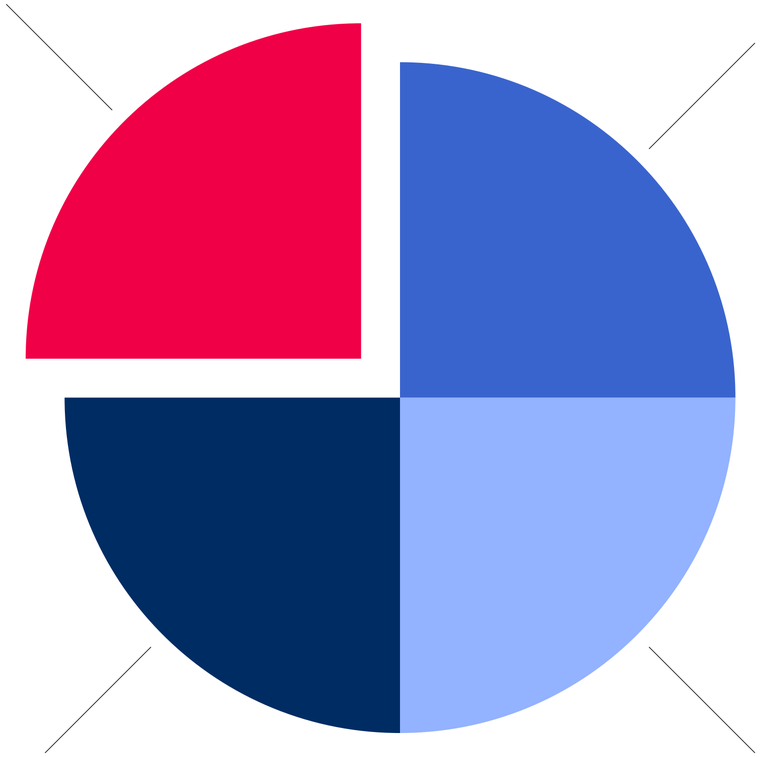
CS-461

Foundation Models and Generative AI

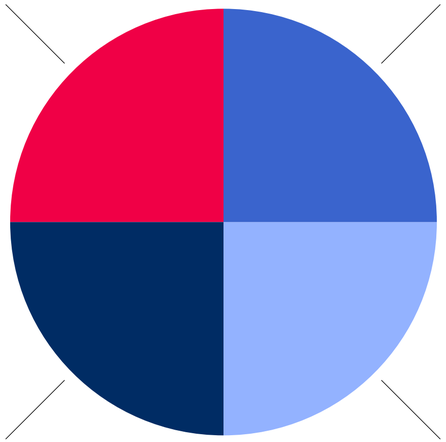
Learning at Scale: Supervised, Self-Supervised, and Beyond

Charlotte Bunne, Fall Semester 2025/26

Learning Principles



Expected Risk Minimization



All learning minimizes expected risk.

The difference is in what we know!

$$\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(\theta; x, y)]$$

- **For supervised**: We have data samples x with labels y. Loss $\ell\left(f_\theta(x), y\right)$.
- **For unsupervised**: We *only* have data samples x. Loss defined on x alone, i.e., $\ell\left(f_\theta(x)\right)$.
- **For semi-supervised**: We have data samples x and few labeled but many unlabeled samples. The loss is a mixture of $\ell_{\text{sup}} + \lambda \cdot \ell_{\text{unsup}}$.
- **For self-supervised**: We only have data samples x but will create *synthetic labels* from x. We will create the task from x itself, i.e., $\ell\left(f_\theta\left(x_{\text{partial}}\right), x_{\text{hidden}}\right)$.

Supervised Learning

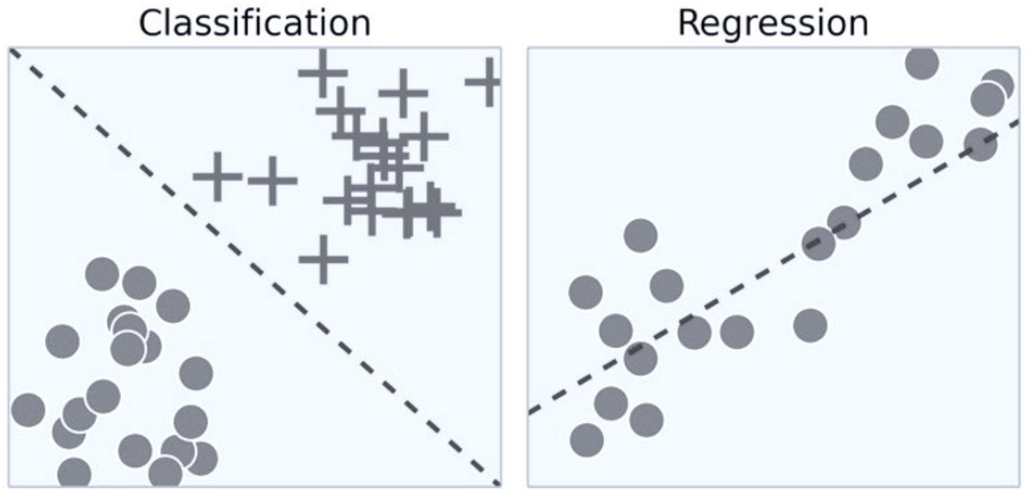
Supervised

Goal: Learn a mapping $x \mapsto y$ from labelled data.

Data: $\big((x_1, y_1), \ldots, (x_n, y_n)\big)$, i.e., pairs of data points and corresponding label.

Model: find function f with parameters θ such that $f_\theta : x \mapsto f_\theta(x)$.

If label y is discrete: **classification**
y is continous: **regression**



Supervised Learning

Supervised

Data: $\big((x_1, y_1), \ldots, (x_n, y_n)\big)$, i.e., pairs of data points and corresponding label.

Model: find function f with parameters θ such that $f_\theta : x \mapsto f_\theta(x)$.

If label y is discrete: **classification**

y is continous: **regression**

Loss: For classification, e.g., **Cross-Entropy Loss** or **Negative Log-Likelihood** (NLL)

K classes

$$\mathcal{L}_{\text{CE}} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} y_{i,k} \log\left(\hat{y}_{i,k}\right)$$

$y_{i,k}$: true label

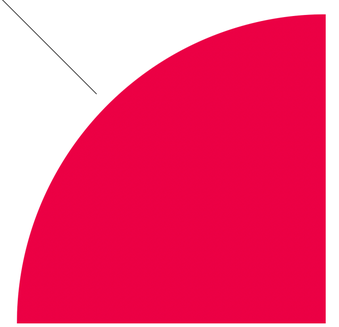
$\hat{y}_{i,k} = p_\theta\left(k \mid x_i\right)$: predicted probability for class k

For binary classification e.g., **Binary Cross-Entropy** (BCE)

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \log\left(\hat{y}_i\right) + \left(1 - y_i\right) \log\left(1 - \hat{y}_i\right)\right]$$

Supervised Learning

Supervised



Data: $\big((x_1, y_1), \ldots, (x_n, y_n)\big)$, i.e., pairs of data points and corresponding label.

Model: find function f with parameters θ such that $f_\theta : x \mapsto f_\theta(x)$.

If label y is discrete: **classification**

y is continous: **regression**

Loss: For regression, e.g., **Mean Squared Error** (MSE)

$$\mathcal{L} = \frac{1}{n} \sum_i \left(y_i - f_\theta\left(x_i\right)\right)^2$$

or Mean Absolute Error (MAE), Huber loss (for outliers), etc.

Unsupervised Learning

Goal: Discover structure in unlabeled data x.

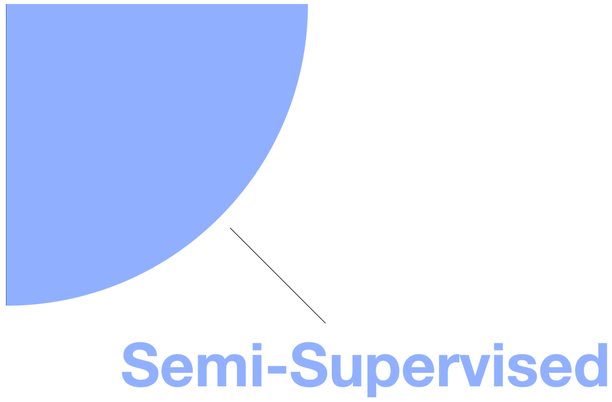
… clustering.

… dimensionality reduction.

… **generative modeling**.

Lecture 3-5: Generative Models

Semi-Supervised Learning



Goal: Combine small labeled and large unlabeled datasets.

$$\mathcal{L} = \mathcal{L}_{\text{supervised}} + \lambda \cdot \mathcal{L}_{\text{unsupervised}}$$

$\mathcal{L}_{\text{supervised}}$: Standard loss on labeled data

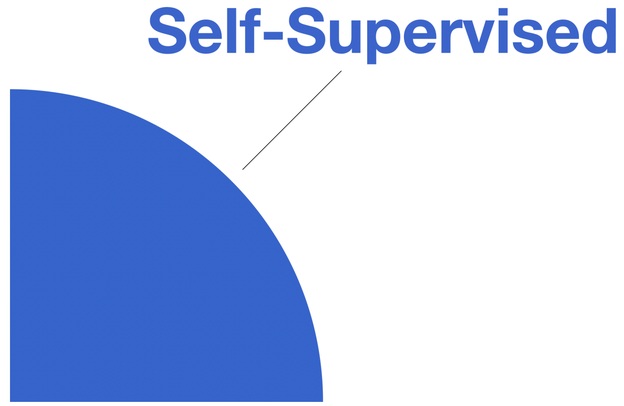
$\mathcal{L}_{\text{unsupervised}}$: **Consistency/pseudo-label loss** on unlabeled data

Key Idea: Adding unlabeled data does more than just increase dataset size.
The unsupervised learning component reveals the underlying structure of the data distribution, enabling more refined decision boundaries than labeled data alone could provide.

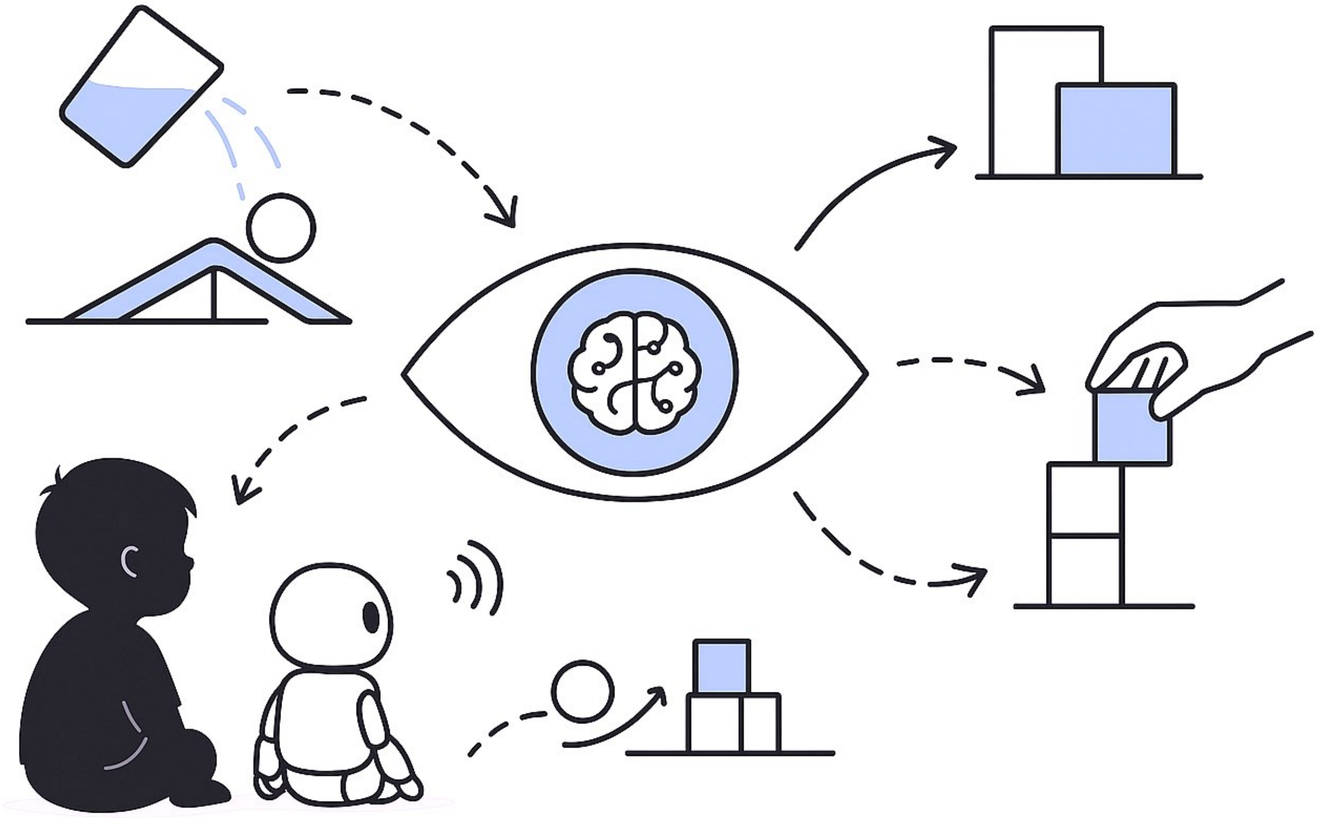
Key Assumptions:

1. **Smoothness**: Similar inputs should have similar outputs
2. **Cluster**: Data points in same cluster likely share same label
3. **Manifold**: Data lies on lower-dimensional manifold in high-dimensional space

Self-Supervised Learning



> The dark matter of intelligence. – LeCun and Misra (2021)



As babies, we learn how the world works largely by observation.

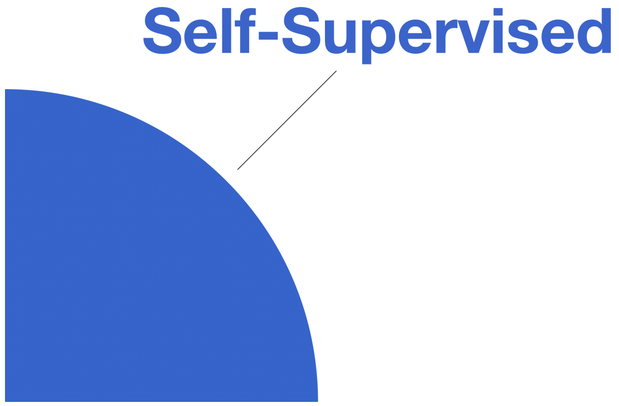
Later in life, we observe the world, act on it, observe again, and build hypotheses to explain how our actions change our environment by trial and error.

Generalized knowledge about the world, or common sense, forms the bulk of biological intelligence in both humans and animals.

Common sense is the dark matter of artificial intelligence.

Self-Supervised Learning

… is the basic training principle of foundation models.



Thomas G. Dietterich
@tdietterich

I propose that we adopt the term "Large Self-Supervised Models (LSSMs)" as a replacement for "Foundation Models" and "LLMs". "LLMs" don't capture non-linguistic data and "Foundation Models" is too grandiose. Thoughts? @percyliang

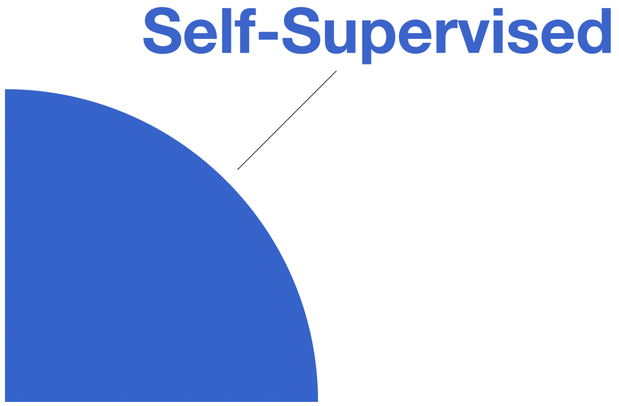
2:58 am · 13 Aug 2022

Percy Liang @percyliang · 13 Aug 2022

The beauty of language is that you can have multiple terms that highlight different aspects of the same object. You don't have to choose. I use "LLM" to talk about LLMs, "self-supervised" for their construction, and "foundation model" for their function. No term can be replaced.

1 4 33

Self-Supervised Learning



Goal: Create surrogate tasks from raw data, no human labels.

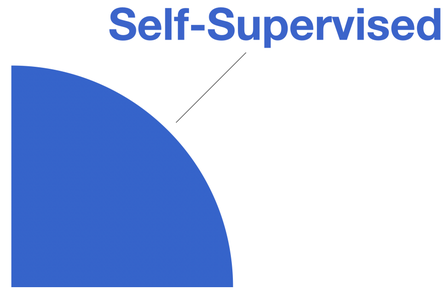
Data: Unlabeled data $(x_1, \ldots, x_n)$ but we create supervision from the data itself.

Core Idea: Design **pretext tasks** that force the model to learn useful representations without labels y.

Key Insight: The data contains its own supervisory signal through structure, context, transformations.

How? Use unlabeled data for initial pretraining of a model to learn general features, which are then adapted to specific tasks through fine-tuning with a smaller labeled dataset.

Self-Supervised Learning



1. Pretraining

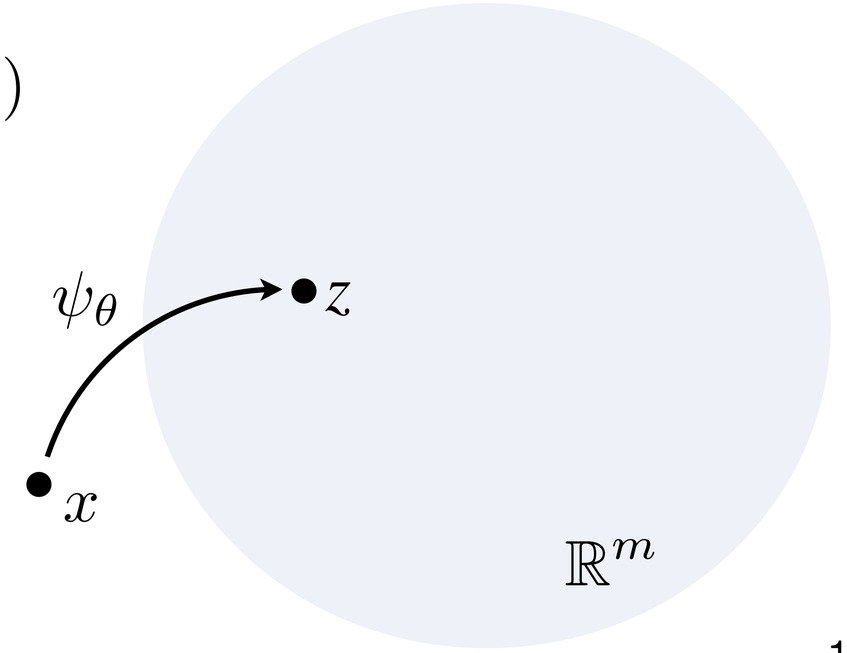
Data $(x_1, \ldots, x_n)$

Model $\psi_\theta : x \mapsto \psi_\theta(x) \in \mathbb{R}^m$ ⟵ *representation / embedding*

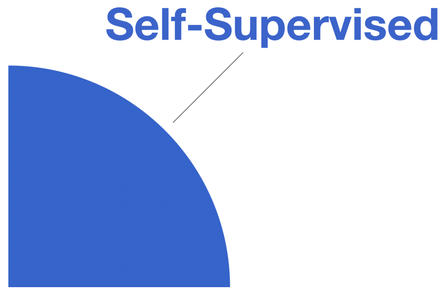
Optimize $\mathcal{L}_{\text{pre}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell_{\text{pre}}(\psi_\theta, x_i)$

to obtain $\hat{\theta}$

and pretrained model $\psi_{\hat{\theta}}$



Self-Supervised Learning



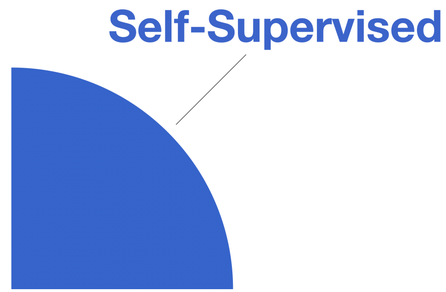
2. Evaluation Data $\left((x_1^{\text{task}}, y_1^{\text{task}}), \ldots, (x_{n_t}^{\text{task}}, y_{n_t}^{\text{task}})\right)$ n_t : number of downstream task samples

$n_t = 0$	**zero-shot learning**	Model performs new tasks without any task-specific examples, using only pretrained representations
$n_t = \text{few}$	**few-shot learning**	Model adapts to new tasks with just a handful of labeled examples (typically 1-10 per class)

→ **Tests true generalization**:
Can the model understand and solve novel tasks beyond its training distribution?

Lecture 10: Emergent Behaviors

Self-Supervised Learning



2. Evaluation Data $\left((x_1^{\text{task}}, y_1^{\text{task}}), \ldots, (x_{n_t}^{\text{task}}, y_{n_t}^{\text{task}})\right)$

Downstream task via **linear probing / linear head**

prediction model $w^\top \psi_{\hat{\theta}}(x), \quad w \in \mathbb{R}^m$

→ train w via $\min_w \frac{1}{n_t} \sum_{i=1}^{n_t} \ell_{\text{task}} \left(y_i^{\text{task}}, w^\top \psi_{\hat{\theta}}(x_i^{\text{task}})\right)$

$\hat{\theta}$❄: *foundation model parameters are frozen*

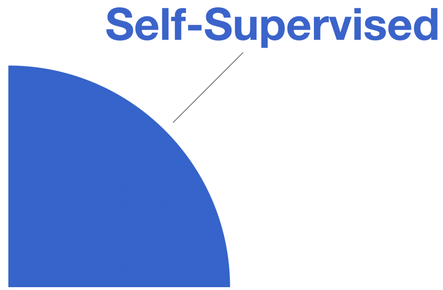
3. Finetuning Optimize both w and θ on downstream task

Initialize $\theta \leftarrow \hat{\theta}$ and w randomly

Lecture 5-9: Architectures

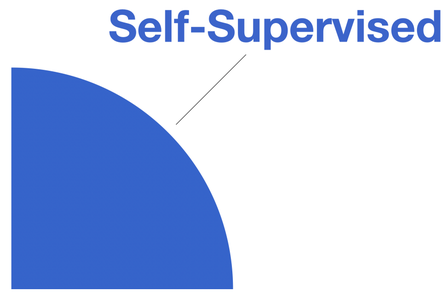
Self-Supervised Learning

How?



1. **Contrastive Learning**
2. **Masked Learning**
3. **Autoregressive Learning**
4. Clustering-Based Methods
5. Distillation-Based Methods

Self-Supervised Learning: Contrastive Learning



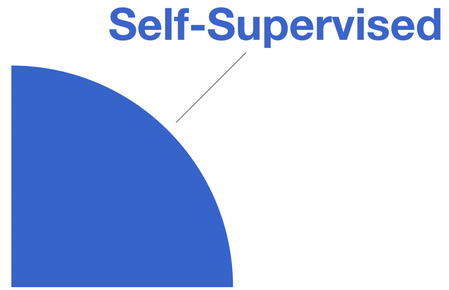
Principle: Learn representations by **contrasting** similar (positive) and dissimilar (negative) pairs.

Defining Positive and Negative Pairs Without Labels

+ **Positive Pairs**: Different views of the same semantic content
  - Created through augmentations that preserve meaning
  - **Assumption**: these transformations do not change the underlying concept

- **Negative Pairs**: Views from different instances
  - Typically all other samples in the batch
  - **Assumption**: random samples are likely semantically different

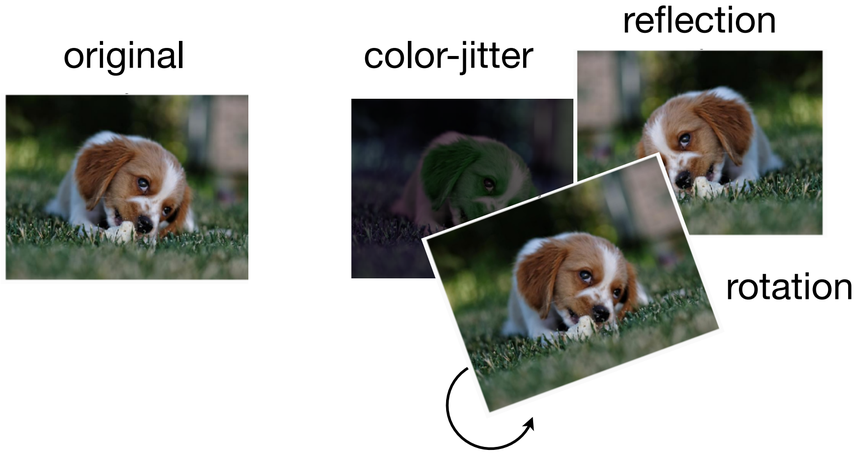
Self-Supervised Learning: Contrastive Learning



Defining Positive and Negative Pairs Without Labels

… in **vision**

+ Positive Pairs

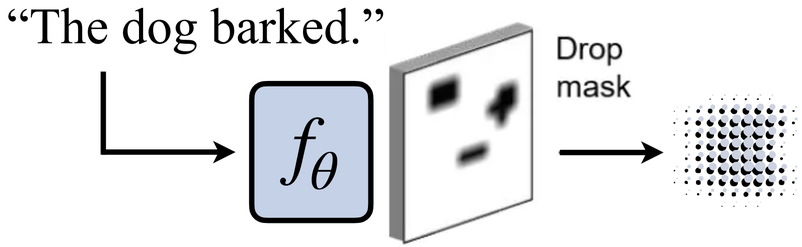


- Negative Pairs

different image

Why? A cropped dog photo is still a dog; different photos are likely different objects.

… in **language**

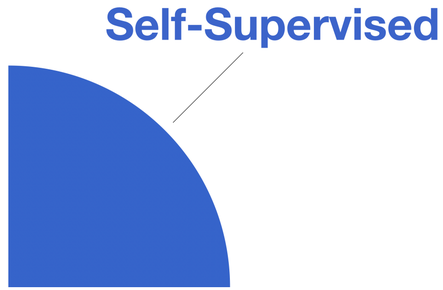


embedding of the same sentence with different dropout masks

“The cat meowed.” different sentence

Why? “The dog barked" ≈ “The dog woofed” ≠ random other sentences

Self-Supervised Learning: Contrastive Learning



Evolution of Contrastive Learning Objectives

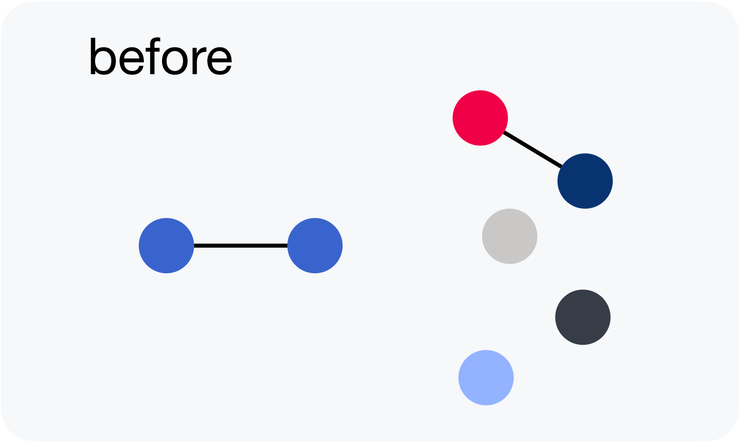
Early Approaches: Single Pairs

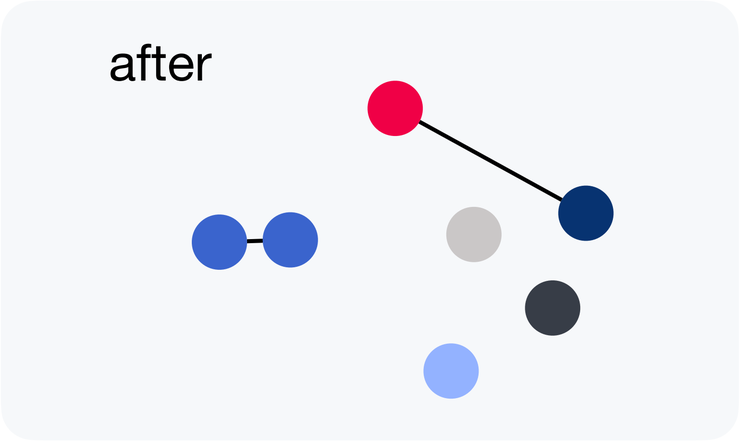
Contrastive Loss

$$\mathcal{L}_{\text{contr}}\left(\mathbf{x}_i, \mathbf{x}_j, \theta\right) = \mathbb{1}\left[y_i = y_j\right] \left\| f_\theta\left(\mathbf{x}_i\right) - f_\theta\left(\mathbf{x}_j\right)\right\|_2^2 + \mathbb{1}\left[y_i \neq y_j\right] \max\left(0, \epsilon - \left\| f_\theta\left(\mathbf{x}_i\right) - f_\theta\left(\mathbf{x}_j\right)\right\|_2\right)^2$$

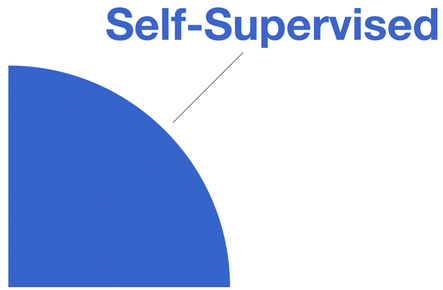
hyperparameter defining the lower bound distance between samples of different classes

Chopra et al., (2005)





Self-Supervised Learning: Contrastive Learning



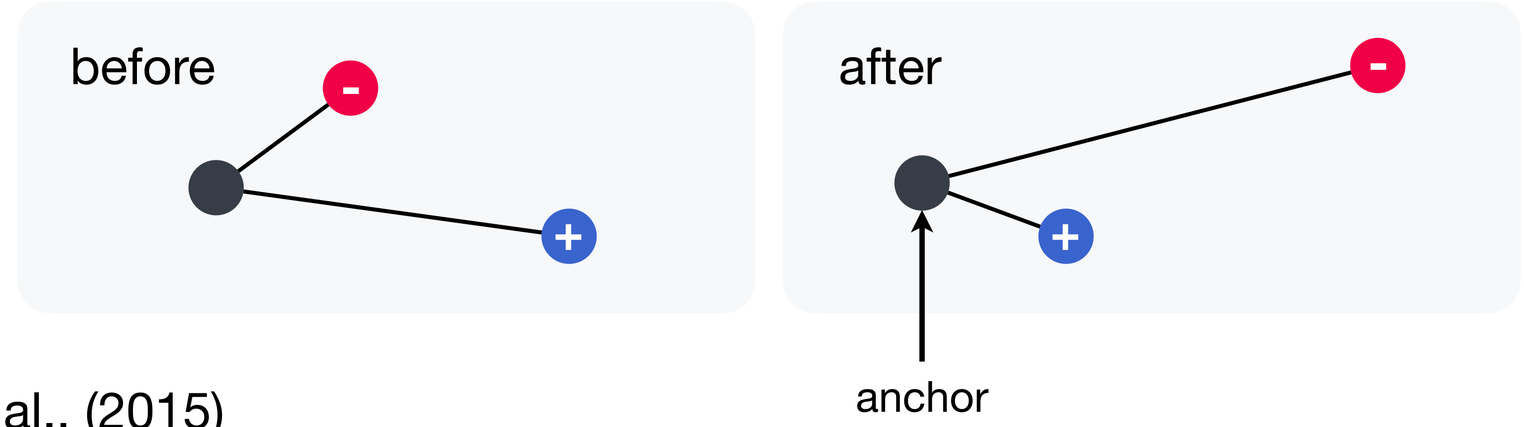
Triplet loss considers an anchor, a positive, and a negative sample simultaneously.
The contrastive loss works with pairs (positive or negative) independently.

→ Triplet loss is less greedy, as it considers relative distances between all three samples.

Triplet Loss

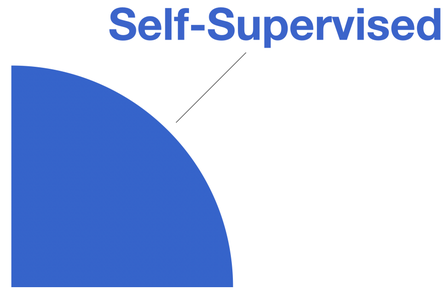
$$\mathcal{L}_{\text{triplet}}\left(\mathbf{x}, \mathbf{x}^{+}, \mathbf{x}^{-}\right) = \sum_{\mathbf{x} \in \mathcal{X}} \max\left(0, \left\|f(\mathbf{x}) - f\left(\mathbf{x}^{+}\right)\right\|_2^2 - \left\|f(\mathbf{x}) - f\left(\mathbf{x}^{-}\right)\right\|_2^2 + \epsilon\right)$$

margin parameter set as minimum offset between distances of similar vs. dissimilar pairs



Schroff et al., (2015)

Self-Supervised Learning: Contrastive Learning



Modern Approaches: Multiple Pairs

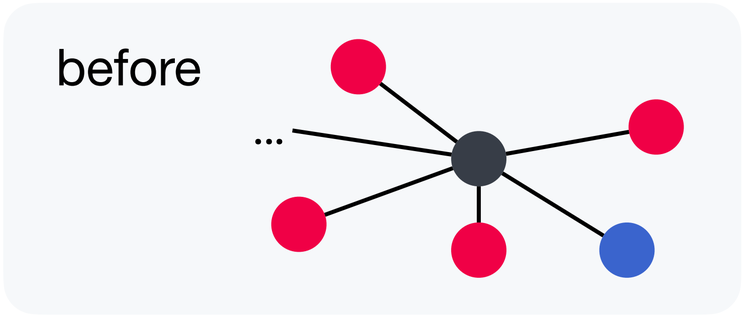
Generalizes triplet loss to include comparison with multiple negative samples.

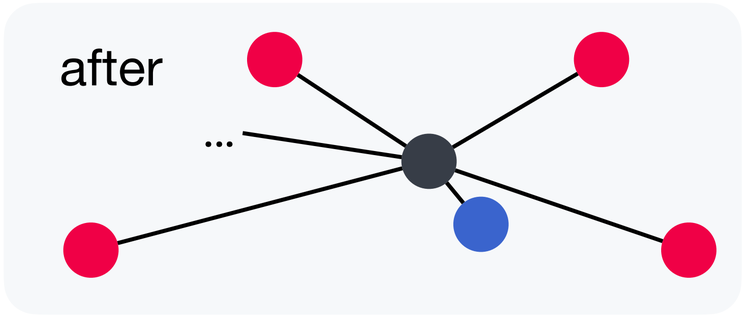
N-Pair Loss

$$\mathcal{L}_{\text{N}-\text{pair}}\left(\mathbf{x}, \mathbf{x}^+, \{\mathbf{x}_i^-\}_{i=1}^{N-1}\right) = \log\left(1 + \sum_{i=1}^{N-1} \exp\left(f(\mathbf{x})^\top f\left(\mathbf{x}_i^-\right) - f(\mathbf{x})^\top f\left(\mathbf{x}^+\right)\right)\right)$$

$$= -\log \frac{\exp\left(f(\mathbf{x})^\top f\left(\mathbf{x}^+\right)\right)}{\exp\left(f(\mathbf{x})^\top f\left(\mathbf{x}^+\right)\right) + \sum_{i=1}^{N-1} \exp\left(f(\mathbf{x})^\top f\left(\mathbf{x}_i^-\right)\right)}$$

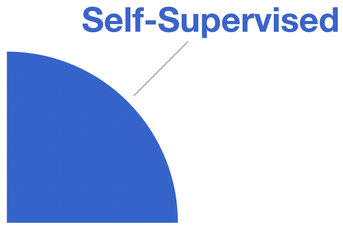
Sohn (2016)





Note: If we only sample one negative sample per class, it is equivalent to the softmax loss for multi-class classification.

Self-Supervised Learning: Contrastive Learning



InfoNCE Loss

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}_{c\sim p(c)}\mathbb{E}_{x\sim p(x|c)}\mathbb{E}_{\{x_i\}\sim p(x)}\left[\log \frac{\exp\left(f'(x)^\top f(c)/\tau\right)}{\exp\left(f'(x)^\top f(c)/\tau\right) + \sum_{i=1}^{N-1}\exp\left(f'\left(x_i\right)^\top f(c)/\tau\right)}\right]$$

Derivation: Noise Contrastive Estimation (NCE)

i.e., a method for estimating parameters of a statistical model

Idea: Instead of learning $p_{\text{data}}(x)$, it learns to distinguish data from noise ← contrastive!

How? Logistic regression with binary crossentropy loss

log-odds ratio

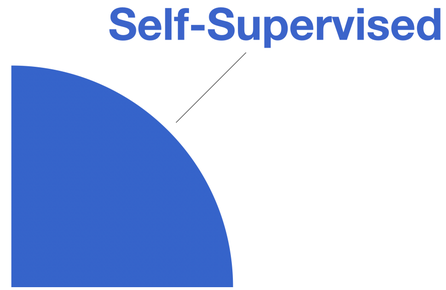
$$s_\theta(x) = \log \frac{p_\theta(x)}{p_n(x)}$$

NCE Loss

$$\mathcal{L}_{\text{NCE}} = -\mathbb{E}_{x\sim p_{\text{data}}}\left[\log \sigma\left(s_\theta(x)\right)\right] - k \cdot \mathbb{E}_{x\sim p_n}\left[\log\left(1-\sigma\left(s_\theta(x)\right)\right)\right]$$

number of noise samples per real sample

Self-Supervised Learning: Contrastive Learning



Generalize binary classification in NCE to N-way classification.
InfoNCE asks 'which one of these N samples is the positive one?'

InfoNCE Loss

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}_{c\sim p(c)}\mathbb{E}_{x\sim p(x|c)}\mathbb{E}_{\{x_i\}\sim p(x)}\left[\log \frac{\exp\left(f'(x)^\top f(c)/\tau\right)}{\exp\left(f'(x)^\top f(c)/\tau\right) + \sum_{i=1}^{N-1}\exp\left(f'\left(x_i\right)^\top f(c)/\tau\right)}\right]$$

van den Oord et al., (2018)

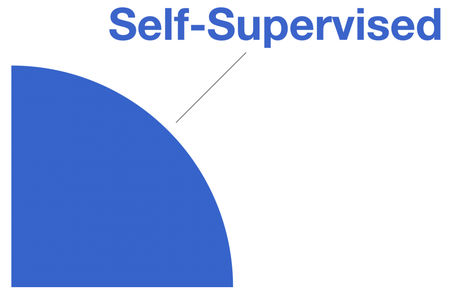
- encoders f, f'
- context c (anchor) and positive sample x (e.g., its augmentation)
- positive pair: (c, x) where $x \sim p(x \mid c)$ and negative samples: $\left\{x_i\right\}_{i=1}^{N-1}$ where $x_i \sim p(x)$
- score function: $s(x, c) = \exp\left(f'(x)^\top f(c)/\tau\right)$

temperature that controls how "sharp" or "smooth" the similarity distribution is

Exercise 1 · Task 1

→ connection to **information theory**

Self-Supervised Learning: Contrastive Learning



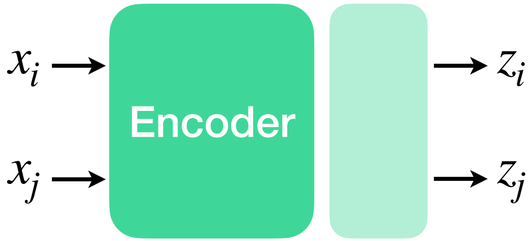
Example from Vision: **SimCLR Framework**

… framework for contrastive learning of visual representations by maximizing agreement between differently augmented views of the same image

Code Notebook 1 · Task A

Strategy: Data Augmentation as Supervision

batch of N,
data augmentation to $2N$
with $2(N-1)$ negative examples

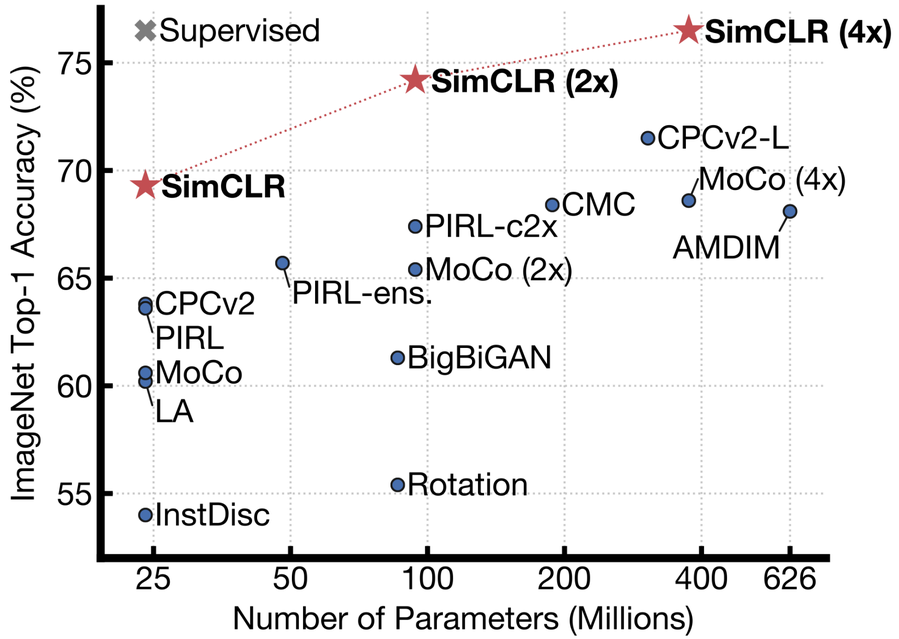


Loss: Symmetric version of InfoNCE

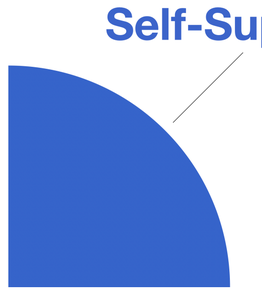
$$\mathcal{L}_{\text{SimCLR}}^{(i,j)} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}$$

Chen et al., (2018)

Key to Success: Large Batch Sizes



Self-Supervised Learning: Contrastive Learning

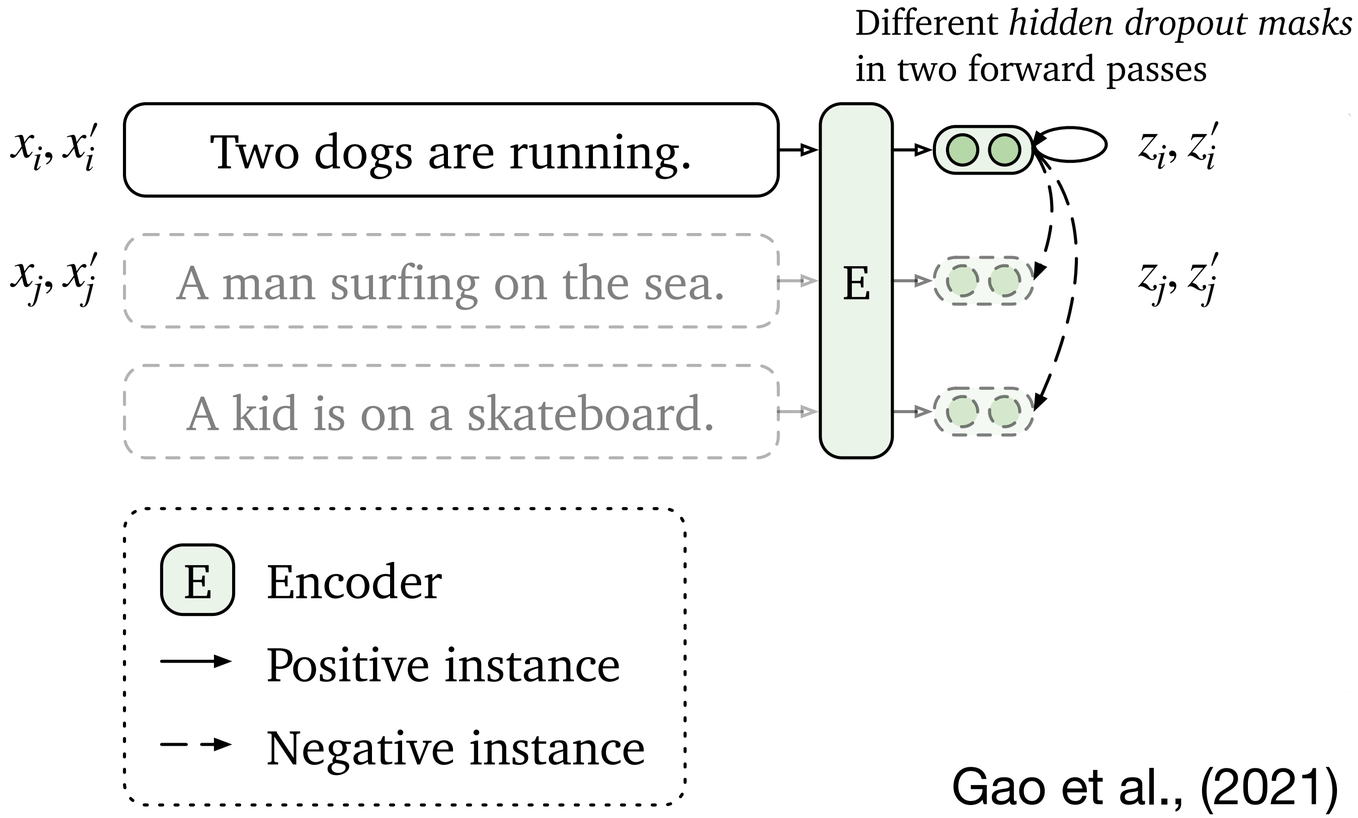


Example from Language: **SimCSE Framework**

… applies contrastive learning to create high-quality sentence embeddings using **dropout as minimal augmentation**

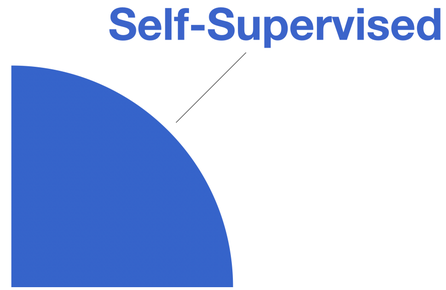
Loss: Symmetric version of InfoNCE

Here, z_i, z_i' is the positive pair.



Gao et al., (2021)

Self-Supervised Learning: Masked Learning

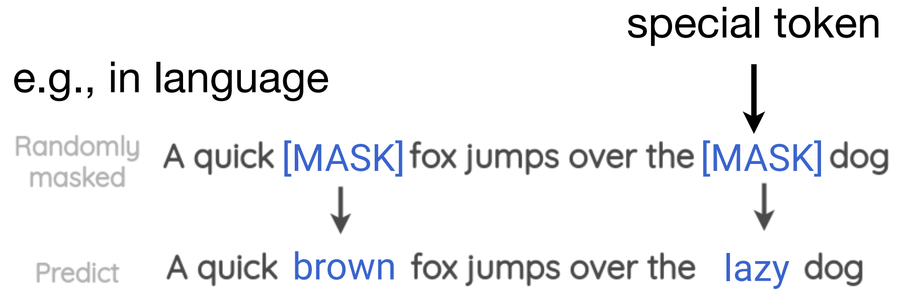


Principle: Hide parts of the data and train the model to predict what is missing.

This forces the model to understand the structure and relationships within the data.

Data $(x_1, x_2, \ldots, x_n)$ and mask $M \subset \{1, \ldots, m\}$.

Predict masked parts from unmasked parts $p(x_M \mid x_{\setminus M})$.



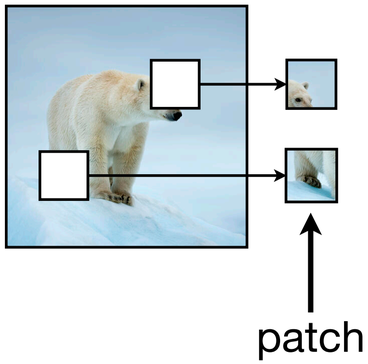
Design choices between *random*
structured
adversarial
curriculum

masking

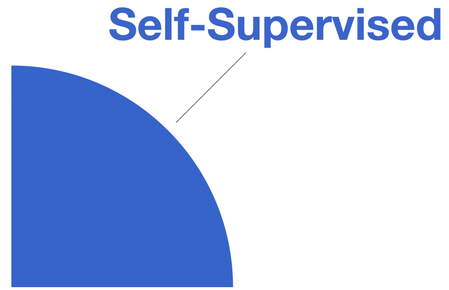
simple and unbiased
mask semantically meaningful units
mask the most informative parts
start with easy masking, increase difficulty

→ 15-75% depending on domain.

e.g., in vision



Self-Supervised Learning: Masked Learning



General Formulation

position-wise loss function
dependent on domain

Masking Loss

$$\mathcal{L}_{\text{mask}} = \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{M}\left[\frac{1}{|M|}\sum_{i\in M}\ell(x_i,\hat{x}_i)\right]$$

$\hat{x}_i = f_\theta\left(x_{\setminus M}\right)_i$ is the model's prediction based on unmasked parts for position i

For **Language**:

$$\mathcal{L}_{\text{MLM}} = -\mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{M}\left[\frac{1}{|M|}\sum_{i\in M}\log p_\theta\left(x_i \mid x_{\setminus M}\right)\right]$$

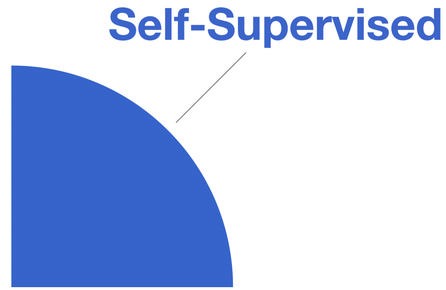
cross-entropy between true distribution δ_{x_i} (one-hot at true token)
and predicted distribution $p_\theta(\,\cdot\,|x_{\setminus M})$

For **Vision**:

$$\mathcal{L}_{\text{MAE}} = \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{M}\left[\frac{1}{|M|}\sum_{i\in M}\left\|x_i - f_\theta\left(x_{\setminus M}\right)_i\right\|^2\right]$$

pixel/patch
reconstruction

Self-Supervised Learning: Masked Learning



Example from Vision:

Masked Autoencoder (MAE)

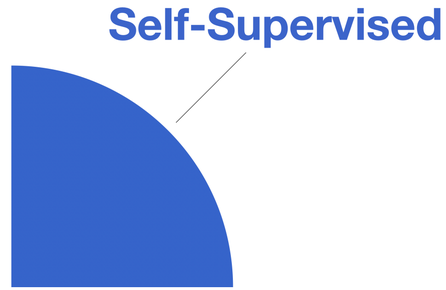
Lecture 6: Vision Foundation Models

Example from Language:

Bidirectional Encoder Representations from Transformers (BERT)

Lecture 6: Language Foundation Models

Self-Supervised Learning: Autoregressive Learning



Principle: Learn by **predicting next** elements in sequence.

Autoregressive Loss

$$\mathcal{L}_{\mathrm{AR}} = -\mathbb{E}_{x\sim\mathcal{D}}\left[\sum_{t=1}^{T}\log p_\theta\left(x_t \mid x_{<t}\right)\right]$$

Any joint distribution can be factored as a product of conditionals:

chain rule of probability

prefix/history $x_{<t} = (x_1, \ldots, x_{t-1})$

$$p\left(x_1, \ldots, x_T\right) = \prod_{t=1}^{T} p\left(x_t \mid x_{<t}\right)$$

Characteristics:

1. **Exact factorization**, i.e., no approximation, allows to objectively compare models, detect anomalies, etc.
2. **Causal structure** that enforces temporal/sequential dependencies, i.e., foundation to generate sequences, one token at a time, left to right.
3. Deep **connection to information theory**, i.e., ideal AR model achieves Shannon-optimal compression.

Excursion: **Perplexity**

Perplexity (PPL) is the exponential of the average negative log-likelihood per token:

Perplexity

$$\mathrm{PPL} = \exp\left(-\frac{1}{T}\sum_{t=1}^{T}\log p\left(x_t \mid x_{<t}\right)\right) = p\left(x_1, x_2, \ldots, x_N\right)^{-1/T}$$

Intuition: How "surprised" the model is on average

- PPL = 10 → Model is as confused as choosing uniformly from 10 options
- PPL = 1 → Perfect prediction (no surprise)

Historical Origin:

- **Information Theory** (1948): Claude Shannon introduced as $2^{H(X)}$ (exponential of entropy H)
- **Language Modeling** (1970s-80s): Frederick Jelinek and IBM team popularized for speech recognition
- **Modern NLP**: Standard metric since statistical language models replaced rule-based systems

Excursion: **Pseudo-Likelihood**

Autoregressive strategies directly learn the joint probability through the chain rule, allowing to compute the exact likelihood *and* perplexity → direct way to evaluate models!

What probability distribution do approaches based on masking learn?

→ Masked language models (MLM) optimize something called the **pseudo-likelihood**!

What is the difference? True joint probablilty: $p\left(x_1, x_2, x_3\right) = p\left(x_1\right) \cdot p\left(x_2 \mid x_1\right) \cdot p\left(x_3 \mid x_1, x_2\right)$

Pseudo joint probability: $p_{\text{pseudo}}\left(x_1, x_2, x_3\right) = p\left(x_1 \mid x_2, x_3\right) \cdot p\left(x_2 \mid x_1, x_3\right) \cdot p\left(x_3 \mid x_1, x_2\right)$

Pseudo-Likelihood

$$p_{\text{pseudo}}\left(x_1, \ldots, x_T\right) = \prod_{t=1}^{T} p\left(x_t \mid x_{\setminus t}\right)$$

→ approximation to the joint probability distribution

Excursion: **Pseudo-Perplexity**

Exercise 1 · Task 2

Masked language models learn the pseudo-likelihood

Negative Pseudo-Log-Likelihood

$$\mathcal{L}_{\text{MLM}} = q \cdot \text{NPLL}(\theta) = -q \cdot \mathbb{E}_{x \sim \mathcal{D}} \left[\sum_{t=1}^{T} \log p_\theta \left(x_t \mid x_{\setminus t} \right) \right]$$

Following the same structure as perplexity:

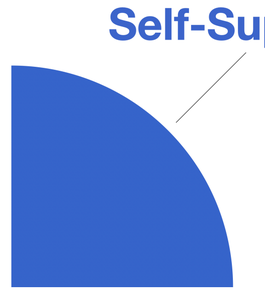
Pseudo-Perplexity

$$\text{Pseudo-PPL} = \exp \left(-\frac{1}{T} \sum_{t=1}^{T} \log p \left(x_t \mid x_{\setminus t} \right) \right)$$

Note: These metrics are not comparable to each other despite similar names and formulas.

Remark: For continuous data (e.g., image patches), pseudo-likelihood extends by assuming Gaussian conditionals $p \left(x_t \mid x_{\setminus t} \right) = \mathcal{N} \left(f_\theta \left(x_{\setminus t} \right), \sigma^2 \right)$, making MSE minimization equivalent to maximizing pseudo-likelihood.

Self-Supervised Learning: Autoregressive Learning



Example from Language:

Generative Pretrained Transformer (GPT)

Radford et al., (2018)

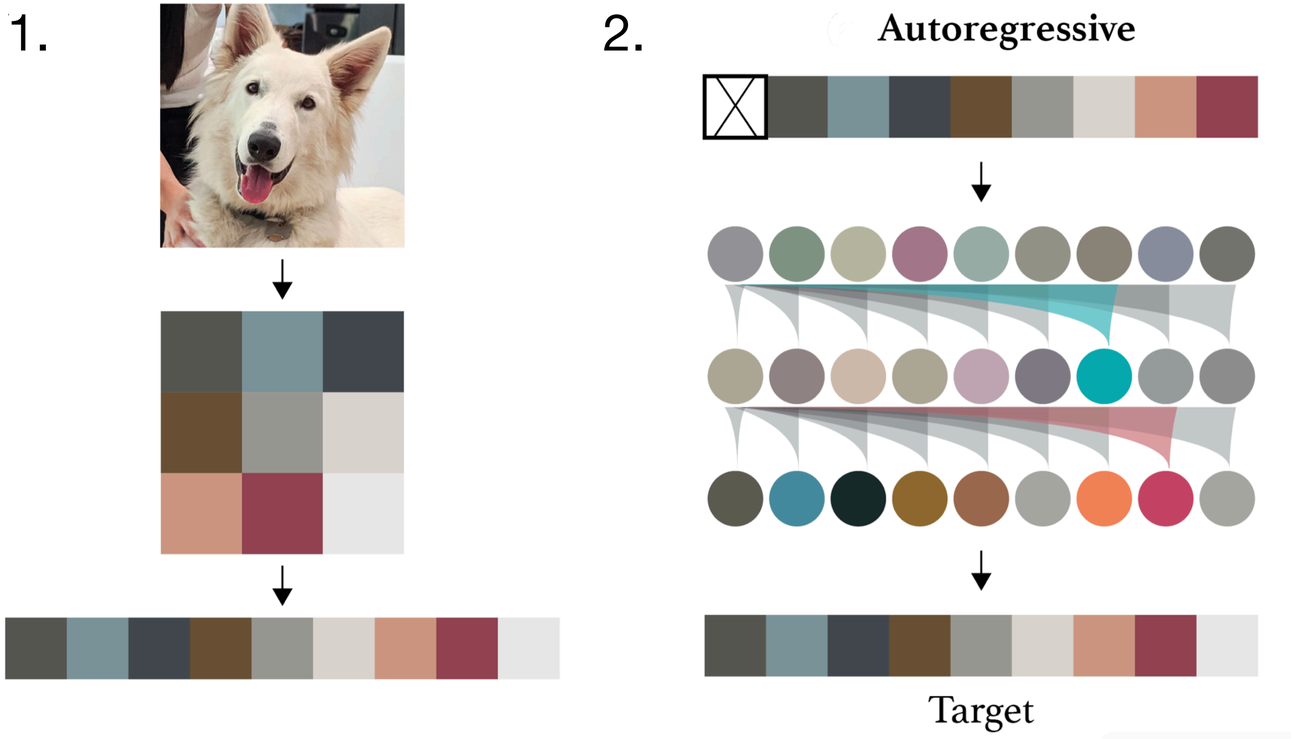
Lecture 6: Language Foundation Models

Example from Vision:

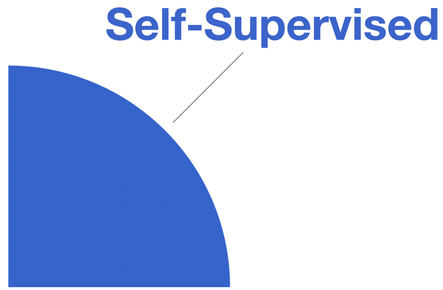
Image GPT (iGPT)

Chen et al., (2020)

1. Pre-process raw images by resizing to a low resolution and reshaping into a 1D sequence.
2. Train via auto-regressive next pixel prediction.



Self-Supervised Learning: Fusions of Learning Principles



Combining multiple self-supervised learning strategies leverages their complementary strengths to learn richer, more versatile representations than any single approach alone.

e.g., **ELECTRA**

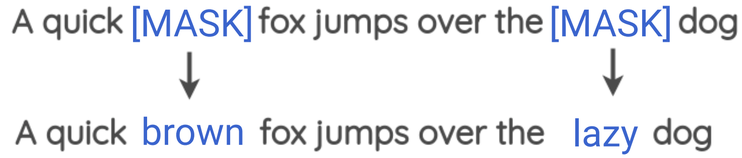
Combines: Masked Learning + Contrastive Learning

How: Generator creates replacements for masked tokens, discriminator distinguishes real vs. replaced tokens.

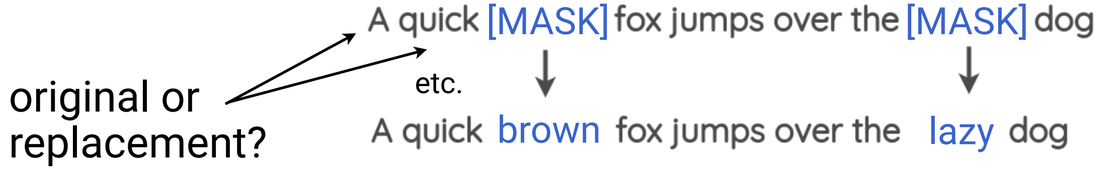
Benefit: More sample-efficient than pure masked learning.

Clark et al., (2020)

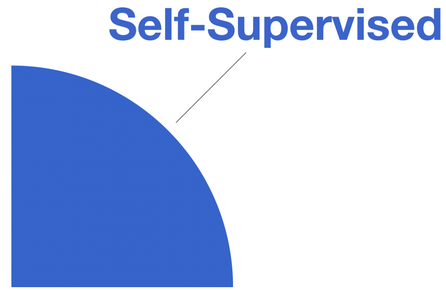
Masked language models:
learn from 15% of tokens



ELECTRA:
learns from 100% of tokens



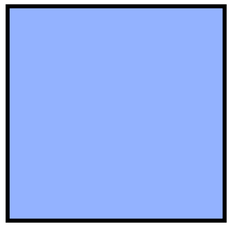
From Learning Principle to Architecture



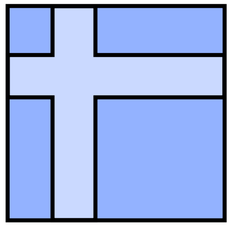
Important!

The **self-supervision objective** is *not* a training detail but the fundamental constraint that **shapes many architectural choices** from attention patterns to encoder-decoder asymmetry.

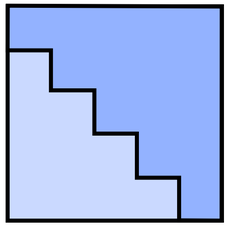
1. **Contrastive** → Must compare → Needs batch/memory
   - Requires denominator with many negatives: $\exp(\text{ sim }^{+})/\sum \exp(\text{sim})$
   - **Solution**: Large batches (SimCLR: 4096) or memory banks (MoCo: 65K queue)
2. **Masked** → Must see all → Needs bidirectional
   - Predict from full context: $p\left(x_{\text{mask}} \mid x_{\text{left}}, x_{\text{right}}\right)$
   - **Solution**: Full attention patterns
3. **Autoregressive** → Must generate → Needs causality
   - Factorization chain: $p(x) = \prod p\left(x_t \mid x_{<t}\right)$
   - **Solution**: Causal self-attention to avoid future leakage



full attention

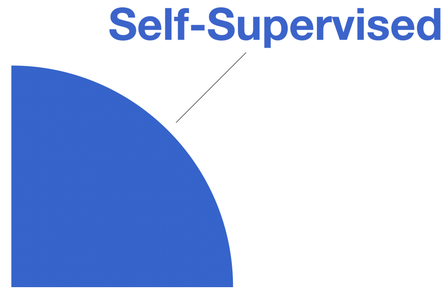


bidirectional masked attention



causal attention

From Learning Principle to Architecture

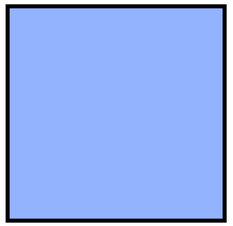


Important!

The **self-supervision objective** is *not* a training detail but the fundamental constraint that **shapes many architectural choices** from attention patterns to encoder-decoder asymmetry.

1. **Contrastive**

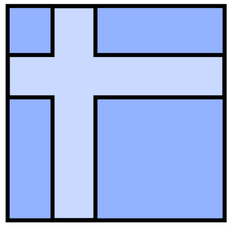
```python
attn = (Q @ K.T) / sqrt(d_k)  # no mask needed
attn = softmax(attn)          # full attention
out = attn @ V                # then global pool
```



full attention

2. **Masked**

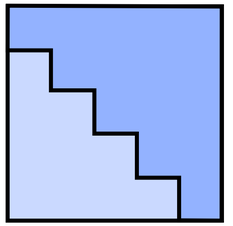
```python
attn = Q @ K.T              # full attention
attn[mask_idx, :] = -inf    # mask positions cannot attend
attn[:, mask_idx] = -inf    # nothing attends to masked
```



bidirectional attention

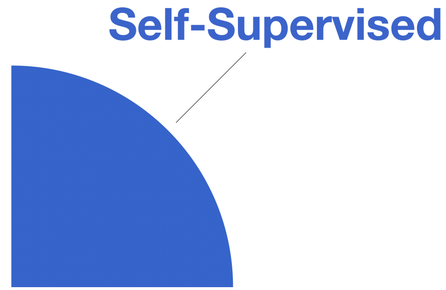
3. **Autoregressive**

```python
mask = torch.tril(torch.ones(n, n))
attn = (Q @ K.T).masked_fill(mask == 0, -inf)
```



causal attention

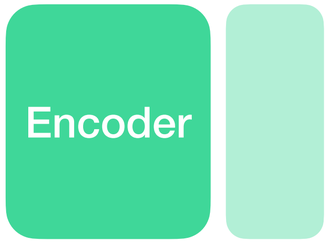
From Learning Principle to Architecture



Important!

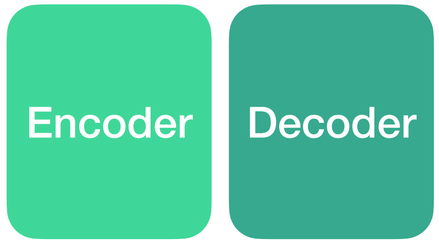
The **self-supervision objective** is *not* a training detail but the fundamental constraint that **shapes many architectural choices** from attention patterns to encoder-decoder asymmetry.

1. **Contrastive**



Encoder + projection head
for comparing global features

2. **Masked**



Encoder-decoder split that
can be asymmetric

3. **Autoregressive**

Unified decoder-only architecture
with causal masking

Lecture 5-7: Architectures

Model Size Evolution of GPT

GPT-1
117M
(2018)

GPT-2
1.5B
(2019)

GPT-3
175B
(2020)

+ sparse attention in alternating layers

→ architecture changed minimally

→ **scale can be more important than architecture**

Improving Language Understanding by Generative Pre-Training

Alec Radford, OpenAI, alec@openai.com
Karthik Narasimhan, OpenAI, karthikn@openai.com
Tim Salimans, OpenAI, tim@openai.com
Ilya Sutskever, OpenAI, ilyasu@openai.com

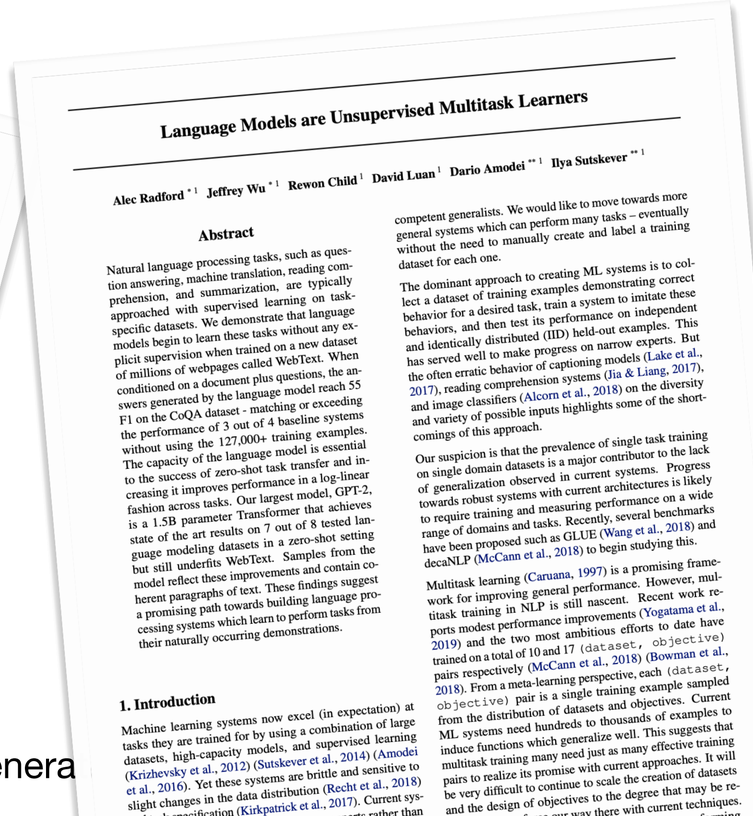
Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific task. In contrast to previous approaches, we make use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. We demonstrate the effectiveness of our approach on a wide range of benchmarks for natural language understanding. Our general task-agnostic model outperforms discriminatively trained models that use architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test), 5.7% on question answering (RACE), and 1.5% on textual entailment (MultiNLI).

1 Introduction

The ability to learn effectively from raw text is crucial to alleviating the dependence on supervised learning in natural language processing (NLP). Most deep learning methods require substantial amounts of manually labeled data, which restricts their applicability in many domains that suffer from a dearth of annotated resources [61]. In these situations, models that can leverage linguistic information from unlabeled data provide a valuable alternative to gathering more annotation, which can be time-consuming and expensive. Further, even in cases where considerable supervision is available, learning good representations in an unsupervised fashion can provide a significant performance boost. The most compelling evidence for this so far has been the extensive use of pre-trained word embeddings [10, 39, 42] to improve performance on a range of NLP tasks [8, 11, 26, 45].

Leveraging more than word-level information from unlabeled text, however, is challenging for two main reasons. First, it is unclear what type of optimization objectives are most effective at learning text representations that are useful for transfer. Recent research has looked at various objectives such as language modeling [44], machine translation [38], and discourse coherence [21], with each method outperforming the others on different tasks.[1] Second, there is no consensus on the most effective way to transfer these learned representations to a target task. Existing techniques involve a combination of making task-specific changes to the model architecture, using intricate learning schemes [21] and adding auxiliary learning objectives [50]. These uncertainties make it difficult to develop effec-



Language Models are Unsupervised Multitask Learners

Alec Radford [*1], Jeffrey Wu [*1], Rewon Child [1], David Luan [1], Dario Amodei [**1], Ilya Sutskever [**1]

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

1. Introduction

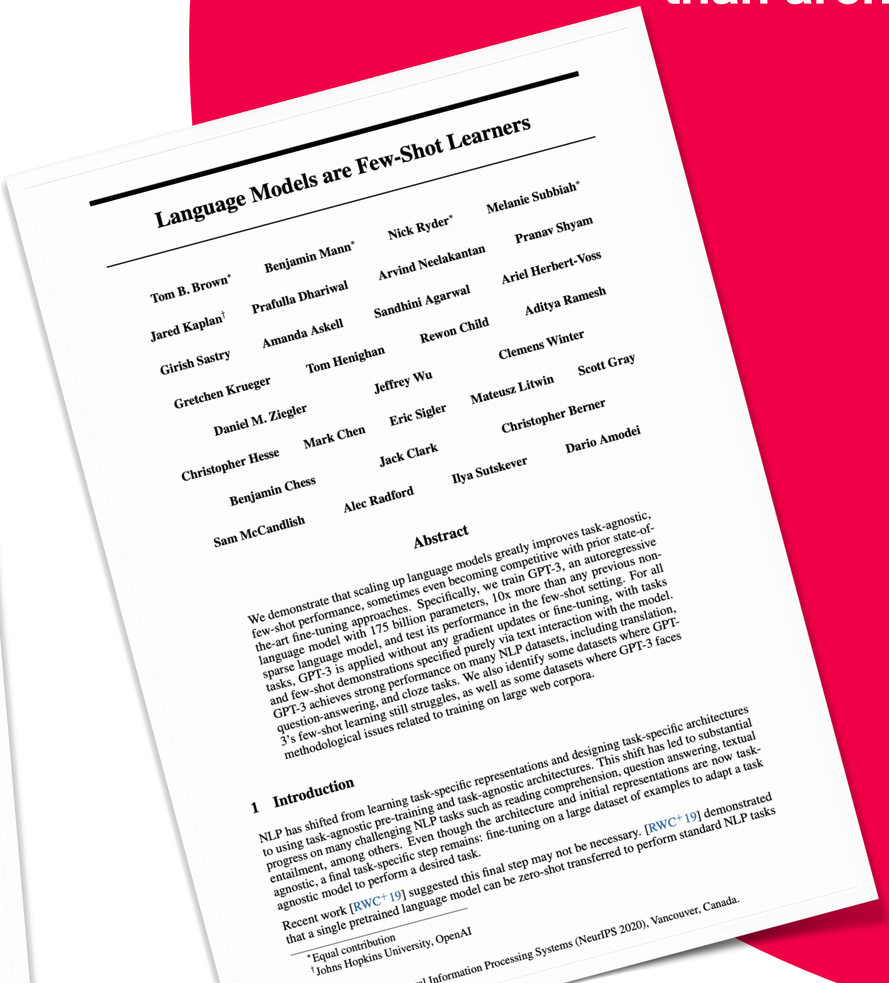
Machine learning systems now excel (in expectation) at tasks they are trained for by using a combination of large datasets, high-capacity models, and supervised learning (Krizhevsky et al., 2012) (Sutskever et al., 2014) (Amodei et al., 2016). Yet these systems are brittle and sensitive to slight changes in the data distribution (Recht et al., 2018) and task specification (Kirkpatrick et al., 2017). Current systems

competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Liang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and decaNLP (McCann et al., 2018) to begin studying this.

Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Yogatama et al., 2019) and the two most ambitious efforts to date have trained on a total of 10 and 17 (`dataset, objective`) pairs respectively (McCann et al., 2018) (Bowman et al., 2018). From a meta-learning perspective, each (`dataset, objective`) pair is a single training example sampled from the distribution of datasets and objectives. Current ML systems need hundreds to thousands of examples to induce functions which generalize well. This suggests that multitask training many need just as many effective training pairs to realize its promise with current approaches. It will be very difficult to continue to scale the creation of datasets and the design of objectives to the degree that may be re-



Language Models are Few-Shot Learners

Tom B. Brown*, Benjamin Mann*, Nick Ryder*, Melanie Subbiah*, Jared Kaplan†, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei

Abstract

We demonstrate that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even becoming competitive with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks. We also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora.

1 Introduction

NLP has shifted from learning task-specific representations and designing task-specific architectures to using task-agnostic pre-training and task-agnostic architectures. This shift has led to substantial progress on many challenging NLP tasks such as reading comprehension, question answering, textual entailment, among others. Even though the architecture and initial representations are now task-agnostic, a final task-specific step remains: fine-tuning on a large dataset of examples to adapt a task-agnostic model to perform a desired task.

Recent work [RWC+19] suggested this final step may not be necessary. [RWC+19] demonstrated that a single pretrained language model can be zero-shot transferred to perform standard NLP tasks

*Equal contribution
†Johns Hopkins University, OpenAI

Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

The Scale Breakthrough

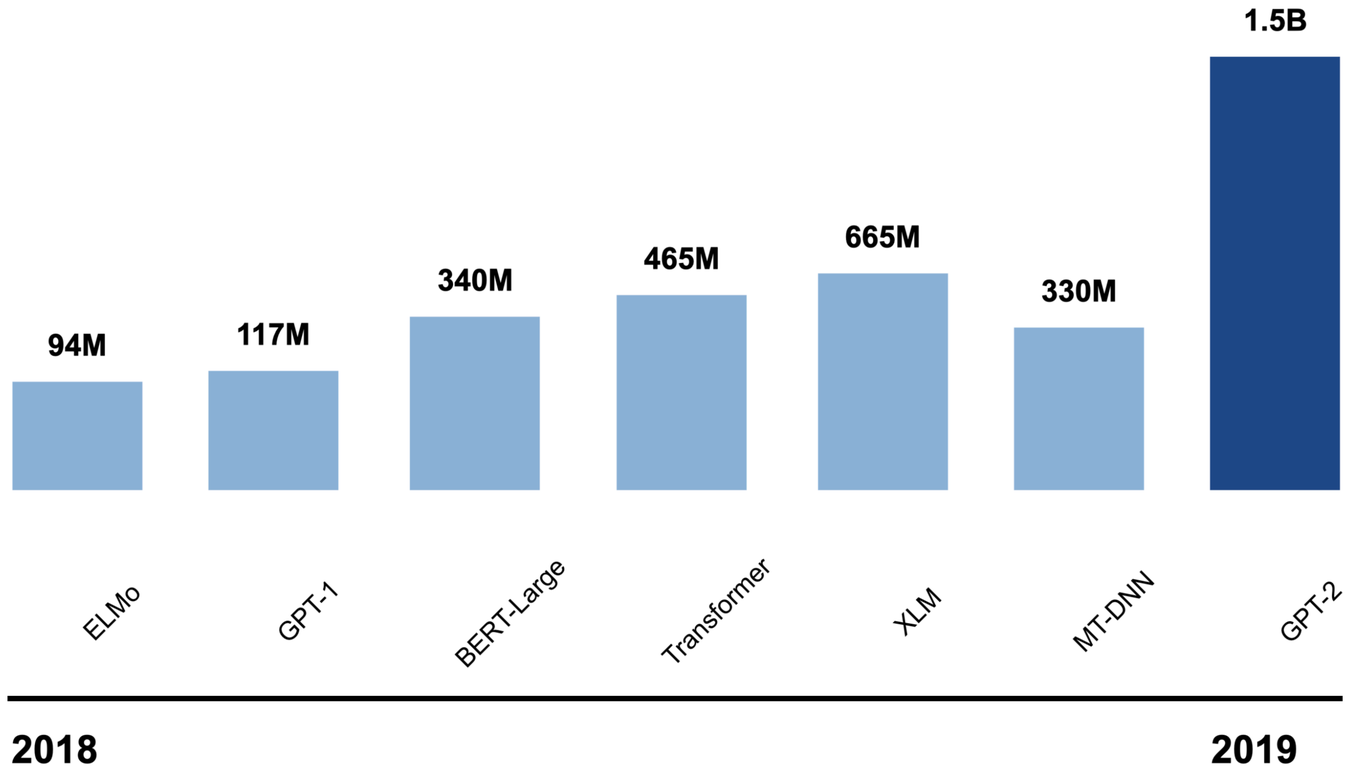
GPT-2's 1.5B parameters in 2019 marked a dramatic **2-3x leap in model scale** compared to previous architectures, signaling the start of the large language model era.

How much data do we need?

How big should our model be?

How much compute should we use?

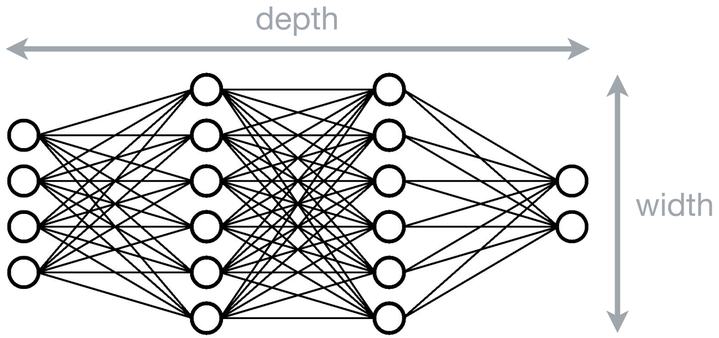
… in a principled way.

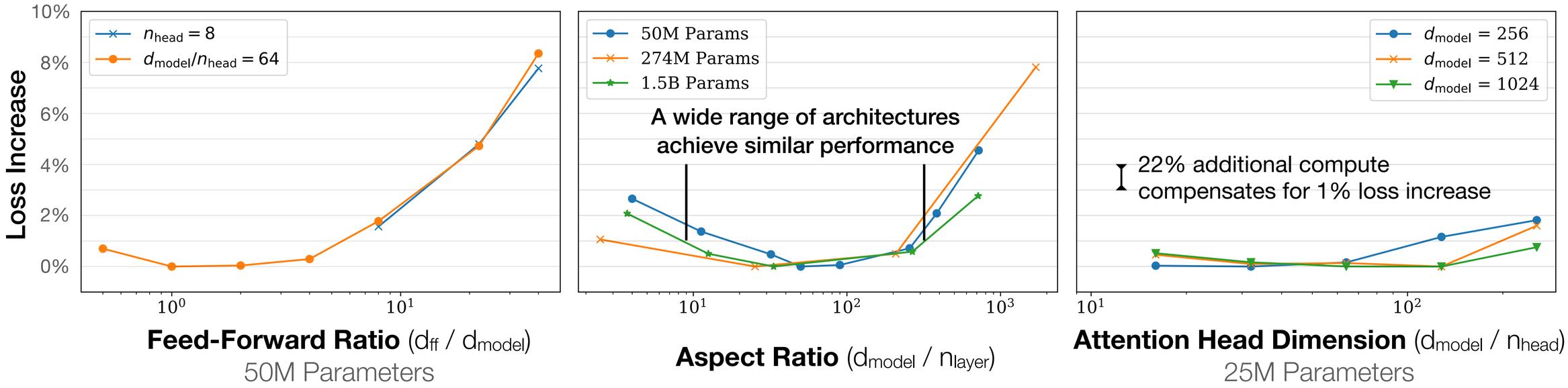


Hyperparameter Tuning vs. Scaling

Classic approach: Hyperparameter tuning on big models.

Wouldn't it be great to tune hyperparameters on small models and extrapolate to large ones?





*Excluding embedding parameters such as token or positional embeddings.

Sample Complexity and Rates

Learning Theory Fundamentals: Generalization Bounds

True Error ≤ Training Error + Complexity

$$\mathbb{E}[\mathcal{L}(f)] \leq \hat{\mathcal{L}}(f) + O\left(\sqrt{\frac{d}{n}}\right)$$

VC dimension d, i.e., model capacity

loss of model f

number of training samples

VC Dimension (d):

- Measures model capacity
- Linear: $d = n_{\text{params}} + 1$
- Neural nets: $O(W \log W)$

Rademacher Complexity:

- Expected correlation with random labels
- Data-dependent measure

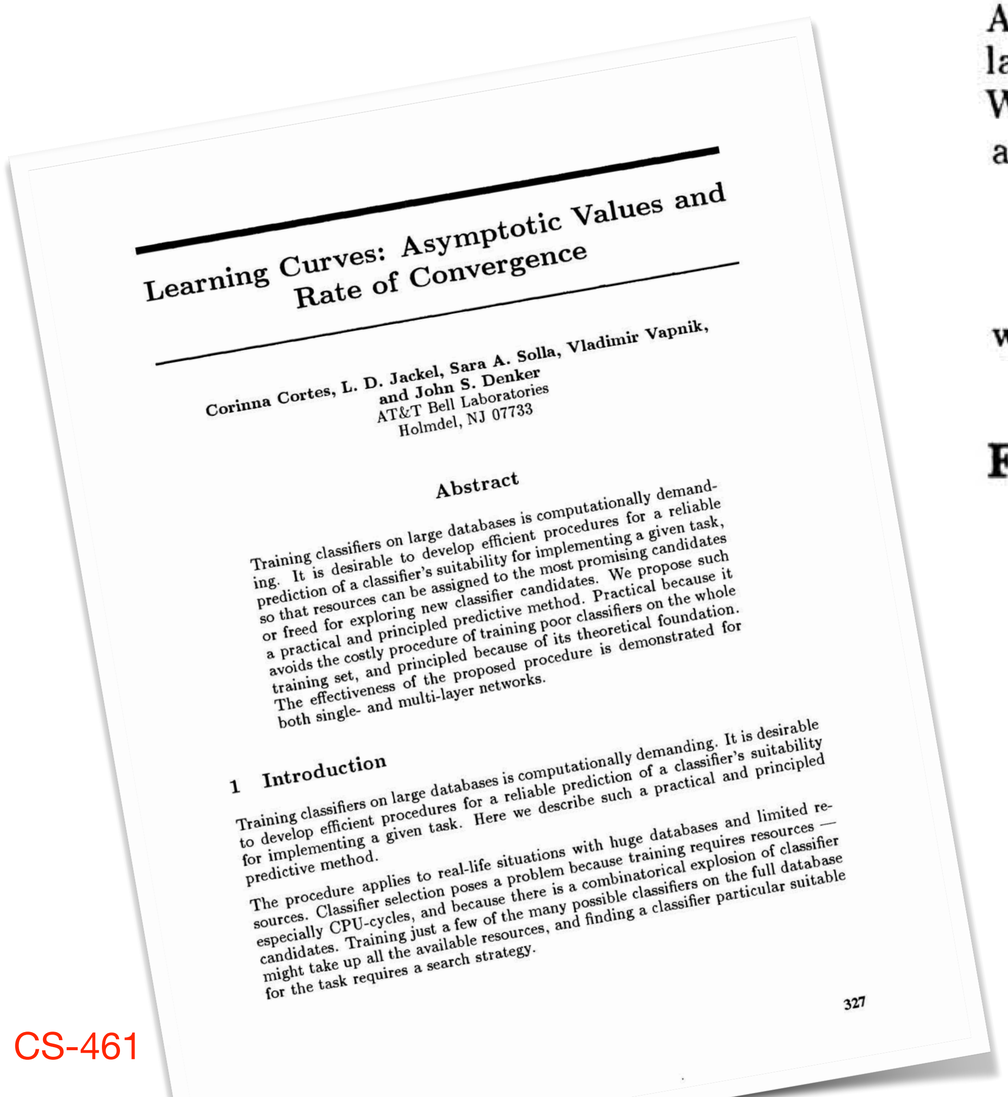
Key Insights: More data → Less overfitting

But fixed model → Performance plateau

Earliest Papers on Scaling Laws

1993

Theory meets reality!



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

Abstract

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 a practical and principled predictive method. Practical because it avoids the costly procedure of training poor classifiers on the whole training set, and principled because of its theoretical foundation. The effectiveness of the proposed procedure is demonstrated for both single- and multi-layer networks.

1 Introduction

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. Here we describe such a practical and principled predictive method.

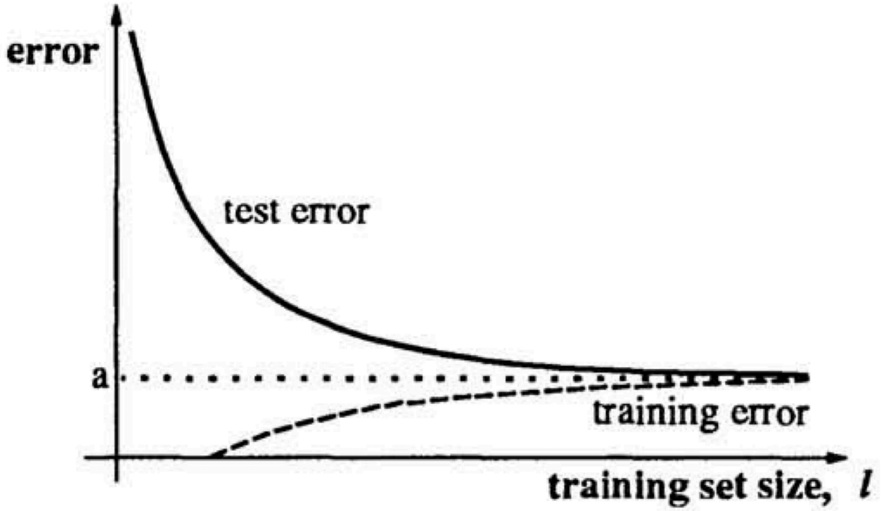
The procedure applies to real-life situations with huge databases and limited resources. Classifier selection poses a problem because training requires resources — especially CPU-cycles, and because there is a combinatorical explosion of classifier candidates. Training just a few of the many possible classifiers on the full database might take up all the available resources, and finding a classifier particular suitable for the task requires a search strategy.

A typical example of learning curves is shown in Fig. 2. The test error is always larger than the training error, but asymptotically they reach a common value, a. We model the errors for large sizes of the training set as power-law decays to the asymptotic error value, a:

$$\mathcal{E}_{\text{test}} = a + \frac{b}{l^{\alpha}} \qquad \text{and} \qquad \mathcal{E}_{\text{train}} = a - \frac{c}{l^{\beta}}$$

where l is the size of the training set, and α and β are positive exponents.

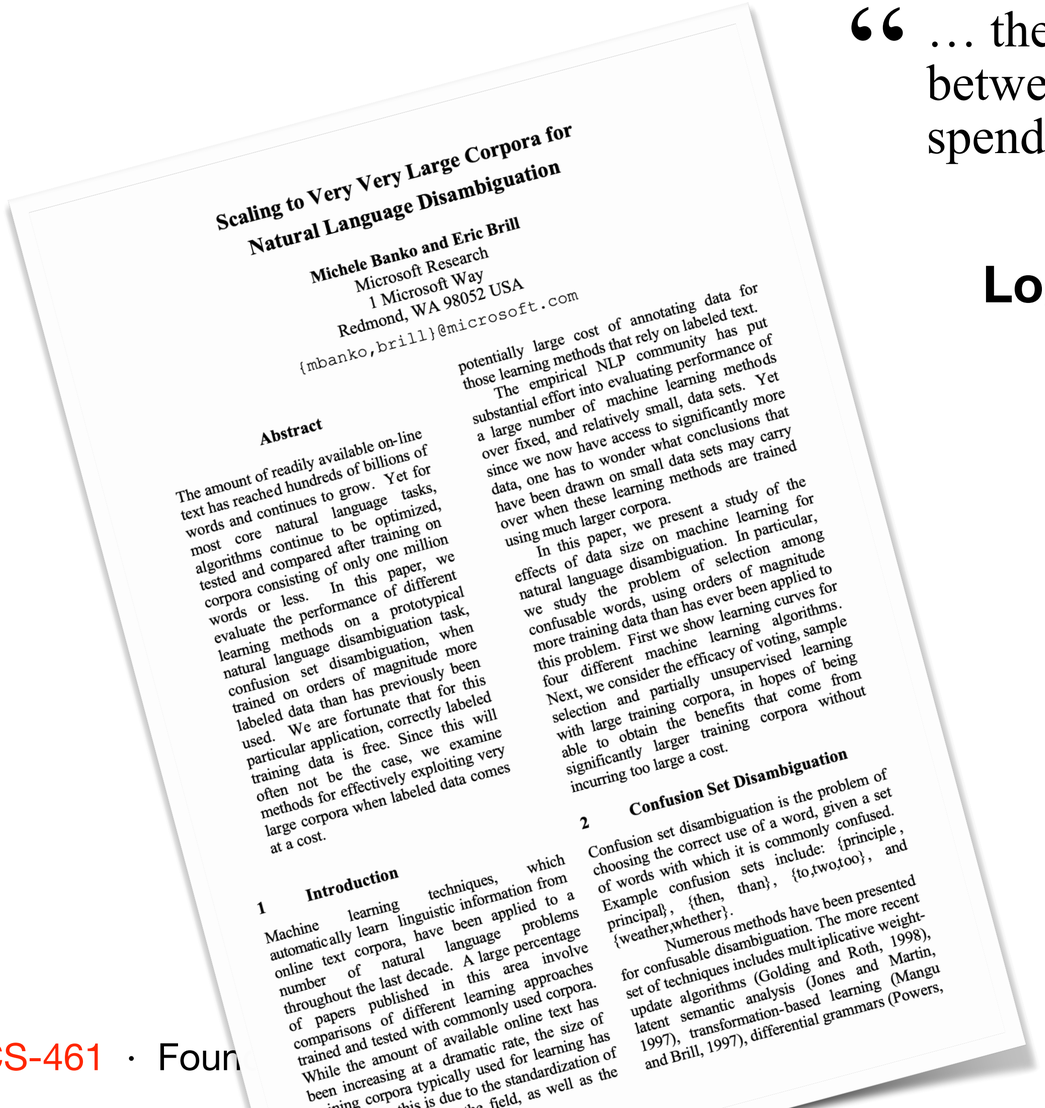
Figure 2



Earliest Papers on Scaling Laws

2001

Data scaling!



Scaling to Very Very Large Corpora for Natural Language Disambiguation

Michele Banko and Eric Brill
Microsoft Research
1 Microsoft Way
Redmond, WA 98052 USA
{mbanko,brill}@microsoft.com

Abstract

The amount of readily available on-line text has reached hundreds of billions of words and continues to grow. Yet for most core natural language tasks, algorithms continue to be optimized, tested and compared after training on corpora consisting of only one million words or less. In this paper, we evaluate the performance of different learning methods on a prototypical natural language disambiguation task, confusion set disambiguation, when trained on orders of magnitude more labeled data than has previously been used. We are fortunate that for this particular application, correctly labeled training data is free. Since this will often not be the case, we examine methods for effectively exploiting very large corpora when labeled data comes at a cost.

1 Introduction

Machine learning techniques, which automatically learn linguistic information from online text corpora, have been applied to a number of natural language problems throughout the last decade. A large percentage of papers published in this area involve comparisons of different learning approaches trained and tested with commonly used corpora. While the amount of available online text has been increasing at a dramatic rate, the size of corpora typically used for learning has remained relatively small. This is due to the standardization of the field, as well as the potentially large cost of annotating data for those learning methods that rely on labeled text. The empirical NLP community has put substantial effort into evaluating performance of a large number of machine learning methods over fixed, and relatively small, data sets. Yet since we now have access to significantly more data, one has to wonder what conclusions that have been drawn on small data sets may carry over when these learning methods are trained using much larger corpora.

In this paper, we present a study of the effects of data size on machine learning for natural language disambiguation. In particular, we study the problem of selection among confusable words, using orders of magnitude more training data than has ever been applied to this problem. First we show learning curves for four different machine learning algorithms. Next, we consider the efficacy of voting, sample selection and partially unsupervised learning with large training corpora, in hopes of being able to obtain the benefits that come from significantly larger training corpora without incurring too large a cost.

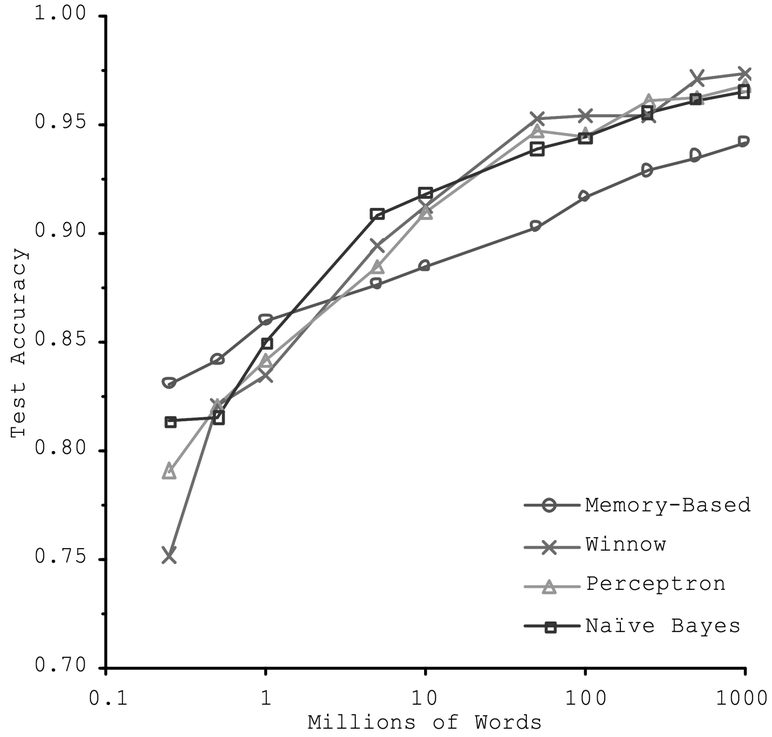
2 Confusion Set Disambiguation

Confusion set disambiguation is the problem of choosing the correct use of a word, given a set of words with which it is commonly confused. Example confusion sets include: {principle, principal}, {then, than}, {to,two,too}, and {weather,whether}.

Numerous methods have been presented for confusable disambiguation. The more recent set of techniques includes multiplicative weight-update algorithms (Golding and Roth, 1998), latent semantic analysis (Jones and Martin, 1997), transformation-based learning (Mangu and Brill, 1997), differential grammars (Powers,

> … these results suggest that we may want to reconsider the trade-off between spending time and money on algorithm development versus spending it on corpus development/

Log-linear scaling with growing amount of data



Earliest Papers on Scaling Laws

2017

Earliest large-scale neural' scaling work. Very ahead of their time!

DEEP LEARNING SCALING IS PREDICTABLE, EMPIRICALLY

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, Yanqi Zhou

{joel,sharan,ardalaninewsha,gregdiamos,junheewoo,hassankianinejad, patwarymostofa,yangyang62,zhouyanqi}@baidu.com

Baidu Research

ABSTRACT

Deep learning (DL) creates impactful advances following a virtuous recipe: model architecture search, creating large training data sets, and scaling computation. It is widely believed that growing training sets and models should improve accuracy and result in better products. As DL application domains grow, we would like a deeper understanding of the relationships between training set size, computational scale, and model accuracy improvements to advance the state-of-the-art.

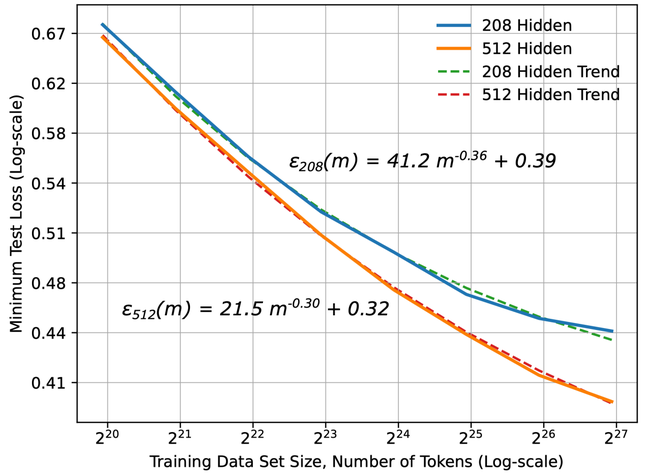
This paper presents a large scale empirical characterization of generalization error and model size growth as training sets grow. We introduce a methodology for this measurement and test four machine learning domains: machine translation, language modeling, image processing, and speech recognition. Our empirical results show power-law generalization error scaling across a breadth of factors, resulting in power-law exponents—the "steepness" of the learning curve—yet to be explained by theoretical work. Further, model improvements only shift the error but do *not* appear to affect the power-law exponent. We also show that model size scales sublinearly with data size. These scaling relationships have significant implications on deep learning research, practice, and systems. They can assist model debugging, setting accuracy targets, and decisions about data set growth. They can also guide computing system design and underscore the importance of continued computational scaling.

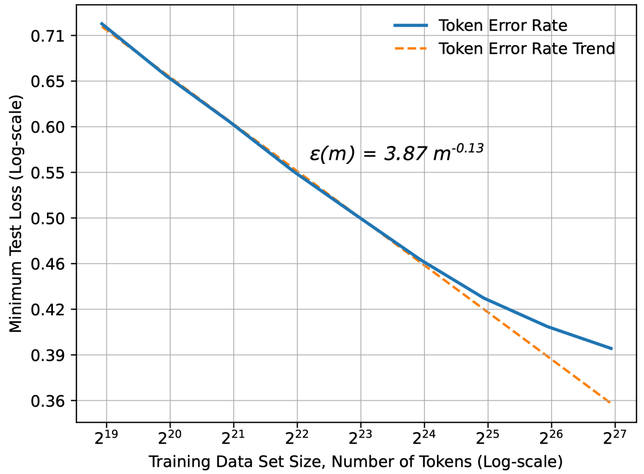
1 INTRODUCTION

The deep learning (DL) community has created impactful advances across diverse application domains by following a straightforward recipe: search for improved model architectures, create large training data sets, and scale computation. This recipe helps improve user experience and product adoption, which drives increased DL development investments in existing and emerging application domains. As data sets grow and new application domains proliferate, we would like to focus our development efforts through a deeper understanding of how the recipe parts coordinate to drive the most valuable product improvements.

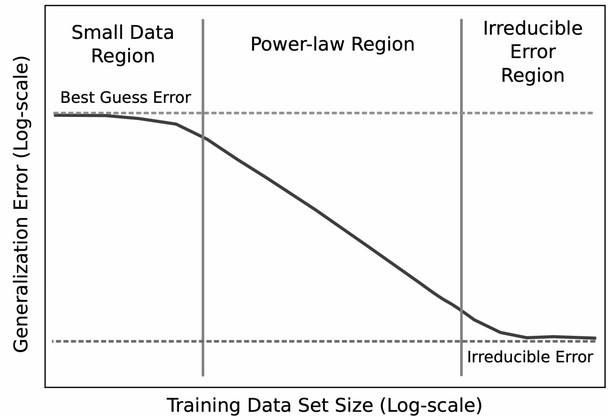
Breaking down the recipe, we note challenges in the search for improved model architectures. Model search can create important new insights and publications, which, in turn, improve products that use the novel models. However, model architecture advances often depend on unreliable epiphany; advances usually involve complex or creative reframing of the modeling problem, and research often involves large-scale hyperparameter search with some serendipity.

As a lower-risk complement to model architecture search, it is important that we investigate the other two recipe parts—creating large training sets and scaling computation—where we may have more control over progress. It is widely believed that simply using more data to train larger models should improve accuracy. We would like to better analyze the relationships between training set scale, computational scale, and model accuracy improvements. In particular, accurately predicting ...ization error scaling with training set size would provide a powerful tool for estimating the ...mpute requirements—for advancing state-of-the-art (SOTA).





different regions of model behavior



Earliest Papers on Scaling Laws

2017

Earliest large-scale neural' scaling work. Very ahead of their time!

DEEP LEARNING SCALING IS PREDICTABLE, EMPIRICALLY

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, Yanqi Zhou

{joel,sharan,ardalaninewsha,gregdiamos,junheewoo,hassankianinejad, patwarymostofa,yangyang62,zhouyanqi}@baidu.com

Baidu Research

ABSTRACT

Deep learning (DL) creates impactful advances following a virtuous recipe: model architecture search, creating large training data sets, and scaling computation. It is widely believed that growing training sets and models should improve accuracy and result in better products. As DL application domains grow, we would like a deeper understanding of the relationships between training set size, computational scale, and model accuracy improvements to advance the state-of-the-art.

This paper presents a large scale empirical characterization of generalization error and model size growth as training sets grow. We introduce a methodology for this measurement and test four machine learning domains: machine translation, language modeling, image processing, and speech recognition. Our empirical results show power-law generalization error scaling across a breadth of factors, resulting in power-law exponents—the "steepness" of the learning curve—yet to be explained by theoretical work. Further, model improvements only shift the error but do *not* appear to affect the power-law exponent. We also show that model size scales sublinearly with data size. These scaling relationships have significant implications on deep learning research, practice, and systems. They can assist model debugging, setting accuracy targets, and decisions about data set growth. They can also guide computing system design and underscore the importance of continued computational scaling.

1 INTRODUCTION

The deep learning (DL) community has created impactful advances across diverse application domains by following a straightforward recipe: search for improved model architectures, create large training data sets, and scale computation. This recipe helps improve user experience and product adoption, which drives increased DL development investments in existing and emerging application domains. As data sets grow and new application domains proliferate, we would like to focus our development efforts through a deeper understanding of how the recipe parts coordinate to drive the most valuable product improvements.

Breaking down the recipe, we note challenges in the search for improved model architectures. Model search can create important new insights and publications, which, in turn, improve products that use the novel models. However, model architecture advances often depend on unreliable epiphany; advances usually involve complex or creative reframing of the modeling problem, and research often involves large-scale hyperparameter search with some serendipity.

As a lower-risk complement to model architecture search, it is important that we investigate the other two recipe parts—creating large training sets and scaling computation—where we may have more control over progress. It is widely believed that simply using more data to train larger models should improve accuracy. We would like to better analyze the relationships between training set scale, computational scale, and model accuracy improvements. In particular, accurately predicting [illegible] error scaling with training set size would provide a powerful tool for estimating the [illegible] requirements—for advancing state-of-the-art (SOTA).

> … it can be difficult to ensure that training data is large enough to see the power-law learning curve region. We have found that models … where accuracy is only as good as best guessing, but the model trains on enough data to be in the power-law region.

> … model size curves may offer a way to project the compute requirements to reach a particular accuracy level. The compute requirements could inform decisions about how to scale computational capacity to unlock these compute-limited applications.

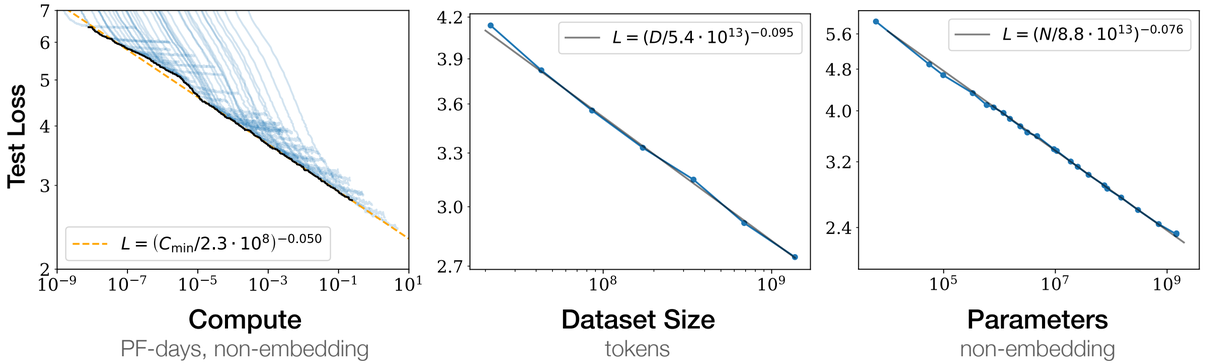
→ foreshadowed the idea of **scaling by compute**.

Scaling Laws for Neural Networks

Neural scaling laws establish a mathematical relationship between the following variables in training a model.

Performance has a **power-law relationship** with each of the three scale factors N, D, C when not bottlenecked by the other two.

- $\mathcal{L}$ prediction loss
- N number of model parameters
- D size of the dataset used to train the model
- C total compute used to train the model



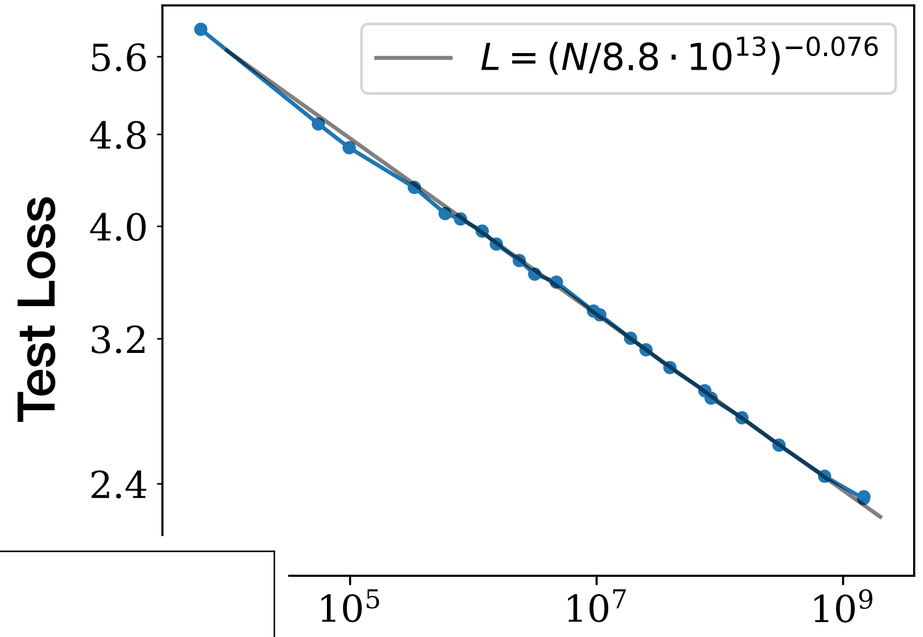
Kaplan et al., (2020)

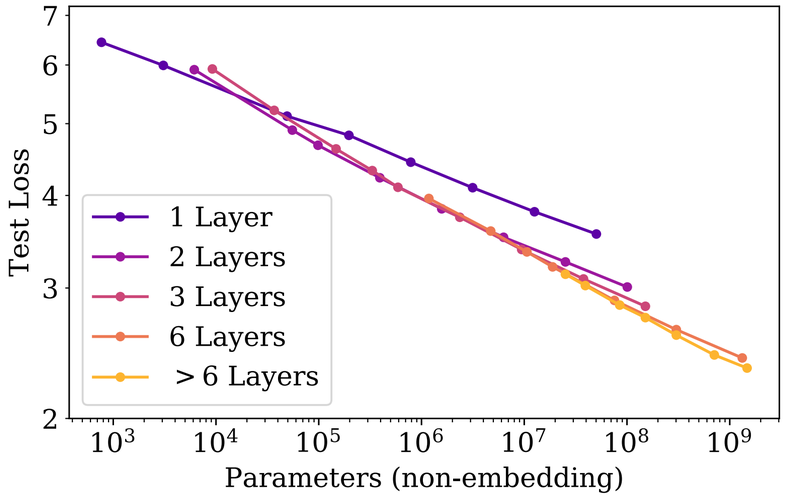
Bottleneck 1: **Number of Parameters**

1. Fix the number of parameters N. Dataset size D and amount of compute C are not considered a bottleneck.
2. Train model until convergence.
3. Plot test loss $\mathcal{L}$ as function of the number of parameters N.

$$\longrightarrow \quad \mathcal{L} \approx \left(\frac{N_c}{N}\right)^{\alpha_N}$$

… specifying the number of parameters does not uniquely specify model architecure but **model size matters more than model shape**

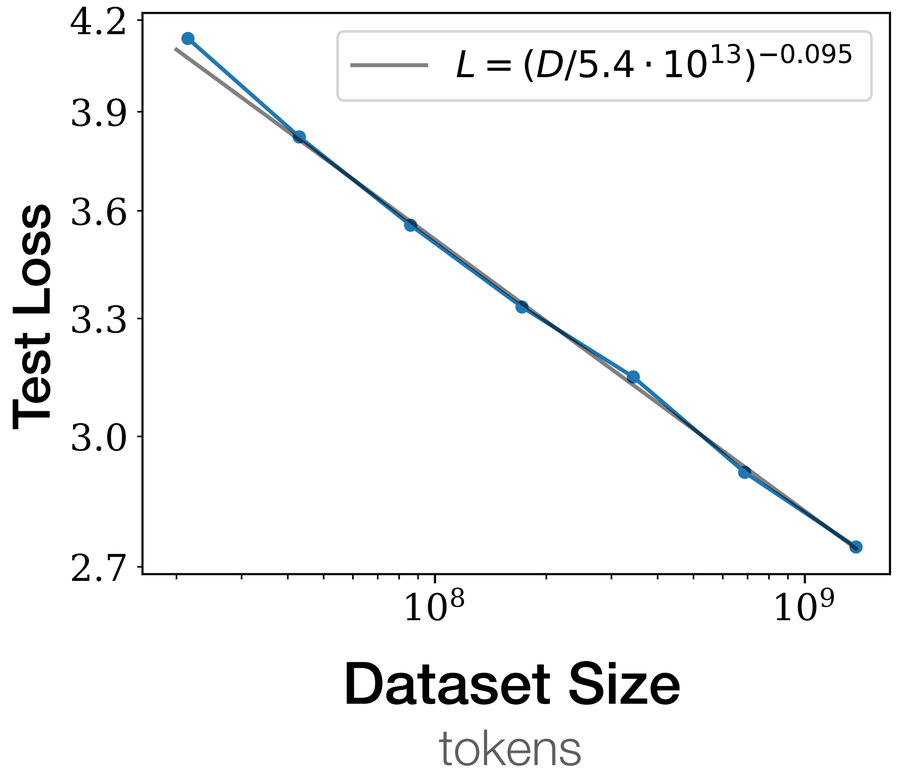




Bottleneck 2: **Dataset Size**

1. Fix the number of training examples D. Number of parameters N and amount of compute C are not considered a bottleneck.
2. Train a model until test loss stops decreasing.
3. Plot test loss $\mathcal{L}$ as function of the number of training examples D.

$$\longrightarrow \quad \mathcal{L} \approx \left(\frac{D_c}{D}\right)^{\alpha_D}$$



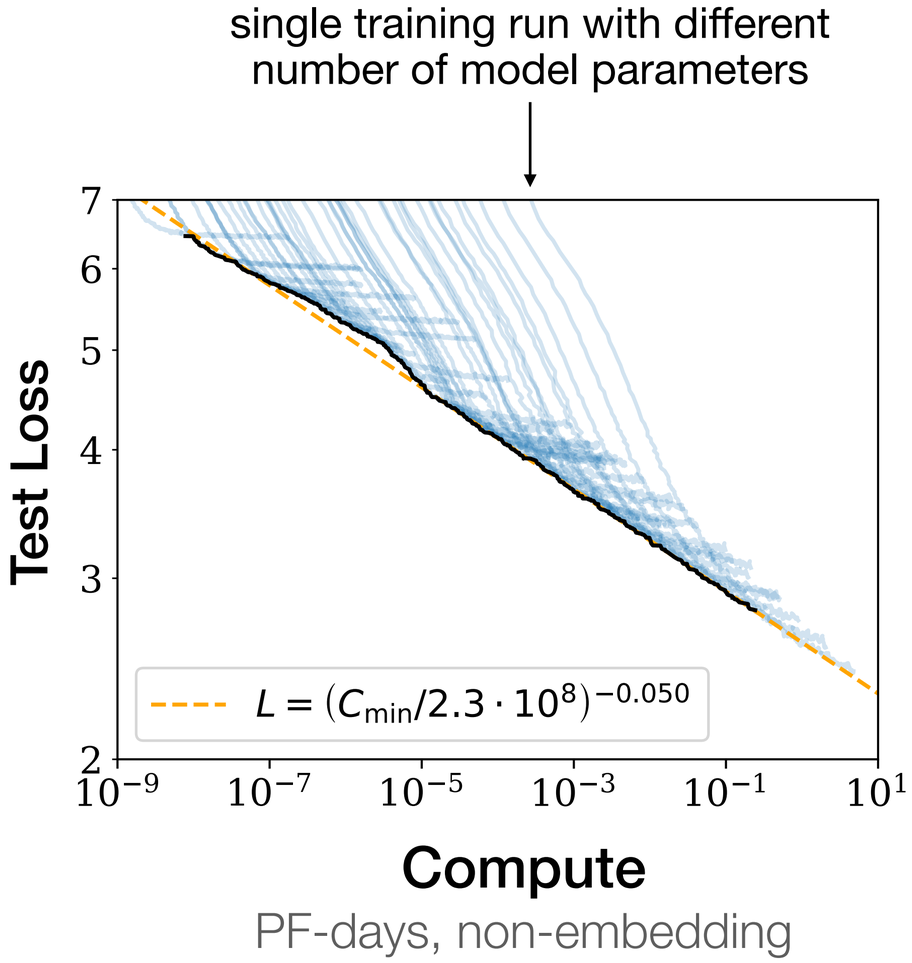
Bottleneck 3: **Amount of Compute**

1. Train models of various sizes N with a very large dataset of size D.
2. For each model, plot the test loss as function of the compute used.
3. Plot minimum test loss $\mathcal{L}$ as function of the amount of compute C used.

$$\longrightarrow \quad \mathcal{L} \approx \left(\frac{C_c}{C}\right)^{\alpha_C}$$

Measuring Compute (in a Transformer)

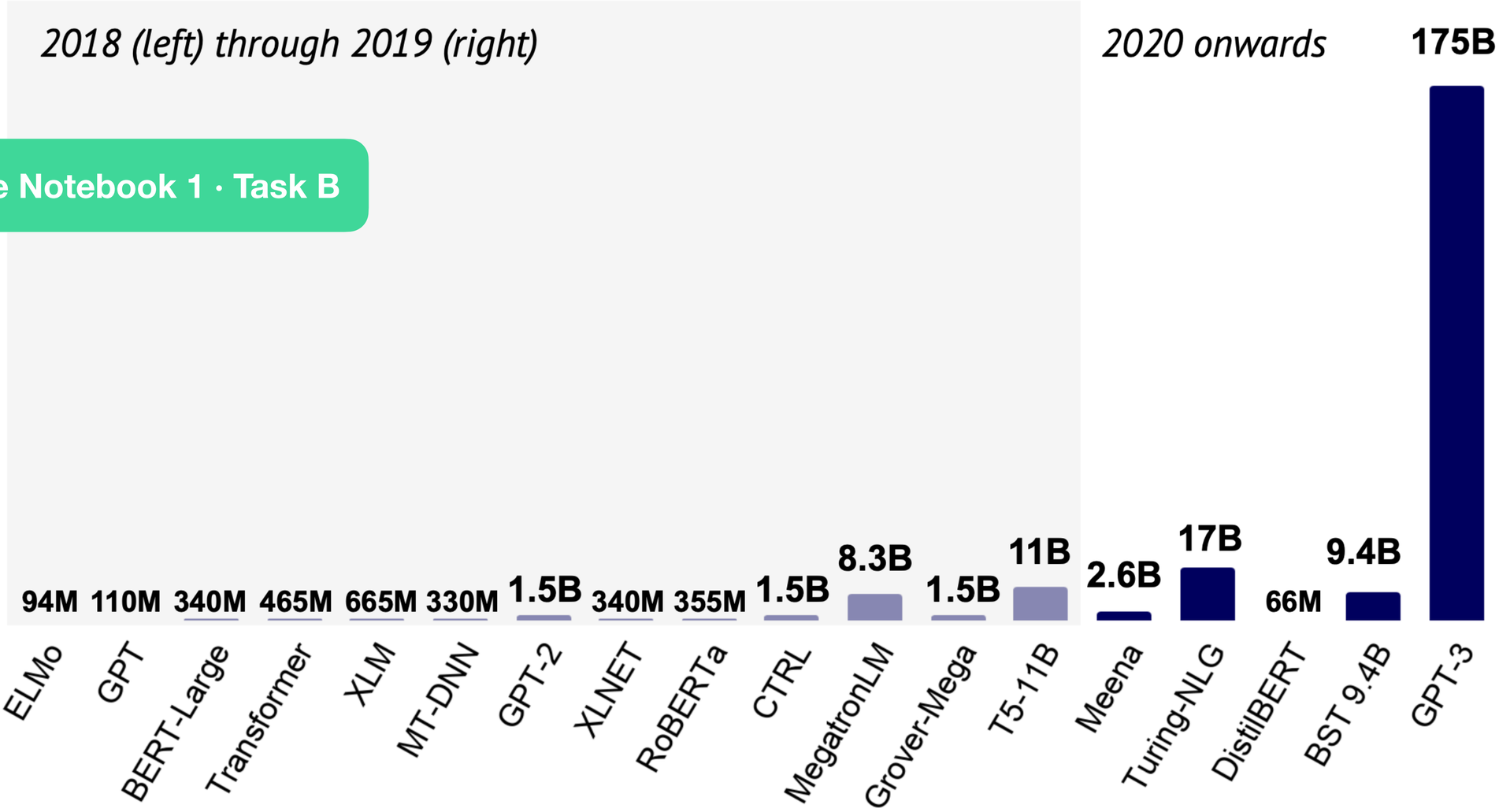
- Forward compute $\approx 2ND$
- Backward compute $\approx 4ND$



Language Model Sizes

State of AI, ‘Report 2020’.
Published online at ‘https://www.stateof.ai/2020'.

Huge models and massive training costs dominate the hottest area of AI.



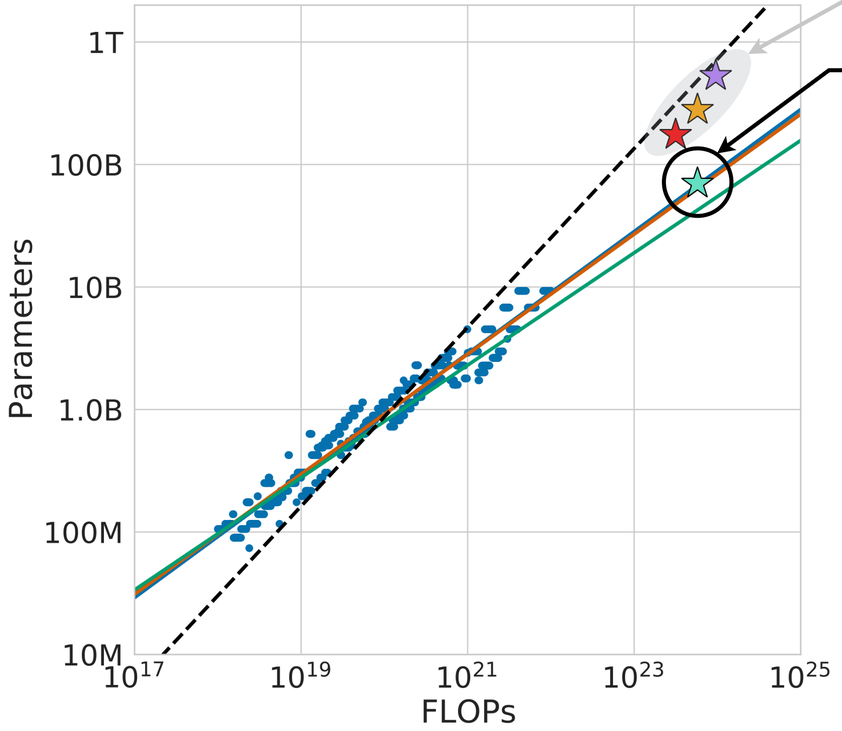
Code Notebook 1 · Task B

Caution: **Model Size Is Not Everything**

Researchers noticed giant models seemed undertrained …

… Kaplan et al., (2020) had fixed one entity and varied the others.

But what if we systematically vary *both* model size and dataset size?



Models were far too big for their data!

Chinchilla: 70B parameters, 4x more data than GPT-3, outperformed GPT-3 using same compute

Approach 1
Approach 2
Approach 3
Kaplan et al (2020) → those scaling prediction proved quite different from empirical estimates

$$N \propto C^{0.73}, D \propto C^{0.27}$$

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

Hoffmann et al., (2022)

Estimating Optimal Parameter/Training Tokens Allocation

Approach 1: Fix model sizes, vary training tokens

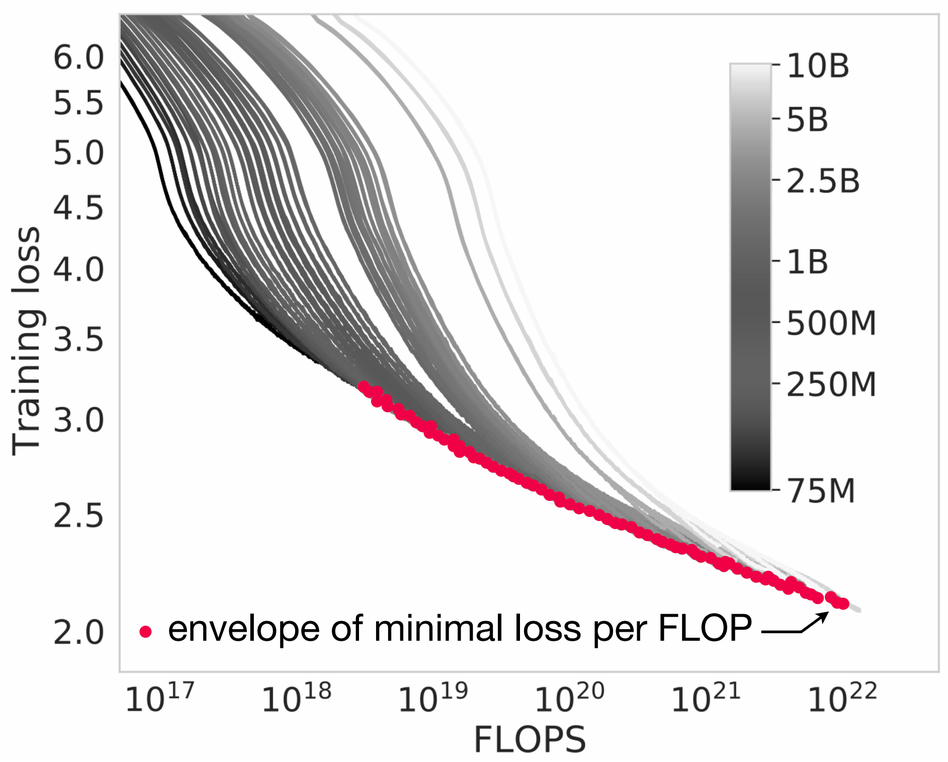
- Trained models (70M to 10B) for different durations
- Found training duration that minimized loss per FLOP
- Extracted the "envelope" of best performance at each compute
- **Finding**: $N \propto C^{0.50}, D \propto C^{0.50}$

Approach 2: Fixed compute budgets

- Fixed 9 compute budgets, varied model size at each
- Found the “valley": optimal model size that minimizes loss
- Clear parabolas showing sweet spot between "too small/large"
- **Finding**: $N \propto C^{0.49}, D \propto C^{0.51}$

Approach 3: Parametric fitting

- Fit equation L(N,D) to >400 training runs
- Minimized loss equation under compute constraints
- Predicted optimal allocation mathematically
- **Finding**: $N \propto C^{0.46}, D \propto C^{0.54}$



→ use envelope points to estimate the optimal model size

Estimating Optimal Parameter/Training Tokens Allocation

Approach 1: Fix model sizes, vary training tokens

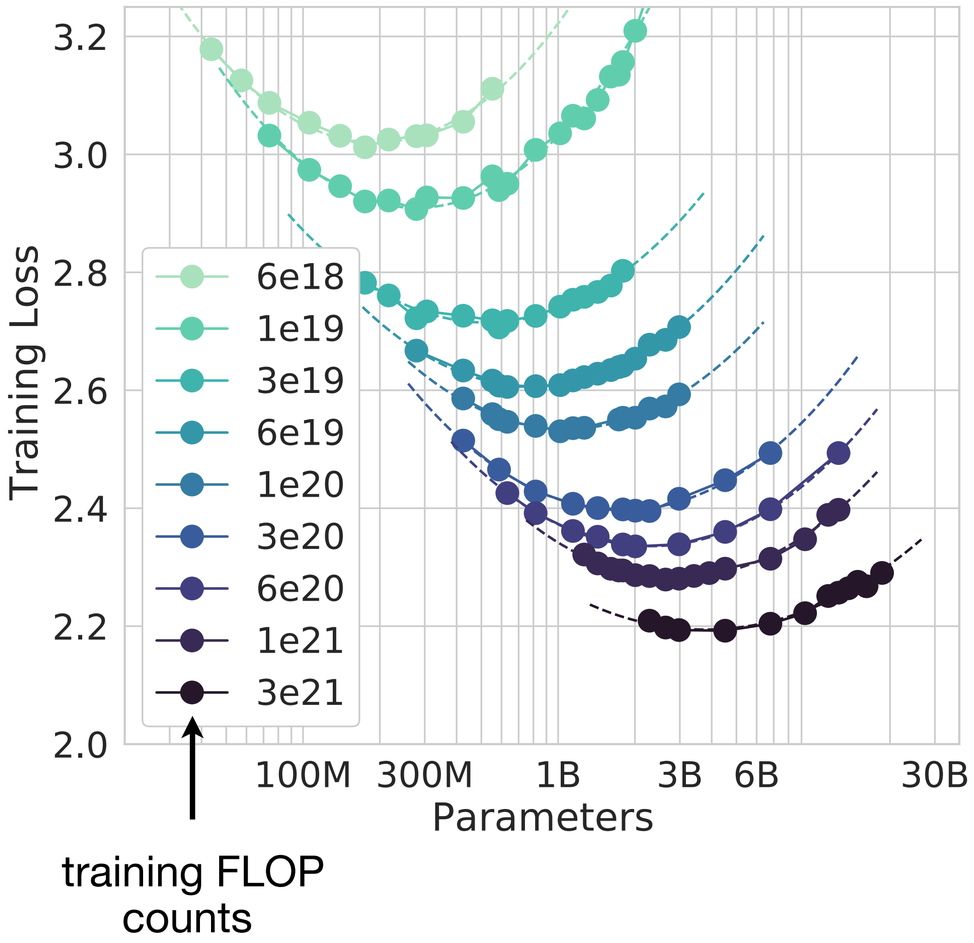
- Trained models (70M to 10B) for different durations
- Found training duration that minimized loss per FLOP
- Extracted the "envelope" of best performance at each compute
- **Finding**: $N \propto C^{0.50}, D \propto C^{0.50}$

Approach 2: Fixed compute budgets

- Fixed 9 compute budgets, varied model size at each
- Found the “valley": optimal model size that minimizes loss
- Clear parabolas showing sweet spot between "too small/large"
- **Finding**: $N \propto C^{0.49}, D \propto C^{0.51}$

Approach 3: Parametric fitting

- Fit equation L(N,D) to >400 training runs
- Minimized loss equation under compute constraints
- Predicted optimal allocation mathematically
- **Finding**: $N \propto C^{0.46}, D \propto C^{0.54}$



Estimating Optimal Parameter/Training Tokens Allocation

Approach 1: Fix model sizes, vary training tokens

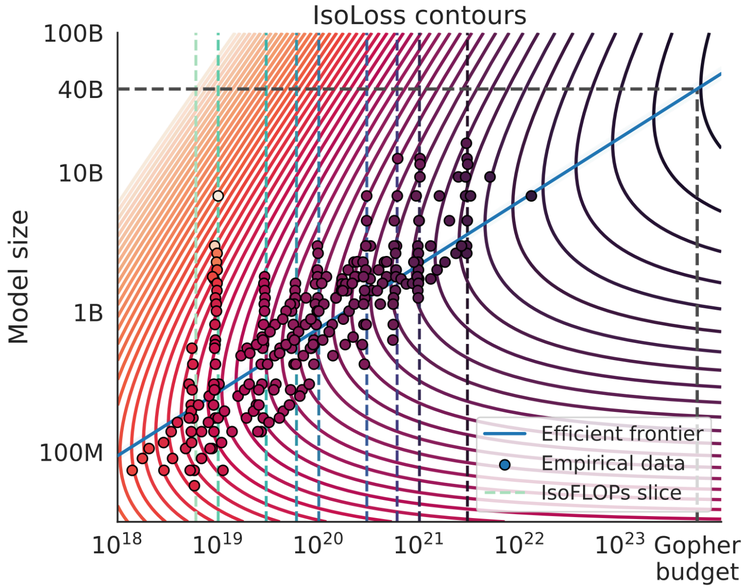
- Trained models (70M to 10B) for different durations
- Found training duration that minimized loss per FLOP
- Extracted the envelope of best performance at each compute
- **Finding**: $N \propto C^{0.50}, D \propto C^{0.50}$

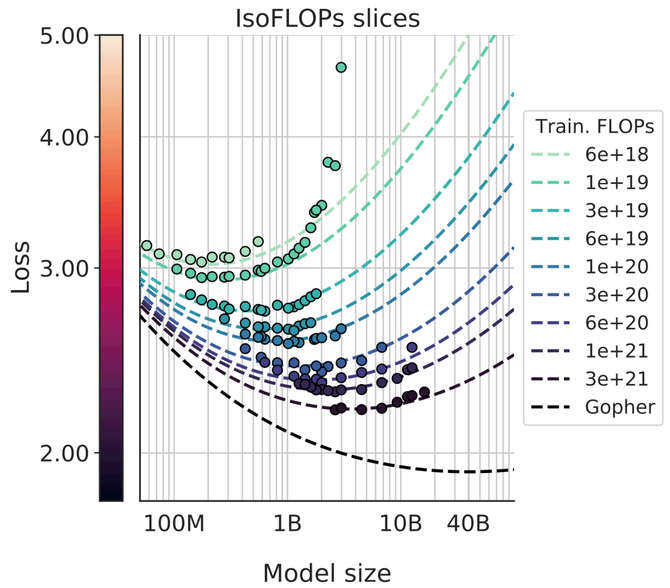
Approach 2: Fixed compute budgets

- Fixed 9 compute budgets, varied model size at each
- Found the “valley": optimal model size that minimizes loss
- Clear parabolas showing sweet spot between "too small/large"
- **Finding**: $N \propto C^{0.49}, D \propto C^{0.51}$

Approach 3: Parametric fitting

- Fit equation L(N,D) to >400 training runs
- Minimized loss equation under compute constraints
- Predicted optimal allocation mathematically
- **Finding**: $N \propto C^{0.46}, D \propto C^{0.54}$

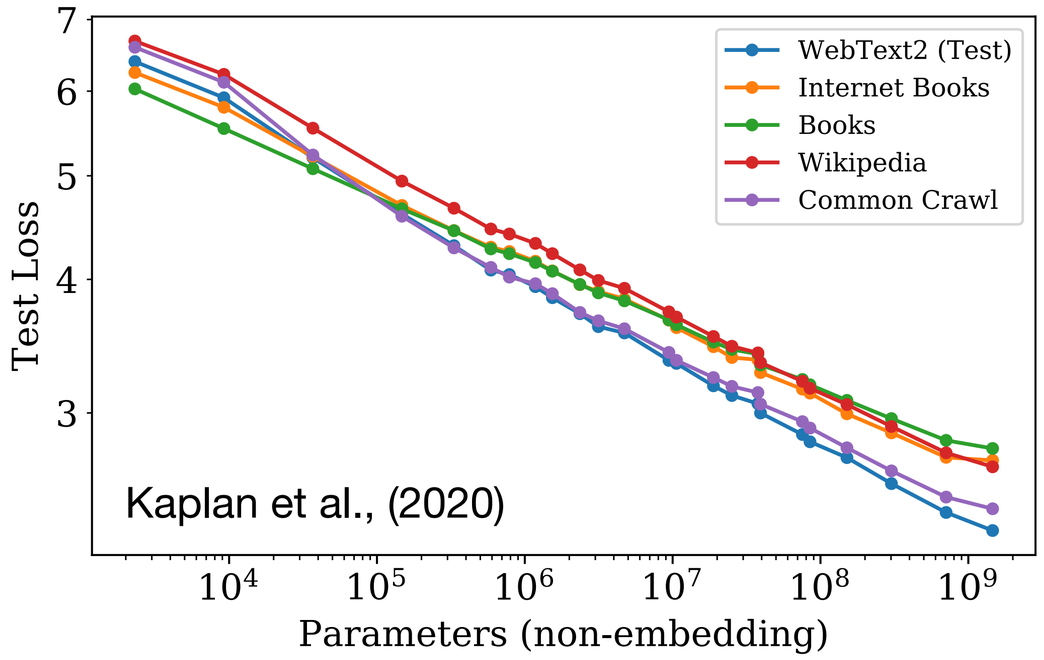




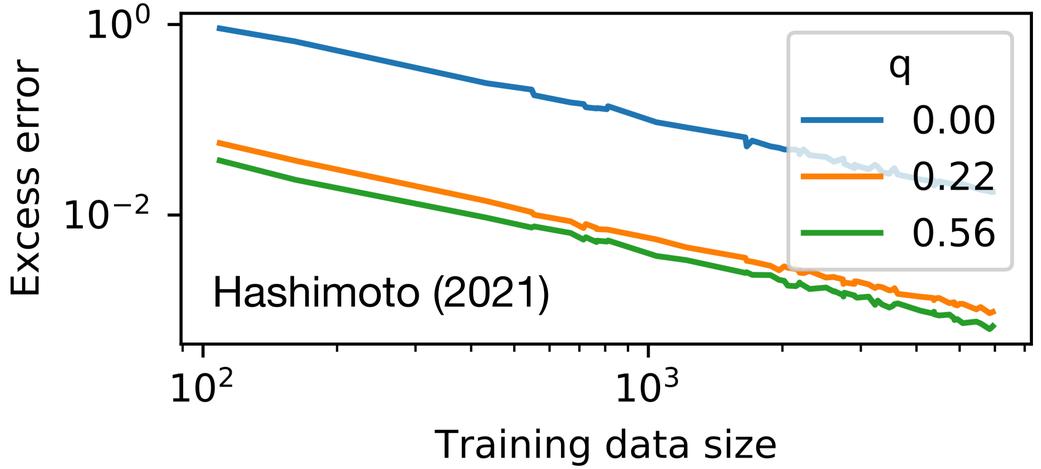
It's Also About Data

Scaling Laws under Distribution Shifts

- Different datasets follow the same power law slope
- Only the vertical offset changes between datasets
- Models can transfer across domains with performance penalties



- Data mixture composition affects performance non-linearly
- Optimal performance often requires diverse data sources
- As q changes from pure ($q = 0$) to mixed datasets ($q > 0$), performance improves

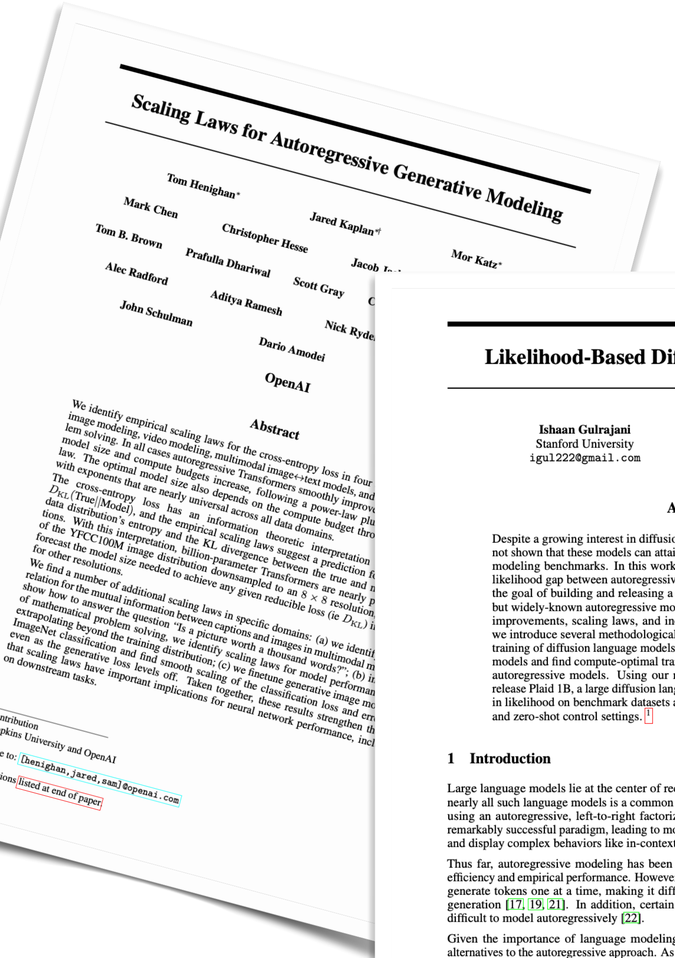


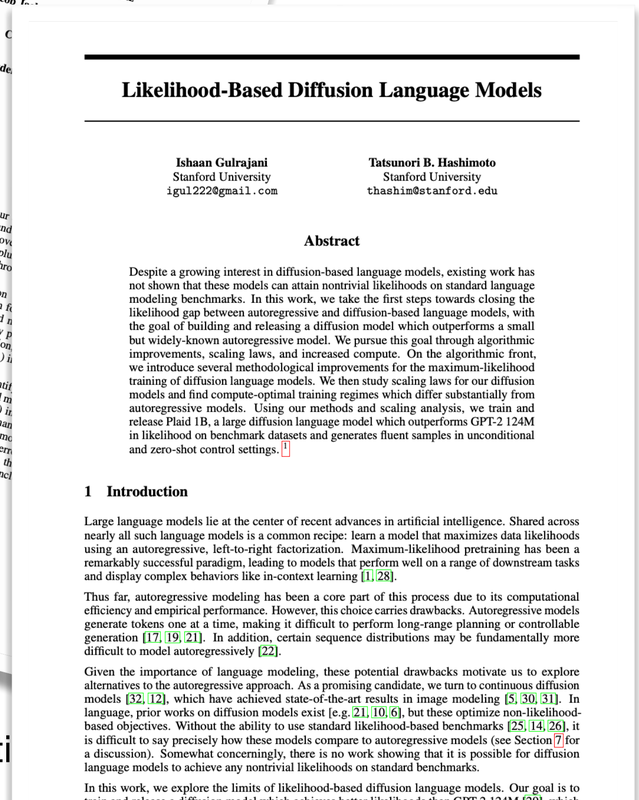
Similar Patterns for Beyond Language

… for generative modeling

… for vision transformers

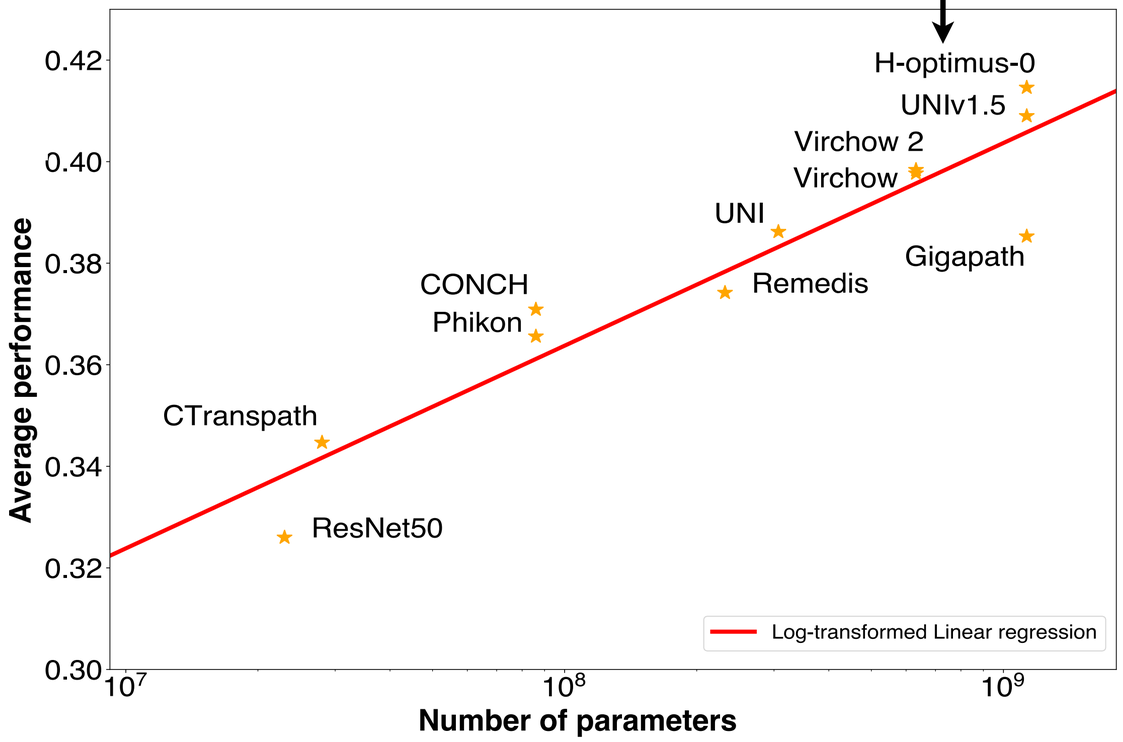
… for molecular representation learning

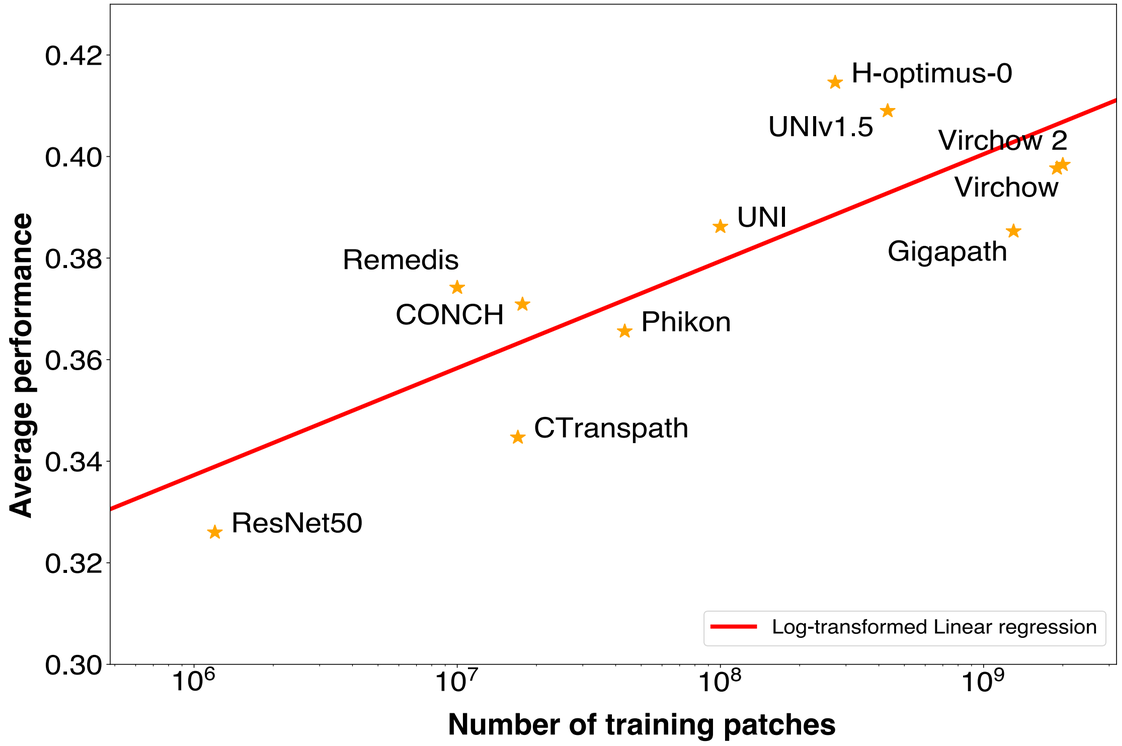




Scaling Laws in Histopathology Foundation Models

Lecture 9: Guest Lecture





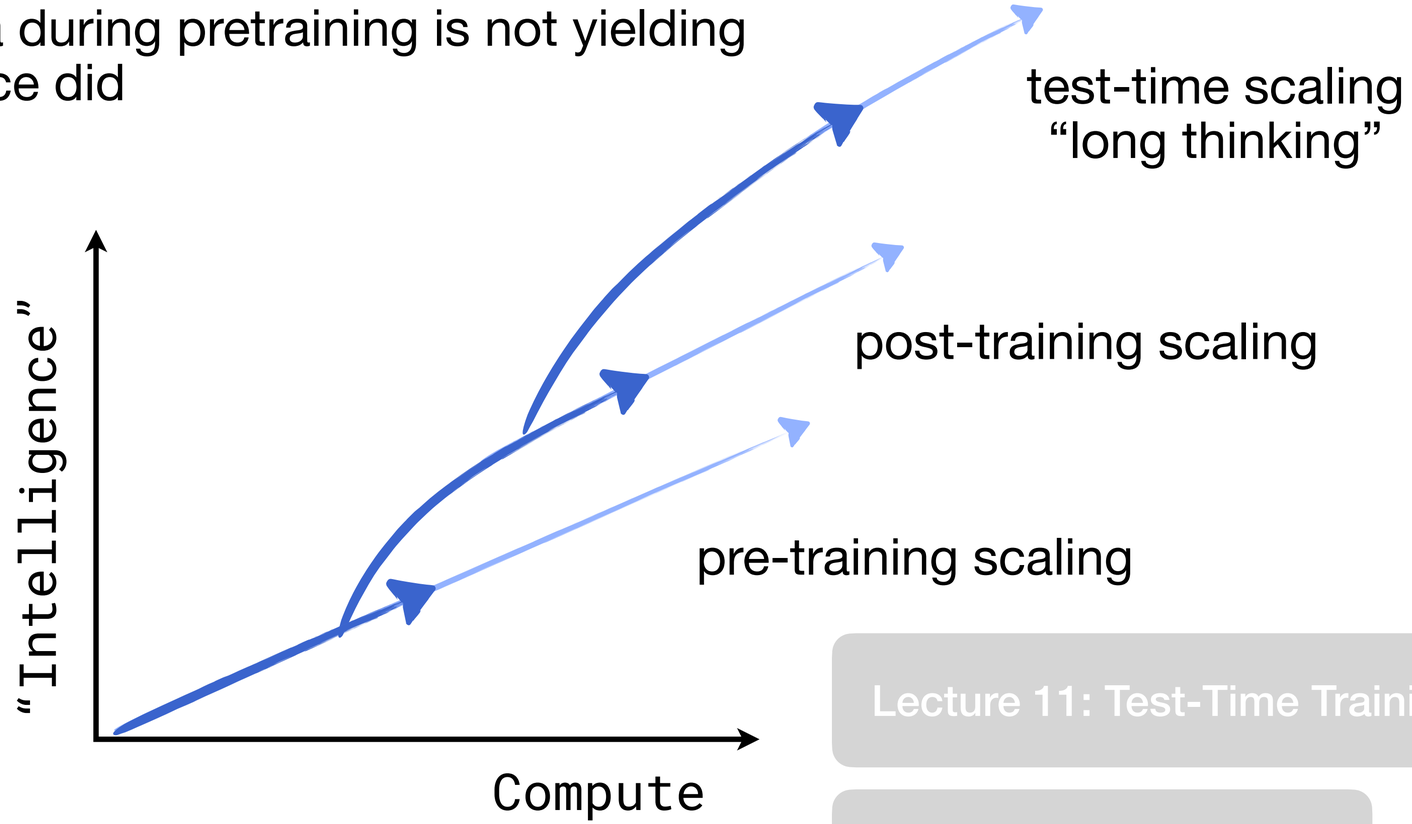
e.g., H-optimus-0 is trained on **1 million+ whole-slide images** from **800,000+ patients** with over **1 billion parameters**

Scale is not Intelligence: The end of scaling laws?

AI scaling laws are showing signs of diminishing returns

- Leading AI labs are seeing models improve more slowly than before
- Simply adding more GPUs and data during pretraining is not yielding the exponential improvements it once did

From One to Three Scaling Laws

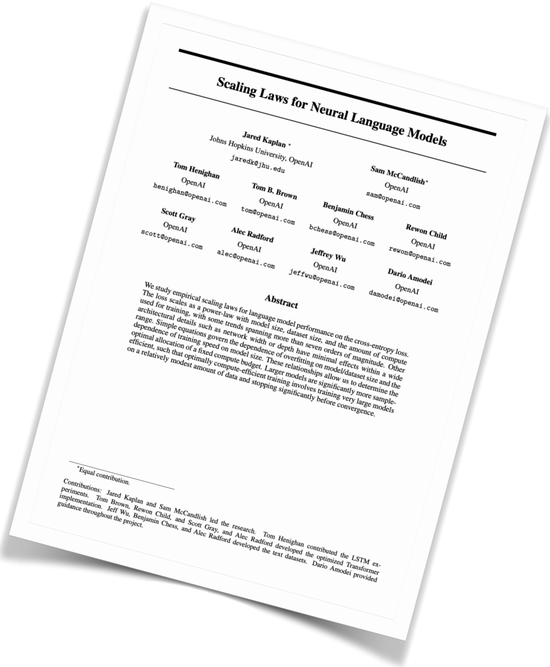


Lecture 11: Test-Time Training

Lecture 13: Reasoning

This Week's Papers

Papers are linked in Moodle.



Kaplan et al., "Scaling Laws for Neural Language Models." arXiv preprint arXiv:2001.08361 (2020).

Hoffmann et al., "Training Compute-Optimal Large Language Models." arXiv preprint arXiv:2203.15556 (2022).

… in addition to papers mentioned on exercise sheet.

This Week's Exercise Sheet

Exercise 1 · Task 1

InfoNCE in Contrastive Learning: Explores the InfoNCE loss from a probabilistic perspective, connecting it to cross-entropy loss and NCE, while analyzing how the number of negative samples K affects mutual information learning.

Exercise 1 · Task 2

Masked Language Modeling as Pseudo-Likelihood: Establishes the mathematical relationship between MLM training objectives and pseudo-likelihood, including connections to pseudo-perplexity and comparisons with autoregressive models.

This Week's Code Demonstration

Code Notebook 1 · Task A

Exploring Contrastive Learning with SimCLR: Hands-on implementation and exploration of the SimCLR framework for self-supervised visual representation learning.
→ Jupyter notebook exercise

Code Notebook 1 · Task B

Exploring Scaling Behaviour of LMs with Pythia Models: Empirical investigation of how language model performance scales with model size using the Pythia model series.
→ Jupyter notebook exercise

CS-461

Foundation Models and Generative AI

Have a great week!