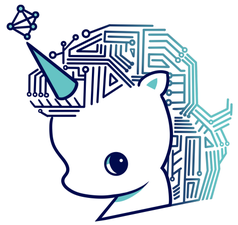
Large-Scale Training

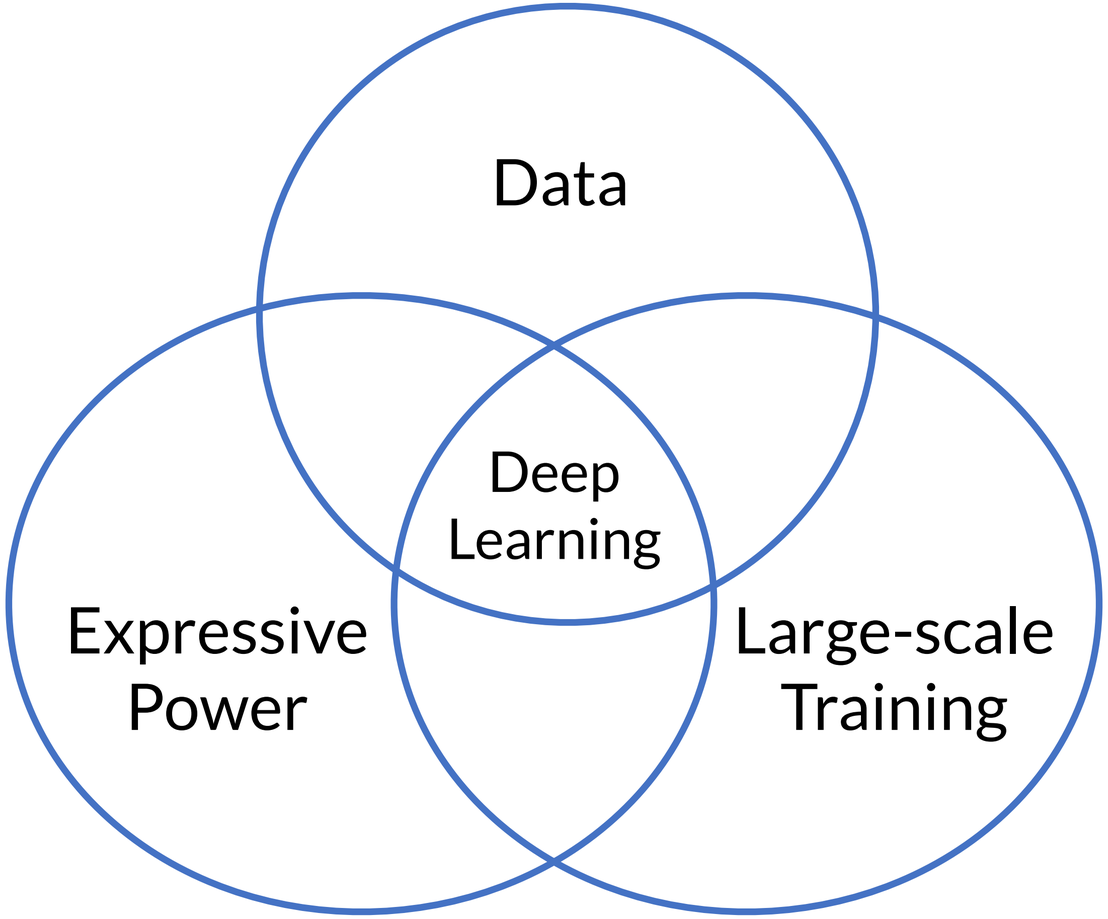
Principles of Deep Learning (AI502/IE408/IE511)

Sungbin Lim (UNIST AIGS & IE)



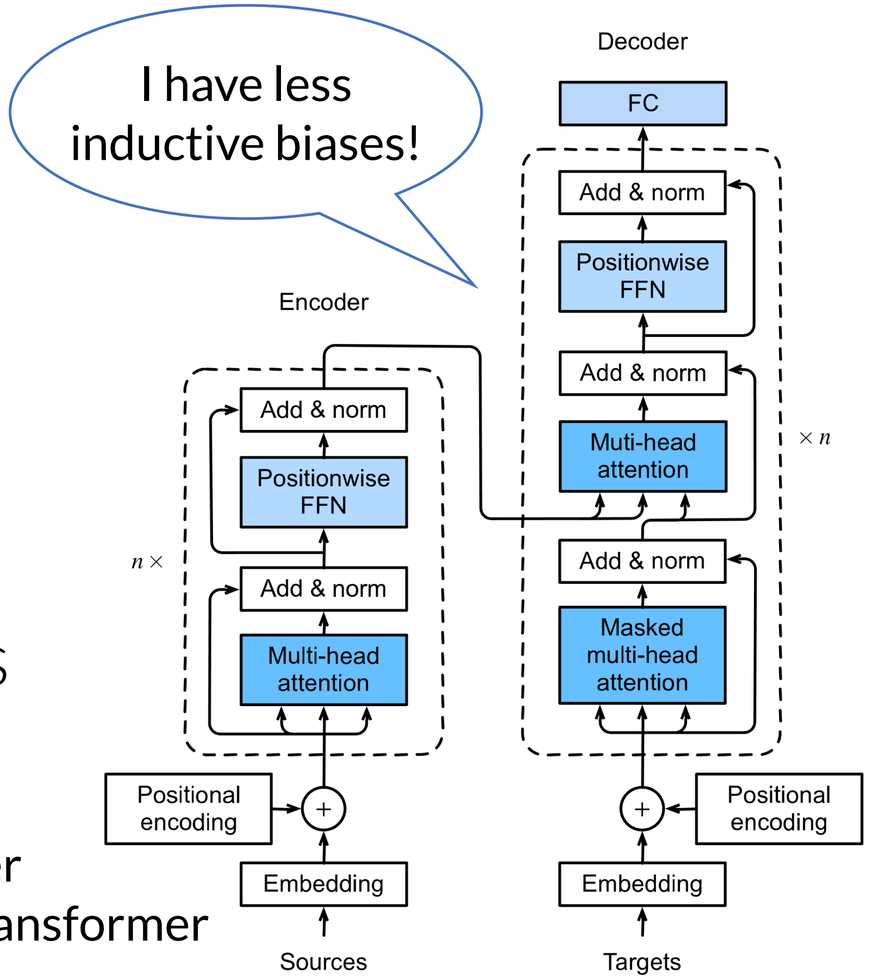
Contact: ai502deeplearning@gmail.com

Essence of Deep Learning



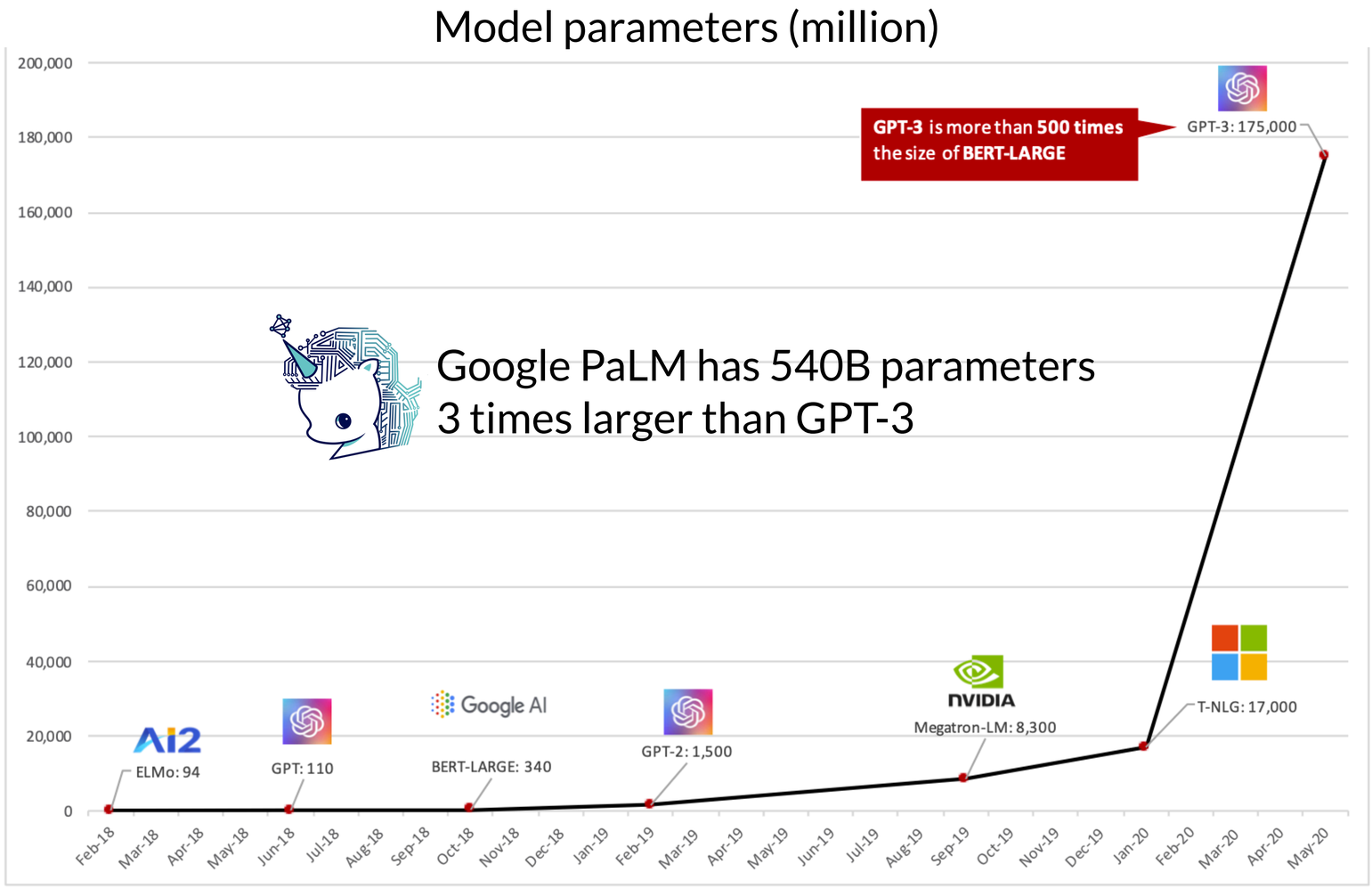
Design Principles in Deep Learning

- Inductive Biases in Convolutional Networks
  - Locality Principle
  - Spatial Invariance
- Inductive Biases in Graph Neural Networks
  - Permutation Invariance
- Inductive Biases in Recurrent Neural Networks
  - Sequentiality
  - Temporal Invariance

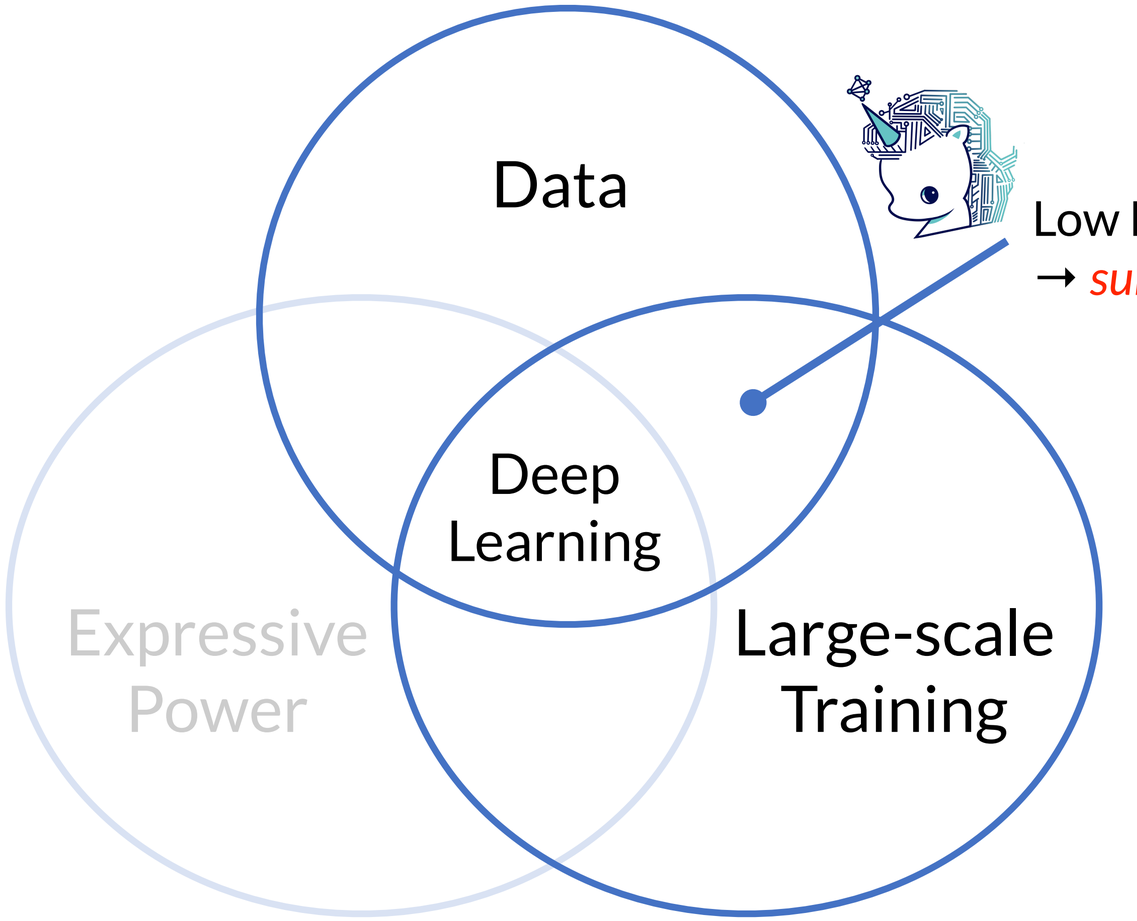


Note that MLP has fewer inductive biases than Transformer

Parameter size of language models

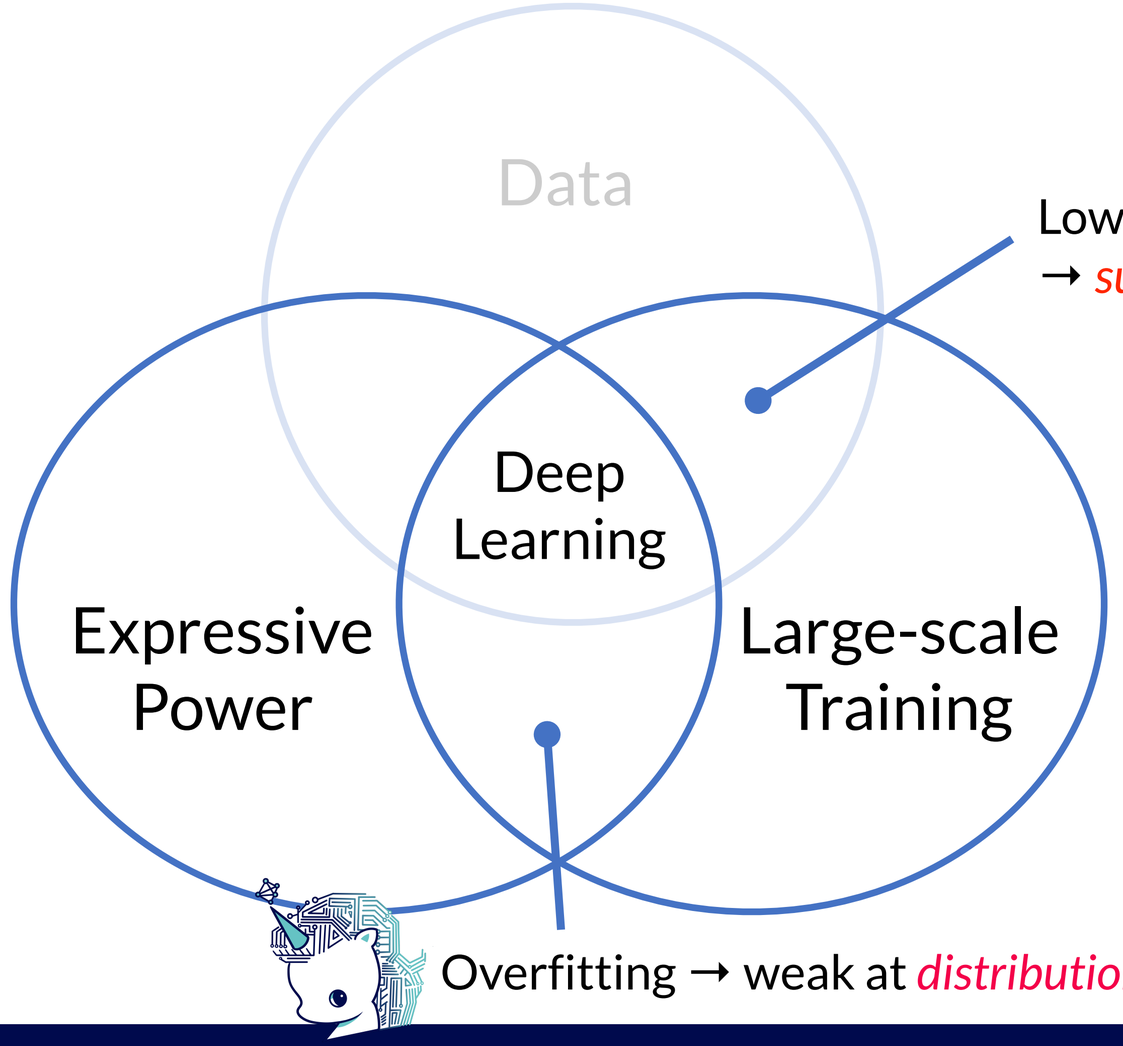


Essence of Deep Learning



Low Expressive Power
→ *suboptimal* performance

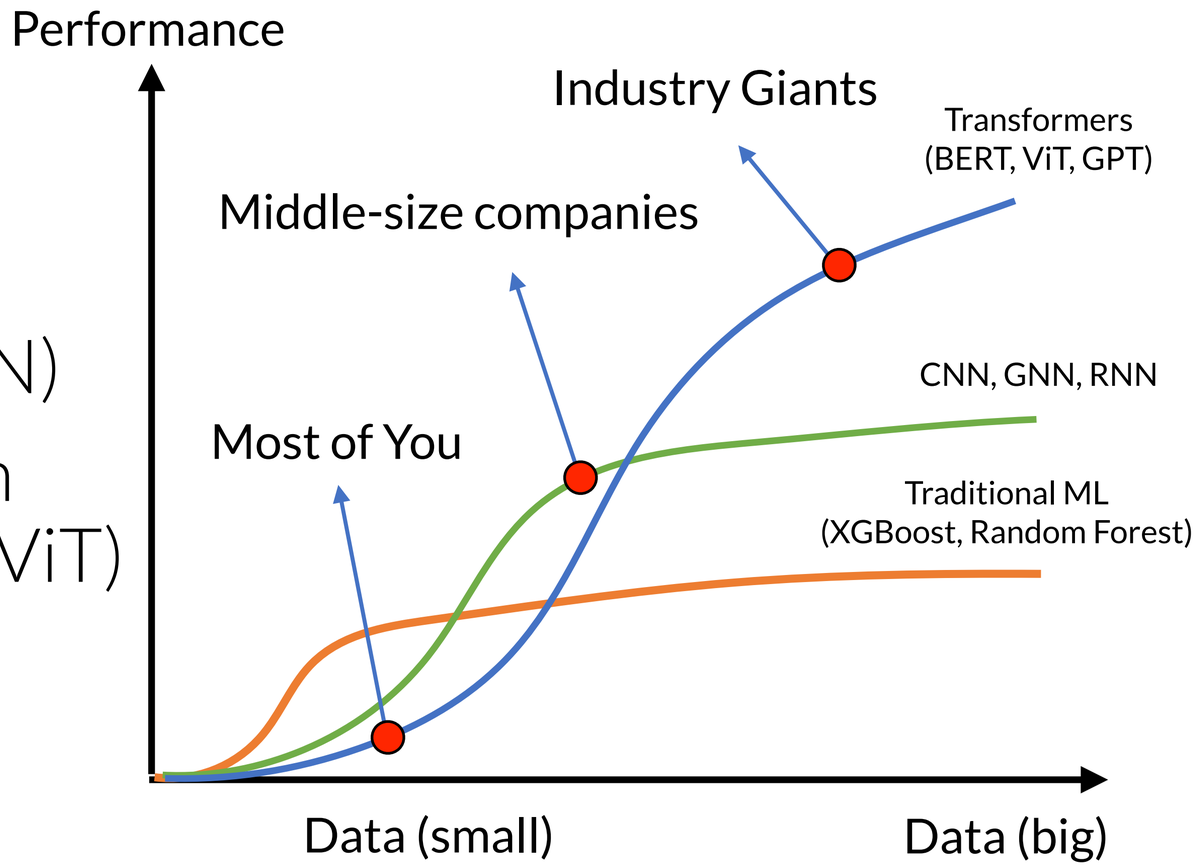
Essence of Deep Learning



Expressive models require more data

- If you have small number of data, then traditional ML performs well
- If you have more data, then you can try deep learning with well-designed inductive biases (e.g. CNN, GNN, RNN)
- If you have large number of data, then you can try Transformer-type (BERT, ViT) with (self-)supervised learning

Data is money, but…



Optimization & Deep Learning

- Optimization algorithms are important for deep learning
  - training can take hours, days, or even weeks
  - algorithms directly affect the model's efficiency

$$\underset{\theta}{\text{minimize}}\ L(f_\theta(\mathbf{x}), \mathbf{y}), \qquad (\mathbf{x}, \mathbf{y}) \in \mathcal{D}$$

$\mathcal{D}$ → data

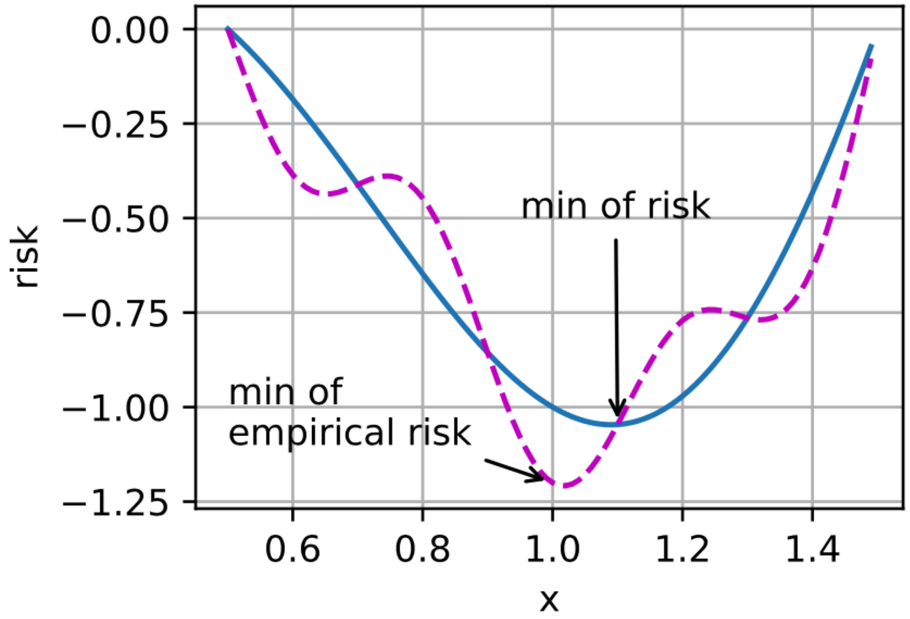
θ → parameters

L → loss function

f_θ → model (neural net)

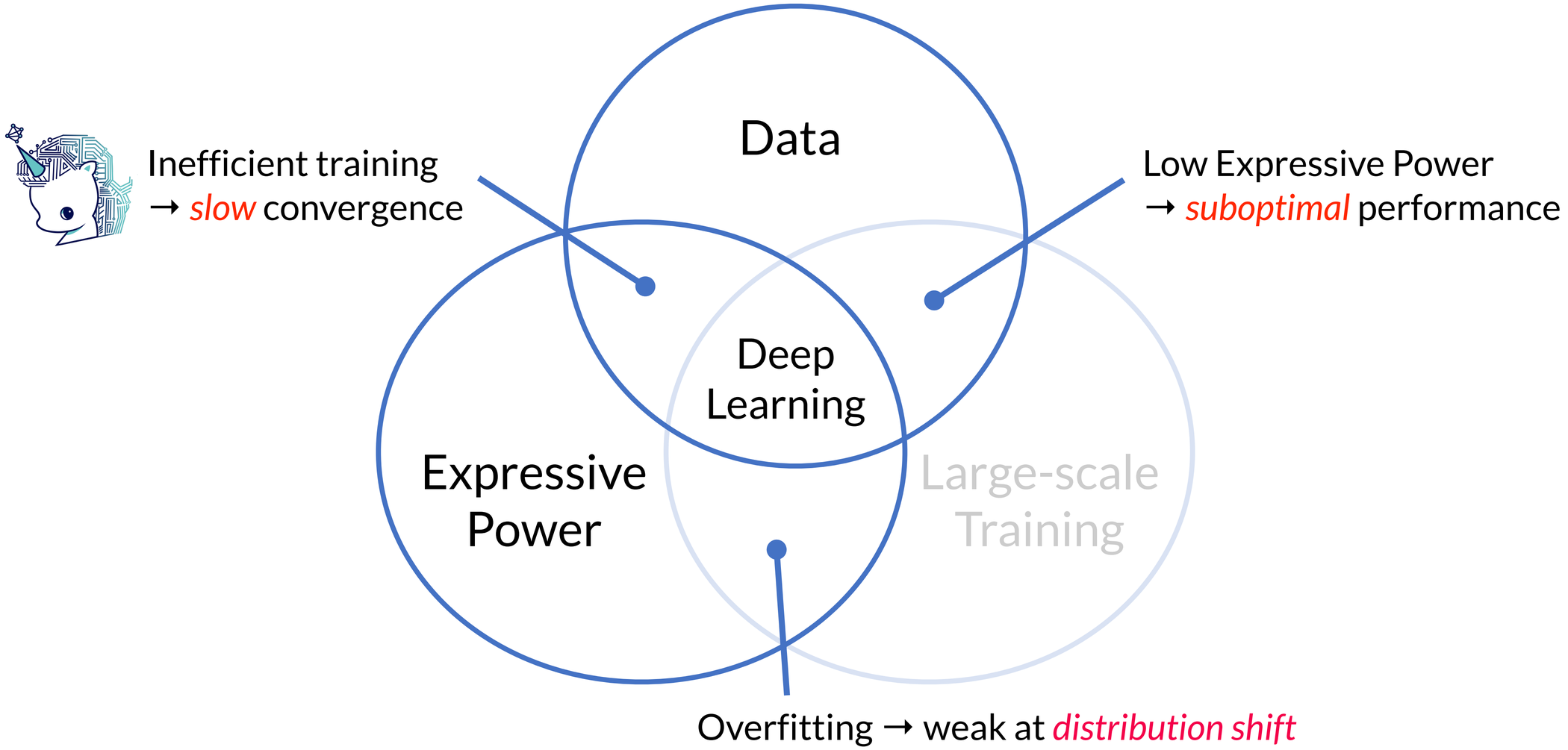
Optimization ≠ Deep Learning

- Almost all optimization problems arising in Deep Learning are non-convex
- Optimization ≠ Deep Learning
  - goal of optimization is to reduce the **training** error
  - goal of statistical inference is to reduce the **generalization** error



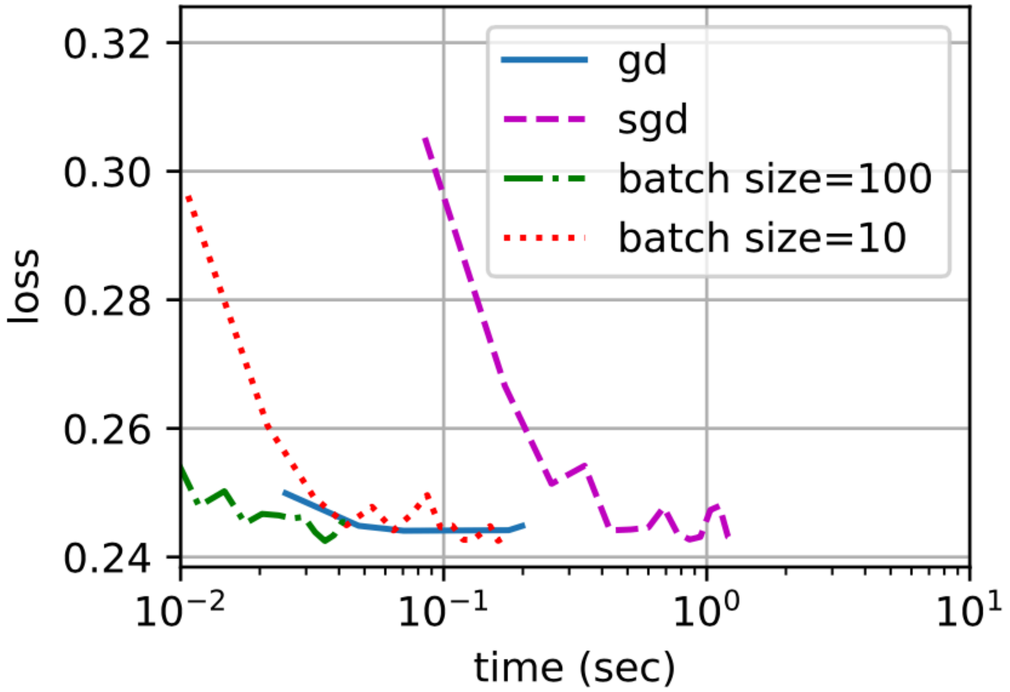
$$L_{\text{train}} \neq L_{\text{test}}$$

Essence of Deep Learning



Workhorse of Deep Learning

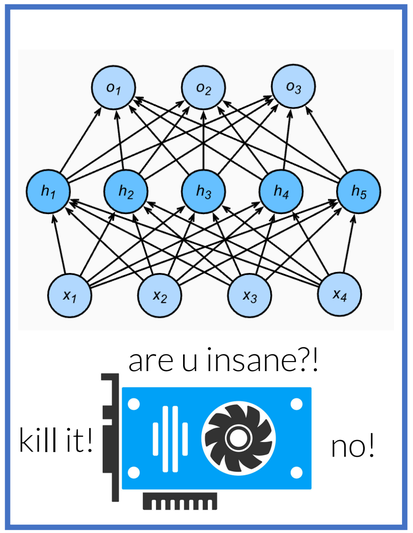
- What is the workhorse of deep learning?
  - parallelization
    - multiple GPUs
    - multiple servers
- Use a hierarchy of CPU caches to supply the processor with data
- Mini-batch SGD reduces overhead when updating parameters

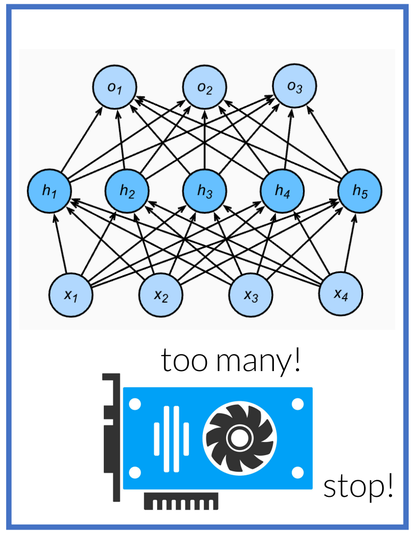


Deep Learning with Big Data

$$256 \times 256 \times 3 \times 1{,}000{,}000 \approx 2^{37}$$

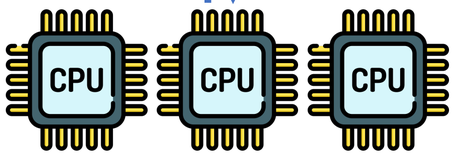
Image data





Training
- matrix calculus
- backprop
- parallel computing
- ...

damn stupid GPUs

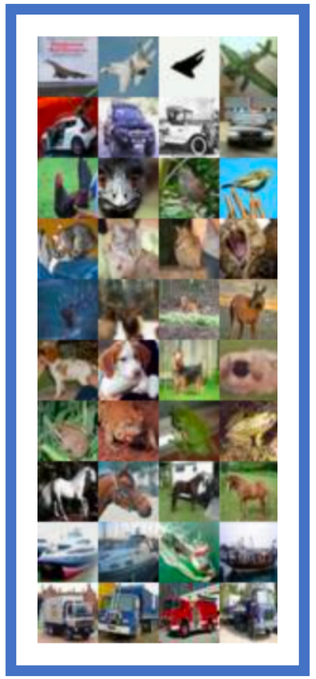


Working
- read & load data
- preprocessing
- aggregate
- ...

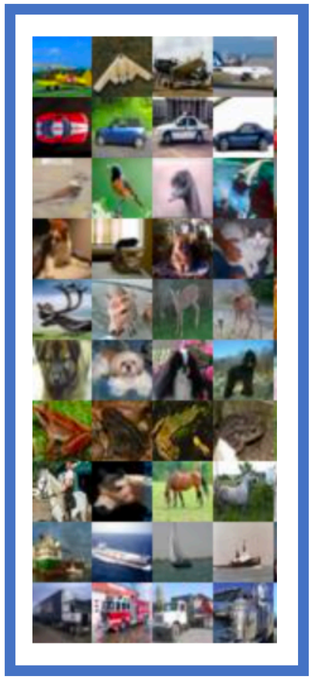
If we upload such data at once, then OOM occurs

Deep Learning with Big Data

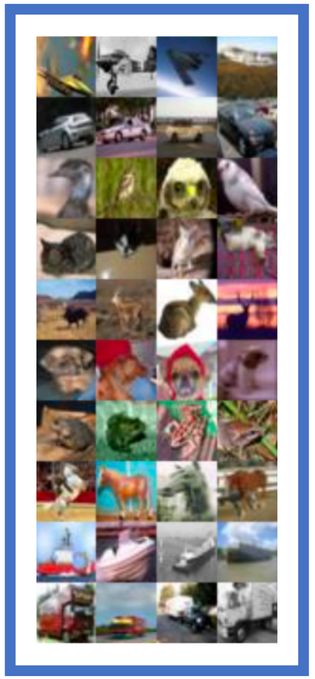
sleeping..



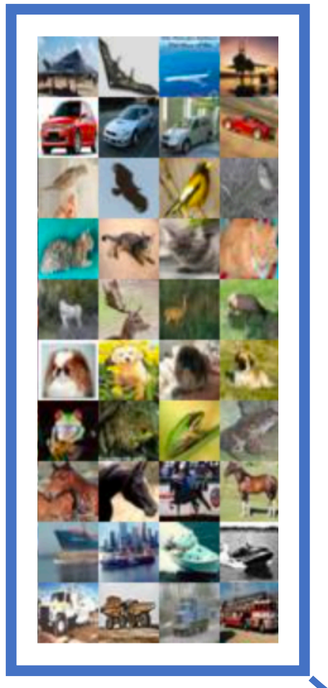
get ready!



I am ready!

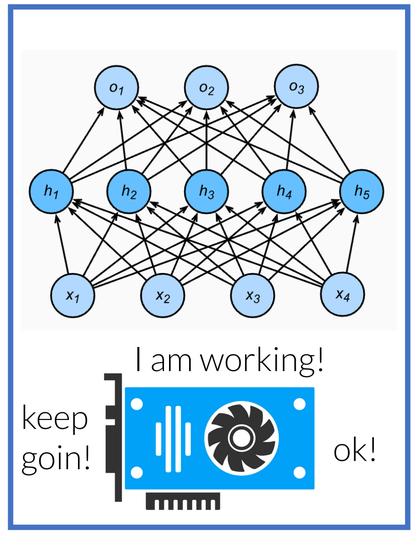


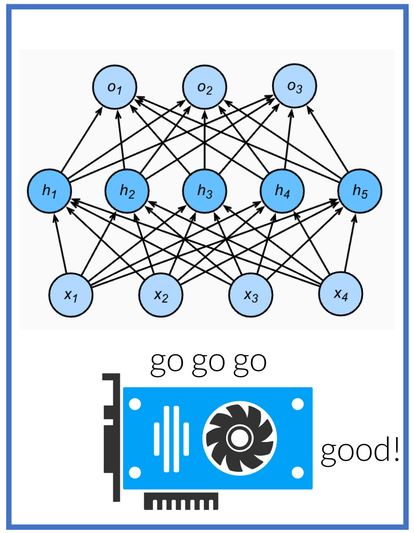
I am going!



$$256 \times 256 \times 3 \times |\mathcal{B}| \leq 2^{18} \cdot |\mathcal{B}|$$

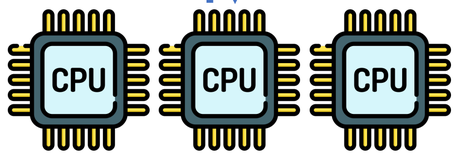
Image data





Training
- matrix calculus
- backprop
- parallel computing
- ...

much easier than before



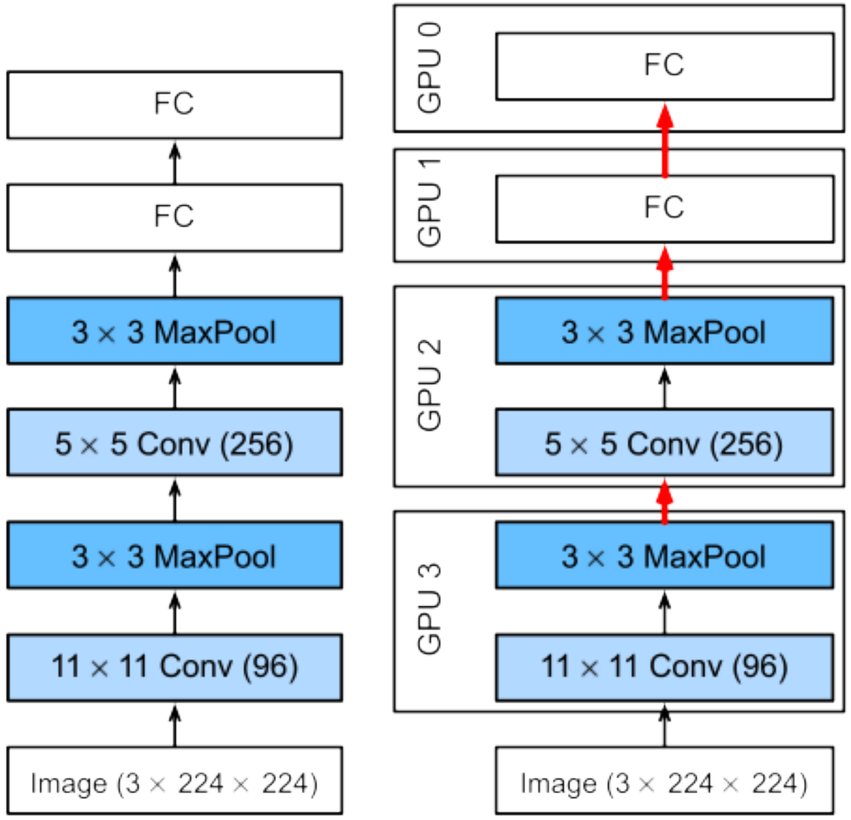
Working
- read & load data
- preprocessing
- aggregate
- ...

Hence mini-batch size is another hyperparameter!

Large-scale training ≠ more GPUs

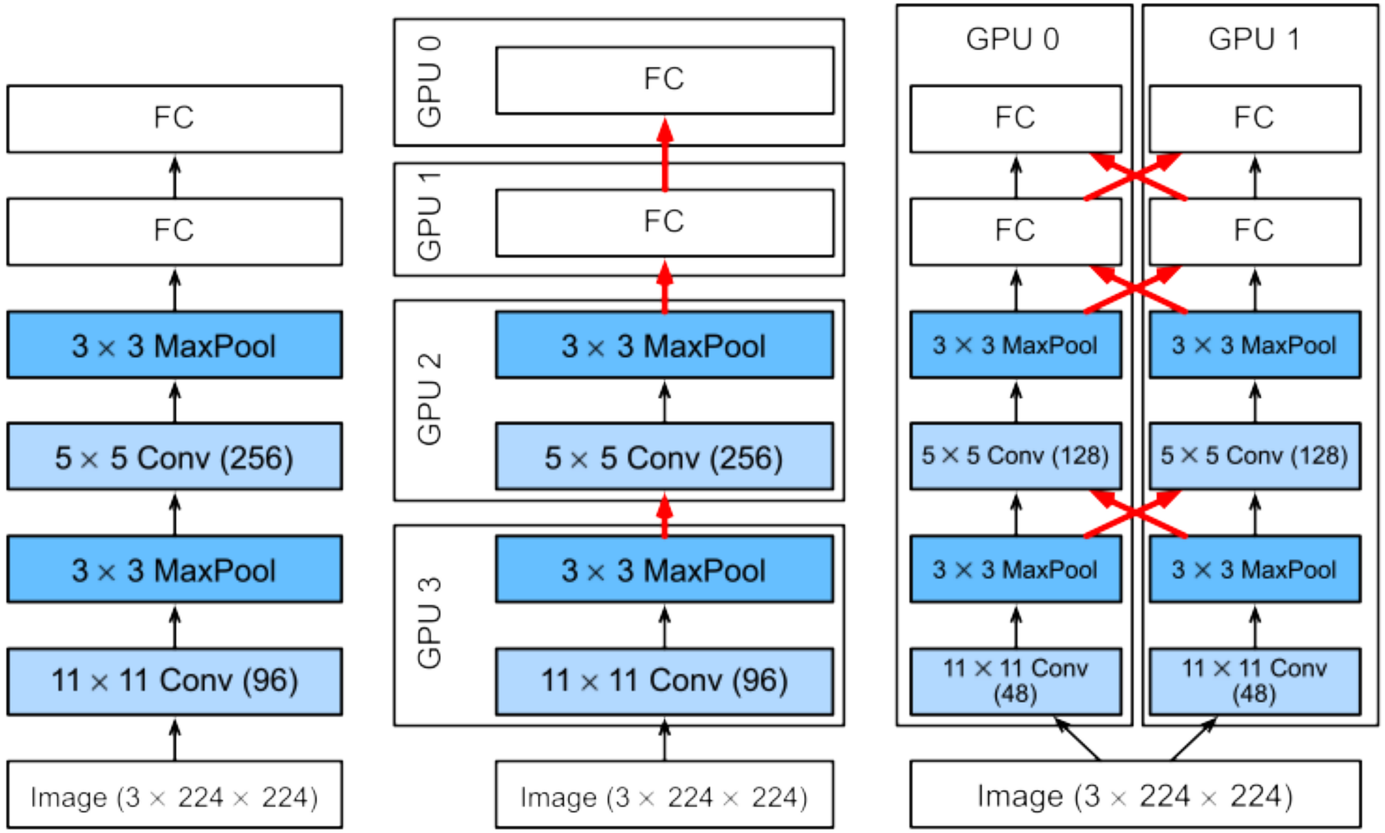
- A machine with multiple GPUs is important factor of large-scale training, but it does not imply scalable training trivially
  - data I/O bottleneck
  - communication between CPU and GPU / **synchronization**
  - mini-batch SGD optimization / normalization
  - model parameter size
- Many deep learning frameworks (TensorFlow, PyTorch) provide solutions
  - we cannot say those solutions are complete but many individuals contribute via open source communities

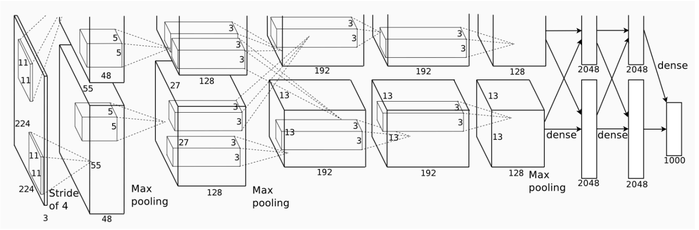
Parallelization on Multiple GPUs



Original Network Network Partitioning

Parallelization on Multiple GPUs

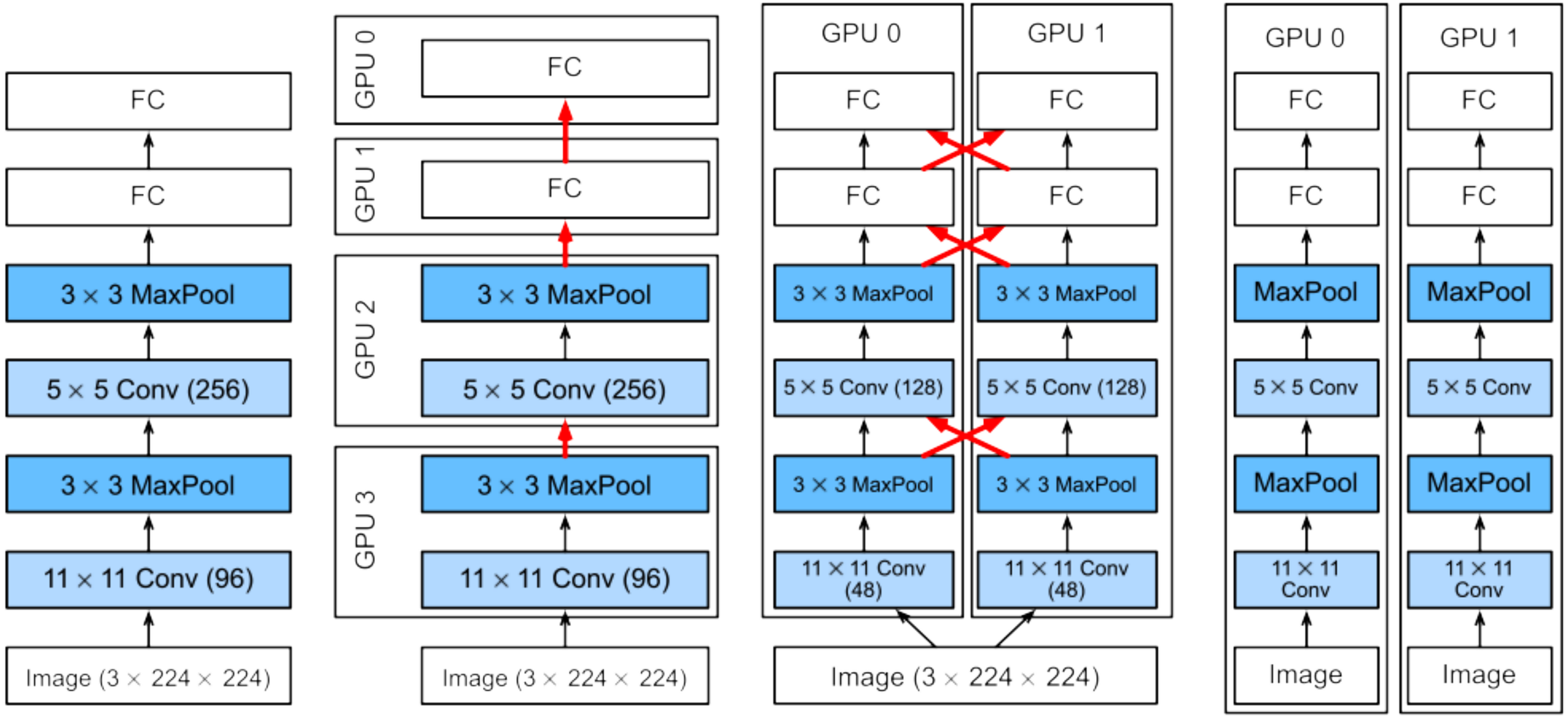




AlexNet (2012)

Original Network | Network Partitioning | Layerwise Partitioning | Model Parallelism

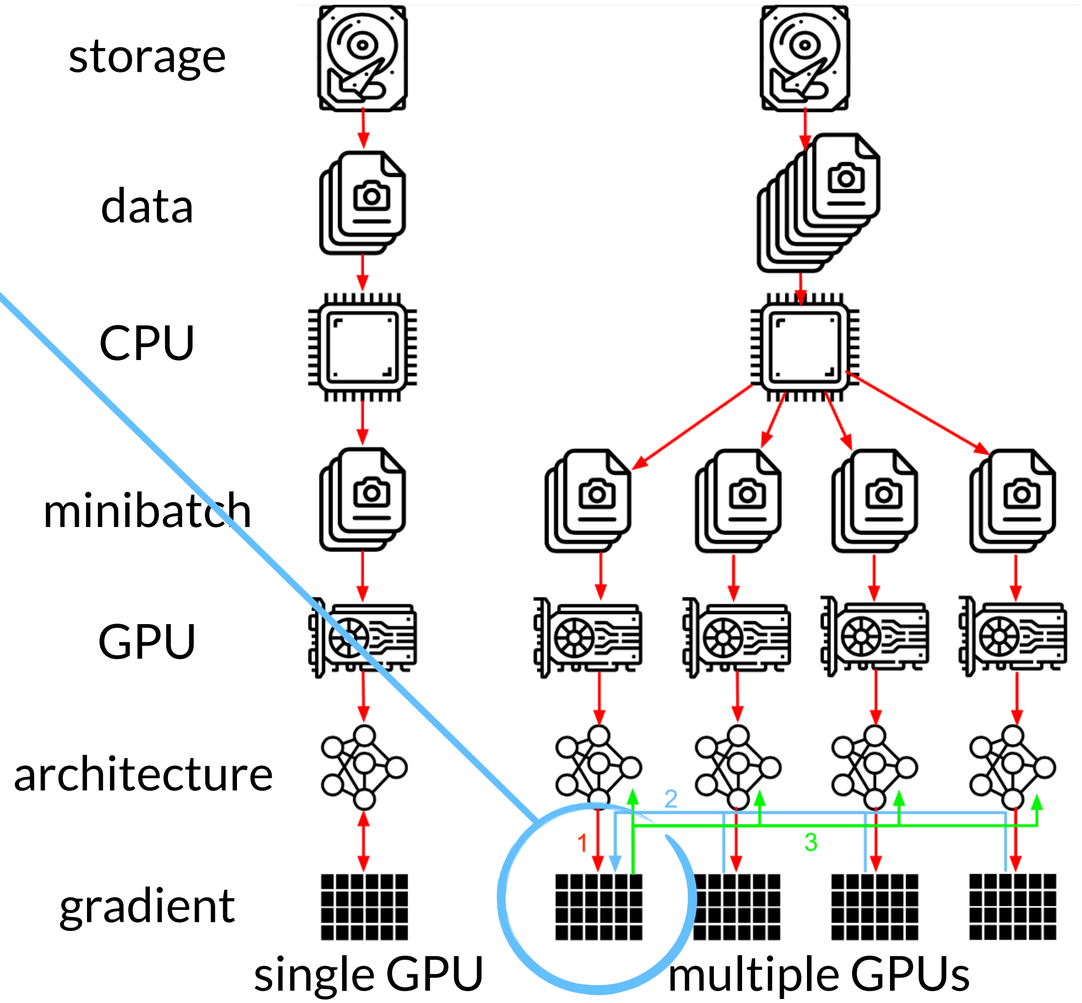
Parallelization on Multiple GPUs



Original Network | Network Partitioning | Layerwise Partitioning | Data Parallelism

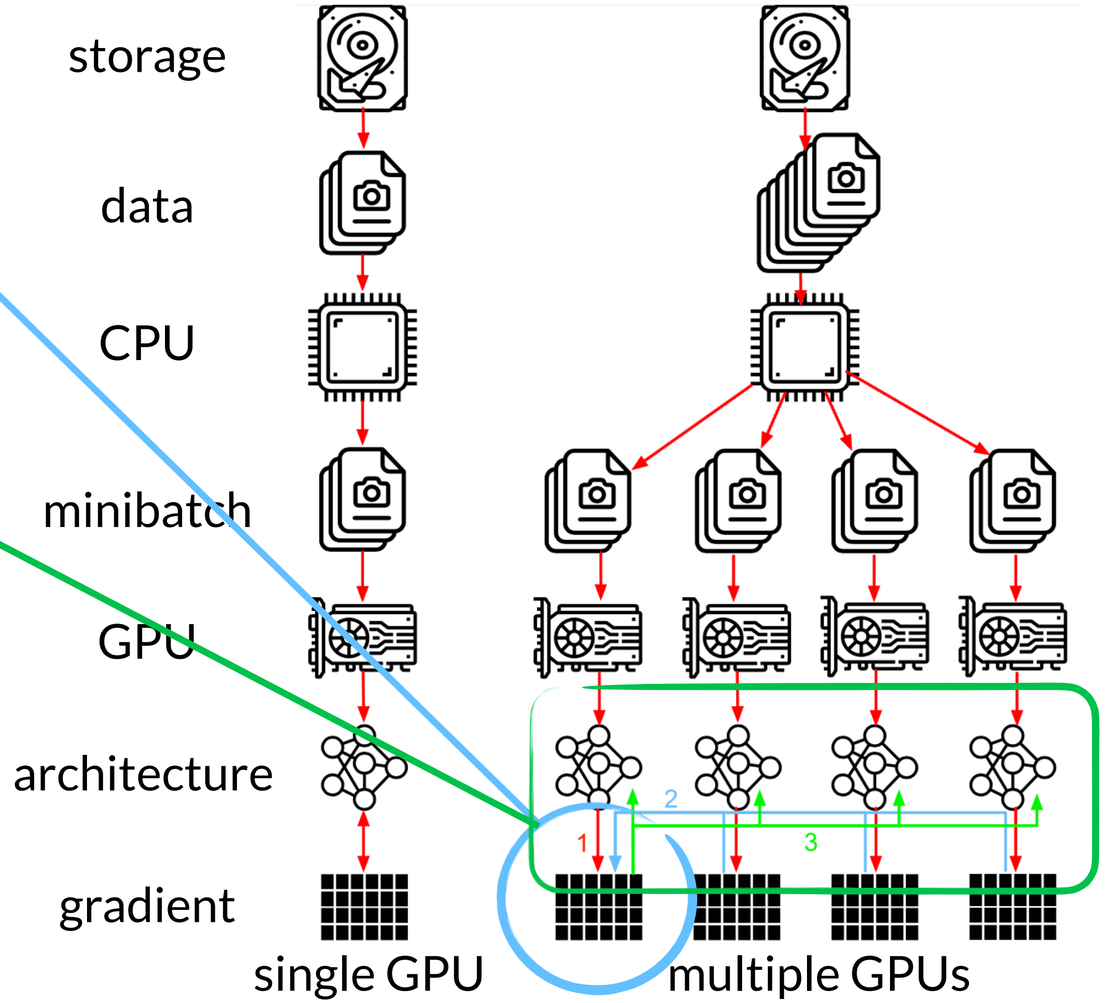
Understanding Data Parallelism

- The key aspect in data parallelism is the **aggregation of gradients**
  - this is **parameter server**



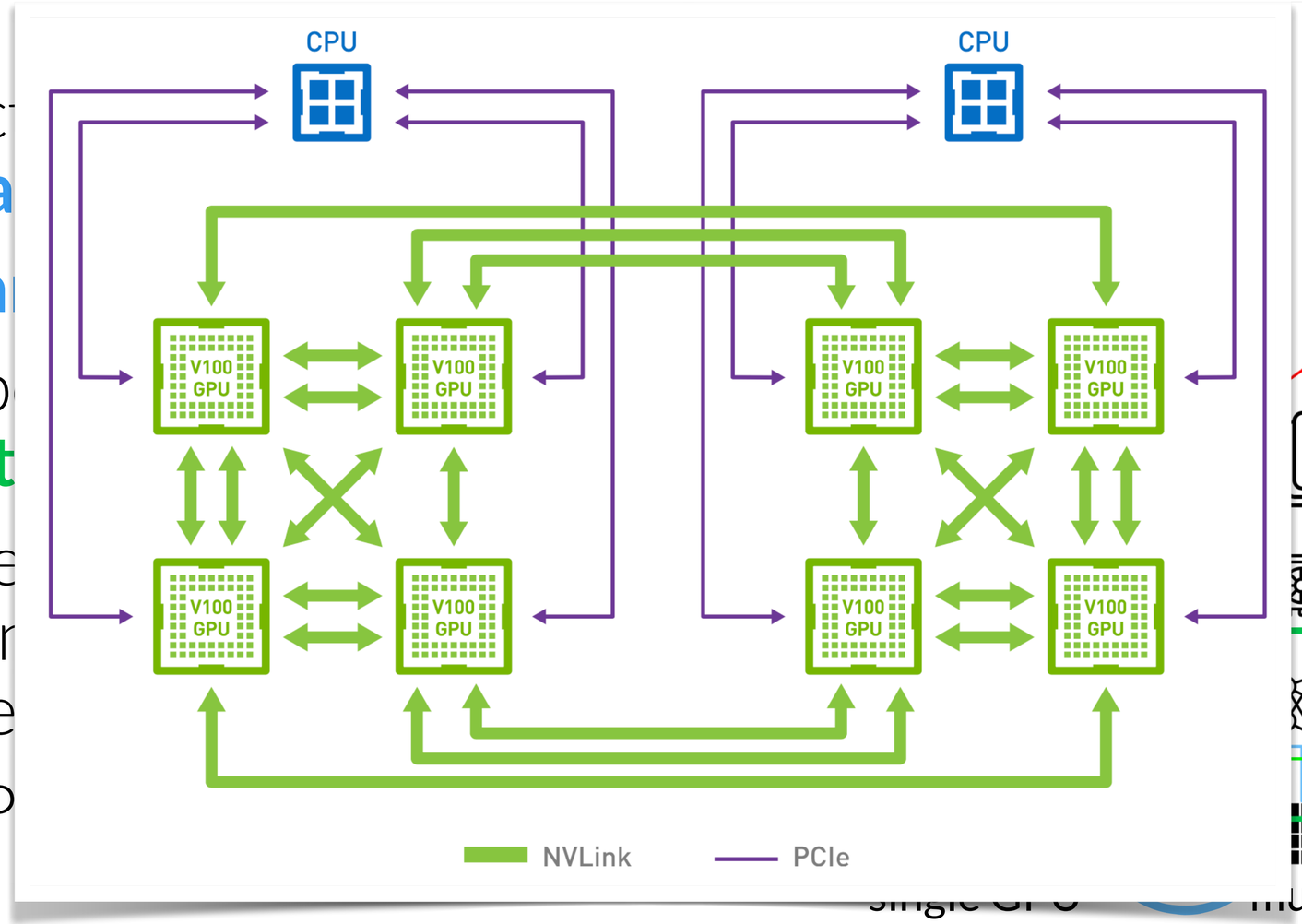
Understanding Data Parallelism

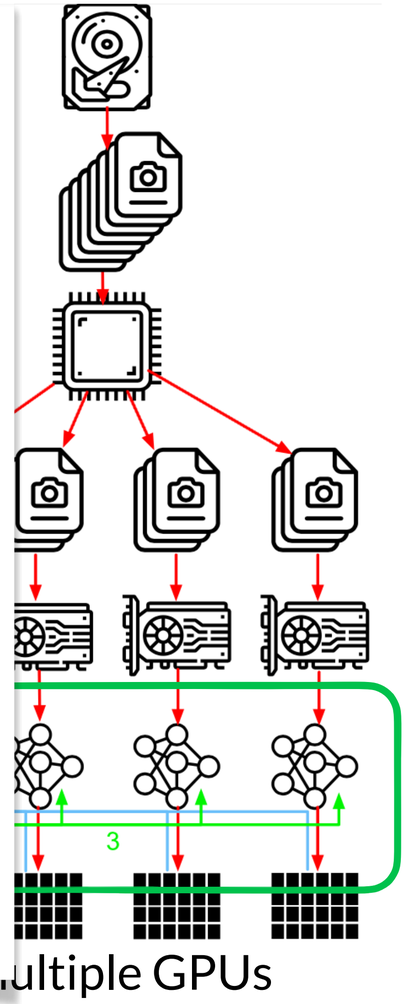
- The key aspect in data parallelism is the **aggregation of gradients**
  - this is **parameter server**
  - then we update parameters and **re-distribute** to all GPUs
- One can aggregate gradients into CPU but recent GPU hardwares provide efficient synchronization
  - NVLink > PCIe



Understanding Data Parallelism

- The key aspec
  is the **aggrega**
  - this is **para**
  - then we up
    and **re-dist**
- One can aggre
  CPU but recer
  provide efficie
  - NVLink > P





ultiple GPUs

Multiple Machine Training

In practice, you will encounter many barriers in connection and DB management

parameter server

storage

data

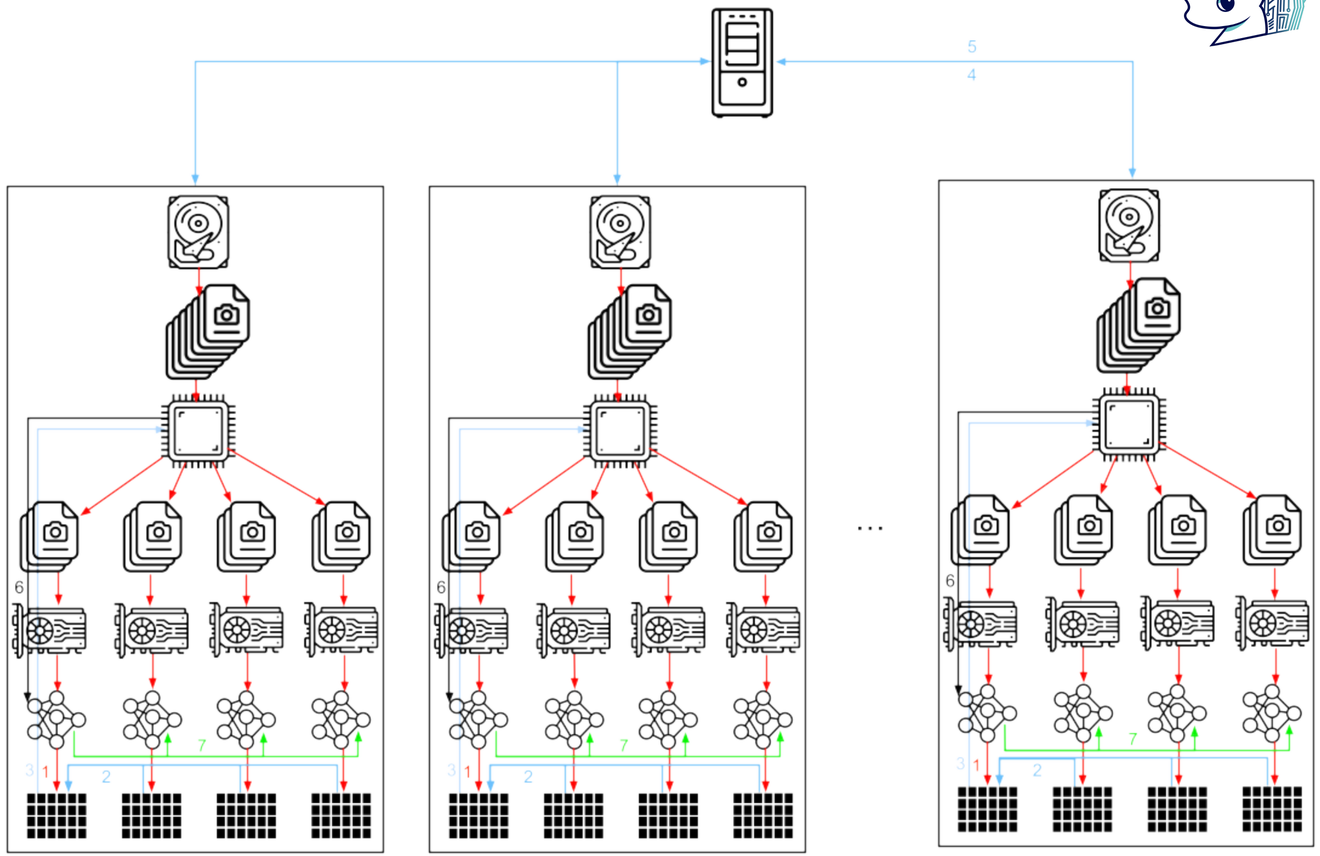
CPU

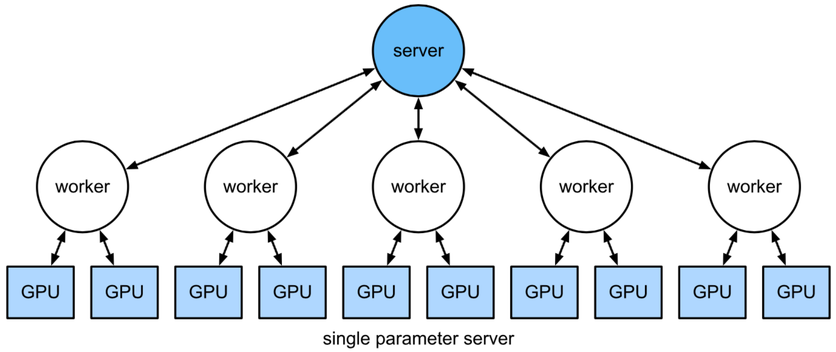
minibatch

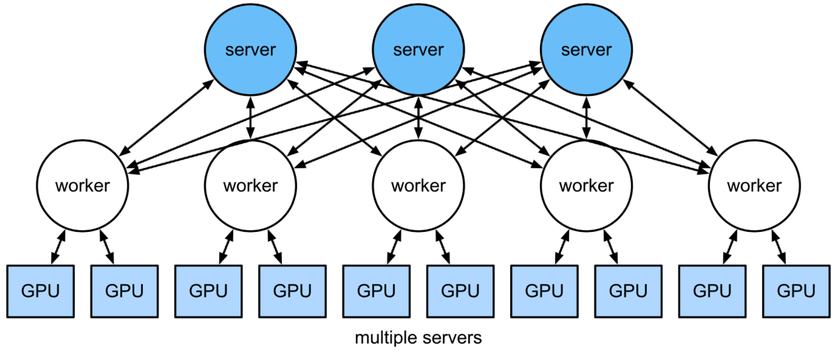
GPU

architecture

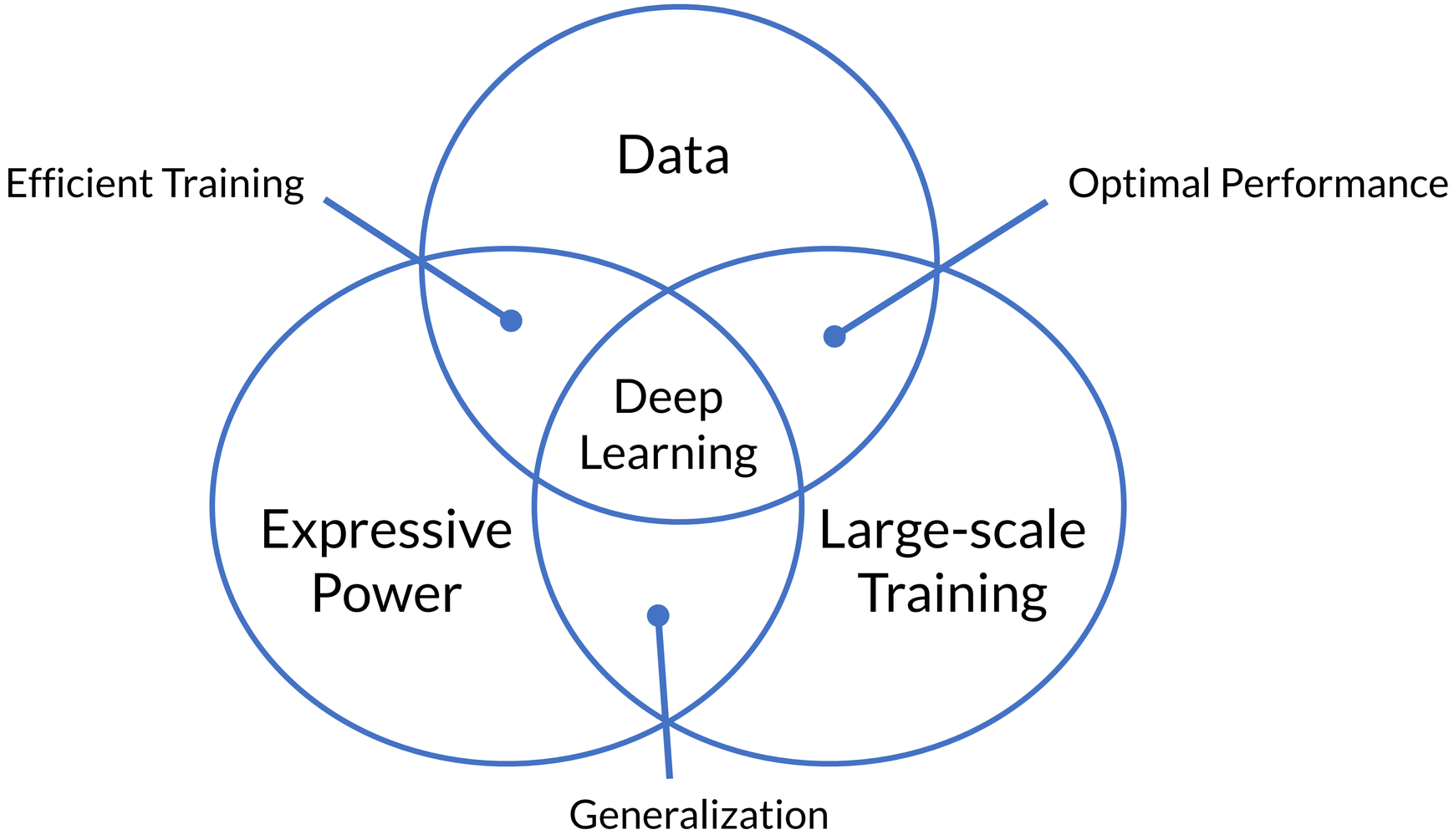
gradient







Essence of Deep Learning

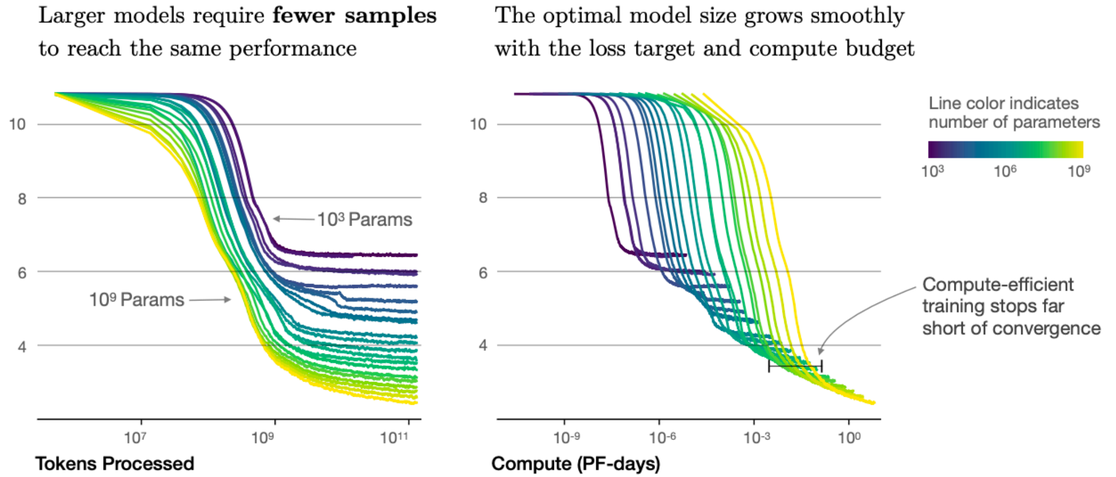


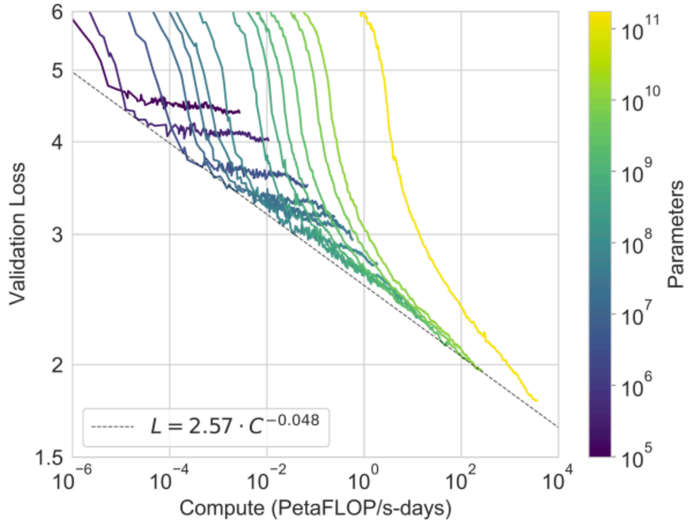
Scaling Law in Transformer

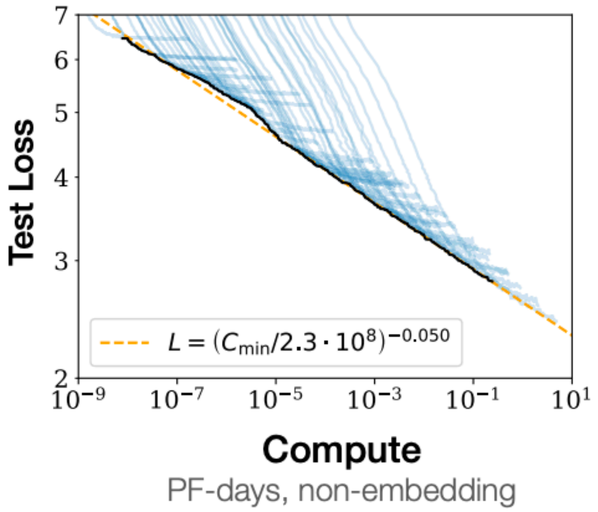
See Kaplan et al., Scaling laws for neural language models for details

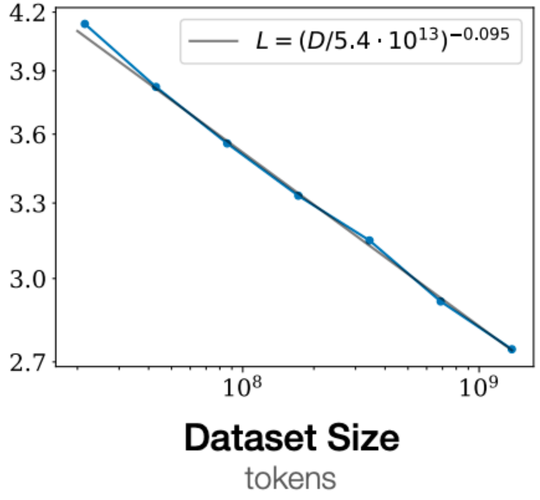
- The empirical scaling behaviors in large transformer models show ***model size***, ***dataset size***, and ***compute budget*** are factors for better performance

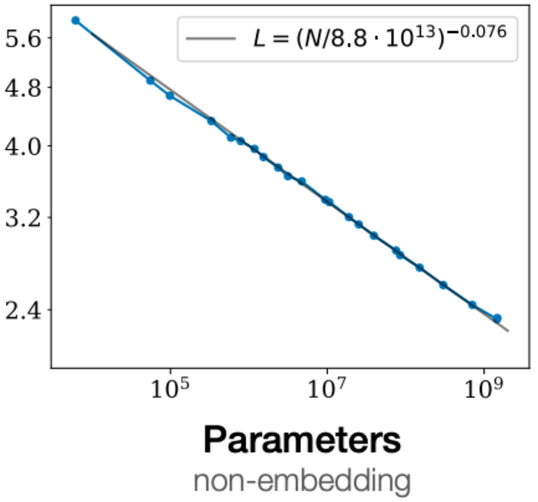
Test loss





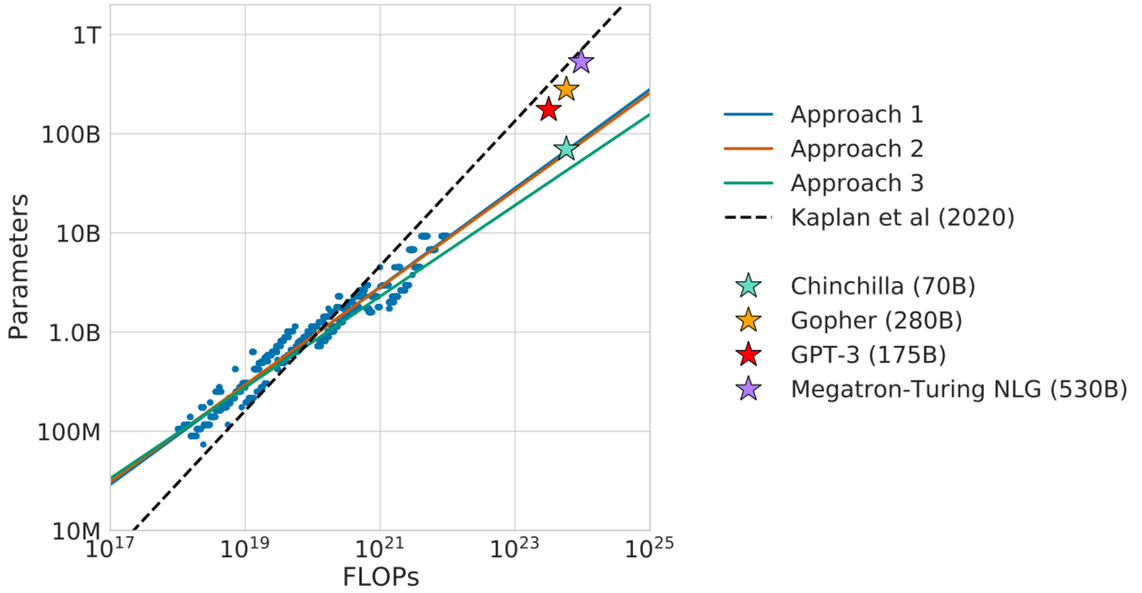






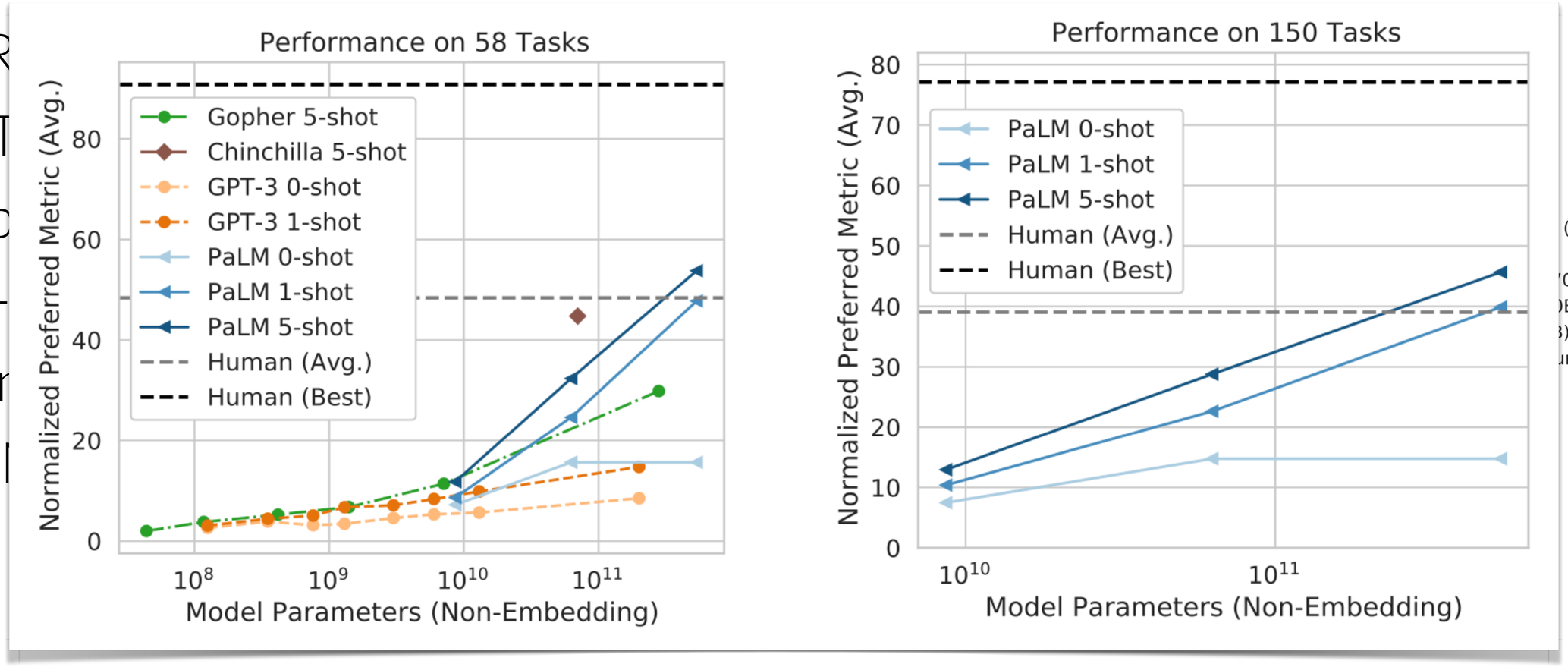
The Era of Large-Scale Language Models

- BERT (Devlin et al., ***NAACL*** 2019) Google AI
- GPT-3 (Brown et al., ***NeurIPS*** 2020) OpenAI
- Gopher (Rae et al., 2021) DeepMind
- MT-NLG (Smith et al., 2022) Microsoft NVIDIA
- Chinchilla (Hoffmann et al., 2022) DeepMind
- PaLM (Chowdhery et al., 2022) Google AI



The Era of Large-Scale Language Models

- BER
- GPT
- Gop
- MT-
- Chir
- PaLI



(2020)

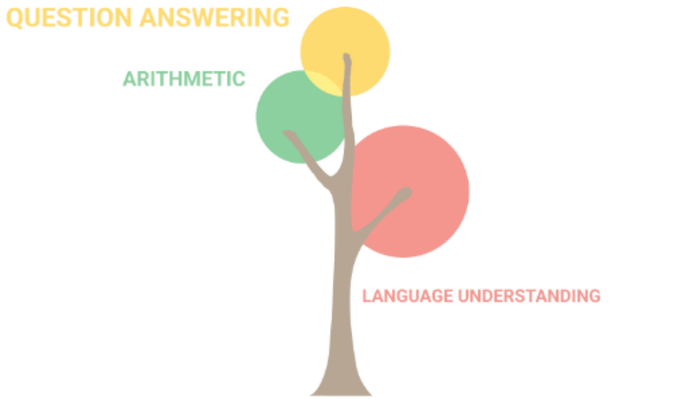
0B)

B)

)

uring NLG (530B)

LLM Can Learn Multi-Tasks



Google AI Blog (2022)

PaLM: Scaling Language Modeling with Pathways, Chowdhery et al., 2022

Which open sources can we use?

- PyTorch
  - `nn.DataParallel`
  - `nn.DistributedDataParallel`
- DeepSpeed
  - targets model parallelism & faster training
  - users can run bigger models on a single GPU without OOM
- Ray
  - users can run your code in multiple machines

Q&A