通过掩码结构增长实现语言模型预训练提速 2 倍

Yiqun Yao[1], Zheng Zhang[1],Jing Li[2],和 Yequan Wang*[1]

[1]北京智源人工智能研究院，中国北京 [2]哈尔滨工业大学，
中国深圳{yqyao,zhangzheng}@baai.ac.cn,
jingli.phd@hotmail.com, tshwangyequan@gmail.com

arXiv:2305.02869v3 [cs.CL] 6 Apr 2024

摘要

加速大型语言模型预训练是当前研究中的一项关键问题。本文聚焦于通过从小型 Transformer 结构逐步扩展至大型结构来加速预训练。渐进式扩展涉及两个主要研究问题：确定最优的增长计划，以及设计高效的增长算子。在增长计划方面，现有工作尚未充分探索各个单一维度对计划效率的影响。对于增长算子，现有方法依赖新权重的初始化来继承知识，只能实现非严格的函数保持，从而限制了训练动态方面的进一步改进。为解决这些问题，我们提出掩码结构增长（MSG），包括 (i) 涵盖所有可能维度的增长计划，以及 (ii) 不依赖新权重初始化的严格函数保持增长算子。实验表明，MSG 显著快于相关工作：在预训练不同类型的语言模型时，我们实现了最高**2.2 倍** 的加速比，同时保持了相当或更优的下游性能。[1]

1 引言

预训练语言模型（PLM）（Devlin 等，2019；Brown 等，2020；OpenAI，2023）为研究和应用带来了巨大潜力。然而，预训练的高昂计算成本仍然是一个瓶颈。除了计算平台的租用费用，研究周期延长和碳足迹增加（Schwartz 等，2020）也是不容忽视的问题。尽管结构化剪枝（Wang 等，2020；Xia 等，2022；Chen 等，2021）可以加快预训练，但由此产生的模型规模更小，性能和知识容量可能较差（Li 等，2020），从而限制其适应性。相反，我们研究的是如何在加速预训练的同时，仍然生成大规模模型这一问题。我们着眼于从较小模型逐步扩展到较大模型，这一自然的想法受到人脑神经发生的启发（Eriksson 等，1998；Van Praag 等，2002；Deng 等，2010）。尽管 Chinchilla 定律等研究（Hoffmann 等，2022）揭示了给定浮点运算次数预算时的最优模型规模，但只考虑了训练过程中模型规模固定的情况。因此，渐进式扩展的最佳实践仍基本是一片未知领域。

我们考虑两个主要问题：增长计划 和增长算子。增长计划决定模型结构在何时以及何处进行扩展。现有研究（Gong 等人，2019；Gu 等，2021）已经探讨了 Transformer（Vaswani 等，2017）的增长计划，涉及层数和前馈网络（FFNs）的宽度。然而，要找到一种涵盖所有可能增长维度的高效多阶段调度方案，仍然十分困难。增长算子 指的是在扩展过程中执行的操作，旨在继承前一个模型的知识。其中，函数保持（Chen 等人，2016；2022）是一项具有重要理论意义的性质，因为它能够确保

*通讯作者[1]代码已公开发布于 https://github.com/cofe-ai/MSG。

表 1：相关工作的总结。○表示“支持但不具备函数保持性”；✗表示“不支持”；□表示“支持，有时具备函数保持性，但在其他情况下可能会有问题（第 2.3.1 节、第 2.3.2 节）”；✓表示“在所有情况下均支持且具备函数保持性”。

方法	层数	隐藏维度	前馈网络维度	头数	任意初始化	调度	速度加速比*
堆叠（Gong 等人，2019）	○	✗	✗	✗	✗	多阶段	1.65x
CompoundGrow（Gu 等，2021）	○	✗	□	✗	✗	多阶段	1.82x
Staged（Shen 等人，2022）	□	□	□	✗	✗	单阶段/两阶段	~1.4x
Bert2BERT（Chen 等人，2022）	○	□	✓	✓	✗	单阶段	1.82x
LiGO（Wang 等人，2022）	○	○	○	○	○	单阶段	1.82倍
MSG（我们的方法）	✓	✓	✓	✓	✓	多阶段	2.2x

* 由于所有研究的实验设置存在巨大差异，本栏中的大多数结果无法直接比较。我们仅汇总在特定设置下报告的最高结果，并在第4.2节讨论公平比较这一问题。

初始化后的大型模型与小型模型的行为相同。现有算子存在两个潜在缺点：首先，函数保持并不严格，导致某些情况下函数存在差异；其次，它们完全依赖新权重的初始化，从而限制了训练动态的改进（第 2.3.3 节）。

为帮助解决这些问题，我们提出掩码结构增长（MSG），这是一种用于 Transformer 语言模型预训练的新型渐进式学习框架。我们采用掩码机制，先消除新神经元的影响以确保函数保持，再在后续训练中逐步增强它们的作用。MSG 为所有可能的维度提供增长算子（第 2.2 节），在调度设计上具有良好的灵活性，由此得到的增长计划在 Bert（Devlin 等，2019）和 GPT-2（Radford 等，2019）上分别实现了最先进的 2.2 倍和 1.4 倍加速比，同时保持了相当或更优的下游性能。MSG 通过解决层归一化（Ba 等，2016）引发的两难问题，首次在任意扩展中实现了严格的函数保持。此外，MSG 算子不依赖于新权重的特定初始化策略，因此有利于未来的研究。

渐进式扩展在训练大语言模型（LLM）方面也被证明是可行的，但由于资源有限且设置多样，难以开展系统而严谨的研究。有关相关主题，请参阅 FLM-101B 技术报告（Li 等人，2023），其中纳入了源自 MSG 的算子。

总而言之，我们的贡献包括：(i) 我们提出了 MSG：一种用于渐进式预训练的新型框架，并设计了具有最先进加速效果的扩展计划；(ii) 我们证明，由于 MSG 能够严格保持函数，且不依赖于新权重的初始化，因此它是未来研究的坚实基础；(iii) 实验表明，MSG 在各种模型配置下均优于现有方法，并揭示了函数保持所发挥作用的相关见解。

2 预备知识

我们在第 2.1 节定义任务，在第 2.2 节描述增长维度，并在第 2.3 节讨论函数保持，以阐明 MSG 的动机。

2.1 任务形式化

我们的目标是找到最优的扩展计划 $s = (s_0, \ldots\ldots, s_T) \in \mathcal{S}$ 和算子 $g = (g_0, \ldots\ldots, g_T) \in \mathcal{G}$，使模型 f 的预训练总运行时间（实际时间）最短，同时保持其在数据集 $\mathcal{D}$ 上的下游性能不变。s 中的每个 s_t 都是经过第 t 次扩展后的中间结构，以超参数集合（第 2.2 节）表示；算子 g_t 控制如何初始化 s_t 的权重，从而基于之前训练好的模型 f^*_{t-1} 构建新模型 f_t：

$$f_t = g_t(f^*_{t-1}, s_{t-1}, s_t). \tag{1}$$

ˆ 设 $c(f_t)$ 为将每个 f_t 训练至 f^*_t 所需的运行时间成本，f 为一个与最终 f^*_T 结构相同、但未经过扩展且从头开始训练的模型。那么，对于每个中间模型，ˆ 比率 $c(f_t)/c(f)$ 都可以通过类似 (Hoffmann 等，2022) 中的解析表达式进行测量。最后，对于某个评估指标 L（越高越好），我们的任务可表述为：

$$\min_{g\in\mathcal{G}, s\in\mathcal{S}} \sum_{t=0}^{T} c(f_t),\ s.t.\ L(f^*_T, \mathcal{D}) \geq L(\hat{f}, \mathcal{D}). \tag{2}$$

2.2 Transformer 的扩展维度

Transformer 中有 4 个可扩展维度（Vaswani 等，2017）。我们根据对应的超参数为它们命名：$s_t =$（隐藏维度、前馈网络维度、头数、层数）。隐藏维度 是特征嵌入的大小；前馈网络维度 是前馈网络的中间层大小。 -

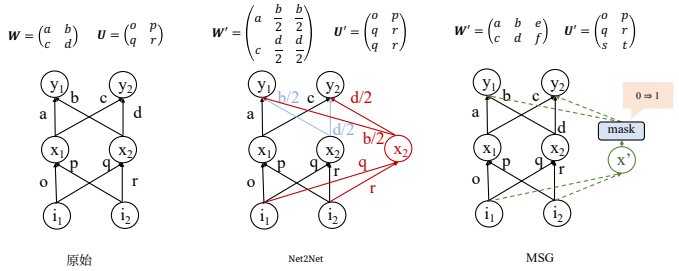


图 1：扩展全连接层时，MSG（右）与 Net2Net（中）的对比。

（前馈神经网络）；头数 是每个多头注意力（MHA）模块中的注意力头数量；层数是 Transformer 模块的总层数。例如，Bert-base 表示为 (768, 3072, 12, 12)，Bert-large 表示为 (1024, 4096, 16,24)（Devlin 等人，2019）。

2.3 函数保持

在公式 1 中，如果 g_t 确保了

$$\forall x, f_t(x) = f_{t-1}^*(x), \tag{3}$$

我们将 g_t a 函数保持 算子命名为函数保持算子。本节首先证明，大多数现有算子在层归一化（2.3.1）和深度维度（2.3.2）上无法实现严格的函数保持。接着，我们研究新权重的初始化，并讨论在哪些情况下严格的函数保持至关重要（2.3.3）。文中提及的方法总结见表 1。

2.3.1 层归一化困境

层归一化（LN）（Ba 等，2016） formulates as:

$$\mathrm{LN}(\boldsymbol{x}) = \boldsymbol{\omega} \odot (\boldsymbol{x} - \mu)/\sigma + \boldsymbol{\beta}, \tag{4}$$

其中，$\boldsymbol{x} \in \mathbb{R}^d$ 是一个特征向量，$\boldsymbol{\omega} \in \mathbb{R}^d$ 和 $\boldsymbol{\beta} \in \mathbb{R}^d$ 分别是可训练的权重和偏置。在大多数实现中，$\mu \in \mathbb{R}$ 和 $\sigma \in \mathbb{R}$ 分别是 $\boldsymbol{x}$中 d 个元素的均值和标准差。$\odot$ 表示逐元素乘积。层归一化（LN）广泛存在于各种模型中。我们证明，层归一化会给主流增长算子带来如下函数保持困境：

Net2Net 全连接网络宽度扩展中一种广泛使用的算子是 Net2Net（Chen 等人，2016）。我们在图 1（中间）中对此进行了演示。当一层 $\boldsymbol{x}$ 从 n扩展到 q个神经元时，Net2Net 会将每个新神经元 i 的 id 映射到一个现有神经元 $m(i)$，如下所示：

$$m(i) = \begin{cases} i & i \leq n \\ \mathrm{map}(\{1, ..., n\}, i) & n < i \leq q \end{cases}. \tag{5}$$

Net2Net 通过从输入权重 $\mathbf{U}$复制权重，并将输出权重$\mathbf{W}$均分，从而初始化新权重，确保神经元 i 是 $m(i)$的副本：

$$\mathbf{U'}_{k,i} = \mathbf{U}_{k,m(i)}, \mathbf{W'}_{i,j} = \frac{1}{|\{x|m(x)=m(i)\}|}\mathbf{W}_{m(i),j}, \tag{6}$$

wh其中 $|\cdot|$表示基数。因此，后续层的输出值保持不变 .

在 Transformer 中，先前的最先进方法（Chen 等人，2022；Qin 等，2022b；Gu 等，2021）本质上是映射函数（式 5）的不同实现，而 Shen 等人（2022）的方法也秉持类似思路。遗憾的是，如果在 $\boldsymbol{x}$之后应用层归一化，Net2Net 就无法严格做到函数保持。我们以 $\boldsymbol{x} = (x_1, x_2, x_3)$ 为例：在隐藏维度从 3 扩展到 5 时，对于 2 个新神经元，Net2Net 会选择现有神经元进行复制。对于 $m(4) = 2$ 和 $m(5) = 1$，结果如下：

$$\boldsymbol{x}' = (x_1, x_2, x_3, x_2, x_1) \Rightarrow \mu(\boldsymbol{x}) \neq \mu(\boldsymbol{x}') \Rightarrow \mathrm{LN}(\boldsymbol{x}) \neq \mathrm{LN}(\boldsymbol{x}'), \tag{7}$$

即使相应地复制了 LN ω 和 β，情况依然如此。因此，$\boldsymbol{y}' = \mathbf{W}'(\mathrm{LN}(\boldsymbol{x}')) \neq \boldsymbol{y}$，函数保持遭到破坏。无论激活函数（例如 ReLU）位于何处，这一结论均成立。

对于像 Net2Net 这样的依赖权重的方法，要解决这一难题，唯一的办法就是将每个神经元恰好复制相同的次数，以保持均值和方差。然而，这要求每次至少将宽度翻倍，导致调度固定呈指数增长，严重损害了灵活性。例如，小模型在 $hidden\ dim = 512$下的训练动态可能优于在 384 下的训练动态（图 2），但如果我们的目标模型具有 $hidden\ dim = 768$，Net2Net 就必须从一个$384 = 768/2$ 维模型开始扩展，才能保持函数不变。

其他算子 类似地，将新的输入权重置零的算子（Evci 等人，2022）会产生$\boldsymbol{x}' = (x_1,\ x_2,\ x_3,\ 0,\ 0)$，无法解决这一难题。此外，这些算子本质上要求激活函数 σ 满足 $\sigma(0) = 0$，这限制了它们的使用。据我们所知，在 MSG 之前，还没有适用于任意隐藏维度 扩展的严格函数保持增长算子。

2.3.2 深度维度

对于深度（层数）扩展，现有工作要么堆叠（Gong 等人，2019；Yang 等，2020；Chen 等人，2022；Gu 等，2021），要么插入（Qin 等，2022b；Li 等，2022）从现有层复制权重的新 Transformer 层。这两种方式都不能保持函数不变。Shen 等人（2022）将新层中的所有层归一化权重置零，用于类似 GPT（Radford 等，2019）的 Transformer：

$$\boldsymbol{x}' = \boldsymbol{x} + \text{Attention}(\text{LN}(\boldsymbol{x})), \quad \boldsymbol{y} = \boldsymbol{x}' + \text{FFN}(\text{LN}(\boldsymbol{x}')), \tag{8}$$

实现了严格的函数保持 $\boldsymbol{y} = \boldsymbol{x}$。然而，对于类似 BERT 的实现 s:

$$\boldsymbol{x}' = \text{LN}(\boldsymbol{x} + \text{Attention}((\boldsymbol{x}))), \quad \boldsymbol{y} = \text{LN}(\boldsymbol{x}' + \text{FFN}(\boldsymbol{x}')), \tag{9}$$

this 方法会产生全零输出。因此，适用于任意结构的深度函数保持算子 h gro扩展也尚未得到充分探索。

2.3.3 对初始化的依赖

大多数旨在实现函数保持的现有增长算子，都会为新权重指定固定 的初始化方式。例如，Net2Net 将新权重初始化为现有权重的一部分或副本，而其他方法则直接将其置零。这些方法（尤其是置零方法）的缺点之一是对称性 问题：某些新神经元始终会收到与其他新神经元或现有神经元相似的梯度，这带来了不必要的限制，减缓了模型收敛速度 [2]。此外，Evci 等人，2022 和 Wang 等人，2022 的研究表明，权重初始化甚至可以被视为一个独立的优化问题。然而，依赖初始化的增长算子会错失进一步改善训练动态的机会。因此，尽管大多数研究都认同函数保持的重要性，但似乎有必要回答：“是否值得为了更好的初始化而牺牲函数保持？”借助 MSG，我们将这两个因素解耦，并通过在不同条件下对比有无函数保持的实验来探索这一问题。

3 掩码结构增长

我们提出了一种渐进式训练新框架，即掩码结构增长（MSG）。在第 3.1 节中，我们介绍适用于所有增长维度的 MSG 算子。在第 3.2 节中，我们讨论调度设计的方法和结果。

3.1 MSG 算子

MSG 的核心思想是推导出外部掩码，彻底消除新结构对模型函数的影响。在扩展后立即通过 $mask = 0$ 严格保持函数不变。随后，我们逐步增大掩码，以提升新结构的影响，最终通过 $mask = 1$ 达到目标结构。其基本原理是：（1）扩展后，当前损失与较小模型的损失完全一致，并且通常会在下一步通过梯度下降而降低；（2）当掩码值接近 0 时，现有神经元的输出能够有效地为新权重的更新提供初始指引。

3.1.1 全连接层

全连接（FC）层是特征向量之间的一种线性投影，后接（可选的）激活函数和层归一化。除了层数之外，它涉及所有增长维度。令 $\boldsymbol{x} \in \mathbb{R}^{d_1}$ and

[2]现有的对称性破解方案无法保持函数不变（Gong 等人，2019；Chen 等人，2022） .

$\boldsymbol{y} \in \mathbb{R}^{d_2}$ 分别为全连接层 $\boldsymbol{W}$的输入和输出。如图 1（右）所示，当将 $\boldsymbol{y}$ 的大小从 d_2 扩展到 d_2' 时，我们引入一个掩码向量 $\boldsymbol{c}$，其中有 d_2 个值为 1 的元素和 $d_2' - d_2$ 个值为 0 的元素：

$$\boldsymbol{c} = [\mathbf{1}_{d_2}; \mathbf{0}_{d_2'-d_2}], \tag{10}$$

其中，$[\cdot]$ 表示沿最后一个维度进行拼接。我们首先从$\boldsymbol{W}$ 复制已有权重，并将新权重初始化为 $arbitrary$ 值：

$$\boldsymbol{W}'_{i,*} = \begin{cases} \boldsymbol{W}_{i,*} & 1 \le i \le d_2 \\ \text{any value} & d_2 < i \le d_2' \end{cases}. \tag{11}$$

然后，扩展后的全连接层的新输出$\boldsymbol{y}' \in \mathbb{R}^{d_2'}$ 计算如下：

$$\boldsymbol{y}' = \boldsymbol{c} \odot (\sigma(\boldsymbol{W}' * \boldsymbol{x})) = [\boldsymbol{y}; \mathbf{0}_{d_2'-d_2}], \tag{12}$$

其中，$*$ 表示矩阵（内）积，σ 表示任意激活函数。对于已有神经元，网络产生的值与原来完全相同；对于新神经元，其输出会被掩码置为 0。

另一方面，如果输入 $\boldsymbol{x}$ 扩展为$\boldsymbol{x}' \in \mathbb{R}^{d_1'}$，它原本应为 0 的形式 f:

$$\boldsymbol{x}' = [\boldsymbol{x}; \mathbf{0}_{d_1'-d_1}], \tag{13}$$

因为它是前几层的输出，并且已经通过公式（12）处理。我们按照以下方式初始化新权重：

$$\boldsymbol{W}'_{*,j} = \begin{cases} \boldsymbol{W}_{*,j} & 1 \le j \le d_1 \\ \text{any value} & d_1 < j \le d_1' \end{cases}. \tag{14}$$

因此，网络的输出 $\boldsymbol{y}' \in \mathbb{R}^{d_2}$ 为：

$$\boldsymbol{y}' = \sigma(\boldsymbol{W}' * \boldsymbol{x}') = \sigma(\boldsymbol{W} * \boldsymbol{x}) = \boldsymbol{y}. \tag{15}$$

如果输出维度 d_2 没有扩展，这里就不需要额外掩码。如果 d_1 和 d_2都进行了扩展，我们先应用 (14)，再应用 (11)，最后通过 (12)对输出进行掩码处理。解决层归一化（第 3.1.2 节）后，整个全连接模块的功能将得以保留，如图 1 所示。

3.1.2 层归一化解决方案

我们通过在层归一化内部加入类似的掩码，来解决层归一化困境（第 2.3.1 节）。将特征 $\boldsymbol{x}$ 及其层归一化的维度从 d 扩展到 d'时，我们引入一个外部掩码 $\boldsymbol{c} \in \mathbb{R}^{d'}$：

$$\boldsymbol{c} = [\mathbf{1}_d; \mathbf{0}_{d'-d}], \tag{16}$$

并推导出扩展后特征 $\boldsymbol{x}' \in \mathbb{R}^{d'}$的均值和方差的新公式：

$$\begin{aligned} \mu' &= (\boldsymbol{x}' * \boldsymbol{c})/\text{sum}(\boldsymbol{c}), \\ (\sigma')^2 &= ((\boldsymbol{x}' - \mu')^2 * \boldsymbol{c})/\text{sum}(\boldsymbol{c}), \end{aligned} \tag{17}$$

其中，sum(·) 表示所有元素的总和，$*$ 表示内积。掩码值为零时，新神经元不会对均值和方差产生贡献。因此，如果保留层归一化权重 $\boldsymbol{\omega}$ 和偏置 $\boldsymbol{\beta}$的前 d 个值，并将 $d'-d$ 个新层归一化权重初始化为任意值，则有：

$$\text{LN}^*(\boldsymbol{x}') = [\text{LN}(\boldsymbol{x}); \xi_{d'-d}], \tag{18}$$

其中，$\xi_{d'-d}$ 的值由连接到 $\boldsymbol{x}'$中新神经元的输入权重决定。最后，我们再次将掩码 $\boldsymbol{c}$乘到结果上：

$$\boldsymbol{c} \odot \text{LN}^*(\boldsymbol{x}') = [\text{LN}(\boldsymbol{x}); \mathbf{0}_{d'-d}]. \tag{19}$$

Th因此，对于任意的 d 和 d'，MSG 在 LN 下严格保持函数不变。随着训练的进行 ，we 逐渐将 $\boldsymbol{c}$ 中最后 $d' - d$ 个值增大至 1。

3.1.3 自注意力头的扩展

自注意力模块（Vaswani 等，2017）包含多个注意力头。通过将每个注意力头视为一个“神经元”，并采用类似公式 10 至 15 的过程进行调整，MSG 可以实现严格的函数保持。由于篇幅有限，我们将证明留在附录 A 中。

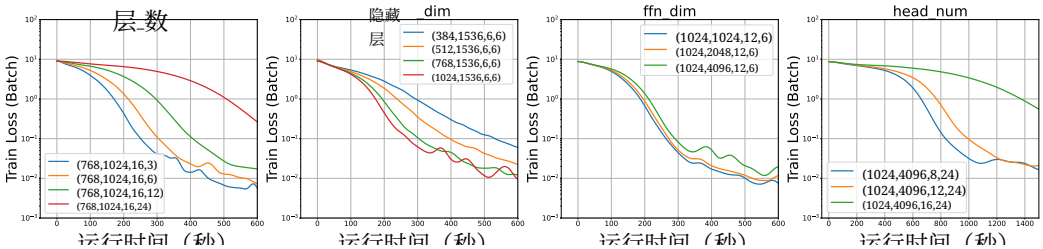


图 2：采用不同结构超参数时的训练损失曲线。我们研究了各增长维度对模型早期“预训练速率”的影响 γ。

3.1.4 深度增长

我们提出了一种受残差连接（He 等人，2016）启发的深度增长算子。设 Layer$_n$ 为新的 Transformer 层，$\boldsymbol{x}_{n-1}$ 为其输入，$\boldsymbol{c}$ 为与 $\boldsymbol{x}_{n-1}$ 大小相同的全零掩码，则该层的输出计算如下：

$$\boldsymbol{x}_n = \boldsymbol{c} \odot \text{Layer}_n(\boldsymbol{x}_{n-1}) + (\boldsymbol{1} - \boldsymbol{c}) \odot \boldsymbol{x}_{n-1}. \tag{20}$$

当掩码值为零时，整个 Layer$_n$ 会被跳过，以保持函数不变；随着掩码值增至 1，残差连接消失，形成与原始 Transformer 相同的结构。

3.2 扩展计划

在本小节中，我们研究训练计划的构建。尽管多阶段扩展天然比单阶段扩展更加灵活，但若不借助事后信息，就很难找到最优的扩展计划[3]。现有工作直接给出最终训练计划，却没有详细说明[3]（Gong 等人（2019）；Gu 等人（2021））。与之相反，我们提出一种网格搜索方法，该方法有望推广至其他 Transformer 变体。我们并未声称能够彻底解决这个问题，而是提供实证见解，以启发后续研究。

3.2.1 通过网格搜索构建调度

理想的扩展计划应包含一系列结构，使其在各自阶段都能快速学习。我们发现，“预训练速率” γ 是衡量早期/中期阶段学习速度的合理指标 [4]：

$$\gamma \triangleq \frac{\Delta \text{loss}}{\Delta \text{t}} = \frac{\Delta \text{loss}}{\Delta \text{step}} \cdot \frac{\Delta \text{step}}{\Delta \text{t}} \triangleq \alpha \cdot \tau. \tag{21}$$

中间结构。 扩展调度是一种权衡：小模型的常数 τ总是更优，但其 α 在训练过程中会迅速下降，因此必须适时扩展。鉴于难以从数学上量化 α，我们选择仅使用 10k 条数据进行网格搜索，研究 4 个增长维度分别如何影响 γ。我们考察了在不同维度上有所变化的候选中间结构，并比较其训练曲线。其他超参数与最终训练计划中的设置保持一致。使用 8 个 GPU，模型通常只需几分钟即可收敛，耗时很少。如图 2 所示，我们总结出适用于 BERT 的以下启发式规则：**(i)** 较大的层数 会严重损害 τ，而 α提升有限。最好在后期阶段再增加层数。**(ii)** 对于浅层模型，较大的隐藏维度 能显著改善 γ。隐藏维度初始值应设得较大。**(iii)** 前馈网络维度 本身可平衡 α 和 τ，对 γ影响很小；而较小的头数 则能改善深层模型的 γ。我们根据这些启发式规则构建了不同的中间结构序列。同样的方法也适用于 GPT。

阶段时长。 与常规预训练一样，总训练步数是预先确定的。给定总步数和一个中间结构序列，我们将每个阶段的训练过程划分为大致相等的步数，同时略微侧重于早期阶段。我们发现，尽管从理论上讲这种简单方法并非最优，但在实践中效率很高。一种最优策略可能是在当前 γ 低于相同运行时间下完整目标结构的水平时，扩展到下一个结构；但这显然需要事后信息，因此并不实用。

[3]Shen 等人（2022）在不同的设定下研究了这一主题，更多讨论请参见附录 B。[4]请注意，当损失曲线几乎趋于平稳时，γ不再适用。例如，在训练后期，我们发现 BERT 的训练/验证损失与下游性能之间不存在稳定的相关性。

表2：在每个阶段扩展一个维度的增长计划。Sch1-B 和 Sch1-L 分别用于 Bert-base 和 Bert-large，Sch1-G 用于 GPT-2。Rapid-L 是一种快速训练计划，可在 100k 步内进行多次扩展。

调度	0~200k	200k~400k	400k~500k	500k~600k	600k~700k	700k~900k
Sch1-B	(512,768,8,3)	(512,3072,8,3)	(512,3072,8,6)	(768,3072,8,6)	(768,3072,12,6)	(768,3072,12,12)
Sch1-L	(768,1024,12,6)	(768,4096,12,6)	(768,4096,12,12)	(1024,4096,12,12)	(1024,4096,16,12)	(1024,4096,16,24)

调度	0~50k	50k~100k	100k~130k	130k~160k	160k~190k	190k~结束
Sch1-G	(512,768,8,3)	(512,3072,8,3)	(512,3072,8,6)	(768,3072,8,6)	(768,3072,12,6)	(768,3072,12,12)

调度	0~400k	400k~420k	420k~440k	440k~500k	500k~800k
Rapid-L	(512,1024,8,6)	(512,4096,8,6)	(1024,4096,8,6)	(1024,4096,16,6)	(1024,4096,16,24)

表3：掩码强度实验 ϕ。

ϕ	0	100	250	500	800	1000
10k 步时的损失	3.36	3.34	3.25	3.23	3.23	3.23

构建的训练计划。 遵循上述相同的方法，我们为 BERT 和 GPT 构建了训练计划，详见表 2。Sch1-B、Sch1-L 和 Sch1-G 分别用于我们的主要实验中的 Bert-base、Bert-large 和 GPT-2。这些训练计划采用多阶段方式，每个阶段扩展一个维度；与使用相同步数从头训练相比，速度分别提升了 1.8 倍、2.2 倍和 1.4 倍。这明显快于 Gong 等人，2019；Gu 等，2021 在相同目标模型上的方法。Rapid-L 是一种快速训练计划，在 10 万步内安排了若干连续的增长阶段，以研究阶段时长的影响。消融实验所用的其他训练计划见附录 G。

掩码强度。 我们将 ϕ 定义为将掩码值从 0 线性增加到 1 所需的步数。最初，基于直觉，我们在所有实验中将 $\phi = 5000$ 设为该值，效果良好。为进一步研究其影响，我们在仅包含 100k 条数据的 Bert-large 上测试了不同的 ϕ 值。我们训练了超过 10 个 epoch，以模拟完整数据训练的中期和后期阶段。这里使用的调度是 Rapid-L 的压缩版本（表2）。我们比较了 10k 步时的训练损失（表3），观察到所有超过 500 的 ϕ 值都能确保后续训练的稳定性，并取得相近的结果。

3.2.2 初始化

MSG 支持对新权重进行任意初始化。我们发现，对于宽度扩展，从正态分布 $\mathcal{N}(0, 0.02)$ 中采样，并对深度采用堆叠 策略（结合公式 20），其效果已经优于当前最先进的方法。鉴于 $\mathcal{N}(0, 0.02)$ 是从头初始化 Transformer 时的常用选择，我们认为它已经经过社区驱动的不断改进和优化。我们将其他“学习如何初始化”的方法留待未来研究。对于优化器，我们复制现有权重所存储的 AdamW（Loshchilov 和 Hutter，2018年）移动平均值，而新权重则从头开始。

4 实验

4.1 实验设置

我们使用经过预处理的英文维基百科和 Book Corpus 组合数据（Zhu 等人，2015）训练 Bert-base 和 Bert-large（Devlin 等，2019），并使用 OpenWebText（Gokaslan 和 Cohen，2019）训练 GPT-2（Radford 等，2019）。增长计划见表2。其他细节请参阅附录D。评估方面，我们在 GLUE（Wang 等人，2018）和 SQuAD v1.1（Rajpurkar 等人，2018）任务上对 Bert 模型进行微调，并在 Wikitext2（Merity 等人，2016）上评估 GPT 模型。我们报告开发集上运行 3 次所得指标的均值和标准差。微调设置详情见附录E。

4.2 对比方法

作为表 1 的补充，为了进行更公平的比较，我们回答以下两个问题 ：

我们应该采用哪种训练计划进行比较？ 如果增长算子能够确保性能不下降，那么加速比就完全取决于训练计划。由于 MSG 在我们自己的训练计划上表现出色，而这些训练计划对于相同的目标模型而言比此前最先进的训练计划更快（第3.2.1节），因此我们在自己的训练计划上比较不同的算子。

表 4：Bert-Large 上的主要结果和消融研究。我们报告 GLUE 平均分数以及 SQuAD 的准确率/F1。数值为 3 次运行的均值（标准差）。

目标	Bert-大型					
调度	Full	Sch1-L			Rapid-L	
运行时间	70h, 48分钟	**32h, 10 分钟**			29h, 20分钟	
算子	-	Bert2BERT	MSG	无掩码	MSG	无掩码
GLUE 平均值	82.2(0.2)	82.1(0.3)	**83.2**(0.2)	82.2(0.2)	**79.1(0.4)**	77.9(0.3)
SQuAD	80.2(0.2)/87.9(0.3)	80.0(0.1)/87.6(0.1)	**81.3(0.2)/88.4(0.1)**	79.7(0.3)/88.0(0.2)	**76.5(0.6)/84.4(0.5)**	75.7(0.4)/84.0(0.3)

表 5：Bert-Base 和 GPT-2 的主要结果与消融研究。对于 GPT-2，我们在 Wikitext2 的验证集上进行评估（Merity 等人，2016年）。“PPL-zs”表示零样本困惑度（目前的标准做法），“PPL-ft”则是使用 Wikitext2 训练数据进行微调的结果。

目标	Bert-Base			目标	GPT-2		
调度	Full	Sch1-B		调度	Full	Sch1-G	
运行时间	26h, 10分钟	**14h, 36分钟**		运行时间	53h, 1min	**37h, 59分钟**	
算子	-	Bert2BERT	MSG	算子	-	Bert2BERT	MSG
GLUE 平均值	80.7(0.2)	80.5(0.2)	**81.0(0.2)**	PPL-zs	41.31	42.24	**41.20**
SQuAD	79.1(0.2)/86.9(0.2)	79.0(0.1)/86.7(0.0)	**79.6(0.5)/87.2(0.4)**	PPL-ft	**23.90**	25.13	24.14

我们应该比较哪些算子？主流增长算子包括置零和 Net2Net。由于置零限制了激活函数的使用（第 2.3.1 节），有时还会导致训练出现问题（Gong 等人，2019），因此大多数相关工作（Gong 等人，2019；Gu 等，2021；Chen 等人，2022；Qin 等，2022b）会将 Net2Net 应用于宽度扩展，并将某种“直接复制”策略应用于深度扩展（表 1）。因此，在主要实验中，我们将 MSG 与自己复现的最新版本 *Bert2BERT*（Qin 等，2022b）算子进行比较，该算子取得了最先进的性能。它在宽度扩展方面采用了与（Chen 等人，2022）相同的 Net2Net 算子，并在深度扩展方面采用了插值策略。其他配置均与 MSG 保持一致。由于篇幅有限，我们在附录 H 中将其与数据驱动的 *LiGO* 方法（Wang 等人，2022）进行比较[5]。

4.3 主要结果

We pBert-Large 的结果见表 4。Bert-base 和 GPT-2 的结果见表 5。各项结果在每个
在任务上的详细结果因篇幅有限，留待附录 F。我们有以下观察：

MSG 训练计划实现了更高的加速比。与不扩展基线模型（Full-$*$）相比，MSG 在 Bert-base 上以 1.8 倍加速达到相当的下游性能，并在 Bert-large 上以 2.2 倍加速实现显著更高的性能。与此前最佳的多阶段策略（Gu 等，2021）相比，该策略在 Bert-base 上的加速比为 1.74 倍，在 Bert-large 上为 1.82 倍。我们的优势在于 MSG 支持沿 4 个可能的维度进行扩展，从而能够灵活设计调度，而 *CompoundGrow*（Gu 等，2021）只探索了其中 2 个维度。GPT-2 上的情况也相同：加速比为 1.4 倍，性能与基线模型相当[6]。

MSG 算子比 *Bert2BERT* 具有更大的潜力。在相同训练计划下与 Bert2BERT 进行比较时，MSG 在 Bert-large 和 GPT-2 上实现了显著更好的性能。这是因为：（1）MSG 严格保持函数不变，节省了恢复函数所需的时间；（2）MSG 支持随机初始化，从而自然地解决了对称性问题，并带来更好的动态特性。

4.4 消融实验

我们通过消融实验尝试回答以下问题：

调度设计中涉及的因素。比较 Rapid-L 和 Sch1-L 的结果（表 4）可以发现，尽管两者的时间成本相近，但性能却存在显著差距。这表明，快速扩展（在短时间内多次扩展模型）不利于 Transformer。这也验证了我们采用的直观策略的有效性，即将阶段时长平均分配（第3.2.1节）。另一个影响因素可能是中间结构的选择。为了研究这一点，我们进一步设计了 Sch0（附录 G 和表 8）作为一种替代调度方案，并遵循与……相同的启发式规则。

[5]Gradmax（Evci 等人，2022）算子尚未被证明可以直接应用于 Transformer。[6]对于 GPT-2，*Bert2BERT* 从一个大型检查点开始，该检查点具有 512 的隐藏维度和 12 层，但未将其训练成本纳入考量；*LiGO* 报告了 1.22 倍的加速比，但其起始检查点相对较小。我们观察到，大多数方法在 GPT 上的加速比低于在 BERT 上的加速比，并推测这是因为对于较浅的模型（3 层等），因果语言模型的损失下降相对缓慢。

Sch1，但采用了不同的中间结构。采用 Sch0 得到的 GLUE 结果如表 9 所示。其得分与 Sch1 接近（Bert-Large 上为 82.8 vs. 83.2，Bert-base 上为 80.9 vs. 81.0），这支持了我们的调度构建方法的稳健性。

函数保持的重要性。为了研究函数保持的影响，我们在扩展后立即移除所有掩码（表 4 中的无掩码），从而采用一种不进行函数保持、随机初始化新权重的策略。尽管 Chen 等人（2022）表明，类似策略在单阶段设置中效果较差，但我们观察到，该策略在 Sch1-L 上表现尚可，与完整基线和 Bert2BERT 的表现相当（表 4）。这表明，随机初始化确实能够比依赖权重的 Bert2BERT 带来更好的训练动态。另一方面，在采用相同初始化策略时，无掩码在 Sch1-L 和 Rapid-L 上的表现都不如 MSG，这支持了以下观点：只要函数保持不会损害初始化，它就是有益的。此外，通过比较 Rapid-L 上无掩码与 MSG 的训练损失曲线（图 3），我们观察到，如果继续训练，MSG 会比无掩码更加稳定。这表明，对于快速结构变化且时间预算有限的情况，函数保持能够提高训练稳定性。

5 相关工作

5.1 加速预训练

渐进式训练（Wen 等，2020；Dong 等，2020）因能够加速计算机视觉（Li 等，2022）和自然语言处理（Gong 等人，2019；Gu 等，2021）的预训练而受到关注。Gong 等人（2019）提出了一种堆叠方法，在每个阶段将模型深度加倍。*CompoundGrow*（Gu 等，2021）在堆叠的基础上，将前馈网络（FFN）扩展纳入调度。Shen 等人（2022）提出了一种分阶段方法，进一步支持扩展特征的隐藏层维度，但仅限于整数倍扩展，因此特征维度、FFN 维度和注意力头数量都受到限制。*Bert2BERT*（Chen 等人，2022；Qin 等，2022b）和*LiGO*（Wang 等人，2022）支持所有可能的增长维度，但侧重于所有维度同时扩展的单阶段增长。然而，每个增长维度对训练动态的影响各不相同（第 3.2 节），这一点尚未得到充分探索。与此不同，我们研究每次仅扩展一个维度的多阶段增长，从而获得更高的灵活性和加速效果。详细总结见表 1。其他加速预训练的工作包括部分更新（Yang 等，2020）、权重共享（Lan 等，2019）、对抗训练（Clark 等，2019）、计算最优缩放定律（Hoffmann 等，2022）和专家混合模型（MoEs）（Fedus 等，2022），这些研究与我们的主题正交。Bartoldson 等，2023 对这些方法进行了系统综述。

5.2 函数保持扩展

函数保持扩展确保扩展后的模型对任意输入的输出都与扩展前的模型一致，这直观上有利于知识继承。通常通过数学方法来实现，以避免耗时的数据驱动蒸馏（Jiao 等人，2020；Qin 等人，2022a）。Chen 等人（2016）提出了 Net2Net 方法，通过拆分现有权重来构造新神经元，从而在宽度扩展时保持函数不变。其他方法（Evci 等人，2022；Wei 等人，2016）则通过将部分权重设为零来保持函数不变。在应用于 Transformer 结构时，现有算子在大多数情况下只能实现非严格的函数保持，限制了其效果（第 2.3 节）。此外，这些方法完全依赖于新权重的初始化。相比之下，我们的方法在所有增长维度上都能严格保持函数不变，且不依赖新权重的初始化。*LiGO*（Wang 等人，2022）设计了一种高效的数据驱动方法，用于映射小模型与大模型之间的权重，通过学习初始化来改善训练动态。由于 MSG 具有支持任意初始化的独特优势，因此可以与类似*LiGO*的方法结合，应用于新权重。我们将这一有趣的话题留待未来研究。

6 结论

我们提出了一种新颖的框架：掩码结构增长（MSG），用于语言模型的渐进式预训练。MSG 支持在所有可能的维度上执行扩展操作，从而实现高效的训练计划。实验结果表明，在下游任务性能持平或有所提升的情况下，MSG 在 Bert-base、Bert-large 和 GPT-2 上均实现了最先进的加速效果。我们还证明，MSG 严格满足函数保持特性，且不依赖于新的权重初始化。这些特性使 MSG 优于其他算子，并为未来的研究带来益处。

局限性

在这项工作中，我们提出了每次仅涉及一个维度的增长计划的启发式规则。然而，仍有尚未探索的空间，因为可以在同一阶段组合多个增长维度的子集。我们认为，基于训练动态主动搜索下一个目标结构的<escape></escape>自适应增长计划仍有很大的探索空间。我们工作的另一个局限在于，Transformer 扩展的最佳初始化策略仍然是一个悬而未决的问题。尽管我们已经证明，随机初始化可能优于 Net2Net，但更深入的研究（即将 Evci 等人，2022 的方法推广至 Transformer）仍有待未来工作进一步开展。

致谢

本研究获得国家科技重大专项（2022ZD0116300）和国家自然科学基金（编号：62106249）的资助。

参考文献

Jimmy Lei Ba、Jamie Ryan Kiros 和 Geoffrey E Hinton。层归一化。*arXiv* 预印本 *arXiv:1607.06450*，2016年。

Brian RBartoldson、Bhavya Kailkhura 和 Davis Blalock。高效计算的深度学习：算法趋势与机遇。机器学习研究期刊，24:1–77，2023年。

Tom Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell 等。语言模型是小样本学习器。神经信息处理系统进展，33:1877–1901，2020年。

Cheng Chen、Yichun Yin、Lifeng Shang、Xin Jiang、Yujia Qin、Fengyu Wang、Zhi Wang、Xiao Chen、Zhiyuan Liu 和 Qun Liu。Bert2BERT：迈向可复用的预训练语言模型。载于计算语言学协会第*60*届年会论文集（第*1*卷：长篇论文），第2134–2148页，2022。

Tianqi Chen、Ian J. Goodfellow 和 Jonathon Shlens。Net2Net：通过知识迁移加速学习。载于 Yoshua Bengio 和 Yann LeCun（编），第*4*届国际学习表征会议（*ICLR 2016*），*2016*年*5*月*2*日至*4*日，波多黎各圣胡安，会议论文集，2016年。网址：http://arxiv.org/abs/1511.05641。

XiaohanChen、Yu Cheng、Shuohang Wang、ZheGan、Zhangyang Wang 和JingjingLiu。Earlybert：通过早期鸟彩票券实现高效的 BERT 训练。载于 第 *59* 届计算语言学协会年会暨第 *11* 届自然语言处理国际联合会议论文集（第 *1* 卷：长篇论文），第 2195–2207 页，2021。

Kevin Clark、Minh-Thang Luong、Quoc V Le 和 Christopher D Manning。Electra：将文本编码器预训练为判别器，而非生成器。载于国际学习表征会议，2019 年。

Wei Deng、James B Aimone 和 Fred H Gage。新神经元与新记忆：成年海马神经发生如何影响学习和记忆？《自然评论：神经科学》，11(5)：339–350，2010 年。

Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。BERT：用于语言理解的深度双向变换器预训练。载于《*2019*年北美计算语言学协会分会年会论文集：人类语言技术》，第*1*卷（长篇与短篇论文），第4171–4186页，2019年。

Chengyu Dong、Liyuan Liu、Zichao Li 和 Jingbo Shang。迈向自适应残差网络训练：神经常微分方程视角。载于国际机器学习会议，第 2616–2626 页。PMLR，2020年。

Peter S Eriksson、EkaterinaPerfilieva、ThomasBj¨ork-Eriksson、Ann-MarieAlborn、Claes Nordborg、Daniel A Peterson 和 Fred H Gage。《成年人的海马体中的神经发生》。《自然医学》，4(11):1313–1317，1998 年。

Utku Evci、Bart van Merrienboer、Thomas Unterthiner、Fabian Pedregosa 和 Max Vladymyrov。Gradmax：利用梯度信息扩展神经网络。发表于国际学习表征会议，2022。

William Fedus、Barret Zoph 和 Noam Shazeer。Switch Transformer：通过简单高效的稀疏性扩展至万亿参数模型。机器学习研究期刊，23(1)：5232–5270，2022。

Aaron Gokaslan 和 Vanya Cohen。OpenWebText 语料库。2019 年。

Linyuan Gong、Di He、Zhuohan Li、Tao Qin、Liwei Wang 和 Tieyan Liu。通过渐进式堆叠实现 BERT 的高效训练。发表于国际机器学习会议，第 2337–2346 页。PMLR，2019 年。

Xiaotao Gu、Liyuan Liu、Hongkun Yu、Jing Li、Chen Chen 和 Jiawei Han。《关于渐进式 BERT 训练中的 Transformer 扩展》。载于*2021* 年计算语言学协会北美分会会议论文集：人类语言技术，第 5174–5180 页，2021 年。

KaimingHe、Xiangyu Zhang、ShaoqingRen 和 JianSun。《用于图像识别的深度残差学习》。载于*IEEE* 计算机视觉与模式识别会议论文集，第 770–778 页，2016年。

Jordan Hoffmann、Sebastian Borgeaud、Arthur Mensch、Elena Buchatskaya、Trevor Cai、Eliza Rutherford、Diego de Las Casas、Lisa Anne Hendricks、Johannes Welbl、Aidan Clark、Tom Hennigan、Eric Noland、Katherine Millican、George van den Driessche、Bogdan Damoc、Au- relia Guy、Simon Osindero、Karen Simonyan、Erich Elsen、Oriol Vinyals、Jack W. Rae 和 Laurent Sifre。《计算最优大型语言模型训练的实证分析》。载于*NeurIPS*，2022。网址：http://papers.nips.cc/paper_files/paper/2022/hash/ c1e2faff6f588870935f114ebe04a3e5-Abstract-Conference.html。

Xiaoqi Jiao、Yichun Yin、Lifeng Shang、Xin Jiang、Xiao Chen、Linlin Li、Fang Wang 和 Qun Liu。Tinybert：为自然语言理解蒸馏 BERT。载于计算语言学协会会议论文集：*EMNLP 2020*年，第 4163–4174 页，2020年。

Zhenzhong Lan、Mingda Chen、Sebastian Goodman、Kevin Gimpel、Piyush Sharma 和 Radu Sori-cut。《Albert：一种用于语言表征自监督学习的轻量级 BERT》。发表于国际学习表征会议，2019年。

Changlin Li、Bohan Zhuang、Guangrun Wang、Xiaodan Liang、Xiaojun Chang 和 Yi Yang。用于视觉Transformer高效训练的自动渐进式学习。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 12486–12496 页，2022。

Xiang Li、Yiqun Yao、Xin Jiang、Xuezhi Fang、Xuying Meng、Siqi Fan、Peng Han、Jing Li、Li Du、Bowen Qin 等人。FLM-101B：一种开放式大语言模型，以及如何以 10 万美元的预算训练它。*arXiv* 预印本 *arXiv:2309.03852*，2023年。

Zhuohan Li、Eric Wallace、Sheng Shen、Kevin Lin、Kurt Keutzer、Dan Klein 和 Joey Gonzalez。先训练大型模型，再进行压缩：重新思考 Transformer 模型的规模，以实现高效训练和推理。载于国际机器学习会议，第 5958–5968 页。PMLR，2020年。

Ilya Loshchilov 和 Frank Hutter。解耦权重衰减正则化。载于国际学习表示会议，2018年。

Stephen Merity、Caiming Xiong、James Bradbury 和 Richard Socher。指针哨兵混合模型。2016年。

OpenAI。GPT-4 技术报告。*arXiv* 预印本 *arXiv:2303.08774*，2023年。

Yujia Qin、Yankai Lin、Jing Yi、Jiajie Zhang、Xu Han、Zhengyan Zhang、Yusheng Su、Zhiyuan Liu、Peng Li、Maosong Sun 和 Jie Zhou。预训练语言模型的知识继承。载于*2022*年计算语言学协会北美分会会议论文集：人类语言技术，第 3921–3937 页。计算语言学协会，2022a。

Yujia Qin、Jiajie Zhang、Yankai Lin、Zhiyuan Liu、Peng Li、Maosong Sun 和 Jie Zhou。Elle：面向新兴数据的高效终身预训练。载于计算语言学协会 *ACL 2022*年会议论文集：研究成果，第2789–2810页，2022b。Alec Radford、Jeffrey Wu、Rewon Child、David Luan、Dario Amodei、Ilya Sutskever 等。语言模型是无监督的多任务学习者。*OpenAI* 博客，1(8):9，2019。Pranav Rajpurkar、Robin Jia 和 Percy Liang。了解自己不知道什么：SQuAD 中的不可回答问题。载于计算语言学协会第*56*届年会论文集（第*2*卷：短论文），第784–789页，2018。Roy Schwartz、Jesse Dodge、Noah A Smith 和 Oren Etzioni。绿色人工智能。美国计算机协会通讯，63(12):54–63，2020。Sheng Shen、Pete Walsh、Kurt Keutzer、Jesse Dodge、Matthew Peters 和 Iz Beltagy。Transformer 语言模型的分阶段训练。载于国际机器学习会议。PMLR，2022。Henriette Van Praag、Alejandro F Schinder、Brian R Christie、Nicolas Toni、Theo D Palmer 和 Fred H Gage。成年海马体中的功能性神经发生。自然，415(6875):1030–1034，2002。Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。注意力机制即你所需的一切。神经信息处理系统进展，30，2017。Alex Wang、Amanpreet Singh、Julian Michael、Felix Hill、Omer Levy 和 Samuel R Bowman。GLUE：自然语言理解的多任务基准与分析平台。*arXiv* 预印本 *arXiv:1804.07461*，2018。Peihao Wang、Rameswar Panda、Lucas Torroba Hennigen、Philip Greengard、Leonid Karlinsky、Rogerio Feris、David Daniel Cox、Zhangyang Wang 和 Yoon Kim。学习扩展预训练模型，以实现高效的 Transformer 训练。载于第十一届国际学习表征会议，2022。Ziheng Wang、Jeremy Wohlwend 和 Tao Lei。大语言模型的结构化剪枝。载于*2020*年自然语言处理经验方法会议（*EMNLP*）论文集，第6151–6162页，2020。Tao Wei、Changhu Wang、Yong Rui 和 Chang Wen Chen。网络形态变换。载于国际机器学习会议，第564–572页。PMLR，2016年。Wei Wen、Feng Yan、Yiran Chen 和 Hai Li。AutoGrow：深度卷积网络中的自动层扩展。载于第*26*届 *ACM SIGKDD* 知识发现与数据挖掘国际会议论文集，第833–841页，2020年。Mengzhou Xia、Zexuan Zhong 和 Danqi Chen。结构化剪枝可学习紧凑且准确的模型。载于计算语言学协会第*60*届年会论文集（第*1*卷：长论文），第1513–1528页，2022。Cheng Yang、Shengnan Wang、Chao Yang、Yuechuan Li、Ru He 和 Jingqiao Zhang。Progressively Stacking 2.0：一种用于加速 BERT 训练的多阶段逐层训练方法。*arXiv* 预印本 *arXiv:2011.13635*，2020年。Yukun Zhu、Ryan Kiros、Rich Zemel、Ruslan Salakhutdinov、Raquel Urtasun、Antonio Torralba 和 Sanja Fidler。让书籍与电影相互映照：通过观看电影和阅读书籍生成类故事的视觉解释。载于*IEEE* 计算机视觉国际会议论文集，第19–27页，2015年。

注意力头上的 MSG 函数保持性证明

在以往的工作中，每个注意力头中查询、键和值向量的大小 d 由隐藏维度/ 头数决定。在这项工作中，我们通过固定 $d = 64$ 并添加新的注意力头来扩展自注意力模块，从而打破这一绑定。权重矩阵 $\boldsymbol{Q}$，$\boldsymbol{K}$，$\boldsymbol{V}$的大小均为（隐藏维度， d）。从 n_1 个注意力头扩展到 n_2 个注意力头时，我们保留现有注意力头的权重，并任意初始化新注意力头。对于注意力头 i:

$$\boldsymbol{Q}'_i, \boldsymbol{K}'_i, \boldsymbol{V}'_i = \begin{cases} \boldsymbol{Q}_i, \boldsymbol{K}_i, \boldsymbol{V}_i & i \leq n_1 \\ \text{any value} & i \in (n_1, n_2] \end{cases}. \tag{22}$$

对于所有新增注意力头 $n_1 < i \leq n_2$，我们将全零掩码 $\boldsymbol{c}$ 乘以其值向量:

$$\boldsymbol{v}'_i = \boldsymbol{c} \odot (\boldsymbol{V}'_i * \boldsymbol{x}) = \boldsymbol{0}, n_1 < i \leq n_2, \tag{23}$$

并得到 $n_1 < i \leq n_2$的全零注意力输出:

$$\boldsymbol{o}'_i = \text{softmax}\left(\boldsymbol{q}'_i * (\boldsymbol{k}'_i)^T / \sqrt{d}\right) * \boldsymbol{v}'_i = \boldsymbol{0}, \tag{24}$$

其中 $\boldsymbol{q}'_i$ 和 $\boldsymbol{k}'_i$ 包含输入序列中的所有查询和键。

对于收集注意力头输出的全连接层 $\boldsymbol{W}_o \in \mathcal{R}^{\text{hidden 维度} \times (d \cdot n)}$，我们将其初始化为:

$$(\boldsymbol{W}_o)'_{*,j} = \begin{cases} (\boldsymbol{W}_o)_{*,j} & j \leq d \cdot n_1 \\ \text{any value} & j \in (d \cdot n_1, d \cdot n_2] \end{cases}. \tag{25}$$

最后，对于任意 n_1 和 n_2，自注意力输出都是函数保持的:

$$\boldsymbol{y}' = \boldsymbol{W}'_o * [\boldsymbol{o}'_1; ...; \boldsymbol{o}'_{n_2}] = \boldsymbol{W}_o * [\boldsymbol{o}_1; ...; \boldsymbol{o}_{n_1}] = \boldsymbol{y}. \tag{26}$$

B 调度中的不同研究问题

关于 分阶段训练的研究（Shen 等人，2022）对如何自动确定扩展时间（步数）的问题进行了深入分析。他们进行了与我们第3.2.1节类似的实验，以寻找与训练曲线相关、可辅助做出这一决策的常数，但仍然依赖事后分析。

主要区别在于，他们实际上回答的是“给定结构序列，应该在哪一步扩展？”，而我们在第3.2.1节的工作侧重于回答“如何设计良好的结构序列，应该优先扩展哪个维度？”。

在他们的工作中，他们通过手动设计训练计划来解决我们提出的问题，这也是大多数现有工作的做法；在我们的工作中，我们通过为每个阶段分配大致相同的步数来解决他们提出的问题。我们认为，这两个问题都是未来研究的重要课题。

C BERT2BERT 简介

Bert2BERT（Chen 等人，2022）是最典型的相关工作，它结合了用于宽度增长的 Net2Net 和用于深度增长的权重复制策略。具体而言，隐藏维度和前馈网络维度的扩展是对全连接层应用标准 Net2Net 方法的实例（公式 5 和 6）。对于头数，他们还将每个注意力头视为一个类神经元单元。注意力头的 Net2Net 映射函数如下:

$$m(i) = \begin{cases} i & i \leq a^s \\ \text{map}(\{1, ..., n\}, i) & a^s < i \leq a^t \end{cases}, \tag{27}$$

其中， a^s 和 a^t 分别是源模型和目标模型的注意力头数量。注意力头 m（i）中的参数直接从注意力头 i 复制而来。在深度维度上，他们遵循 Gong 等人，2019 的做法，将整个模型（嵌入层除外）反复堆叠在自身之上，直到达到目标 层数 。在后续的实现中（Qin 等，2022b），则改用插值策略，逐层复制，并将副本置于原始层之上。在 Bert2BERT 中，主要预训练阶段之前引入了一个预热阶段，在此阶段中仅调整顶层。请注意，由于 MSG 采用多阶段方式，因此没有这个预热阶段。

表 6：Bert-large 的 GLUE 任务结果。

调度	算子	GLUE 平均值	CoLA	SST-2	MRPC	STS-B
Full	-	82.2(0.2)	57.9(1.3)	**91.9(0.1)**	85.8(0.5)/89.7(0.2)	89.0(0.1)/88.7(0.1)
Sch1-L	Net2Net	82.1(0.3)	57.6(1.3)	91.9(0.2)	85.1(0.8)/89.6(0.4)	89.5(0.2)/89.0(0.2)
Sch1-L	MSG	**83.2(0.2)**	**61.9(0.4)**	91.6(0.3)	**87.3(0.7)/91.0(0.4)**	**89.7(0.0)/89.2(0.0)**
Sch1-L	无掩码	82.2(0.2)	56.8(1.0)	91.0(0.4)	86.2(0.6)/90.4(0.3)	89.1(0.1)/88.6(0.2)
Rapid-L	MSG	**79.1(0.4)**	**48.0(1.8)**	**90.7(0.4)**	**81.3(0.6)/87.3(0.3)**	**87.4(0.2)/87.0(0.1)**
Rapid-L	无掩码	77.9(0.3)	45.0(2.0)	88.6(0.2)	77.3(1.0)/85.0(0.6)	86.7(0.1)/86.3(0.1)

调度	算子	QQP	MNLI(m/mm)	QNLI	RTE	SQuAD v1.1
Full	-	90.7(0.3)/87.7(0.4)	83.3(0.5)/83.6(0.3)	90.4(0.2)	68.1(0.7)	80.2(0.2)/87.9(0.3)
Sch1-L	Net2Net	90.8(0.2)/87.8(0.3)	82.9(0.3)/83.2(0.2)	90.4(0.2)	68.0(1.1)	80.0(0.1)/87.6(0.1)
Sch1-L	MSG	**90.9(0.1)/87.9(0.2)**	**83.7(0.5)/84.5(0.3)**	91.2(0.3)	**69.0(1.6)**	**81.3(0.2)/88.4(0.1)**
Sch1-L	无掩码	90.8(0.2)/87.7(0.2)	83.7(0.3)/84.0(0.4)	**91.3(0.3)**	68.5(1.1)	79.7(0.3)/88.0(0.2)
Rapid-L	MSG	89.8(0.4)/86.2(0.6)	**80.1(0.1)/80.8(0.2)**	**88.2(0.2)**	66.1(0.7)	**76.5(0.6)/84.4(0.5)**
Rapid-L	无掩码	**89.8(0.1)/86.4(0.2)**	79.7(0.2)/79.7(0.2)	87.3(0.1)	**66.7(0.3)**	75.7(0.4)/84.0(0.3)

D 预训练详情

BERT Bert-base 拥有 1.1 亿个参数，而 Bert-large 拥有 3.36 亿个参数。对于不采用扩展的基线模型（Full-*），我们训练 100 万步，因为在 90 万步至 100 万步之间，下游性能仍持续提升。对于（Bert-base，Bert-large），我们分别采用（1e-4，1e-5）的学习率和（1 万步，3 万步）的预热步数，并且两者均使用线性学习率调度器。在第 1 次和第 2 次增长阶段，我们按照 Gu 等，2021 的方法，将学习率重置为其最大值。批大小设置为 256，最大序列长度设置为 128。对于 Bert-large，我们将梯度范数裁剪至 1.0。所有实验均在一个配备 8 块 Nvidia A100 GPU 的计算节点上进行。预训练数据集包含 6700 万个句子对，其中 15% 的词元被掩码。

GPT GPT-2 拥有 1.1 亿个参数。我们使用长度为 1024 的序列和 140 的批大小进行训练，共训练 240k 步，相当于在 OpenWebText 上训练 2 个 epoch。对于调度 Sch1-G，我们继续使用完整模型规模进行训练，直至达到表 5 中报告的耗时。我们使用 1e-4 的学习率和 0.01 的学习率预热比例。实验在 2 个机箱上使用 14 个 Nvidia A100 GPU 进行。

E 微调详情

对于所有 GLUE 任务，我们使用批大小为 32、序列长度为 128，学习率为 2e-5。对于 CoLA、MRPC、STS-B 和 RTE 等小型数据集，我们微调 5 个 epoch（参照大多数相关工作，我们排除 WNLI（Devlin 等，2019；Chen 等人，2022；Gong 等人，2019））；对于其他任务，则微调 3 个 epoch。

对于 SQuAD，我们使用 12 的批大小和 3e-5 的学习率进行微调，共训练 2 个 epoch。SQuAD 指标对序列长度非常敏感，而且大多数模型只有在序列长度超过 384 时才能取得良好效果。因此，在整个调度完成后，我们继续进行预训练：所有对比方法都使用 512 的序列长度训练 100k 步。这使 Bert-large 的实际加速比略有下降，从 2.2x 降至 2.0x。显然，如果在整个预训练过程中都使用 512 的序列长度，就能减轻这种下降。此外，在所有情况下，MSG 的 SQuAD 结果都显著优于基线模型（表 4、5），这还能转化为额外的时间节省。

对于 Wikitext2 微调，我们使用 1e-4 的学习率训练 3 个 epoch，序列长度为 1024，批大小为 8。我们报告验证集上的零样本困惑度和微调后困惑度。

特定 GLUE 任务上的结果

我们在表 6 和表 7 中列出了各项 GLUE 任务的得分。评估指标方面，CoLA 使用 Matthews 相关系数，STS-B 使用 Pearson/Spearman 相关系数，MRPC、QQP 和 SQuAD 使用准确率/F1 值，其余任务均使用准确率。数值为 3 次运行的均值（标准差）。

表 7：Bert-Base 的 GLUE 任务结果。

调度	算子	GLUE A vg.	CoLA SST	-2	MRPC	STS-B
Full	-	80.7(0.2)	52.2(0.8)	90.4(0.2)	**85.9(0.9)/90.1(0.5)**	**88.8(0.1)/88.4(0.1)**
Sch1-B	Net2Net	80.5(0.2)	52.4(1.5)	**91.1(0.4)**	84.3(0.9)/88.8(0.6)	88.3(0.1)/88.0(0.1)
Sch1-B	MSG	**81.0(0.2)**	**58.2(1.6)**	91.0(0.2)	85.0(0.5)/89.4(0.5)	88.1(0.1)/87.6(0.1)

调度	算子	QQP	MNLI(m/mm)	QNLI	RTE	SQuAD v1.1
Full	-	**90.6(0.1)/87.3(0.1)**	**82.5(0.3)/82.9(0.1)**	89.9(0.1)	**65.1(0.7)**	79.1(0.2)/86.9(0.2)
Sch1-B	Net2Net	90.1(0.3)/87.0(0.1)	81.1(0.2)/82.1(0.1)	89.2(0.1)	66.3(0.5)	79.0(0.1)/86.7(0.0)
Sch1-B	MSG	90.0(0.1)/87.0(0.1)	81.8(0.3)/82.4(0.2)	**89.9（0.1）**	63.1(1.6)	**79.6(0.5)/87.2(0.4)**

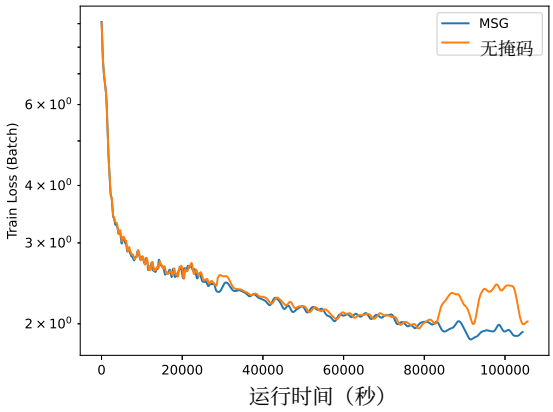


图 3：采用 Rapid-L 训练计划时，使用和不使用掩码的 MSG 训练损失曲线。

G 替代训练计划与结果

根据我们在第3.2.1节中制定的启发式规则，从最大的头数 和隐藏维度开始，仅扩展其他维度，也可以作为一种替代调度方案。在消融实验中，我们构建了这样的训练计划，即用于 Bert-large/base 的 Sch0-L/B，如表 8 所示。采用 Sch0 得到的 GLUE 结果见表 9。

表 8：其他增长计划。“B”和“L”分别代表基础版和大型模型规模。

调度	0~200k	200k~400k	400k~600k	600k~900k
Sch0-L	(768,1024,16,6)	(768,4096,16,6)	(768,4096,16,12)	(768,4096,16,24)

调度	0~200k	200k~400k	400k~600k	600k~700k	700k~900k
sch0-B	(768,768,12,3)	(768,768,12,6)	(768,768,12,9)	(768,768,12,12)	(768,3072,12,12)

H 与 LiGO 的比较

我们采用 *LiGO* 在 (Wang 等人，2022) 的图 2 中报告的实际耗时加速比数据进行比较，因为训练设置与我们的设置非常接近。相对于从头开始训练的基线模型，**MSG** 与*LiGO*的加速比分别为：在 Bert-base 上为**1.8x** 对 1.69x，在 Bert-large 上为**2.2x** 对 1.82x。这表明，我们在数学上严格的算子和他们的数据驱动算子都能有效地实现扩展。再次注意，两者在一定程度上是正交的（第 5.2 节），因为 MSG 可能与任意新的权重初始化兼容。

I 合理性检验：扩展至 1000 亿参数

MSG 导向的增长算子已用于训练大语言模型，模型参数规模从 16B 扩展至 51B，随后扩展至 101B（Li 等人，2023年）。鉴于计算预算有限，无法与完整规模训练进行直接比较，因此我们在此总结该技术报告中的一些合理性检查研究，

表 9：MSG 和 Sch0-L/B 在 GLUE 上的补充结果。

调度	Operator	GLUE A vg.	CoLA SST	-2 MRP	C	
Sch0-L（36小时，8分钟）	MSG	82.8(0.1)	60.0(0.8)	91.9(0.2)	86.8(0.7)/90.7(0.5)	
Sch0-B（17小时，46分钟）	MSG	80.9(0.1)	55.0(0.2)	90.7(0.1)	84.3(0.7)/88.9(0.5)	
调度	**算子**	**STS-B**	**QQP**	**MNLI(m/mm)**	**QNLI**	**RTE**
Sch0-L（36小时，8分钟）	MSG	88.9(0.5)/88.6(0.4)	90.9(0.3)/87.9(0.2)	83.7(0.3)/84.6(0.6)	91.2(0.3)	68.1(0.6)
Sch0-B（17小时，46分钟）	MSG	88.6(0.2)/88.1(0.2)	90.3(0.1)/87.1(0.0)	82.4(0.2)/82.8(0.1)	90.2(0.0)	65.5(1.2)

以展示 MSG 的训练动态在 100B+ 规模的模型上保持稳定的证据，并说明该增长策略成功利用了大型模型的能力。

设置。一个类GPT大语言模型使用24台DGX-A800 GPU（8×80G）服务器进行训练。训练数据由约300B个中英混合词元组成。由于预算有限，Li 等人，2023并未遵循第3.2.1节中的方法。相反，16B、51B和101B模型分别使用245.37B、39.64B和26.54B个词元进行训练。这是一种非常“快速”的增长计划。目标模型的超参数为（10240, 40960, 80, 80）。训练在21.54天内完成，预算为10万美元。

结果。 如图4所示（选自（Li 等人，2023年）中的图 2），MSG 的训练损失曲线在每次扩展后总体保持稳定，并且扩展后相对于 token 数量的损失下降率有所提高。此外，根据其表 5 中的评估结果，101B模型仅消耗较少的数据，便取得了比 50B 模型更大的性能提升。这表明，MSG 扩展后，新权重已成功融入训练过程，这与损失曲线所反映的情况一致。基准评测表明，与使用相近数据量训练的 100B 规模模型（即 GLM-130B）相比，该模型仅需 10% 的训练成本，便能达到基线模型至少 80% 的性能。需要注意的是，这并非严格的比较，因为两个模型及其预训练数据存在很大差异。总而言之，面向MSG的算子在 100B 规模模型的渐进式训练中，展现出了稳定增长后训练过程并从小模型继承知识的能力。

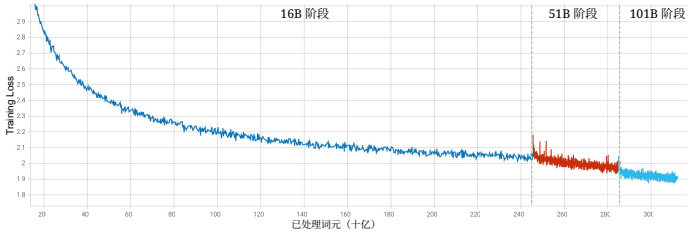


图4：101B模型的训练曲线。

J 附加 GPT 结果

由于我们在表 5 中 GPT 在 Wikitext2 上的困惑度高于原始 GPT-2 工作（Radford 等，2019）报告的结果，因此我们怀疑附录D中的设置导致预训练不足。因此，我们尝试了一种新设置：将批大小增加到 480，并总共训练 100k 步，相当于在 OpenWebText 上训练约 6 个轮次。数据处理方式与一个公开实现（NanoGPT[7]）保持一致。此新设置得到的训练损失为 ~3.10，也接近 NanoGPT 报告的数值。

我们通过网格搜索方法（第3.2.1节）制定了新的调度方案（表 10）。与完整规模训练所需的 54 小时 33 分钟相比，该调度方案将训练时间缩短至 39 小时 30 分钟，加速比为 38%。训练损失曲线随步数的变化如图 5 所示。Wikitext2困惑度分别为 37.65（MSG）和 37.31（完整），两者相近。不过，与 Radford 等，2019 的结果相比，我们仍存在显著差距（~30）。一种可能的解释是，开源的 OpenWeb-Text 数据与 GPT-2 使用的非公开数据不同。请注意，这并不影响我们的结论，因为我们所有的方法都是在相同数据上进行比较的。

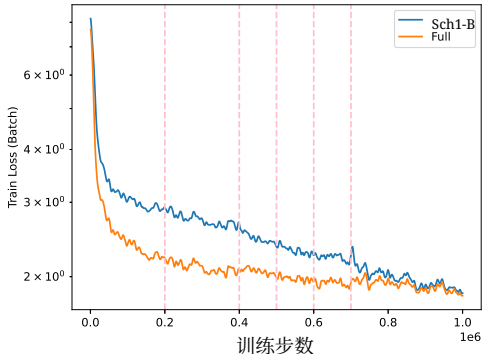
[7]https://github.com/karpathy/nanoGPT

表 10：其他 GPT 结果的调度。

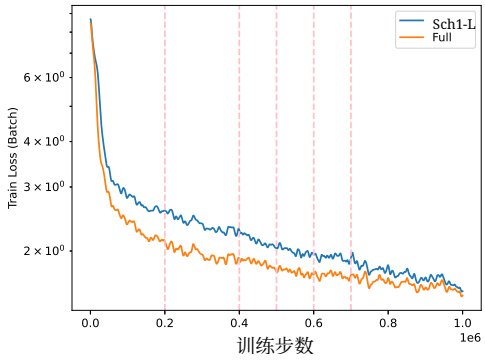
调度	0~30k	30k~60k	60k~80k	80k~100k
Sch-GPT-New	(512,768,8,12)	(768,768,8,12)	(768,768,12,12)	(768,3072,12,12)

K MSG 损失曲线

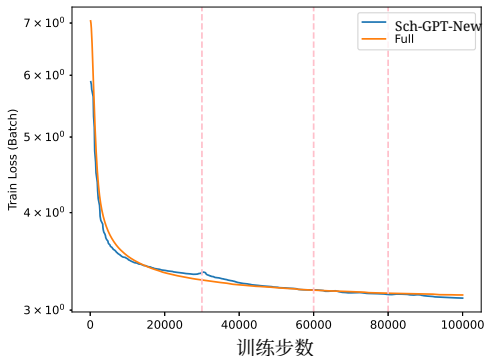
MSG 的训练损失曲线随步数的变化如图 5 所示。GPT-2 的曲线基于附录 J 中的设置，而 BERT 的曲线则来自主要实验（第 4.3 节）。我们观察到，随着模型逐步扩展，MSG 的训练损失逐渐赶上。我们还发现，训练后期较低的损失并不一定意味着性能更优（因为完整语料的处理超过了 1 个周期），因此损失曲线仅能定性地反映模型的能力。



(a) MSG 与完整 Bert-base 的对比



(b) MSG 与完整 Bert-large 的对比



(c) MSG 与完整 GPT-2 的对比

图 5：MSG 损失曲线。粉色竖线标记每个扩展阶段的开始 .