Milestones of NLP

Vijeta Deshpande
Text Machine Lab, Umass Lowell

NL what?

Natural Language Processing (NLP) aims to make computers understand and generate natural languages. Loosely,

1. Understanding relates to converting text into a numerical form (referred as **encoding**).
2. Generation associates with using the numerical representation to predict next word (referred as **decoding**).

Milestones

01 Self-Supervision

02 Transformers

03 Double Descent

04 Scaling Laws

05 Efficient Attention

06 Quantization

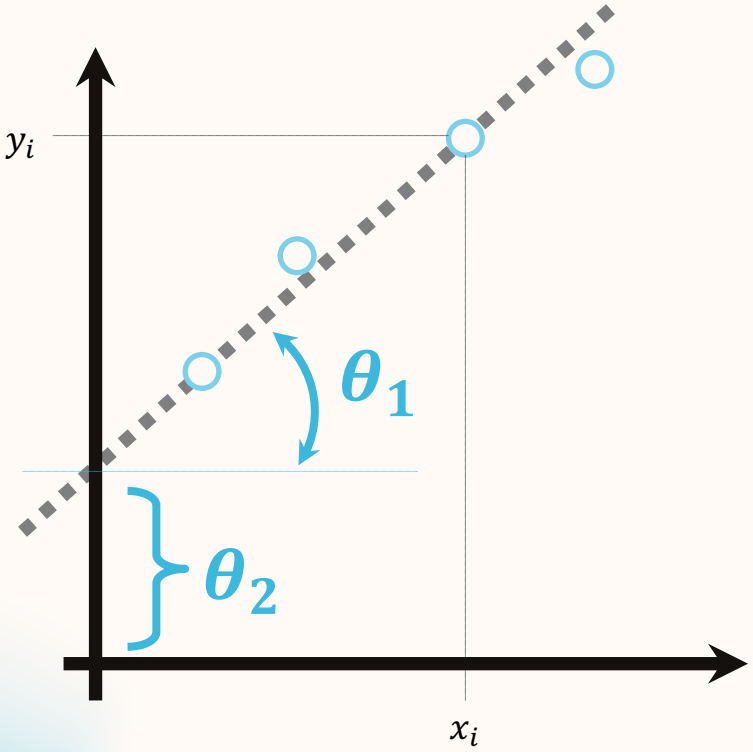
07 Efficient Fine-Tuning

08 Alignment

00

Finding θs

Linear Regression



Given the Data

We assume that some data is available to us, generally denoted with (x_i, y_i)

And the Hypothesis

We hypothesize that there exists some sort of mathematical relationship between (x, θ) and y. For example, $y = \theta_1 x + \theta_2$

Find θs

We want to find a set of θ values that can fetch us (nearly) correct y values of unseen x values (AKA unbiased estimator of y). Generally written as,

$$\arg_\theta \min(y - f(x;\theta))$$

A Non-linear Case

Change the Hypothesis

We make use of functions that can add non-linearity to the linear transformations (AKA activations), for example, sigmoid function.
Now,

$$y = \sigma(\theta_1 x + \theta_2)$$

Neural Networks

Recurring Operations

 # Recurring operations
$h = x$
For _ in num_layers:
$h = \sigma(\theta_1 h + \theta_2)$

 # final operation
$y = f(h)$

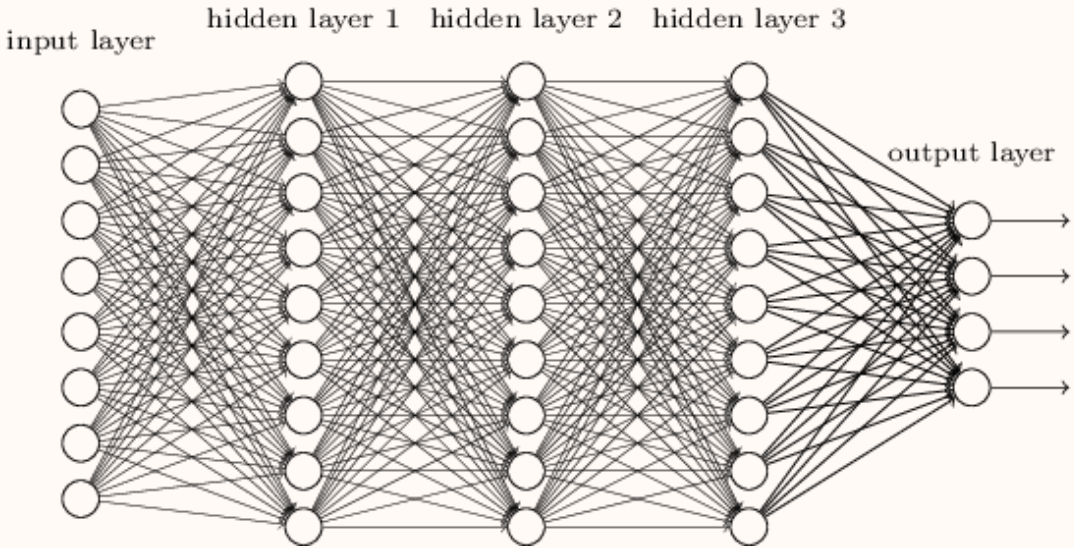


Image Source: Nielsen, M. A. (2015). Neural networks and deep learning (Vol. 25, pp. 15-24). San Francisco, CA, USA: Determination press.

Transformer-based LLMs

Recurring Operations

 # Recurring operations
$h = x$
For _ in num_layers:
$h = Layer\ Norm(h)$
$h = Attention(h)$
$h = Residual\ Connection(h)$
$h = Layer\ Norm(h)$
$h = FFN(h)$
$h = Residual\ Connection(h)$

 # final operation
$y = f(h)$

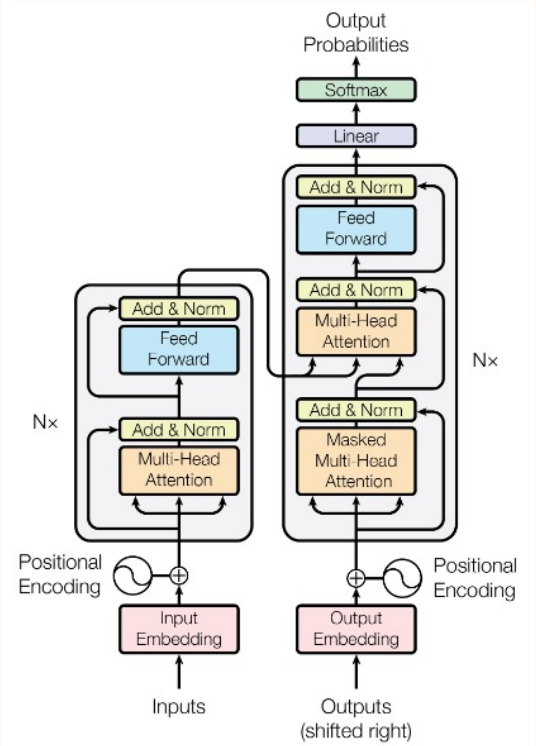


Figure 1: The Transformer - model architecture.

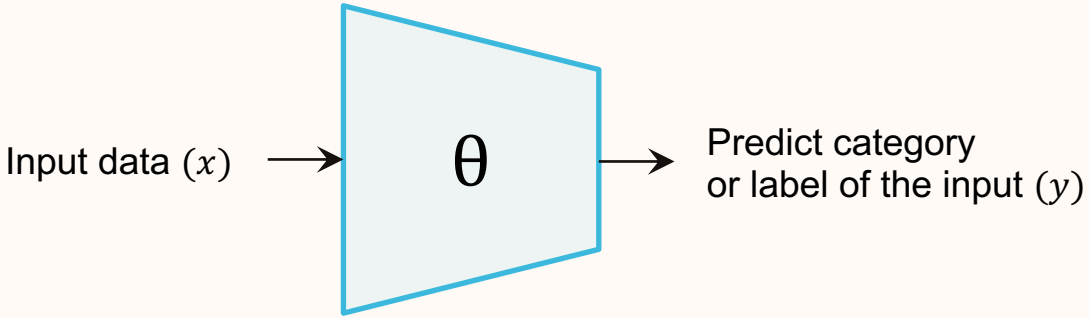
Image Source: Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.

01

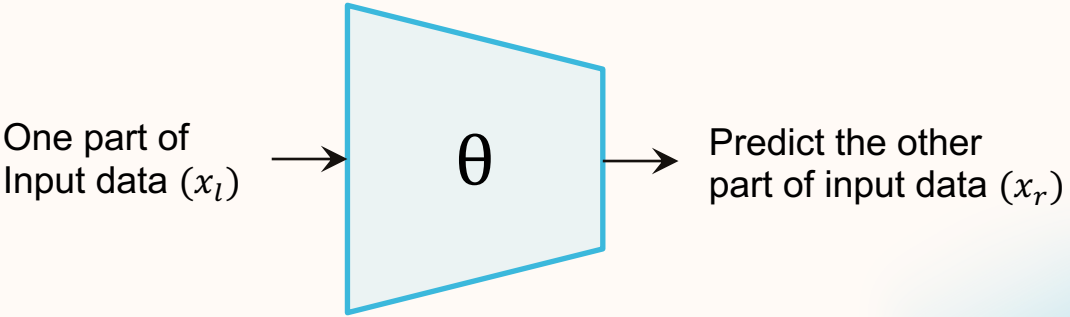
Self-Supervision

Types of Supervision

Supervised Learning



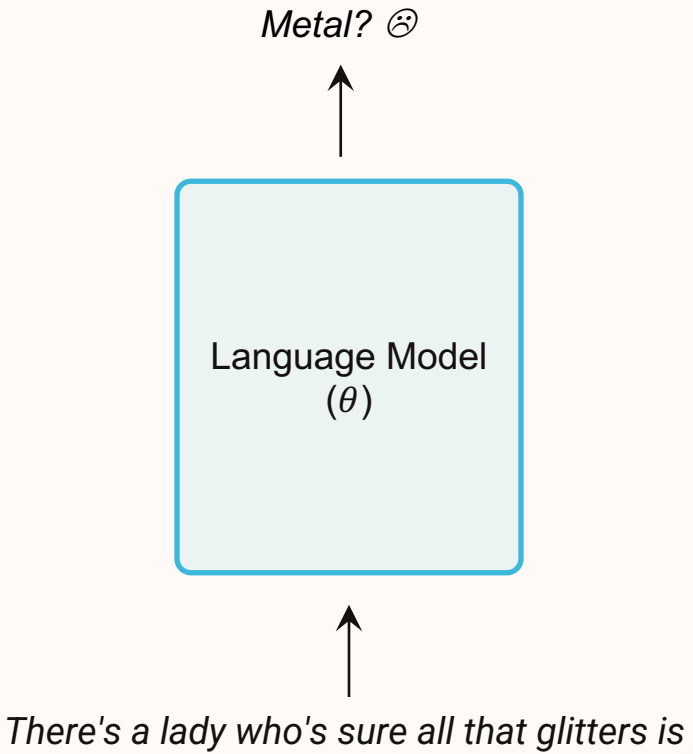
Self-Supervised Learning



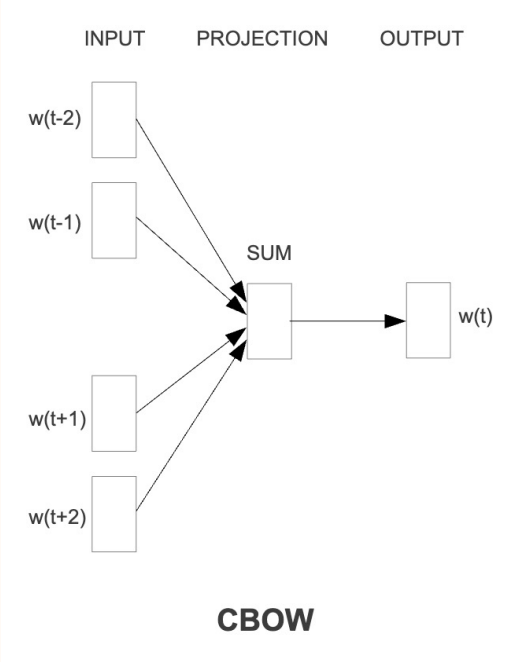
Self-Supervision in NLP

Self-Supervised Learning in NLP

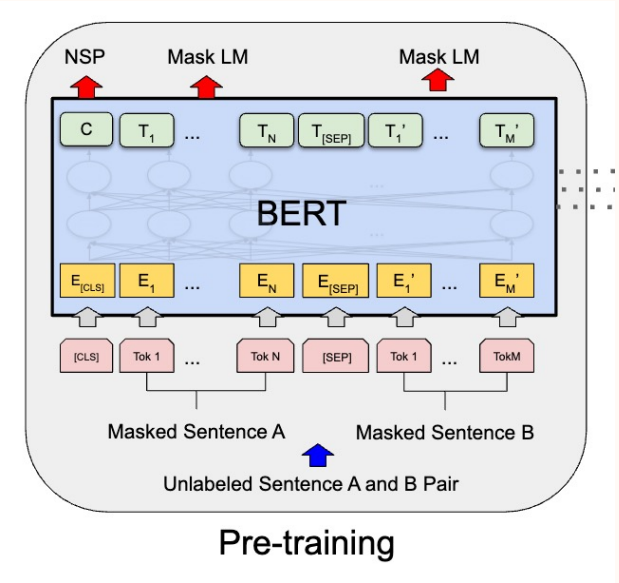
Provide part of sentence to the model as an input and predict rest of the sentence from the model. The goal is to learn representation of each word such that, given representations of a few words the model should be able to generate rest of words.



Self-Supervision in NLP



Continuous Bag of Words
https://arxiv.org/abs/1301.3781



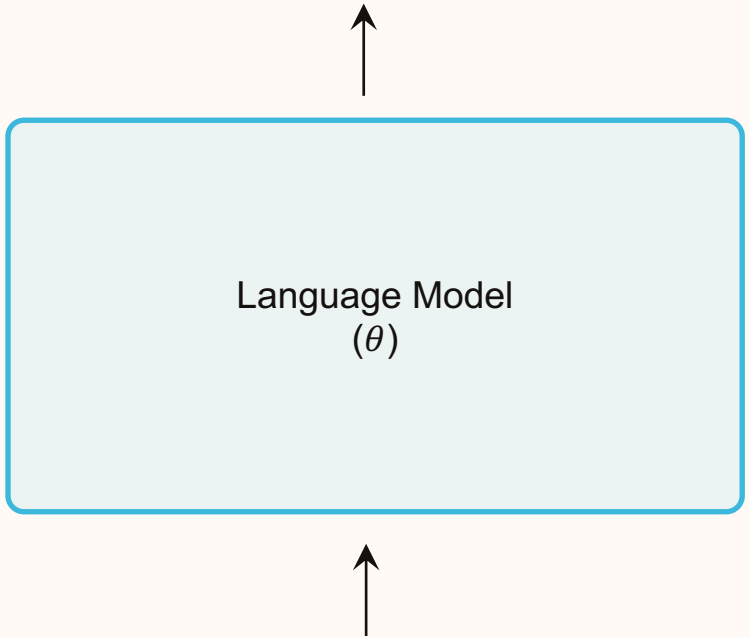
Masked Language Modeling
https://arxiv.org/abs/1810.04805

Self-Supervision in NLP

Next Word Prediction

This the most famous method (AKA causal language modeling) of training language models as of today. In this method, model predict the next word given the previous sequence of words,

$$p(x_i|x_i, \dots, x_0) = LLM(x_{i-1}, \dots, x_0|\theta)$$



Key Points

Self-Supervision

- ✓ Leverage massive amount of unlabeled data.
- ✓ Train larger models.
- ✓ Conversion of all tasks into natural language tasks, next word prediction.

02

Transformers

The Architecture

Operations

 # Recurring operations
$h = x$
For _ in num_layers:
$h = Layer\ Norm(h)$
$h = Attention(h)$
$h = Residual\ Connection(h)$
$h = Layer\ Norm(h)$
$h = FFN(h)$
$h = Residual\ Connection(h)$

 # final operation
$y = f(h)$

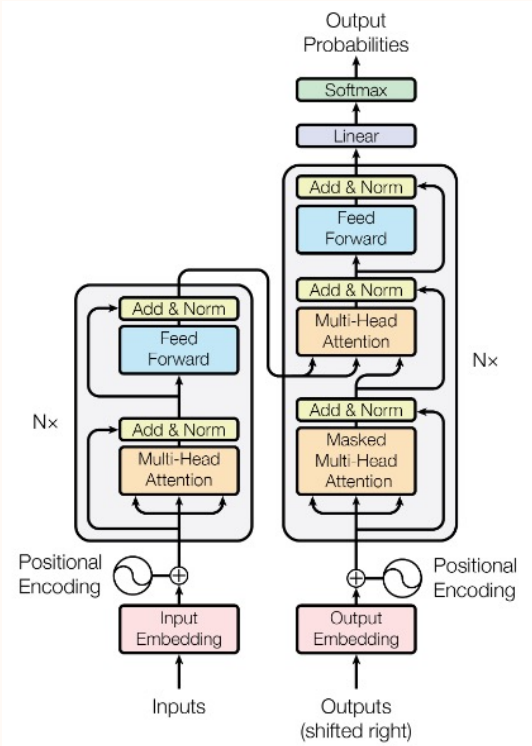


Figure 1: The Transformer - model architecture.

Image Source: Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.

Building Blocks of Transformers

Operation	Study	Year
$Attention$	Weighted representation of input sequence for translation https://arxiv.org/abs/1409.0473	2014
$Residual\ Connection$	Deeper networks need skip connections (residual connections) https://arxiv.org/abs/1512.03385	2015
$Layer\ Norm$	Normalization of activations along sequence length https://arxiv.org/abs/1607.06450	2016
$Activation$	Analysis of various activation functions that work better for LMs https://arxiv.org/abs/2002.05202	2020
$Position\ Embeddings$	Non-parametric, rotation-based, position encodings https://arxiv.org/abs/2104.09864	2021

Faster Training of Networks

RNNs
(sequential processing of sequence)

$h_m = h_{init}$
$h_{states} = []$
For word in sequence:
$h_m = f(word, h_m; \theta)$
$h_{states} \mathrel{+}= h_m$

Transformers
(parallel processing of sequence)

$h_{states} = f(sequence; \theta)$

Key Points

Transformers

- ✓ Parallel and faster pre-training of the models compared to LSTMs.

03

Double Descent

The Classical and the Modern Regimes

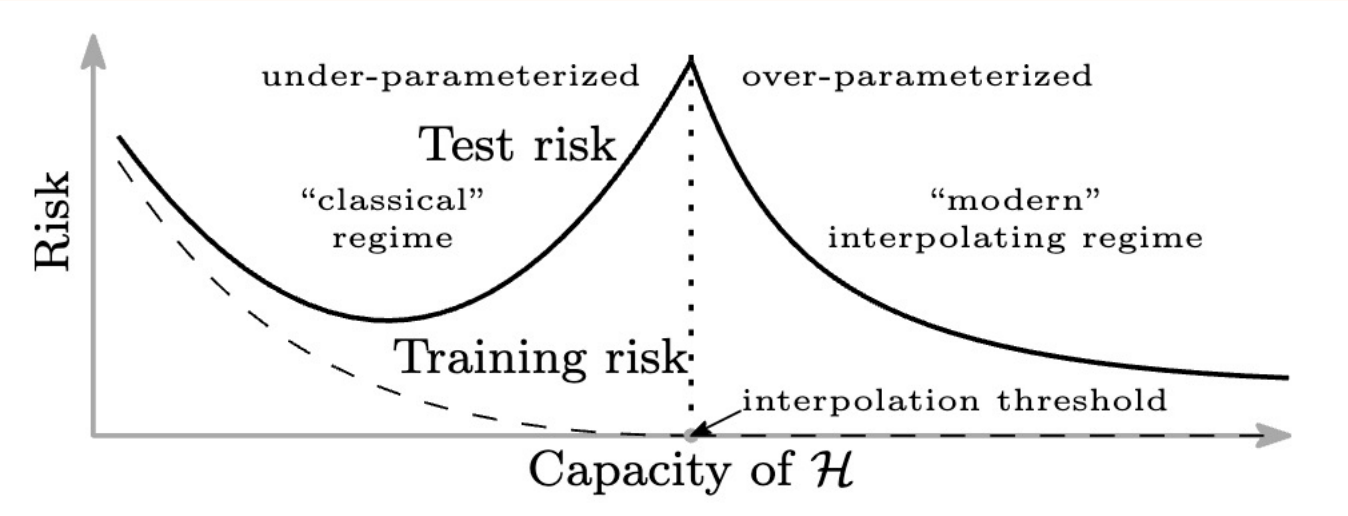


Image Source: Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. Proceedings of the National Academy of Sciences, 116(32), 15849-15854.

The Classical and the Modern Regimes

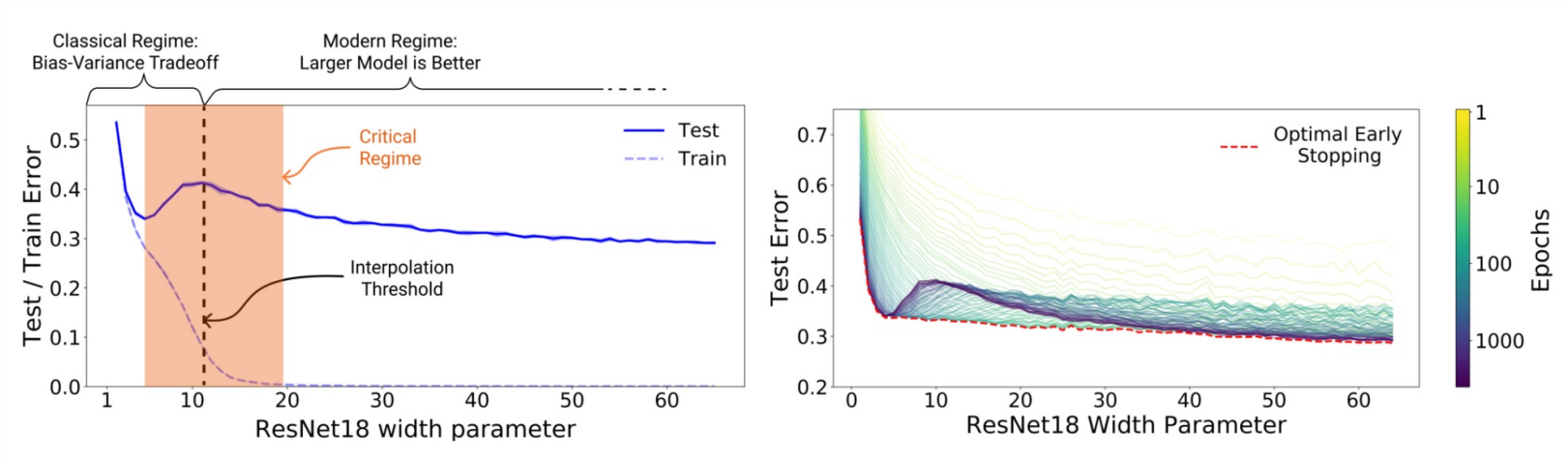
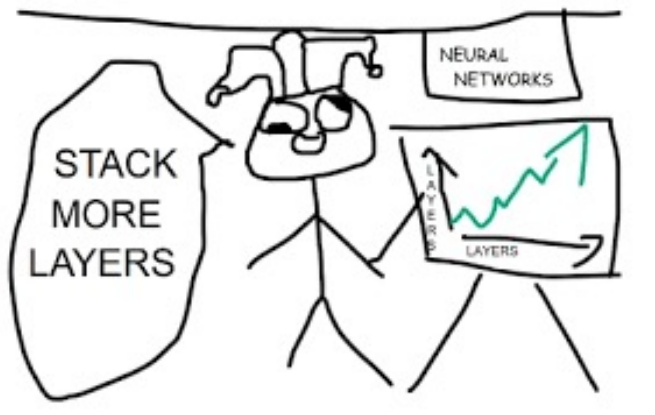


Image Source: Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., & Sutskever, I. (2021). Deep double descent: Where bigger models and more data hurt. Journal of Statistical Mechanics: Theory and Experiment, 2021(12), 124003.

Key Points

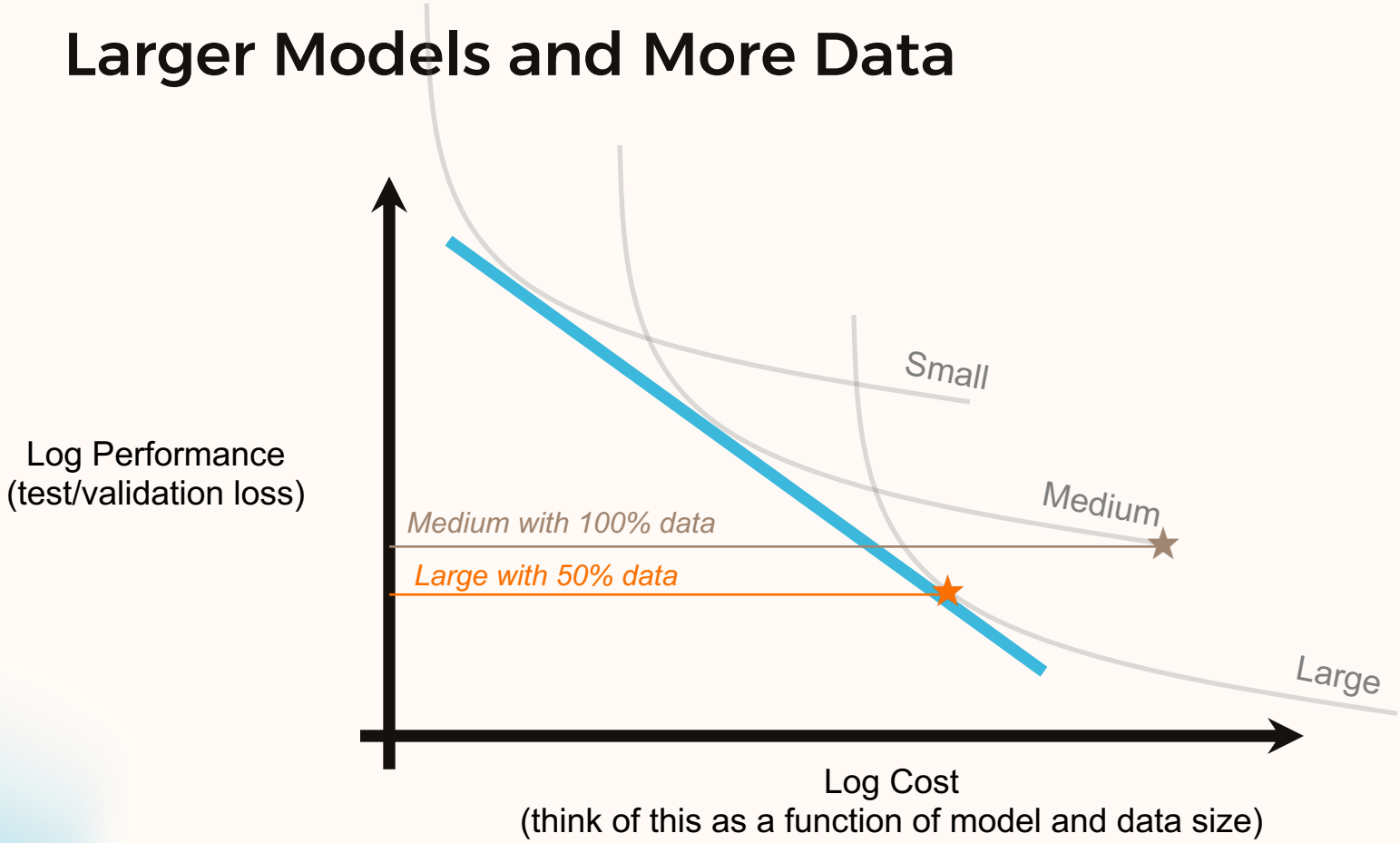
Double Descent



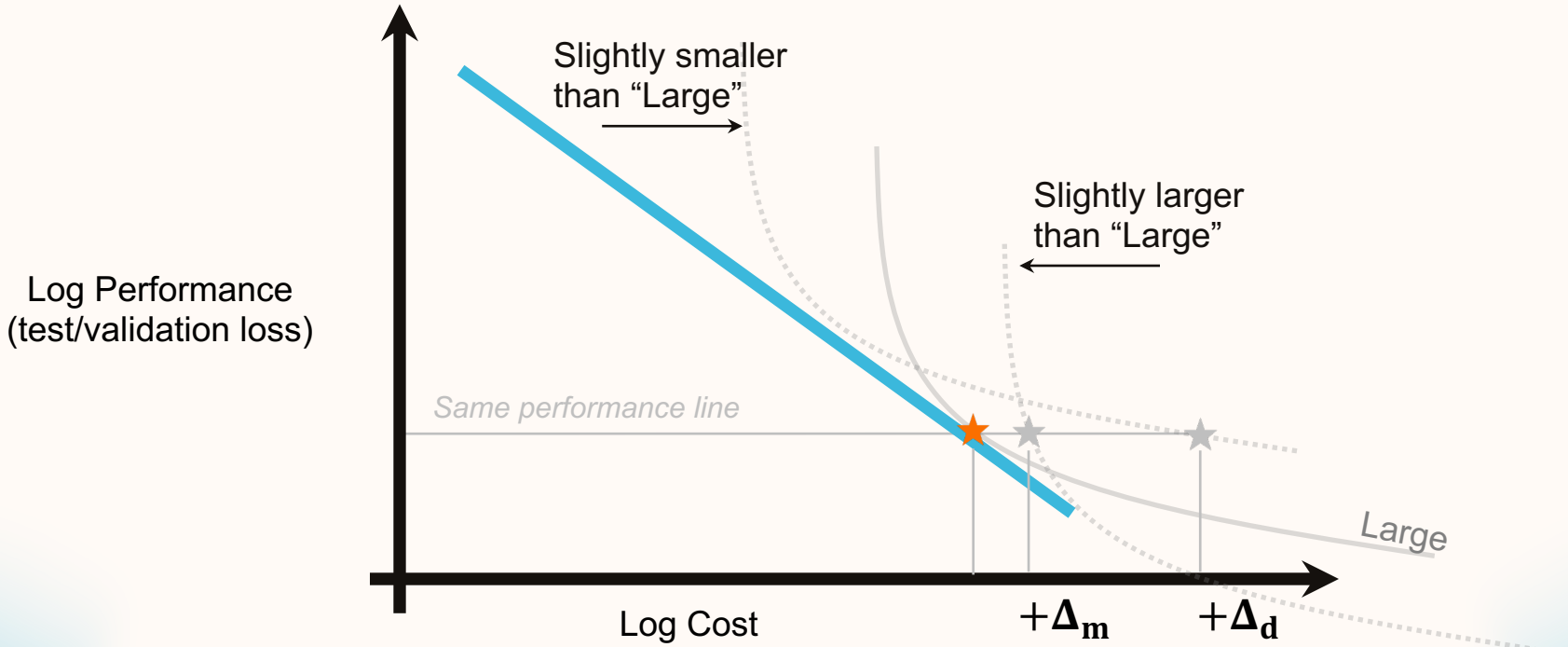
04

Scaling Laws

Larger Models and More Data



Compute Optimal and its Vicinity



Model Size > Data Size

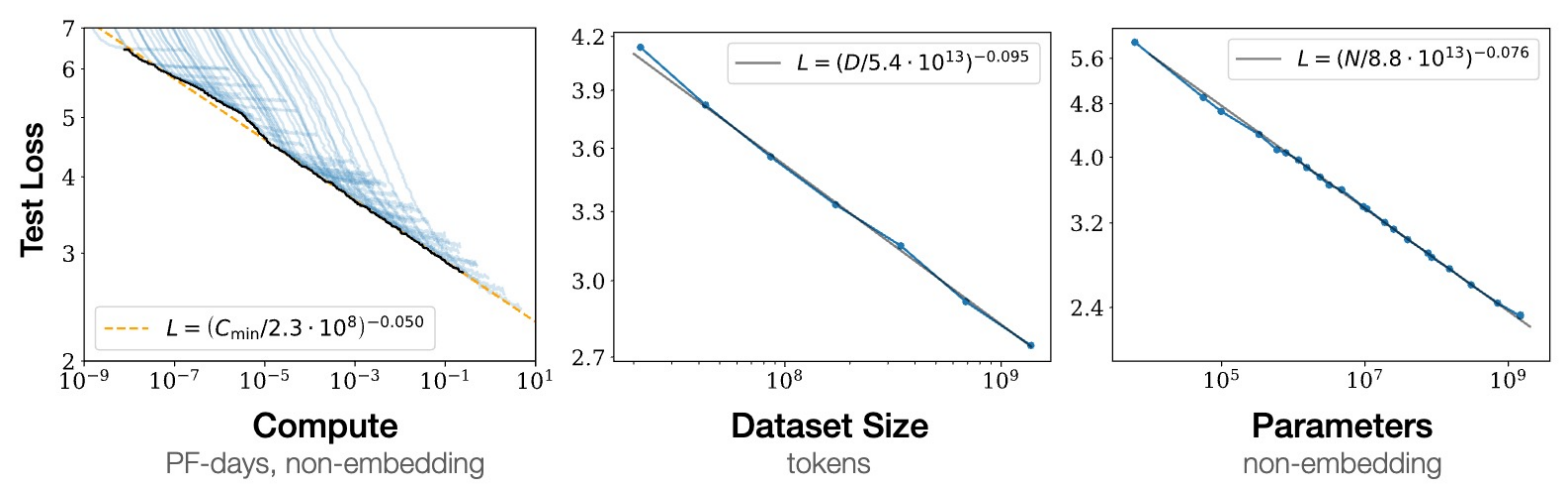


Image Source: Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020). Scaling laws for neural language models. arXiv preprint arXiv:2001.08361.

~~Model Size > Data Size~~ Model Size = Data Size

Table 2 | **Estimated parameter and data scaling with increased training compute.** The listed values are the exponents, a and b, on the relationship $N_{opt} \propto C^a$ and $D_{opt} \propto C^b$. Our analysis suggests a near equal scaling in parameters and data with increasing compute which is in clear contrast to previous work on the scaling of large models. The 10^{th} and 90^{th} percentiles are estimated via bootstrapping data (80% of the dataset is sampled 100 times) and are shown in parenthesis.

Approach	Coeff. a where $N_{opt} \propto C^a$	Coeff. b where $D_{opt} \propto C^b$
1. Minimum over training curves	0.50 (0.488, 0.502)	0.50 (0.501, 0.512)
2. IsoFLOP profiles	0.49 (0.462, 0.534)	0.51 (0.483, 0.529)
3. Parametric modelling of the loss	0.46 (0.454, 0.455)	0.54 (0.542, 0.543)
Kaplan et al. (2020)	0.73	0.27

Image Source: Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., ... & Sifre, L. (2022). Training compute-optimal large language models. arXiv preprint arXiv:2203.15556.

Planning the Pre-Training of LLMs

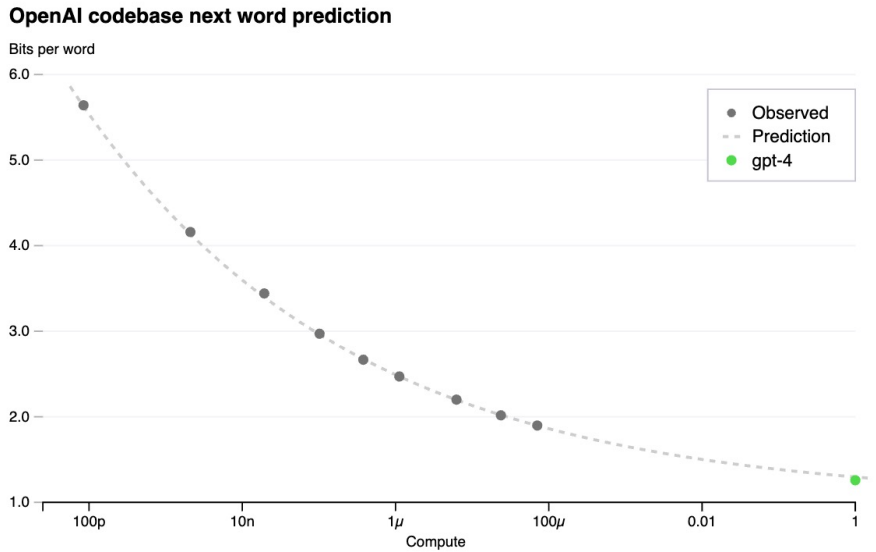


Figure 1. Performance of GPT-4 and smaller models. The metric is final loss on a dataset derived from our internal codebase. This is a convenient, large dataset of code tokens which is not contained in the training set. We chose to look at loss because it tends to be less noisy than other measures across different amounts of training compute. A power law fit to the smaller models (excluding GPT-4) is shown as the dotted line; this fit accurately predicts GPT-4's final loss. The x-axis is training compute normalized so that GPT-4 is 1.

Image Source: Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., ... & McGrew, B. (2023). Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Key Points

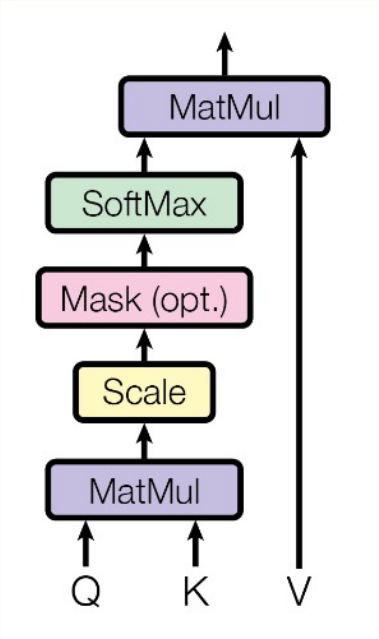
Scaling Laws

- ✓ Pre-training performance (next word prediction) is predictive with data and model size.
- ✓ Data and model size both are equally important in scaling.
- ✓ Larger model pre-training can be accurately planned by pre-training a bunch of smaller models.

05

Efficient Attn

Standard Attention



Algorithm 0 Standard Attention Implementation

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM.

1: Load $\mathbf{Q}, \mathbf{K}$ by blocks from HBM, compute $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top$, write $\mathbf{S}$ to HBM.
2: Read $\mathbf{S}$ from HBM, compute $\mathbf{P} = \mathrm{softmax}(\mathbf{S})$, write $\mathbf{P}$ to HBM.
3: Load $\mathbf{P}$ and $\mathbf{V}$ by blocks from HBM, compute $\mathbf{O} = \mathbf{P}\mathbf{V}$, write $\mathbf{O}$ to HBM.
4: Return $\mathbf{O}$.

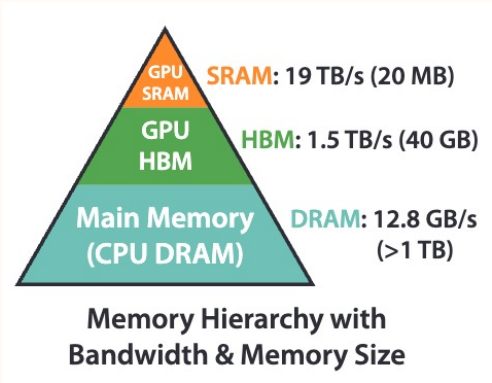
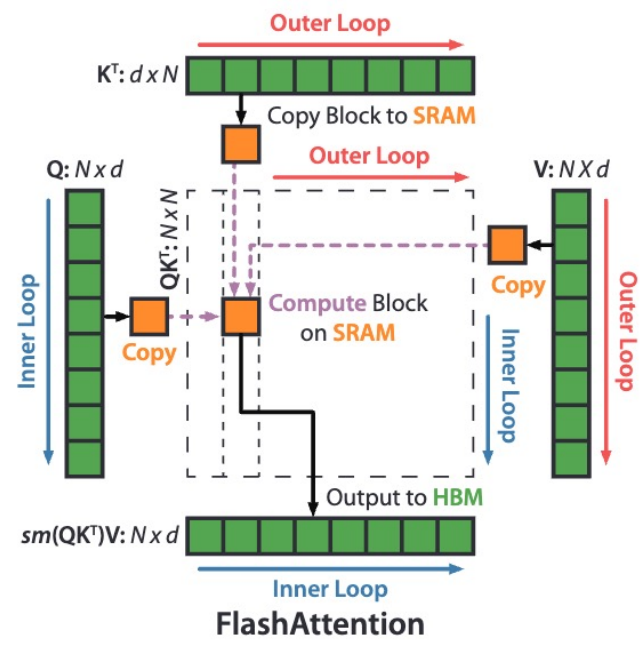


Image Source:

- *Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.*
- *Dao, T., Fu, D., Ermon, S., Rudra, A., & Ré, C. (2022). Flashattention: Fast and memory-efficient exact attention with io-awareness. Advances in Neural Information Processing Systems, 35, 16344-16359.*

Flash Attention



Algorithm 0 Standard Attention Implementation

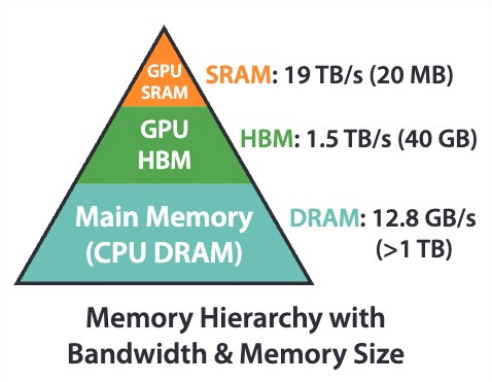
Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM.

1: Load $\mathbf{Q}, \mathbf{K}$ by blocks from HBM, compute $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top$, ~~write $\mathbf{S}$ to HBM.~~

2: Read $\mathbf{S}$ from HBM, compute $\mathbf{P} = \mathrm{softmax}(\mathbf{S})$, ~~write $\mathbf{P}$ to HBM.~~

3: Load $\mathbf{P}$ and $\mathbf{V}$ by blocks from HBM, compute $\mathbf{O} = \mathbf{P}\mathbf{V}$, write $\mathbf{O}$ to HBM.

4: Return $\mathbf{O}$.



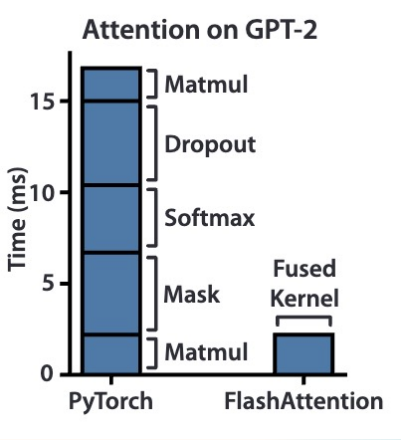


Image Source: Dao, T., Fu, D., Ermon, S., Rudra, A., & Ré, C. (2022). Flashattention: Fast and memory-efficient exact attention with io-awareness. Advances in Neural Information Processing Systems, 35, 16344-16359.

Key Points

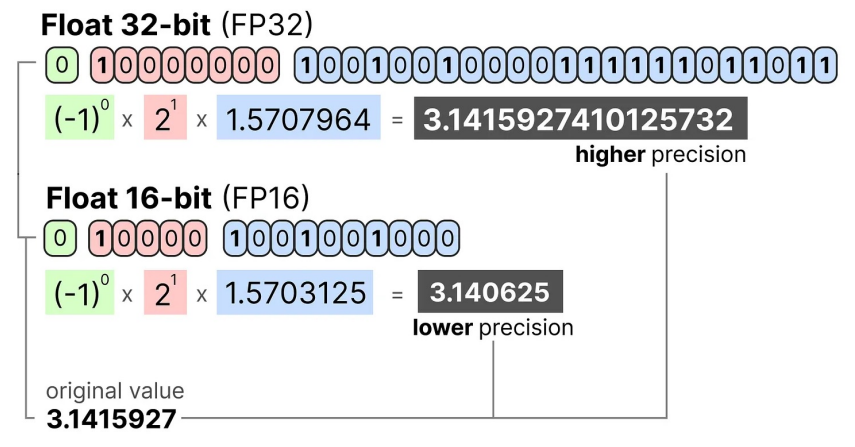
Efficient Attn

- ✓ Greatly reduced GPU memory consumption. Although, it is not always faster.
- ✓ Allowed processing of longer sequences with nominal GPU memory.

06

Quantization

Changing the Datatype



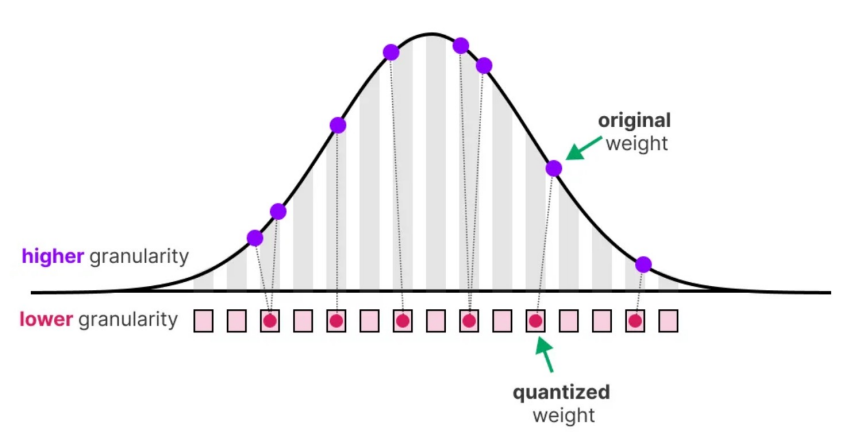
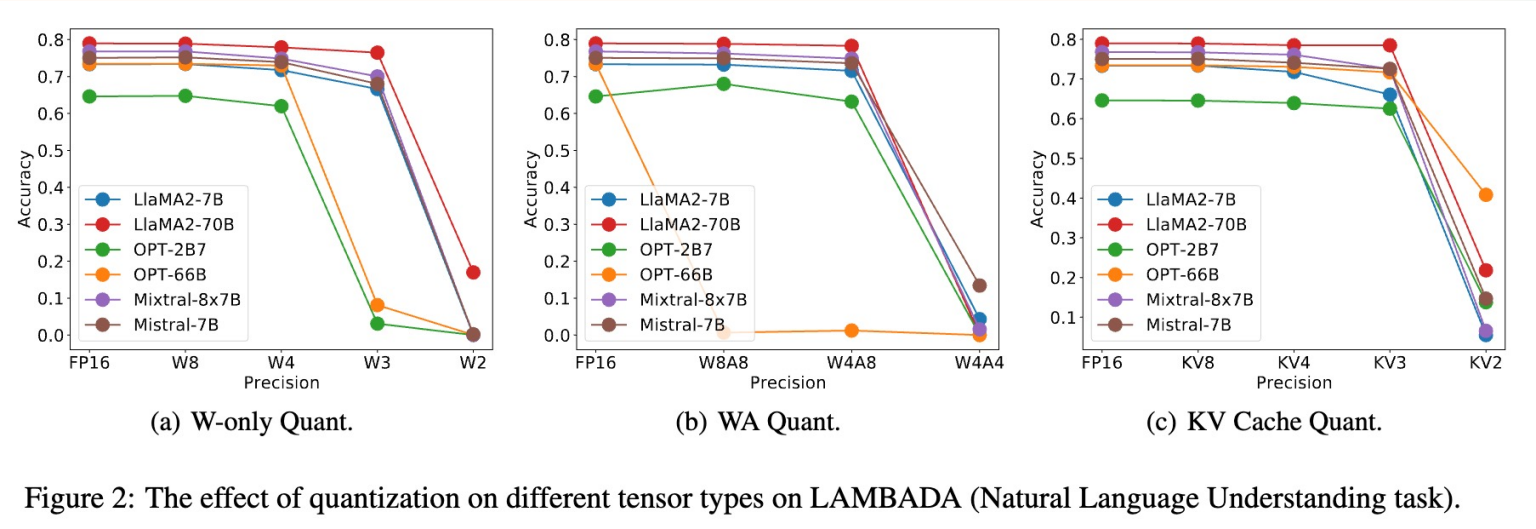


Image Source: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-quantization

Effect on Performance



(a) W-only Quant. (b) WA Quant. (c) KV Cache Quant.

Figure 2: The effect of quantization on different tensor types on LAMBADA (Natural Language Understanding task).

Image Source: Li, S., Ning, X., Wang, L., Liu, T., Shi, X., Yan, S., ... & Wang, Y. (2024). Evaluating quantized large language models. arXiv preprint arXiv:2402.18158.

Effect on Scaling

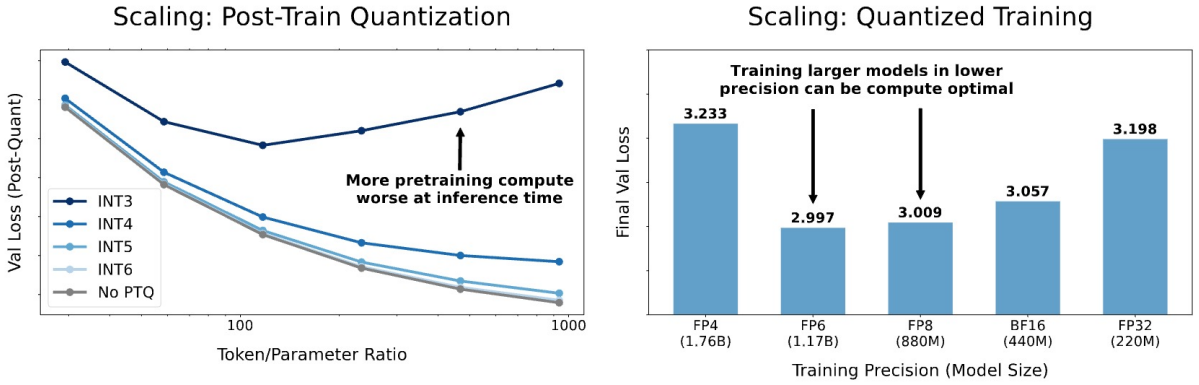


Figure 1: Schematic of key findings. (Left) Training a fixed model size to various data budgets in BF16 and quantizing weights at the end. We find that degradation due to post-train quantization increases with tokens seen during pretraining, so that eventually **additional pretraining data can be harmful**. (Right) Our scaling suggests **training *larger* models in *lower* precision can be compute-optimal** according to the cost model in Section 4.3. Weights, activations, attention quantized, all models trained on the same data budget, details in Appendix H.

Image Source: Kumar, T., Ankner, Z., Spector, B. F., Bordelon, B., Muennighoff, N., Paul, M., ... & Raghunathan, A. (2024). Scaling Laws for Precision. arXiv preprint arXiv:2411.04330.

Key Points

Quantization

- ✓ Reduces the GPU memory requirements to host a model during inference.
- ✓ Can significantly affect the performance, that gets worse with increasing context.
- ✓ More pre-training data can lead to worse performance degradation due to post-training quantization.

Questions?

Thank You!

Feel free to reach out at:
vijeta_deshpande@student.uml.edu