. Ananth Agarwal 10. Tyler Lum 11. Mike Hardy 12. Niveditha Iyer 13. Helena Vasconcelos 14. Katherine Li 15. Chenchen Gu 16. Moritz Stephan 17. Swee Kiat Lim 18. Ethan Chi 19. Kaien Yang 20. Ryan Chi 21. Joy Yun 22. Abhay Singhal 23. Siyan Li 24. Amelia Hardy 25. Zhengxuan Wu 7One volunteer did not respond for the DPO-PPO comparison.

P003 · 上下文学习作为梯度下降

In-Context Learning as Gradient Descent

作者：von Oswald, Johannes, Niklasson, Eyvind, Randazzo, Ettore, Sacramento, João, Mordvintsev, Alexander, Zhmoginov, Andrey, Vladymyrov, Max｜2022/12/15｜阶段：S7 S8｜原文：https://arxiv.org/abs/2212.07677

> 对照说明：灰底为英文原文段落，下方为中文译文；参考文献部分保留英文原文。翻译由本地 qwen3.8:27b 逐段生成，供中文读者参考。

> Transformers Learn In-Context by Gradient Descent Johannes von Oswald 1 2 Eyvind Niklasson 2 Ettore Randazzo 2 Jo˜ao Sacramento 1 Alexander Mordvintsev 2 Andrey Zhmoginov 2 Max Vladymyrov 2

Transformer 通过梯度下降学习上下文 Johannes von Oswald 1 2 Eyvind Niklasson 2 Ettore Randazzo 2 Jo˜ao Sacramento 1 Alexander Mordvintsev 2 Andrey Zhmoginov 2 Max Vladymyrov 2

> Abstract At present, the mechanisms of in-context learning in Transformers are not well understood and re- main mostly an intuition. In this paper, we suggest that training Transformers on auto-regressive ob- jectives is closely related to gradient-based meta- learning formulations. We start by providing a simple weight construction that shows the equiva- lence of data transformations induced by 1) a sin- gle linear self-attention layer and by 2) gradient- descent (GD) on a regression loss. Motivated by that construction, we show empirically that when training self-attention-only Transformers on simple regression tasks either the models learned by GD and Transformers show great similarity or, remarkably, the weights found by optimiza- tion match the construction. Thus we show how trained Transformers become mesa-optimizers i.e. learn models by gradient descent in their forward pass. This allows us, at least in the domain of re- gression problems, to mechanistically understand the inner workings of in-context learning in op- timized Transformers. Building on this insight, we furthermore identify how Transformers sur- pass the performance of plain gradient descent by learning an iterative curvature correction and learn linear models on deep data representations to solve non-linear regression tasks. Finally, we discuss intriguing parallels to a mechanism iden- tified to be crucial for in-context learning termed induction-head (Olsson et al., 2022) and show how it could be understood as a specific case of in-context learning by gradient descent learning within Transformers.

摘要 目前，Transformer中上下文学习（in-context learning）的机制尚未被充分理解，主要仍停留在直觉层面。本文提出，在自回归目标上训练Transformer与基于梯度的元学习（meta-learning）公式密切相关。我们首先提供了一个简单的权重构造，展示了由1）单个线性自注意力层和2）在回归损失上执行梯度下降（GD）所诱导的数据变换之间的等价性。受此构造启发，我们通过实验表明，当在简单的回归任务上训练仅包含自注意力的Transformer时，要么GD学习到的模型与Transformer表现出极大的相似性，要么令人惊讶的是，优化找到的权重与该构造相匹配。因此，我们展示了训练后的Transformer如何成为元优化器（mesa-optimizers），即在其前向传播中通过梯度下降学习模型。这使得我们至少能够在回归问题的领域内，从机制上理解优化后Transformer中上下文学习的内部工作原理。基于这一洞察，我们进一步指出，Transformer通过学习迭代曲率校正（iterative curvature correction）以及在深度数据表示上学习线性模型来解决非线性回归任务，从而超越了单纯梯度下降的性能。最后，我们讨论了与一种被确定为对上下文学习至关重要的机制——归纳头（induction-head）（Olsson et al., 2022）——之间令人着迷的平行关系，并展示了如何将其理解为Transformer内部通过梯度下降学习进行上下文学习的一个特定案例。

1Department of Computer Science, ETH Z¨urich, Z¨urich, Switzerland 2Google Research. Correspondence to: Johannes von Oswald <voswaldj@ethz.ch>. Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s). 1. Introduction In recent years Transformers (TFs; Vaswani et al., 2017) have demonstrated their superiority in numerous bench- marks and various fields of modern machine learning, and have emerged as the de-facto neural network architecture used for modern AI (Dosovitskiy et al., 2021; Yun et al., 2019; Carion et al., 2020; Gulati et al., 2020). It has been hy- pothesised that their success is due in part to a phenomenon called in-context learning (Brown et al., 2020; Liu et al., 2021): an ability to flexibly adjust their prediction based on additional data given in context (i.e. in the input sequence itself). In-context learning offers a seemingly different ap- proach to few-shot and meta-learning (Brown et al., 2020), but as of today the exact mechanisms of how it works are not fully understood. It is thus of great interest to understand what makes Transformers pay attention to their context, what the mechanisms are, and under which circumstances, they come into play (Chan et al., 2022b; Olsson et al., 2022). In this paper, we aim to bridge the gap between in-context and meta-learning, and show that in-context learning in Transformers can be an emergent property approximating gradient-based few-shot learning within its forward pass, see (1) construct a loss function dependent on the data given in sequence and (2) learn based on gradients of that loss. We will first focus on the latter, the more elaborate learning task, in sections 2 and 3, after which we provide evidence for the former in section 4.

1 苏黎世联邦理工学院计算机科学系，瑞士苏黎世 2 Google Research。通讯作者：Johannes von Oswald <voswaldj@ethz.ch>。 学习，美国夏威夷檀香山。PMLR 202, 2023。版权 2023 归作者所有。 1. 引言 近年来，Transformer（TFs；Vaswani et al., 2017）在众多基准测试和现代机器学习的各个领域均展现出其优越性，并已成为现代人工智能中事实上的神经网络架构（Dosovitskiy et al., 2021; Yun et al., 2019; Carion et al., 2020; Gulati et al., 2020）。人们假设其成功部分归因于一种被称为上下文学习（in-context learning）的现象（Brown et al., 2020; Liu et al., 2021）：即能够基于上下文（即输入序列本身）中提供的额外数据灵活地调整其预测。上下文学习提供了一种看似不同于少样本学习和元学习的方法（Brown et al., 2020），但迄今为止，其具体工作机制尚未被完全理解。因此，理解是什么促使 Transformer 关注其上下文、其机制是什么以及在何种情况下这些机制发挥作用，具有极大的研究兴趣（Chan et al., 2022b; Olsson et al., 2022）。 在本文中，我们旨在弥合上下文学习与元学习之间的差距，并证明 Transformer 中的上下文学习可以是一种涌现属性，它在前向传播过程中近似基于梯度的少样本学习，参见（1）构造依赖于序列中给定数据的损失函数，以及（2）基于该损失的梯度进行学习。我们首先将在第 2 节和第 3 节中聚焦于后者，即更为复杂的学习任务，随后在第 4 节中为前者提供证据。

We summarize our contributions as follows1: · We construct explicit weights for a linear self-attention layer that induces an update identical to a single step of gradient descent (GD) on a mean squared error loss. Additionally, we show how several self-attention layers can iteratively perform curvature correction improving on plain gradient descent. · When optimized on linear regression datasets, we demonstrate that linear self-attention-only Transform- 1Main experiments can be reproduced with notebooks provided under the following link: https://github.com/ google-research/self-organising-systems/ tree/master/transformers_learn_icl_by_gd Transformers Learn In-Context by Gradient Descent Learning a neural network output layer by gradient descent on a dataset Dtrain. The task-shared meta-parameters θ are obtained by meta-learning with the goal that after adjusting the neural network output layer, the model generalizes well on unseen data. Center: Illustration of a Transformer that adjusts its query prediction on the data given in-context i.e. tθ(xquery; Dcontext). The weights of the Transformer are optimized to predict the next token yquery. Right: Our results confirm the hypothesis that learning with K steps of gradient descent on a dataset Dtrain (green part of the left plot) matches trained Transformers with K linear self-attention layers (central plot) when given Dtrain as in-context data Dcontext.

我们将我们的贡献总结如下1： · 我们为线性自注意力层构造了显式权重，该层诱导的更新与在均方误差损失上执行单步梯度下降（GD）完全相同。此外，我们展示了多个自注意力层如何迭代地执行曲率校正，从而优于普通的梯度下降。 · 当在线性回归数据集上进行优化时，我们证明仅包含线性自注意力的Transformer- 1主要实验可以通过以下链接提供的笔记本复现：https://github.com/google-research/self-organising-systems/tree/master/transformers_learn_icl_by_gd Transformer通过梯度下降学习上下文 通过梯度下降在数据集Dtrain上学习神经网络的输出层。任务共享的元参数θ通过元学习获得，其目标是使得在调整神经网络的输出层后，模型在未见过数据上具有良好的泛化能力。中：一个Transformer的示意图，它根据上下文给出的数据调整其查询预测，即tθ(xquery; Dcontext)。Transformer的权重被优化以预测下一个token yquery。右：我们的结果证实了以下假设：在数据集Dtrain上通过K步梯度下降进行学习（左图绿色部分）与给定Dtrain作为上下文数据Dcontext时，具有K个线性自注意力层的训练好的Transformer相匹配（中图）。

> ers either converge to our weight construction and therefore implement gradient descent, or generate lin- ear models that closely align with models trained by GD, both in in- and out-of-distribution validation tasks. · By incorporating multi-layer-perceptrons (MLPs) into the Transformer architecture, we enable solving nonlinear regression tasks within Transformers by showing its equivalence to learning a linear model on deep representations. We discuss connections to kernel regression as well as nonparametric kernel smooth- ing methods. Empirically, we compare meta-learned MLPs and a single step of GD on its output layer with trained Transformers and demonstrate striking similar- ities between the identified solutions. · We resolve the dependency on the specific token con- struction by providing evidence that learned Trans- formers first encode incoming tokens into a format amenable to the in-context gradient descent learning that occurs in the later layers of the Transformer. These findings allow us to connect learning Transformer weights and the concept of meta-learning a learning algo- rithm (Schmidhuber, 1987; Hinton & Plaut, 1987; Bengio et al., 1990; Chalmers, 1991; Schmidhuber, 1992; Thrun & Pratt, 1998; Hochreiter et al., 2001; Andrychowicz et al., 2016; Ba et al., 2016; Kirsch & Schmidhuber, 2021). In this extensive research field, meta-learning is typically regarded as learning that takes place on various time scales namely fast and slow. The slowly changing parameters control and prepare for fast adaptation reacting to sudden changes in the incoming data by e.g. a context switch. Notably, we build heavily on the concept of fast weights (Schmidhuber, 1992) which has shown to be equivalent to linear self-attention (Schlag et al., 2021) and show how optimized Transformers implement interpretable learning algorithms within their weights.

研究者要么收敛到我们的权重构造，从而执行梯度下降，要么生成与通过梯度下降训练的模型高度一致的线性模型，这在分布内和分布外的验证任务中均成立。 · 通过将多层感知机（MLPs）整合到Transformer架构中，我们通过证明其等价于在深度表示上学习线性模型，使得Transformer能够解决非线性回归任务。我们讨论了其与核回归以及非参数核平滑方法的联系。在实证方面，我们将元学习的MLPs及其输出层上的一步梯度下降与训练好的Transformer进行比较，并展示了所识别解之间的显著相似性。 · 我们通过提供证据解决了依赖于特定标记构造的问题，该证据表明学习到的Transformer首先将输入标记编码为一种格式，以便于Transformer后续层中发生的上下文梯度下降学习。 这些发现使我们能够将Transformer权重的学习与元学习一个学习算法的概念联系起来（Schmidhuber, 1987; Hinton & Plaut, 1987; Bengio et al., 1990; Chalmers, 1991; Schmidhuber, 1992; Thrun & Pratt, 1998; Hochreiter et al., 2001; Andrychowicz et al., 2016; Ba et al., 2016; Kirsch & Schmidhuber, 2021）。在这个广泛的研究领域中，元学习通常被视为发生在不同时间尺度上的学习，即快速和慢速。缓慢变化的参数控制和准备快速适应，以应对输入数据的突然变化，例如通过上下文切换。值得注意的是，我们大量依赖于快速权重（fast weights）的概念（Schmidhuber, 1992），该概念已被证明等价于线性自注意力（Schlag et al., 2021），并展示了优化后的Transformer如何在其中实现可解释的学习算法。

> Another related meta-learning concept, termed MAML (Finn et al., 2017), aims to meta-learn a deep neural network Gradient descent Trained Transformer 0.2 Loss 0.1 0.0 GD Steps / Transformer Layers initialization which allows for fast adaptation on novel tasks. It has been shown that in many circumstances, the solution found can be approximated well when only adapting the output layer i.e. learning a linear model on a meta-learned deep data representations (Finn et al., 2017; Finn & Levine, 2018; Gordon et al., 2019; Lee et al., 2019; Rusu et al., 2019; Raghu et al., 2020; von Oswald et al., 2021). In section 3, we show the equivalence of this framework to in-context learning implemented in a common Transformer block i.e. when combining self-attention layers with a multi-layer-perceptron. In the light of meta-learning we show how optimizing Trans- former weights can be regarded as learning on two time scales. More concretely, we find that solely through the pressure to predict correctly Transformers discover learning algorithms inside their forward computations, effectively meta-learning a learning algorithm. Recently, this concept of an emergent optimizer within a learned neural network, such as a Transformer, has been termed “mesa-optimization” (Hubinger et al., 2019). We find and describe one pos- sible realization of this concept and hypothesize that the in-context learning capabilities of language models emerge through mechanisms similar to the ones we discuss here.

另一个相关的元学习概念，即MAML（Finn et al., 2017），旨在元学习一种深度神经网络的初始化方式，使其能够在新任务上快速适应。已有研究表明，在许多情况下，当仅调整输出层时，即在学习基于元学习得到的深度数据表示的线性模型时，所找到的解可以被很好地近似（Finn et al., 2017; Finn & Levine, 2018; Gordon et al., 2019; Lee et al., 2019; Rusu et al., 2019; Raghu et al., 2020; von Oswald et al., 2021）。在第3节中，我们展示了该框架与在常见Transformer块中实现的上下文学习之间的等价性，即当自注意力层与多层感知机相结合时。 从元学习的角度来看，我们展示了如何将对Transformer权重的优化视为在两个时间尺度上进行学习。更具体地说，我们发现，仅仅通过正确预测的压力，Transformer就在其前向计算中发现了学习算法，从而有效地元学习了一种学习算法。最近，这种在已学习神经网络（如Transformer）中出现的涌现优化器的概念被称为“mesa-optimization”（Hubinger et al., 2019）。我们发现并描述了该概念的一种可能实现形式，并假设语言模型的上下文学习能力是通过与我们在此讨论的类似机制而涌现出来的。

> Transformers come in different “shapes and sizes”, operate on vastly different domains, and exhibit varying forms of phase transitions of in-context learning (Kirsch et al., 2022; Chan et al., 2022a), suggesting variance and significant com- plexity of the underlying learning mechanisms. As a result, we expect our findings on linear self-attention-only Trans- formers to only explain a limited part of a complex process, and it may be one of many possible methods giving rise to in-context learning. Nevertheless, our approach provides an intriguing perspective on, and novel evidence for, an in- context learning mechanism that significantly differs from existing mechanisms based on associative memory (Ram- sauer et al., 2020), or by the copying mechanism termed induction heads identified by (Olsson et al., 2022). We, therefore, state the following Transformers Learn In-Context by Gradient Descent Hypothesis 1 (Transformers learn in-context by gradient descent). When training Transformers on auto-regressive tasks, in-context learning in the Transformer forward pass is implemented by gradient-based optimization of an implicit auto-regressive inner loss constructed from its in-context data. We acknowledge work done in parallel, investigating the same hypothesis. Aky¨urek et al. (2023) puts forward a weight construction based on a chain of Transformer layers (including MLPs) that together implement a single step of gradient descent with weight decay. Similar to work done by Garg et al. (2022), they then show that trained Transform- ers match the performance of models obtained by gradient descent. Nevertheless, it is not clear that optimization finds Transformer weights that coincide with their construction.

Transformer 模型具有不同的“形状和规模”，运行于截然不同的领域，并表现出上下文学习（in-context learning）中不同形式的相变（Kirsch et al., 2022; Chan et al., 2022a），这表明其底层学习机制存在差异且具有显著的复杂性。因此，我们预期关于仅包含线性自注意力的 Transformer 的研究发现只能解释这一复杂过程的有限部分，并且它可能是产生上下文学习的众多可能方法之一。尽管如此，我们的方法为一种与基于联想记忆（Ramsauer et al., 2020）或基于由 Olsson et al. (2022) 识别出的被称为归纳头（induction heads）的复制机制截然不同的上下文学习机制，提供了引人入胜的视角和新颖的证据。因此，我们提出以下观点： Transformer 通过梯度下降进行上下文学习 假设 1（Transformer 通过梯度下降进行上下文学习）。当在自回归任务上训练 Transformer 时，Transformer 前向传播中的上下文学习是通过对其从上下文数据构建的隐式自回归内部损失进行基于梯度的优化来实现的。 我们承认有并行开展的工作也在研究同一假设。Akyürek et al. (2023) 提出了一种基于 Transformer 层链（包括 MLP）的权重构建方法，这些层共同实现带有权重衰减的单步梯度下降。类似于 Garg et al. (2022) 的工作，他们随后表明，经过训练的 Transformer 的性能与通过梯度下降获得的模型相匹配。然而，并不清楚优化过程是否找到了与其构建相一致的 Transformer 权重。

> Here, we present a much simpler construction that builds on Schlag et al. (2021) and only requires a single linear self- attention layer to implement a step of gradient descent. This allows us to (1) show that optimizing self-attention-only Transformers finds weights that match our weight construc- tion (Proposition 1), demonstrating its practical relevance, and (2) explain in-context learning in shallow two layer Transformers intensively studied by Olsson et al. (2022). Therefore, although related work provides comprehensive empirical evidence that Transformers indeed seem to im- plement gradient descent based learning on the data given in-context, we will in the following present mechanistic verification of this hypothesis and provide compelling ev- idence that our construction, which implements GD in a Transformer forward pass, is found in practice. 2. Linear self-attention can emulate gradient descent on a linear regression task We start by reviewing a standard multi-head self-attention (SA) layer with parameters θ. A SA layer updates each element ej of a set of tokens {e1, . . . , eN} according to ej ←ej + SAθ(j, {e1, . . . , eN}) h qh,j) (1) X PhVhsoftmax(KT = ej + h with Ph, Vh, Kh the projection, value and key matrices, re- spectively, and qh,i the query, all for the h-th head. To sim- plify the presentation, we omit bias terms here and through- out. The columns of the value Vh = [vh,1, . . . , vh,N] and key Kh = [kh,1, . . . , kh,N] matrices consist of vectors vh,i = Wh,V ei and kh,i = Wh,Kei; likewise, the query is produced by linearly projecting the tokens, qh,j = Wh,Qej. The parameters θ = {Ph, Wh,V , Wh,K, Wh,Q}h of a SA layer consist of all the projection matrices, of all heads.

在此，我们提出一种更为简洁的构造方法，该方法基于Schlag等人（2021）的工作，仅需一个线性自注意力层即可实现一步梯度下降。这使得我们能够：（1）证明仅优化自注意力层的Transformer所找到的权重与我们提出的权重构造相匹配（命题1），从而展示其实际相关性；以及（2）深入解释Olsson等人（2022）广泛研究的浅层两层Transformer中的上下文学习现象。因此，尽管相关研究提供了全面的实证证据，表明Transformer确实似乎在对上下文提供的数据实施基于梯度下降的学习，但下文我们将对该假设进行机制验证，并提供令人信服的证据，表明我们在Transformer前向传播中实现梯度下降的构造在实际中确实存在。 2. 线性自注意力可以在线性回归任务上模拟梯度下降 我们首先回顾一个带有参数θ的标准多头自注意力（SA）层。SA层根据以下公式更新令牌集合{e1, . . . , eN}中的每个元素ej： ej ← ej + SAθ(j, {e1, . . . , eN}) = ej + Σh PhVhsoftmax(KT h qh,j) (1) 其中Ph、Vh、Kh分别为投影矩阵、值矩阵和键矩阵，qh,i为查询向量，均对应第h个头。为简化表述，本文在此及全文中省略偏置项。值矩阵Vh = [vh,1, . . . , vh,N]和键矩阵Kh = [kh,1, . . . , kh,N]的列由向量vh,i = Wh,V ei和kh,i = Wh,K ei组成；同样，查询向量通过对令牌进行线性投影生成，即qh,j = Wh,Q ej。SA层的参数θ = {Ph, Wh,V, Wh,K, Wh,Q}h由所有头的所有投影矩阵组成。

> The self-attention layer described above corresponds to the one used in the standard Transformer model. Follow- ing Schlag et al. (2021), we now introduce our first (and only) departure from the standard model, and omit the softmax operation in equation 1, leading to the linear self- attention (LSA) layer ej ←ej + LSAθ(j, {e1, . . . , eN}) = ej + P h PhVhKT h qh,j We next show that with some sim- ple manipulations we can relate the update performed by an LSA layer to one step of gradient descent on a linear regression loss. Data transformations induced by gradient descent We now introduce a reference linear model y(x) = Wx parameterized by the weight matrix W ∈RNy×Nx, and a training dataset D = {(xi, yi)}N i=1 comprising of input samples xi ∈RNx and respective labels yi ∈RNy. The goal of learning is to minimize the squared-error loss: N X L(W) = 2N ∥Wxi −yi∥2. (2) i=1 One step of gradient descent on L with learning rate η yields the weight change N X ΔW = −η∇W L(W) = −η (Wxi −yi)xT i . (3) N i=1 Considering the loss after changing the weights, we obtain N X L(W + ΔW) = 2N ∥(W + ΔW)xi −yi∥2 i=1 (4) N X = 2N ∥Wxi −(yi −Δyi)∥2 i=1 where we introduced the transformed targets yi −Δyi with Δyi = ΔWxi. Thus, we can view the outcome of a gradient descent step as an update to our regression loss (equation 2), where data, and not weights, are updated. Note that this formulation is closely linked to predicting based on nonpara- metric kernel smoothing, see Appendix A.8 for a discussion.

上述描述的自注意力层对应于标准Transformer模型中使用的层。遵循Schlag等人（2021）的工作，我们现在引入对标准模型的第一处（也是唯一一处）偏离，即省略公式1中的softmax操作，从而得到线性自注意力（LSA）层 $e_j \leftarrow e_j + \text{LSA}_\theta(j, \{e_1, \dots, e_N\}) = e_j + \sum_h P_h V_h K_h^T q_{h,j}$。接下来我们将展示，通过一些简单的变换，我们可以将LSA层执行的更新与在线性回归损失上执行一步梯度下降联系起来。 梯度下降诱导的数据变换 我们现在引入一个参考线性模型 $y(x) = Wx$，其由权重矩阵 $W \in \mathbb{R}^{N_y \times N_x}$ 参数化，以及一个训练数据集 $D = \{(x_i, y_i)\}_{i=1}^N$，该数据集包含输入样本 $x_i \in \mathbb{R}^{N_x}$ 和相应的标签 $y_i \in \mathbb{R}^{N_y}$。学习的目标是最小化平方误差损失： $L(W) = \frac{1}{2N} \sum_{i=1}^N \|W x_i - y_i\|^2$. (2) 在损失 L 上以学习率 η 执行一步梯度下降，产生的权重变化为 $\Delta W = -\eta \nabla_W L(W) = -\frac{\eta}{N} \sum_{i=1}^N (W x_i - y_i) x_i^T$. (3) 考虑改变权重后的损失，我们得到 $L(W + \Delta W) = \frac{1}{2N} \sum_{i=1}^N \|(W + \Delta W) x_i - y_i\|^2$ (4) $= \frac{1}{2N} \sum_{i=1}^N \|W x_i - (y_i - \Delta y_i)\|^2$ 其中我们引入了变换后的目标 $y_i - \Delta y_i$，且 $\Delta y_i = \Delta W x_i$。因此，我们可以将梯度下降步骤的结果视为对我们回归损失（公式2）的更新，其中被更新的是数据，而不是权重。注意，这种表述与非参数核平滑（nonparametric kernel smoothing）基于的预测密切相关，参见附录A.8中的讨论。

> Returning to self-attention mechanisms and Transformers, we consider an in-context learning problem where we are given N context tokens together with an extra query to- ken, indexed by N + 1. In terms of our linear regression problem, the N context tokens ej = (xj, yj) ∈RNx+Ny correspond to the N training points in D, and the N+1-th token eN+1 = (xN+1, yN+1) = (xtest, ˆytest) = etest to the test input xtest and the corresponding prediction ˆytest. We use the terms training and in-context data interchangeably, as well as query and test token/data, as we establish their equivalence now. Transformers Learn In-Context by Gradient Descent Transformations induced by gradient descent and a linear self-attention layer can be equivalent We have re-cast the task of learning a linear model as di- rectly modifying the data, instead of explicitly computing and returning the weights of the model (equation 4). We proceed to establish a connection between self-attention and gradient descent. We provide a construction where learning takes place simultaneously by directly updating all tokens, including the test token, through a linear self-attention layer. In other words, the token produced in response to a query (test) token is transformed from its initial value W0xtest, where W0 is the initial value of W, to the post-learning prediction ˆy = (W0 +ΔW)xtest obtained after one gradient descent step. Proposition 1. Given a 1-head linear attention layer and the tokens ej = (xj, yj), for j = 1, . . . , N, one can con- struct key, query and value matrices WK, WQ, WV as well as the projection matrix P such that a Transformer step on every token ej is identical to the gradient-induced dynam- ics ej ←(xj, yj) + (0, −ΔWxj) = (xj, yj) + P V KT qj such that ej = (xj, yj −Δyj). For the test data token (xN+1, yN+1) the dynamics are identical.

回到自注意力机制和Transformer，我们考虑一个上下文学习问题，其中给定N个上下文token以及一个额外的查询token，其索引为N+1。就我们的线性回归问题而言，N个上下文token ej = (xj, yj) ∈ RNx+Ny 对应于D中的N个训练点，而第N+1个token eN+1 = (xN+1, yN+1) = (xtest, ˆytest) = etest 对应于测试输入 xtest 及其相应的预测 ˆytest。由于我们现在将建立它们的等价性，因此我们互换使用“训练”与“上下文数据”、“查询”与“测试token/数据”这些术语。 Transformer通过梯度下降进行上下文学习 梯度下降与线性自注意力层诱导的变换可以等价 我们将学习线性模型的任务重新表述为直接修改数据，而不是显式地计算并返回模型的权重（公式4）。接下来，我们建立自注意力与梯度下降之间的联系。我们提供了一种构造方法，其中学习通过线性自注意力层直接更新所有token（包括测试token）来同时发生。换言之，响应于查询（测试）token而产生的token，其值从初始值 W0xtest（其中 W0 是 W 的初始值）变换为经过一步梯度下降后获得的预测 ˆy = (W0 +ΔW)xtest。 命题1。给定一个单头线性注意力层以及token ej = (xj, yj)（其中 j = 1, . . . , N），可以构造键、查询和值矩阵 WK, WQ, WV 以及投影矩阵 P，使得对每个token ej 的Transformer步骤与由梯度诱导的动力学 ej ←(xj, yj) + (0, −ΔWxj) = (xj, yj) + P V KT qj 完全相同，从而得到 ej = (xj, yj −Δyj)。对于测试数据token (xN+1, yN+1)，其动力学也是相同的。

> The simple construction can be found in Appendix A.1 and we denote the corresponding self-attention weights by θGD. Below, we provide some additional insights on what is needed to implement the provided LSA-layer weight con- struction, and further details on what it can achieve: · Full self-attention. Our dynamics model training is based on in-context tokens only, i.e., only e1, . . . , eN are used for computing key and value matrices; the query token eN+1 (containing test data) is excluded. This leads to a linear function in xtest as well as to the correct ΔW, induced by gradient descent on a loss consisting only of the training data. This is a minor deviation from full self-attention. In practice, this modification can be dropped, which corresponds to assuming that the underlying initial weight matrix is zero, W0 ≈0, which makes ΔW in equation 8 independent of the test token even if incorporating it in the key and value matrices. In our experiments, we see that these assumptions are met when initializing the attention weights θ to small values. · Reading out predictions. When initializing the y- entry of the test-data token with −W0xN+1, i.e. etest = (xtest, −W0xtest), the test-data prediction ˆy can be eas- ily read out by simply multiplying again by −1 the updated token, since −yN+1 + ΔyN+1 = −(yN+1 − ΔyN+1) = yN+1 + ΔWxN+1. This can easily be done by a final projection matrix, which incidentally is usually found in Transformer architectures. Impor- tantly, we see that a single head of self-attention is

简单构造方法见附录 A.1，我们将对应的自注意力权重记为 θGD。 下文提供了关于实现所给出的 LSA 层权重构造所需条件的额外见解，以及关于其所能实现效果的进一步细节： · 完整自注意力。我们的动力学模型训练仅基于上下文内 token，即仅使用 e1, . . . , eN 来计算键矩阵和值矩阵；查询 token eN+1（包含测试数据）被排除在外。这导致了对 xtest 的线性函数，以及由仅由训练数据组成的损失函数的梯度下降所诱导的正确 ΔW。这是相对于完整自注意力的一种轻微偏差。在实践中，可以舍弃这一修改，这相当于假设底层初始权重矩阵为零，W0 ≈0，这使得即使将测试 token 纳入键矩阵和值矩阵，方程 8 中的 ΔW 也与测试 token 无关。在我们的实验中，我们发现当将注意力权重 θ 初始化为小值时，这些假设是满足的。 · 读取预测值。当将测试数据 token 的 y 分量初始化为 −W0xN+1，即 etest = (xtest, −W0xtest) 时，测试数据的预测值 ˆy 可以通过简单地再次将更新后的 token 乘以 −1 来轻松读取，因为 −yN+1 + ΔyN+1 = −(yN+1 − ΔyN+1) = yN+1 + ΔWxN+1。这可以通过一个最终投影矩阵轻松完成，而该投影矩阵通常也存在于 Transformer 架构中。重要的是，我们看到单个自注意力头

> sufficient to transform our training targets as well as the test prediction simultaneously. · Uniqueness. We note that the construction is not unique; in particular, it is only required that the prod- ucts PWV as well as WKWQ match the construc- tion. Furthermore, since no nonlinearity is present, any rescaling s of the matrix products, i.e., PWV s and WKWQ/s, leads to an equivalent result. If we correct for these equivalent formulations, we can experimen- tally verify that weights of our learned Transformers indeed match the presented construction. · Meta-learned task-shared learning rates. When training self-attention parameters θ across a family of in-context learning tasks τ, where the data (xτ,i, yτ,i) follows a certain distribution, the learning rate can be implicitly (meta-)learned such that an optimal loss re- duction (averaged over tasks) is achieved given a fixed number of update steps. In our experiments, we find this to be the case. This kind of meta-learning to im- prove upon plain gradient descent has been leveraged in numerous previous approaches for deep neural net- works (Li et al., 2017; Lee & Choi, 2018; Park & Oliva, 2019; Zhao et al., 2020; Flennerhag et al., 2020). · Task-specific data transformations. A self-attention layer is in principle further capable of exploiting statis- tics in the current training data samples, beyond mod- eling task-shared curvature information in θ. More concretely, a LSA layer updates an input sample ac- cording to a data transformation xj ←xj + Δxj = (I + P(X)V (X)K(X)T WQ)xj = Hθ(X)xj, with

足以同时转换我们的训练目标和测试预测。 · 唯一性。我们注意到该构造并非唯一；具体而言，仅要求乘积 PWV 以及 WKWQ 与构造相匹配。此外，由于不存在非线性，对矩阵乘积的任何重新缩放 s，即 PWV s 和 WKWQ/s，都会导致等效的结果。如果我们修正这些等效的表述，可以通过实验验证我们学习到的 Transformer 的权重确实与所呈现的构造相匹配。 · 元学习的任务共享学习率。当在上下文学习任务族 τ 上训练自注意力参数 θ 时，其中数据 (xτ,i, yτ,i) 遵循某种分布，学习率可以被隐式地（元）学习，从而在给定的固定更新步数下实现最优的损失减少（在任务上取平均）。在我们的实验中，我们发现确实如此。这种通过元学习来改进普通梯度下降的方法已被用于众多针对深度神经网络的前期方法中（Li et al., 2017; Lee & Choi, 2018; Park & Oliva, 2019; Zhao et al., 2020; Flennerhag et al., 2020）。 · 任务特定的数据变换。原则上，自注意力层还具备利用当前训练数据样本中统计信息的能力，这超出了对 θ 中任务共享曲率信息的建模。更具体地说，LSA 层根据数据变换 xj ←xj + Δxj = (I + P(X)V (X)K(X)T WQ)xj = Hθ(X)xj 来更新输入样本，其中

> X the Nx × N input training data matrix, when ne- glecting influences by target data yi. Through Hθ(X), a LSA layer can encode in θ an algorithm for carrying out data transformations which depend on the actual input training samples in X. In our experiments, we see that trained self-attention learners employ a simple form of H(X) and that this leads to substantial speed ups in for GD and TF learning. 3. Trained Transformers do mimic gradient descent on linear regression tasks We now experimentally investigate whether trained attention-based models implement gradient-based in- context learning in their forward passes. We gradually build up from single linear self-attention layers to multi-layer non- linear models, approaching full Transformers. In this sec- tion, we follow the assumption of Proposition 1 tightly and construct our tokens by concatenating input and target data, ej = (xj, yj) for 1 ⩽j ⩽N, and our query token by con- catenating the test input and a zero vector, eN+1 = (xtest, 0). We show how to lift this assumption in the last section of the Transformers Learn In-Context by Gradient Descent 0.40 2.5 GD Trained TF Model cos Preds diff Model diff 1.00 2.0 0.35 0.95 L2 Norm 1.5 Loss 0.30 0.90 1.0 0.25 0.85 0.5 0.80 0.0 0.20 Training steps Training steps is identical to the one of gradient descent. Center left: Almost perfect alignment of GD and the model generated by the SA layer after training, measured by cosine similarity and the L2 distance between models as well as their predictions. Center right: Identical loss of GD, the LSA layer model as well as the model obtained by interpolating between the construction and the optimized LSA layer weights for different N = Nx. Outer right: The trained LSA layer, gradient descent and their interpolation show identically loss (in log-scale) when provided input data different than during training i.e. with scale of 1. We display the mean/std. or the single runs of 5 seeds.

X 为 N×N 的输入训练数据矩阵，此处忽略目标数据 yi 的影响。通过 Hθ(X)，LSA 层可以在 θ 中编码一种算法，该算法执行依赖于 X 中实际输入训练样本的数据变换。在我们的实验中，我们观察到经过训练的自注意力学习器采用了 H(X) 的简单形式，这使得 GD 和 TF 学习获得了显著的速度提升。 3. 经过训练的 Transformer 在线性回归任务中确实模拟了梯度下降 我们现在通过实验研究，经过训练的基于注意力的模型是否在其前向传播中实现了基于梯度的上下文学习。我们从单个线性自注意力层逐步构建到多层非线性模型，逐渐接近完整的 Transformer。在本节中，我们严格遵循命题 1 的假设，通过拼接输入数据和目标数据来构建 token，即 ej = (xj, yj)，其中 1 ⩽ j ⩽ N；并通过拼接测试输入和零向量来构建查询 token，即 eN+1 = (xtest, 0)。我们在本文最后一节展示了如何放宽这一假设。 Transformer 通过梯度下降进行上下文学习 0.40 2.5 GD 训练的 TF 模型余弦相似度 预测差异 模型差异 1.00 2.0 0.35 0.95 L2 范数 1.5 损失 0.30 0.90 1.0 0.25 0.85 0.5 0.80 0.0 0.20 训练步数 训练步数 与梯度下降完全一致。中左图：经过训练的 SA 层生成的模型与 GD 之间几乎完美的对齐，通过模型之间及其预测之间的余弦相似度和 L2 距离来衡量。中右图：GD、LSA 层模型以及通过对构造权重和优化后的 LSA 层权重进行插值得到的模型，在不同的 N = Nx 下具有相同的损失。右外图：当提供与训练期间不同的输入数据（即尺度为 1）时，经过训练的 LSA 层、梯度下降及其插值显示出相同的损失（对数尺度）。我们展示了 5 个随机种子的单次运行或均值/标准差。

paper. The prediction ˆyθ({eτ,1, . . . , eτ,N}, eτ,N+1) of the attention-based model, which depends on all tokens and on the parameters θ, is read-out from the y-entry of the updated N + 1-th token as explained in the previous section. The objective of training, visualized in Figure 1, is to mini- mize the expected squared prediction error, averaged over tasks minθ Eτ[||ˆyθ({eτ,1, . . . , eτ,N}, eτ,N+1) −yτ,test||2]. We achieve this by minibatch online minimization (by Adam (Kingma & Ba, 2014)): At every optimization step, we con- struct a batch of novel training tasks and take a step of stochastic gradient descent on the loss function: B X L(θ) = 1 i=1, eτ,N+1) −yτ,test||2 (5) || ˆyθ({eτ,i}N B τ=1 where each task (context) τ consists of in-context training data Dτ = {(xτ,i, yτ,i)}N i=1 and test point (xτ,N+1, yτ,N+1), which we use to construct our tokens {eτ,i}N+1 i=1 as described above. We denote the optimal pa- rameters found by this optimization process by θ∗. In our setup, finding θ∗may be thought of as meta-learning, while learning a particular task τ corresponds to simply evaluat- ing the model ˆyθ({eτ,1, . . . , eτ,N}, eτ,N+1). Note that we therefore never see the exact same training task twice during training. See Appendix A.12, especially Figure 16 for an analyses when using a fixed dataset size which we cycle over during training. We focus on solvable tasks and similarly to Garg et al. (2022) generate data for each task using a teacher model with parameters Wτ ∼N(0, I). We then sample xτ,i ∼ U(−1, 1)nI and construct targets using the task-specific teacher model, yτ,i = Wτxτ,i. In the majority of our exper- iments we set the dimensions to N = nI = 10 and nO = 1. Since we use a noiseless teacher for simplicity, we can ex- pect our regression tasks to be well-posed and analytically solvable as we only compute a loss on the Transformers last token, which stands in contrast to usual autoregressive training and the training setup of Garg et al. (2022). Full details and results for training with a fixed training set size may be found in Appendix A.12.

基于注意力机制的模型预测值 $\hat{y}_\theta(\{e_{\tau,1}, \dots, e_{\tau,N}\}, e_{\tau,N+1})$ 依赖于所有标记（tokens）以及参数 θ，如前一节所述，该预测值是从更新后的第 $N+1$ 个标记的 y 分量中读取的。 如图1所示，训练的目标是最小化在任务上取平均的期望平方预测误差： $\min_\theta \mathbb{E}_\tau [\|\hat{y}_\theta(\{e_{\tau,1}, \dots, e_{\tau,N}\}, e_{\tau,N+1}) - y_{\tau,\text{test}}\|^2]$ 我们通过小批量在线最小化（使用 Adam 优化器 (Kingma & Ba, 2014)）来实现这一目标：在每一个优化步骤中，我们构建一批新的训练任务，并对损失函数执行一步随机梯度下降： $L(\theta) = \frac{1}{B} \sum_{\tau=1}^B \|\hat{y}_\theta(\{e_{\tau,i}\}_{i=1}^N, e_{\tau,N+1}) - y_{\tau,\text{test}}\|^2$ (5) 其中，每个任务（上下文）τ 由上下文训练数据 $D_\tau = \{(x_{\tau,i}, y_{\tau,i})\}_{i=1}^N$ 和测试点 $(x_{\tau,N+1}, y_{\tau,N+1})$ 组成，我们利用这些数据按照上述方法构建标记 $\{e_{\tau,i}\}_{i=1}^{N+1}$。我们将通过此优化过程找到的最优参数记为 θ^*。在我们的设置中，寻找 θ^* 可以被视为元学习，而学习特定任务 τ 则对应于简单地评估模型 $\hat{y}_\theta(\{e_{\tau,1}, \dots, e_{\tau,N}\}, e_{\tau,N+1})$。因此，在训练过程中，我们永远不会两次看到完全相同的训练任务。关于使用固定数据集大小并在训练期间循环使用的分析，请参见附录 A.12，特别是图16。 我们关注可解任务，并类似于 Garg et al. (2022)，使用参数为 $W_\tau \sim \mathcal{N}(0, I)$ 的教师模型为每个任务生成数据。然后，我们采样 $x_{\tau,i} \sim U(-1, 1)^{n_I}$，并使用任务特定的教师模型构建目标值，即 $y_{\tau,i} = W_\tau x_{\tau,i}$。在大多数实验中，我们将维度设置为 $N = n_I = 10$ 和 $n_O = 1$。由于我们为了简单起见使用了无噪声的教师模型，我们可以预期我们的回归任务是适定且可解析求解的，因为我们仅在 Transformer 的最后一个标记上计算损失，这与通常的自回归训练以及 Garg et al. (2022) 的训练设置形成对比。关于使用固定训练集大小进行训练的完整细节和结果，请参见附录 A.12。

Test on larger inputs GD Interpolated Trained TF Cosine sim Loss 0.5 1.0 1.5 2.0 where x U(,) One-step of gradient descent vs. a single trained self-attention layer Our first goal is to investigate whether a trained single, linear self-attention layer can be explained by the provided weight construction that implements GD. To that end, we compare the predictions made by a LSA layer with trained weights θ∗ (which minimize equation 5) and with constructed weights θGD (which satisfy Proposition 1). Recall that a LSA layer yields the prediction ˆyθ(xtest) = eN+1 + LSAθ({e1, . . . , eN}, eN+1) = ∆Wθ,Dxtest, which is linear in xtest. We denote by ∆Wθ,D the matrix generated by the LSA layer following the construction provided in Proposition 1, with query token eN+1 set such that the initial prediction is set to zero, ˆytest = 0. We compare ˆyθ(xtest) to the prediction of the control LSA ˆyθGD(xtest), which under our token construction corresponds to a linear model trained by one step of gradient descent starting from W0 = 0. For this control model, we determine the optimal learning rate η by minimizing L(η) over a training set of 104 tasks through line search, with L(η) defined analogously to equation 5. More concretely, to compare trained and constructed LSA layers, we sample Tval = 104 validation tasks and record the following quantities, averaged over validation tasks: (1) the difference in predictions measured with the L2 norm, // ˆyθ(xτ,test) −ˆyθGD(xτ,test) //, (2) the cosine similarity be-

在更大输入上的测试 GD 插值 训练 TF 余弦相似度 损失 0.5 1.0 1.5 2.0 其中 x ~ U(,) 一步梯度下降与单个训练后的自注意力层 我们的首要目标是探究一个训练好的单层线性自注意力层（LSA）能否由实现梯度下降（GD）的权重构造来解释。为此，我们将具有训练权重 θ*（最小化公式 5）的 LSA 层的预测，与具有构造权重 θGD（满足命题 1）的预测进行比较。 回顾一下，LSA 层产生的预测为 ˆyθ(xtest) = eN+1 + LSAθ({e1, . . . , eN}, eN+1) = ∆Wθ,Dxtest，它对 xtest 是线性的。我们用 ∆Wθ,D 表示根据命题 1 中提供的构造生成的矩阵，其中查询 token eN+1 被设定为使初始预测为零，即 ˆytest = 0。我们将 ˆyθ(xtest) 与对照 LSA 的预测 ˆyθGD(xtest) 进行比较，后者在我们的 token 构造下对应于从 W0 = 0 开始通过一步梯度下降训练的线性模型。对于该对照模型，我们通过在线搜索中在包含 10^4 个任务的训练集上最小化 L(η) 来确定最优学习率 η，其中 L(η) 的定义类似于公式 5。 更具体地说，为了比较训练好的 LSA 层和构造的 LSA 层，我们采样 Tval = 10^4 个验证任务，并记录以下在验证任务上取平均的量：(1) 用 L2 范数测量的预测差异， // ˆyθ(xτ,test) −ˆyθGD(xτ,test) //，(2) 余弦相似度

∂ ˆ yθGD(xτ,test) ∂xtest and ∂ ˆ yθ(xτ,test) tween the sensitivities ∂xtest as well as ∂ ˆ yθGD(xτ,test) ∂xtest − ∂ ˆ yθ(xτ,test) (3) their difference // ∂xtest // again accord- ing to the L2 norm, which in both cases yields the explicit models computed by the algorithm. We show the results of these comparisons in Figure 2. We find an excellent agreement between the two models over a wide range of hyperparameters. We note that as we do not have direct ac- cess to the initialization of W in the attention-based learners (it is hidden in θ), we cannot expect the models to agree exactly. Although the above metrics are important to show simi- larities between the resulting learned models (in-context Transformers Learn In-Context by Gradient Descent vs. gradient-based), the underlying algorithms could still be different. We therefore carry out an extended set of analyses: 1. Interpolation. We take inspiration on recent work (Benzing et al., 2022; Entezari et al., 2021) that showed approximate equivalence of models found by SGD af- ter permuting weights within the trained neural net- works. Since our models are deep linear networks with respect to xtest we only correct for scaling mismatches between the two models – in this case the construction that implements GD and the trained weights. As shown in Figure 2, we observe (and can actually inspect by eye, see Appendix Figure 9) that a simple scaling cor- rection on the trained weights is enough to recover the weight construction implementing GD. This leads to an identical loss of GD, the trained Transformer and the linearly interpolated weights θI = (θ + θGD)/2. See details in Appendix A.3 on how our weight correction and interpolation is obtained.

$\frac{\partial \hat{y}_{\theta_{GD}}(x_{\tau, test})}{\partial x_{test}}$ 与 $\frac{\partial \hat{y}_{\theta}(x_{\tau, test})}{\partial x_{test}}$ 之间的敏感度，以及 $\frac{\partial \hat{y}_{\theta_{GD}}(x_{\tau, test})}{\partial x_{test}} - \frac{\partial \hat{y}_{\theta}(x_{\tau, test})}{\partial x_{test}}$ 的差值 $\|\frac{\partial \hat{y}_{\theta_{GD}}(x_{\tau, test})}{\partial x_{test}} - \frac{\partial \hat{y}_{\theta}(x_{\tau, test})}{\partial x_{test}}\|$，再次根据 L_2 范数计算，这两种情况均得出由算法计算出的显式模型。我们在图 2 中展示了这些比较的结果。我们发现，在广泛的超参数范围内，两个模型之间具有极好的一致性。我们注意到，由于我们无法直接访问基于注意力的学习器中 W 的初始化（它隐藏在 θ 中），因此我们不能期望模型完全一致。 尽管上述指标对于展示所得学习模型（上下文 Transformer 与基于梯度的模型）之间的相似性至关重要，但底层算法仍可能不同。因此，我们进行了一系列扩展分析： 1. 插值。我们受到近期工作的启发（Benzing et al., 2022; Entezari et al., 2021），这些工作表明，在训练后的神经网络内对权重进行置换后，SGD 找到的模型具有近似等价性。由于我们的模型相对于 x_{test} 是深度线性网络，我们仅对两个模型之间的缩放不匹配进行校正——在这种情况下，即实现 GD 的构造与训练后的权重。如图 2 所示，我们观察到（实际上可以通过肉眼检查，见附录图 9），对训练后的权重进行简单的缩放校正就足以恢复实现 GD 的权重构造。这导致 GD、训练后的 Transformer 以及线性插值权重 $\theta_I = (\theta + \theta_{GD})/2$ 的损失完全相同。关于如何获得我们的权重校正和插值，详见附录 A.3。

2. Out-of-distribution validation tasks. To test if our in-context learner has found a generalizable update rule, we investigate how GD, the trained LSA layer and its interpolation behave when providing in-context data in regimes different to the ones used during train- ing. We therefore visualize the loss increase when (1) sampling the input data from U(−α, α)Nx or (2) scal- ing the teacher weights by α as αW when sampling validation tasks. For both cases, we set α = 1 dur- ing training. We again observe that when training a single linear self-attention Transformer, for both inter- ventions, the Transformer performs equally to gradient descent outside of this training setups, see Figure 2 as well Appendix Figure 6. Note that the loss obtained through gradient descent also starts degrading quickly outside the training regime. Since we tune the learning rate for the input range [−1, 1] and one gradient step, tasks with larger input range will have higher curvature and the optimal learning rate for smaller ranges will lead to divergence and a drastic increase in loss also for GD. 3. Repeating the LSA update. Since we claim that a sin- gle trained LSA layer implements a GD-like learning rule, we further test its behavior when applying it re- peatedly, not only once as in training. After we correct the learning rate of both algorithms, i.e. for GD and the trained Transformer with a dampening parameter λ = 0.75 (details in Appendix A.6), we see an identi- cal loss decrease of both GD and the Transformer, see

2. 分布外验证任务。为了检验我们的上下文学习器是否发现了可泛化的更新规则，我们研究了当提供的上下文数据处于与训练阶段不同的分布时，梯度下降（GD）、训练后的线性自注意力（LSA）层及其插值模型的表现。因此，我们在采样验证任务时，可视化了损失函数的增加情况，具体包括：(1) 从 U(−α, α)^Nx 中采样输入数据，或 (2) 在采样验证任务时将教师权重缩放为 αW。在这两种情况下，训练期间均设置 α = 1。我们再次观察到，当训练单个线性自注意力 Transformer 时，对于这两种干预措施，Transformer 在该训练设置之外的表现与梯度下降相当，参见图 2 以及附录图 6。需要注意的是，通过梯度下降获得的损失在训练分布之外也会迅速恶化。由于我们针对输入范围 [−1, 1] 和单步梯度下降调整了学习率，具有更大输入范围的任务将具有更高的曲率，而针对较小范围优化的学习率会导致梯度下降发散，并引起损失的急剧增加。 3. 重复 LSA 更新。由于我们声称单个训练后的 LSA 层实现了类似梯度下降的学习规则，我们进一步测试了重复应用该层（而不仅仅像训练时那样只应用一次）时的行为。在修正两种算法的学习率之后，即对梯度下降和训练后的 Transformer 引入衰减参数 λ = 0.75（详见附录 A.6），我们观察到梯度下降和 Transformer 表现出相同的损失下降趋势，参见

> To conclude, we present evidence that optimizing a single LSA layer to solve linear regression tasks finds weights that (approximately) coincide with the LSA-layer weight construction of Proposition 1, hence implementing a step of gradient descent, leading to the same learning capabilities on in- and out-of-distribution tasks. We comment on the random seed dependent phase transition of the loss during training in Appendix A.11. Multiple steps of gradient descent vs. multiple layers of self-attention We now turn to deep linear self-attention-only Transform- ers. The construction we put forth in Proposition 1, can be immediately stacked up over K layers; in this case, the final prediction can be read out from the last layer as before by negating the y-entry of the last test token: −yN+1 + PK k=1 ∆yk,N+1 = −(yN+1 −PK k=1 ∆yk,N+1) = yN+1 + PK k=1 ∆WkxN+1, where yk,N+1 are the test token values at layer k, and ∆yk,N+1 the change in the y-entry of the test token after applying the k-th step of self-attention, and ∆Wk the k-th implicit change in the underlying linear model parameters W. When optimizing such Transformers with K layers, we observe that these models generally outper- form K steps of plain gradient descent, see Figure 3. Their behavior is however well described by a variant of gradient descent, for which we tune a single parameter γ defined through the transformation function H(X) which trans- forms the input data according to xj ←H(X)xj, with H(X) = (I −γXXT). We term this gradient descent variant GD++ which we explain and analyze in Appendix A.10.

综上所述，我们提供了证据表明，优化单个LSA层以解决线性回归任务所找到的权重，与命题1中LSA层的权重构造（近似）一致，从而实现了梯度下降的一步，进而在分布内和分布外任务上表现出相同的学习能力。我们在附录A.11中讨论了训练过程中损失函数依赖于随机种子的相变现象。 多步梯度下降与多层自注意力 我们现在转向仅包含深度线性自注意力的Transformer。命题1中提出的构造可以立即在K层上进行堆叠；在这种情况下，最终预测值可以像之前一样通过取最后一层最后一个测试token的y分量的负值来读出：$-y_{N+1} + \sum_{k=1}^{K} \Delta y_{k,N+1} = -(y_{N+1} - \sum_{k=1}^{K} \Delta y_{k,N+1}) = y_{N+1} + \sum_{k=1}^{K} \Delta W_k x_{N+1}$，其中 $y_{k,N+1}$ 是第k层中测试token的值，$\Delta y_{k,N+1}$ 是应用第k步自注意力后测试token的y分量的变化量，ΔW_k 是底层线性模型参数 W 的第k个隐式变化量。当优化具有K层的此类Transformer时，我们观察到这些模型通常优于K步普通梯度下降，见图3。然而，它们的行为可以用梯度下降的一个变体来很好地描述，该变体通过变换函数 $H(X)$ 定义了一个单一参数 γ，该函数根据 $x_j \leftarrow H(X)x_j$ 对输入数据进行变换，其中 $H(X) = (I - \gamma XX^T)$。我们将此梯度下降变体称为GD++，并在附录A.10中对其进行解释和分析。

> To analyze the effect of adding more layers to the architec- ture, we first turn to the arguably simplest extension of a single SA layer and analyze a recurrent or looped 2-layer LSA model. Here, we simply repeatably apply the same layer (with the same weights) multiple times i.e. drawing the analogy to learning an iterative algorithm that applies the same logic multiple times. Somewhat surprisingly, we find that the trained model sur- passes plain gradient descent, which also results in decreas- ing alignment between the two models (see center left col- umn), and the recurrent Transformer realigns perfectly with GD++ while matching its performance on in- and out-of distribution tasks. Again, we can interpolate between the Transformer weights found by optimization and the LSA- weight construction with learned η, γ, see Figure 3 & 6. We next consider deeper, non-recurrent 5-layer LSA-only Transformers, with different parameters per layer (i.e. no weight tying). We see that a different GD learning rate as well as γ per step (layer) need to be tuned to match the Transformer performance. This slight modification leads again to almost perfect alignment between the trained TF and GD++ with in this case 10 additional parameters and Transformers Learn In-Context by Gradient Descent (a) Comparing two steps of gradient descent with trained recurrent two-layer Transformers. GD vs trained TF 0.40

为了分析在架构中增加更多层的效果，我们首先转向单层SA模型的一个被认为是最简单的扩展，并分析循环或带环的两层LSA模型。在这里，我们简单地重复应用相同的层（具有相同的权重）多次，即类比于学习一个应用相同逻辑多次的迭代算法。 有些令人惊讶的是，我们发现训练后的模型超越了普通的梯度下降，这也导致了两个模型之间对齐程度的降低（参见中间左列），并且循环Transformer与GD++完美重新对齐，同时在分布内和分布外任务上匹配其性能。同样，我们可以在优化找到的Transformer权重和具有学习到的η、γ的LSA权重构造之间进行插值，参见图3和图6。 接下来，我们考虑更深的、非循环的五层纯LSA Transformer，每层具有不同的参数（即没有权重共享）。我们看到，需要调整不同的GD学习率以及每步（层）的γ，以匹配Transformer的性能。这种轻微修改再次导致训练后的TF与GD++之间几乎完美的对齐，在这种情况下增加了10个额外参数， Transformer通过梯度下降进行上下文学习 (a) 比较两步梯度下降与训练后的循环两层Transformer。 GD与训练后的TF 0.40

2.5 GD GD+ + Model cos 0.35 1.0 2.0 Trained TF 0.30 0.9 Cosine sim L2 Norm 1.5 Loss Preds diff Model diff 0.25 0.8 1.0 0.20 0.7 0.5 0.15 0.6 0.10 0.5 0.0 Training steps Training steps (b) Comparing five steps of gradient descent with trained five-layer Transformers. GD vs trained TF 0.4 2.0 GD 1.05 Model cos GD+ + 5 steps 0.3 Trained TF 1.00 1.5 Cosine sim L2 Norm 0.95 Loss Preds diff Model diff 0.2 1.0 0.90 0.5 0.1 0.85 0.80 0.0 Training steps Training steps data transformation. On both cases, we tuned the gradient descent learning rates as well as the scalar γ which governs the data transformation H(X). Center left & center right columns: We measure the alignment between the GD as well as the GD++ models and the trained TF. In both cases the TF aligns well with GD in the beginning of training but aligns much better with GD++ after training. Far right column: TF performance (in log-scale) mimics the one of GD++ well when testing on OOD tasks (α = 1). loss close to 0, see Figure 3. Nevertheless, we see that the naive correction necessary for model interpolation used in the aforementioned experiments is not enough to interpolate without a loss increase. We leave a search for better weight corrections to future work. We further study Transformers with different depths for recurrent as well as non-recurrent architectures with multiple heads and equipped with MLPs, and find qualitatively equivalent results, see Appendix Fig- ure 7 and Figure 8. Additionally, in Appendix A.9, we provide results obtained when using softmax SA layers as well as LayerNorm, thus essentially retrieving the standard Transformer architecture. We again observe and are able to explain (after slight architectural modifications) good learning performance and as well as alignment with the con- struction of Proposition 1, though worse than when using linear self-attention. These findings suggest that the in- context learning abilities of the standard Transformer with these common architecture choices can be explained by the gradient-based learning hypothesis explored here. Our findings also question the ubiquitous use of softmax atten- tion, and suggest further investigation is warranted into the performance of linear vs. softmax SA layers in real-world learning tasks, as initiated by Schlag et al. (2021).

2.5 GD GD+ + 模型余弦相似度 0.35 1.0 2.0 训练后的TF 0.30 0.9 余弦相似度 L2范数 1.5 损失 预测差异 模型差异 0.25 0.8 1.0 0.20 0.7 0.5 0.15 0.6 0.10 0.5 0.0 训练步数 训练步数 (b) 比较梯度下降的五步与训练后的五层Transformer。 GD vs 训练后的TF 0.4 2.0 GD 1.05 模型余弦相似度 GD+ + 5步 0.3 训练后的TF 1.00 1.5 余弦相似度 L2范数 0.95 损失 预测差异 模型差异 0.2 1.0 0.90 0.5 0.1 0.85 0.80 0.0 训练步数 训练步数 数据变换。在这两种情况下，我们均调整了梯度下降的学习率以及控制数据变换 H(X) 的标量 γ。中间左列与中间右列：我们测量了 GD 以及 GD++ 模型与训练后的 TF 之间的对齐程度。在这两种情况下，TF 在训练初期与 GD 对齐良好，但在训练后与 GD++ 的对齐程度要高得多。最右列：在 OOD 任务（α = 1）上进行测试时，TF 的性能（对数尺度）很好地模拟了 GD++ 的性能。 损失接近 0，参见图 3。尽管如此，我们观察到，在上述实验中用于模型插值的朴素校正不足以在不增加损失的情况下进行插值。我们将寻找更好的权重校正方法留作未来工作。我们进一步研究了具有不同深度的 Transformer，包括循环和非循环架构，配备多头和 MLP，并发现定性上等效的结果，参见附录图 7 和图 8。此外，在附录 A.9 中，我们提供了使用 softmax SA 层以及 LayerNorm 时获得的结果，从而基本上恢复了标准的 Transformer 架构。我们再次观察到并能够解释（在轻微的架构修改后）良好的学习性能以及与命题 1 构造的对齐，尽管不如使用线性自注意力时那么好。这些发现表明，标准 Transformer 在这些常见架构选择下的上下文学习能力可以由此处探讨的基于梯度的学习假设来解释。我们的发现也质疑了 softmax 注意力的普遍使用，并建议对线性 SA 层与 softmax SA 层在真实世界学习任务中的性能进行进一步调查，正如 Schlag et al. (2021) 所开启的那样。

Transformers solve nonlinear regression tasks by gradient descent on deep data representations It is unreasonable to assume that the astonishing in-context learning flexibility observed in large Transformers is ex- Test on larger inputs GD+ + vs trained TF 2.5 GD GD+ + Model cos 1.0 2.0 Interpolated Trained TF 0.9 Cosine sim L2 Norm 1.5 Preds diff Model diff Loss 0.8 1.0 0.7 0.5 0.6 0.5 0.0 0.5 1.0 1.5 2.0 where x U(,) Training steps Test on larger inputs GD+ + vs trained TF 2.0 1.05 GD GD+ + Model cos 1.00 Trained TF 1.5 Cosine sim L2 Norm 0.95 Loss Preds diff Model diff 1.0 0.90 0.5 0.85 0.80 0.0 0.5 1.0 1.5 2.0 where x U(,) Training steps plained by gradient descent on linear models. We now show that this limitation can be resolved by incorporating one additional element of fully-fledged Transformers: preceding self-attention layers by MLPs enables learning linear models by gradient descent on deep representations which motivates our illustration in Figure 1. Empirically, we demonstrate this by solving non-linear sine-wave regression tasks, see A.7. We state Proposition 2. Given a Transformer block i.e. a MLP m(e) which transforms the tokens ej = (xj, yj) followed by an at- tention layer, we can construct weights that lead to gradient descent dynamics descending 2N PN i=1 ||Wm(xi) −yi||2. Iteratively applying Transformer blocks therefore can solve kernelized least-squares regression problems with kernel function k(x, y) = m(x)⊤m(y) induced by the MLP m(・).

Transformer 通过对深度数据表示进行梯度下降来解决非线性回归任务 假设在大型 Transformer 中观察到的惊人的上下文学习灵活性仅由线性模型上的梯度下降来解释是不合理的。我们现在表明，通过纳入完整 Transformer 的另一个要素可以解决这一局限性：在自注意力层之前放置 MLP（多层感知机）使得能够对深度表示进行梯度下降以学习线性模型，这启发了我们在图 1 中的图示。在实证上，我们通过解决非线性正弦波回归任务来证明这一点，参见 A.7。我们陈述 命题 2。给定一个 Transformer 块，即一个将标记 $e_j = (x_j, y_j)$ 进行变换的 MLP $m(e)$，随后是一个注意力层，我们可以构造权重，使得梯度下降动力学下降 $\sum_{i=1}^{2N} \|W m(x_i) - y_i\|^2$。因此，迭代应用 Transformer 块可以解决由 MLP $m(\cdot)$ 诱导的核函数 $k(x, y) = m(x)^\top m(y)$ 的核化最小二乘回归问题。

A detailed discussion on this form of kernel regression as well as kernel smoothing w/wo softmax nonlinearity through gradient descent on the data can be found in Appendix A.8. The way MLPs transform data in Transformers di- verges from the standard meta-learning approach, where a task-shared input embedding network is optimized by backpropagation-through-training to improve the learning performance of a task-specific readout (e.g., Raghu et al., 2020; Lee et al., 2019; Bertinetto et al., 2019). On the other hand, given our token construction in Proposition 1, MLPs in Transformers intriguingly process both inputs and targets. The output of this transformation is then processed by a sin- Transformers Learn In-Context by Gradient Descent gle linear self-attention layer, which, according to our theory, is capable of implementing gradient descent learning. We compare the performance of this Transformer model, where all weights are learned, to a control Transformer where the final LSA weights are set to the construction θGD which is therefore identical to training an MLP by backpropagation through a GD updated output layer. Intriguingly, both obtained functions show again surprising similarity on (1) the initial (meta-learned) prediction, read out after the MLP, and (2) the final prediction, after altering the output of the MLP through GD or the self-attention layer. This is again reflected in our alignment measures that now, since the obtained models are nonlinear w.r.t. xtest, only rep- resent the two first parts of the Taylor approximation of the obtained functions. Our results serve as a first demonstra- tion of how MLPs and self-attention layers can interplay to support nonlinear in-context learning, allowing to fine-tune deep data representations by gradient descent. Investigating the interplay between MLPs and SA-layer in deep TFs is left for future work.

关于这种形式的核回归，以及通过数据上的梯度下降进行带或不带softmax非线性的核平滑的详细讨论，见附录A.8。Transformer中MLP对数据的变换方式与标准的元学习方法不同，在后者中，任务共享的输入嵌入网络通过反向传播穿过训练（backpropagation-through-training）进行优化，以提升任务特定读出（readout）的学习性能（例如，Raghu et al., 2020; Lee et al., 2019; Bertinetto et al., 2019）。另一方面，鉴于我们在命题1中的token构造，Transformer中的MLP令人着迷地同时处理输入和目标。该变换的输出随后由一个单层线性自注意力层处理，根据我们的理论，该层能够实现梯度下降学习。我们将此Transformer模型（其中所有权重均被学习）的性能与一个对照Transformer进行比较，在后者中，最终LSA权重被设置为构造θGD，因此这与通过梯度下降更新的输出层对MLP进行反向传播训练是相同的。 令人着迷的是，所获得的两个函数再次在（1）初始（元学习）预测（在MLP之后读出）和（2）最终预测（通过GD或自注意力层改变MLP输出之后）上表现出惊人的相似性。这再次反映在我们的对齐度量中，由于所获得的模型关于xtest是非线性的，这些度量现在仅表示所获得函数泰勒近似的前两部分。我们的结果作为首次演示，展示了MLP和自注意力层如何相互作用以支持非线性上下文学习，从而允许通过梯度下降对深度数据表示进行微调。在深度TF中研究MLP与SA层之间的相互作用留待未来工作。

4. Do self-attention layers build regression tasks? The construction provided in Proposition 1 and the previ- ous experimental section relied on a token structure where both input and output data are concatenated into a single token. This design is different from the way tokens are typi- cally built in most of the related work dealing with simple few-shot learning problems as well as in e.g. language mod- eling. We therefore ask: Can we overcome the assumption required in Proposition 1 and allow a Transformer to build the required token construction on its own? This motivates Proposition 3. Given a 1-head linear or softmax atten- tion layer and the token construction e2j = (xj), e2j+1 = (0, yj) with a zero vector 0 of dim Nx −Ny and concate- nated positional encodings, one can construct key, query and value matrix WK, WQ, WV as well as the projection matrix P such that all tokens ej are transformed into tokens equivalent to the ones required in Proposition 1. The construction and its discussion can be found in Ap- pendix A.5. To provide evidence that copying is per- formed in trained Transformers, we optimize a two-layer self-attention circuit on in-context data where alternating tokens include input or output data i.e. e2j = (xj) and e2j+1 = (0, yj). We again measure the loss as well as the mean of the norm of the partial derivative of the first layer’ s output w.r.t. the input tokens during training, see Figure 5. First, the training speeds are highly variant given different training seeds, also reported in Garg et al. (2022). Never- theless, the Transformer is able to match the performance of a single (not two) step gradient descent. Interestingly,

4. 自注意力层能否构建回归任务？ 命题1中给出的构造以及前文的实验部分依赖于一种令牌结构，其中输入数据和输出数据被拼接成单个令牌。这种设计与大多数处理简单少样本学习问题的相关工作以及例如语言建模中通常构建令牌的方式不同。因此我们提出：我们能否克服命题1中所需的假设，并允许Transformer自行构建所需的令牌结构？这引出了命题3。给定一个单头线性或softmax注意力层，以及令牌构造 e2j = (xj)，e2j+1 = (0, yj)，其中0是维度为 Nx −Ny 的零向量，并带有拼接的位置编码，可以构造键、查询和值矩阵 WK, WQ, WV 以及投影矩阵 P，使得所有令牌 ej 都被转换为与命题1中所需的令牌等效的令牌。 该构造及其讨论见附录A.5。为了提供证据表明在训练好的Transformer中执行了复制操作，我们在上下文数据上优化了一个两层自注意力电路，其中交替的令牌包含输入或输出数据，即 e2j = (xj) 和 e2j+1 = (0, yj)。我们再次测量损失以及训练过程中第一层输出相对于输入令牌的部分导数范数的均值，见图5。首先，正如Garg et al. (2022)所报告的，给定不同的训练种子，训练速度差异很大。尽管如此，Transformer能够匹配单步（而非两步）梯度下降的性能。有趣的是，

> before the Transformer performance jumps to the one of GD, token ej transformed by the first self-attention layer becomes notably dependant on the neighboring token ej+1 while staying independent on the others which we denote as eother in Figure 5. 0.55 3.5 GD 1 step TF 2 layers Norm part. derivatives 0.50 3.0 0.45 2.5 t(ej)/ ej 0.40 2.0 Loss t(ej)/ ej + 1 0.35 1.5 t(ej)/ eother 0.30 1.0 0.25 0.5 0.20 0.0 10000 20000 30000 40000 Training steps 10000 20000 30000 40000 Training steps standard token construction. Left: The loss of trained TFs matches one step of GD, not two, and takes an order of magnitude longer to train. Right: Norm of the partial derivatives of the output of the first self-attention layer w.r.t. input tokens. Before the Transformer performance jumps to the one of GD, the first layer becomes highly sensitive to the next token. We interpret this as evidence for a copying mechanism of the Transformer' s first layer to merge input and output data into single tokens as required by Proposition 1. Then, in the second layer the Transformer performs a single step of GD. Notably, we were not able to train the Transformer with linear self-attention layers, but had to incorporate the softmax operation in the first layer. These preliminary find- ings support the study of Olsson et al. (2022) showing that softmax self-attention layers easily learn to copy; we con- firm this claim, and further show that such copying allows the Transformer to proceed by emulating gradient-based learning in the second or deeper attention layers.

在Transformer性能跃升至GD水平之前，经第一层自注意力层变换的token e_j 显著依赖于相邻token e_j+1，而与其他token保持独立，我们在图5中将后者记为 e_{other}。 0.55 3.5 GD 1 step TF 2 layers Norm part. derivatives 0.50 3.0 0.45 2.5 $t(e_j)/e_j$ 0.40 2.0 Loss $t(e_j)/e_{j+1}$ 0.35 1.5 $t(e_j)/e_{other}$ 0.30 1.0 0.25 0.5 0.20 0.0 10000 20000 30000 40000 Training steps 10000 20000 30000 40000 Training steps 标准token构造。左图：训练后的TF损失与GD的一步相匹配，而非两步，且训练时长多出一个数量级。右图：第一层自注意力层输出相对于输入token的偏导数范数。在Transformer性能跃升至GD水平之前，第一层对下一个token变得高度敏感。 我们将此解释为Transformer第一层复制机制的证据，该机制将输入和输出数据合并为单个token，以满足命题1的要求。随后，在第二层，Transformer执行GD的一步。值得注意的是，我们无法使用线性自注意力层训练Transformer，而必须在第一层中引入softmax操作。这些初步发现支持了Olsson等人（2022）的研究，该研究表明softmax自注意力层容易学会复制；我们证实了这一论断，并进一步表明，这种复制允许Transformer通过在第二层或更深层注意力层中模拟基于梯度的学习来继续运行。

> We conclude that copying through (softmax) attention layers is the second crucial mechanism for in-context learning in Transformers. This operation enables Transformers to merge data from different tokens and then to compute dot products of input and target data downstream, allowing for in-context learning by gradient descent to emerge. 5. Discussion Transformers show remarkable in-context learning behavior. Mechanisms based on attention, associative memory and copying by induction heads are currently the leading expla- nations for this remarkable feature of learning within the Transformer forward pass. In this paper, we put forward the hypothesis, similar to Garg et al. (2022) and Aky¨urek et al. (2023), that Transformer' s in-context learning is driven by gradient descent, in short – Transformers learn to learn by gradient descent based on their context. Viewed through the lens of meta-learning, learning Transformer weights cor- responds to the outer-loop which then enables the forward Transformers Learn In-Context by Gradient Descent 0.006 GT Data GD init GD step 1 Tr. TF init Tr. TF step 1 0.6 0.005 0.4 0.004 0.2 Loss 0.003 y 0.0 0.002 0.2 0.001 0.4 0.000 x with one step of gradient descent. Left: Plots of the learned initial functions as well as the adjusted functions through either a layer of self-attention or a step of GD. We observe similar initial functions as well as solutions for the trained TF compared fine-tuning a meta-learned MLP. Center: The performance of the trained Transformer is matched by meta-learned MLPs. Left: We observe strong alignment when comparing the prediction as well as the partial derivatives of the the meta-learned MLP and the trained Transformer.

我们得出结论，通过（softmax）注意力层进行复制是Transformer中上下文学习的第二个关键机制。该操作使Transformer能够合并来自不同标记的数据，并在下游计算输入数据与目标数据的点积，从而使得基于梯度下降的上下文学习得以涌现。 5. 讨论 Transformer展现出卓越的上下文学习行为。基于注意力、联想记忆以及归纳头（induction heads）的复制机制，目前是对Transformer前向传播过程中这一显著学习特性的主要解释。在本文中，我们提出了一个假设，与Garg等人（2022）和Akyürek等人（2023）的观点类似，即Transformer的上下文学习是由梯度下降驱动的，简而言之——Transformer基于其上下文，通过梯度下降来学习如何学习。从元学习的视角来看，学习Transformer的权重对应于外循环，随后使得前向Transformer通过梯度下降进行上下文学习 0.006 GT数据 GD初始化 GD第1步 Tr. TF初始化 Tr. TF第1步 0.6 0.005 0.4 0.004 0.2 损失 0.003 y 0.0 0.002 0.2 0.001 0.4 0.000 x 执行一步梯度下降。左图：展示了学习到的初始函数，以及通过一层自注意力或一步GD调整后的函数。我们观察到，与微调元学习MLP相比，训练后的TF具有相似的初始函数和解决方案。中图：训练后的Transformer的性能与元学习MLP相匹配。左图：在比较元学习MLP和训练后的Transformer的预测值以及偏导数时，我们观察到强烈的对齐现象。

> pass to transform tokens by gradient-based optimization. To provide evidence for this hypothesis, we build on Schlag et al. (2021) that already provide a linear self-attention layer variant with (fast-)inner loop learning by the error-correcting delta rule (Widrow & Hoff, 1960). We diverge from their set- ting and focus on (in-context) learning where we specifically construct a dataset by considering neighboring elements in the input sequence as input- and target training pairs, see assumptions of Proposition 1. This construction could be realized, for example, due to the model learning to imple- ment a copying layer, see section 4 and proposition 3, and allows us to provide a simple and different construction to Schlag et al. (2021) that solely is built on the standard lin- ear, and approximately softmax, self-attention layer but still implements gradient descent based learning dynamics. We, therefore, are able to explain gradient descent based learn- ing in these standard architectures. Furthermore, we extend this construction based on a single self-attention layer and provide an explanation of how deeper K-layer Transformer models implement principled K-step gradient descent learn- ing, which deviates again from Schlag et al. and allows us to identify that deep Transformers implement GD++, an accelerated version of gradient descent. We highlight that our construction of gradient descent and GD++ is not suggestive but when training multi-layer self- attention-only Transformers on simple regression tasks, we provide strong evidence that the construction is actually found. This allows us, at least in our restricted problems settings, to explain mechanistically in-context learning in trained Transformers and its close resemblance to GD ob- served by related work. Further work is needed to incor- porate regression problems with noisy data and weight reg- ularization into our hypothesis. We speculate aspects of learning in these settings are meta-learned – e.g., the weight magnitudes to be encoded in the self-attention weights. Ad- ditionally, we did not analyze logistic regression for which one possible weight construction is already presented in Zhmoginov et al. (2022).

通过基于梯度的优化来转换标记。 为了为这一假设提供证据，我们基于Schlag等人（2021）的研究，该研究已经提供了一种线性自注意力层变体，利用误差校正delta规则（Widrow & Hoff, 1960）进行（快速）内环学习。我们偏离了他们的设置，专注于（上下文）学习，并特别通过考虑输入序列中的相邻元素作为输入-目标训练对来构建数据集，参见命题1的假设。这种构建可以通过模型学习实现复制层来实现，参见第4节和命题3，并且允许我们提供一个简单且不同于Schlag等人（2021）的构建，后者仅基于标准的线性且近似softmax的自注意力层，但仍实现了基于梯度下降的学习动态。因此，我们能够解释这些标准架构中基于梯度下降的学习。此外，我们基于单个自注意力层扩展了这一构建，并解释了更深的K层Transformer模型如何实现有原则的K步梯度下降学习，这再次与Schlag等人不同，并使我们能够识别出深层Transformer实现了GD++，即梯度下降的加速版本。 我们强调，我们对梯度下降和GD++的构建并非暗示性的，而是在简单回归任务上训练多层仅自注意力Transformer时，我们提供了强有力的证据表明该构建确实被找到。这允许我们至少在我们受限的问题设置中，从机制上解释训练好的Transformer中的上下文学习及其与相关工作中观察到的GD的密切相似性。需要进一步的工作将带有噪声数据的回归问题和权重正则化纳入我们的假设。我们推测在这些设置中学习的一些方面是元学习的——例如，权重幅度被编码在自注意力权重中。此外，我们没有分析逻辑回归，Zhmoginov等人（2022）已经提出了一种可能的权重构建。

> Our refined understanding of in-context learning based on 0.08 GD Trained TF Partial cosine 1.0 0.06 0.9 Cosine sim L2 Norm 0.8 0.04 0.7 0.02 Preds diff Partial diff 0.6 0.5 0.00 Training steps Training steps gradient descent motives us to investigate how to improve it. We are excited about several avenues of future research. First, to exceed upon a single step of gradient descent in every self-attention layer it could be advantageous to incor- porate so called declarative nodes (Amos & Kolter, 2017; Bai et al., 2019; Gould et al., 2021; Zucchet & Sacramento, 2022) into Transformer architectures. This way, we would treat a single self-attention layer as the solution of a fully optimized regression loss leading to possibly more efficient architectures. Second, our findings are restricted to small Transformers and simple regression problems. We are ex- cited to delve deeper into research trying to understand how further mechanistic understanding of Transformers and in- context learning in larger models is possible and to what extend. Third, we are excited about targeted modifications to Transformer architectures, or their training protocols, leading to improved gradient descent based learning algo- rithms or allow for alternative in-context learners to be im- plemented within Transformer weights, augmenting their functionality, as e.g. in Dai et al. (2023). Finally, it would be interesting to analyze in-context learning in HyperTrans- formers (Zhmoginov et al., 2022) that produce weights for target networks and already offer a different perspective on merging Transformers and meta-learning. There, Transform- ers transform weights instead of data and could potentially allow for gradient computations of weights deep inside the target network lifting the limitation of GD on linear models analyzed here.

基于梯度下降对我们对上下文学习（in-context learning）的精细化理解，促使我们探索如何改进它。我们对几个未来的研究方向感到兴奋。首先，为了超越每个自注意力层中单步梯度下降的效果，将所谓的声明式节点（declarative nodes）（Amos & Kolter, 2017; Bai et al., 2019; Gould et al., 2021; Zucchet & Sacramento, 2022）整合到 Transformer 架构中可能具有优势。通过这种方式，我们将单个自注意力层视为一个完全优化的回归损失函数的解，从而可能获得更高效的架构。其次，我们的发现仅限于小型 Transformer 和简单的回归问题。我们渴望深入研究，以理解如何以及在多大程度上，对大型模型中 Transformer 和上下文学习进行更深入的机制性理解是可能的。第三，我们对针对 Transformer 架构或其训练协议的特定修改感到兴奋，这些修改可能导致基于梯度下降的学习算法得到改进，或者允许在 Transformer 权重中实现替代的上下文学习器，从而增强其功能，例如 Dai et al. (2023) 中的做法。最后，分析 HyperTransformers（Zhmoginov et al., 2022）中的上下文学习将很有趣，因为它们为目标网络生成权重，并且已经提供了将 Transformer 与元学习（meta-learning）相结合的不同视角。在那里，Transformer 转换的是权重而不是数据，并且有可能允许对目标网络内部深处的权重进行梯度计算，从而突破本文分析的线性模型中梯度下降的限制。

Acknowledgments Jo˜ao Sacramento and Johannes von Oswald deeply thank Angelika Steger for her support and guidance. The authors also thank Seijin Kobayashi, Marc Kaufmann, Nicolas Zuc- chet, Yassir Akram, Guillaume Obozinski and Mark Sandler for many valuable insights throughout the project and Dale Schuurmans and Timothy Nguyen for their valuable com- ments on the manuscript. Jo˜ao Sacramento was supported by an Ambizione grant (PZ00P3 186027) from the Swiss National Science Foundation and an ETH Research Grant (ETH-23 21-1). Transformers Learn In-Context by Gradient Descent

致谢 João Sacramento 和 Johannes von Oswald 衷心感谢 Angelika Steger 的支持与指导。作者们还感谢 Seijin Kobayashi、Marc Kaufmann、Nicolas Zucchet、Yassir Akram、Guillaume Obozinski 和 Mark Sandler 在项目过程中提供的许多宝贵见解，以及 Dale Schuurmans 和 Timothy Nguyen 对稿件提出的宝贵意见。João Sacramento 获得了瑞士国家科学基金会 Ambizione 资助（PZ00P3 186027）和 ETH 研究资助（ETH-23 21-1）的支持。 Transformer 通过梯度下降学习上下文

— 参考文献（原文保留） —

Aky¨urek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. What learning algorithm is in-context learn- ing? investigations with linear models. In The Eleventh International Conference on Learning Representations, 2023. URL https://openreview.net/forum? id=0g0X4H8yN4I. Amos, B. and Kolter, J. Z. Optnet: Differentiable opti- mization as a layer in neural networks. In International Conference on Machine Learning, 2017. Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In Advances in Neural Information Processing Systems, 2016. Ba, J., Hinton, G. E., Mnih, V., Leibo, J. Z., and Ionescu, C. Using fast weights to attend to the recent past. In Advances in Neural Information Processing Systems 29, 2016. Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. Advances in Neural Information Processing Systems, 2019. Bengio, Y., Bengio, S., and Cloutier, J. Learning a synaptic learning rule. Technical report, Universit´e de Montr´eal, D´epartement d’ Informatique et de Recherche op´erationnelle, 1990. Benzing, F., Schug, S., Meier, R., von Oswald, J., Akram, Y., Zucchet, N., Aitchison, L., and Steger, A. Random initialisations performing above chance and how to find them. OPT2022: 14th Annual Workshop on Optimization for Machine Learning, 2022. Bertinetto, L., Henriques, J. F., Torr, P. H. S., and Vedaldi, A. Meta-learning with differentiable closed-form solvers. In International Conference on Learning Representations, 2019.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. arXiv preprint Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In Computer Vision – ECCV 2020. Springer International Publishing, 2020. Chalmers, D. J. The evolution of learning: an experiment in genetic connectionism. In Touretzky, D. S., Elman, J. L., Sejnowski, T. J., and Hinton, G. E. (eds.), Connectionist Models, pp. 81–90. Morgan Kaufmann, 1991. Chan, S. C. Y., Dasgupta, I., Kim, J., Kumaran, D., Lampinen, A. K., and Hill, F. Transformers generalize differently from information stored in context vs in weights. arXiv preprint arXiv:2210.05675, 2022a. Chan, S. C. Y., Santoro, A., Lampinen, A. K., Wang, J. X., Singh, A., Richemond, P. H., McClelland, J., and Hill, F. Data distributional properties drive emergent in-context learning in transformers. Advances in Neural Information Processing Systems, 2022b. Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mo- hiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. Rethinking attention with performers. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum? id=Ua6zuk0WRH.

Dai, D., Sun, Y., Dong, L., Hao, Y., Ma, S., Sui, Z., and Wei, F. Why can GPT learn in-context? language models im- plicitly perform gradient descent as meta-optimizers. In ICLR 2023 Workshop on Mathematical and Empirical Un- derstanding of Foundation Models, 2023. URL https: //openreview.net/forum?id=fzbHRjAd8U. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2021. URL https:// openreview.net/forum?id=YicbFdNTTy. Entezari, R., Sedghi, H., Saukh, O., and Neyshabur, B. The role of permutation invariance in linear mode connectivity of neural networks. arXiv preprint arXiv:2110.06296, 2021. Finn, C. and Levine, S. Meta-learning and universality: Deep representations and gradient descent can approx- imate any learning algorithm. In International Confer- ence on Learning Representations, 2018. URL https: //openreview.net/forum?id=HyjC5yWCW. Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta- learning for fast adaptation of deep networks. In Interna- tional Conference on Machine Learning, 2017. Flennerhag, S., Rusu, A. A., Pascanu, R., Visin, F., Yin, H., and Hadsell, R. Meta-learning with warped gradi- ent descent. In International Conference on Learning Representations, 2020. Transformers Learn In-Context by Gradient Descent Garg, S., Tsipras, D., Liang, P., and Valiant, G. What can transformers learn in-context? a case study of sim- ple function classes. In Oh, A. H., Agarwal, A., Bel- grave, D., and Cho, K. (eds.), Advances in Neural In- formation Processing Systems, 2022. URL https: //openreview.net/forum?id=flNZJ2eOet.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. Meta-learning probabilistic inference for prediction. In International Conference on Learning Representations, 2019. URL https://openreview. net/forum?id=HkxStoC5F7. Gould, S., Hartley, R., and Campbell, D. J. Deep declarative networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021. Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., and Pang, R. Conformer: Convolution-augmented transformer for speech recognition. arXiv preprint arXiv:2005.08100, 2020. Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415, 2016. Hinton, G. E. and Plaut, D. C. Using fast weights to deblur old memories. 1987. Hochreiter, S., Younger, A. S., and Conwell, P. R. Learning to learn using gradient descent. In Dorffner, G., Bischof, H., and Hornik, K. (eds.), Artificial Neural Networks — ICANN 2001, pp. 87–94, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44668-2. Hubinger, E., van Merwijk, C., Mikulik, V., Skalse, J., and Garrabrant, S. Risks from learned optimization in advanced machine learning systems. arXiv [cs.AI], Jun 2019. URL http://arxiv.org/abs/1906.01820. Irie, K., Schlag, I., Csord´as, R., and Schmidhuber, J. Going beyond linear transformers with recurrent fast weight programmers. CoRR, abs/2106.06295, 2021. URL https://arxiv.org/abs/2106.06295. Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014.

Kirsch, L. and Schmidhuber, J. Meta learning backpropaga- tion and improving it. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), Advances in Neural Information Processing Systems, 2021. URL https: //openreview.net/forum?id=hhU9TEvB6AF. Kirsch, L., Harrison, J., Sohl-Dickstein, J., and Metz, L. General-purpose in-context learning by meta-learning transformers. In Sixth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems, 2022. URL https://openreview.net/forum? id=t6tA-KB4dO. Lee, K., Maji, S., Ravichandran, A., and Soatto, S. Meta- learning with differentiable convex optimization. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019. Lee, Y. and Choi, S. Gradient-based meta-learning with learned layerwise metric and subspace. In International Conference on Machine Learning, 2018. Li, Z., Zhou, F., Chen, F., and Li, H. Meta-SGD: Learning to learn quickly for few shot learning. arXiv preprint Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. arXiv preprint arXiv:2107.13586, 2021. Nadaraya, E. A. On estimating regression. Theory of Prob- ability & its Applications, 9(1):141–142, 1964. Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. In- context learning and induction heads. arXiv preprint

Park, E. and Oliva, J. B. Meta-curvature. In Advances in Neural Information Processing Systems, 2019. Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. Grokking: Generalization beyond overfitting on small algorithmic datasets. abs/2201.02177, 2022. Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. In International Conference on Learning Representations, 2020. Ramsauer, H., Sch¨afl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlovi´c, M., Sandve, G. K., Greiff, V., Kreil, D., Kopp, M., Klambauer, G., Brandstetter, J., and Hochreiter, S. Hopfield networks is all you need. arXiv preprint arXiv:2008.02217, 2020. Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. Meta-learning with latent embedding optimization. In International Confer- ence on Learning Representations, 2019. Schlag, I., Irie, K., and Schmidhuber, J. Linear transformers are secretly fast weight programmers. In ICML, 2021. Transformers Learn In-Context by Gradient Descent Schmidhuber, J. Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook. Diploma thesis, Institut f¨ur Informatik, Technische Universit¨at M¨unchen, 1987. Schmidhuber, J. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. Neural Computation, 4(1):131–139, 1992. doi: 10.1162/neco. 1992.4.1.131. Thrun, S. and Pratt, L. Learning to learn. Springer US,

1998. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2017. von Oswald, J., Zhao, D., Kobayashi, S., Schug, S., Caccia, M., Zucchet, N., and Sacramento, J. Learning where to learn: Gradient sparsity in meta and continual learning. In Advances in Neural Information Processing Systems, 2021. Watson, G. S. Smooth regression analysis. Sankhy¯a: The Indian Journal of Statistics, Series A, pp. 359–372, 1964. Widrow, B. and Hoff, M. E. Adaptive switching circuits. In 1960 IRE WESCON Convention Record, Part 4, pp. 96–104, New York, 1960. IRE. Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. Graph transformer networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d ´Alch´e-Buc, F., Fox, E., and Garnett, R. (eds.), Advances in Neural Information Processing Systems, 2019. Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. Dive into deep learning. arXiv preprint arXiv:2106.11342, 2021. Zhao, D., Kobayashi, S., Sacramento, J., and von Oswald, J. Meta-learning via hypernetworks. In NeurIPS Workshop on Meta-Learning, 2020. Zhmoginov, A., Sandler, M., and Vladymyrov, M. Hy- perTransformer: Model generation for supervised and semi-supervised few-shot learning. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), Proceedings of the 39th International Confer- ence on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pp. 27075–27098. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr. press/v162/ zhmoginov22a.html.

Zucchet, N. and Sacramento, J. Beyond backpropagation: bilevel optimization through implicit differentiation and equilibrium propagation. Neural Computation, 34(12), December 2022. Transformers Learn In-Context by Gradient Descent A. Appendix A.1. Proposition 1 First, we highlight the dependency on the tokens ei of the linear self-attention operation X PhVhKT ej ←ej + LSAθ({e1, . . . , eN}) = ej + h with ⊗the outer product between two vectors. With this we can now easily draw connections to one step of gradient descent on L(W) = 2N PN i=1 ∥Wxi −yi∥2 with learning rate η which yields weight change ΔW = −η∇W L(W) = −η We first restate Proposition 1. Given a 1-head linear attention layer and the tokens ej = (xj, yj), for j = 1, . . . , N, one can construct key, query and value matrices WK, WQ, WV as well as the projection matrix P such that a Transformer step on every token ej is identical to the gradient-induced dynamics ej ←(xj, yj) + (0, −ΔWxj) = (xi, yi) + P V KT qj such that ej = (xj, yj −Δyj). For the test data token (xN+1, yN+1) the dynamics are identical. We provide the weight matrices in block form: WK = WQ = 0 W0 −Iy Ny respectively. Furthermore, we set WV = wish to train and P = η N I with identity matrix of size Nx + Ny. With this simple construction we obtain the following dynamics W0 −Iy N X j i + η ← N I yj yj i=1 N X j W0xi −yi + η = N I yj i=1 for every token ej = (xj, yj) including the query token eN+1 = etest = (xtest, −W0xtest) which will give us the desired result. A.2. Comparing the out-of-distribution behavior of trained Transformers and GD

We provide more experimental results when comparing GD with tuned learning rate η and data transformation scalar γ and the trained Transformer on other data distributions than provided during training, see Figure 6. We do so by changing the in-context data distribution and measure the loss of both methods averaged over 10.000 tasks when either changing α that 1) affects the input data range x ∼U(−α, α)Nx or 2) the teacher by αW with W ∼N(0, I). This setups leads to results shown in the main text, in the first two columns of Figure 6 and in the corresponding plots of Figure 7. Although the match for deeper architectures starts to become worse, overall the trained Transformers behaves remarkably similar to GD and GD++ for layer depth greater than 1. Furthermore, we try GD and the trained Transformer on input distributions that it never has seen during training. Here, we chose by chance of 1/3 either a normal, exponential or Laplace distribution (with JAX default parameters) and depict the average loss value over 10.000 tasks where the α value now simply scales the input values that are sampled from one of the distributions αx. The teacher scaling is identical to the one described above. See for results the two right columns of X X vh,i ⊗kh,iqh,j h qh,j = ej + Ph i h X X eh,i ⊗eh,iW T = ej + PhWh,V h,KWh,Qej (6) i h N X (Wxi −yi)xT i . (7) N i=1 x with Ix and Iy the identity matrices of size Nx and with the weight matrix W0 ∈RNy×Nx of the linear model we x j x i ⊗ yj yi yi −ΔWxj i j

j ⊗ . (8) + = yj Transformers Learn In-Context by Gradient Descent (a) Comparing one step of gradient descent with trained one layer Transformers on OOD data. Test on larger inputs Test on larger targets GD Interpolated Trained TF Loss Loss GD Interpolated Trained TF 0.5 1.0 1.5 2.0 where x U(,) W where W N(0, I) (b) Comparing two steps of gradient descent with trained recurrent two layer Transformers on OOD data. Test on larger inputs Test on larger targets GD Interpolated Trained TF Loss Loss GD GD+ + Interpolated Trained TF 0.5 1.0 1.5 2.0 where x U(,) W where W N(0, I) (c) Comparing five steps of gradient descent with trained five layer Transformers on OOD data. Test on larger inputs Test on larger targets GD GD+ + Trained TF Loss Loss GD GD+ + Trained TF 0.5 1.0 1.5 2.0 where x U(,) W where W N(0, I) all setups, the trained Transformer behaves remarkably similar to GD or GD++. Right & center right: Comparing Transformers, GD and their weight interpolation on data distributions never seen during training. Again, in all setups, the trained Transformer behaves remarkably similar to GD or GD++ with less good match for deep non-recurrent Transformers far away from training regimes. architectures far away from the training range of α = 1. Note that for deeper Transformers (K > 2) the corresponding GD and GD++ version, see for more experimental details Appendix section A.12, we include a harsh clipping of the token values after every step of transformation between [−10, 10] (for the trained TF and GD) to improve training stability. Therefore, the loss increase is restricted to a certain value and plateaus.

A.3. Linear mode connectivity between the weight construction of Prop 1 and trained Transformers In order to interpolate between the construction θGD and the trained weights of the Transformer θ, we need to correct for some scaling ambiguity. For clarification, we restate here the linear self-attention operation for a single head ej ←ej + PWV = ej + WP V Now, to match the weight construction of Prop. 1 we have the aim for the matrix product WKQ to match an identify matrix (except for the last diagonal entry) after re-scaling. Therefore we compute the mean of the diagonal of the matrix product of the trained Transformer weights WKQ which we denote by β. After resealing both operations i.e. WKQ ←WKQ/β and WP V ←WP V β we interpolate linearly between the matrix products of GD as well as these rescaled trained matrix products i.e. WI,KQ = (WGD,KQ + WT F,KQ)/2 as well as WI,P V = (WGD,P V + WT F,P V)/2. We use these parameters to obtain results throughout the paper denote with Interpolated. We do so for GD as well as GD++ when comparing to Test on larger inputs Test on larger targets Loss Loss GD Interpolated Trained TF GD Interpolated Trained TF W where W N(0, I) where x Test on larger inputs Test on larger targets GD GD+ + Interpolated Trained TF Loss Loss GD GD+ + Interpolated Trained TF W where W N(0, I) where x Test on larger inputs Test on larger targets GD GD+ + Trained TF Loss Loss GD GD+ + Trained TF W where W N(0, I) where x X ei ⊗eiW T KWQej (9) i X ei ⊗eiWKQej (10) i Transformers Learn In-Context by Gradient Descent

GD vs trained TF 0.4 2.0 GD GD+ + 0.3 Trained TF 1.5 L2 Norm Loss 0.2 1.0 0.5 0.1 0.0 Training steps Training steps Test on larger inputs Test on larger targets GD Interpolated Trained TF Loss Loss GD GD+ + Interpolated Trained TF 0.5 1.0 1.5 2.0 where x U(,) W where W N(0, I) Transformer with two layers, see Figure 3, but now with 10 repeated layers. We again observe for deeper recurrent linear self-attention only Transformers that overall GD++ and the trained Transformer align very well with one another and are again interpolatable leading to very similar behavior insight as well as outside training situations. Note the inferior performance to the non-recurrent five-layer Transformer which highlights the importance on specific learning rate as well γ parameter per layer/step. layer. Results comparable to the deep recurrent Transformer, see Figure 7, but now with 12 independent Transformer blocks including MLPs and 4-head linear self-attention. We omit LayerNorm. We again observe a close resemblance of the trained Transformers and GD++. We hypotheses that even when equipped with multiple heads and MLPs, Transformers approximate GD++. GD+ + vs trained TF 2.0 1.05 1.05 Model cos Model cos 1.00 1.00 1.5 Cosine sim Cosine sim L2 Norm 0.95 0.95 Preds diff Model diff Preds diff Model diff 1.0 0.90 0.90 0.5 0.85 0.85 0.80 0.80 0.0 Training steps Test on larger inputs Test on larger targets GD GD+ + Interpolated Trained TF Loss Loss GD GD+ + Interpolated Trained TF W where W N(0, I) where x Transformers Learn In-Context by Gradient Descent Weights of WT

Weight of PWV KWV 1.0 1.0 0.5 0.5 0.0 0.0 0.5 0.5 1.0 1.0 1 2 3 4 5 6 7 8 9 1011 1 2 3 4 5 6 7 8 9 1011 self-attention layer. We see (after scalar correction) a perfect resemblance of our construction. Right & outer right: Weight matrix products of a trained 3-layer recurrent linear self-attention Transformer. Again, we see (after scalar correction) a perfect resemblance of our construction and an additional curvature correction i.e. diagonal values in PWV of the same magnitude except the last entry that functions as the learning rate. recurrent Transformers. Note that for non-recurrent Transformers, we face more ambiguity that we have to correct for since e.g. scalings influence each other across layer. We also see this in practice and are not able (only for some seeds) to interpolate between weights with our simple correction from above. We leave the search for more elaborate corrections for future work. A.4. Visualizing the trained Transformer weights The simplicity of our construction enables us to visually compare trained Transformers and the construction put forward in Proposition A.1 in weight space. As discussed in the previous section A.3 there is redundancy in the way the trained Transformer can construct the matrix products leading to the weights corresponding to gradient descent. We therefore visualize WKQ = W T KWQ as well as WP V = PKWV in Figure 9. A.5. Proof and discussion of Proposition 3 We state here again Proposition 3, provide the necessary construction and a short discussion. Proposition 3. Given a 1-head linear- or softmax attention layer and the token construction e2j = (xj), e2j+1 = (0, yj) with a zero vector 0 of dim Nx −Ny and concatenated positional encodings, one can construct key, query and value matrix WK, WQ, WV as well as the projection matrix P such that all tokens ej are transformed into tokens equivalent to the ones required in proposition 1.

To get a simple and clean construction, we choose wlog xj ∈R2N+1 and (0, yj) ∈R2N+1 as well as model the positional encodings as unit vectors pj ∈R2N+1 and concatenate them to the tokens i.e. ej = (xj/2, pj). We wish for a construction that realizes ⊠⊠xj/2 + PV KT WQ ej ← pj ⊠j/2 ⊠+ pj This means that a token replaces its own positional encoding by coping the target data of the next token to itself leading to ej = (xj/2, 0, yj/2+1), with slight abusive of notation. This can simply be realized by (for example) setting P = I, ⊠0 Ix −Ix,off ⊠⊠0 Ix ⊠⊠0 IT WV = , WK = and WQ = Nx. Note that then simply KT WQej = pj+1 i.e. it chooses the j + 1 element of V which stays pj+1 if we apply the softmax operation on KT qj. Since the j + 1 entry of V is (0, yj/2+1 −pj) we obtain the desired result. For the (toy-)regression problems considered in this manuscript, the provided result would give N/2 tokens for which we also copy (parts) of xj underneath yj. This is desired for modalities such as language where every two tokens could be considered an in-and output pair for the implicit autoregressive inner-loop loss. These tokens do not have be necessarily next to each other, see for this behavior experimental findings presented in (Olsson et al., 2022). For the experiments conducted here, one solution is to zero out these tokens which could be constructed by a two-head self-attention layer that given uneven j simply subtracts itself resulting in a zero token. For all even tokens, we use the construction from above which effectively coincides with the token construction required in Proposition 1.

Weights of WT Weight of PWV KWV 1 2 3 4 5 6 7 8 9 1011 1 2 3 4 5 6 7 8 9 1011 ⊠⊠xj/2 (11) pj ⊠yj/2+1 −pj ⊠+ . (12) ⊠with Ix,off the lower diagonal identity matrix fo size x,off Transformers Learn In-Context by Gradient Descent Rolling out experiment with different dampening strength Dampening = 1 Dampening = 0.875 GD Trained TF 0.2 0.2 Loss Loss 0.1 0.1 0.0 0.0 GD Steps / Transformer Layers GD Steps / Transformer Layers dampening strengths affect the generalization of both methods with slightly better robustness for GD which matching performance for 50 steps when λ = 0.75. A.6. Dampening the self-attention layer As an additional out-of-distribution experiment, we test the behavior when repeating a single LSA-layer trained to lower our objective, see equation 5, with the aim to repeat the learned learning/update rule. Note that GD as well as the self- attention layer were optimized to be optimal for one step. For GD we line search the otpimal learning rate η on 10.000 task. Interestingly, for both methods we observe quick divergence when applied multiple times, see left plot of Figure 10. Nevertheless, both of our update functions are described by a linear self-attention layer for which we can control the norm, post training, by a simple scale which we denote as λ. This results in the new update ytest + λ∆Wxtest for GD and ytest + λPV KT WQxtest for the trained self-attention layer which effectively re-tunes the learning rate for GD and the trained self-attention layer. Intriguingly, both methods do generalize similarly well (or poorly) on this out-of-distribution experiment when changing λ, see again Figure 10. We show in Figure 1 the behavior for λ = 0.75 for which we see both methods steadily decreasing the loss within 50 steps.

A.7. Sine wave regression For the sine wave regression tasks, we follow (Finn et al., 2017) and other meta-learning literature and sample for each task an amplitude a ∼U(0.1, 5) and a phase ρ ∼U(0, π). Each tasks consist of N = 10 data points where inputs are sampled x ∼U(−5, 5) and targets computed by y = a sin(ρ + x). We choose here for the first time, for GD as well as for the Transformer, an input embedding emb that maps tokens ei = (xi, yi) into a 40 dimensional space emb(ei) = Wembei through an affine projection without bias. We skip the first self-attention layer but, as usually done in Transformers, then transform the embedded tokens through an MLP m with a single hidden layer, widening factor of 4 (160 hidden neuros) and GELU nonlinearity (Hendrycks & Gimpel, 2016) i.e. ej ←m(emb(ej)) + emb(ej). We interpret the last entry of the transformed tokens as the (transformed) targets and the rest as a higher-dimensional input data representation on which we train a model with a single gradient descent step. We compare the obtained meta-learned GD solution with training a Transformer on the same token embeddings but instead learn a self-attention layer. Note that the embeddings of the tokens, including the transformation through the MLP, are not dependent on an interplay between the tokens. Furthermore, the initial transformation is dependent on ei = (xi, yi), i.e., input as well as on the target data except for the query token for which ytest = 0. This means that this construction is, except for the additional dependency on targets, close to a large corpus of meta-learning literature that aims to find a deep representation optimized for (fast) fine tuning and few-shot learning. In order to compare the meta-training of the MLP and the Transformer, we choose the same seed to initialize the network weights for the MLPs and the input embedding trained by meta-learning i.e. backprop through training or the Transformer. This leads to the plots and almost identical learned initial function and updated functions shown in Figure 4.

A.8. Proposition 2 and connections between gradient descent, kernelized regression and kernel smoothing Let’ s consider the data transformation induced by an MLP ˜m(x) and a residual connection commonly used in Transformer blocks i.e. ej ←ej + ˜m(ej) = (xj, yj) + (˜m(xj), 0) = (m(xj), yj) with m(xj) = xj + ˜m(xj) and ˜m not changing the targets y. When simply applying Proposition 1, it is easy to see that given this new token construction, a linear self-attention layer can induce the token dynamics ej ←(m(xj), yj)+(0, −∆Wm(xj)) with ∆W = −η∇L(W) given the loss function L(W) = 2N PN i=1 ||Wm(xi) −yi||2. Dampening = 0.75 GD Trained TF GD Trained TF 0.2 Loss 0.1 0.0 GD Steps / Transformer Layers Transformers Learn In-Context by Gradient Descent Interestingly, for the test token etest = (xtest, 0) this induces, after a multiplication with −1, an initial prediction after a single Transformer block given by ˆy = ∆Wm(xtest) = −η∇W L(0)m(xtest) = with m(xi)T m(xtest) = k(xi, xtest) ∈R interpreted as a kernel function. Concluding, we see that the combination of MLPs and a single self-attention layer can lead to dynamics induced when descending a kernelized regression (squared error) loss with a single step of gradient-descent. Interestingly, when choosing W0 = 0, we furthermore see that a single self-attention layer or Transformer block can be regarded as doing nonparametric kernel smoothing ˆy = PN i=1 yik(xi, xtest) based on the data given in-context (Nadaraya, 1964; Watson, 1964). Note that we made a particular choice of kernel function here and that this view still holds when m(xj) = 1 i.e. consider Transformers without MLPs or leverage the well-known view of softmax self-attention layer as a kernel function used to measure similarity between tokens (e.g. Choromanski et al., 2021; Zhang et al., 2021). Thus, implementing one step of gradient descent through a self-attention layer (w/wo softmax nonlinearity) is equivalent to performing kernel smoothing estimation. We however argue that this nonparametric kernel smoothing view of in-context learning is limited, and arises from looking only at a single self-attention layer. When considering deeper Transformer architectures, we see that multiple Transformer blocks can iteratively transform the targets based on multiple steps of gradient descent leading to minimization of a kernelized squared error loss L(W). One way to obtain a suitable construction is by neglecting MLPs everywhere except in the first Transformer block. We leave the study of the exact mechanics, especially how the Transformer makes use of possibility transforming the targets through the MLPs, and the possibility of iteratively changing the kernel function throughout depth for future study.

A.9. Linear vs. softmax self-attention as well LayerNorm Transformers Although linear Transformers and their variants have been shown to be competitive with their softmax counterpart (Irie et al., 2021), the removal of this nonlinearity is still a major departure from classic Transformers and more importantly from the Transformers used in related studies analyzing in-context learning. In this section we investigate whether and when gradient-based learning emerges in trained softmax self-attention layers, and we provide an analytical argument to back our findings. First, we show, see Figure 12, that a single layer of softmax self-attention is not able to match GD performance. We tuned the learning rate as well as the weight initialization but found no significant difference over the hyperparameters we used througout this study. In general, we hypothesize that GD is an optimal update given the limited capacity of a single layer of (single-head) self-attention. We therefore argue that the softmax induces (at best) a linear offset of the matrix product of training data and query vector 1 qj, . . . , ekT Nqj)T /(softmax(KT qj) = (ekT 1 WKQxj, . . . , exT NWKQxj)T /(= (exT 1 WKQxj, . . . , 1 + xT NWKQxj)T /(≈(1 + xT ∝KT qj + ϵ (17) proportional to a factor dependent on all {xτ,i}N+1 i=1 . We speculate that the dependency on the specific task τ, for large Nx vanishes or that the x-dependent value matrix could introduce a correcting effect. In this case the softmax operation introduces an additive error w.r.t. to the optimal GD update. To overcome this disadvantageous offset, the Transformer can (approximately) introduce a correction with a second self-attention head by a simple subtraction i.e.

P1V1softmax(KT 1 WQxj) + P2V2softmax(KT 2 WQxj) (18) ≈PV ((1 + xT 1 W1,KQxj, . . . , 1 + xT NW1,KQxj) −(1 + xT = PV (xT 1 (W1,KQ −W2,KQ)xj, . . . , xT N(W1,KQ −W2,KQ)xj) (20) ∝PV KT qj. (21) N X N X yim(xi)T m(xtest) = yik(xi, xtest) (13) i=1 i=1 ekT X i qj) (14) i exT X i WKQxj) (15) i X 1 + xT i WKQxj) (16) i 1 W2,KQxj, . . . , 1 + xT NW2,KQxj)) (19) Transformers Learn In-Context by Gradient Descent 1W1, KQ 1 2 3 4 5 6 7 8 9 1011 1 2 3 4 5 6 7 8 9 1011 plot show the matrix product WKQ = W T KWQ including its scaling by η induced through PWV of the two heads of the trained softmax self-attention layer. We observe that both of the matrices are approximate diagonal almost perfect sign reversed values on the off-diagonal terms. After adding the matrices (right plot), we observe a diagonal matrix and therefore to much improved approximation of our construction and therefore gradient descent dynamics. Here we assume that PV 1) subsumes the dividing factor of the softmax and that 2) is the same (up to scaling) for each head. Note that if (W1,KQ −W2,KQ) is diagonal, and P and V chosen as in the Proposition of Appendix A.1, we recover our gradient descent construction. We base this derivation on empirical findings, see Figure 12, that, first of all, show the softmax self-attention performance increases drastically when using two heads instead of one. Nevertheless, the self-attention layer has difficulties to match the loss values of a model trained with GD. Furthermore, this architecture change leads to a very much improved alignment of the trained model and GD. Second, we can observe that when training a two-headed softmax self-attention layer on regression tasks the correction proposed above is actually observed in weight space, see Figure 11. Here, we visualize the matrix product within the softmax operation Wh,KQ per head which we scale with the last diagonal entry of PhWh,V which we denote by ηh = PhWh,V (−1, −1). Intriguingly, this results in an almost perfect cancellation (right plot) of the off-diagonal terms and therefore in sum to an improved approximation of our construction, see the derivation above.

We would like to reiterate that the stronger inductive bias for copying data of the softmax layer remains, and is not invalidated by the analysis above. Therefore, even for our shallow and simple constructions they indeed fulfill an important role in support for our hypotheses: The ability to merge or copy input and target data into single tokens allowing for their dot product computation necessary for the construction in Proposition 1, see Section 4 in the main text. We end this section by analysing Transformers equipped with LayerNorm which we apply as usually done before the self-attention layer: Overall, we observe qualitatively similar results to Transformers with softmax self-attention layer i.e. a decrease in performance compared to GD accompanied with a decrease in alignment between models generated by the Transformer and models trained with GD, see Figure 14. Here, we test again a single linear self-attention layer succeeding LayerNorm as well as two layers where we skip the first LayerNorm and only include a LayerNorm between the two. Including more heads does not help substantially. We again assume the optimality of GD and argue that information of targets and inputs present in the tokens is lost by averaging when applying LayerNorm. This naturally leads to decreasing performance compared to GD, see first row of Figure 14. Although the alignment to GD and GD++, especially for two layers, is high, we overall see inferior performance to one or two steps of GD or two steps of GD++. Nevertheless, we speculate that LayerNorm might not only stabilize Transformer training but could also act as some form of data normalization procedure that implicitly enables better generalization for larger inputs as well as targets provided in-context, see OOD experiments in Figure 14.

Overall we conclude that common architecture choices like softmax and LayerNorm seem supoptimal for the constructed in-context learning settings when comparing to GD or linear self-attention. Nevertheless, we speculate that the potentially small performance drops of in-context learning are negligible when turning to deep and wide Transformers for which these architecture choices have empirically proven to be superior. 1W1, KQ + 2W2, KQ 2W2, KQ 1 2 3 4 5 6 7 8 9 1011 Transformers Learn In-Context by Gradient Descent (a) Comparing one step of GD with a trained softmax one-headed self-attention layer. 3.5 0.40 Model cos Preds diff Model diff GD Trained TF 3.0 0.35 2.5 L2 Norm 2.0 Loss 0.30 1.5 1.0 0.25 0.5 0.20 0.0 Training steps Training steps (b) Comparing one step of GD with a trained softmax two-headed self-attention layer. 3.5 0.40 Model cos Preds diff Model diff GD Trained TF 3.0 0.35 2.5 L2 Norm 2.0 Loss 0.30 1.5 1.0 0.25 0.5 0.20 0.0 Training steps Training steps linear regression tasks. Left column: Softmax self-attention is not able to match gradient descent performance with hand-tuned learning rate, but adding a second attention head significantly reduces the gap, as expected by our analytical argument. Center left: The alignment suffers significantly for single-head softmax SA. We observe good but not as precise alignment when compared to linear Transformers for the two-headed softmax SA layer. Center right & right: The two-headed self-attention compared to the single-head layer shows similar robust out-of-distribution behavior compared to gradient descent.

A.10. Details of curvature correction We give here a precise construction showing how to implement in a single head, a step of GD and the discussed data transformation, resulting in GD++. Recall again the linear self-attention operation with a single head ej ←ej + PWV We provide again the weight matrices in block form of the construction of Prop. 1 but now enabling additionally our □k □described data transformation: WK = WQ = □□k W −Iy with the weight matrix W ∈RNy×Nx of the linear model we wish to train and Furthermore, we set WV = □□−γIx η N P = . This leads to the following update □N X □□k W −Iy □j □□j □ □γIx η N ← + yj yj i=1 □N X □j □□γIx η N □i Wxi −yi = + yj i=1 for every token ej = (xj, yj) including the query token eN+1 = etest = (xtest, 0) which will give us the desired result. Why does GD++ perform better? We give here one possible explanation of the superior performance of GD++ compared to GD. Note that there is a close resemblance of the GD transformation and a heavily truncated Neuman series approximation of the inverse XXT . We provide here a more heuristic explanation for the observed acceleration. Given γ ∈R, GD++ transforms every input according to xi ←xi −γXXT xi = (I −γXXT)xi. We can therefore look at the change of squared regression loss L(W) = 1 i=0(Wxi −yi)2 induced by this transformation i.e. L++(W) = PN Test on larger inputs Test on larger targets GD Trained TF 1.0 0.8 Cosine sim 0.6 Loss Loss 0.4 GD Trained TF 0.2 0.0 0.5 1.0 1.5 2.0 where x U(,) W where W N(0, I) Test on larger inputs

Test on larger targets GD Trained TF 1.0 0.8 Cosine sim 0.6 Loss Loss 0.4 GD Trained TF 0.2 0.0 0.5 1.0 1.5 2.0 where x U(,) W where W N(0, I) X ei ⊗eiW T K. (22) i with Ix the identity matrix of size Nx, Iy od size Ny resp. □□i □□□□k □□i □□□□x □□j □⊗ yi yi yj −γXXT xj □□i □□j □□j □□□⊗ = + . (23) −∆Wxj yj Transformers Learn In-Context by Gradient Descent different γ observed in practice. Center: Given we know the maximum and minimum of eigenvalues λ1, λn of the loss Hessian XXT with X = (x0, . . . , xN) for different N, we compare the original condition number (depicted by *' s at γ = 0) and the condition number (in log scale) of the GD++ altered loss Hessian when varying γ. We plot in dotted lines the γ values that we observe in practice which are close the optimal ones i.e. the local minimum derived through our analysis. Right: ForN = 25, we plot for different γ values the distribution of condition numbers κ = λ1/λn for 10000 tasks and observe favorable κ values close to 1 when approaching the γ = 0.099 value was found in practice. The κ values quickly explode for γ > 0.1. i=0(W(I −γXXT)xi −yi)2 = 1 2(W(I −γXXT)X −Y)2 which in turn leads to a change of the loss Hessian from ∇2L = XXT to ∇2L++ = (I −γXXT)X((I −γXXT)X)T . PN Given the original Hessian H = XXT = UΣU T with it' s set of sorted eigenvalues {λ1, . . . , λn} and λi ⩾0 on the diagonal matrix Σ we can express the new Hessian through U, Σ i.e. H++ = (I −γXXT)X((I −γXXT)X)T = (I −γUΣU T)UΣU T (I −γUΣU T)T . We can simplify H++ further as H++ = (I −γUΣU T)UΣU T (I −γUΣU T)T = U(Σ −γΣ2)U T U(I −γΣ)U T (24)

= U(Σ −2γΣ2 + γ2Σ3)U T (25) Given the eigenspectrum {λ1, . . . , λn} of H, we obtain an (unsorted) eigenspecturm for H++ with {λ1 −2γλ2 1 + γ2λ3 1, . . . , λn −2γλ2 n + γ2λ3 n} which we visualize in Figure 13 for different γ observed in practice. We hypotheses that the Transformer chooses γ in a way that on average, across the distribution of tasks, the data transformation (iteratively) decreases the condition number λ1/λn leading to accelerated learning. This could be achieved, for example, by keeping the smallest eigenvalue λn ≈λ++ n fixed and choosing γ such that the largest eigenvalue of the transformed data λ++ is reduced, while the original λ1 stays within [λ++ , λ++ n]. To support our hypotheses empirically, we computed the minimum and maximum eigenvalues of XXT across 10000 tasks while changing the number of datapoints N ∈[10, 25, 50, 100] i.e. X = (x0, . . . , xN) leading to better conditioned loss Hessians i.e. [1e−10, 0.097, 0.666, 2.870] and [4.6, 7.712, 10.845, 17.196] as the minimum and maximum eigenvalues of XXT across all tasks where we cut the smallest eigenvalue for N = 10 at 1e−10. Furthermore, we extract the γ values from the weights of optimized recurrent 2-layer Transformers trained on different task distributions and obtain γ values of [0.179, 0.099, 0.056, 0.029], see again Figure 13. Note that the observed eigenvalues stay within [0, 1/γ] i.e. the two roots of f(λ, γ) = λ −2γλ2 + γ2λ3. Given the derived function of eigenvalue change f(λ, γ), we compute the condition number of H++ by dividing the novel maximum eigenvalues λ++

= f(1/(3γ), γ) where λ = 1/(3γ) as the local maximum of f(λ, γ), for fixed γ, and the novel minimum eigenvalue λ++ n = min(f(λ1, γ), f(λn, γ)). Note that with too small γ, we move the original λn closer to the root of f(λ, γ) i.e. λ = 1/γ and therefore can change the smallest eigenvalue. Given the task distribution and its corresponding eigenvalue distribution, we see that choosing γ reduces the new condition number κ++ = λ++ /λ++ n which leads to better conditioned learning, see center plot of Figure 13. Note that the optimal γ based on our derivation above is based on the maximum and minimum eigenvalue across all tasks and does not take the change of the eigenvalue distribution into account. We argue therefore that the simplicity of the arguments above does not capture the task statistics and distribution shifts entirely and therefore obtains a slightly larger γ as an optimal value. Transformers Learn In-Context by Gradient Descent (a) Comparing one step of GD with a single-layer LSA Transformer with LayerNorm. 3.5 0.40 Model cos Preds diff Model diff GD Trained TF 3.0 0.35 2.5 L2 Norm 2.0 Loss 0.30 1.5 1.0 0.25 0.5 0.20 0.0 Training steps Training steps (b) Comparing two steps of GD with a two-layer LSA Transformer with LayerNorm. GD vs trained TF 0.30 GD GD+ + Model cos 1.0 1.5 0.25 Trained TF 0.9 Cosine sim L2 Norm Loss Preds diff Model diff 1.0 0.20 0.8 0.7 0.5 0.15 0.6 0.10 0.5 0.0 Training steps Training steps Left column: The Transformers is not able to match the gradient descent performance with hand-tuned learning rate. Alignment plots: The alignment suffers significantly when comparing to linear self-attention layers although still reasonable alignment is obtained which decreases slightly when comparing to GD++ for the two-layer Transformer.Center right & right: The LayerNorm Transformer outperforms when GD when providing training input data that is significantly larger than the data provided during training.

We furthermore visualize the condition number change for N = 25 and 10000 tasks in the right plot of Figure 13 and observe the distribution moving to desirable κ values close to 1. For γ values larger than 0.1 the distribution quickly exhibits exploding condition numbers. A.11. Phase transitions We comment shorty on the curiously looking phase transitions of the training loss observed in many of our experiments, see Figure 2. Nevertheless, simply switching from a single-headed self-attention layer to a two-headed self-attention layer mitigates the random seed dependent training instabilities in our experiments presented in the main text, see left and center plot of Figure 15. Furthermore, these transitions look reminiscent of the recently observed ” grokking” behaviour (Power et al., 2022). Interestingly, when carefully tuning the learning rate and batchsize we can also make the Transformers trained in these linear regression tasks grokk. For this, we train a single Transformer block (self-attention layer and MLP) on a limited amount of data (8192 tasks), see right plot of Figure 15, and observe grokking like train and test loss phase transitions where test set first increases drastically before experiencing a sudden drop in loss almost matching the desired GD loss of 0.2. We leave a thorough investigation of these phenomena for future study. A.12. Experimental details We use for most experiments identical hyperparameters that were tuned by hand which we list here · Optimizer: Adam (Kingma & Ba, 2014) with default parameters and learning rate of 0.001 for Transformer with depth K < 3 and 0.0005 otherwise. We use a batchsize of 2048 and applied gradient clipping to obtain gradients with global norm of 10. We used the Optax library.

· Haiku weight initialisation (fan-in) with truncated normal and std 0.002/K where K the number of layers. · We did not use any regularisation and observed for deeper Transformers with K > 2 instabilities when reaching GD performance. We speculate that this occurs since the GD performance is, for the given training tasks, already close to divergence as seen when providing tasks with larger input ranges. Therefore, training Transformers also becomes Test on larger inputs Test on larger targets GD Trained TF 1.0 0.8 Cosine sim 0.6 Loss Loss 0.4 GD Trained TF 0.2 0.0 0.5 1.0 1.5 2.0 where x U(,) W where W N(0, I) Test on larger inputs GD+ + vs trained TF GD GD+ + Model cos 1.0 1.5 Trained TF 0.9 Cosine sim L2 Norm Preds diff Model diff 1.0 Loss 0.8 0.7 0.5 0.6 0.5 0.0 0.5 1.0 1.5 2.0 where x U(,) Training steps Transformers Learn In-Context by Gradient Descent attention layer. We observe for some seeds very long initial phases of virtually zero progress after which the loss drops suddenly to the desired GD loss. Center: The same experiment but optimizing a two-headed self-attention layer. We observe fast and robust convergence to the loss of GD. Right: Training a single Transformer block i.e. a self-attention layer with MLP and a reduced training set size of 8192 tasks. We observe grokking like train and test loss phase transitions where test set first increases drastically before experiencing a sudden drop in loss almost matching the desired GD loss of 0.2. instable when we approach GD with an optimal learning rate. In order to stabilize training, we simply clipped the token values to be in the range of [−10, 10].

· When applicable we use standard positional encodings of size 20 which we concatenated to all tokens. · For simplicity, and to follow the provided weight construction closely, we did use square key, value and query parameter matrix in all experiments. · The training length varied throughout our experimental setups and can be read off our training plots in the article. · When training meta-parameters for gradient descent i.e. η and γ we used an identical training setup but usually training required much less iterations. · In all experiments we choose inital W0 = 0 for gradient descent trained models. Inspired by (Garg et al., 2022), we additionally provide results when training a single linear self-attention layer on a fixed number of training tasks. Therefore, we iterate over a single fixed batch of size B instead of drawing new batch of tasks at every iteration. Results can be found in Figure 16. Intriguingly, we find that (meta-)gradient descent finds Transformer weights that align remarkable well with the provided construction and therefore gradient descent even when provided with an arguably very small number of training tasks. We argue that this again highlights the strong inductive bias of the LSA-layer to match (approximately) gradient descent learning in its forward pass. Transformers Learn In-Context by Gradient Descent (a) Comparing 1 step of gradient descent with training a LSA-layer on 128 tasks. GD Trained TF Model cos 1.0 0.8 L2 Norm Loss Preds diff Model diff 0.6 0.4 0.2 0.0 Training steps

Training steps (b) Comparing 1 step of gradient descent with training a LSA-layer on 512 tasks. 0.40 GD Trained TF 3.5 Model cos 3.0 0.35 1.0 2.5 0.8 L2 Norm 2.0 Loss Preds diff Model diff 0.30 0.6 1.5 0.4 1.0 0.25 0.2 0.5 0.0 0.0 Training steps 0.20 Training steps (c) Comparing 1 step of gradient descent with training a LSA-layer on 2048 tasks. 0.40 GD Trained TF 3.5 Model cos 3.0 0.35 1.0 2.5 0.8 L2 Norm 2.0 Loss Preds diff Model diff 0.30 0.6 1.5 0.4 1.0 0.25 0.2 0.5 0.0 0.0 Training steps 0.20 Training steps (d) Comparing 1 step of gradient descent training a LSA-layer on 8192 tasks. 0.40 GD Trained TF 3.5 Model cos 3.0 0.35 1.0 2.5 0.8 L2 Norm 2.0 Loss Preds diff Model diff 0.30 0.6 1.5 0.4 1.0 0.25 0.2 0.5 0.0 0.0 Training steps 0.20 Training steps training set size B. Across our alignment measures as well as our tests on out-of-training behaviour, trained Transformers fail to align with GD when provided with a very small amount of tasks. Nevertheless, we see already almost perfect alignment in our base setting N = Nx = 10 when provided with B > 2048 tasks. In all settings, we train the Transformer on (non-stochastic) gradient descent iterating over a single batch of tasks of size B equal to the number provided in the Figure titles (128, 512, 2048, 8192). Test on larger inputs Test on larger targets GD Interpolated Trained TF Cosine sim Loss Loss GD Interpolated Trained TF 0.1 0.1 0.5 1.0 1.5 2.0 where x U(,) W where W N(0, I) Test on larger inputs Test on larger targets GD Interpolated Trained TF Cosine sim Loss Loss GD Interpolated Trained TF 0.1 0.1 0.5 1.0 1.5 2.0 where x U(,)

W where W N(0, I) Test on larger inputs Test on larger targets GD Interpolated Trained TF Cosine sim Loss Loss GD Interpolated Trained TF 0.1 0.1 0.5 1.0 1.5 2.0 where x U(,) W where W N(0, I) Test on larger inputs Test on larger targets GD Interpolated Trained TF Cosine sim Loss Loss GD Interpolated Trained TF 0.1 0.1 0.5 1.0 1.5 2.0 where x U(,) W where W N(0, I)