# 目 录

# 介绍 Deep mind 开发的 Alpha chip（芯片设计方法）

NVIDIA 黄仁勋

最近顶级 AI 公司 Deep Mind 开发了基于 AI 用于自动化设计芯片布局方法 Alpha Chip，英伟达（NVIDIA）黄仁勋作了介绍，如下：

黄仁勋介绍：由顶级 AI 公司 Deep Mind 开发的 Alpha Chip 是一种超出人类设计水平、用来进行芯片设计的一种方法。能实现芯片设计全流程自动化，通过互搏可产生人类还没有的棋类数据，可用于自动进行算力优化，用于自动训练大模型、自动生成合成数据，还能在通用人工智能(AGI)研发成功后再经几个小时便能完成其智能超越人类智力水平一万倍的超级人工智能（ASI）最强的芯片布局，还可用于科研（包括 AI 科研）。目前谷歌已用于设计 TPU，联发科已用于加速芯片开发，优化芯片的性能、能耗和面积，Deep Mind 正在用 Alpha Chip 致力于 AI 开始自我进化。

# 介绍 Alpha Chip 芯片 AI 自动设计方法

北京大学 李智宇

芯片平面规划是设计计算机芯片物理布局的工程任务。尽管经过了五十年的研究，芯片平面规划一直难以实现自动化，需要物理设计工程师付出数月的紧张努力才能制造出可生产的布局。在这里，Deep Mind 科学家们提出了一种深度强化学习方法来处理芯片平面规划问题，即 Alpha Chip。在不到六小时的时间内，他们的方法可以自动生成在所有关键指标上优于或与人类产生得相当的芯片平面图，包括功耗、性能和芯片面积。为了实现这一点，他们将芯片平面规划视为一个强化学习问题，并开发了一种基于边缘的图卷积神经网络架构，能够学习到芯片的丰富和可转移的表示。因此，他们的方法利用过去的经验，变得更好、更快地解决新的问题实例，允许由比任何人类设计师都更有经验的人工代理来执行芯片设计。我们的方法已经被用来设计下一代谷歌人工智能（AI）加速器，并且有潜力为每一代新产品节省数千小时的人力。最后，他们相信，更强大的 AI 设计硬件将推动 AI 的进步，创造出两个领域之间的共生关系。

# 一种基于 AI 用于自动快速芯片设计的图形布置方法论

A graph placement methodology for fast chip design

Azalia Mirhoseini 等，Deepmind
北京大学荆琦团队译

## 摘要

芯片平面规划是设计计算机芯片物理布局的工程任务。尽管经过了五十年的研究，芯片平面规划一直难以实现自动化，需要物理设计工程师付出数月的紧张努力才能制造出可生产的布局。在这里，我们提出了一种深度强化学习方法来处理芯片平面规划问题。在不到六小时的时间内，我们的方法可以自动生成在所有关键指标上优于或与人类产生得相当的芯片平面图，包括功耗、性能和芯片面积。为了实现这一点，我们将芯片平面规划视为一个强化学习问题，并开发了一种基于边缘的图卷积神经网络架构，能够学习到芯片的丰富和可转移的表示。因此，我们的方法利用过去的经验，变得更好、更快地解决新的问题实例，允许由比任何人类设计师都更有经验的人工代理来执行芯片设计。我们的方法已经被用来设计下一代谷歌人工智能（AI）加速器，并且有潜力为每一代新产品节省数千小时的人力。最后，我们相信，更强大的 AI 设计硬件将推动 AI 的进步，创造出两个领域之间的共生关系。

## 引言

在这项工作中，我们提出了一种新的基于强化学习（RL）的图布局方法，并在芯片平面规划这一长期以来一直难以实现自动化的挑战性问题上展示了最先进的结果，尽管已经进行了五十年的研究。我们的方法在不到 6 小时内就能生成可制造的芯片平面图，相比之下，最强的基线方法需要人类专家数月的紧张努力。

计算机芯片分为几十个块，每个块都是一个单独的模块，例如内存子系统、计算单元或控制逻辑系统。这些块可以用网表来描述，网表是电路组件的框图，例如宏（内存组件）和标准单元（逻辑门，如 NAND、NOR 和 XOR），所有这些都由导线连接。芯片布局规划涉及将网表布局在芯片画布（二维网格）上，以便优化性能指标（例如功耗、时序、面积和线长），同时遵守对密度和路由拥塞的硬约束。

自 20 世纪 60 年代以来，人们提出了许多芯片平面规划方法，分为三类：基于分区的方法[3-5]、随机/爬坡方法[6-8]和分析求解器[9-14]。然而，这些方法都不能达到人类水平的性能，芯片复杂性的指数增长使得这些技术在现代芯片上基本上无法使用。

这些先前方法的局限性各不相同。例如，基于分区的方法牺牲了全局解决方案的质量，以便扩展到更大的网表，而糟糕的早期分区可能会导致不可接受的最终结果。爬山方法的收敛速度较低，不能扩展到具有数百万或数十亿个节点的现代芯片网表 [13]。在这项工作之前，解析求解器是领先的方法，但它们只能针对可微损失函数进行优化，这意味着它们不能有效地优化关键指标，例如路由拥塞或时序违规。另一方面，我们的方法可以扩展到具有数百万个节点的网表，并直接针对可微或不可微成本函数的任何混合进行优化。此外，我们的方法提高了结果的速度和质量，因为它暴露了更多的芯片布局问题实例。

由于这些先前方法的局限性，人类物理设计人员必须使用商业电子设计自动化 (EDA) 工具迭代几个月，将芯片网表的寄存器传输级别 (RTL) 描述作为输入，生成该网表手动布局在芯片画布上，并为 EDA 工具等待多达 72 小时以评估该布局。在此反馈的基础上，人类设计者要么得出结论，已实现设计标准，生成更新的平面图进行评估，要么向上游 RTL 设计者提供反馈，然后修改低级代码以使布局任务更容易（例如，解决定时违规）。

为了解决芯片平面规划问题，我们开发了一种能够跨芯片泛化的 RL 方法——这意味着它可以从经验中学习，变得更好、更快地布局。

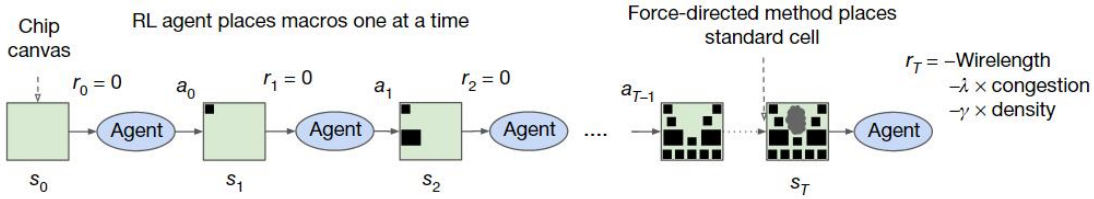



图 1 |我们的方法和训练方案概述。在每个训练迭代中，RL 代理一次布局一个宏（动作、状态和报酬分别用 ai、si 和 ri 表示）。一旦布局所有宏，标准单元就会使用力定向方法布局。中间报酬为零。每次迭代结束时的报酬计算为近似线长、拥塞和密度的线性组合，并作为代理的反馈，以优化下一次迭代的参数。

在这项工作中，我们提出了一种新的基于强化学习的图布局方法，并展示了在芯片平面规划这一长期挑战性问题上的最新成果，这个问题长期以来一直难以实现自动化，尽管已经进行了五十年的研究。与需要人类专家数月紧张工作的强大基线相比，我们的方法能够在不到 6 小时内生成可制造的芯片平面图。

为了解决芯片平面规划问题，我们开发了一种能够跨芯片泛化的 RL 方法——这意味着它可以从经验中学习，变得越来越好和更快地布局新的芯片——-允许芯片设计人员由经验大于任何人类的人工代理辅助。

训练能够跨芯片泛化的布局策略极具挑战性，因为这需要学习优化所有可能的芯片网表示所有可能的画布上。芯片平面规划类似于一个游戏，其中有变化的部分（例如，网表拓扑、宏单元计数、宏单元大小和长宽比）、棋盘（变化的画布大小和长宽比）和获胜条件（不同评估指标的相对重要性或不同的密度和路由拥堵约束）。即使是这个游戏的一个实例（将特定的网表布局在特定的画布上）也有一个巨大的状态－动作空间。例如，在有 1000 个单元的网格上布局 1000 个节点簇的状态空间是 1000 的阶乘（大于 10^2500），而围棋的状态空间是 10^360（引用[15]）。

为了实现泛化，我们专注于学习芯片的可迁移表示，将表示学习建立在预测布局质量的监督任务中。通过设计能够准确预测各种网表及其布局报酬的神经架构，我们能够生成输入网表的丰富特征嵌入。然后，我们使用这种架构作为策略和价值网络的编码器来实现迁移学习。在我们的实验中，我们表明，由于我们的代理暴露在更大的芯片体积和各种芯片中，为新的芯片块生成优化布局既更快又更好，这使我们更接近芯片设计者由具有巨大芯片布局经验的人工代理辅助的未来。
除了对芯片平面规划的直接影响外，我们方法的泛化能力和快速生成高质量解决方案的能力具有重大意义，它解锁了与芯片设计过程的早期阶段共同优化的机会。以前，由于需要数月的人力工作来准确评估给定的架构候选，大规模的架构探索是不可能的。然而，修改架构设计可以对性能产生巨大的影

响，并将促进芯片设计过程的完全自动化。自动化和加速芯片设计过程还可以实现人工智能和硬件的共同设计，产生为重要工作负载定制的高性能芯片，如自动驾驶汽车、医疗设备和数据中心。

在抽象层面上，我们的方法学会了在一定约束下，将超图的节点映射到有限的资源集上。这种形式的布局优化出现在广泛的科学和工程应用中，包括硬件设计[1]、城市规划[16]、疫苗测试和分发[17]以及大脑皮层布局[18]。因此，我们相信我们的布局优化方法可以应用于超出芯片设计的有影响力的布局问题。

除了此处报告的实验结果外，我们的方法已经在现实世界中产生了影响，我们的平面图解决方案已经在最新一代谷歌张量处理单元（TPU）加速器的产品磁带输出中。

# 将芯片平面规划作为一个学习问题

但是，其潜在问题是一个高维上下文强盗问题[19]，正如以前的工作[20-23]，我们选择将其重新表述为一个顺序的马尔可夫决策过程（MDP），因为这样可以更容易地结合下面描述的问题约束。我们的 MDP 由四个关键元素组成：

（1）状态编码部分布局的信息，包括网表（邻接矩阵）、节点特征（宽度、高度、类型）、边缘特征（连接数）、当前节点（宏单元）的布局，以及网表图的元数据（路由分配、总线数、宏单元和标准单元簇）。

（2）动作是所有可能的位置（芯片画布的网格单元），当前宏单元可以布局在这些位置上，而不违反任何关于密度或阻塞的硬约束。

（3）状态转换定义了给定状态和动作的下一个状态的概率分布。

（4）除了最后一个动作外，所有动作的报酬都是 0，最后一个动作的报酬是代理线长、拥堵和密度的负加权和，如下所述。

我们训练了一个由神经网络建模的策略（RL 代理），通过重复情节（状态、动作和报酬的序列），学习采取最大化累积报酬的动作（见图 1）。给定每个布局的累积报酬，我们使用近端策略优化 (PPO) [24] 来更新策略网络的参数。

我们可以正式定义目标函数如下：

$$J(\theta, G) = \frac{1}{K} \sum_{g \approx G} E_{g, p \approx \pi_\theta}[Rp, g] \quad (1)$$

这里 $J(\theta, G)$ 是成本函数。代理由 θ 参数化。大小为 K 的网表数据集用 G 表示，数据集中的每个单个网表写为 g。$R_{p,g}$ 是从应用于网表 g 的策略网络中提取的布局 p 的情节报酬。$E_{g,p}[R_{p,g}]$ 是预期报酬，给定从策略分布 $\pi_\theta$ 中提取的网表 g 和布局 p。

$$R_{p,g} = -Wirelength(p, g) - \lambda Congestion(p, g) - \gamma Density(p, g) \quad (2)$$

在每次迭代中，RL 代理（策略网络）依次布局宏单元。一旦所有宏单元布局完毕，我们使用力导向方法[11,25-27]来近似布局标准单元簇。每次迭代结束时的报酬计算为近似线长、拥堵和密度的线性组合。在

我们的实验中，拥堵权重 λ 设置为 0.01，密度权重 γ 设置为 0.01，最大密度阈值设置为 0.6。

# 设计领域自适应策略

如前所述，微芯片平面规划问题开发领域自适应策略极具挑战性，因为这个问题类似于一个游戏，其中有变化的部分、棋盘和获胜条件，并且有一个巨大的状态－动作空间。为了应对这一挑战，我们首先专注于学习状态空间的丰富表示。我们的直觉是，能够胜任芯片布局这一通用任务的策略也应该能够在推理时将与新未见芯片相关的状态编码成有意义的信号。因此，我们训练了一个神经网络架构，能够预测新网络列表布局的报酬，最终目标是使用该架构作为我们策略的编码器层。

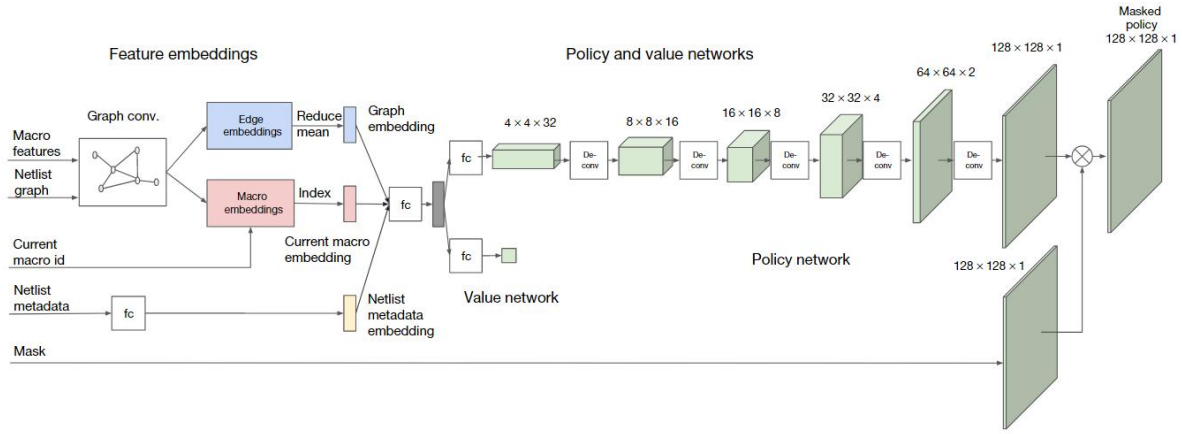



图 2　策略与价值网络架构。嵌入层对有关网表邻接关系、节点特征和当前要布局的宏单元进行编码。然后，该网络分别输出可用网格单元的概率分布和对当前位置的预期报酬的估计。

为了训练这个监督模型，我们需要一个大的芯片布局数据集和相应的报酬标签。因此，我们创建了一个包含 10,000 个芯片布局的数据集，其中输入是与给定布局相关的状态，标签是该布局的报酬。

为了准确预测报酬标签并推广到未知数据，我们开发了一种基于边缘的图神经网络架构，我们称之为 Edge-GNN（基于边的图神经网络）。该网络的作用是嵌入网表，将节点的类型和连通性信息提炼成可用于下游任务的低维向量表示。我们基于边的神经结构一般性的作用如扩展数据图 2 所示。

在 Edge-GNN 中，我们通过连接每个节点的特征（包括节点类型、宽度、高度、x 和 y 坐标以及与其他节点的连通性）来创建每个节点的初始表示。然后，我们迭代地执行以下更新：(1) 每条边通过将全连接网络应用于它所连接的两个节点的串联来更新其表示，以及 (2) 每个节点通过将所有进出边的平均值传递给另一个全连接网络来更新其表示。节点和边缘更新如式(3)所示。

$$e_{i,j} = fc_e(concat(v_i \mid v_j \mid w_{ij}^{\ e}))$$
$$v_i = mean_{j \in Neighbours(v_i)}(e_{ij}) \qquad (3)$$

当 1≤i≤N 时，节点嵌入用$v_i$表示，其中 N 为宏和标准单元簇的总数。连接节点$v_i$和 $v_j$ 的边的向量表示记为 $e_{ij}$。$fc_e$ 表示完全连接层，$w_{ij}^{e}$ 对应边 $e_{ij}$（连接节点 i 和 j）的可学习权值。算法的输出是节点嵌入和边嵌入。

表 1 |与基线的比较

| Name | Method | Timing | | Total area (μm²) | Total power (W) | Wirelength (m) | Congestion | |
|---|---|---|---|---|---|---|---|---|
| | | WNS (ps) | TNS (ns) | | | | H (%) | V (%) |
| Block 1 | RePlAce | 374 | 233.7 | 1,693,139 | 3.70 | 52.14 | 1.82 | 0.06 |
| | Manual | 136 | 47.6 | 1,680,790 | 3.74 | 51.12 | 0.13 | 0.03 |
| | Our method | 84 | 23.3 | 1,681,767 | 3.59 | 51.29 | 0.34 | 0.03 |
| Block 2 | RePlAce | 97 | 6.6 | 785,655 | 3.52 | 61.07 | 1.58 | 0.06 |
| | Manual | 75 | 98.1 | 830,470 | 3.56 | 62.92 | 0.23 | 0.04 |
| | Our method | 59 | 170 | 694,757 | 3.13 | 59.11 | 0.45 | 0.03 |
| Block 3 | RePlAce | 193 | 3.9 | 867,390 | 1.36 | 18.84 | 0.19 | 0.05 |
| | Manual | 18 | 0.2 | 869,779 | 1.42 | 20.74 | 0.22 | 0.07 |
| | Our method | 11 | 2.2 | 868,101 | 1.38 | 20.80 | 0.04 | 0.04 |
| Block 4 | RePlAce | 58 | 11.2 | 944,211 | 2.21 | 27.37 | 0.03 | 0.03 |
| | Manual | 58 | 17.9 | 947,766 | 2.17 | 29.16 | 0.00 | 0.01 |
| | Our method | 52 | 0.7 | 942,867 | 2.21 | 28.50 | 0.03 | 0.02 |
| Block 5 | RePlAce | 156 | 254.6 | 1,477,283 | 3.24 | 31.83 | 0.04 | 0.03 |
| | Manual | 107 | 97.2 | 1,480,881 | 3.23 | 37.99 | 0.00 | 0.01 |
| | Ours | 68 | 141.0 | 1,472,302 | 3.28 | 36.59 | 0.01 | 0.03 |

Here, we compare our method with the state-of-the-art method (RePlAce[14]) and with manual placements using an industry-standard EDA tool. For all metrics in this table, lower is better. H, horizontal; V, vertical.

在这里，我们将我们的方法与最先进的方法 (RePlAce14) 和使用行业标准 EDA 工具手动布局进行比较。对于该表中的所有指标，越低越好。H，水平； V，垂直。

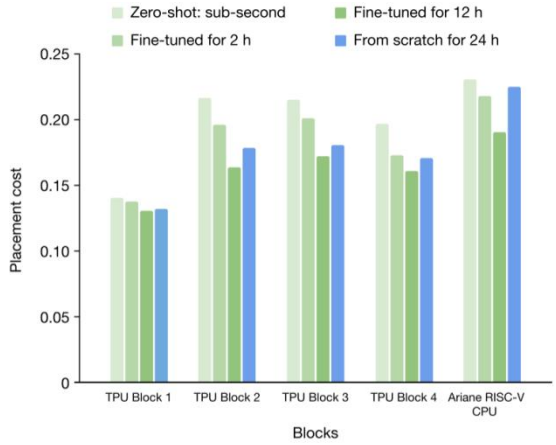



**图 3 从头开始训练与不同时间量的微调的比较。**对于每个块，我们显示了基于 zero-shot，2 小时和 12 小时微调后的结果，以及从头开始训练策略的结果。从表中可以看出，预训练的策略网络的性能始终优于从头开始训练的策略网络，这证明了离线从训练数据中学习的有效性。

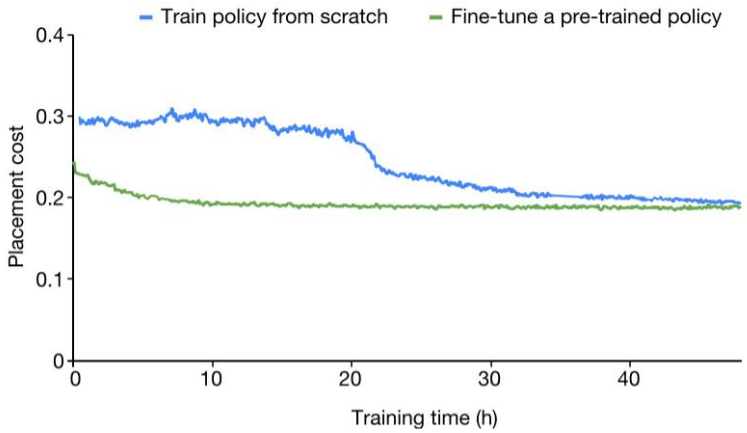



**图 4 Ariane RISC-V CPU 上的收敛图。**从头开始训练策略网络与为一块 Ariane RISC-V CPU 微调预训练策略网络的布局成本的比较。密度权重设为 0.1，拥塞权重设为 0。

监督模型通过回归训练来最小化加权均方损失（负报酬）。这一监督任务使我们能够找到在不同网表中泛化报酬预测所需的特征和架构。为了将 Edge-GNN 集成到我们的 RL 策略网络中，我们移除了预测层，然后将其用作策略网络的编码器，如图 2 所示。

在推理时布局新的网表时，我们会加载策略网络的预训练权重并应用于新的网表。我们将使用预训练策略且未经过微调生成的布局称为零样本布局。这样的布局可以在亚秒级时间内生成，因为每个宏仅需要通过预训练策略进行一次前向传递。我们可以通过微调策略网络进一步优化布局质量。这样做可以让我们灵活地选择使用预训练的权重（这些权重已经学习到了输入状态的丰富表示）或进一步微调这些权重以针对特定芯片网表的属性进行优化。

图 1 展示了所提出的策略网络（由等式(1)中的$\pi_\theta$建模）和价值网络架构的概览。输入包括网表超图（表示为邻接矩阵和节点特征列表）、当前要布局的节点的身份、网表的元数据以及工艺技术节点（例如，7 纳米）。网表被送入我们的 Edge-GNN 架构以生成部分布局网表和当前节点的嵌入。我们使用前馈网络对元数据进行嵌入。这些嵌入向量随后被连接起来形成状态嵌入，该状态嵌入被传递给另一个前馈神经网络以生成状态的最终表示。这个状态接着被送入策略网络（由五个反卷积、批标准化[28]和修正线性单元（ReLU）激活层[29]组成）以生成动作的概率分布，并被送入价值网络（由前馈网络组成）以预测输入状态的价值。反卷积层具有 3×3 的内核大小、2 的步幅以及 16、8、4、2 和 1 个滤波器通道。

# 实证评估

在本节中，我们评估了我们方法的泛化能力，探讨了使用预训练策略的影响，并将我们的方法与最先进基线进行了比较。我们还检查了生成布局的视觉外观，并提供了关于我们策略行为的见解。

就资源使用而言，对于预训练，我们使用的工作者数量与训练数据集中的块数相同（例如，对于包含 20 个块的最大训练集，我们使用了 20 个工作者进行预训练），预训练运行时间为 48 小时。为了生成表 1 中的微调结果，我们的方法在 16 个工作者上运行，最长 6 小时，但由于早期停止，实际运行时间通常显著较低。对于预训练和微调，每个工作者都由一个 Nvidia Volta 图形处理单元（GPU）和 10 个中央处理单元（CPU）组成，每个 CPU 配备 2GB 的 RAM。对于零样本模式（将预训练策略应用于新的网表而不进行微调），我们可以在单个 GPU 上不到一秒钟的时间内生成一个布局。

# 域适应结果

图 3 对比了使用预训练策略生成的布局质量和从头开始训练策略生成的布局质量。训练数据集由 TPU 模块和开源 Ariane RISC-V CPU[30]组成。在每次实验中，我们在所有模块上预训练策略，除了目标模块，在该模块上我们进行评估。我们展示了零样本模式的结果，以及在特定设计上微调预训练策略 2 小时和 12 小时后的结果。

从头开始训练的策略收敛所需时间长得多，即使在 24 小时后，其结果（根据报酬函数评估）也比微调策略在 12 小时内达到的结果差。这表明，在预训练过程中接触到许多不同的设计，能够使新遇到的模块更快地生成更高质量的布局。

图 4 显示了从头开始训练与基于预训练策略网络训练 Ariane RISC-V CPU[30]的收敛曲线。预训练的策略不仅初始布局成本更低，而且比从头开始训练的策略快超过 30 小时达到收敛。

**从更大的数据集中学习。**接下来，我们探讨训练数据对我们的策略学习能力的影响。TPU 芯片模块非常多样化，我们精心挑选了涵盖代表性的功能范围（例如，片上和片间网络模块、计算核心、内存控制器、数据传输缓冲区和逻辑，以及各种接口控制器）、饱和度（宏的总面积与画布面积之比，<30%，30-60% 和 >60%）和宏计数（最多 107 个）的模块。小训练集包含 2 个模块，中等训练集包含 5 个模块，而大训练集则包含 20 个模块。随着我们在更多的芯片模块上进行预训练，我们能够更快地为新的未见过的芯片模块生成更高质量的布局。图 5 展示了更大训练集对性能的影响。随着训练集从 2 个模块增加到 5 个模块，最后增加到 20 个模块，无论是在零样本情况下还是在经过相同小时数的微调之后，策略网络都能生成更好的布局。这表明，当我们让策略网络接触到更多种类不同的芯片设计时，它就越不容易拟合，并且更能泛化到新的未见过的设计。

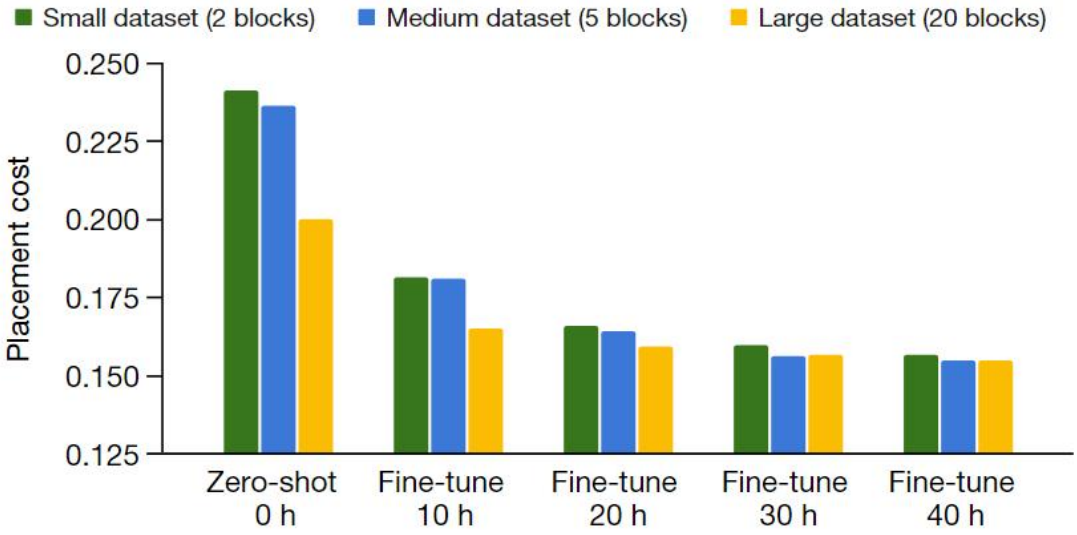



图 5 |预训练数据集大小的影响。我们在三个不同的训练数据集（小数据集是中等数据集的子集）上预训练策略网络，中等数据集是大数据集的子集）。然后，我们在测试块上微调这个预训练的策略网络，并报告各种训练持续时间的成本。随着数据集大小的增加，收敛时间和生成的布局质量都会增加。

**与基准方法的比较。**在本节中，我们将我们的方法与最先进水平的 RePlAce[14]以及上一代 TPU 的生产设计进行了比较，后者是由一组人类物理设计师生成的。结果见表 1。

为了进行公平的比较，我们确保所有方法都有相同的实验设置，包括相同的输入和 EDA 工具设置。值得注意的是，我们自行运行了所有 RePlAce 和我们方法的评估，但我们依赖于 TPU 物理设计团队分享他们最佳手动布局的指标，他们可能使用了略有不同的 EDA 版本进行评估。更多详情请参阅扩展数据表 1。

对于我们的方法，我们使用在最大数据集（20 个 TPU 模块）上预训练的策略，然后在五个目标未见过的模块（标记为模块 1-5）上进行不超过 6 小时的微调。出于保密原因，我们不能透露这些模块的具体细节，但每个模块最多包含 131 个宏和数百万个标准单元。

在评估芯片平面图的质量时，有几个重要的指标会相互权衡。没有单一的指标可以用来捕捉布局的整体质量，因此我们报告了所有关键指标，包括总线长、时序、路由拥塞（水平和垂直）、面积和功耗。

时序通过总负松弛（TNS）和最坏负松弛（WNS）来报告。负松弛是信号延迟超出预期延迟程度的衡量标准。时序和拥塞是约束条件，而线长、功耗和面积是需要优化的指标。

为了与具有不同目标函数的 RePlAce 进行比较，我们将商业 EDA 工具的输出视为真实情况。为了进行这种比较，我们固定了由我们的方法和 RePlAce 生成的宏布局，并允许商业 EDA 工具根据我们的生产工作流程中的设置进一步优化标准单元的布局。我们使用了参考文献[31]中提供的 RePLace 版本，该版本基于 2020 年 1 月 9 日发布的代码版本。除了密度阈值（RePlAce 受益于低于默认值的阈值外），我们使用了默认设置，并没有使用 RePlAce 的时间驱动能力。

如表 1 所示，我们的方法在生成符合设计标准的布局方面优于 RePlAce。尽管 RePlAce 运行速度更快，可以在单个 3.7GHz 的 Intel CPU 上不到一小时内完成，但生成的布局通常质量较低。鉴于底层工艺技术节点施加的限制，如果 WNS 显著高于 150ps，或者水平或垂直拥塞超过 1%，那么在设计流程的后期阶段将无法满足时序约束，这使得许多 RePlAce 布局（模块 1、2、3）不可用。这些结果表明，我们的方法在生成符合设计标准的高质量布局方面是有效的。

表 1 还显示了人工基线的结果，即上一代 TPU 芯片的实际生产设计。这一基线是由 TPU 的物理设计团队生成的，涉及了数月期间由商业 EDA 工具反馈指导的多次布局优化迭代。我们的方法和人类专家都能够一致地生成符合时序和拥塞要求的可行布局。然而，我们的方法在面积、功耗和线长方面也超过了或匹配了人工布局。此外，我们基于端到端学习的方法花费的时间远少于满足设计标准所需的时间。

## 结论

在这项工作中，我们提出了一种基于强化学习（RL）的芯片布局方法，该方法能够进行领域适应。随着布局更多的芯片网表，RL 代理变得更加高效且更快速地优化布局。我们展示了我们的方法能够在不到六小时的时间内生成与人类专家相当或更优的芯片布局，而人类通常需要几个月的时间才能为现代加速器生成可接受的布局。我们的方法已在生产中用于设计下一代 Google TPU。

## 在线内容

任何方法、附加参考资料、《自然研究》报告摘要、源数据、扩展数据、补充信息、致谢、同行评审信息；作者贡献和竞争利益的详细说明；以及数据和代码可用性的声明，均可通过以下链接获取：https://doi.org/10.1038/s41586-021-03544-w。

# 方法

在下文中，我们提供了所提出方法的细节。

## 问题陈述

在本工作中，我们针对芯片平面规划问题，其目标是将网表（描述芯片的超图）的节点映射到芯片画布（一个有界的二维空间）上，以优化最终的功耗、性能和面积（PPA）。在本节中，我们概述了如何将问题表述为强化学习（RL）问题，随后详细描述了报酬函数、动作和状态表示、策略架构及策略更新。

## 我们方法的概述

我们采用深度强化学习方法解决芯片平面规划问题，其中 RL 代理（策略网络）顺序布局宏。一旦所有宏都被布局完毕，我们使用力导向（FD）方法[11,25-27]来布局标准单元簇，如图 1 所示。

在本节中，我们定义了报酬 r、状态 s、动作 a、参数为θ的策略网络架构 $\pi_\theta(a|s)$，以及最后用于训练这些参数的优化方法。在我们的设定中，初始状态 $s_0$ 下，我们有一个空的芯片画布和一个未布局的网表。每一步布局一个宏，最终状态 $s_T$ 对应于完全布局的网表。因此，T 等于网表中宏的总数。在每个时间进一步 t，代理从状态 $s_t$ 开始，采取一个动作 $a_t$，到达一个新的状态 $s_{t+1}$，并从环境中接收一个报酬 $r_t$（对于 t < T 时为 0，对于 t = T 时为负代理成本）。

我们定义 $s_t$ 为时间 t 时状态的特征拼接，包括网表的图嵌入（包含已布局和未布局的节点）、当前待布局宏的节点嵌入、关于网表的元数据，以及一个表示将当前节点布局在网格每个单元上的可行性的掩码。

动作空间是第 t 个宏的所有有效布局位置，这是密度掩码的函数。动作 $a_t$ 是 RL 策略网络预测的第 t 个宏的单元布局位置。$s_{t+1}$ 是下一个状态，包括一个更新的表示，其中包含了关于新布局宏的信息、更新的密度掩码以及下一个待布局节点的嵌入。在我们的表述中，除了最终的 $r_T$，其余每个时间步的 $r_t$ 均为 0，而 $r_T$ 是一个近似线长、拥塞和密度的加权和。

通过重复剧集（状态、动作和报酬的序列），策略网络学会了采取能最大化累积报酬的动作。我们使

用 PPO[24]根据每个布局的累积报酬来更新策略网络的参数。

# 详细方法论

我们的目标是在路由拥塞和密度的约束下最小化功耗、性能和面积（PPA）。我们的真实报酬是商业 EDA 工具的输出，包括线长、路由拥塞、密度、功耗、时序和面积。然而，RL 策略需要数千个样本来有效地学习，因此至关重要的是报酬函数必须快速评估，理想情况下在几毫秒内完成。为了有效，这些近似报酬函数必须与真实报酬正相关。因此，我们成本的一个组成部分是线长，因为它不仅评估成本低廉，而且还与功耗和性能（时序）相关。

为了将多个目标合并成一个可以优化的报酬函数，我们取了代理线长、拥塞和密度的加权和，其中权重可以用来探索这些指标之间的权衡。虽然我们将拥塞视为软约束（即，较低的拥塞改善了报酬函数），但我们把密度作为硬约束处理，屏蔽掉那些密度超过目标密度的动作（即将节点布局到的网格单元）。

为了保持每次迭代的运行时间较小，我们对报酬函数的计算应用了几种近似方法：

1. 使用基于最小割目标的分区技术 hMETIS[32]，将数百万个标准单元分组为几千个簇。一旦所有宏都被布局，我们使用力导向（FD）方法来布局标准单元簇。这样做使我们能够生成一个近似但快速的标准单元布局，从而促进策略网络的优化。

2. 将网格离散化为几千个网格单元，并将宏和标准单元簇的中心布局在网格单元的中心。

3. 在计算线长时，我们简化假设所有离开标准单元簇的导线都起源于簇的中心。

4. 计算路由拥塞成本时，我们只考虑最拥挤的 10%网格单元的平均拥塞。

一个芯片网表通常由数百个宏和数百万个标准单元组成。由于标准单元的面积可忽略不计，可以近似为零面积的点，允许分析求解器以较小的误差最优地布局它们。相比之下，宏的面积要大得多，不能使用相同的分析技术进行最优布局。我们选择了宏布局作为目标，因为这是一个更具挑战性的问题，过去需要人类专家花费数月时间迭代才能生成高质量的布局。

**输入网表的综合。**我们使用商业工具从 RTL 综合网表。综合过程具有物理感知性，这意味着它可以访问平面规划的尺寸和输入/输出引脚的位置，这些信息来自跨模块和模块内部的信息。

**网格行和列的选择。**鉴于芯片画布的尺寸，有许多方式可以将二维画布离散化为网格单元。这个决定影响着优化的难度和最终布局的质量。我们将行和列的最大数量限制为 128。我们将选择最优行数和列数视为一个装箱问题，并根据它们产生的浪费空间量对不同的行列组合进行排名。在我们的实验中，平均使用了 30 行和 30 列。

**宏顺序的选择。**为了确定宏布局的顺序，我们按降序排列宏的大小，并使用拓扑排序来打破平局。通过先布局较大的宏，我们减少了后续宏无可行布局位置的可能性。拓扑排序有助于策略网络学习将连接的节点布局得靠近彼此。另一种潜在的方法是联合优化宏的顺序及其布局，使选择下一个要布局的节点成为行动空间的一部分。然而，这种扩大的行动空间会显著增加问题的复杂性，我们发现这种启发式方法在实践中是有效的。

**标准单元的聚类。**为了快速布局标准单元以向我们的 RL 策略提供信号，我们首先将数百万个标准单元聚类为几千个簇。对于芯片网表的聚类，已有大量的研究工作[33-38]。正如文献[39]所建议的那样，这种聚类不仅有助于减少问题规模，还有助于“防止错误”（例如，防止时序路径被拆开）。我们还将聚类后的网表提供给予之比较的每个基线方法。为了执行这种聚类，我们使用了一个基于多级超图划分方案的标准开源库 hMETIS[39]，该方案有两个重要阶段：（1）粗化阶段，（2）非粗化和细化阶段。使用 hMETIS 聚类后，我们根据物理综合的初步布局，即芯片设计过程中的前一步骤，使用启发式方法重新平衡簇的大小。

**邻接矩阵的生成。**为了将网表超图转换为可以由 Edge-GNN 编码器消费的邻接矩阵，我们应用了以下转换。对于聚类网表中每对节点（无论是宏还是标准单元簇），我们根据以下权重在邻接矩阵中生成一条边。如果两个节点之间的寄存器距离大于 4，则不创建边。否则，我们随着距离的增长应用指数衰减权重，如果距离为 0 则权重为 1，每增加一个单位的距离权重减半。

**标准单元的布局。**为了布局标准单元簇，我们采用了类似于经典力导向（FD）方法[40]的方法。我们将网表表示为一套弹簧系统，该系统根据权重×距离公式对每个节点施加力，使紧密连接的节点相互吸引。我们还在重叠节点之间引入排斥力以降低布局密度。应用所有力后，我们沿着力矢量的方向移动节点。为了减少振荡，我们为每次移动设置了最大距离。

**后处理。**为了准备由商业 EDA 工具评估的布局，我们执行了一个简单的合法化步骤，将宏对齐到最近的电源网格。然后，我们固定宏的布局，并使用 EDA 工具来布局标准单元并评估布局。

# 报酬

**连线长度**。按照文献[40-43]，我们采用半周长连线长度（HPWL），这是最常用的连线长度近似。HPWL 定义为网表中所有节点的边界框的半周长。给定网（边）i 的 HPWL 是：

$$HPWL(i) = (\max_{b \in i}\{x_b\} - \min_{b \in i}\{x_b\} + 1) + (\max_{b \in i}\{y_b\} - \min_{b \in i}\{y_b\} + 1) \quad (4)$$

这里 $x_b$ 和 $y_b$ 是网 i 的端点的 x 和 y 坐标。然后通过取所有半周长边界框的归一化和来计算总 HPWL，如方程（5）所示。这里 q(i)是一个归一化因子，通过增加节点数量来提高估计的准确性，其中 $N_{netlist}$ 是网的数量。我们按如下方式计算总 HPWL：

$$HPWL(netlist) = \sum_{i=1}^{N_{netlist}} q(i) HPWL(i) \quad (5)$$

连线长度还有与功率和时序等其他重要指标相关的优势。尽管我们的方法没有直接针对这些其他指标进行优化，但它生成的布局满足设计标准，包括功率和时序（如表 1 所示）。

**路由拥塞。**在计算代理拥塞[44]时，我们也遵循了惯例，使用了一种基于网络上驱动器和负载位置的简单确定性路由方法。路由的网络在其穿过的每个网格单元中占用了一定比例的可用路由资源（由底层半导体制造技术决定）。我们分别跟踪每个网格单元中的垂直和水平分配。为了平滑拥塞估计，我们在垂直和水平方向上运行 5×1 的卷积滤波器。所有网络路由完成后，我们取前 10%拥塞值的平均值，这一做法受到了 MAPLE[44]中 ABA10 指标的启发。等式(2)中的拥塞成本就是通过此过程计算出的前 10%平均拥塞值。

**密度。**我们将密度视为硬约束，不允许策略网络将宏布局在会导致密度超过目标值（$\max_{density}$）或导致宏重叠不可行的位置。这种方法有两个好处：（1）减少了策略网络生成的无效布局数量，（2）缩小了优化问题的搜索空间，使其在计算上更加可行。

一个标准单元簇的可行布局必须满足以下标准：每个网格单元中布局项目的密度不得超过给定的目标密度阈值（$\max_{density}$）。在我们的实验中，我们将这一阈值设为 0.6，以避免过度利用，这将使布局变得不可用。为了满足这一约束，在每个 RL 步骤中，我们计算当前的密度掩码，这是一个二进制的 m×n 矩阵，表示我们可以布局当前节点中心的网格单元，而不违反密度阈值。在选择动作之前，我们首先计算掩码与策略网络输出的点积，然后从结果的概率分布中采样可行位置。这种方法防止了策略网络生成带有重叠宏或密集标准单元区域的布局。我们还通过将阻塞区域的密度函数设置为 1 来启用对阻塞区域（如时钟带）的布局。

# 动作表示

为了策略优化目的，我们将画布转换为 m×n 网格。因此，对于任何给定的状态，动作空间（或策略网络的输出）是当前宏单元在 m×n 网格上布局的概率分布。然后从这个概率分布中采样动作。

# 状态表示

我们的状态包含有关聚类网表的邻接矩阵、其节点特征（宽度、高度、类型）、边特征（连接数）、当前要布局的节点（宏单元）以及网表和底层技术（例如，路由分配、总线数、宏单元和标准单元簇）的元数据的信息。接下来，我们讨论如何处理这些特征，以学习芯片平面规划问题的有效表示。

# 启用迁移学习

为了发现领域适应性架构，我们提出将策略架构搜索基于监督任务，即预测报酬函数的值。我们采取这种方法是因为在 RL 设置中探索的成本会高得多，而且训练一个领域适应性策略网络的基础复杂性会非常高，因为它涉及到涵盖所有可能芯片的所有可能布局的巨大状态空间。此外，不同的网表和网格大小可能具有非常不同的属性，包括节点数量、宏大小、网格拓扑结构和画布的宽度和高度的不同。这种方法背后的直觉是，一个能够在芯片之间转移布局优化的策略网络架构也应该能够在推理时将与新未见过的芯片相关的状态编码成有意义的信号。因此，我们提出了训练一个能够预测新网表报酬的神经网络架构，最终目标是将此架构用作我们策略网络的编码层。

为了训练这个监督模型，我们需要一个大型的芯片平面规划及其相应的报酬标签的数据集。因此，我们创建了一个包含 10,000 个芯片平面规划的数据集，其中输入是与给定平面规划相关联的状态，标签是该平面规划的报酬（线长和拥塞）。我们通过为五个 TPU 模块中的每一个生成 2,000 个平面规划来构建这个数据集。为了收集多样化的平面规划，我们使用不同的拥塞权重（从 0 到 1）和随机种子训练了一个普通的策略网络，并在策略训练过程中收集了平面规划的快照。未训练的策略网络最初具有随机权重，生成的平面规划质量较低，但随着策略网络的训练，生成的平面规划质量提高，使我们能够收集一个包含不同质量平面规划的多样化数据集。

为了训练一个能够准确预测线长和拥塞标签并泛化到未见过数据的监督模型，我们开发了一种图神经网络架构（Edge-GNN），用于嵌入网表信息。Edge-GNN 的作用是将有关节点类型和连接性的信息提炼成可用于下游任务的低维向量表示。这样的下游任务示例包括节点分类[45]、设备布局[46]、链接预测[47]和设计规则检查（DRC）违规预测[48]。

我们首先通过连接每个节点的特征（包括节点类型、宽度、高度和 x、y 坐标）来创建每个节点的向量表示。我们还将节点邻接信息作为算法的输入。然后，我们反复执行以下更新：（1）每条边通过对其相邻节点嵌入的聚合表示应用全连接网络来更新其表示，（2）每个节点通过取相邻边嵌入的平均值来更新其表示。节点和边的更新如等式(3)所示。

节点嵌入用 $v_i$ 表示（$1 \le i \le N$），其中 N 是所有宏单元和标准单元簇的总数。连接节点 $v_i$ 和 $v_j$ 的边的向量表示为 $e_{ij}$。边（$e_{ij}$）和节点（$v_i$）嵌入都是 32 维的。$v_i$ 是由将节点特征（类型、宽度、高度、x、y 坐标）通过前馈网络进行初始化的。$fc_e$ 是一个 65×32 的前馈网络，$w_{ij}^{e}$ 是其权重。是一个 1×1 的权重，对应于相邻节点之间的网格数。Neighbours($v_i$)表示 $v_i$ 的邻居。人工智能是一门研究如何让计算机模拟人类智能的科学。该算法的输出是节点和边嵌入。

我们的监督模型包括以下部分。（1）前面提到的图神经网络（Edge-GNN），它嵌入了关于节点类型和网表邻接矩阵的信息。（2）一个全连接的前馈网络，它嵌入了网表元数据，包括关于底层半导体技术（水平和垂直路由容量）、总网络数（边）、宏、标准单元簇、画布大小和网格中的行数和列数的信息。（3）一个全连接的前馈网络（预测层），其输入是网表邻接矩阵和元数据嵌入的连接，输出是报酬预测。网表嵌入是通过对边嵌入应用简化均值函数创建的。监督模型通过回归训练以最小化线长和拥塞的均方损失的加权和。

这项监督任务使我们能够找到泛化网表间报酬预测所需的特征和架构。为了将这种架构融入我们的策略网络中，我们简单地移除了预测层，然后将剩余的网络用作策略网络的编码器，如图 2 所示。

为了处理对应于不同行和列选择的不同网格大小，我们将网格大小设置为 128×128，并为小于 128 行和列的网格大小屏蔽未使用的 L 形部分。在推理时布局一个新的测试网表时，我们加载策略网络的预训练权重并将其应用于新的网表。我们将由预训练策略网络生成的未经微调的布局称为零样本布局。这种布局可以在不到一秒钟的时间内生成，因为它只需要对每个宏执行一次预训练策略网络的推理步骤。我们可以通过微调策略网络进一步优化布局质量，这意味着我们有选择直接在推理时使用预训练权重（这些权重已经学习到了输入状态的丰富表示）或进一步微调这些权重以优化特定芯片网表的特性。

# 策略网络架构

图 1 展示了我们为芯片平面规划开发的策略网络（由等式(1)中的$\pi_\theta$建模）和价值网络架构的概览。这些网络的输入包括对应于网表超图的邻接矩阵和节点特征、当前待布局节点的身份以及网表和半导体技术的元数据。网表通过我们前面描述的图神经网络架构（Edge-GNN）传递。Edge-GNN 生成了（1）部分布局的超图和（2）当前节点的嵌入。我们使用一个简单的前馈网络来嵌入（3）元数据。这三个嵌入向量随后被连接起来形成状态嵌入，该状态嵌入被传递给一个前馈神经网络。前馈网络的输出接着被送入策略网络（由五个反卷积、批归一化和 ReLU 激活层组成）以生成动作的概率分布，并被送入价值网络（由一个前馈网络组成）以预测输入状态的价值。

**策略网络更新：训练参数θ。**在等式(1)中，目标是训练一个策略网络$\pi_\theta$，该网络在策略网络的布局分布上最大化报酬（$R_{p,g}$）的期望值（E）。为了优化策略网络的参数，我们使用带有裁剪目标的 PPO[24]，如下所示：

$$L^{CLIP}(\theta)=\hat{E}_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta),1-\varepsilon,1+\varepsilon)\hat{A}_t)]$$

其中$\hat{E}_t$表示时间步长 t 的预期值，$r_t$是新政策和旧政策，$\hat{A}_t$是估计的优势时间步长 t。

# 实验设置

为了进行公平的比较，我们确保我们的方法和所有基线方法都能访问相同的输入和相同的评估设置。扩展数据图 1 显示了我们用于进行评估的流程。

一旦每种方法完成了网表的布局，宏的位置就会被冻结并对齐到电源网格。接下来，EDA 工具执行标准单元的布局。EDA 工具的设置直接来自我们的生产流程，因此我们无法分享所有细节。表 1 中的最终指标是在 PlaceOpt 之后报告的，这意味着全局路由已由 EDA 工具完成。

对标准单元进行聚类使我们的方法能够更有效地优化宏的布局。因此，我们让 RePlAce 也能访问聚类的标准单元，并发现其性能也得到了提升，所以我们在聚类标准单元的网表上报告了 RePlAce 的结果。尽管 RePlAce 的默认密度阈值为 1.0，但我们发现设置为 0.6 时性能更好，所以我们用这个设置来报告 RePlAce 的性能。在所有其他情况下，我们使用了 RePlAce 的默认设置和成本函数。为了保证可复现性，我们在扩展数据表 1 中提供了我们 RL 算法的所有架构细节和超参数设置，以及在扩展数据表 2 中提供了用于布局标准单元的 FD 方法的相关信息。

反卷积层的核大小为 3×3，步幅为 2，滤波器通道数分别为 16、8、4、2 和 1。为了对每个芯片块的标准单元进行聚类，我们使用了 hMETIS[32]，它将数百万个标准单元划分为数千个簇。hMETIS 的超参数列在扩展数据表 3 中。对于所有其他 hMETIS 超参数，我们只是使用默认设置（有关这些默认值的具体数值和每个超参数的更多详细信息，请参见 hMETIS 手册[49]）。我们注意到我们使用的是授权版本的 hMETIS，但据我们所知，开放源代码版本也提供了相同的功能。

为了避免过拟合，我们采用了一种早期停止机制，在策略收敛后停止 RL 训练。更具体地说，当评估回

报在两小时内没有比迄今为止的最佳回报提高至少 0.5%时，训练就会停止。

# 开源基准测试： Ariane RISC-V

对于 Ariane 基准测试，我们使用了以下开源设计[30]（https://github.com/pulp-platform/ariane），并将所有逻辑存储器映射到大小为 256×16 的物理存储器上，从而得到 133 个宏单元。在扩展数据图 4 中，我们比较了通过从头开始训练我们的方法所生成的布局，以及通过预训练策略以零样本模式生成的布局。

# 在生产环境中的应用

我们的方法被用于最近的 Google TPU 产品的 tapeout 过程中。我们通过 PlaceOpt 完全自动化了布局过程，在这一点上，设计被发送给第三方进行布局后的优化，包括详细的布线、时钟树综合和时钟后的优化。这是许多硬件团队的标准做法，物理设计师们会花几个月的时间与商业 EDA 工具迭代，以产生符合严格要求的设计，进而进入下一阶段。

在生产流程中，我们使用了表 1 中描述的相同的 RL 方法和相同的 EDA 工作流来布局标准单元。尽管 RL 布局已经与手动设计相当，我们还是进行了额外的模拟退火（SA）微调步骤以进一步提升性能，这有助于改善宏的方向，因为我们目前在 RL 中不执行宏镜像。添加这一微调步骤平均提高了 1.07%的线长（标准差=0.04%），略微减少了时序（平均 TNS 减少了 1.18 纳秒；标准差=2.4 纳秒），并且几乎不影响拥塞（所有情况下垂直或水平拥塞的变化不超过 0.02%）。最终的端到端运行时间平均为 8 小时。自从那次生产发布以来，我们已经在生产工作流中用一个贪婪的后处理步骤替换了 SA，该步骤在几分钟内调整宏的方向，显著减少了我们的端到端运行时间而不会降低质量。

# 成本权衡的影响

在扩展数据表 4 中，我们进行了一项消融研究，以考察拥塞权重对 PlaceOpt 后结果质量（来自商业 EDA 工具的最终结果质量）的影响。正如预期的那样，增加拥塞权重在一定程度上改善了水平和垂直拥塞，但由于这两个指标之间存在固有的权衡，也会导致线长恶化。在这种情况下，拥塞权重为 0.1 代表了一个“甜蜜点”，因为此时路由拥塞已经很低，而线长尚未过度恶化，这两者共同有助于降低 TNS 和 WNS。

### 对噪声的鲁棒性

为了展示我们方法对噪声的敏感性，我们在 Ariane RISC-V 模块上进行了八次微调运行，每次使用不同的随机种子，并在扩展数据表 5 中报告了结果（包括代理线长、拥塞和密度）。我们的评估表明，随机种子的选择对所有指标的影响微乎其微，包括代理线长、拥塞和密度，所有运行的整体成本标准差仅为 0.0022。

# 泛化能力与训练数据

随着我们在更多的芯片块上进行训练，我们能够加快在新块上的微调过程，并更快地生成更高质量的

结果。正如前面所讨论的，当我们从 2 个块（小数据集）增加到 5 个块（中等数据集），最后增加到 20 个块（大数据集）时，策略网络在零样本情况下以及在相同小时数的微调后都能生成更好的布局。扩展数据图 3 展示了策略网络在预训练过程中一个测试块（一个未包含在训练集中的 TPU 块）的布局成本。我们可以看到，对于小训练数据集，策略网络迅速过拟合到训练数据，导致测试数据上的性能下降；而在最大的数据集上，策略网络过拟合所需的时间更长，且在这个较大数据集上预训练的策略网络在测试数据上产生了更好的结果。这一图表表明，随着我们让策略网络接触到更多不同种类的训练块，策略网络的预训练时间将会延长，但它将变得更不容易过拟合，并且在为新的未见过的块寻找优化布局方面表现得更好。

# 见解与可视化

在这里，我们分享了一些关于我们方法行为的观察，这些观察可能为表 1 中的指标提供一些见解。一个观察是，RL 策略倾向于将同一数据路径上的宏布局得靠近彼此，这导致了更好的时序性能。Edge-GNN 编码器通过迭代地平均并应用非线性变换到节点的 k 跳邻近节点和边上，来嵌入每个节点的特征，其中 k 是应用的迭代次数。因此，一个假设是，给定数据路径中节点的表示彼此相似，这使得我们的策略网络对它们应该在画布上的布局位置生成类似的预测。这自然导致同一数据路径中的节点被布局得靠近彼此，从而改善时序性能。

另一个观察是，我们的策略学会了为后续标准单元的布局预留足够的空间，因为这有效地优化了它的报酬函数。即使在零样本的情况下（意味着我们在不到一秒钟的时间内对策略网络进行推理），我们的方法已经表现出这种行为，如扩展数据图 4 所示。

扩展数据图 5 将一个人类物理设计师生成的布局（左侧）与我们的方法生成的布局（右侧）进行了对比，针对的是一个最近的 TPU 模块。白色区域显示了宏的布局，绿色区域显示了标准单元的布局。我们的方法创建了围绕标准单元的甜甜圈形状的宏布局，这导致了总线长的减少。这些布局图像经过模糊处理以保护机密性。

# 与模拟退火的比较

在正文部分，我们将我们的方法与两个基准进行了比较：学术领域最先进的 RePlAce 方法以及人类专家的布局方法。在本节中，我们将提供与模拟退火（SA）的额外比较。为了生成我们方法的结果，我们采用与表 1 相同的流程，即在最大数据集（20 个 TPU 模块）上对策略进行预训练，然后在同样的五个未见过的测试模块上对其进行微调。

模拟退火是一种强大但运行缓慢的优化方法。然而，与强化学习类似，模拟退火能够优化任意不可微的代价函数。为了展示强化学习的相对样本效率，我们进行了一些实验，在这些实验中我们用模拟退火优化器替代了强化学习。

我们的模拟退火算法如下运行：在每次模拟退火迭代（步骤）中，我们执行 2N 个宏单元动作（其中 N 为宏单元的数量）。一个宏单元动作有以下三种形式之一：交换、移动和镜像。交换操作会随机选择两个宏单元并在可行的情况下交换它们的位置。移动操作会随机选择一个宏单元并将其移动到相邻位置（左、右、上或下）。镜像操作会随机选择一个宏单元并使其在 x 轴、y 轴或 x 轴和 y 轴上进行翻转。我们对这三种移动类型赋予相同的概率，这意味着在每个时间步，有 1/3 的概率进行交换操作，1/3 的概率进行移动操作，1/3 的概率进行镜像操作。在执行 N 个宏单元动作之后，我们使用

有限差分（FD）方法来布局标准单元集群，同时保持宏单元位置固定，就如同我们在强化学习方法中所做的那样。

对于每个宏单元动作或有限差分动作，如果新状态导致代价降低，则接受该新状态。否则，以概率 $\exp[(\text{prevcost} - \text{newcost})/t]$ 接受新状态，其中 $t = t_{max}$ 。$\exp\{-\log[(t_{max}/t_{min})(\text{step}/\text{steps})]\}$。这里，前一成本指的是前一次迭代的成本；当前成本指的是当前迭代的成本；t 是温度，它控制着算法接受局部性能下降以进行探索的意愿；$t_{max}$ 和 $t_{min}$ 分别对应允许的最大温度和最小温度。

为了使比较公平，我们进行了 80 次模拟退火实验，遍历了不同的超参数，包括最大温度（$\{10^{-5}, 3\times10^{-5}, 5\times10^{-5}, 7\times10^{-5}, 10^{-4}, 2\times10^{-4}, 5\times10^{-4}, 10^{-3}\}$）、最大模拟退火轮次长度（$\{5\times10^{4}, 10^{5}\}$）和随机种子（五个不同的随机种子），并在扩展数据表 6 中报告了以近似线长和拥塞成本衡量的最佳结果。80 个模拟退火实验中的每一个都对应着五个随机种子、两种轮次长度和八个最大温度的某一种特定选择所进行的一次实验。

模拟退火（SA）基线使用的计算资源比我们的方法多（80 个 SA 工作节点 × 每个 SA 工作节点 2 个 CPU × 18 小时的运行时间 = 2880 个 CPU 小时），而我们的方法使用的是（16 个 RL 工作节点 ×（1 个 GPU + 每个 RL 工作节点 10 个 CPU） × 6 小时 = 1920 个 CPU 小时）。在这里，我们将一个 GPU 的成本视为一个 CPU 的大约 10 倍。如果我们在 12 小时后停止了 SA（并且我们没有在 RL 中使用提前停止），那么这两种方法将使用等量的计算资源，但是在 12 小时后 SA 的结果甚至远未达到竞争水平。即使有更多的时间（SA 的 18 小时对比 RL 的 6 小时），SA 在产生高质量的布局方面仍然比我们的方法困难，平均产生了比我们的方法高出 14.4%的线长和高出 24.1%的拥堵。

# 对于更广泛类别问题的影响

我们认为，所提出的方法对芯片设计的其他阶段和其他布局优化任务具有更广泛的影响。例如，我们的零样本模式允许通过基于物理现实的快速评估来进行计算机架构的设计空间探索。自动化和优化架构探索及其与物理设计的接口不仅能够进一步加速芯片设计过程，还能在关键硬件指标（如功耗和时序）上带来额外的改进。

此外，这种方法适用于芯片设计之外的一系列布局优化问题，比如城市规划（如交通灯的布置）、编译器优化（如数据中心资源分配）和环境工程（如水坝的选址）。

# 相关工作

芯片平面规划是一个长期的挑战，需要在日益复杂的电路上进行多目标优化。自 20 世纪 60 年代以来，已经提出了许多方法，迄今为止可以分为三大类：（1）基于分区的方法，（2）随机/爬山方法和（3）解析求解器。

从 20 世纪 60 年代开始，工业和学术实验室使用基于分区[3,4,50]和电阻网络[51,52]的方法来进行芯片平面规划，以及基于电阻网络的方法[51,52]。这些方法的特点是分而治之的方法；网络列表和芯片画布被递归地分割，直到出现足够小的子问题，此时使用最优求解器将子网络列表布局到子区域。这种方法执行速度相当快，它们的层次结构使它们能够扩展到任意大的网络列表。然而，通过孤立地优化每个子问题，基于分区的方法牺牲了全局解决方案的质量，特别是路由拥堵。此外，早期分区不佳可能导致最终布局无法挽救。

在 20 世纪 80 年代，解析方法出现了，但很快就被随机/爬山算法所取代，特别是模拟退火（SA）[6-8]。SA 的命名是因为它类似于冶金学，即金属首先被加热，然后逐渐冷却以诱导或退火，形成能量最优的晶体表面。SA 对给定的布局（例如，宏的移动、交换或镜像）应用随机扰动，然后测量它们对目标函数（例如，HPWL）的影响。如果扰动是改进，它就被应用；如果不是，它仍然以一定的概率被应用，这被称为温度。温度被初始化为一个特定的值，然后逐渐退火到一个较低的值。尽管 SA 能够产生高质量的解决方案，但它非常慢，难以并行化，因此无法扩展到 90 年代及以后日益庞大和复杂的电路。

1990 年代至 2000 年代的特点是多层次分区方法[5,53]，以及解析技术的复兴，如力导向（FD）方法[9-11,25-27]和非线性优化器[55-58]。二次方法的重新成功部分是由于算法的进步，但也是由于现代电路的大规模（10-1 亿个节点），这证明了将布局问题近似为布局零面积节点的问题。然而，尽管二次方法具有计算效率，它们通常不如非线性对应物可靠，并且产生的解决方案质量较低。

非线性优化使用平滑的数学函数来近似成本，例如用于线长的 log-sum-exp[59]和加权平均[60,61]模型，以及用于密度的高斯[62]和亥姆霍兹模型。然后，这些函数通过拉格朗日惩罚或松弛组合成一个单一的目标函数。由于这些模型的复杂性更高，因此需要采取层次化的方法，布局簇而不是单个节点，这种近似会降低布局的质量。

在过去的十年中，现代解析技术有所兴起，包括更先进的二次方法[12,44,63-65]，以及最近基于静电力的方法，如 ePlace[13]和 RePlAce[14]。ePlace[13]将网络列表布局建模为一个静电力系统，提出了密度惩罚的新公式，其中网络列表的每个节点（宏或标准单元）类似于一个正电荷粒子，其面积对应于其电荷量。在这种设置中，节点之间以与它们的电荷（面积）成比例的力相互排斥，密度函数和梯度对应于系统的势能。基于这种静电力方法的变体已被提出，以解决标准单元布局[13]和混合尺寸布局[60,66]问题。RePlAce[14]是一种最新的混合尺寸布局技术，它通过引入一个局部密度函数来为每个单独的箱大小定制惩罚因子，进一步优化了 ePlace 的密度函数。DREAMPlace[67]通过采用基于深度学习的方法来优化布局，并利用 GPU 加速，进一步加快了 RePlAce 的速度。然而，DREAMPlace 的重点是标准单元布局优化，而不是宏布局，报告的质量与 RePlAce 相当。因此，我们将我们的方法的性能与 RePlAce 进行比较。

机器学习在推进物理设计方面有许多机会[68-70]。最近的工作[71]提出了训练一个模型来预测给定宏布局的 DRC 违规数量。DRC 是确保布局和布线的网络列表符合带出要求的规则。为了生成更少 DRC 违规的宏布局，参考文献[71]使用这个训练有素的模型的预测作为模拟退火中的评估函数。尽管这项工作代表了有趣的方向，但它报告的结果是在不超过六个宏的网络列表上，远远少于任何现代块，并且没有考虑布局和布线优化的影响，这可能会显著改变 DRC 违规的数量。此外，尽管遵守 DRC 标准是一个必要条件，但宏布局的主要目标是优化线长、时序（例如，WNS 和 TNS）、功率和面积，而这项工作甚至没有考虑这些指标。

为了解决这个经典问题，我们提出了一种新的方法类别：端到端基于学习的方法。这种新方法与解析求解器最为接近，特别是非线性的，因为所有这些方法都通过梯度更新来优化目标函数。然而，我们的方法与以往方法的不同之处在于其能够从过去的经验中学习，为新芯片生成更高质量的布局。与现有方法不同，它们为每个新芯片从头开始优化布局，我们的工作利用了从先前布局芯片中获得的知识，随着时间的推移而变得更好。此外，我们的方法能够直接优化目标指标，如线长、密度和拥堵，而无需像其他方法[13,14]那样定义这些函数的凸近似。我们的公式不仅可以轻松地纳入新的成本函数，还可以根据特定芯片块的需求（例如，时序关键或功率受限）来权衡它们的相对重要性。

领域适应是训练能够在多个经验中学习并将获得的知识转移到新未见样本上以更好执行任务的策略问题。在芯片平面规划的背景下，领域适应涉及在一组芯片网络列表上训练策略，然后将该训练策略应用于新的未见网络列表。日益增长的兴趣领域，包括解决旅行推销员问题[21,72]、神经架构搜索[73]和模型并行性[22,23]的方法。最近，还有关于编译器优化[46,74-76]和最大割问题[77]的领域适应的工作。我们的方法不仅利用过去的经验来减少训练时间，而且在接触到更多问题实例时还能产生更高质量的结果。据我们所知，我们的方法是在生产中用于解决组合优化问题的第一个深度强化学习方法，即在最新一代 Google TPU 的设计中。

## 数据可用性

本研究支持其发现的数据可以在论文和扩展数据中找到。

## 代码可用性

用于生成这些数据的代码可以在以下 GitHub 仓库中找到：
https://github.com/google-research/circuit_training。

# 参考文献

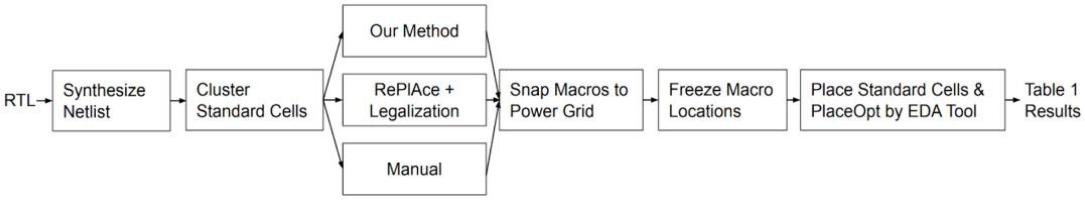
1. Markov, I. L., Hu, J. & Kim, M. Progress and challenges in VLSI placement research. Proc. IEEE 103, 1985–2003 (2015).
2. Tang, M. & Yao, X. A memetic algorithm for VLSI floorplanning. IEEE Trans. Syst. Man Cybern. B 37, 62–69 (2007).
3. Breuer, M. A. A class of min-cut placement algorithms. In Proc. 14th Design Automation Conference (DAC 1977) 284–290 (IEEE, 1977).
4. Fiduccia, C. M. & Mattheyses, R. M. A linear-time heuristic for improving network partitions. In 19th Design Automation Conference 175–181 (IEEE, 1982).
5. Roy, J. A., Papa, D. A. & Markov, I. L. in Modern Circuit Placement (eds Nam, G.-J. & Cong, J. J.) 97–133 (Springer, 2007).
6. Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. Science 220, 671–680 (1983).
7. Sechen, C. M. & Sangiovanni-Vincentelli, A. L. TimberWolf3.2: a new standard cell placement and global routing package. In 23rd ACM/IEEE Design Automation Conference 432–439 (IEEE, 1986).
8. Sarrafzadeh, M., Wang, M. & Yang, X. in Modern Placement Techniques 57–89 (Springer, 2003).
9. Luo, T. & Pan, D. Z. DPlace2.0: a stable and efficient analytical placement based on diffusion. In 2008 Asia and South Pacific Design Automation Conference 346–351 (IEEE, 2008).
10. Hu, B. & Marek-Sadowska, M. Multilevel fixed-point-addition-based VLSI placement. IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 24, 1188–1203 (2005).
11. Viswanathan, N., Pan, M. & Chu, C. in Modern Circuit Placement (eds Nam, G.-J. & Cong, J. J.) 193–228 (Springer, 2007).
12. Kim, M., Lee, D., Markov, I. L. & Sim, P. L. An effective placement algorithm. IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 31, 50–60 (2012).
13. Lu, J. et al. ePlace: electrostatics-based placement using fast Fourier transform and Nesterov's Method. ACM Trans. Des. Autom. Electron. Syst. 20, 17 (2015).
14. Cheng, C.-K., Kahng, A. B., Kang, I. & Wang, L. RePlAce: advancing solution quality and routability validation in global placement. IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 38, 1717–1730 (2019).
15. Silver, D. et al. Mastering the game of Go without human knowledge. Nature 550, 354–359 (2017).
16. Aslam, B., Amjad, F. & Zou, C. C. Optimal roadside units placement in urban areas for vehicular networks. In 2012 IEEE Symposium on Computers and Communications (ISCC) 000423–000429 (IEEE, 2012).
17. Medlock, J. & Galvani, A. P. Optimizing influenza vaccine distribution. Science 325, 1705–1708 (2009).
18. Cherniak, C., Mokhtarzada, Z., Rodriguez-Esteban, R. & Changizi, K. Global optimization of cerebral cortex layout. Proc. Natl Acad. Sci. USA 101, 1081–1086 (2004).
19. Langford, J. & Zhang, T. The Epoch-Greedy algorithm for multi-armed bandits with side

information. In Advances in Neural Information Processing Systems Vol. 20, 817–824 (2008).
20. Usunier, N., Synnaeve, G., Lin, Z. & Chintala, S. Episodic exploration for deep deterministic policies: an application to starcraft micromanagement tasks. In Proc. International Conference on Learning Representations (2017).
21. Bello, I., Pham, H., Le, Q. V., Norouzi, M. & Bengio, S. Neural combinatorial optimization with reinforcement learning. Preprint at https://arxiv.org/abs/1611.09940 (2016).
22. Mirhoseini, A. et al. Device placement optimization with reinforcement learning. In Proc. International Conference on Machine Learning 2430–2439 (PMLR, 2017).
23. Mirhoseini, A. et al. A hierarchical model for device placement. In Proc. International Conference on Learning Representations (2018).
24. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. Preprint at https://arxiv.org/abs/1707.06347 (2017).
25. Obermeier, B., Ranke, H. & Johannes, F. M. Kraftwerk: a versatile placement approach. In Proc. 2005 International Symposium on Physical Design 242–244 (ACM, 2005).
26. Spindler, P., Schlichtmann, U. & Johannes, F. M. Kraftwerk2 – a fast force-directed quadratic placement approach using an accurate net model. IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 27, 1398–1411 (2008).
27. Viswanathan, N. et al. RQL: global placement via relaxed quadratic spreading and linearization. In Proc. Design Automation Conference 453–458 (ACM/IEEE, 2007).
28. Ioffe, S. & Szegedy, C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In Proc. 32nd International Conference on Machine Learning 448–456 (JMLR, 2015).
29. Nair, V. & Hinton, G. E. Rectified linear units improve restricted Boltzmann machines. In Proc. International Conference on Machine Learning 807–814 (Omnipress, 2010).
30. Zaruba, F. & Benini, L. The cost of application-class processing: energy and performance analysis of a Linux-ready 1.7-GHz 64-Bit RISC-V core in 22-nm FDSOI technology. IEEE Trans. Very Large Scale Integr. VLSI Syst. 27, 2629–2640 (2019).
31. RePlAce software – the OpenROAD project https://github.com/The-OpenROAD-Project/RePlAce (2020).
32. Karypis, G. & Kumar, V. Hmetis: a hypergraph partitioning package http://glaros.dtc.umn.edu/gkhome/metis/hmetis/overview (1998).
33. Alpert, C. J., Hagen, L. W. & Kahng, A. B. A hybrid multilevel/genetic approach for circuit partitioning. In Proc. APCCAS'96 – Asia Pacific Conference on 1012 Circuits and Systems 298–301 (IEEE, 1996).
34. Caldwell, A. E., Kahng, A. B. & Markov, I. L. Improved algorithms for hypergraph 1014 bipartitioning. In Proc. 2000 Design Automation Conference 661–666 (IEEE, 2000).
35. Chen, H. et al. An algebraic multigrid solver for analytical placement with layout 1017 based clustering. In Proc. 40th annual Design Automation Conference 794–799 (ACM, 2003); 10.1145/775832.776034.
36. Alpert, C., Kahng, A., Nam, G.-J., Reda, S. & Villarrubia, P. A semi-persistent clustering technique for vlsi circuit placement. In Proc. 2005 International Symposium on Physical Design, 200–207 (ACM, 2005).

37. Fogaça, M., Kahng, A. B., Reis, R. & Wang, L. Finding placement-relevant clusters with fast modularity-based clustering. In Proc. 24th Asia and South Pacific Design Automation Conference 569–576 (ACM, 2019); https://doi.org/10.1145/3287624.3287676.
38. Fogaça, M. et al. On the superiority of modularity-based clustering for deter mining placement-relevant clusters. Integration 74, 32–44 (2020).
39. Kahng, A. B. Futures for partitioning in physical design (tutorial). In Proc. 1998 International Symposium on Physical Design 190–193 (ACM, 1998); https://doi.org/10.1145/274535.274563.
40. Shahookar, K. & Mazumder, P. VLSI cell placement techniques. ACM Comput. Surv. 23, 143–220 (1991).
41. Caldwell, A. E., Kahng, A. B., Mantik, S., Markov, I. L. & Zelikovsky, A. On wirelength estimations for row-based placement. IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 18, 1265–1278 (1999).
42. Kahng, A. B. & Xu, X. Accurate pseudo-constructive wirelength and congestion estimation. In Proc. 2003 International Workshop on System-Level Interconnect Prediction 61–68 (ACM, 2003); https://doi.org/10.1145/639929.639942.
43. Kahng, A. B. & Reda, S. A tale of two nets: studies of wirelength progression in physical design. In Proc. 2006 International Workshop on System-Level Interconnect Prediction |17–24 (ACM, 2006); https://doi.org/10.1145/1117278.1117282.
44. Kim, M.-C., Viswanathan, N., Alpert, C. J., Markov, I. L. & Ramji, S. MAPLE: Multilevel Adaptive Placement for Mixed-Size Designs. In Proc. 2012 ACM International Symposium on International Symposium on Physical Design 193–200 (ACM, 2012).
45. Nazi, A., Hang, W., Goldie, A., Ravi, S. & Mirhoseini, A. GAP: generalizable approximate graph partitioning framework. In International Conference on Learning Representations Workshop (2019).
46. Zhou, Y. et al. GDP: generalized device placement for dataflow graphs. Preprint at https://arxiv.org/abs/1910.01578 (2019).
47. Zhang, M. & Chen, Y. Link prediction based on graph neural networks. In Proc. International Conference on Neural Information Processing 5171–5181 (Curran Associates Inc., 2018).
48. Xie, Z. et al. RouteNet: routability prediction for mixed-size designs using convolutional neural network. In 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 1–8 (IEEE, 2018).
49. hMETIS – hypergraph and circuit partitioning manual http://glaros.dtc.umn.edu/gkhome/metis/hmetis/download.
50. Dunlop, A. E. & Kernighan, B. W. A procedure for placement of standard-cell VLSI circuits. IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 4, 92–98 (1985).
51. Cheng C. K. & Kuh. E. S. Module placement based on resistive network optimization. IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 3, 218–225 (1984).
52. Tsay, R.-S., Kuh, E. & Hsu, C.-P. Proud: a fast sea-of-gates placement algorithm. In Proc. Design Automation Conference 1988, 318–323 (IEEE, 1988).
53. Agnihotri, A., Ono, S. & Madden, P. Recursive bisection placement: Feng Shui 5.0 implementation details. In Proc. International Symposium on Physical Design 230–232 (ACM, 2005).

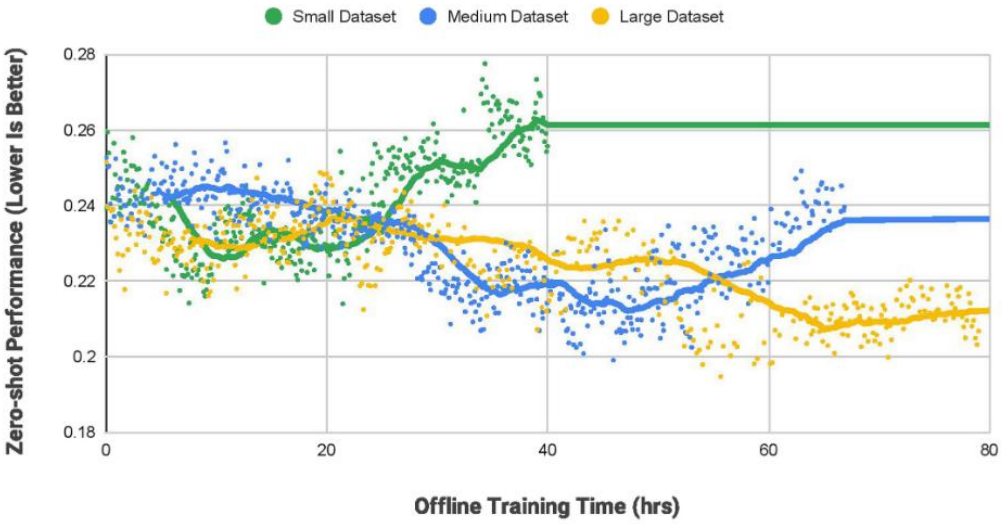
54. Alpert, C. et al. Analytical engines are unnecessary in top-down partitioning based placement. VLSI Des. 10, 99–116 (2002).
55. Kahng, A. B., Reda, S. & Wang, Q. Architecture and details of a high quality, large-scale analytical placer. In IEEE/ACM International Conference on Computer-Aided Design 2005 891–898 (IEEE, 2005).
56. Kahng, A. B. & Wang, Q. An analytic placer for mixed-size placement and timing.driven placement. In IEEE/ACM International Conference on Computer Aided Design 2004 565–572 (IEEE, 2004).
57. Kahng, A. B. & Wang, Q. Implementation and extensibility of an analytic placer. IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 24, 734–747 (2005).
58. Chen, T.-C., Jiang, Z.-W., Hsu, T.-C., Chen, H.-C. & Chang, Y.-W. A High-quality mixed-size analytical placer considering preplaced blocks and density constraints. In Proc. 2006 IEEE/ACM International Conference on Computer-Aided Design 187–192 (ACM, 2006).
59. Naylor, W., Donelly, R. & Sha, L. Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer. US Patent US6301693B1 (2001).
60. Lu, J., Zhuang, H., Kang, I., Chen, P. & Cheng, C.-K. Eplace-3d: electrostatics based placement for 3d-ics. In International Symposium on Physical Design 11–18 (ACM, 2016).
61. Hsu, M.-K., Chang, Y.-W. & Balabanov, V. TSV-aware analytical placement for 3D IC designs. In Proc. 48th Annual Design Automation Conference 664–669 (ACM, 2011).
62. Chen, T., Jiang, Z., Hsu, T., Chen, H. & Chang, Y. NTUplace3: an analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 27, 1228–1240 (2008).
63. Kim, M.-C. & Markov, I. L. ComPLx: a competitive primal-dual Lagrange optimization for global placement. In Design Automation Conference 2012 747– 755 (ACM, 2012).
64. Brenner, U., Struzyna, M. & Vygen, J. BonnPlace: placement of leading-edge chips by advanced combinatorial algorithms. Trans. Comp.-Aided Des. Integ.Cir. Sys. 27, 1607–1620 (2008).
65. Lin, T., Chu, C., Shinnerl, J. R., Bustany, I. & Nedelchev, I. POLAR: placement based on novel rough legalization and refinement. In Proc. International Conference on Computer-Aided Design 357–362 (IEEE, 2013).
66. Lu, J. et al. ePlace-MS: electrostatics-based placement for mixed-size circuits. IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 34, 685–698 (2015).
67. Lin, Y. et al. DREAMPlace: deep learning toolkit-enabled GPU acceleration for modern VLSI placement. In Design Automation Conference 1–6 (ACM/IEEE, 2019).
68. Kahng, A. B. Machine learning applications in physical design: recent results and directions. In Proc. 2018 International Symposium on Physical Design 68–73 (ACM, 2018); https://doi.org/10.1145/3177540.3177554.
69. Kahng, A. B. Reducing time and effort in ic implementation: a roadmap of challenges and solutions. In Proc. 55th Annual Design Automation Conference (ACM, 2018); https://doi.org/10.1145/3195970.3199854.
70. Ajayi, T. et al. Toward an open-source digital flow: first learnings from the openroad project. In Proc. 56th Annual Design Automation Conference 2019 (ACM, 2019); https://doi.org/10.1145/3316781.3326334.

71. Huang, Y. et al. Routability-driven macro placement with embedded CNN-based prediction model. In Design, Automation & Test in Europe Conference & Exhibition 180–185 (IEEE, 2019).
72. Khalil, E., Dai, H., Zhang, Y., Dilkina, B. & Song, L. Learning combinatorial optimization algorithms over graphs. Adv. Neural Inf. Process Syst. 30, 6348–6358 (2017).
73. Zoph, B. & Le, Q. V. Neural architecture search with reinforcement learning. In Proc. International Conference on Learning Representations (2017).
74. Addanki, R., Venkatakrishnan, S. B., Gupta, S., Mao, H. & Alizadeh, M. Learning generalizable device placement algorithms for distributed machine learning. Adv. Neural Inf. Process Syst. 32, 3981–3991 (2019).
75. Paliwal, A. et al. Reinforced genetic algorithm learning for optimizing computation graphs. In Proc. International Conference on Learning Representations (2020).
76. Zhou, Y. et al. Transferable graph optimizers for ML compilers. Prepint at https://arxiv.org/abs/2010.12438 (2021).
77. Barrett, T. D., Clements, W. R., Foerster, J. N. & Lvovsky, A. I. Exploratory combinatorial optimization with reinforcement learning. Preprint at https://arxiv.org/abs/1909.04063 (2020).
78. Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks. Prepint at https://arxiv.org/abs/1609.02907 (2016).

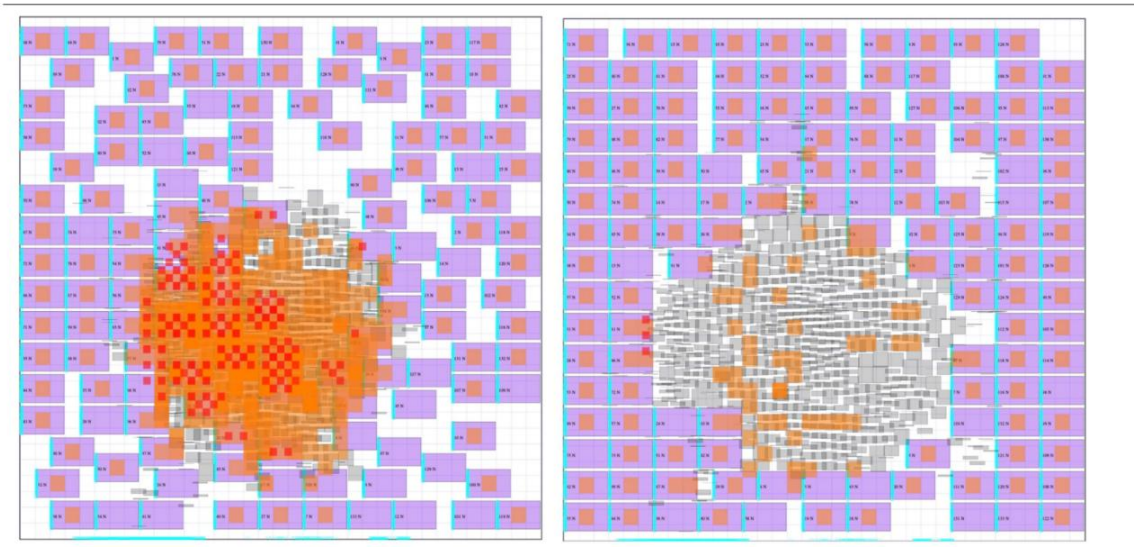



扩展数据图 1 |产生表 1 结果的评估工作流程。我们允许每个方法访问同一个聚类的网表超图。我们在所有的方法中使用相同的超参数(在可能的范围内)。在每个方法(这包括 RePlAce 的合法化步骤)完成布局后，我们将宏抓取到电网，冻结宏位置，并使用商业 EDA 工具布局标准单元并报告最终结果。

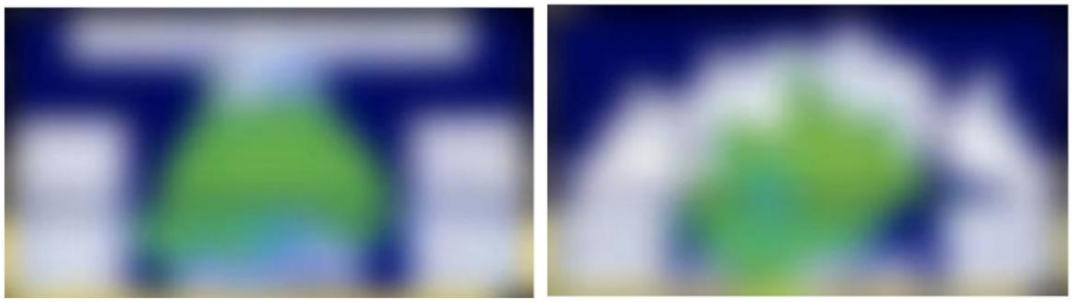

扩展数据图 2 | Edge-GNN 与 GCN（图卷积网络）78 的零样本性能。具有 Edge-GNN 架构的代理对过度拟合更加稳健并产生更高质量的结果，这是通过扩展数据图 1 所示的测试块的平均零样本性能来衡量的。




扩展数据图 3 | 泛化性能与预训练数据集大小的关系。我们在三个不同的训练数据集上对策略网络进行了预训练（具有 2 个块的小型数据集是具有 5 个块的中型数据集的子集，中型数据集是具有 20 个块的大型数据集的子集）。对于每个策略，我们在预训练期间的各个快照中报告其在未见过的测试块上的推理性能。随着数据集大小的增加，测试块上生成的位置的质量和策略的泛化性能都会提高。在最大数据集上训练的策略对过度拟合最有鲁棒性。



扩展数据图 4 | Ariane 布局的可视化。左侧为预训练策略的零样本布局；右侧为微调策略的布局。零样本布局是在推理时在之前未见过的芯片上生成的。预训练策略网络（未进行微调）在画布中心保留了一个凸包，标准单元可以布局在其中，这种行为可以减少线长，并且只有在从头开始训练的策略经过数小时的微调后才会出现。



扩展数据图 5 |实际 TPU 芯片的可视化。人类专家的位置显示在左边，我们的方法的结果显示在右边。白色区域表示宏，绿色区域表示标准单元格。这些数字是故意模糊的，因为这些设计是专有的。人类专家设计的波长为 57.07 m，而我们的波长为 55.42 m。此外，我们的方法在 6 小时内达到这些结果，而手动基线需要几个星期。

**扩展数据表 1 |用于微调 RL 代理的超参数**

| 超参数 | 值 |
|---|---|
| 学习率 | 1.00E-04 |
| 优化器 | Adam |
| Epoch 数量 | 4 |
| GPU 数量 | 16 |
| 每个工作 GPU 的 CPU 数量 | 10 |
| 每个 GPU 的批处理大小 | 64 |
| 有效的批处理大小<br>(GPU 数量*每个 GPU 的批处理大小) | 1024 |
| 梯度裁剪 | 1 |
| 每个 GPU 的角色数量 | 32 |
| 权重初始化器 | Xavier |
| 每次滚动的轮次数量 | 2 |
| PPO 损失： | |
| 参数裁剪 epsilon | 0.2 |
| 价值系数 | 0.5 |
| 熵系数 | 0.01 |
| 折扣 gamma | 1 |

**扩展数据表 2 | 用于布局标准单元簇的 FD 算法的超参数**

| 超参数 | 值 | 描述 |
|---|---|---|
| 调度数量 | 3 | 运行力导向算法时的调度数量 |
| 步骤 | [100,100,100] | 在每个调度中，力导向算法的步骤数量 |
| 移动距离因子 | [1.0,1.0,1.0] | 在力导向算法的单一步骤中，节点相对于画布大小可以移动的最大距离 |
| 吸引因子 | [100,1e-3,1e-5] | 在力导向算法中，两个相连节点之间的弹簧常数 |
| 排斥因子 | [0,1e6,1e7] | 在力导向算法中，用于分散节点以避免拥堵的排斥因子 |
| I/O 因子 | [1.0,1.0,1.0] | 在力导向算法中，用于将 I/O 端口的力乘以到节点的 I/O 因子 |

**扩展数据表 3 | 使用 hMETIS[32] 生成标准单元簇的超参数**

| 名称 | 值 | 定义 |
|---|---|---|
| UBfactor | 5 | 允许的不平衡分区的程度 |
| Nruns | 10 | hMETIS 执行的二分次数，其中最佳结果作为最终解决方案返回 |
| CType | 5 | Edge Coarsening (EC)算法(重边最大匹配)，在此模式下，如果顶点由多个超边连接，则将它们成对分组 |
| RType | 3 | Early-Exit Fiduccia-Mattheyses refinement scheme(FM-EE)算法，在此模式下，如果解决方案的质量在相对少量的顶点移动后没有改善，则中止 FM 迭代 |
| Vcycle | 3 | 对每个中间解决方案执行 Vcycle 细化，意味着使用 Vcycles 对 Nruns 二分中的每一个也进行细化 |

**扩展数据表 4 | 不同成本权衡对表 1 中块 1 布局后性能的影响**

| 拥堵权重 | WNS（皮秒） | TNS (纳秒) | 线长（米） | 水平拥堵（%） | 垂直拥堵（%） |
|---|---|---|---|---|---|
| 0.01 | 163 | 22.85 | 48190736 | 0.30 | 0.03 |
| 0.1 | 154 | 11.80 | 50841227 | 0.06 | 0.03 |
| 1.0 | 118 | 34.73 | 53153141 | 0.07 | 0.02 |

正如预期，增加拥堵权重可以改善水平和垂直拥堵，直到某一点，但由于这两个度量之间的固有权衡，会导致线长退化。

**扩展数据表 5 | 结果对随机种子选择的敏感性，以 Ariane RISC-V 块为基准进行测量**

| 种子 | 代理线长 | 代理拥堵 | 代理密度 |
|---|---|---|---|
| 111 | 0.1187 | 0.9856 | 0.5780 |
| 222 | 0.1237 | 1.0251 | 0.5691 |
| 333 | 0.1207 | 0.9456 | 0.5714 |
| 444 | 0.1189 | 0.9559 | 0.5681 |
| 555 | 0.1174 | 0.9168 | 0.5561 |
| 666 | 0.1187 | 0.9676 | 0.5815 |
| 777 | 0.1200 | 0.9693 | 0.5772 |
| 888 | 0.1199 | 1.0087 | 0.5819 |
| 平均 | 0.1198 | 0.9718 | 0.5729 |
| 标准差 | 0.0019 | 0.0346 | 0.0086 |

我们观察到在所有三个成本函数中对随机种子的敏感性很小，尽管线长和密度的变化低于拥堵。这八次运行的总体成本标准差为 0.0022。

**扩展数据表 6 | 我们的方法与 SA（模拟退火算法）的性能比较**

| | 替换我们框架中的深度学习为模拟退火 | | 我们的 | |
|---|---|---|---|---|
| | 代理线长 | 代理拥堵 | 代理线长 | 代理拥堵 |
| 块 1 | 0.048 | 1.21 | 0.047 | 0.87 |
| 块 2 | 0.045 | 1.11 | 0.041 | 0.93 |
| 块 3 | 0.044 | 1.14 | 0.034 | 0.96 |
| 块 4 | 0.030 | 0.87 | 0.024 | 0.78 |
| 块 5 | 0.045 | 1.29 | 0.038 | 0.88 |

表中为每个块的代理线长和拥堵。我们注意到，因为这些代理度量是相对的，所以比较仅对同一块的不同布局有效。即使额外花费时间（SA 18 小时，RL 6 小时），SA 平均生成线长高出 14.4%，拥堵高出 24.1%。

# 国外对阿里云通义千问 Qwen2.5 大模型的评议

COPU

最近COPU收集了阿里云通义千问Qwen2.5大模型在国外测试和人们的反应：

阿里云的通义千问 Qwen2.5 大模型在几个知名的基准测试平台: Chatbot Arena、Open Compass 的测试中取得了好成绩,，跻身于全球先进的大模型之列。在基准测试平台 Chatbot Arena 公布的盲测榜单上, Qwen2.5 闯入全球 10 强,在权威基准 Open Compass 上,得分追平 GPT-4 Turbo，该基准首次录得国产大模型取得如此出色的成绩。

通义千问 Qwen2.5 大模型在国外获得了广泛的好评：一位外国工程师点赞 Qwen2.5，直言相比 Mistral-Small-Instruct-2409,通义大模型在函数调用评估中有更好的表现；在日本也有不少开发者追捧 Qwen2.5，表示实测好用；还有一位美国开发者，认为中国大模型真厉害，他们拿 Qwen2.5 和 GPT-4o、Claude3.5 做对比，发现通义千问 Qwen2.5 的性能，介于二者之间，进步太疯狂了！据说目前通义开源模型累计访问下载量已突破 4000 万，再次证明千问很受开发者欢迎。

当下全球开源大模型产品各有特色，都很好用，而千问 Qwen 大模型真不比美国先进的大模型差。

还有一事也应提出：2024 年 6 月 25 日，OpenAI 邮件通知，以极端的意识形态划线，自 7 月起终止向中国提供语言大模型 LLM 的 API 服务（即封杀中国等四国)，立即遭到了谷歌、Meta 的抨击，说这是一个愚蠢的举措：他们指出，OpenAI 一声吆喝，惊起了一滩鸥鹭，一夜之间，一批中国优秀的大模型企业(包括阿里云在内)通过自主开发的大模型(包括通义千问 Qwen2.5 在内)接替因 Open AI 关停 API 接口完成对标和平替。

# AI 时代 Open Source（开源）新定义

COPU 陆首群

OSI 发布在 AI 时代 Open Source (开源) 的定义与 IT 时代的定义比有所变化。

AI 时代对 Open Source (开源) 定义的要求:

提供训练数据详细信息，完整构建和运行 AI 的代码，以及训练时的设置和权重。

在网上，有人把 Meta、谷歌开发的 Llama 和 Gemini 所谓开源大模型进行打假(“假开源”)，颇为耸人听闻！但事实上这些开源大模型遵循的开源定义已经落后了，没有完整地开放源代码和数据集。我们现在实行的开源也都是 IT 时代的开源定义，今天我们再次讨论“AIOS”，也要采用 AI 时代的开源新定义。

要求 COPU 秘书处，联系 OSI，获取 AI 时代 Open Source (开源）的新定义及有关说明，随后告知 COPU 成员单位和相关联系单位。

几个月前在 COPU 例会上，我们赞成荷兰学者提问时指出，进入 AI 时代，开源的开放程度将进一步扩大(当时提出 OSI 也正在修改开源的定义)，以增加利益相关的人们的知情度。今天 OSI 颁布 AI 时代开源的新定义，证实了这一点！至于在 AI 时代实行的闭源是不开放的，它无法为人们提供知情度，所以闭源的 AI 如何能发展下去是一个问题。

# 红帽人工智能操作系统

## Red Hat 的下一个前沿：在人工智能领域执行开源的新定义，拥抱开源价值观

Red Hat 大中华区首席架构师　张家驹

### 开源人工智能的产业需求

随着生成式人工智能 (GenAI) 技术的发展，人工智能又迎来了新的春天——GenAI 能够处理大量数据集、生成类似人类产生的文本并在各个领域创建新内容，千行百业都在尝试用该项技术升级自己的业务流程、降本增效、加速创新，GenAI 已经成为促进行业发展的新的动力引擎。然而，如何开发、部署和使用这项技术则是众说纷纭：有的倾向于直接采用公有云上提供的 AI 服务；有的倾向于调用公有云上 AI 模型的 API，实现自己的定制化应用；也有些用户，特别是企业用户，考虑到安全性、透明度和道德治理等方面的因素，选择可本地部署的开源大模型。

### 安全与信任

部署 GenAI 时，安全性是一个关键问题，因为这些模型在训练和推理过程中会处理大量敏感数据。与传统软件不同，GenAI 模型是包含算法和训练数据的复杂系统，这会带来数据泄露、滥用或嵌入偏差的风险。开源 GenAI 不仅为模型代码提供透明度，还为所使用的数据集和训练过程提供透明度，从而有助于降低这些风险。因此为检查、审计和解决潜在的漏洞提供便捷，确保模型不会无意中泄露敏感信息或表现出非预期行为。一份 Linux 基金会的调查报告【1】发现，专有 GenAI 解决方案与开源替代方案相比，并没有明显的安全优势。

开源 GenAI 允许使用者和更广泛的社区一起，不仅分析代码，还可以分析训练数据和模型架构，从而识别潜在的安全风险。这种集体监督有助于确保 GenAI 模型保持安全，同时保持决策过程的透明度——而使用限制对模型内部进行访问的专有系统则很难实现这一点。

## 透明度和可访问性

透明度对于人工智能至关重要，因为它可以让使用者了解模型如何做出决策。开源 GenAI 提高了数据控制和透明度，因为它提供了对模型内部工作原理的访问，而专有模型通常会隐藏这些工作原理。开放系统有助于识别和减轻偏见和错误，从而有助于实现更符合道德规范的人工智能。有报告【2】显示，69% 的受访者认为使用开源人工智能可以提高数据控制和透明度。

因此，开源 AI 通过提供模型架构、算法和数据集的可访问性来提高透明度。这种开放性使结果的独立验证、可审计和可重复性成为可能，从而确保企业和用户可以信任 AI 的决策过程。

## 中立治理

中立的治理方法对于促进人工智能创新中的协作、多样性和公平性至关重要。它有助于防止少数大公司垄断并进而控制人工智能技术的方向和应用。GenAI 中的中立治理支持道德标准，确保各部门负责任地、公平地使用人工智能。近 95% 的受访者强调了中立治理在 GenAI 发展中的重要性。【2】

而开源 AI 在中立治理下恰好可以蓬勃发展，因为它可以避免单一实体对技术产

生不当影响。这种协作治理模式允许不同的利益相关者为 AI 的发展做出贡献，确保创新更具包容性、更符合道德规范且具有长期可持续性。

## 企业采纳人工智能的三大挑战

人工智能技术，特别是基础模型和生成式人工智能最近几年得到了快速发展。人工智能的发展最初是由专有模型驱动的，但长远看这限制了开放性和创新性。特别是，对于旨在保持灵活性和促进创新的企业来说，避免跨各个技术层面的锁定至关重要。红帽的 CEO Matt Hicks 在 2024 年红帽峰会的主题演讲中预测："人工智能不会由单一供应商构建，也不会围绕单一的整体模型。您可以选择在任何地方运行人工智能。它将基于开源"。为什么会这么说呢？让我们先来看看这些"锁定"：

1. 硬件锁定：

   硬件锁定通常表现为企业大量投资单一供应商的专用 AI 处理器或 GPU。虽然此类投资可能带来初始成本效益或性能优化，但它们可能会限制企业转向其他供应商的能力，这些供应商可能提供更具竞争力的价格或创新产品。为了解决这个问题，企业应该优先考虑与硬件无关的架构。利用 Kubernetes 等可以跨不同硬件平台管理资源的解决方案，有助于确保 AI 应用程序保持可移植性和适应性，不受任何特定硬件供应商的限制。

2. 开发框架锁定：

   例如，某些并行计算平台和 API 模型，利用特定加速器的强大功能实现显著加速，并为开发者提供了强大的工具，但因其专有性质只适用于特定硬件，限制了与其他供应商硬件的互操作性。这就造成了一种锁定，因为切

换到可能提供更好性价比的硬件需要彻底改造现有的代码库以支持不同的平台。为了避免这种锁定，企业应该考虑采用更开放、与供应商无关的框架，支持多种硬件类型并减少对单一供应商生态系统的依赖。

3. 模型锁定：

   当企业依赖特定的专有模型时，就会发生 AI 模型锁定，而这些模型通常不是完全透明或可扩展的。例如，许多商业 AI 模型都是黑盒系统，其底层数据、训练参数和训练方法均未公开。这限制了企业根据特定需求修改或微调模型或将其与其他系统无缝集成的能力。更大的挑战在于人们常误认为所有开源模型都提供完全开放性；然而，一些“开源”模型在使用方面可能仍然存在限制，或者需要依赖专有数据或软件才能实现全部功能。为了解决这些问题，企业应该使用真正的开源模型框架，例如 LF AI & Data Foundation 支持的框架，这些框架促进了透明度和由社区驱动的发展。

企业解决了硬件、开发框架和模型中锁定的问题，才可以更好地定位自己，有效地利用人工智能技术，同时保持适应未来创新和满足市场需求的灵活性。

因此，红帽提出了业内亟需的、真正的在人工智能领域同样适用的开源价值观，并付诸实践。

**人工智能领域红帽的开源价值观**

因为人工智能领域的“模型”不同于软件领域的“程序”，程序的每一条指令精准的记录着软件的行为，是完全可解释的；而模型的权重虽然控制着模型的行为，但人类目前还无法对其做准确的解释，常被称为“黑盒子”或“灰盒子”。因此，权重的生成过程，即什么样的数据，以什么样的方式喂给大模型，就变得至关重要。

单单看模型运行时的权重和推理代码，是无法准确预测模型的行为的，这和软件领域的软件源代码不可同日而语。所以，人工智能领域的“开源”，不能仅从字面上去理解，不能认为“开放源代码”就算开源；而应该从开源的内涵上去理解。在代码包含着所有”秘密“的软件时代，”开源“指”开放源代码“是没有问题的；在代码已经不能包含太多”秘密“的人工智能时代，代码、权重、训练数据、训练方法这些，都应该被纳入开放的范畴。只有这些都开放了，才能被认为是真正意义上的”开源“。具体来说，我们主要关注三个方面：

1. 模型元素的可用性

   可用性是指模型代码、权重和训练数据等关键组件的开放性。要使模型真正开放，模型架构、训练数据集、模型权重和用于强化学习的数据（如果适用）等元素都应该公开可用。开放性使社区能够检查、复制和改进模型，从而促进透明度和协作开发。许多 AI 模型声称是开放的，但并未提供对所有关键元素的完全访问权限，从而限制了开发人员真正利用或审查模型的能力。真正的开放性可确保所有这些元素均在开放许可下可用，从而允许不受限制地使用和修改。

2. 模型文档

   模型文档可确保用户能够充分理解模型的构建、训练和运行方式。开放模型必须有详尽的文档支持，以解释模型的架构、数据源和训练方法。这包括模型卡或数据表等资源，其中陈述了道德考虑、数据源和潜在偏见。如果没有这样的文档，用户就无法批判性地评估模型是否适合他们的需求或潜在的道德影响。全面的文档是真正的开源 AI 的标志，允许外部研究人员和开发人员重现结果、改进模型并回馈社区。文档还可以增强使用开放

AI 模型的责任感。

3. 模型访问

   模型访问意味着用户可以通过公共 API 或可下载的包与模型进行交互的容易程度和公开程度。开放访问不仅限于源代码；它还包括为用户提供测试或实现模型的方法，例如通过公开可用的 API 或可下载的模型权重和推理代码。虽然某些模型通过专有 API 提供访问，但完全开放的 AI 模型允许下载和自托管整个系统，让用户完全控制模型的使用。许多模型被标记为开放，但通过有限的 API 限制实际访问，这阻碍了真正的所有权和灵活性。真正开放的 AI 模型可确保用户可以不受限制地下载、修改和部署模型。

**红帽实践：红帽的开放人工智能战略及实施**

基于此，红帽提出了开放的人工智能战略并付诸实施，加速企业对人工智能的采纳，同时保持开源价值。具体来说，主要包含三个方面：

1. 开放混合的人工智能平台：提供在任何地方运行 AI 的选择，具有 AI/ML 库、框架、运行时和在混合的 AI 算力加速基础设施上扩展计算的灵活性和可组合性。
2. 开放的模型运维：通过将开放的基础模型集成到可信的软件供应链中，帮助企业大规模构建、部署和运行安全的人工智能应用程序，释放数据价值。
3. 开放的模型: 与 IBM 合作发布高性能、Apache 2.0 许可的 Granite 语言和代码模型及工具，以便作为开源项目不断发展。例如，提供 InstructLab 模型对齐工具，InstructLab 本身就是一个开源社区，技术上为众多非 IT 专业

人士提供简单易用的访问接口，以社区的运作方式，让任何人都有机会参与到大模型的开发过程中。


在 COPU 例会上介绍了 AI 时代红帽的开源价值观。首先提到 AI 带来巨大潜力的同时也带来了在安全性、透明度和道德治理方面的挑战，红帽拥抱 100%开源的价值观，并提出在 AI 时代开源的概念要与时俱进，不仅是代码开源、权重开源，同时还涉及到训练数据开源、训练方法开源，等等。基于此理念，红帽发起了 InstructLab-一个旨在让任何人都有机会参与大模型开发的社区项目。同时，红帽还在众多的开源 AI 项目中积极投入和贡献，秉承着一贯的“上游优先”(upstream first)的原则，在下游开发了“开放混合 AI”的产品组合，包括 RHEL AI， OpenShift AI 等，满足企业用户的需求。最后，总结了 AI 时代红帽的开源价值观：让 AI 成为一个普惠大众的技术，让任何人都有机会参与开发、贡献、反馈，并从中受益。

## 结语

生成式 AI 带来巨大潜力的同时也带来了在安全性、透明度和道德治理方面的挑战，AI 时代开源的概念要与时俱进，不仅是代码开源、权重开源，同时还涉及到训练数据开源、训练方法开源，等等。红帽作为一家有着 100%开源文化基因的公司，提出“开放混合的 AI”的战略并付诸实践，满足不同用户的需求，让 AI 成为一个普惠大众的技术，让任何人都有机会参与开发、贡献、反馈，并从中受益。

【1】https://www.linuxfoundation.org/hubfs/LF%20Research/GenAI_Report_2023_020524.pdf

【2】https://doi.org/10.1145/3571884.3604316

# 讨论人工智能操作系统（AIOS）研制问题

COPU　2024.09.10

**主持人：陆首群（COPU 名誉主席）**

在当前人工智能时代，国内外关于人工智能操作系统的研制犹如井喷，我们要与日俱进，跟上时代的步伐。国内外研制 AIOS 分两类：一类是准 AIOS(feature AIOS),从应用入手,在传统 OS 中加智能模块，另一类是全功能 OS（All AIOS）从框架或内核入手，结合 AI 模型全面构建和研发 AIOS。研制单位也有两类：一类是企业界，另一类是学术界，在企业界中也有桌面、手机和汽车等几个层次。

这次会议我们请清华大学计算机系陈渝老师介绍国内外研制 AIOS 的情况，还请电子系汪玉教授团队的核心成员颜深根老师介绍他们研制情况；这次讨论会，我们邀请正在研制的十几家单位的代表参加讨论（学术界 2 位代表、企业界 9 位代表)，今天先做简短的一般性讨论，以后还要深入研究讨论（包括如何组织开发问题）。

**下面请清华大学计算机系陈渝教授发言：**

目前 AI 与 OS 的结合有两种思路方向：

1）**产业界思路：**OS for AI，从应用入手，自顶向下，形成 AI 实用+以大模型为基础的框架结构+网络+传统操作系统，这种方式对传统 OS 和网络几乎没有改动，适会企业尽快进入 AI 应用领域，并逐步向下修改 OS 本身。从发展现状来看，AI 应用的推广可能首先出现在以 AI 手机为代表的智能移动终端领域，以及自动驾驶汽车为代表的新能源智能网联汽车领域。同时在传统领域,

如桌面、服务器也会迅速引入各种 AI 应用，而未来的工业/服务智能机器人将会把各种 AI 应用综合在一种硬件平台上。

**2）学术界思路：**AI for OS，从框架或内核入手，自底向上把 AI 模型或深度学习模型直接引入 OS 内核和框架的构建研发中来，这种方式会对 OS 内核和框架产生较大的改变，有很多技术挑战，难度较大，需要在面向 AI 的计算理论和模型、AI 加速硬件优化、编程语言、软件工程、系统架构、优化技术、软硬协同优化等多个层面形成突破。由于这方面难以迅速突破，所以基于全新革命性的自底向上的研究需要时间和技术积累，产业落地周期长，以致企业难以采用这种方式。

无论哪种思路，都会向着一个趋势发展，即 AI 应用的强大需求，会进一步推动面向 AI 加速的芯片、网联、计算基础设施等的快速发展，并推动各种系统软件支撑、编程与 SDK 接口、服务框架的发展。从而推动计算机领域新的一个跳跃式发展。

**请清华大学电子系颜深根教授介绍研制 AIOS 的情况：**

**大模型带来的变化：**1）任务负载的变化。对计算资源、存储资源、带宽资源消耗更多。2）人工智能能力的变化，从识别到理解+生成。理解了之后，可以给出下一步的操作，可以转化成具体的指令。

**对操作系统的影响：**1）对应第一点，OS for AI，为了应对 AI 的负载，需要在 OS 层面做一些改变，比如计算效率要更高，支持更高带宽的存储。从以 CPU 为中心到以协处理器为中心。2）对应第二点，AI for OS，通过 AI 的能力，增强 OS 本来的一些功能。比如 OS 人机交互界面的更新，从触摸屏升级到语音、图像交互。OS 自带文件检索功能，OS 运行日志的自动整理等。

**目前大模型+OS 层面需要解决的问题：**1）云侧系统太过复杂，尤其在中国，底层芯片种类非常多，如何兼容各种不同的芯片。如何稳定地运行。尤其是国产芯片性能比不过英伟达的芯片，需要更大的并行规模，带来了效率和稳定性方面的双重挑战。2）在端侧目前大模型运行有三个方面的挑战，一个是算力，第二个是功耗，第三个是存储空间。算力和功耗目前主要是通过芯片工艺改进、针对性的芯片设计、计算指令优化来解决。存储分为两个方面，一方面是存储带宽，这个目前有存内计算、近存计算等思路。存储的第二个方面是存储容量，更大容量的存储导致成本上升，这个目前没有太好的解法。

**关于 AI+OS 的建议：**个人感觉 AI+OS 在云侧目前投入已经很多了，主要是端侧目前投入还比较少，希望能够有一个机制，把端侧相关的应用公司、系统公司、芯片公司拉到一起，大家商量怎么样统一接口，从试点开始，逐步增加 AI+OS 的应用，通过应用带动系统发展。

**COPU 副主席、北京大学教授陈钟：**谈到 AIOS, 我建议更多关注苹果公司 Apple-16, iOS，因为苹果公司是一个从硬件芯片到整机到软件全链条的生态。特别是其对结合大模型能力集成的理解具有标杆性，并且能够根据 AI 需要将芯片功能模块级、操作系统级和应用级完整设计和集成。9 月 10 日 iPhone16 正式发布，苹果公司推出产品级的设计成熟而稳定，甚至比微软更具有全球前沿性和引领性。

**COPU 专家委员会委员安泱谈：**安卓操作系统配置 AIcore 智能模块（推出 Android-15 版本），在 Android 平台上提供 AIcore 服务，在手机断网时 AI core 还可保留内容摘要（不致丢失内容）。我们应对此研究，在研制 AIOS 中是否有参考价值？

**统信软件产品经理顾雨婷介绍：**统信软件正在将 AI 能力变成操作系统的基础服务之一：1）在人机交互方面，统信软件推出的 UOS AI 智能助手通过全新升级的自然语言处理技术，能更好地理解用户需求，提供智能搜索、推荐系统、自动化办公助手等个性化服务；2）在应用生态方面，60 余款应用已接入 UOS AI 生态，通过本地/在线模型，向各应用提供 AI 能力支持；3）在开发赋能方面，统信软件打造的 UOSAIAPI 是国内首个系统级 AI 应用开发框架，为 AI 应用开发提供系统级 AI 接口，赋能生态伙伴应用开发。

**开源鸿蒙 PMC 主席任革林谈到:**鸿蒙操作系统原生支持端侧 AI，包括支持 AI 基础能力和利用 AI 自主改造的能力。AI 基础能力主要包括对 AI 硬件的管理、内置 AI 开发框架和 AI 推理模型等，方便用户程序高效利用系统的 AI 资源。操作系统利用 AI 自主改造的能力主要体现在利用预置的 AI 能力进一步提升当前操作系统智能化水平。譬如通过周边环境感知、运行上下文感知等 AI 技术让系统资源的管理更加高效；通过语音识别，机器视觉，用户操作习惯学习等 AI 能力，让交互操作个性化、更自然；通过 AI 能力将传统 UI 控件改造为智能 UI 控件，支持智能抠图、图像分割、图像理解、文字提取、文本朗读和智能纠错等；通过 AI 对分布式能力进行改造，提供智能搜索和智能协同等体验。

**华为开源陈道清谈到：**将自主开发的盘古智能模块加入到鸿蒙操作系统中。

**中兴通讯技术规划部标准规划总工高峰表示：**算力基础设施是 AIOS 的一个重要考虑方向，通过 OS 实现算力+网络+存储资源的分配、调度和优化，为 AI 提供基础服务能力。

**OPPO 技术总监罗升阳谈了几点：**1）大模型的端侧部署问题，一是功耗，二是内存带宽，三个计算生态。2）功耗问题：当前的旗舰平台手机运行 7B 模型需要的功耗还比较大，功耗和玩重度游戏差不多，会有发热发烫的问题，影响用户体验。3）内存带宽问题：一个 7B 模型，经过量化压缩之后，大概还占 4GB 左右的内存。由于大模型自回归推理的特点，导致每输出一个 token 都需要加载一遍参数到 NPU 计算。以 20 tokens/秒的推理速度计算，需要 80GB/秒的内存带宽。这对移动平台的芯片设计是一个很大的挑战。4）计算生态问题：当前移动芯片平台主要有高通和 mtk 两个大玩家，但是他们都是各玩各的，导致手机厂商开发商在部署模型时，面临碎片化问题。需要有一套统一的计算生态 API，它既是统一的，又像 CUDA 一样强大。

**麒麟软件副总李震宁介绍到：**当前，国外一些大模型热度减退，用户更加关注新技术能给自己带来的价值。人工智能的客户场景在哪里成为行业关注点。麒麟软件是国内自主创新领域应用量最大，应用程度最深的国产操作系统。针对政务和关键信息基础设施行业的用户需求。推出的 AIPCOS 从几个层面去帮助用户提升效率，拥抱 AI 未来：1）与传统桌面操作系统相比，融入了“AI 子系统”，该系统实现了模型和硬件、应用和模型的双重解耦。2）针对办公和行业应用场景，让 AI 和办公场景更深结合，实现智能搜索、AI 助手、记忆地图等增值功能。3）对于数据安全要求较高的行业，用户可以自由切换到本地大模型，摆脱网络依赖。

**COPU 常务副秘书长谭中意认为:** 一方面在 IDC 的服务器集群内，如何支持大模型的高效训练; 一方面在端侧，如何支持大模型的高效推理。两个方面的迭代，方向不同，要求不同，都应该是 AIOS 的范围。

**陆主席谈到：**

三个月来 COPU 曾多次组织业内企业讨论 AIOS，大家提出了两种研制模式：一是以应用入手，形成 AI 应用+AI 框架+传统 OS (OS for AI)，或是在现有的 OS 中加入智能模块( OS for LM)，也可称之为 AI-Powered OS 模式；二是以改造 OS 的框架或内核入手，把 AI 模型引入或融合到 OS 中(AI for OS)，也可称之为 AI-Refactored OS 模式。

在今天的讨论会上绝大多数企业正在研制前一种模式；下面我也向大家推荐一种语言大模型智能代理操作系统(我们叫 LLM Agent OS)，其主要特点是将大模型嵌入到操作系统中，作为操作系统的“大脑”，以实现一个“有灵魂”的操作系统，该操作系统的主要职能有：①优化运行在 LLM 上的资源分配，②促进各集成智能代理之间上下文转换，③实行各智能代理并发执行，④维护各智能代理之间的访问控制，而该 OS 的主要职能有：①内存管理，②存储管理，③驱动管理，④访问管理等。我认为前一种模式，只具有局部智能 OS 的特点，可能是一个过渡性产品，为了扩大大家的思维，有必要研制后一种模式。

**COPU 专家委员会副主任委员章文嵩补充说：**关于 LLM OS，从设计理念来看，应该保持每个模块的简单高效，不要把什么东西往内核里装。我们过去为了高性能更多地把内核的功能往用户态搬，网络有 DPDK，存储有 SPDK，例如 DPDK 版本的 LVS 在高性能网卡上性能大幅高于内核版本的 LVS，我们很多分布式存储系统用 SPDK 越过内核直接管理存储设备。LLM 应该是用户态的服务，为所有应用程序共享，它是 OS 的一部分。

# 为什么 AI 永远无法替代人类的大脑

HumanBrain，伦敦大学，2024.11.25

陈伟转译

今日主题：牛津大学重磅报告：为什么 AI 永远无法替代人类的大脑？

当人工智能（AI）在围棋、医学诊断、会计考试等领域不断超越人类时，你是否也曾疑惑：有朝一日，AI 会否完全替代人类？答案并非简单的“是”或“否”。

牛津大学的 Teppo Felin 与 Matthias Holweg 在研究报告《Theory Is All You Need: AI, Human Cognition, and Causal Reasoning》中，给出了一个令人深思的论断：AI 虽然在处理数据和预测方面无比强大，但在真正理解世界、创新和面对未知时，依然难以企及人类思维的深度。AI 与人类的分水岭：就在于“理论”——理论即一切。

## 那“理论”究竟是什么？它为什么如此重要？

什么是“理论”？作者如何定义它？

报告将“理论”定义为人类认知的核心工具，一种不仅描述现实，还能生成未来可能性的前瞻性思维方式。与仅仅依赖数据总结的模式不同，理论更像是认知的发动机，驱使我们去思考“为什么”和“如果……”的问题。

以下是报告中关于理论的几个核心特点：

1. 超越数据的创造性

理论并非简单地将数据堆叠，而是通过提炼抽象规律，预见新可能性。例如，相对论的提出，并非基于已有的观察结果，而是爱因斯坦通过理论推理，预测了时间膨胀和黑洞的存在。

2. 因果推理的基石

理论让我们能够理解变量之间的因果关系，而非仅仅发现相关性。例如，地球变暖不仅与温室气体排放相关，理论帮助我们推断了二者之间的因果机制，并指导具体的行动。

3. 生成新数据的能力

理论的独特之处在于它能推动我们主动干预世界、设计实验，生成从未被观察过的新数据。例如，在科学史上，从量子力学到基因编辑技术，背后都离不开理论驱动的实验创新。

**一个生动的比喻：**

理论就像登山时的指南针，数据是脚下的每一步。没有数据，你无法迈出探索的步伐；但没有理论，你永远不知道山顶在哪里，甚至可能在山脚徘徊。

为什么理论如此重要？AI 又为何无法真正掌握理论？

报告强调，AI 目前的优势在于从数据中总结模式，但它无法像人类一样提出超越数据的理论。这是因为：

1. AI 只看“过去”，人类看向“未来”

AI 的训练基于既有数据，缺乏对未来的假设能力。而理论是一种超越现实的思维方式，帮助人类预见尚未发生的可能性。

2. AI 无法理解“为什么”

理论让人类能够探讨“为什么某件事会发生”。而 AI 更多地依赖相关性，无法真正掌握因果逻辑。

3. AI 不能主动干预世界

理论赋予人类主动设计实验、生成新数据的能力。相比之下，AI 只能被动地处理已有数据。

**案例：青霉素的发现**

1928 年，亚历山大·弗莱明在实验室偶然发现了一种抑制细菌生长的霉菌。他并没有直接的数据支持这个霉菌能治疗感染，但通过理论推导，他设计了进一步的实验，最终开发了青霉素，改变了人类医学史。AI 则需要明确的数据和指令，无法在无先验信息的情况下提出这种创造性的假设。

**理论与数据的关系：一场微妙的平衡**

在讨论理论的定义时，我不太理解和赞同人类和 AI 的差别。似乎人类基于“理论”，难道“理论”不也是某种程度上后人被前人所建立数据的“投喂训练”吗？带着这样的质疑，我又仔细查阅了作者对“理论”的定义。

是的，理论离不开数据，但它更重要的作用在于“指导我们如何理解数据，并帮助我们超越数据”。

**作者将这种关系比作“信念与证据”的不对称性：**

- 数据是我们脚下的基石，但理论决定我们能否飞跃。

- AI 沉浸于数据，而人类依赖理论去塑造未来。

从理论出发，理解 AI 的局限

**报告深入剖析了 AI 的三大核心局限：**

**1. AI 的模仿性**

AI 系统通过训练数据模仿既有模式，但它无法创造出超越历史的新知识。

**2. AI 的因果推理缺失**

例如，AI 可以发现某种疾病与特定基因的相关性，但只有人类科学家才能通过理论，探讨那种基因如何导致疾病，并开发干预措施。

**3. AI 缺乏主动性**

AI 无法自己提出假设或主动设计实验来验证假设，这种能力正是理论驱动认知的核心。

## 理论即人类认知的未来

《Theory Is All You Need》为我们揭示了一件重要的事：人类的优势，不在于计算能力，而在于理论驱动的思维方式。这种能力让我们不仅能理解现实，还能突破现实的限制，去探索未知的未来。

AI 是强大的工具，但我们通过这份报告也可以稍感欣慰：它依然是我们认知的助手，而非替代者。

**未来属于那些相信理论力量的人。**