# Cramming: Training a Language Model on a Single GPU in One Day

University of Maryland, College Park

研究在单个消费级GPU上训练BERT一天可实现的下游性能。

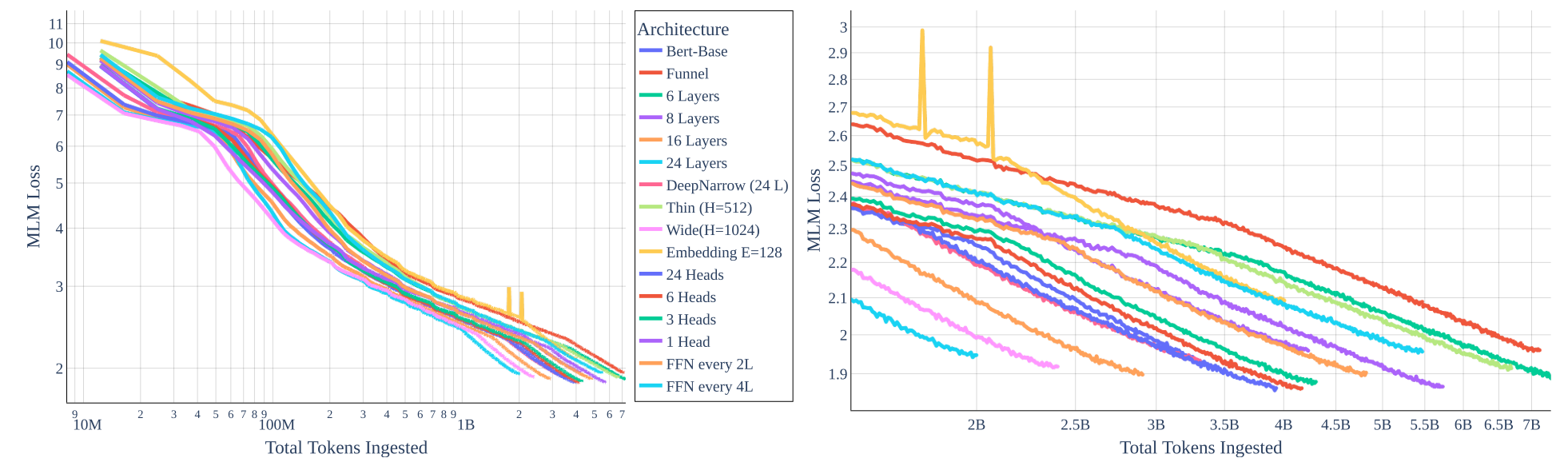
使用经典rtx2080ti GPU（2018年9月发布）的设置和使用更现代的rtxa4000或rtxa6000 GPU（2020年10月发布），将每个GPU与4个CPU核心和32GB RAM配对。

零、深度学习框架

允许自动操作符融合[1]，其可应用于所有组件。在选择最终的架构变体之后，重新启用高效注意内核[2]。对于标准的16位和32位浮点精度，使用相同的自动混合精度[3]设置。（在全32位，Scaled-16位[4]和bfloat-16[5]，ZeRO-Offload[6]没有益处。）

一、模型架构的探索

Scaling-Law[7]是缩小模型规模的一个障碍。模型Per-token的训练效率强烈地依赖于模型大小，而不是Transformer的类型（例如不同的架构）。较小的模型学习效率较低；但是在相同大小的模型中，训练效率几乎是恒定的，这意味着可以通过架构的修改来提高性能，从而在保持参数几乎恒定的同时加快梯度计算。



模型类型和尺寸的变化对24小时训练后的最终损失的影响极小。具有更多参数的模型学习效率更高，因为它们的损失降低得更快。然而，较小的体系结构通过较高的吞吐量来弥补其较慢的学习效率，并因此在有限的预算上处理更多的Tokens。

不同的架构在整个训练初始阶段（前1B个Tokens）都是不可预测的，之后每个Token的效率仅相差一个乘法常数（对数轴引起的水平偏移×该常数）。该常数几乎完全取决于模型大小，而不是模型类型。

做出的一些改动：

1. Attention Block

1. 禁用QKV偏置[8]通过移除一层计算来利用缩放定律，使得向前和向后传递稍微快一些，同时保持模型大小几乎恒定。

2. 减少注意力头的数量[9-12]在GPU上并行化更好，并提供了轻微的性能提升。然而实验发现，减少注意力头数量也会降低微调性能，因此最终仍保留所有12个注意力头。

3. 此外验证了一些其他机制无效，其他方面仍采用原注意力机制。

1. Feedforward Block

1. 禁用所有线性层偏置[8]，与Attention Block一致。

2. 改变为GELU以外的任何激活函数没有任何益处。

3. 将Block重新排序为门控线性单元[13]有小改进。

4. 不增加Block中的参数[14]来补偿由于门控线性单元[13]而导致的隐藏维度的减半。

1. Embedding

1. Scaled-正弦函数位置嵌入[15]，相比传统的位置嵌入有优势。

2. 在Embedding层的最后加一个Layer normalization。

1. Layer Structure

1. 禁用所有线性层偏置[8]，与Attention Block一致。

2. 使用Layer normalization进行Pre-normalization比使用Layer normalization进行Post-normalization更有益[16-17]。

3. Pre-normalization的关键作用是稳定训练、允许使用更大的学习率，减少warmup，单独使用Pre-normalization的好处有限。

1. Head Block

1. 禁用非线性注意力头，对性能没有影响。

2. 禁用解码器偏置[18]，有优势。

3. 使用sparse token prediction[19-20]，对存储有优势。

二、训练设置的探索

研究了训练超参数对BERT-base结构的影响。原始BERT训练方法在论文的设置中导致了很差的模型性能，因此需要重新考虑一些超参数选择。

1. Objective

使用BERT-base的掩码设置，15%掩码率；所有掩码的10%填充有随机词，10%不变。

1. Optimizer

选用Adam优化器，为了稳定训练，以0.5概率裁剪梯度。

1. Learning Rate

单周期学习率[14]，峰值学习率为0.001，可以使训练前的损失最小。

1. Batch Size

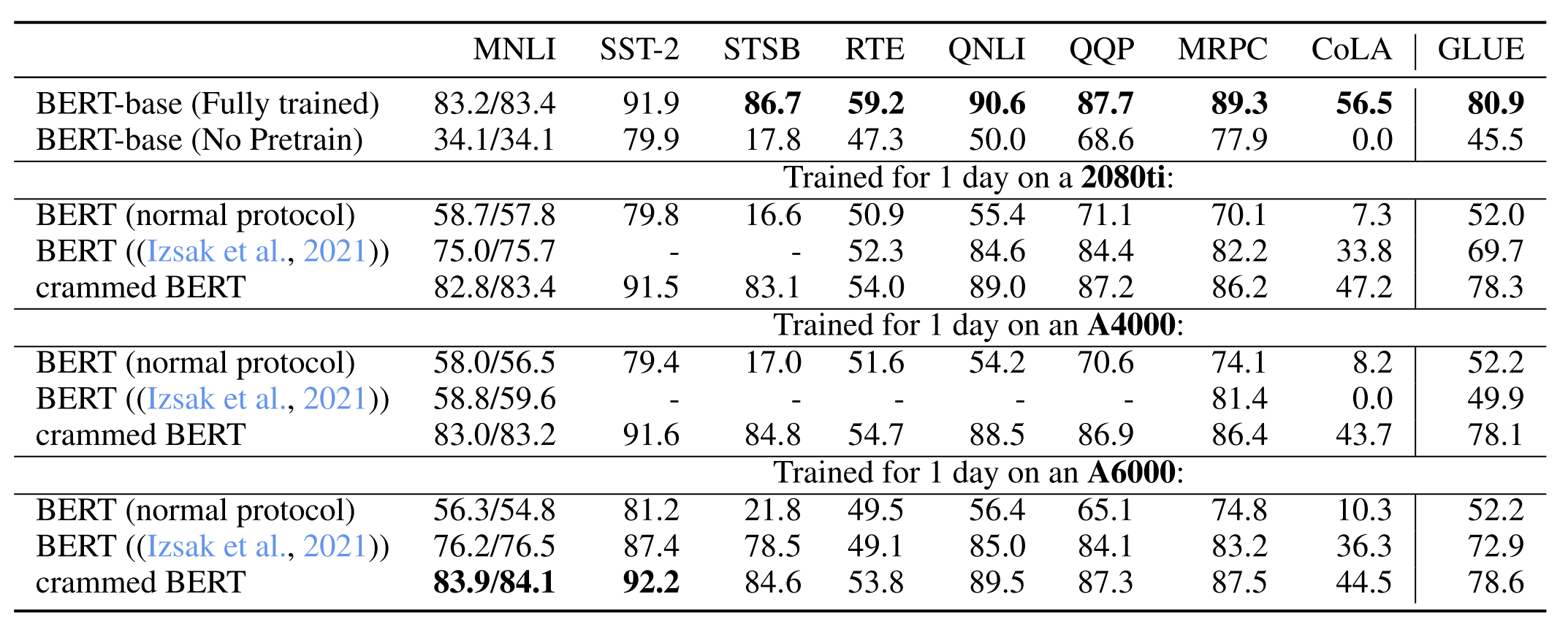
受限于单个GPU，因此选择的批量大小（大多数实验为96）比最佳批量大小小几倍。对于2080ti，最佳批量大小约为1536时预训练损失最小，但批量大小为4032时下游性能最大。针对这两种优势，分别每16次和42次前向/反向传递执行一次权值更新。

1. Dropping Dropout

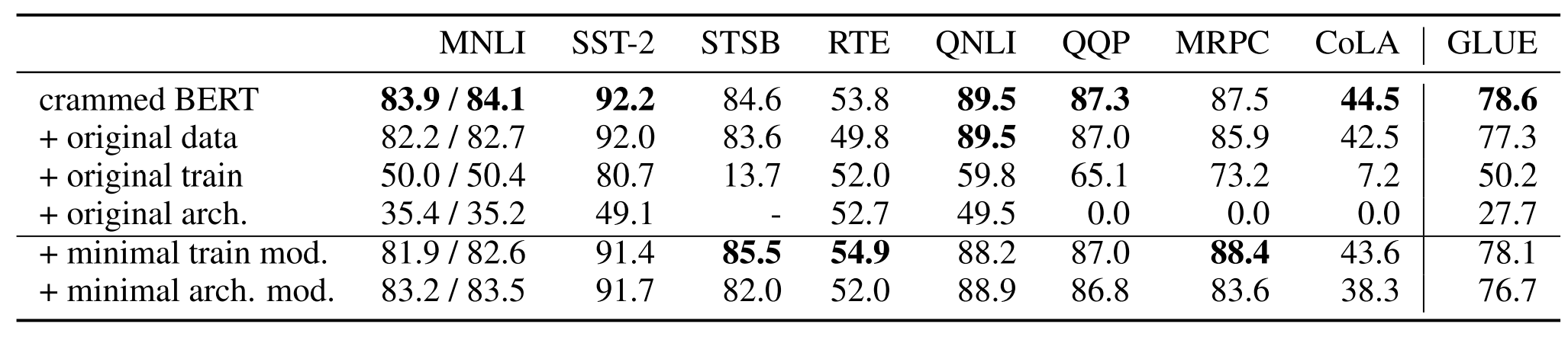
预训练期间禁用了Dropout，以最大化更新权值。但微调阶段采用0.1的Dropout Rate，此时数据量小容易过拟合。

三、数据集的探索

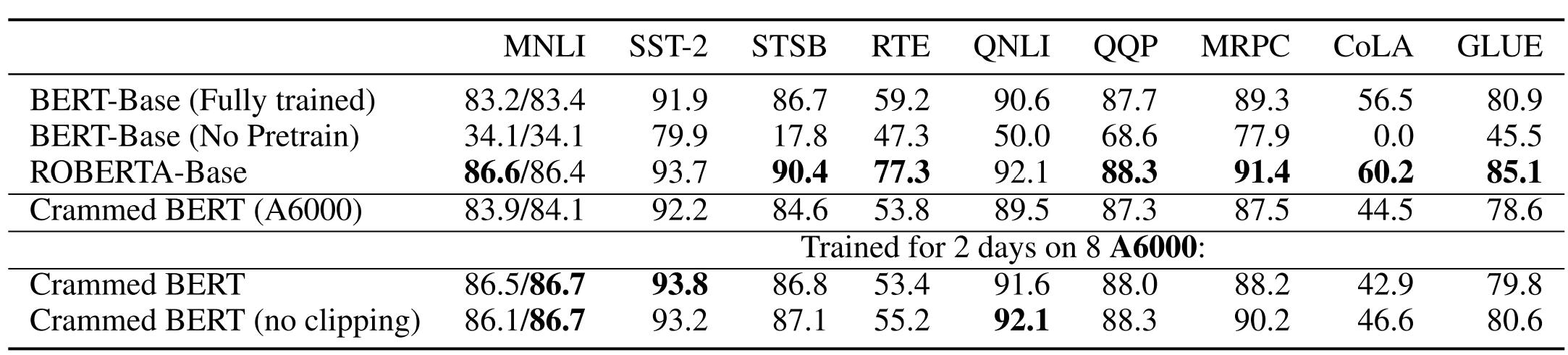
Crammed BERT的精度：



消融测试：



模型参数更大：



[1] Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zach DeVito, Zeming Lin, Alban Desmaison, Luca Antiga and Adam Lerer. “Automatic differentiation in PyTorch.” (2017).

[2] Dao, Tri, Daniel Y. Fu, Stefano Ermon, Atri Rudra and Christopher R'e. “FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness.” ArXiv abs/2205.14135 (2022).

[3] Micikevicius, Paulius, Sharan Narang, Jonah Alben, Gregory Frederick Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh and Hao Wu. “Mixed Precision Training.” ArXiv abs/1710.03740 (2017).

[4] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20). Association for Computing Machinery, New York, NY, USA, 3505–3506.

[5] Shibo Wang and Pankaj Kanwar. BFloat16: The secret to high performance on Cloud TPUs, August 2019. <https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus/>

[6] Ren, Jie, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyang Yang, Minjia Zhang, Dong Li and Yuxiong He. “ZeRO-Offload: Democratizing Billion-Scale Model Training.” USENIX Annual Technical Conference (2021).

[7] Kaplan, Jared, Sam McCandlish, T. J. Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu and Dario Amodei. “Scaling Laws for Neural Language Models.” ArXiv abs/2001.08361 (2020).

[8] Boris Dayma, Suraj Patil, Pedro Cuenca, Khalid Saifullah, Tanishq Abraham, Phuc Le Khac, LukeMelas, and Ritobrata Ghosh. DALL·E Mini, July 2021. <https://github.com/borisdayma/dalle-mini>

[9] Merity, Stephen. “Single Headed Attention RNN: Stop Thinking With Your Head.” ArXiv abs/1911.11423 (2019).

[10] Araabi, Ali and Christof Monz. “Optimizing Transformer for Low-Resource Neural Machine Translation.” ArXiv abs/2011.02266 (2020).

[11] Liu, Liyuan, Jialu Liu and Jiawei Han. “Multi-head or Single-head? An Empirical Comparison for Transformer Training.” ArXiv abs/2106.09650 (2021).

[12] Javaheripi, Mojan, S. Shah, Subhabrata Mukherjee, Tomasz L. Religa, Caio Cesar Teodoro Mendes, Gustavo de Rosa, Sébastien Bubeck, Farinaz Koushanfar and Debadeepta Dey. “LiteTransformerSearch: Training-free On-device Search for Efficient Autoregressive Language Models.” ArXiv abs/2203.02094 (2022).

[13] Dauphin, Yann, Angela Fan, Michael Auli and David Grangier. “Language Modeling with Gated Convolutional Networks.” International Conference on Machine Learning (2016).

[14] Smith, Leslie N. and Nicholay Topin. “Super-convergence: very fast training of neural networks using large learning rates.” Defense + Commercial Sensing (2017).

[15] Hua, Weizhe, Zihang Dai, Hanxiao Liu and Quoc V. Le. “Transformer Quality in Linear Time.” International Conference on Machine Learning (2022).

[16] Baevski, Alexei and Michael Auli. “Adaptive Input Representations for Neural Language Modeling.” ArXiv abs/1809.10853 (2018).

[17] Xiong, Ruibin, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang and Tie-Yan Liu. “On Layer Normalization in the Transformer Architecture.” International Conference on Machine Learning (2020).

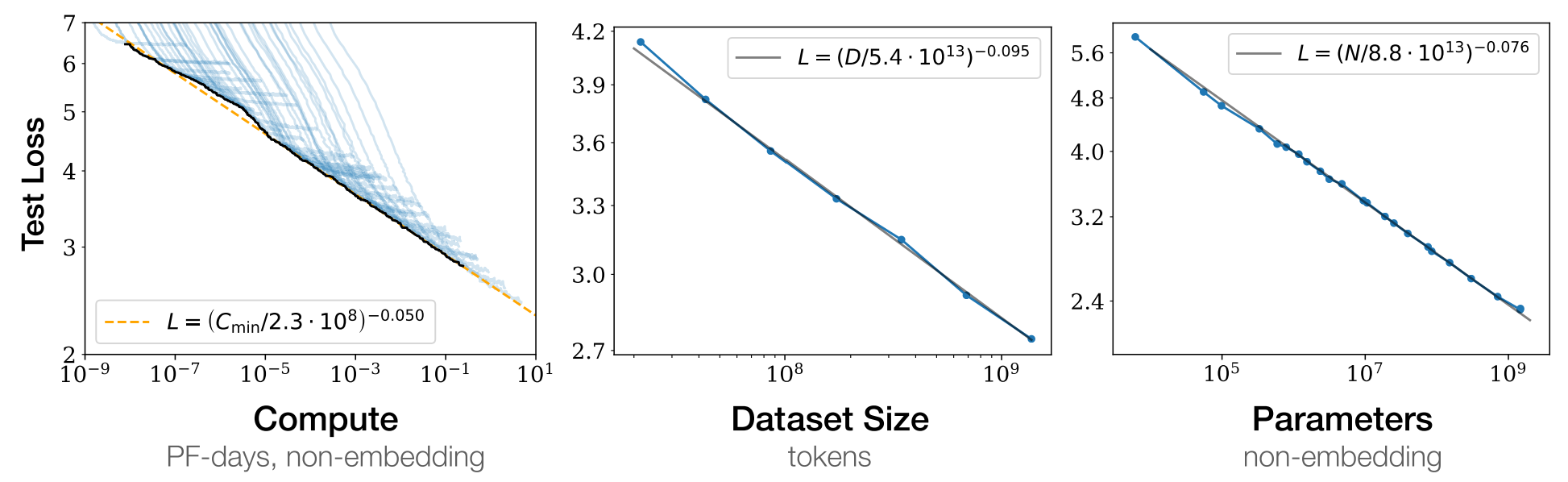
[18] Radford, Alec, Jeff Wu, Rewon Child, David Luan, Dario Amodei and Ilya Sutskever. “Language Models are Unsupervised Multitask Learners.” (2019).

[19] Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer and Veselin Stoyanov. “RoBERTa: A Robustly Optimized BERT Pretraining Approach.” ArXiv abs/1907.11692 (2019).

[20] Izsak, Peter, Moshe Berchansky and Omer Levy. “How to Train BERT with an Academic Budget.” Conference on Empirical Methods in Natural Language Processing (2021).

# Scaling Laws for Neural Language Models

Johns Hopkins University, OpenAI



（每秒执行10的15次方(1 peta)次运算并持续一天时间）

1. 性能在很大程度上取决于scale，在很小程度上取决于模型形状。

模型性能主要取决于scale，由三个因素组成：模型参数的数量N（不包括Embedding层的参数）、数据集的大小D以及用于训练的计算量C。在合理的范围内，性能对其他架构超参数（如注意力的深度与宽度）的依赖性非常弱。

1. 性能与三个比例因子N、D、C中的每一个都呈幂律关系（当不受其他两个瓶颈影响时）。
2. 过拟合具有普遍性。

只要同时增加N和D到相同比例，性能就会提高；但如果N或D保持固定，而另一个增加，则可能会进入收益递减的状态。性能损失取决于比率N^0.74/D，这意味着每次将模型参数增加8倍，只需将数据增加大约5倍即可避免损失。

1. 训练的可预测性。

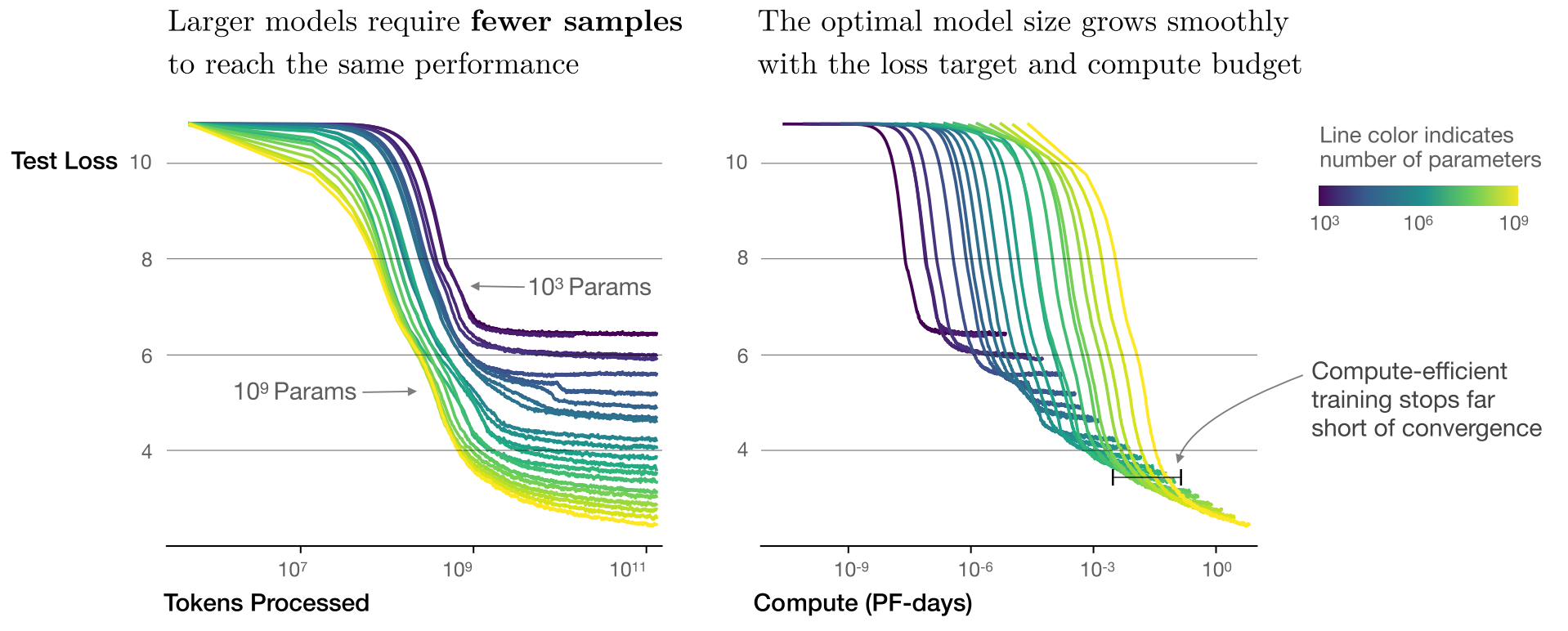
训练曲线遵循可预测的幂律，其参数大致与模型大小无关。通过推断训练曲线的早期部分，我们可以粗略地预测训练更长时间后会达到多少损失。

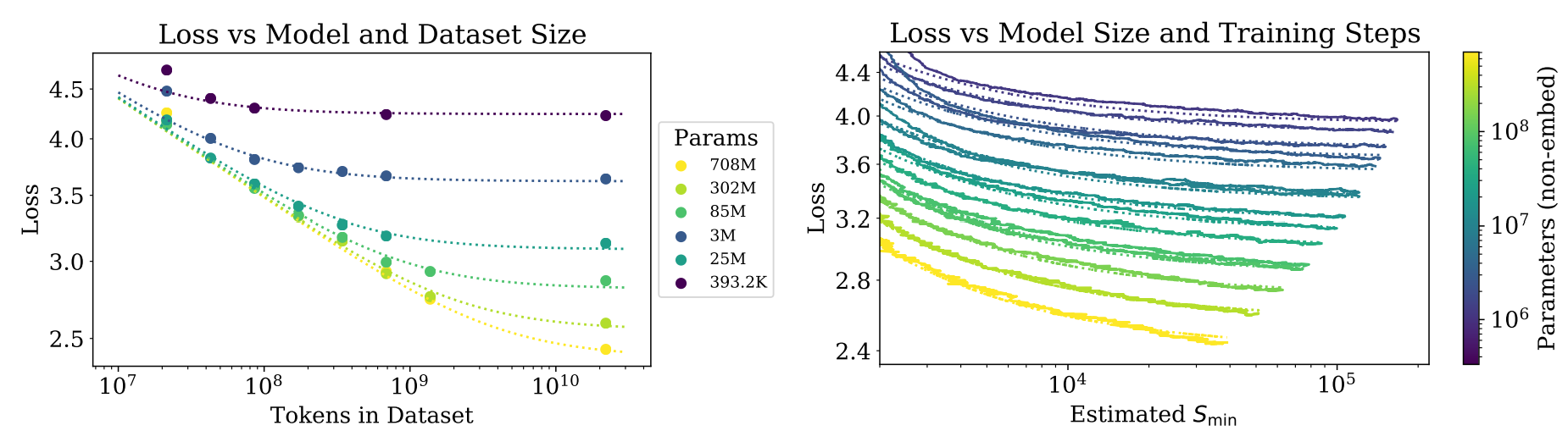
1. 迁移学习的性能随着训练集性能的提高而提高。

当我们在分布情况与训练时所用的分布不同的文本上验证模型时，结果与训练验证集上的结果强相关，损失的偏移量大致恒定。换句话说，迁移到不同的分布会导致恒定的损失，但在其他方面的改进大致与训练集上的性能一致。

1. 样本效率。

大型模型比小型模型的样本效率更高，可以用更少的优化步骤和更少的数据达到相同的性能水平。





1. 收敛效率低下。

当在固定计算预算C内工作，但对模型大小N或可用数据D没有任何其他限制时，通过训练非常大的模型并在明显未达到收敛时停止来实现最佳性能。对于最佳计算效率的训练，大部分增加应该用于增加模型大小。需要相对较小的数据增长以避免重复使用。在增加的数据中，大部分可用于通过较大的批量增加并行性，所需的串行训练时间仅增加非常少。

