Scaling Laws 与涌现能力：模型规模、计算与能力边界

2025 年 10 月 25 日

1 模型规模、数据量、计算量的幂律关系

1.1 经验幂律的提出与意义

OpenAI、DeepMind 等团队在大规模实验中发现：只要保持训练流程与数据分布一致，语言模型的损失随着模型参数 N、训练样本数 D、计算预算 C 呈现稳定的幂律（Power Law）下降。以测试困惑度（Perplexity）或交叉熵损失 $\mathcal{L}$ 为例：

$$\mathcal{L}(N) \approx A_N N^{-\alpha} + B, \quad \mathcal{L}(D) \approx A_D D^{-\beta} + B, \quad \mathcal{L}(C) \approx A_C C^{-\gamma} + B, \tag{1}$$

其中 $A_*, B, \alpha, \beta, \gamma$ 为常数。幂律体现了“规模 = 性能”的基本趋势，是分配计算资源、确定预训练策略的理论依据。

幂律关系的稳定性带来两个重要启示：

- **可预测性**：通过小规模实验拟合幂律指数，可外推更大规模模型的期望损失与所需计算量。
- **资源最优分配**：在总计算预算固定情况下，选择合适的模型大小与训练 tokens 数，可最大化性能。

1.2 Pythia/GPT 系列实验趋势

Pythia、GPT-3 等族谱验证了上述幂律，指出当数据量不足时模型进入“数据受限”阶段，参数继续增大收益减弱；反之，模型容量不足会成为“模型受限”瓶颈。图 ?? 以示意方式呈现不同受限区域。

示意图：左半区为数据受限，右半区为模型受限；对角线附近代表计算最优分配。

图 1: 幂律缩放示意：在双对数坐标下，损失沿幂律衰减。

1.3 计算预算与训练效率

综合考虑模型规模与数据量，可将训练 FLOPs 近似表示为

$$C \approx 6ND, \tag{2}$$

其中常数系数受网络深度、序列长度、并行策略影响。工程实践中常见的两种策略：

- **参数翻倍优先**：在推理为主的场景中倾向于增大模型，使困惑度下降带来更强能力。
- **数据扩展优先**：在线上持续学习场景，通过增加高质量数据与训练步数维持性能。

此外，复用冻结底座模型并在增量数据上微调，可显著降低额外计算成本。

2 Chinchilla Scaling Law

2.1 Chinchilla 论文核心结论

DeepMind 在 2022 年提出的 Chinchilla Scaling Law 对传统幂律做出修正：在固定计算预算下，最优策略是让训练 token 数 D 与模型参数 N 近似相等（$D \propto N$），而非 GPT-3 式的“参数远大于训练 token”。论文给出的最优损失公式：

$$\mathcal{L}(N, D) = \left(\frac{N}{N_0}\right)^{-0.34} + \left(\frac{D}{D_0}\right)^{-0.28} + \mathcal{L}_\infty, \tag{3}$$

并通过实验验证当 $D \approx 20N$ (以 token 计算) 时性能最佳。Chinchilla 模型以 70B 参数在近 1.4T token 上训练，性能超越更大但训练不足的 Gopher。

2.2 对资源规划的影响

Chinchilla Scaling Law 带来以下实操建议：

- **优先增加数据**：如果过去的训练只覆盖 $D \ll N$，应优先扩充高质量语料而不是盲目增加参数。
- **高效再训练**：将旧模型重训一次可能比增加参数或长时间微调更划算。
- **数据质量重要性**：数据集多样性、去重、过滤对最终性能影响更大，尤其在需要海量 token 的 regime。

Chinchilla 还强调了推理成本：较小但训练充分的模型不仅性能更好，而且推理延迟更低，有利于部署。

2.3 算法与硬件协同

要满足 Chinchilla 的 Data-Optimal 需求，需要在工程上实现高速数据流与分布式训练：

- **流水线并行 + 张量并行**：平衡参数与数据维度的通信负载。
- **异构存储**：利用 NVMe 缓存与分布式文件系统保证 token 读写速度。
- **数据增广与清洗流水线**：自适应采样、重复过滤、语种配比等策略避免冗余。

3 “涌现能力”案例：推理、记忆与组合泛化

3.1 涌现能力的定义与争议

“涌现能力”（Emergent Abilities）指模型在扩展至某一规模阈值后，突然在特定任务上表现出质的提升，如复杂推理、多步规划等。实践中常通过阈值效应或非线性指标观察：

$$\text{Ability} \approx \begin{cases} \text{baseline}, & N < N^\star \\ \text{rapid improvement}, & N \geq N^\star \end{cases} \tag{4}$$

然而，最近研究指出涌现可能是指标选择导致的“测量幻觉”。若采用平滑的损失函数（如对数失真）衡量能力，增长通常仍呈幂律连续变化。因此，需要精心设计评价指标，避免将离散准确率的跳变误认为真正的涌现。

3.2 推理（Reasoning）能力

案例包括多步算术、符号推理、链式思考（Chain-of-Thought, CoT）：

- **算术推理**：在 GSM8K、SVAMP 等数据集上，大模型通过 CoT 提示实现接近小学甚至初中水平的分步解题。
- **程序辅助推理**：结合外部工具（Python 解释器）执行算术，可显著提升正确率，显示出“工具使用”能力。
- **多跳问答**： HotpotQA、StrategyQA 等任务要求跨段落推演，大模型在上下文检索与逻辑连续性方面表现显著优于小模型。

推理能力常通过自一致性（self-consistency）采样增强：对同一问题生成多条思路，汇聚最常见结论，从而平滑不可控的输出。

3.3 记忆（Memory）与长期上下文

大型模型能够在极长上下文（>32k token）中准确检索与引用信息，体现出类记忆能力：

- **内隐记忆**：模型从预训练数据中存储事实，如 API 调用、歌曲歌词。Chinchilla 最优策略下的充分训练有助于巩固这些记忆。
- **外显记忆**：通过检索增强（RAG）、工具调用、插件系统，在推理阶段查询数据库，实现动态记忆。
- **长上下文测试**：Needle-in-a-Haystack、Long Range Arena 等基准衡量模型在海量文本中定位关键信息的能力。

记忆表现与分词策略、KV Cache 设计相关；缓存压缩、相对位置编码、分块注意力等技术对长上下文至关重要。

3.4 组合泛化（Compositional Generalization）

组合泛化指模型在训练未覆盖的组合上仍能正确执行任务。例如，在 SCAN、COGS、PCFG 等数据集上，模型需组合已学动作/语法以完成新任务。大型语言模型的表现要点：

- **少样本学习**：通过 prompt 中的示例组合出新的逻辑结构，展示“类系统 2”推理迹象。
- **思维树（Tree-of-Thought, ToT）**：在搜索树中探索不同子计划，再合成最终答案，提高解题稳健性。
- **程序合成**： Codex、AlphaCode 等系统在编程竞赛题上展示组合泛化能力，能够综合多个子函数与数据结构。

3.5 评估与监测工具

为了追踪涌现能力，需要构建细粒度的评估矩阵：

- **Benchmark 套件**： BIG-Bench、MMLU、AGIEval 汇集语言、逻辑、专业知识等多维任务，观察性能曲线。
- **能力触发实验**：比较不同规模模型在特定任务上的“阈值点”，并使用平滑指标验证增长是否真正非线性。
- **任务标签与元数据**：为每个任务记录思维类型、步骤长度、外部工具需求，以分析能力涌现的模式。

4 附录：幂律拟合示例代码

Listing 1: 根据实验点拟合规模幂律指数的示例

```python
import numpy as np
from scipy import stats

params = np.array([1e8, 3e8, 1e9, 3e9])
loss = np.array([3.1, 2.7, 2.4, 2.2])  # 以困惑度或交叉熵为例

log_params = np.log10(params)
slope, intercept, r, p, stderr = stats.linregress(log_params, loss)

alpha = -slope
print(f"拟合到的幂律指数 alpha   {alpha:.3f}")
print(f"模型损失近似 L(N)   10^{intercept:.3f} * N^-{alpha:.3f}")
```

5 工程实践建议

- **实验规划：**先在小规模上采样多个组合 (N, D)，拟合幂律后再决定大规模训练配置。
- **数据治理：**针对目标能力设计数据配方（math/code/多语种），并持续监测能力曲线。
- **对齐策略：**对于推理、记忆和组合泛化任务，在基础模型上叠加指令微调、CoT 提示、工具链使用以稳定能力输出。

延伸阅读

- Kaplan et al. “Scaling Laws for Neural Language Models.” arXiv:2001.08361, 2020.
- Hoffmann et al. “Training Compute-Optimal Large Language Models.” arXiv:2203.15556, 2022.
- Wei et al. “Emergent Abilities of Large Language Models.” arXiv:2206.07682, 2022.
- Schaeffer et al. “Are Emergent Abilities of Large Language Models a Mirage?” arXiv:2304.15004, 2023.
- Srivastava et al. “Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models.” BIG-Bench, 2022.