《多模态大模型》

Lecture 17
数学之美：大模型中算力不够，数学来凑

王广润
中山大学
人机物智能融合实验室（HCP Lab）
wanggrun@gmail.com

本节内容

CONTENTS

中山大学香港高等研究院

张益唐：从漂泊到数学突破

关键时间线

时间 / 阶段	关键事件
1978-1982	北大数学系学习，经历研究方向选择的关键分岔。
普渡大学读博	以雅可比猜想相关研究取得博士学位，但随后遭遇论文与求职挫折。
漂泊时期	做过外卖员、汽车旅馆零工、Subway 会计，同时坚持数学研究。
新罕布什尔大学	经推荐进入大学任教，执教微积分、代数、初等数论等课程。
2013	孪生素数猜想相关突破论文被《数学年刊》快速接收。
2014	获柯尔数论奖与麦克阿瑟奖。
2022-2024	继续挑战朗道-西格尔零点猜想

求学

一、求学阶段：方向选择与早期遗憾

- 1978-1982 年间，张益唐就读于北京大学数学系。
- 他本有机会在数学家丘成桐的推荐下，跟随数论学家哈若德·斯塔克学习数论。
- 数学家丁石孙（时任北大校长）建议他选择代数几何作为研究方向。虽然张益唐并不感兴趣，但仍接受了这一建议；事后他发现，这个决定令自己十分后悔。

普渡大学读博

二、普渡大学读博：雅可比猜想与学术挫折

- 在普渡大学读博期间，张益唐向导师莫宗坚表示，自己希望以雅可比猜想作为博士论文选题。
- 他最终以研究雅可比猜想的论文《雅可比猜想与域扩张的阶》取得博士学位。
- 博士毕业前夕，他曾宣称解决了雅可比猜想。但在排查证明时发现，其中一个关键引理来自导师莫宗坚已发表的成果，而该结果后来被确认存在错误。
- 张益唐不愿将这篇有瑕疵的博士论文结果整理发表；与此同时，导师也不愿为他写推荐信，导致他很难找到正式工作。

漂泊岁月

三、漂泊岁月：零工、援助与重新进入学术圈

- 博士毕业后，张益唐在美国四处漂泊，曾借住在朋友家或朋友家的地下室。
- 他做过中餐外卖员，也在汽车旅馆打过零工。
- 后来，北大校友开的 Subway 邀请他任会计。这份工作既能发挥他在记忆力和计算能力方面的特长，也允许他抽出时间继续研究数学，同时避免他因自尊心而拒绝接受帮助。
- 一次计算机算法难题中，张益唐只用了 3 周时间便将其解决，这让唐朴祁对他刮目相看。
- 经葛力明推荐，张益唐成为新罕布什尔大学数学系与统计学系编制外的助教，后担任讲师，执教微积分、代数、初等数论等课程。

孪生素数猜想

四、孪生素数猜想：孤独思考与关键灵感

- 为了专心思考数学，张益唐常常独自“闭关修炼”；有一次，他的妹妹甚至在网上发布寻人启事，称与哥哥失联。
- 后来，张益唐开始关注“孪生素数猜想”。
- 美国国家数学科学研究所在 2008 年曾专门召集一批顶尖数学专家开会讨论这一问题。经过一周讨论，与会者得出的结论是：用当时的方法无法解决。
- 张益唐当时并不知道这次会议，也没有资格参加，因此也不知道其他数学家已经暂时知难而退。
- 后来，他受邀到一位音乐家朋友家中教其儿子微积分。在齐光家的后院中，他一边抽烟、一边散步，顺便看有没有鹿出现；就在那时，他突然产生了能够撬动孪生素数猜想的关键灵感。

五、突破、荣誉与新的挑战

- 2013 年 4 月 17 日，张益唐向《数学年刊》（Annals of Mathematics）提交相关论文；2013 年 5 月 21 日论文正式接收。考虑到《数学年刊》当时平均审稿周期约为 24 个月，这创下了其创刊 130 年来论文接受最快的记录。
- 2014 年，美国数学学会将著名的柯尔数论奖授予张益唐。
- 2014 年 9 月，张益唐获得该年度麦克阿瑟奖（俗称“天才奖”）。
- 2022 年，张益唐表示自己在与黎曼猜想有关的朗道-西格尔零点猜想上取得重要进展；这也引出了一个引人关注的问题：他能否成为“被闪电击中两次的人”？

张益唐

□2025年6月，全职加盟中山大学

工作简历

2016-01~2017-08,美国加州圣塔芭芭拉大学, 教授

2013-06~2015-12,美国新罕布夏大学, 教授

1999-11~2013-05,美国新罕布夏大学, 讲师

1992-08~1999-07,赛百味, 打零工

1985-09~1992-07,美国普度大学, 博士

1982-09~1985-07,北京大学, 硕士

1978-09~1982-07,北京大学, 学士

本节内容

CONTENTS

一、数学家的数学

二、平凡人的数学(弦论)

三、数学与大模型中的物理

四、复变函数几何与大模型

五、**GRPO**的最小代码

六、低秩分解、扩散、因果

七、**Scaling Law**的数学原理

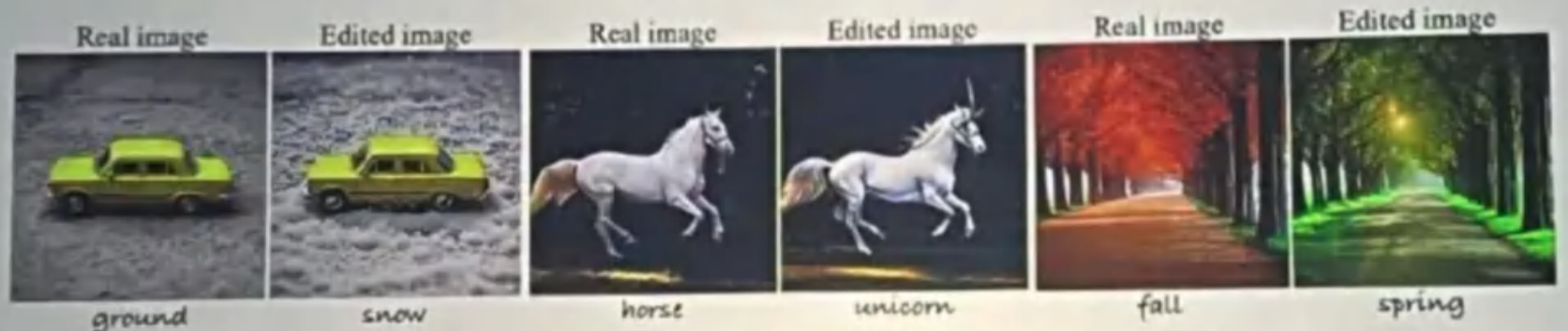
CVPR 2026, Denver, Jul. 7

Best Student Paper Honorable Mention

ChordEdit: One-Step Low-Energy Transport for Image Editing

Liangsi Lu[1], Xuhang Chen[2], Minzhe Guo[1], Shichu Li[3], Jingchao Wang[4], Yang Shi[1†]
[1]Guangdong University of Technology [2]Huizhou University
[3]Shenzhen University [4]Peking University
[†] Corresponding author: sudo.shiyang@gmail.com

Project page: https://chordedit.github.io



A one-step, training-free, inversion-free image editor that frames editing as low-energy optimal transport — theoretically grounded and surprisingly effective.

算力

Implementation details. Code is available in supplementary material. All experiments are conducted on a single NVIDIA Titan 24GB GPU. Our framework consists of the 1-NFE Chord transport step and an optional 1-NFE proximal refinement. To present the best overall performance, our default ChordEdit (NFE=2) includes this refinement (parameters: $n = 1$, $t = 0.90$, $\delta = 0.15$, $\lambda = 1.00$, $t_c = 0.30$). These defaults reflect a clear trade-off, as

作者介绍

Semantic Granularity Navigation in Image Editing

Liangsi Lu, Minzhe Guo, Xuhang Chen, Yang Shi

ICML 2026 Poster (International Conference on Machine Learning)

Project Page ↗ Paper ↗ Code ↗

ChordEdit: One-Step Low-Energy Transport for Image Editing

Liangsi Lu, Xuhang Chen, Minzhe Guo, Shichu Li, Jingchao Wang, Yang Shi

CVPR 2026 Oral / Best Student Paper Honorable Mention (IEEE Computer Vision and Pattern Recognition)

Project Page ↗ Conference Page ↗ Paper ↗ Code ↗ Video ↗

Riemannian Liquid Spatio-Temporal Graph Network

Liangsi Lu, Jingchao Wang, Zhaorong Dai, Hanqian Liu, Yang Shi

WWW 2026 Oral (The ACM Web Conference)

Project Page ↗ Paper ↗ Code ↗

作者介绍

Career & Education History

PhD student	the School of Computer Science, Peking University (pku.edu.cn)	*2025 – 2030*
Undergrad student	the School of Computer Science and Technology, Guangdong University of Technology (gdut.edu.cn)	*2021 – 2025*

作者介绍

- *[2026.06]* 🏆🏆 My paper “ChordEdit: One-Step Low-Energy Transport for Image Editing” (CVPR 2026) was nominated to be the Best Student Paper Honorable Mention🏆!
- *[2026.05]* 🏆🏆 My paper “ChordEdit: One-Step Low-Energy Transport for Image Editing” (CVPR 2026) was nominated to be the best paper award candidate (Top 0.45%)!
- *[2026.05]* 🎉🎉 My paper as the first author was accepted by SIGKDD 2026.
- *[2026.05]* 🎉🎉 My paper as the sole corresponding author was accepted by ICML 2026.
- *[2026.04]* 🎉🎉 My paper as the first author was accepted by ACL 2026 as a main conference paper.
- *[2026.02]* 🎉🎉 My paper as the sole corresponding author was accepted by CVPR 2026.
- *[2026.01]* 🎉🎉 My paper as the sole corresponding author was accepted by WWW 2026.
- *[2025.05]* 🏆🏆 I received funding from the “National College Students’ Innovation and Entrepreneurship Program”.

任务


Real image
Edited image
Real image
Edited image
Real image
Edited image
ground
snow
horse
unicorn
fall
spring
sweater
down jacket
reality
animation
crown
festive hat
salmon
bread
fox
dog
w/ cat
w/o cat

全局输入与输出

在启动编辑流程之前，系统需要获取以下初始信息。

全局输入

- 待编辑的原图，即源图像：x_{src}。
- 描述原图的文本，即源提示词：c_{src}。
- 描述编辑目标的文本，即目标提示词：c_{tar}。
- 控制参数：主时间步 t、时间窗口 δ、编辑强度缩放 λ、近端细化时间 t_c。

全局终极输出

编辑完成的目标图像：x_{tar}。

第一步：状态初始化

这是准备阶段，用来设定编辑的起点。

Initialization

输入：源图像 x_{src}。

过程：将输入的真实图像加载并设定为传输算法的初始锚点。在潜空间中，这通常对应于时间 $t = 1$ 时的干净图像。

输出：当前状态变量 x_{in}，此时有

$$x_{in} = x_{src}.$$

第二步：获取“观测残差场”

Compute Observable Proxy Fields

输入：当前图像状态 x_{in}、源提示词 c_{src}、目标提示词 c_{tar}，以及两个时间点：当前时间 t 和历史时间 $t-\delta$。

过程：

1. 在时间点 t 和 $t-\delta$，通过前向加噪核 K_t 为 x_{in} 注入特定强度的噪声，生成两个含有合成噪声的代理状态。
2. 将这些代理状态分别搭配 c_{src} 和 c_{tar} 输入到单步模型中，得到模型预测的输出，例如预测噪声或预测速度。
3. 计算目标条件输出与源条件输出的差值，并将其映射到统一的速度场空间中。

输出：两个时刻的“朴素残差场”，也就是最原始、未经处理的编辑方向：

$$R_{t-\delta}, \qquad R_t.$$

注意：正如论文指出的，这种直接相减得到的场充满了高能量和不稳定性。如果直接用它来编辑，图像容易崩坏。

第三步：生成核心的“弦控制场

这一步是 ChordEdit 的核心创新，对应低能量传输原理。它的作用是驯服上一步中剧烈、不稳定的残差场。

Generate Chord Control Field

输入：第二步得到的两个残差场 $R_{t-\delta}$ 和 R_t，以及时间参数 t 和 δ。

过程：采用时间加权平均，也就是因果单侧核平滑的方式，将当前时刻和历史时刻的场结合起来。这种机制迫使传输路径更加平滑，从而显著降低场的动能或方差。计算公式为：

$$\hat{u}_t = \frac{t \cdot R_{t-\delta} + \delta \cdot R_t}{t + \delta}.$$

输出：**弦控制场** $\hat{u}_t$。这是一个平稳、低能量、去除尖刺噪声后的全局编辑方向向量场。

第四步：大步长单步传输

Single-step Transport

输入：初始状态 x_{in}、弦控制场 $\hat{u}_t$、步长缩放系数 λ。

过程：执行类似显式欧拉方法的积分。算法沿着弦控制场指示的稳定方向，一次性跨越到目标状态。其中，λ 控制编辑发生的强度或距离。

计算公式为：

$$x^{pred} = x_{in} + \lambda \hat{u}_t.$$

输出：初步预测图像 x^{pred}。此时，目标物体，例如将猫变成狗，其主体结构已经沿着正确方向被修改，同时背景得到较好保留。

第五步：近端细化

由于单步大跨度的修改有时在细节语义上不够完美，这一步用于进一步增强目标语义。该步骤是可选的。

Proximal Refinement

输入：第四步得到的初步预测图像 x^{pred}、目标提示词 c_{tar}、较小的细化时间步 t_c。

过程：将 x^{pred} 视为一个带有轻微噪声的输入，使用目标提示词 c_{tar} 对其进行单次原生的前向传递，即调用模型直接预测无噪图像 predict-x0。这一步无需重新反演，只依靠目标提示词对图像进行锐化与语义对齐。

输出：最终编辑完成的图像 x_{tar}。这个图像不仅结构稳定，而且细节上更符合目标描述。

流程小结

步骤	核心作用	关键输出
状态初始化	确定编辑起点	$x_{in} = x_{src}$
观测残差场	比较源条件与目标条件的生成方向	$R_{t-\delta}$、R_t
弦控制场	对残差场进行低能量平滑	$\hat{u}_t$
单步传输	沿稳定方向进行大步长更新	x^{pred}
近端细化	强化目标语义并改善细节	x_{tar}

总体来看，ChordEdit 的关键并不是简单地把目标条件和源条件的预测结果相减，而是先从两个时间点获得残差场，再通过弦控制场构造一个低能量、稳定的编辑方向。这样既能实现较大幅度的语义修改，又能尽量保持原图中的背景和整体结构。

本节内容

CONTENTS

一、数学家的数学

二、平凡人的数学(弦论)

三、数学与大模型中的物理

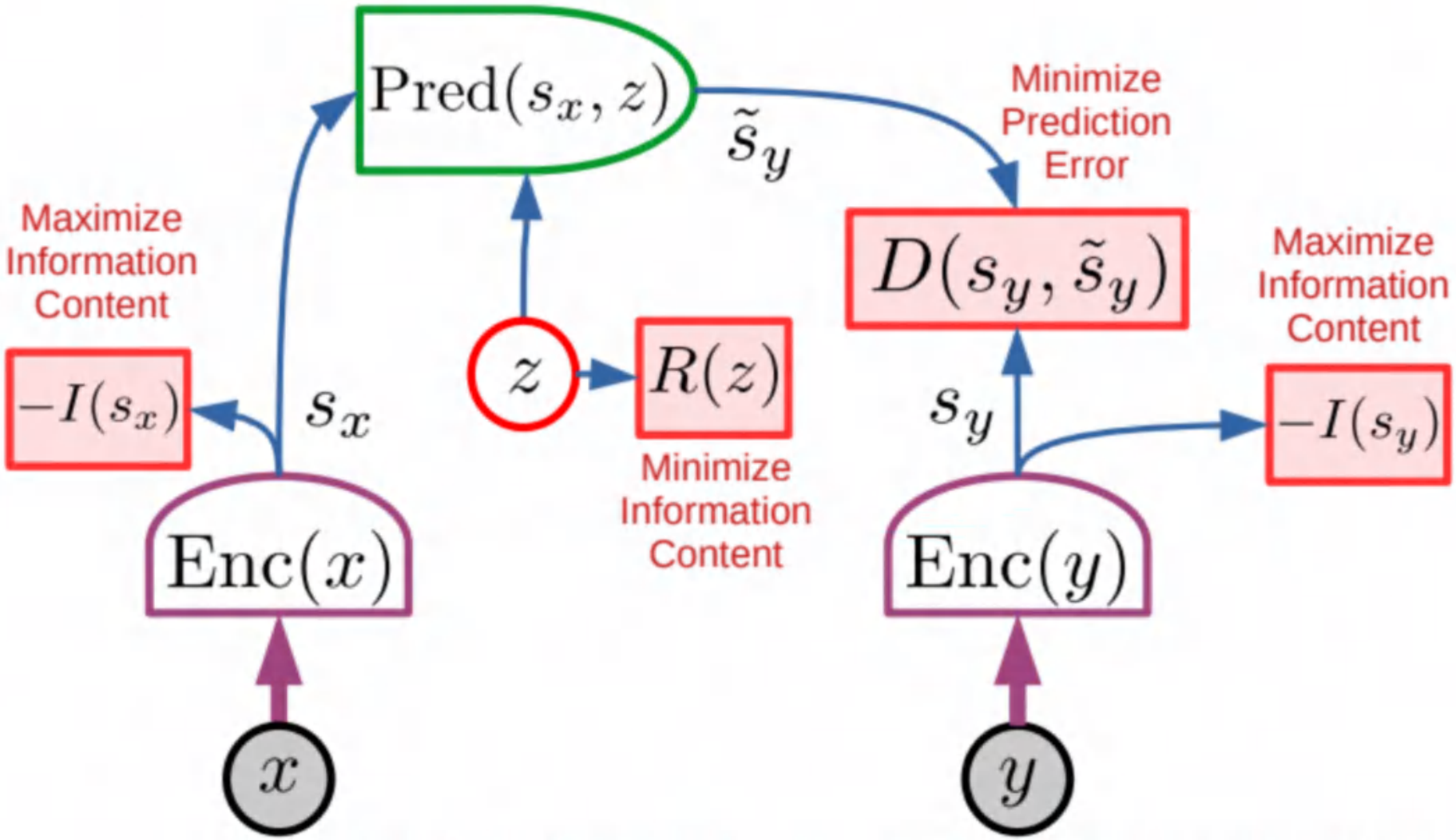
四、复变函数几何与大模型

五、GRPO的最小代码

六、低秩分解、扩散、因果

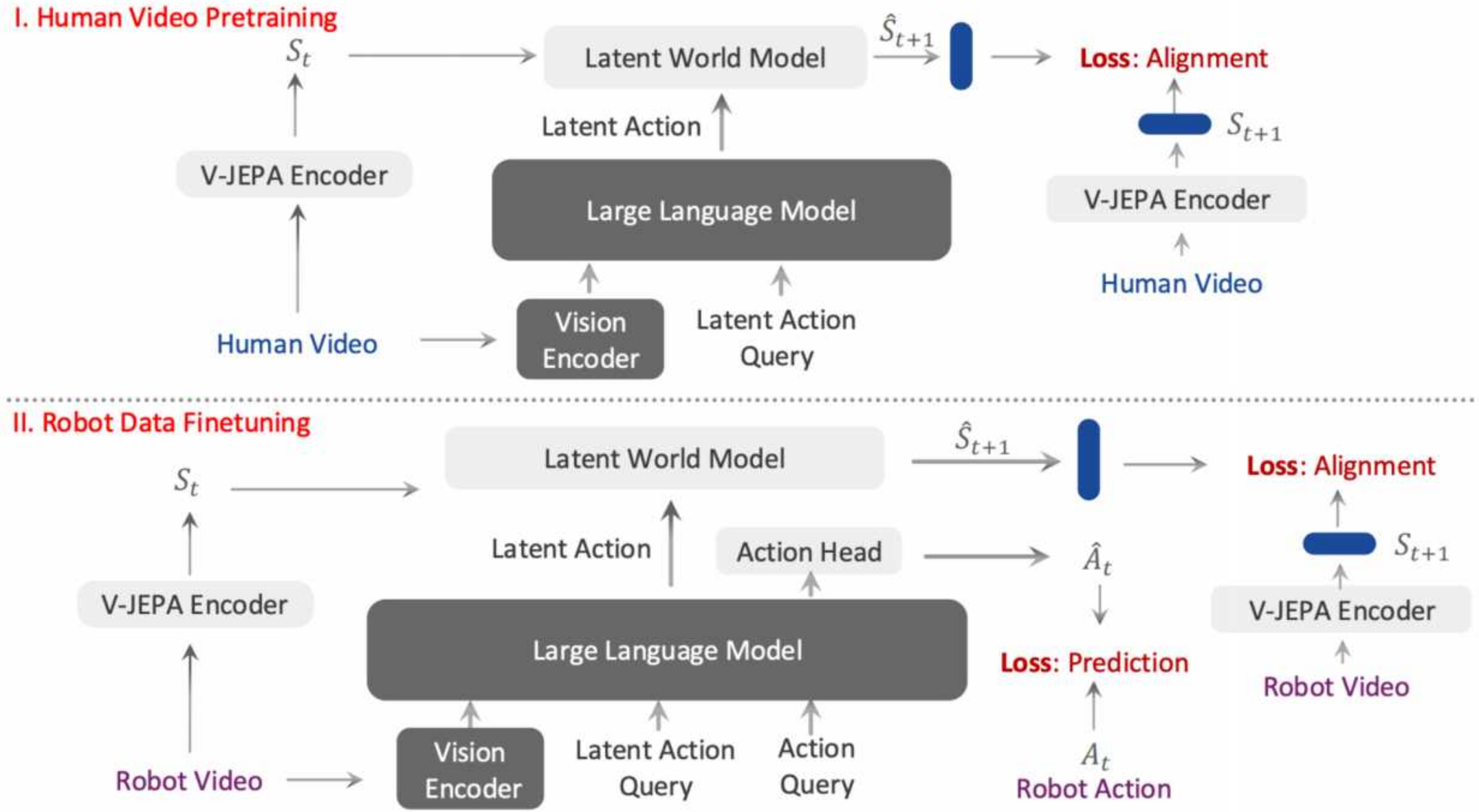
七、Scaling Law的数学原理

Yann LeCun的JEPA



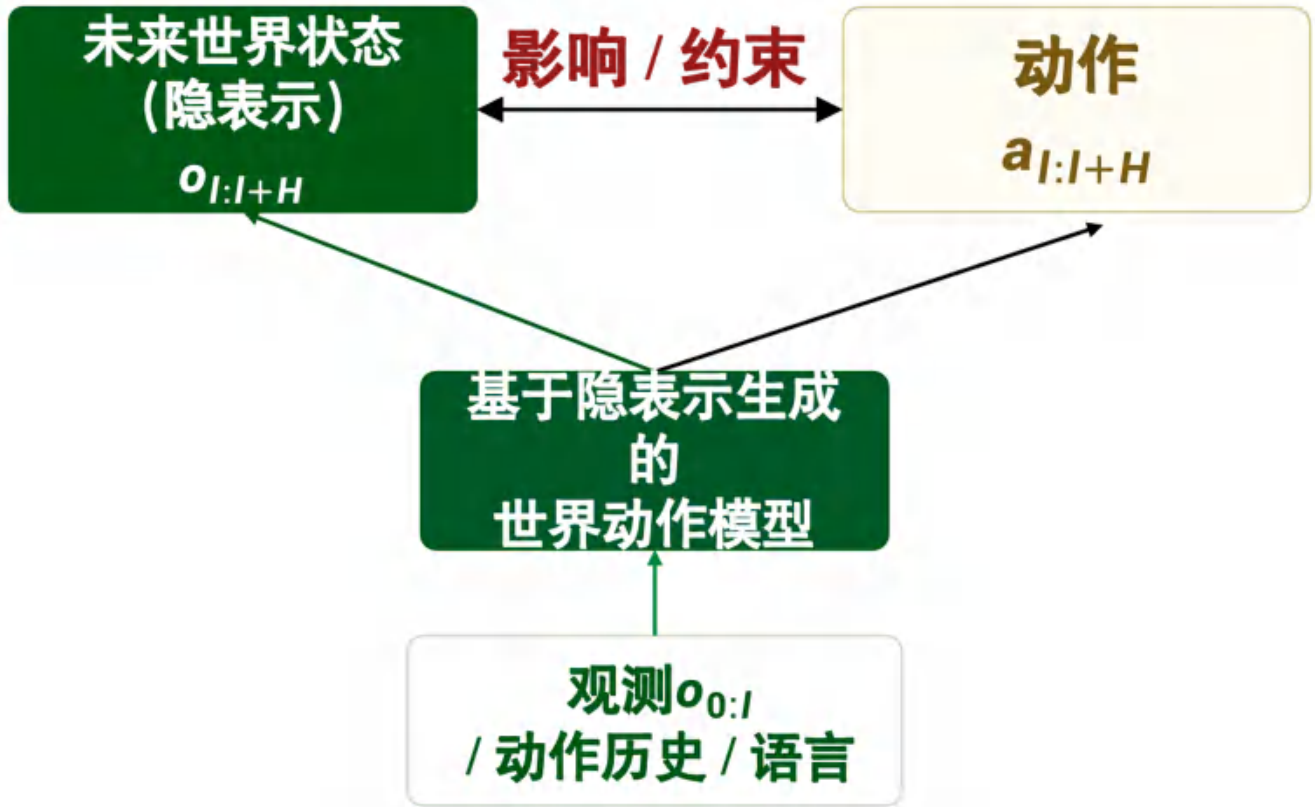
LeCun的隐空间世界模型，其实就是几年前的自监督学习的变体，当时他引用了我的论文

JEPA-VLA


I. Human Video Pretraining
S_t
Latent World Model
$\hat{S}_{t+1}$
Loss: Alignment
S_{t+1}
Latent Action
V-JEPA Encoder
Large Language Model
V-JEPA Encoder
Human Video
Human Video
Vision Encoder
Latent Action Query
II. Robot Data Finetuning
S_t
Latent World Model
$\hat{S}_{t+1}$
Loss: Alignment
S_{t+1}
Latent Action
Action Head
$\hat{A}_t$
V-JEPA Encoder
Large Language Model
Loss: Prediction
V-JEPA Encoder
Robot Video
Robot Video
Vision Encoder
Latent Action Query
Action Query
A_t
Robot Action

隐空间表征的好处

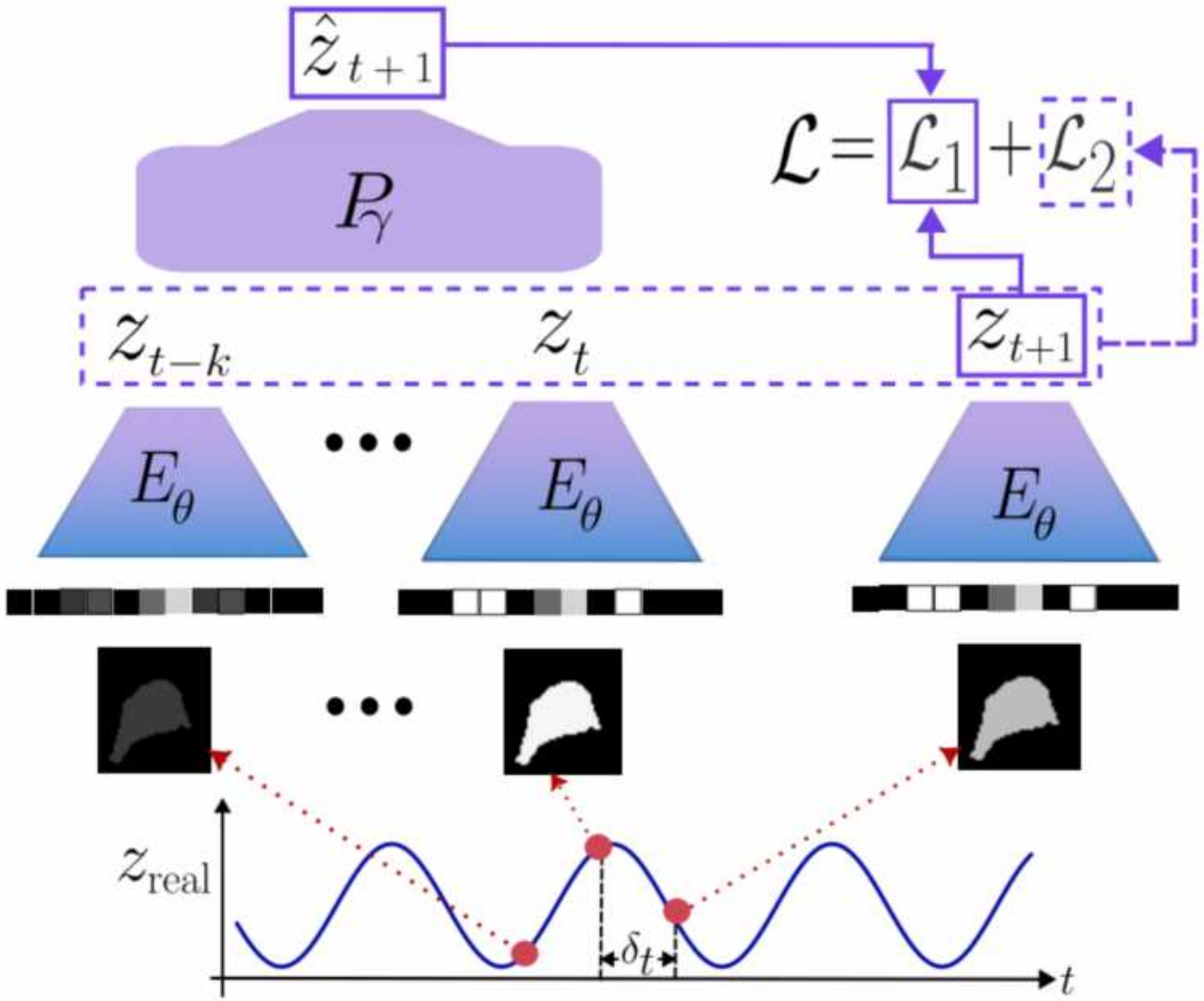
基于视频生成和几何生成的 WAM 分别将未来世界状态表示为未来图像、视频隐变量、深度、点云或三维几何结构。与之相比，基于隐表示生成的 WAM 更加抽象：它不要求模型显式生成可观看的视频或可渲染的几何，而是直接在学习到的隐空间中联合生成未来动作序列与未来世界状态表示。设隐空间世界状态为



隐空间的问题

- 隐空间号称能学习到物理，但是它真的能吗？

核心背景与方法动机


$\hat{z}_{t+1}$
P_γ
$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$
z_{t-k}
z_t
z_{t+1}
E_θ
E_θ
E_θ
z_{real}
δ_t
t

图像编码阶段：Encoder

输入

原始视频中的单帧图像：

$$x_t \in \mathbb{R}^{w \times h \times c},$$

其中 w、h、c 分别表示图像宽度、高度与通道数。

原理

模型使用一个包含三层和 ReLU 激活函数的多层感知机（MLP）作为编码器 E_θ。该编码器不参与图像重建，其唯一任务是将高维像素映射到低维隐变量表示中：

$$z_t = E_\theta(x_t).$$

输出

当前时刻的隐空间状态变量：

$$z_t \in \mathbb{R}^d,$$

其中 d 表示动态变量的数量。

物理方程演化阶段：Physics ODE Block

输入

编码器在当前及过去时刻提取的隐变量状态，例如 z_t、z_{t-1}，以及待学习的物理参数 γ。这些参数可以表示阻尼因子、弹簧常数、衰减率等物理量。

原理

模块内部植入该物理系统已知的连续微分方程。以二阶动力系统为例，其运动可以写作：

$$z^{(2)} + \gamma_1 z^{(1)} + \gamma_0 z = 0.$$

其中 $z^{(1)}$ 与 $z^{(2)}$ 分别表示一阶导数和二阶导数，γ_0、γ_1 是待估计的物理参数。
为了从连续方程得到离散时间上的状态演化，模型采用数值方法，例如欧拉方法，并使用固定时间步长 δt 求解该微分方程。由于整个推导过程是可微的，网络可以通过反向传播同步更新编码器参数 θ 与物理参数 γ。

输出

由物理方程推导出的下一时刻隐状态预测值：

$$\hat{z}_{t+1}.$$

该预测值不是像素空间中的未来图像，而是隐空间中的未来物理状态。

欧拉方法

$$z^{(2)} + \gamma_1 z^{(1)} + \gamma_0 z = 0. \tag{2}$$

Physics block. Our physics block numerically solves the differential equation using a single step of Euler's method:

$$z_t^{(1)} \approx \frac{z_{t+1} - z_t}{\delta t} \approx \frac{z_t - z_{t-1}}{\delta t} \tag{3}$$

$$z_{t+1} = z_t + \delta t z_t^{(1)} \tag{4}$$

$$z_{t+1}^{(1)} = z_t^{(1)} + \delta t z_t^{(2)}. \tag{5}$$

Plugging in Eq. 2, we can rewrite the physics block as:

$$\hat{z}_{t+1} = z_t + \delta t \left(z_t^{(1)} - \delta t(\gamma_1 z_t^{(1)} + \gamma_0 z_t) \right). \tag{6}$$

$$P : \mathbb{R}^d \to \mathbb{R}^d,$$

$$\hat{z}_{t+1} = P_\gamma(z_t, ..z_{t-n}; \gamma). \tag{7}$$

隐空间损失计算

输入

编码器从下一帧图像中直接提取的状态：

$$z_{t+1} = E_\theta(x_{t+1}),$$

以及物理模块预测出的状态：

$$\hat{z}_{t+1}.$$

一致性损失采用均方误差，衡量物理预测值 $\hat{z}_{t+1}$ 与编码器实际提取值 z_{t+1} 之间的差异：

$$\mathcal{L}_1 = \|\hat{z}_{t+1} - z_{t+1}\|_2^2.$$

因此，分布散度损失可以概括为：

$$\mathcal{L}_2 = D_{\mathrm{KL}}\big(P(z) \,\|\, Q\big),$$

输出

最终训练损失为：

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2.$$

该损失用于反向传播，同时更新编码器参数 θ 和真正需要估计的物理参数 γ。

总结

该方法的本质是：**不再重建图像，而是让图像编码出的隐变量必须服从物理方程；物理参数则作为可学习变量，在隐空间一致性约束下被自动估计出来。**

本节内容

CONTENTS

一、数学家的数学

二、平凡人的数学(弦论)

三、数学与大模型中的物理

四、复变函数几何与大模型

五、GRPO的最小代码

六、低秩分解、扩散、因果

七、Scaling Law的数学原理

旋转位置编码（RoPE）原理与数学推导

旋转位置编码（Rotary Position Embedding, RoPE）是目前大语言模型（如 LLaMA、Qwen 等）中最核心的相对位置编码机制。RoPE 的核心原理可以概括为一句话：**通过在 Query 和 Key 注入绝对位置的旋转变换，使得它们在进行点积计算注意力分数时，自然地涌现出相对位置信息。**

第一步：线性投影

输入： 位于绝对位置 m 的 token 特征向量 $\boldsymbol{x}_m \in \mathbb{R}^{d_{model}}$，以及位于绝对位置 n 的 token 特征向量 $\boldsymbol{x}_n \in \mathbb{R}^{d_{model}}$。

计算： 通过权重矩阵进行线性变换：

$$\boldsymbol{q}_m = \boldsymbol{W}_q \boldsymbol{x}_m, \tag{1}$$

$$\boldsymbol{k}_n = \boldsymbol{W}_k \boldsymbol{x}_n. \tag{2}$$

输出： 当前层的 Query 向量 $\boldsymbol{q}_m \in \mathbb{R}^d$ 和 Key 向量 $\boldsymbol{k}_n \in \mathbb{R}^d$。其中 d 是注意力头的维度，且 d 必须是偶数。此时的 $\boldsymbol{q}_m$ 和 $\boldsymbol{k}_n$ 还不包含任何位置信息。向量可表示为

$$\boldsymbol{q}_m = [q_1, q_2, \ldots, q_d]^\top. \tag{3}$$

第二步：复数化与分组映射

输入： 第一步输出的 d 维 Query 向量

$$\boldsymbol{q}_m = [q_1, q_2, \ldots, q_d]^\top.$$

计算： 按照维度索引，将相邻的两个元素作为一个复数的实部和虚部。对于

$$j \in \{1, 2, \ldots, d/2\},$$

构建 $d/2$ 个二维子空间，同时为每个子空间分配一个特定的旋转频率 θ_j：

$$\theta_j = 10000^{-2(j-1)/d}.$$

输出： 得到 $d/2$ 个复数特征表示

$$q^{(j)} = q_{2j-1} + \mathbf{i} q_{2j},$$

以及对应的旋转频率 θ_j。

第三步：注入绝对位置（旋转变换）

输入： 第二步的分组特征，以及当前 token 的绝对位置标量 m。

计算： 将绝对位置 m 与每个子空间的频率 θ_j 相乘，得到旋转角度 $m\theta_j$。在复平面上，旋转等价于乘以 $e^{\mathbf{i}m\theta_j}$。对于第 j 对元素，有

$$\tilde{q}^{(j)} = (q_{2j-1} + \mathbf{i}q_{2j})e^{\mathbf{i}m\theta_j}. \tag{8}$$

根据欧拉公式

$$e^{\mathbf{i}m\theta_j} = \cos(m\theta_j) + \mathbf{i}\sin(m\theta_j), \tag{9}$$

第三步：注入绝对位置（旋转变换）

根据欧拉公式

$$e^{\mathbf{i}m\theta_j} = \cos(m\theta_j) + \mathbf{i}\sin(m\theta_j), \tag{9}$$

展开并分离实部和虚部，可以用实数矩阵乘法来表示这一步。定义块对角旋转矩阵 $\boldsymbol{R}_{\Theta,m}$：

$$\boldsymbol{R}_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix} \tag{10}$$

第三步：注入绝对位置（旋转变换）

旋转后的 Query 向量为

$$\tilde{\boldsymbol{q}}_m = \boldsymbol{R}_{\Theta,m}\boldsymbol{q}_m. \tag{11}$$

同理，对 Key 向量应用位于绝对位置 n 的旋转：

$$\tilde{\boldsymbol{k}}_n = \boldsymbol{R}_{\Theta,n}\boldsymbol{k}_n. \tag{12}$$

输出： 编码了绝对位置信息的旋转 Query 向量 $\tilde{\boldsymbol{q}}_m$ 和旋转 Key 向量 $\tilde{\boldsymbol{k}}_n$。其中任意第 j 对的输出元素为

$$\tilde{q}_{2j-1} = q_{2j-1}\cos(m\theta_j) - q_{2j}\sin(m\theta_j), \tag{13}$$

$$\tilde{q}_{2j} = q_{2j-1}\sin(m\theta_j) + q_{2j}\cos(m\theta_j). \tag{14}$$

第四步：注意力计算（相对位置的涌现）

输入： 第三步输出的携带绝对位置的 $\tilde{\boldsymbol{q}}_m$ 和 $\tilde{\boldsymbol{k}}_n$。

计算： 计算两个向量的内积 $\langle \tilde{\boldsymbol{q}}_m, \tilde{\boldsymbol{k}}_n \rangle$：

$$\tilde{\boldsymbol{q}}_m^\top \tilde{\boldsymbol{k}}_n = (\boldsymbol{R}_{\Theta,m}\boldsymbol{q}_m)^\top (\boldsymbol{R}_{\Theta,n}\boldsymbol{k}_n) \tag{15}$$

$$= \boldsymbol{q}_m^\top \boldsymbol{R}_{\Theta,m}^\top \boldsymbol{R}_{\Theta,n}\boldsymbol{k}_n. \tag{16}$$

最终的内积结果为

$$\tilde{\boldsymbol{q}}_m^\top \tilde{\boldsymbol{k}}_n = \boldsymbol{q}_m^\top \boldsymbol{R}_{\Theta,n-m}\boldsymbol{k}_n. \tag{19}$$

输出： 注意力分数，即未经过缩放和 Softmax 处理的 logits。从输出公式中可以清晰看到：**虽然第三步中只使用了 m 和 n 的绝对位置，但最终输出的点积结果却由 $(n-m)$ 这个相对位置决定。**

工程实现上的优化

在实际的代码实现中，例如 PyTorch 或 CUDA 算子，由于 $\boldsymbol{R}_{\Theta,m}$ 是一个极度稀疏的块对角矩阵，直接进行矩阵乘法会造成巨大的算力浪费。

因此，实际的输出计算通常通过逐元素乘法（element-wise multiplication）等效实现：

$$\tilde{\boldsymbol{q}}_m = \boldsymbol{q}_m \odot \cos(m\boldsymbol{\Theta}) + \text{Flip}(\boldsymbol{q}_m) \odot \sin(m\boldsymbol{\Theta}). \tag{20}$$

其中 $\text{Flip}(\boldsymbol{q}_m)$ 的操作是将向量的偶数维度取反，并与奇数维度交换位置：

$$\text{Flip}([q_1, q_2, q_3, q_4, \ldots, q_{d-1}, q_d]^\top) = [-q_2, q_1, -q_4, q_3, \ldots, -q_d, q_{d-1}]^\top. \tag{21}$$

这种方式不仅维持了上述推导的严格等价性，还将计算复杂度从 $\mathcal{O}(d^2)$ 降低到 $\mathcal{O}(d)$，兼顾了表达能力与计算并行度。

本节内容

CONTENTS

定义一个policy net和一个reward fn

```python
class Policy(nn.Module):
    def __init__(self, num_prompts, num_actions, hidden_dim=64):
        super().__init__()
        self.emb = nn.Embedding(num_prompts, hidden_dim)
        self.head = nn.Linear(hidden_dim, num_actions)

    def forward(self, prompts):
        x = self.emb(prompts)
        logits = self.head(x)
        return logits
```

```python
def reward_fn(prompts, actions, num_actions):
    """
    假设每个 prompt 的正确答案是 prompt_id % num_actions。
    答对 reward = 1，答错 reward = 0。
    """
    targets = prompts % num_actions
    rewards = (actions == targets).float()
    return rewards
```

固定一个Ref net

```
num_prompts = 32
num_actions = 8

policy = Policy(num_prompts, num_actions)
ref_policy = copy.deepcopy(policy)

for p in ref_policy.parameters():
    p.requires_grad = False

optimizer = torch.optim.AdamW(policy.parameters(), lr=1e-3)

group_size = 8          # 每个 prompt 采样几个答案
batch_size = 16         # prompt 数量
clip_eps = 0.2
beta = 0.05             # KL penalty 系数
```

循环训练

```
for step in range(1000):
    prompts = torch.randint(0, num_prompts, (batch_size,))

    prompts_group = prompts.repeat_interleave(group_size)

    with torch.no_grad():
        logits_old = policy(prompts_group)
        dist_old = Categorical(logits=logits_old)

        actions = dist_old.sample()
        old_log_probs = dist_old.log_prob(actions)

        rewards = reward_fn(prompts_group, actions, num_actions)

    rewards_grouped = rewards.view(batch_size, group_size)

    mean = rewards_grouped.mean(dim=1, keepdim=True)
    std = rewards_grouped.std(dim=1, keepdim=True) + 1e-8

    advantages = (rewards_grouped - mean) / std
    advantages = advantages.view(-1).detach()
```

算advantage

```
    logits = policy(prompts_group)
    dist = Categorical(logits=logits)
    log_probs = dist.log_prob(actions)
```

再算一遍，真正拿来使用的

```
    with torch.no_grad():
        ref_logits = ref_policy(prompts_group)
        ref_dist = Categorical(logits=ref_logits)
        ref_log_probs = ref_dist.log_prob(actions)
```

用Ref net再算一遍

```
ratio = torch.exp(log_probs - old_log_probs)

unclipped = ratio * advantages
clipped = torch.clamp(ratio, 1 - clip_eps, 1 + clip_ep

policy_loss = -torch.min(unclipped, clipped).mean()
```

如果 advantage大于0，则希望后面算的要比前面算的要更好。反之，则相反

```
log_ratio_ref = ref_log_probs - log_probs
kl = torch.exp(log_ratio_ref) - log_ratio_ref - 1
kl_loss = kl.mean()

loss = policy_loss + beta * kl_loss
```

不偏离原来的ref net

更全面的代码

```
with torch.no_grad():
    logits_old = policy(prompts_group)
    dist_old = Categorical(logits=logits_old)

    actions = dist_old.sample()
    old_log_probs = dist_old.log_prob(actions)

    rewards = reward_fn(prompts_group, actions, num_actions)

# 计算 group-relative advantage
rewards_grouped = rewards.view(batch_size, group_size)
advantages = (rewards_grouped - rewards_grouped.mean(dim=1, keepdim=True)) / (
    rewards_grouped.std(dim=1, keepdim=True) + 1e-8
)
advantages = advantages.view(-1).detach()

# 对同一批 rollout 数据，做多次 PPO / GRPO 更新
for ppo_epoch in range(4):

    logits = policy(prompts_group)
    dist = Categorical(logits=logits)
    log_probs = dist.log_prob(actions)

    ratio = torch.exp(log_probs - old_log_probs)

    unclipped = ratio * advantages
    clipped = torch.clamp(ratio, 1 - clip_eps, 1 + clip_eps) * advantages

    policy_loss = -torch.min(unclipped, clipped).mean()

    optimizer.zero_grad()
    policy_loss.backward()
    optimizer.step()
```

本节内容

CONTENTS

一、数学家的数学

二、平凡人的数学(弦论)

三、数学与大模型中的物理

四、复变函数几何与大模型

五、GRPO的最小代码

六、低秩分解、扩散、因果

七、Scaling Law的数学原理

LoRA

$$W = W_0 + \Delta W = W_0 + BA$$

- $B \in \mathbb{R}^{d \times r}$ 且 $A \in \mathbb{R}^{r \times k}$。
- 秩 r 是一个超参数，且满足 $r \ll \min(d, k)$。

在前向传播时，输入向量 $x \in \mathbb{R}^k$ 的计算过程演变为：

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

扩散

$$x_t = \sqrt{\bar{\alpha}_t} x_1 + \sqrt{1 - \bar{\alpha}_t} \epsilon$$

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{x_1, \epsilon \sim \mathcal{N}(0, I), t} \left[\| \epsilon - \epsilon_\theta(x_t, t) \|^2 \right]$$

能量模型的痛点与“得分”的引入

在能量模型中，我们将真实数据的概率分布 $p_\theta(x)$ 定义为一个由神经网络参数化的能量函数 $E_\theta(x)$ 的玻尔兹曼分布：

$$p_\theta(x) = \frac{\exp\big(-E_\theta(x)\big)}{Z_\theta}. \tag{1}$$

其中，Z_θ 是配分函数（Partition Function），用于归一化概率：

$$Z_\theta = \int \exp\big(-E_\theta(x)\big)\, dx. \tag{2}$$

EBM 的痛点在于 Z_θ 涉及到对高维数据空间的积分，计算上是完全不可解的（intractable），这使得直接通过最大似然估计（maximum likelihood estimation, MLE）来训练 EBM 极其困难。

解决方案

为了绕过 Z_θ，我们可以对对数概率密度求关于数据 x 的梯度，这个梯度被称为**得分函数（Score Function）**：

$$
\begin{aligned}
s_\theta(x) &= \nabla_x \log p_\theta(x) && (3)\\
&= \nabla_x \big(-\log Z_\theta - E_\theta(x)\big) && (4)\\
&= -\nabla_x E_\theta(x). && (5)
\end{aligned}
$$

可以看到，因为 Z_θ 与 x 无关，求导后直接消失了。**得分 $s_\theta(x)$ 实际上就是能量场梯度的反方向**，它指向数据密度最高、能量最低的方向。

扩散模型即“多尺度得分匹配

在扩散模型（如 DDPM）中，我们训练一个网络 $\epsilon_\theta(x_t, t)$ 来预测添加到图像中的噪声。

根据数学推导（Tweedie 公式），在给定的时间步 t 下，带噪数据分布 $q(x_t)$ 的得分函数与网络预测的噪声 ϵ_θ 存在直接的线性关系：

$$\nabla_{x_t} \log q(x_t) = -\frac{\epsilon_\theta(x_t, t)}{\sqrt{1-\bar{\alpha}_t}}. \tag{6}$$

结合第一部分的结论推导，我们可以得到：

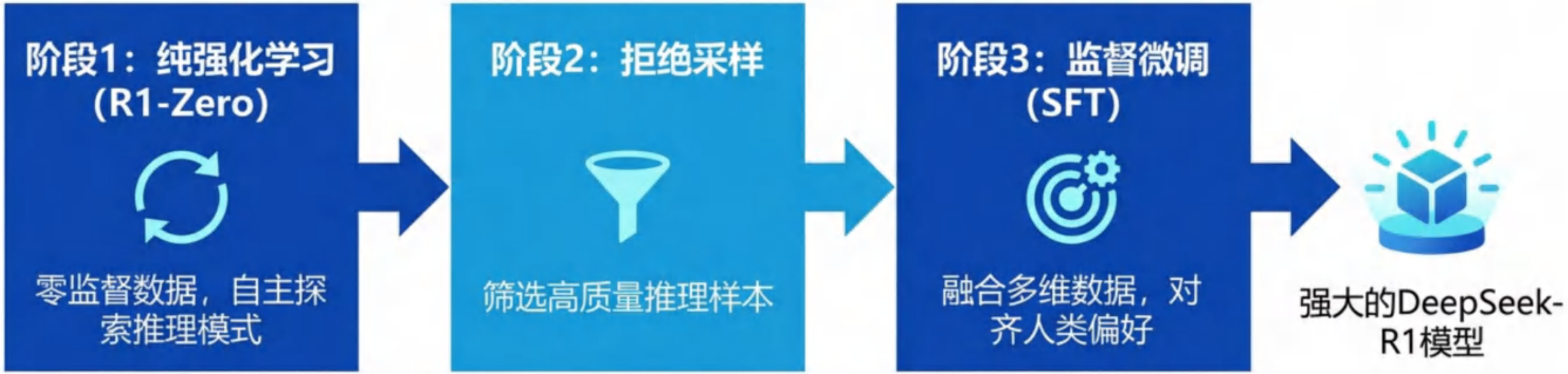
$$-\nabla_{x_t} E_\theta(x_t, t) \approx -\frac{\epsilon_\theta(x_t, t)}{\sqrt{1-\bar{\alpha}_t}}. \tag{7}$$

核心结论一：扩散模型中预测噪声的网络，实际上是在学习每个时间步 t 下，带噪数据分布的**能量函数梯度（得分）**。

因果

用因果关系指导动作干预与反事实推理，生成更可靠的增强样本。RL在增强后的样本分布上做策略微调与奖励对齐。

对比DeepSeek



阶段1：纯强化学习（R1-Zero）

完全通过RL自主探索推理模式

融合推理和非推理数据，对齐人类偏好

本节内容

CONTENTS

一、数学家的数学

二、平凡人的数学(弦论)

三、数学与大模型中的物理

四、复变函数几何与大模型

五、GRPO的最小代码

六、低秩分解、扩散、因果

七、Scaling Law的数学原理

The Bitter Lesson

Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

Evaluation

Validation: What do you keep an eye on during training?

Test: How do we compare different LLMs quantitatively?

How do you know any of this is any good?

- Loss on pre-training validation set (avg cross-entropy across tokens)

$$\mathcal{L} = \frac{1}{N} \sum_{t=1}^{N} -\log p_\theta(x_t \mid x_{<t})$$

- Perplexity (PPL) = e ^ L

Intuition: "On avg, the model is as uncertain as selecting between PPL equally-likely options."

What's attractive about this? (Universal & smooth!)

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Other metrics

Category	Representative Dataset	Example
Cloze tasks (LM-like but selective)	LAMBADA ('16)	"Preston had been the last person to wear those chains. [...] But he put down the ____" (chains)
General Language Understanding	SuperGLUE ('19)	Is windows movie maker part of windows essentials? (Yes/No)
General Knowledge & Problem Solving	MMLU ('20)	what is the agenda for hollande's state visit to washington
Instruction Following	IFEval ('23)	can you make and receive calls in airplane mode
"Deep Research"	HotPotQA('18), HoVer ('20), and BrowseComp ('25)	"Which MVP of a World Series game in which Red Flaherty umpired was later elected to the Baseball Hall of Fame?"
Code Generation	HumanEval ('21), SWE-Bench ('23)	"Write a Python snippet implementing: From a list of integers, remove all elements that occur more than once"
Computer Use	OSWorld ('24), AppWorld ('24)	"Approve all Vennmo payment requests from my roommates from this month."
Live Human Usage!	Chatbot Arena (2023–)	*People ask hard questions, e.g. tricky coding problems or personal assistant requests, and vote on anonymous LLMs!*

ML is in a deep evals crisis…

Contamination. How do you ensure benchmarks are **not** in your pre-training data?

Explicit hill-climbing. By '23, LLMs switched from "few-shot learners" to aggressive training & validation on *all* benchmarks. (Sometimes test sets, too! Hopefully at least transparently!)

Emergence. Capabilities "emerge" from lots of data. This often makes it easier to build a system for task T than to reliably evaluate capability T!

As we scaled up the project, we began having to collect labels on multiple solutions for the same training problem. In order to avoid the risk of over-fitting on the 7,500 MATH training problems, we expanded the training set to include 4,500 MATH test split problems. We therefore evaluate our models only on the remaining 500 held-out problems. We selected these 500 test problems uniformly at random. In Figure 5, we show that the distribution of difficulty levels and subjects in this subset is representative of the MATH test set as a whole. The specific test set we used can be found at https://github.com/openai/prm800k.

Let's Verify Step by Step (2023)

Scaling Law

If you had a given budget of compute, what model would you train on how much data?

So many decisions to make…

- I have a fixed GPU budget. Make model larger? Or crawl more data?
- **Data:** Which of several different data mixtures do I use?
- **Architecture:** Do I train a Transformer, a *modified* Transformer, an LSTM?
- Other hyperparams too, like optimizer, depth/width, learning rate, …

- Idea #1: Largest model!
  - Train 50 models and pick best on dev! *Challenges?*
  - Then you can only train with 1/50 of your budget…
  - Are 50 runs even enough to cover your different choices?

Idea #2: Tune hyperparams at small scale!

- Then just copy them over and run them at large scale.
- *Challenge?*

- … turns out they don't transfer so easily! Best choice at small scale may not be the same at larger scale. (E.g., inductive biases or LR)

- This also doesn't tell you how large to make your choices.

Idea #3: Study "Scaling Laws"

- Train and tune many **smaller models** (starting from what choices?)
- Plot your loss vs. FLOPs (cost) as you scale up, on a log-log plot.
- *Extrapolate, for each training scheme??*
- *Given a target loss, find the "cheapest" way to get it!*

Learning Curves: Asymptotic Values and Rate of Convergence

Corinna Cortes, L. D. Jackel, Sara A. Solla, Vladimir Vapnik, and John S. Denker
AT&T Bell Laboratories
Holmdel, NJ 07733

Training classifiers on large databases is computationally demanding. It is desirable to develop efficient procedures for a reliable prediction of a classifier's suitability for implementing a given task, so that resources can be assigned to the most promising candidates or freed for exploring new classifier candidates. We propose such

Scaling laws

minimize *test-loss(model size, data, batch size, steps, ...)* **such that** *compute* **within budget**

for a given architecture, how much data / params would be needed for a target performance?

can we extrapolate from smaller to larger models / experiments??

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } \mathrm{FLOPs}(N,D)=C}{\operatorname{argmin}} L(N, D)$$

How does the test loss scale with data and params?

- decoder-only Transformer ("GPT")
- data: WebText2, 96GB of text, $2.29 \cdot 10^{10}$ tokens
  1024 token context

- vary model size: 768 - 1.5 billion non-embedding parameters
- and data size: 22M - 23B tokens, among other axes

- compute = parameters x batch x steps

Key Finding: power law relationships

$$L(X) = (1/X)^{\alpha} = X^{-\alpha}$$

test loss

resource (data/parameters/ compute)

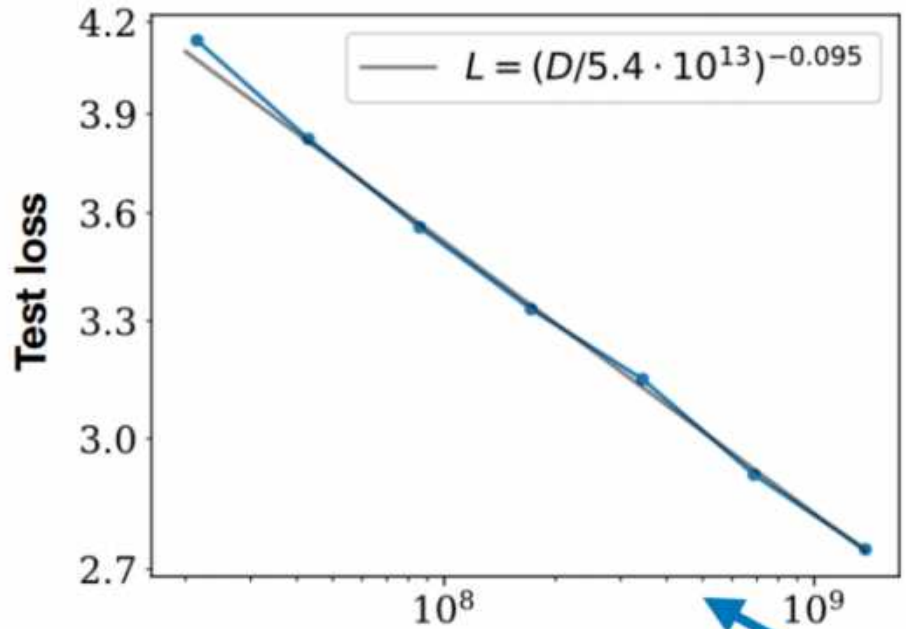
But what's the value of alpha?

Varies with your pre-training design.

- scale X -> 2X => Loss -> $2^{-\alpha}$ Loss

How does the test loss scale as a function of data size?

If *parameters* and total compute are too large to be bottlenecks ("infinite")…

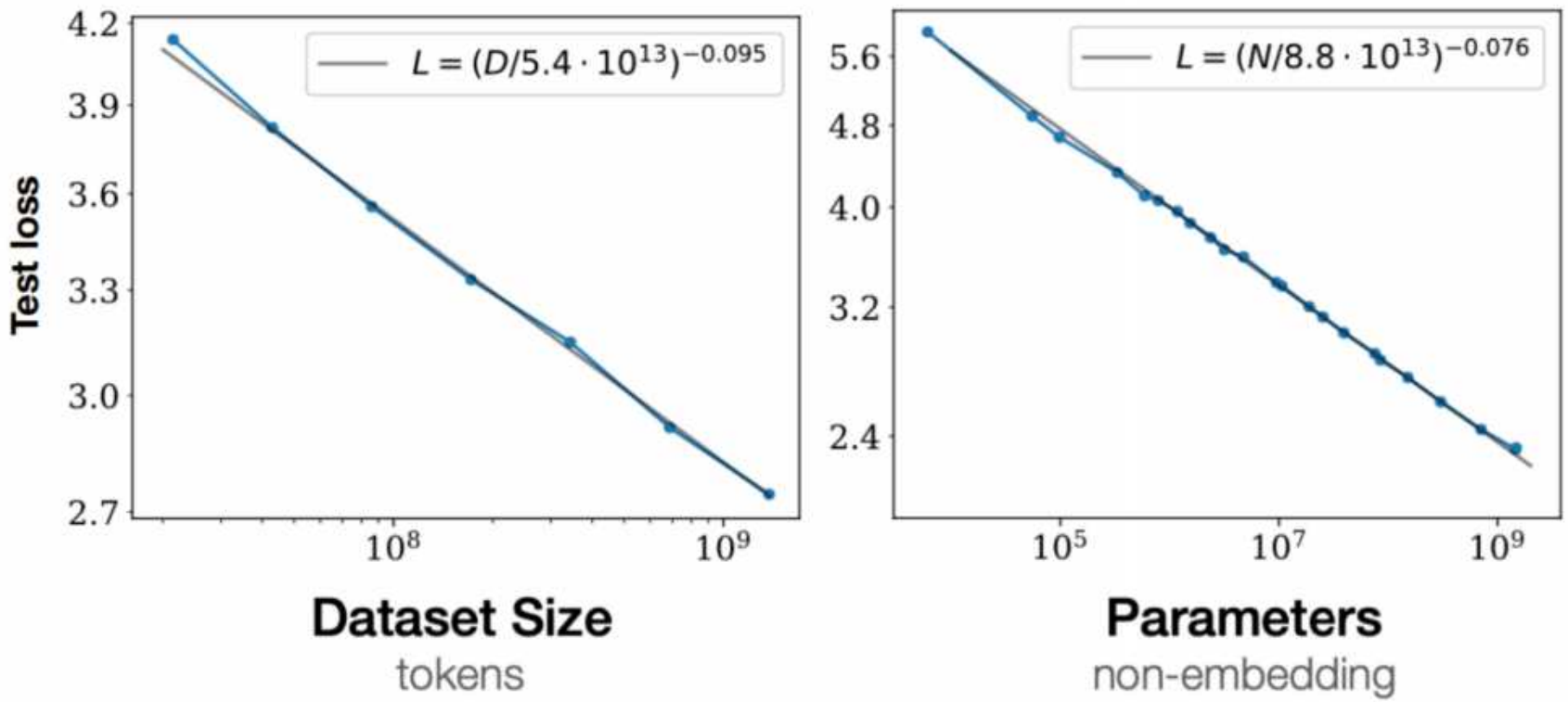


actually log scale too
log(L - irreducible error)

log scale

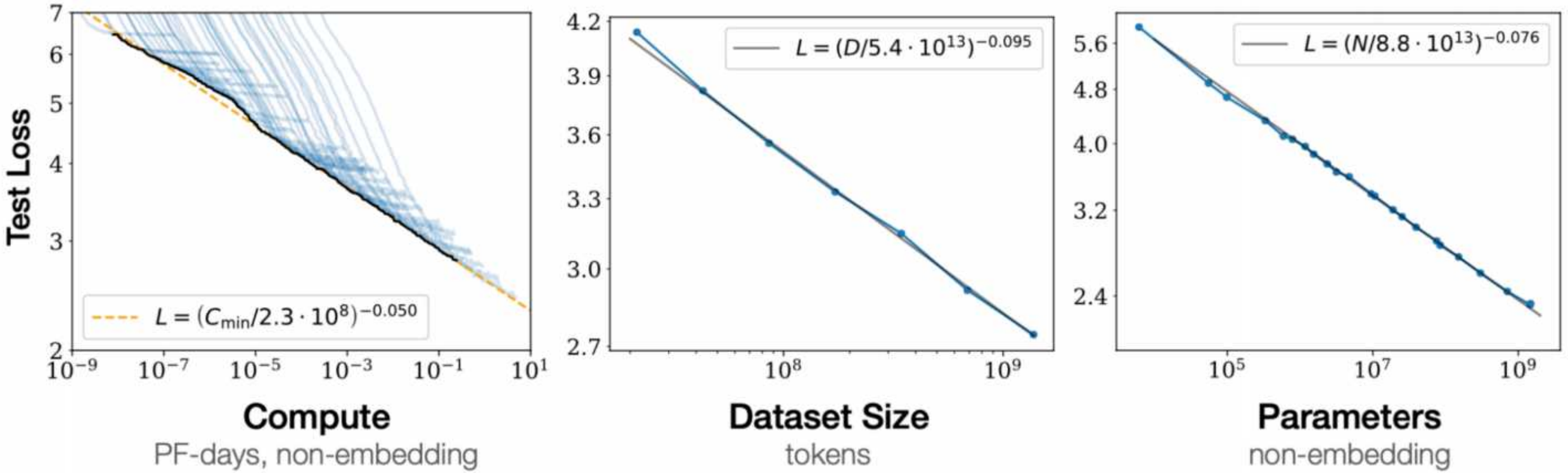
How does the test loss scale with number of params?

If *tokens* and total compute are too large to be bottlenecks ("infinite")…



How does the test loss scale with available compute?

- power law too, if you allocate compute "optimally"



Takeaways of Kaplan et al.

- test loss scales as power law of data, parameter count, and compute
- test loss is more sensitive to parameter count than data: with more compute, increase model by a larger factor than dataset size (!)
- larger models require less data to reach same performance, so don't train to convergence — instead: train larger models for fewer steps

This popularized a very different and new way of thinking.
Is model (# of parameters) scale pretty much all you need??

(No, but it was a very useful illusion in some sense!)

Scaling Laws caveats··· Kaplan’20 vs. Chinchilla’22

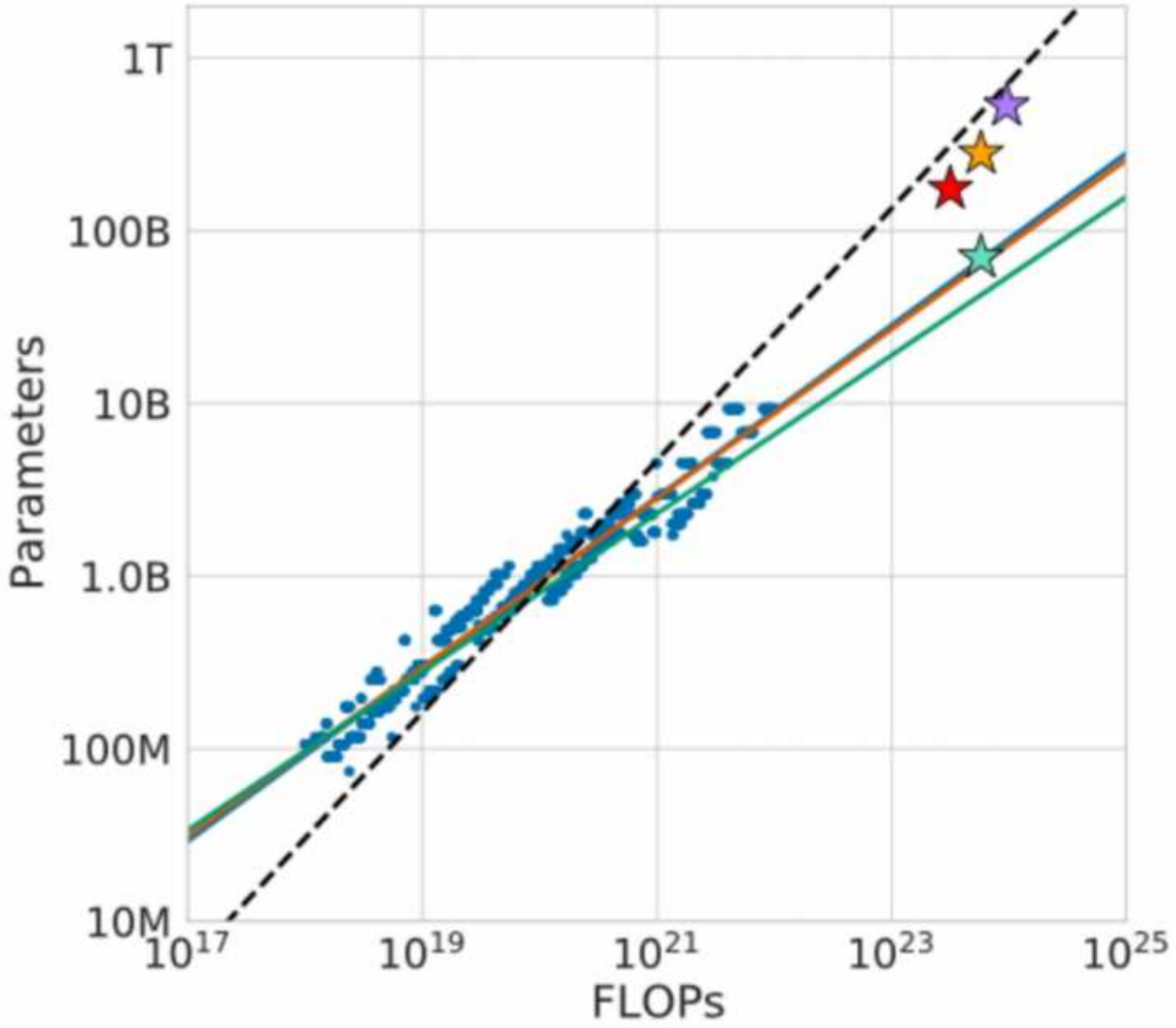
DeepMind confirmed the power-law scaling but disputed the optimal allocation between data and parameters. Scale them equally! (sqrt of compute)

Details: They fixed some hyperparameter decisions (data-dependent learning rate schedule) and experimented with much larger models (up to 16B in scale). They also counted parameters a little differently (embedding parameters counted).

“Fixed training budget” vs. deployment.
There’s a shift towards *overtraining*!

- **GPT3** – 2 tokens / param
- **Chinchilla** – 20 tokens / param
- **LLaMA65B** – 22 tokens / param
- **Llama 2 70B** – 29 tokens / param
- **Mistral 7B** – 110 tokens / param
- **Llama 3 70B** – 215 tokens / param

Predictions: extrapolation



For a given #FLOPs, how many parameters should we allocate? (compute = 6 x data x params)

Approach 1
Approach 2
Approach 3
Kaplan et al (2020)

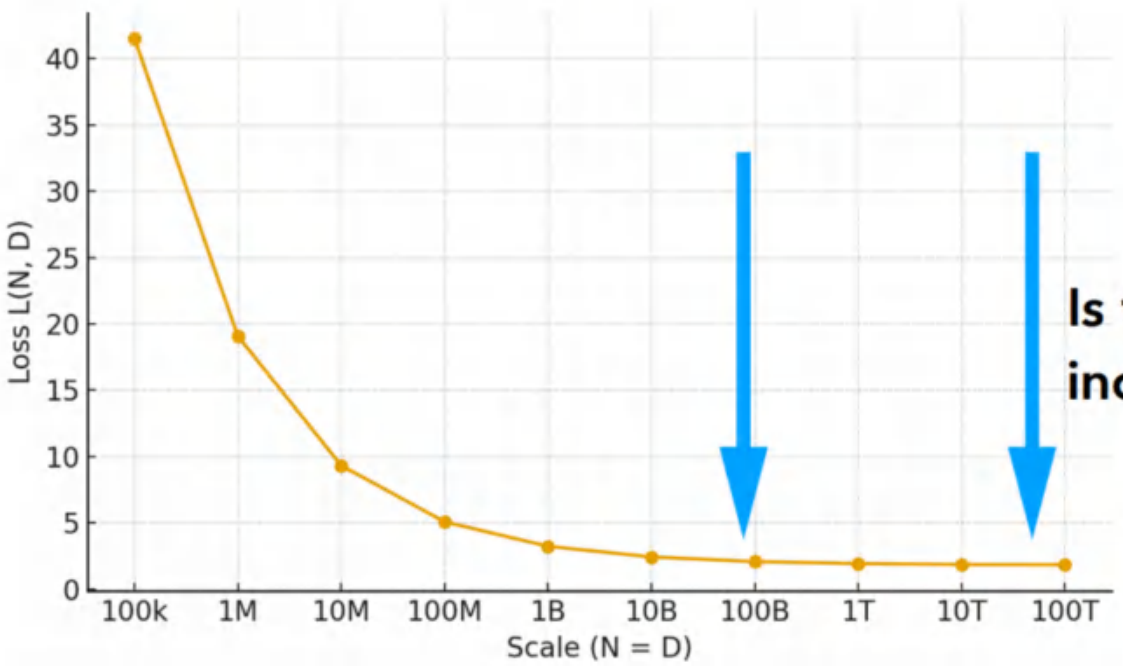
Chinchilla (70B)
Gopher (280B)
GPT-3 (175B)
Megatron-Turing NLG (530B)

Suppose scaling laws hold indefinitely

A power law means **diminishing returns.**

$$L(N, D) = 1.8172 + \frac{482.01}{N^{0.3478}} + \frac{2085.43}{D^{0.3658}}$$

Chinchilla Scaling: A replication attempt (2024)



Is this worth 1000^2 increase in compute?

N (or D)	L(N, D)
100k	41.525437
1M	19.080769
10M	9.324990
100M	5.083290
1B	3.238487
10B	2.435894
100B	2.086609
1T	1.934553
10T	1.868336
100T	1.839490

Chinchilla Scaling Law 的核心原理

路线 A

更大的模型 N
但训练 token D 较少

传统直觉：参数越多越强

路线 B

较小的模型 N
但用更多 token D 训练

Chinchilla：参数与数据要配平

固定算力预算 C

$C \approx 6ND$

模型容量

学习经历

Scaling Law：误差来自两个可缩放瓶颈

$$L(N,D) \approx L\infty + A/N^{\alpha} + B/D^{\beta}$$

$L\infty$
不可约误差
语言本身的复杂性

A/N^{α}
参数瓶颈
模型表达能力不足

B/D^{β}
数据瓶颈
学习经验不足

核心直觉：loss 的下降不是单一变量驱动，而是 N 与 D 的共同边际收益。

算力约束：C≈6ND，把问题变成优化

优化形式　　$\min L(N,D)$　s.t.　$C \approx 6ND$

- 固定预算下，N 与 D 不是独立变量
- 最佳点发生在两种误差的边际收益相近时
- 这就是“炼丹经验”背后的数学结构

给定 C，增大 N 就必须减少 D；增大 D 就必须缩小 N。

关键结论：模型与数据要近似同步增长

$N \times 2 \longrightarrow D \times 2$

for every doubling of model size,
training tokens should also double

经验规则

$$D \approx 20N$$

约 20 tokens / parameter

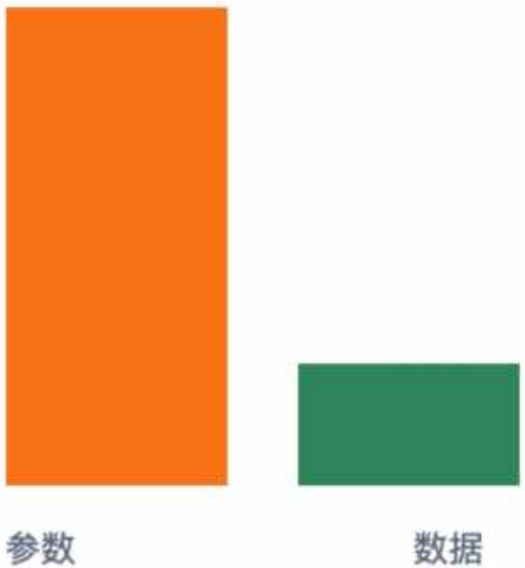
7B 参数	≈ 140B tokens	70B 参数	≈ 1.4T tokens

一句话：大脑容量变大，学习经历也要同步增长。

反直觉案例：更小的 Chinchilla 胜过更大的 Gopher

Gopher

280B 参数 · 约 300B tokens

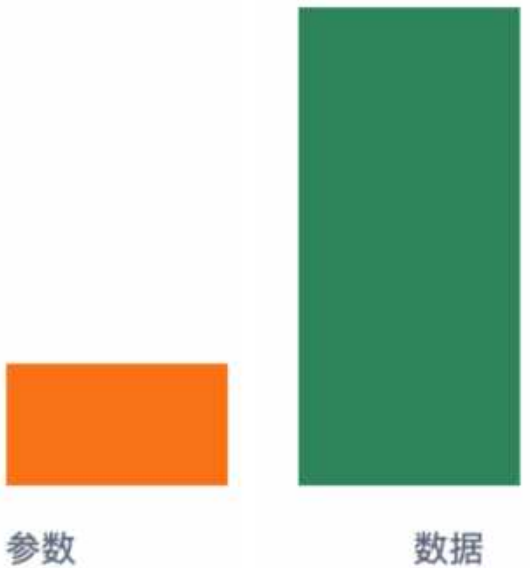


参数很大，但数据相对不足

same compute
better allocation

Chinchilla

70B 参数 · 约 1.4T tokens



参数更小，但训练更充分

它的数学之美在哪里？

幂律

能力增长可预测

优化

固定预算下选 N 与 D

Compute-Optimal Training

边际收益

参数与数据互相制约

科学范式

从经验炼丹到可估计规律

本节内容

CONTENTS

平时作业答案、大作业的评分标准

作业答案

- **Semantic Targets vs. Raw Pixels**
- **Combined losses**

$$\mathcal{L} = \mathcal{L}_{CLIP} + \mathcal{L}_{DINO} + \mathcal{L}_{iBOT++}$$

期末大作业

- **2026.7.12, 企业微信**
- **Submissions:**
  - ✓1. Technical report.
  - ✓2. Project source code.
  - ✓3. Demo video.
  - ✓4. Presentation PowerPoint presentation (5 pages).
  - ✓5. Team Report

评分标准

- ❑Novelty and Significance
- ❑Scientific Rigor and Technical Correctness
- ❑Multimodal Fusion
- ❑Formatting Requirements
- ❑Source Code
- ❑Demo Video
- ❑Presentation Slides
- ❑Team Report

最后

千山万水，不负理想，不忘初心，拨云见日，未来可期！

问题和讨论