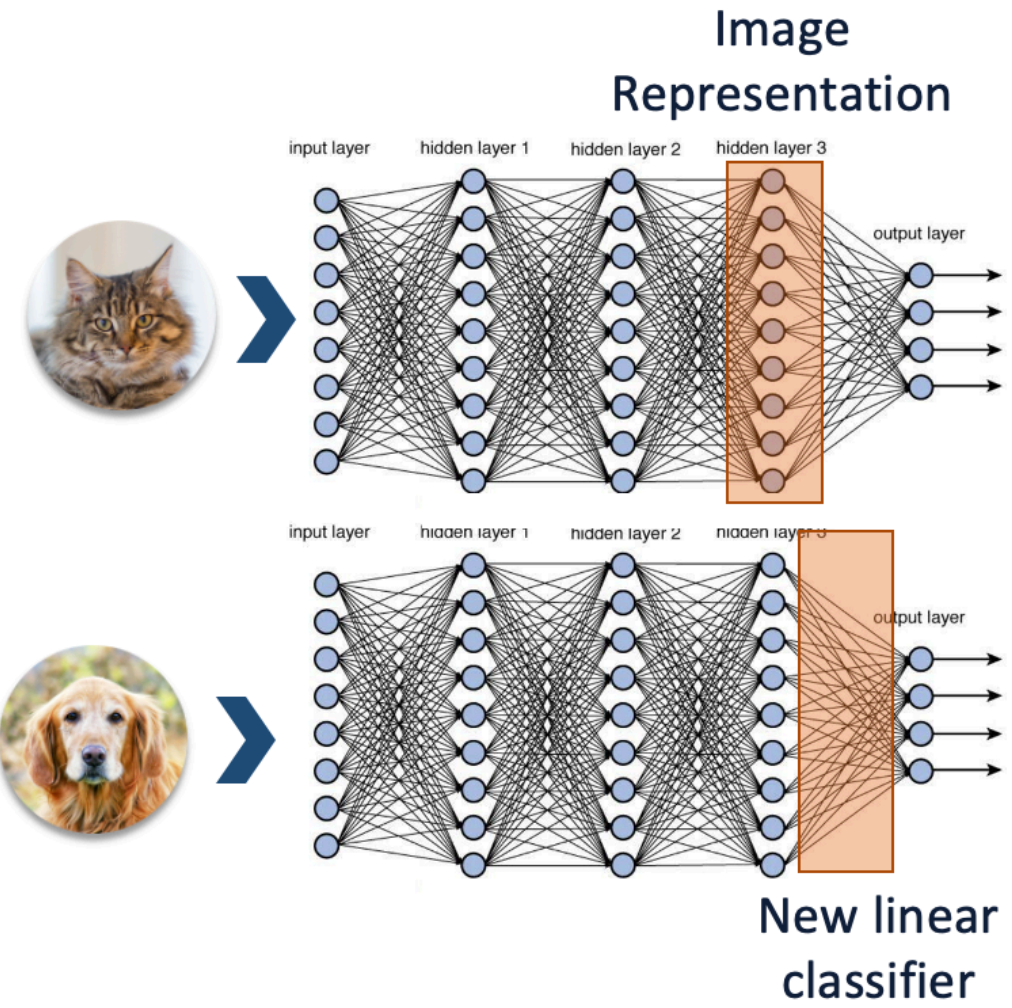
$\{(x_i, y_i)\} \overset{i.i.d.}{\sim} D_{target}$

1) train on general large data
2) train on target dist

Representation Learning
Pre-training

Example in image representation



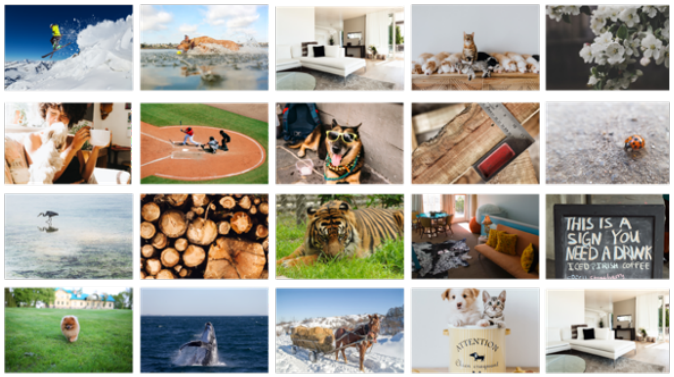
Train a neural network (ResNet) on ImageNet (1M data, 1000 classes)

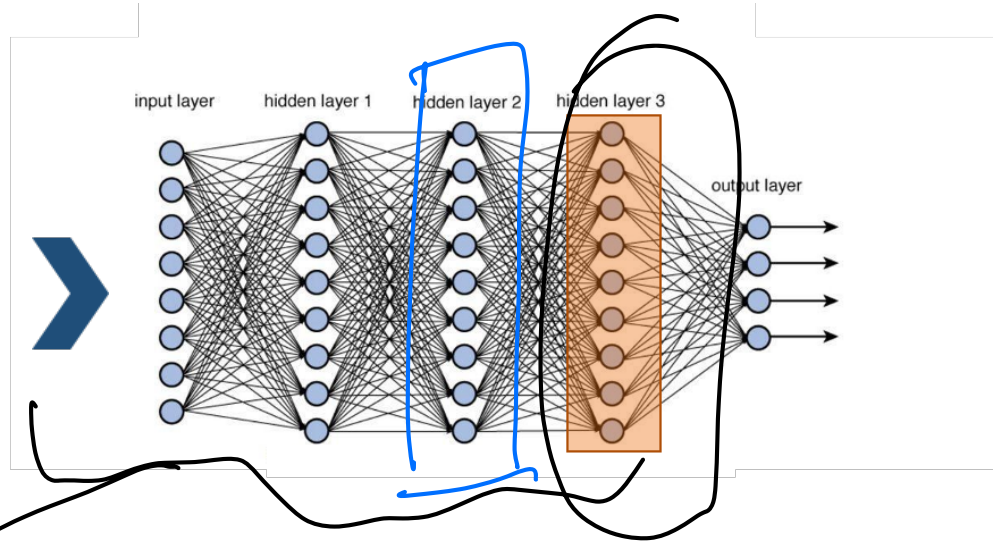
Representation (feature extractor): The mapping from image to the second-to-the-last layer.

Fix the representation, just re-train the last linear layer.

Example in image representation

Source tasks (for training representation): ImageNet





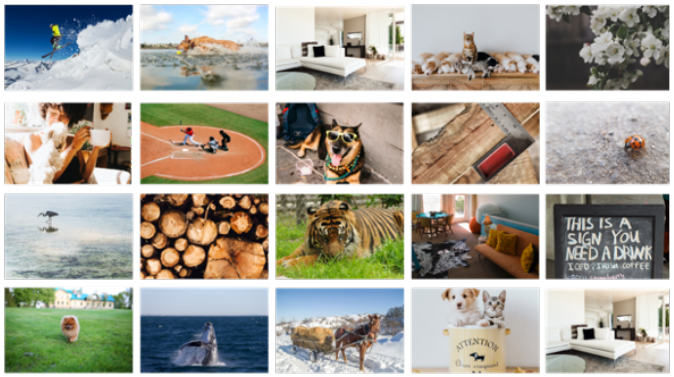
Target task: Few-shot Learning on VOC07 dataset (20 classes, 1-8 examples per class)

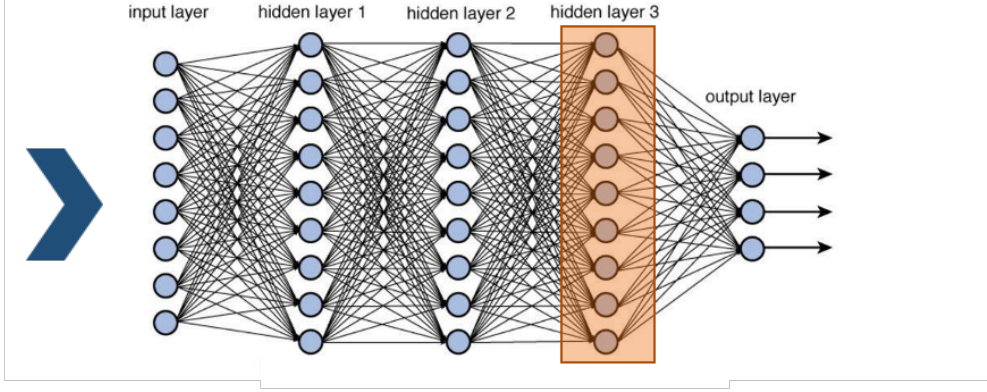
vector

- Without representation learning: **5% - 10%** (random guess = **5%)**
- With representation learning: **50% - 80%**

Example in image representation

Source tasks (for training representation): ImageNet





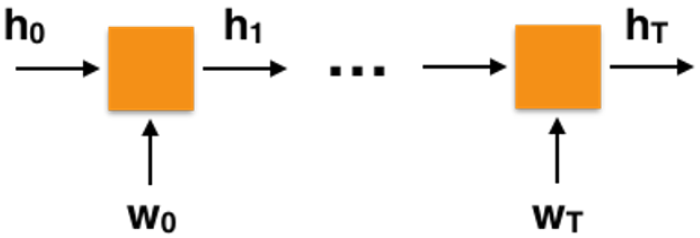
Target task: Few-shot Learning on VOC07 dataset (20 classes, 1-8 examples per class)

- Without representation learning: **5% - 10%** (random guess = **5%)**
- With representation learning: **50% - 80%**

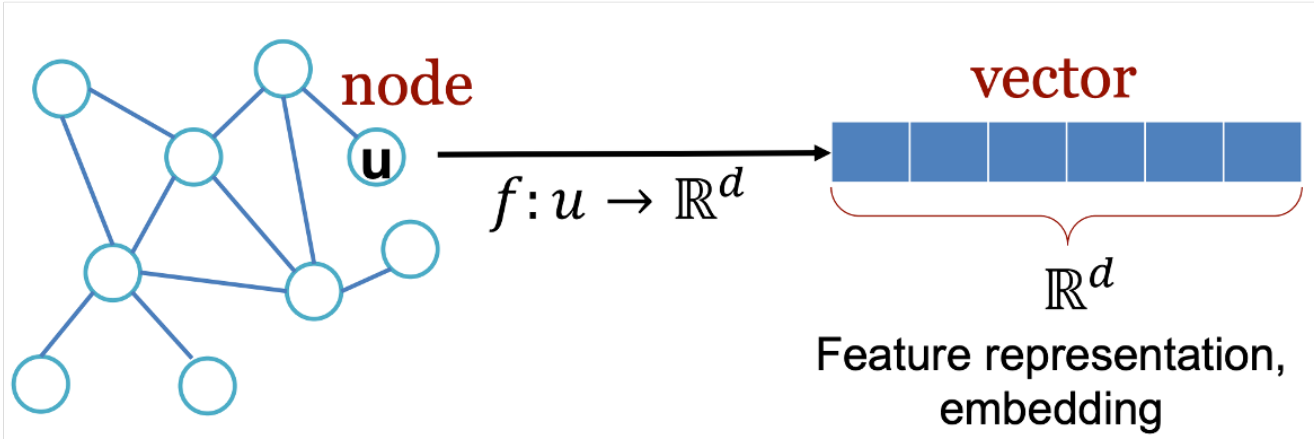
Examples

Natural Language Processing

Final hidden state:
Sentence representation



Graph Representation Learning



Representation learning

- A function that maps the raw input to a compact representation (feature vector). Learn an **embedding / feature / representation** from **labeled/unlabeled data.**
- Supervised:
  - Multi-task learning
  - Meta-learning
  - Multi-modal learning
  - …
- Unsupervised:
  - PCA
  - ICA
  - Dictionary learning
  - Sparse coding
  - Boltzmann machine
  - Autoencoder
  - Contrastive learning
  - Self-supervised learning
  - …

Desiderata for representations

Many possible answers here.

- **Downstream usability:** the learned features are "useful" for downstream tasks:
  - Example: a linear (or simple) classifier applied on the learned features only requires a small number of labeled samples. A classifier on raw inputs requires a large mount of data.

- **Interpretability:** the learned features are semantically meaningful, interpretable by a human, can be easily evaluated.
  - Not well-defined mathematically.
  - **Sparsity** is an important subcase.

Desiderata for representations

From Bengio, Courville, Vincent ’14:

- **Hierarchy / compositionality:** video/image/text are expected to have hierarchial structure: need *deep* learning.

- **Semantic clusterability**: features of the same “semantic class” (e.g. images in the same class) are clustered together.

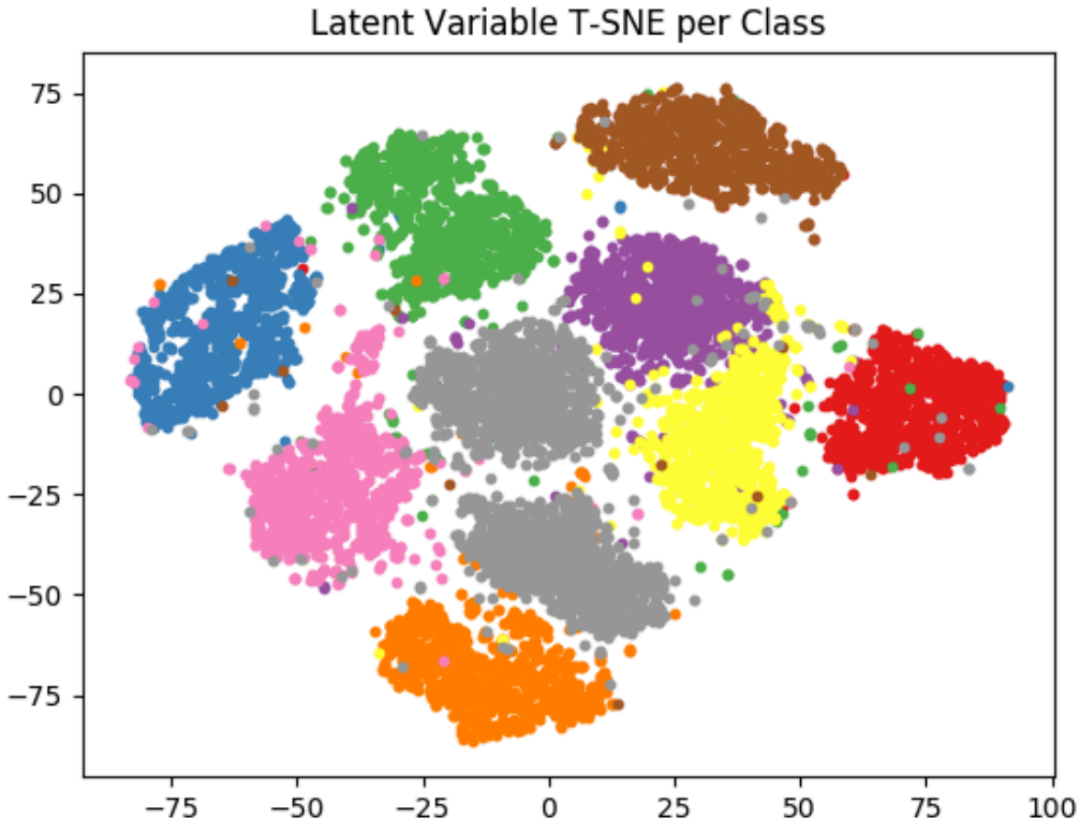
- **Linear interpolation**: in the representation space, linear interpolations produce meaningful data points (latent space is convex). Also called *manifold flattening.*

- **Disentanglement**: features capture “independent factors of variation” of data. A popular principle in modern unsupervised learning.

Semantic clustering

Semantic clusterability: features of the same "semantic class" (e.g. images in the same class) are clustered together.

0, ··· , 9

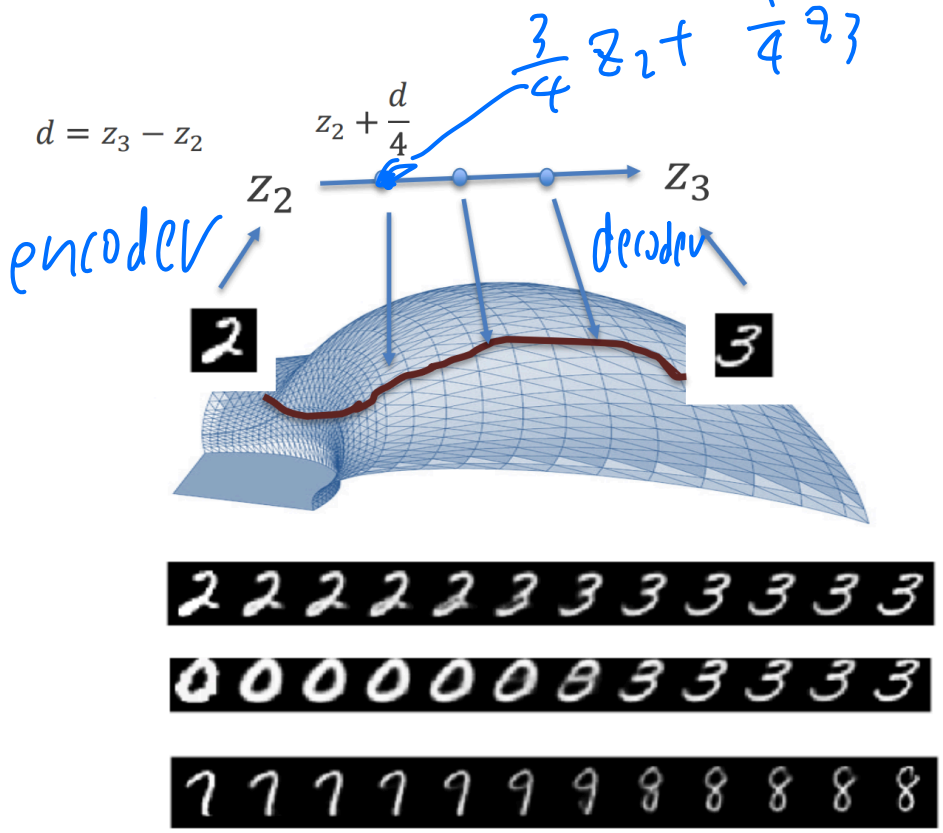


Intuition: If semantic classes are linearly separable, and labels on downstreams tasks depend linearly on semantic classes: we only need to learn a simple classifer.

t-SNE projection (a data visualization method) of VAE-learned features of 10 MNIST classes.

Linear interpolation

Linear interpolation: in the representation space, linear interpolations produce meaningful data points (latent space is convex).



Intuition: the data lies on a manifold which is complicated/ curved.

The latent variable manifold is a convex set: moving in straight lies is still on it.

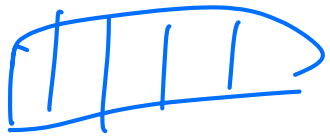
Interpolations for a VAE trained feature on MNIST.

Linear interpolation

Linear interpolation: in the representation space, linear interpolations produce meaningful data points (latent space is convex).

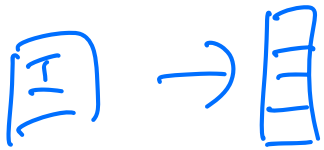
Interpolations for a BigGAN image.

Disentanglement



Disentanglement: features capture "independent factors of variation" of data (Bengio, Courville, Vincent '14).

- Very popular in modern unsupervised learning.
- Strong connections with generative models: $p_\theta(z) = \Pi_i p_\theta(z_i)$.



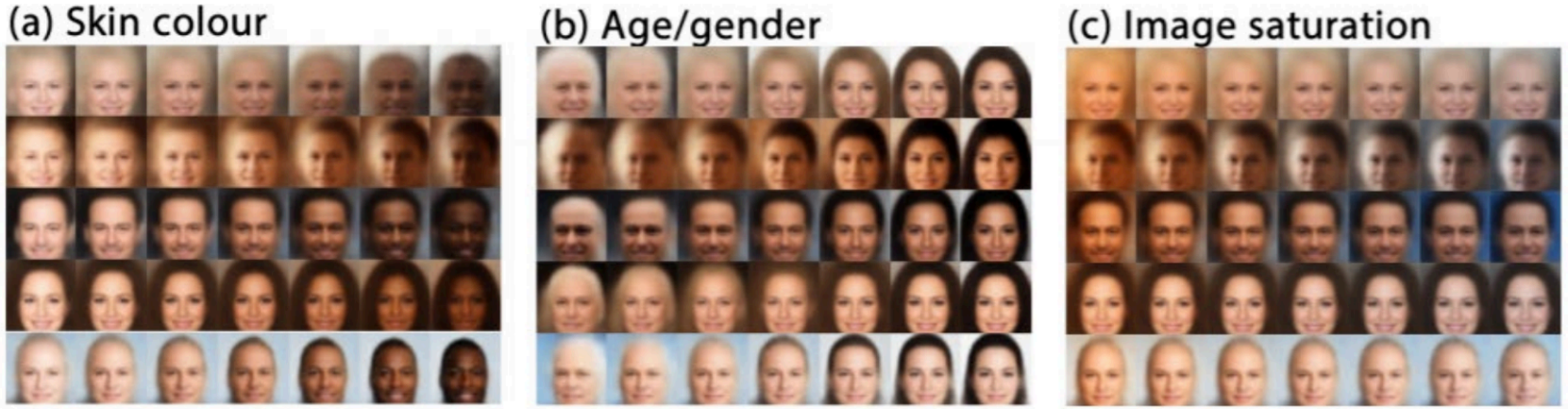


Figure 4: **Latent factors learnt by β-VAE on celebA:** traversal of individual latents demonstrates that β-VAE discovered in an unsupervised manner factors that encode skin colour, transition from an elderly male to younger female, and image saturation.

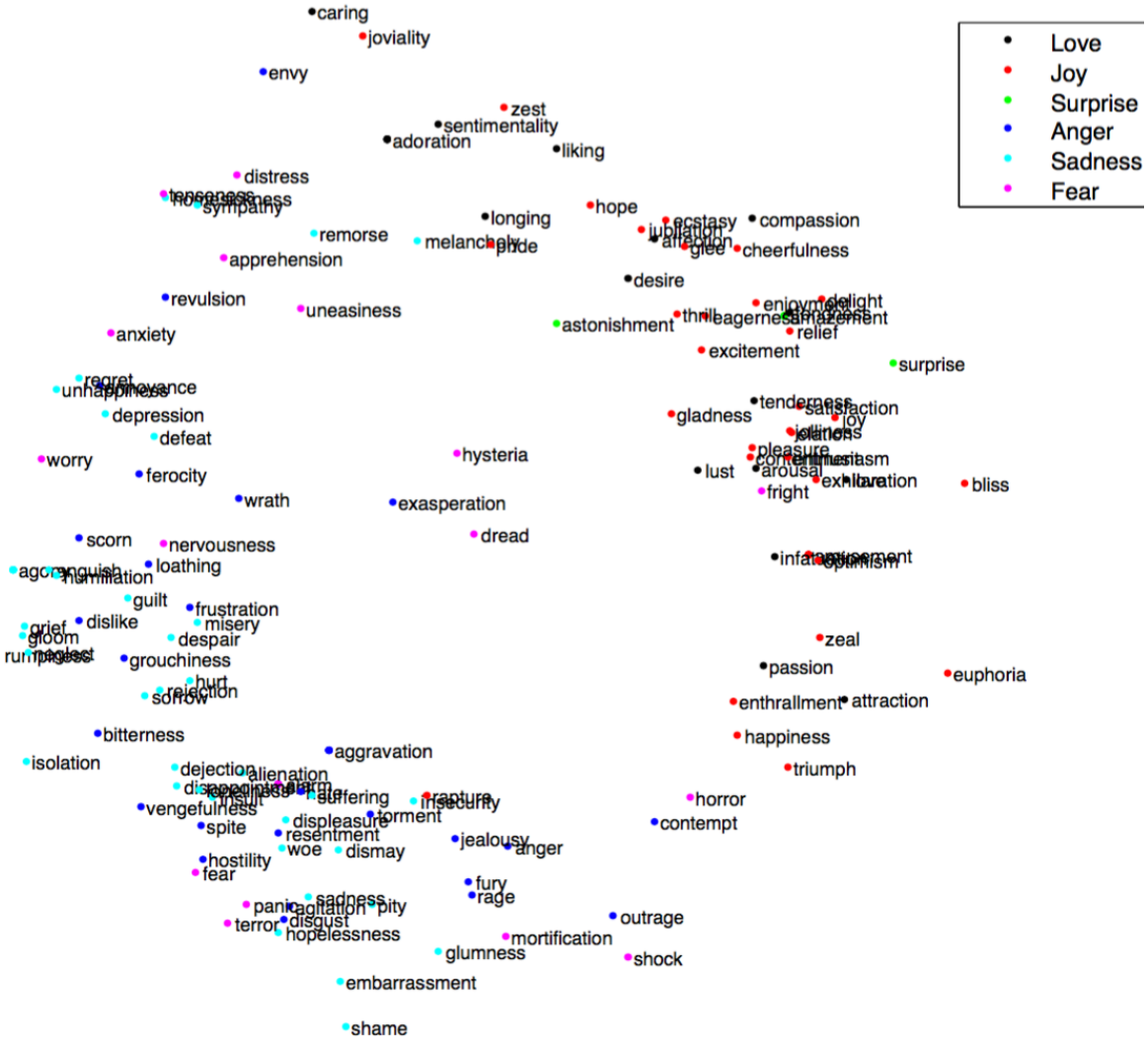
Pre-training Learning Methods for Text

Word embeddings, word2vec

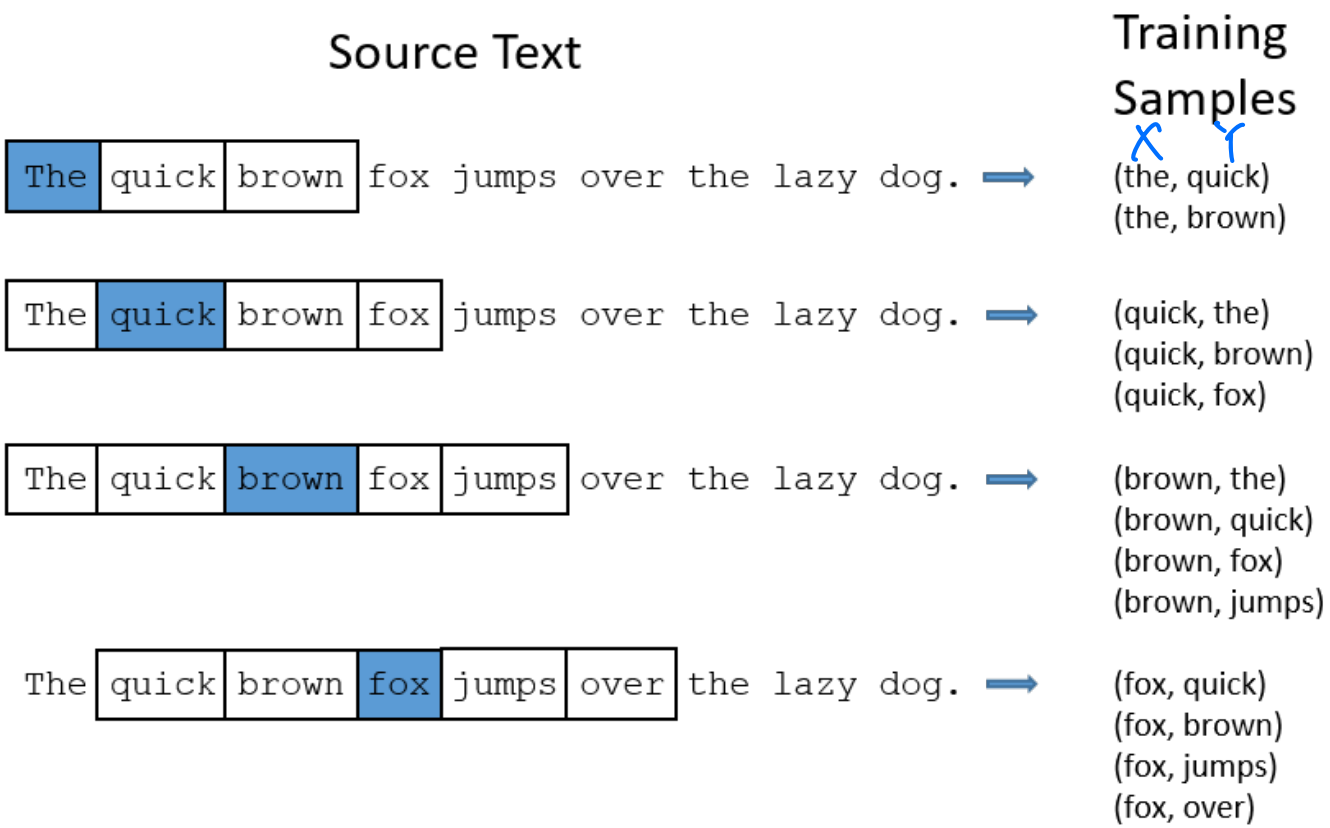
Can we **embed words** into a latent space?

This embedding came from directly querying for relationships.

word2vec is a popular unsupervised learning approach that just uses a text corpus (e.g. nytimes.com)

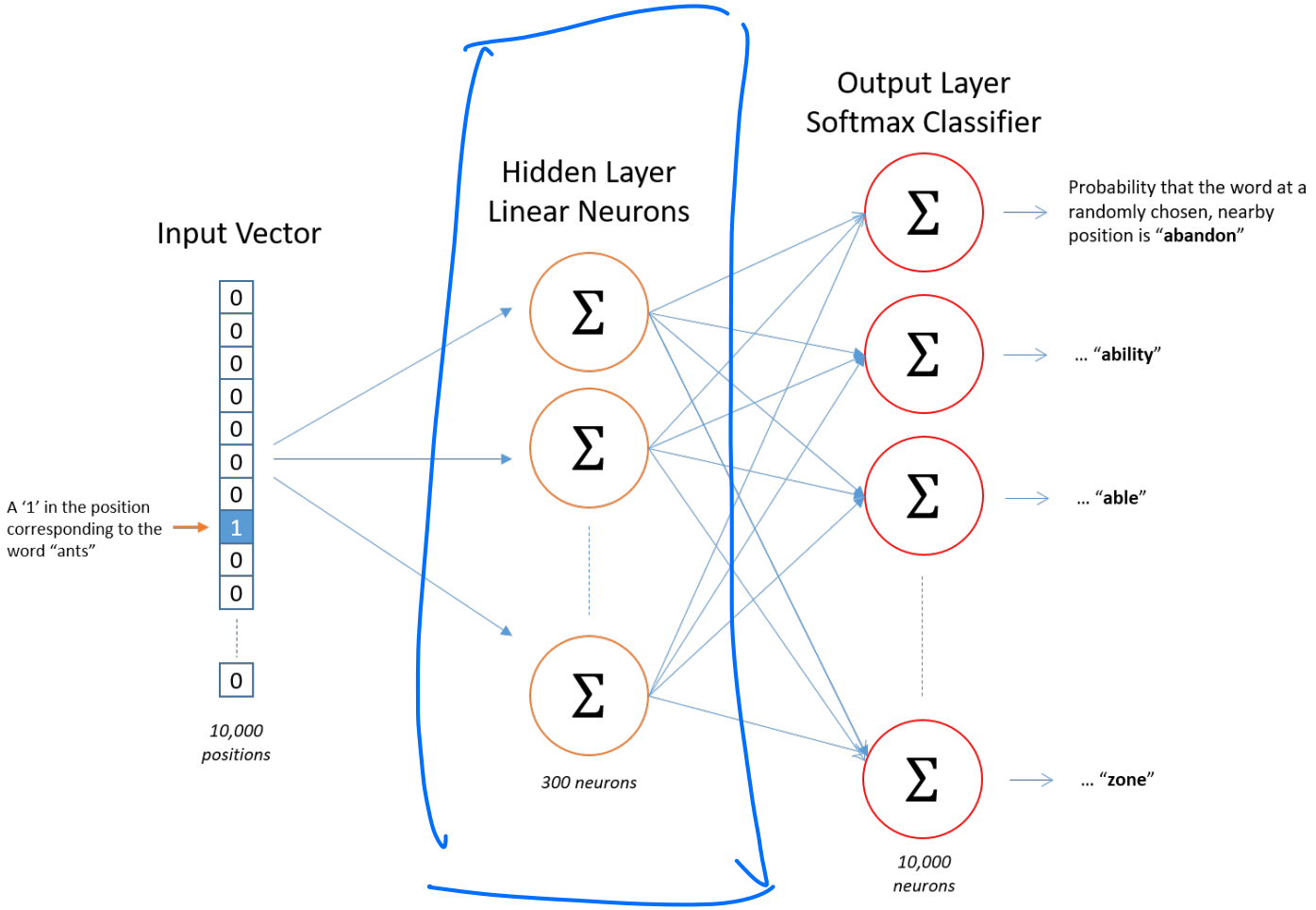


Word embeddings, word2vec



slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

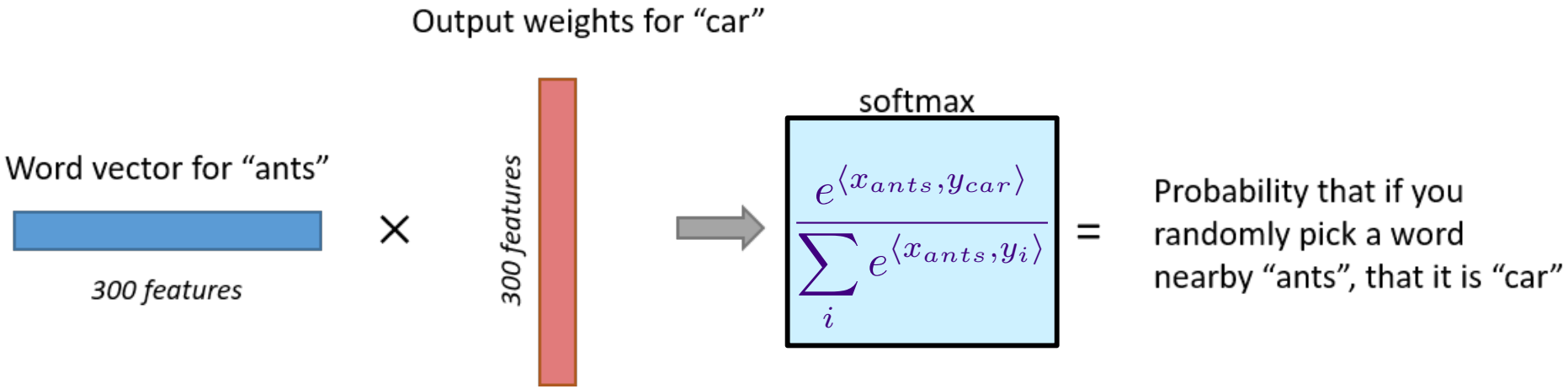
Word embeddings, word2vec



Training neural network to predict co-occuring words. Use first layer weights as embedding, throw out output layer

slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word embeddings, word2vec

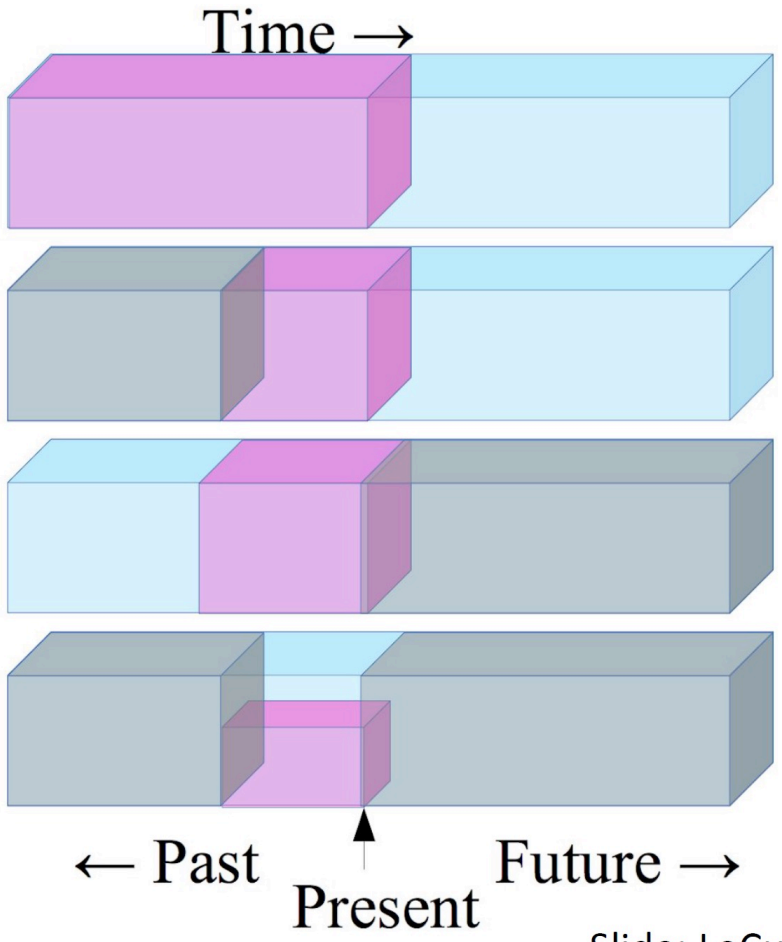


Training neural network to predict co-occuring words. Use first layer weights as embedding, throw out output layer

slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Self-supervised learning

- Predict any part of the input from any other part.
- Predict the future from the past.
- Predict the future from the recent past.
- Predict the past from the present.
- Predict the top from the bottom.
- Predict the occluded from the visible
- Pretend there is a part of the input you don't know and predict that.



Slide: LeCun

Tokenizer

- Break raw text into smaller units (tokens).
  - raw text can be document, code, math, etc.
- A language model places a probability distribution over sequences of tokens.

- Raw text: string = "Hello, 🌍! 你好!”
- indices = [15496, 11, 995, 0]
- Two functions:
  - Encode: from string to indices
  - Decode: from indices to a string

- Metric: compression ratio
  - How many original characters or bytes are represented by one token on average
  - Depends on text distribution, vocabulary size (often 32k - 50k), etc
  - English: 3-4 chars/token, Chinese/Japanese/Korean: 1-1.8 chars/token, Code: 2-3 chars/tokens, Math/Latex: 1-2 chars/token.

Tokenizers

- Character-based tokenization: bad compression ratio
  - ord("🌍") == 127757, chr(127757) == "🌍"
- Byte-based tokenization (UTF-8): bad compression ratio
  - bytes("🌍", encoding="utf-8") == b"\xf0\x9f\x8c\x8d"
- Word-based tokenization:
  - Split strings into words, then construct encoding/decoding.
  - # of words is huge, many words are rare, no fixed voc size.

- **Byte Pair Encoding (BPE):**
  - An algorithm by Philip Gage (1994).
  - Adopted in machine translation. Used in GPT-2.
  - Idea: **train** the tokenizer on raw text to determine the vocabulary.
  - Intuition: common sequences of characters are represented by a single token, rare sequences are represented by many tokens => improve compression ratio.
  - Training distribution/data matters!

Byte Pair Encoding

```
def train_bpe(string: str, num_merges: int) -> BPETokenizerParams:  # @inspect string, @inspect num_merges
    Start with the list of bytes of string.
    indices = list(map(int, string.encode("utf-8")))  # @inspect indices
    merges: dict[tuple[int, int], int] = {}  # index1, index2 => merged index
    vocab: dict[int, bytes] = {x: bytes([x]) for x in range(256)}  # index -> bytes

    for i in range(num_merges):
        Count the number of occurrences of each pair of tokens
        counts = defaultdict(int)
        for index1, index2 in zip(indices, indices[1:]):  # For each adjacent pair
            counts[(index1, index2)] += 1  # @inspect counts

        Find the most common pair.
        pair = max(counts, key=counts.get)  # @inspect pair
        index1, index2 = pair

        Merge that pair.
        new_index = 256 + i  # @inspect new_index
        merges[pair] = new_index  # @inspect merges
        vocab[new_index] = vocab[index1] + vocab[index2]  # @inspect vocab
        indices = merge(indices, pair, new_index)  # @inspect indices

    return BPETokenizerParams(vocab=vocab, merges=merges)
```

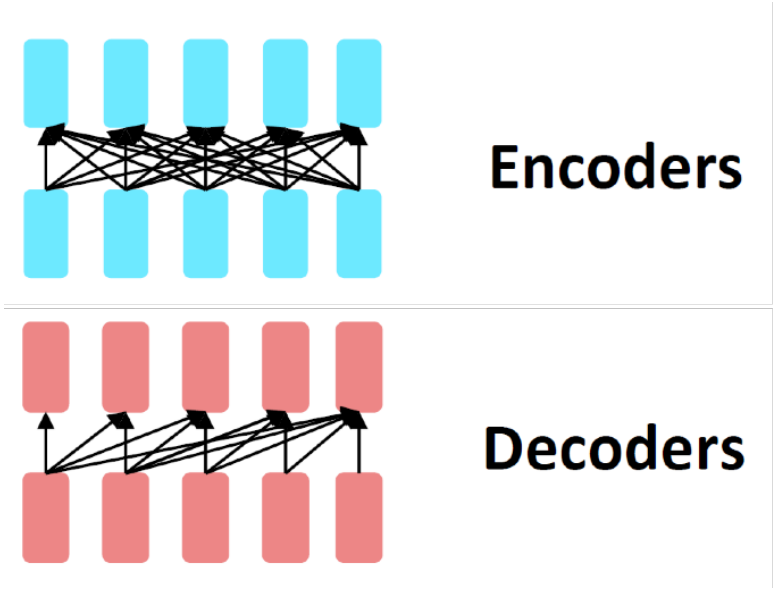
from Stanford CS 336

Transformer Pre-training

- Collect a large amount of corpus (wiki) and pre-train a large transformer

- For down-stream tasks, fine-tune the pretrained model
  - Or use the pretrained model to extract features

- How to pretrain a transformer on texts?
  - Pretrain an encoder
    - bi-directional
  - Pretrain a decoder
    - auto-regressive

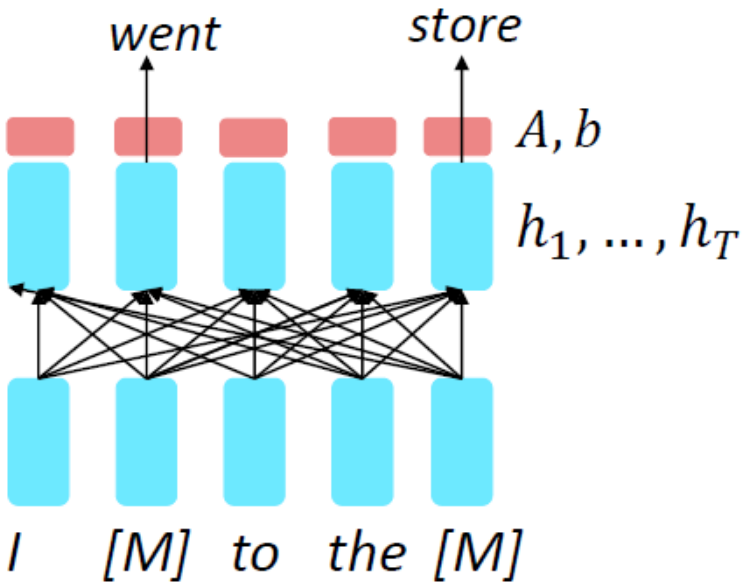


Pre-training Transformer Encoder

- Pre-training a bi-directional encoder
  - Cannot directly adopt language modeling
  - **Idea:** word prediction given contexts (similar to word2vec)

- Masked language model
  - Randomly “masked out” some words
  - Run full transformer encoder
  - Predict the words at masked positions

- Designed for feature extraction
  - Suitable for down-stream tasks

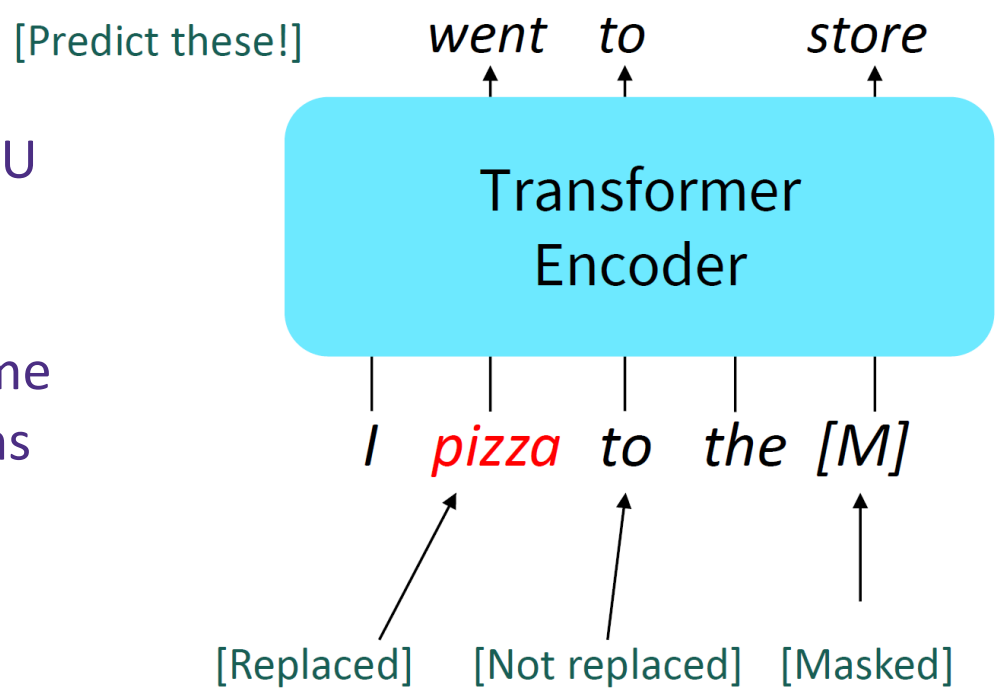


Pre-training Transformer Encoder

- **BERT:** Pre-training of Deep Bidirectional Transformers

- Devlin et al., Google, 2018
  - BERT-base: 12 layers, 110M params
  - BERT-large: 24 layers, 340M params
  - Training on 64 TPUs in 4 days
  - Fine-tuning can be down in a single GPU

- Masked language model
  - Masked out input words 80% of the time
  - Replace 10% words with random tokens
  - 10% words remain unchanged
  - Predict 15% of word tokens

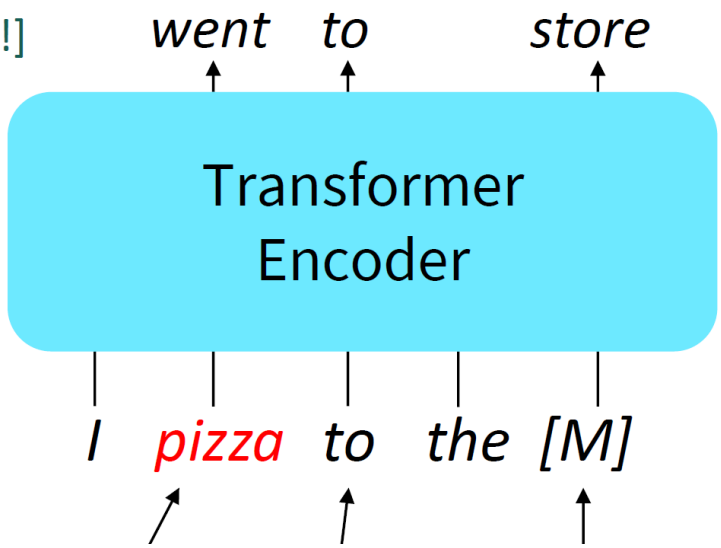


Pre-training Transformer Encoder

- **BERT:** Pre-training of Deep Bidirectional Transformers

- Devlin et al., Google, 2018
  - BERT-base: 12 layers, 110M params
  - BERT-large: 24 layers, 340M params
  - Training on 64 TPUs in 4 days
  - Fine-tuning can be down in a single GPU

- Masked language model
  - Masked out input words 80% of the time
  - Replace 10% words with random tokens
  - 10% words remain unchanged



System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Pre-training Transformer Encoder

- **BERT:** Pre-training of Deep Bidirectional Transformers

- **RoBERTa**: A robustly optimized BERT Pretraining approach
  - Facebook AI and UW, '19
  - More compute, data, and improved objective

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

Pre-training Decoder

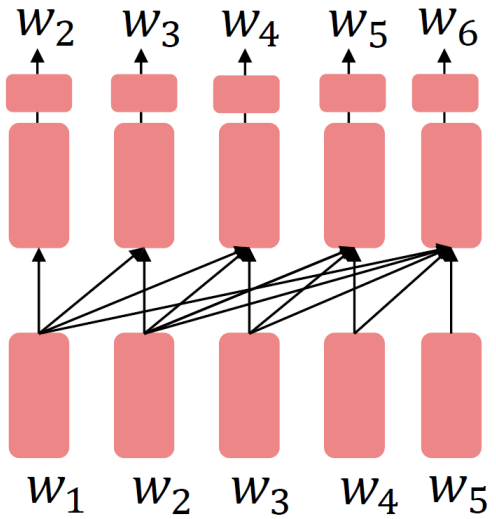
- Decoder Pretraining
  - Just train a language model over corpus.
  - Good for generative task (e.g., text generation)

- Generative Pretrained Transformer (GPT, Open AI ’18)
  - 120 layers transformer, 7680d hidden, 3072-d MLP
  - Data: BooksCropus (>7k books)

- GPT-2 (Radford et al., OpenAI ’19)
  - 1.5B parameters, 40GB internet texts

- GPT-3 (OpenAI ’20)
  - Language models are few-shot learners
  - 175B parameters

- Also Image GPT (OpenAI ’20)



Pre-training Decoder

- GPT-3 (OpenAI ’20)
  - You may not need to fine-tune the model parameters for downstream tasks.
  - New paradigm: prompt learning

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Code: `px.line(df.query("continent == 'Europe' and country == 'France'"), x='year', y='gdpPercap', color='country', log_y=False, log_x=False)`

Description: Actually, replace GDP with population

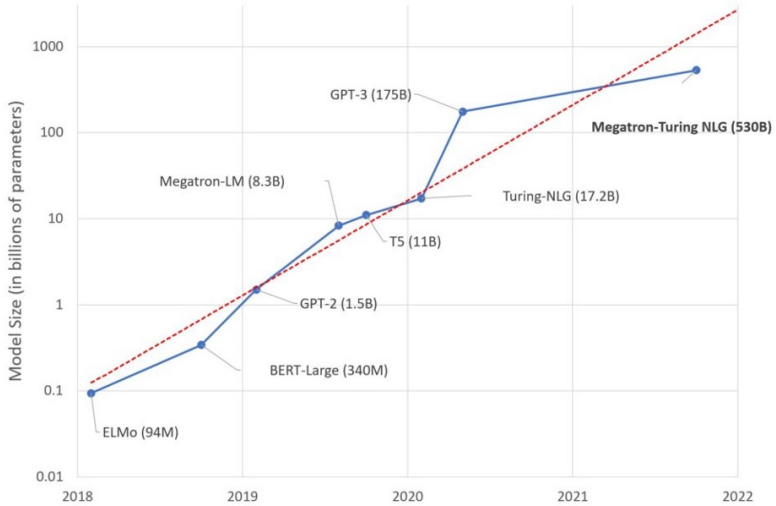
Code: `px.line(df.query("continent == 'Europe' and country == 'France'"), x='year', y='pop', color='country', log_y=False, log_x=False)`

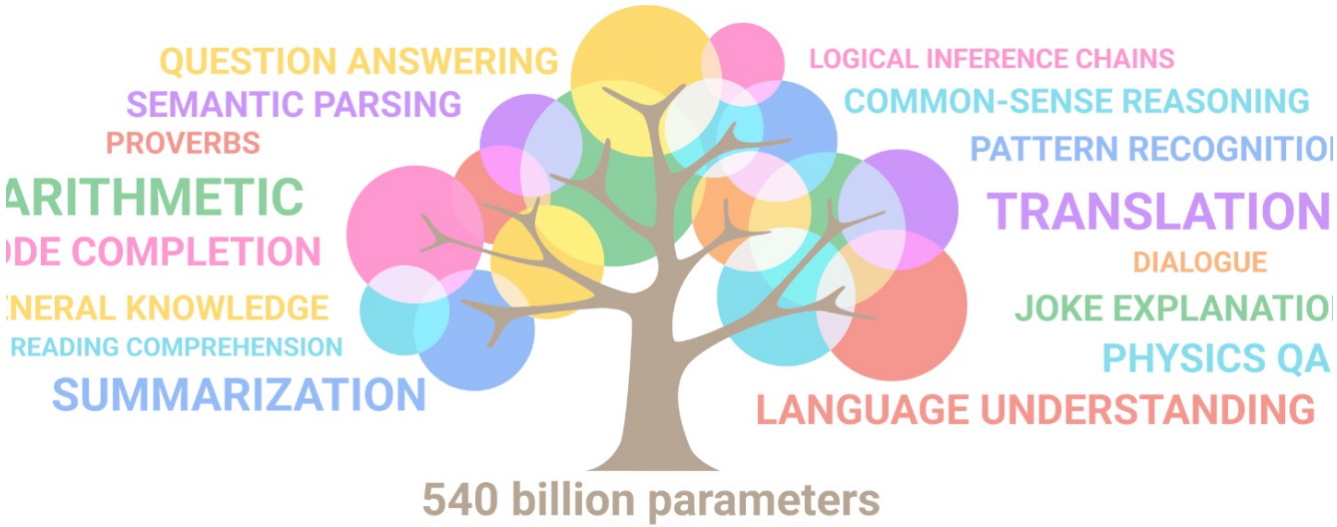
Description: Put y-axis on log scale

Code: `px.line(df.query("continent == 'Europe' and country == 'France'"), x='year', y='pop', color='country', log_y=True, log_x=False)`

Pre-training Decoder

- A big ongoing race on training large language models
  - Megatron-Turing NLG (530B, Microsoft, ’22)
  - Pathways Language Model (540B, Google, ’22)





Pre-training data

- Data is even more important than architecture / optimizer.
  - Llama 3 has full recipe of architecture and training, but little information about data.

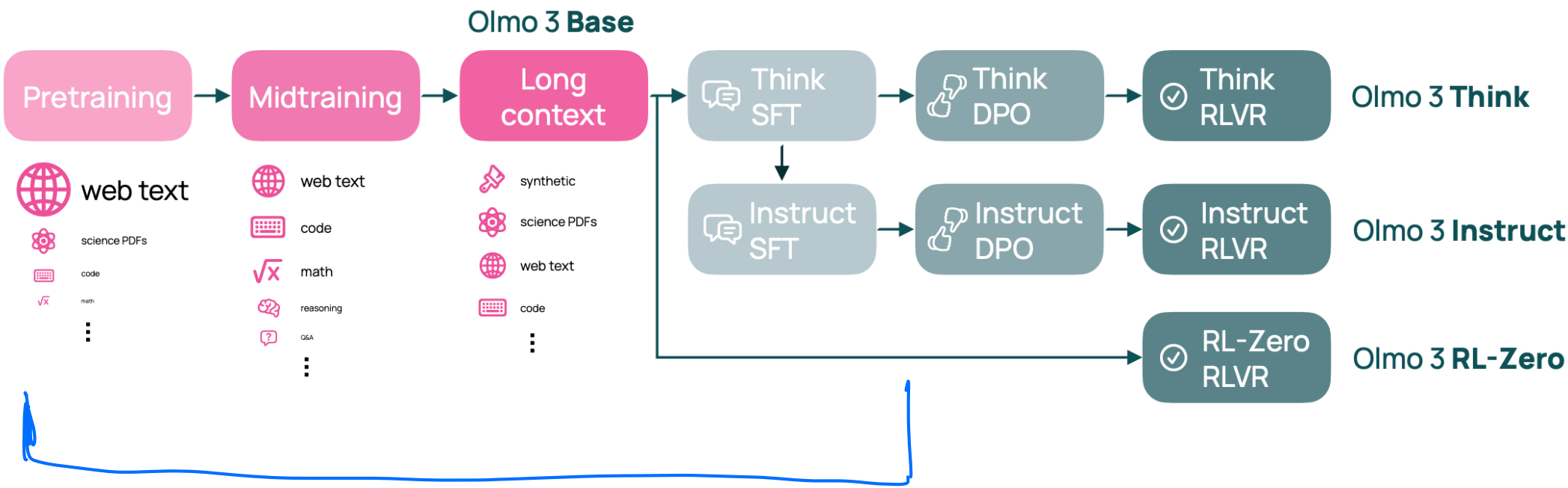
3.1 Pre-Training Data

We create our dataset for language model pre-training from a variety of data sources containing knowledge until the end of 2023. We apply several de-duplication methods and data cleaning mechanisms on each data source to obtain high-quality tokens. We remove domains that contain large amounts of personally identifiable information (PII), and domains with known adult content.

LLM Pipeline

close-source model: gpt
open: llama, qwen gemini

- Olmo 3
  - Best Open **Recipe** Model: Everything open



Pre-training data

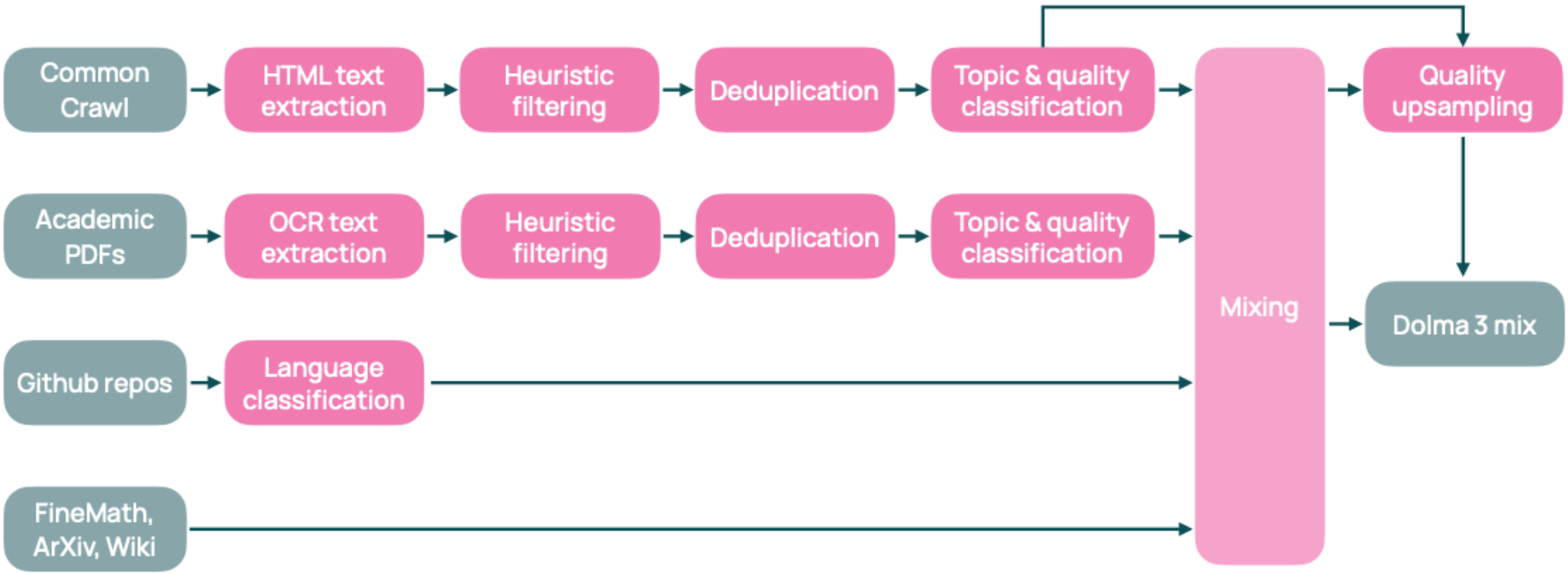
- Pre-training data

Source	Type	9T Pool		6T Mix		150B Mix	
		Tokens	Docs	Tokens	Docs	Tokens	Docs
Common Crawl	Web pages	8.14T	9.67B	4.51T (76.1%)	3.15B	121B (76.9%)	84.5M
olmOCR science PDFs	Academic documents	972B	101M	805B (13.6%)	83.8M	19.9B (12.6%)	2.25M
Stack-Edu (Rebalanced)	GitHub code	137B	167M	409B (6.89%)	526M	11.1B (7.06%)	14.3M
arXiv	Papers with LaTeX	21.4B	3.95M	50.8B (0.86%)	9.10M	1.29B (0.82%)	247K
FineMath 3+	Math web pages	34.1B	21.4M	152B (2.56%)	95.5M	4.10B (2.60%)	2.57M
Wikipedia & Wikibooks	Encyclopedic	3.69B	6.67M	2.51B (0.04%)	4.24M	64.6M (0.04%)	119K
Total		**9.31T**	**9.97B**	**5.93T (100%)**	**3.87B**	**157B (100%)**	**104M**

Table 4 Composition of Dolma 3 Mix including our 9T pool of data, the 6T mix we used for final model training, and the 150B mix we used for experimentation.

Pre-training data

fast



- Heuristic filtering: spam filtering, excessive symbols, sentence too short, etc…
- Deduplication: min-hash, fuzzy duplicate removal
- Classification: topic classification, whether useful or not (fastText classifier), etc
- Mixing: every document has topic label and quality score, balance topic, upweight quality

distilled from llm

Mid-training Data

- Improve some fundamental capabilities (higher quality data)

Type	Source	2T Pool		100B Mix	
		Tokens	Docs	Tokens	Docs
Math (synth)	TinyMATH Mind**	899M	1.42M	898M (0.9%)	1.52M
Math (synth)	TinyMATH PoT**	241M	729K	241M (0.24%)	758K
Math (synth)	CraneMath*	5.62B	6.55M	5.62B (5.63%)	7.24M
Math (synth)	MegaMatt*	3.88B	6.79M	1.73B (1.73%)	3.23M
Math (synth)	Dolmino Math^^	10.7B	21M	10.7B (10.7%)	22.3M
Code	StackEdu (FIM)^	21.4B	32M	10.0B (10.0%)	16.2M
Python (synth)	CraneCode*	18.8B	19.7M	10.0B (10.0%)	11.7M
QA (synth)	Reddit To Flashcards**	21.6B	370M	5.90B (5.9%)	101M
QA (synth)	Wiki To RCQA**	4.22B	22.3M	3.0B (3.0%)	16.3M
QA (synth)	Nemotron Synth QA^	487B	972M	5.0B (5.0%)	10.6M
Thinking (synth)	Math Meta-Reasoning**	1.05B	984K	381M (0.38%)	401K
Thinking (synth)	Code Meta-Reasoning**	1.27B	910K	459M (0.46%)	398K
Thinking (synth)	Program-Verifiable**	438M	384K	159M (0.16%)	158K
Thinking (synth)	OMR Rewrite FullThoughts^	850M	291K	850M (0.85%)	394K
Thinking (synth)	QWQ Reasoning Traces^	4.77B	438K	1.87B (1.87%)	401K
Thinking (synth)	General Reasoning Mix^	2.48B	668K	1.87B (1.87%)	732K
Thinking (synth)	Gemini Reasoning Traces^	246M	55.2K	246M (0.25%))	85.1K
Thinking (synth)	Llama Nemotron Reasoning Traces^	20.9B	3.91M	1.25B (1.25%)	368K
Thinking (synth)	OpenThoughts2 Reasoning Traces^	5.6B	1.11M	1.25B (1.25%)	402K
Instruction (synth)	Tulu 3 SFT^^	1.61B	1.95M	1.1B (1.1%)	1.45M
Instruction (synth)	Dolmino 1 Flan^^	16.8B	56.9M	5.0B (5.0%)	14.8M
PDFs	olmOCR science PDFs (HQ subset)^	240B	28.7M	4.99B (5.0%)	1.20M
Web pages	STEM-Heavy Crawl^	5.21B	5.16M	4.99B (5.0%)	5.53M
Web pages	Common Crawl (HQ subset)^	1.32T	965M	22.4B (22.5%)	18.3M
Total		**2.19T**	**2.52B**	**99.95B (100%)**	**236M**

Long-context Extension

- Extend context window from 8k to 64k

Source	Length bucket	600B Pool		50B Mix	
		Tokens	Docs	Tokens	Docs
olmOCR PDFs	8K-16K	144B (22.5%)	12.7M	2.27B (4.55%)	235K
olmOCR PDFs	16K-32K	115B (18.0%)	5.06M	1.85B (3.70%)	110K
olmOCR PDFs	32K-64K	106B (16.6%)	2.30M	4.81B (9.63%)	177K
olmOCR PDFs	64K-128K	96.0B (15.0%)	1.05M	–	–
olmOCR PDFs	128K-256K	60.8B (9.5%)	342K	–	–
olmOCR PDFs	256K-512K	35.1B (5.49%)	97.1K	–	–
olmOCR PDFs	512K-1M	21.5B (3.36%)	30.2K	–	–
olmOCR PDFs	1M+	26.9B (4.21%)	12.2K	–	–
olmOCR PDFs + synth **CWE**	32K-64K	8.77B (1.37%)	189K	1.94B (3.88%)	71.3K
olmOCR PDFs + synth **REX**	32K-64K	24.1B (3.77%)	492K	6.08B (12.2%)	217K
Midtraining data mix	Variable	–	–	33.0B (66.1%)	79.2M
Total		**639B**	**22.3M**	**50.0B (100%)**	**80.0M**

Supervised Fine-Tuning

(Q, A)

Supervised Fine-Tuning (Instruction Following): Token Masking

<BOS>	User	:	Explain	SGD	.	Assistant	:	SGD	is	...	<EOS>
Mask	Mask	Mask	Mask	Mask	Mask	Loss	Loss	Loss	Loss	Loss	Loss

Loss

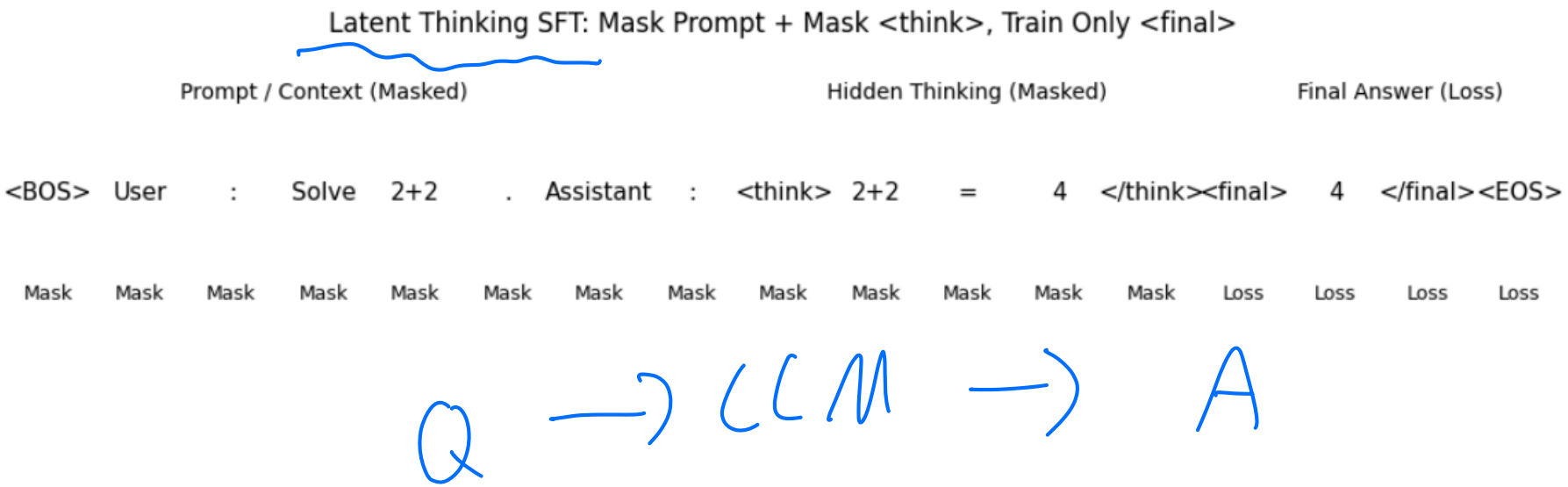
Total loss for sentence: $\sum_{i=1}^{N} L_i$, L_i loss for i^{th} position

$\sum_{i=7}^{N} L_i$

- Do not train on masked tokens.
- Data format: prompt, response.

Supervised Fine-Tuning with Thinking

- Chain-of-Thought: COT.



SFT Data

- High quality.

Category	Prompt Dataset	7B Count	32B Count	Reference
Chat &	WildChat	83,054	76,209	Zhao et al. (2024a)
Precise IF	OpenAssistant	6,800	6,647	Köpf et al. (2024)
	Dolci Think Persona Precise IF	223,123	220,530	–
	Dolci Think Precise IF	135,792	135,722	–
Math	Dolci Think OpenThoughts 3+ Math⇑	752,997	752,997	Guha et al. (2025a)
	Dolci Think OpenThoughts 3+ STEM⇑	99,269	99,268	Guha et al. (2025a)
	SYNTHETIC-2-SFT-Verified	104,569	104,548	PrimeIntellect (2025)
Coding	Nemotron Post-Training Code	113,777	113,777	NVIDIA AI (2025)
	Dolci Think OpenThoughts 3+ Code⇑	88,900	88,899	Guha et al. (2025a)
	Dolci Think Python Algorithms⇑	466,677	466,676	–
Safety	CoCoNot	10,227	9,549	Brahman et al. (2024)
	WildGuardMix	38,315	36,673	Han et al. (2024)
	WildJailbreak	41,100	40,002	Jiang et al. (2024)
Multilingual	Aya	98,597	97,156	Singh et al. (2024)
Other	TableGPT	4,981	4,973	Zha et al. (2023)
	Olmo Identity Prompts	290	290	–
Total		2,268,468	2,253,916	

Parameter-Efficient Fine-Tuning

LoRA: Low-Rank Adaptation of Large Language Models
(Hu et al. 2021)

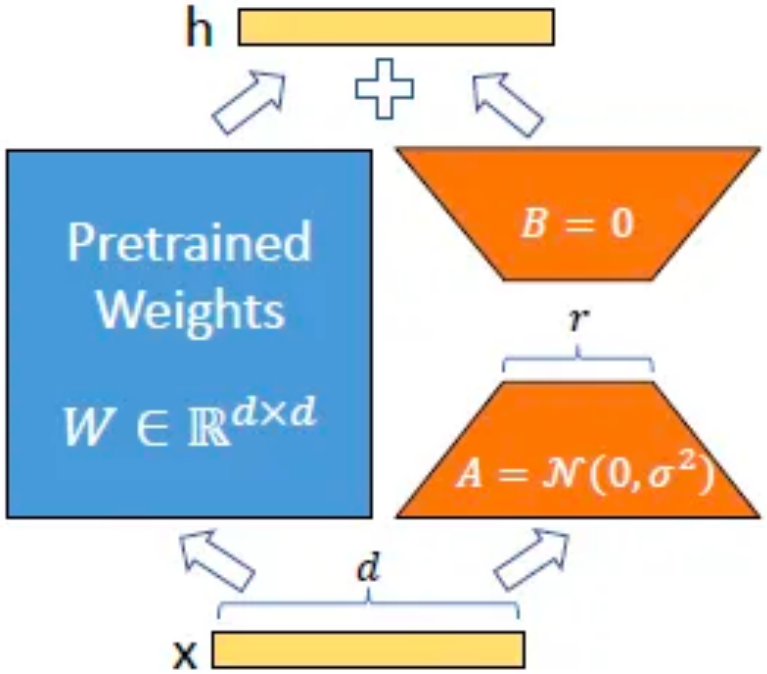


Figure 1: Our reparametrization. We only train A and B.

$2(dr)$ vs. d^2
$r \ll d$

$A: d \times r$
$B: r \times d$

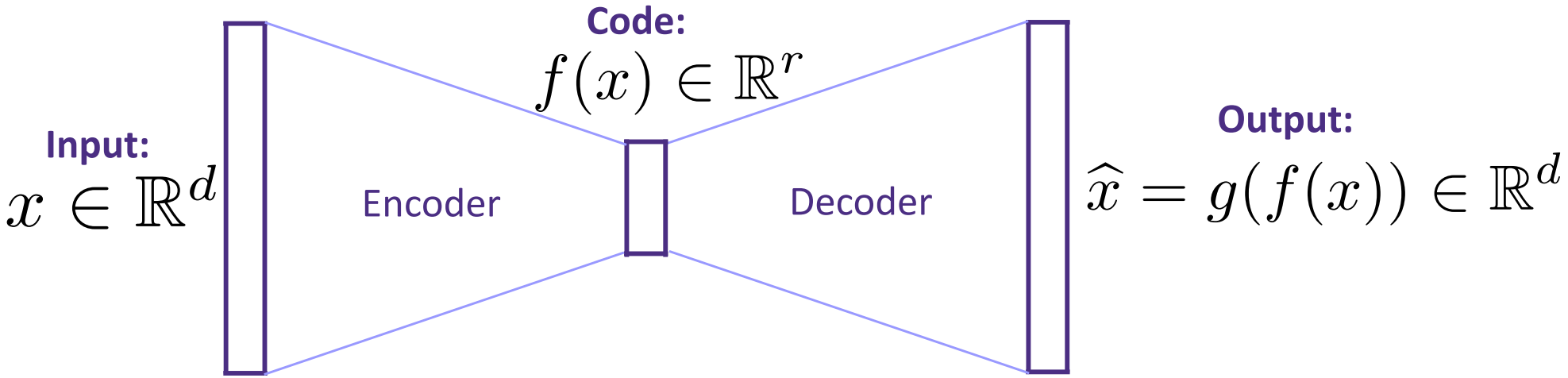
Fine-tune:
$W + AB$
σ 0

Pre-training Learning Methods for Vision

Autoencoders

$r < d$

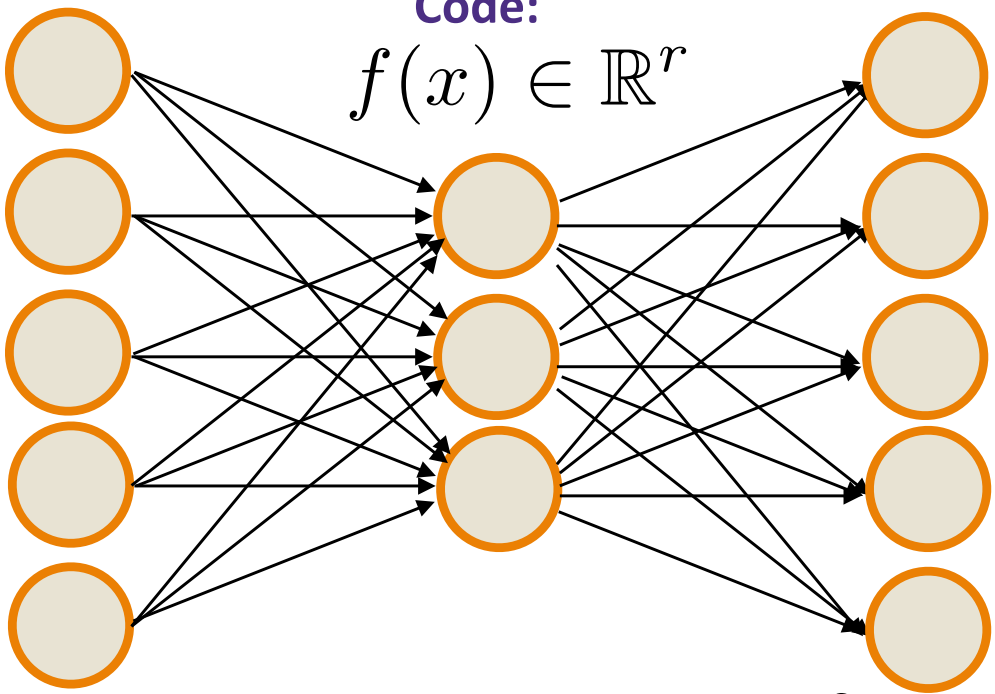
Find a low dimensional representation for your data by predicting your data



$$\underset{f,g}{\text{minimize}} \sum_{i=1}^{n} \|x_i - g(f(x_i))\|_2^2$$

reconstruction

Autoencoders



Code:

$$f(x) \in \mathbb{R}^r$$

Input:

$$x \in \mathbb{R}^d$$

Output:

$$\widehat{x} = g(f(x)) \in \mathbb{R}^d$$

$$\underset{f,g}{\text{minimize}} \sum_{i=1}^{n} \|x_i - g(f(x_i))\|_2^2$$

$A : r \times d$

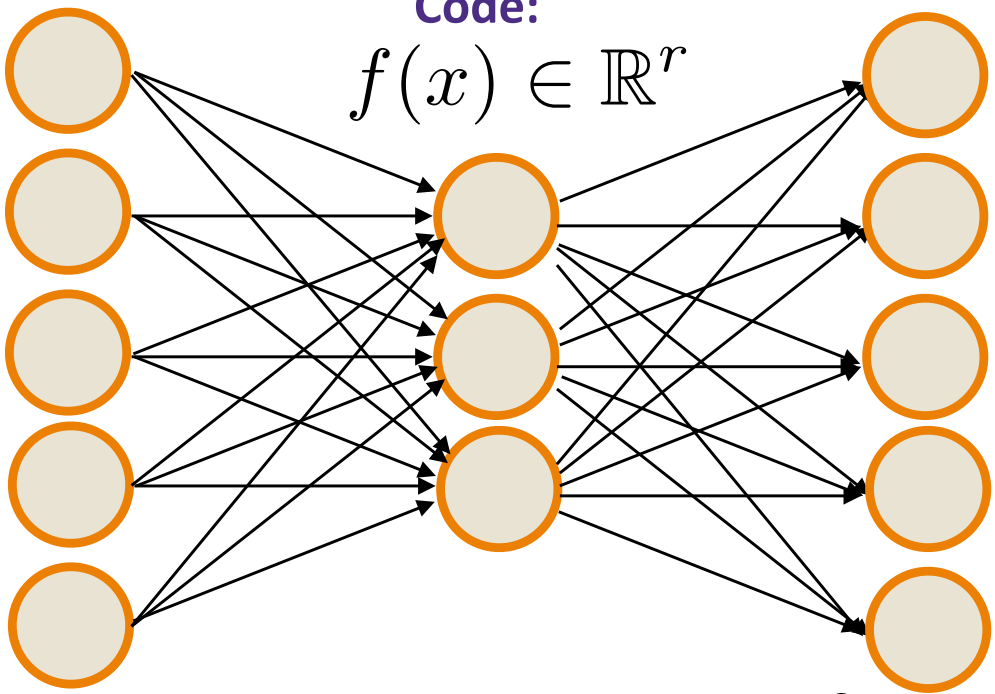
$B : d \times r$

X: data matrix

What if $f(X) = Ax$ and $g(y) = By$?

$$\sum_{i=1}^{n} \|X_i - BAX_i\|_2^2 \Leftrightarrow \|X - BAX\|_F^2$$

Autoencoders



Code:

$f(x) \in \mathbb{R}^r$

Input:

$x \in \mathbb{R}^d$

Output:

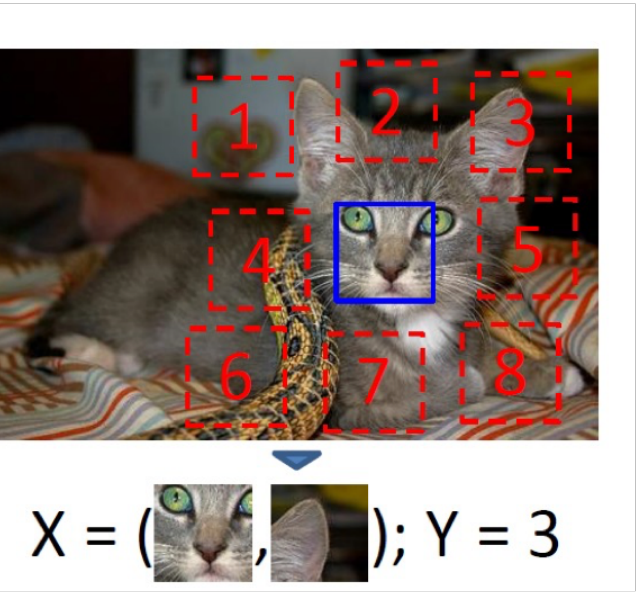
$\widehat{x} = g(f(x)) \in \mathbb{R}^d$

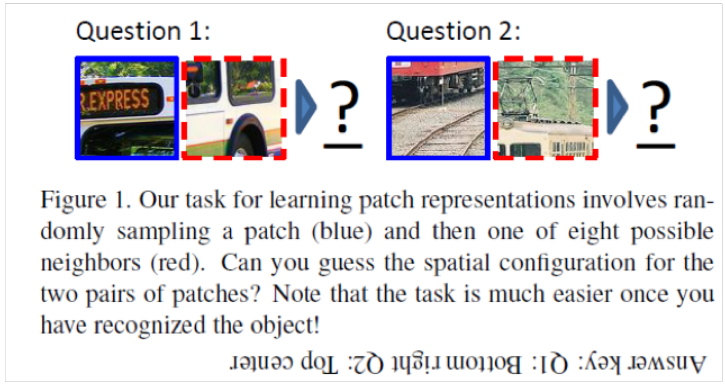
$$\underset{f,g}{\text{minimize}} \sum_{i=1}^{n} \|x_i - g(f(x_i))\|_2^2$$

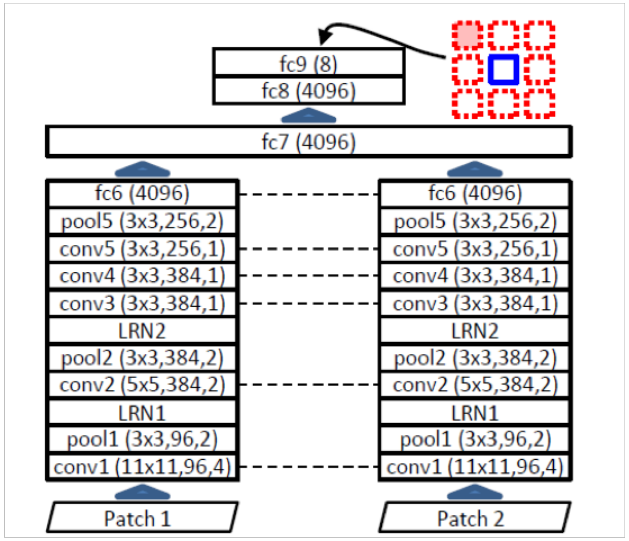
What if $f(X) = Ax$ and $g(y) = By$?

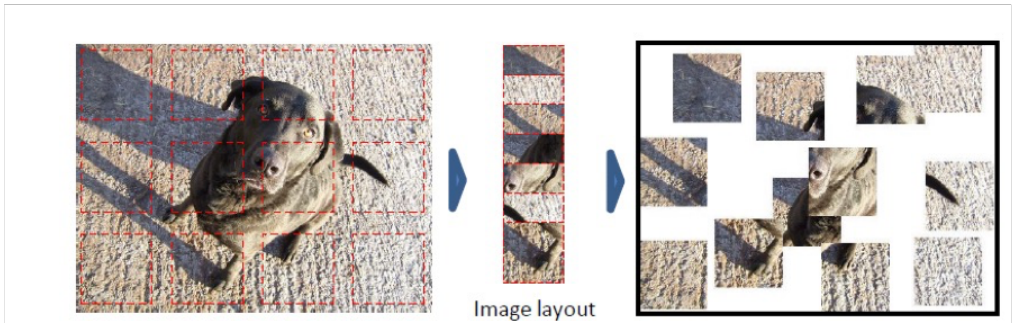
Self-supervised learning in computer vision

Context Prediction (Pathak et al., ‘15)









Self-supervised learning in computer vision

- **Feature learning by Inpainting** (Pathak et al., '16)
  - The most obvious analogue to word embeddings: predict parts of image from the remainder of image

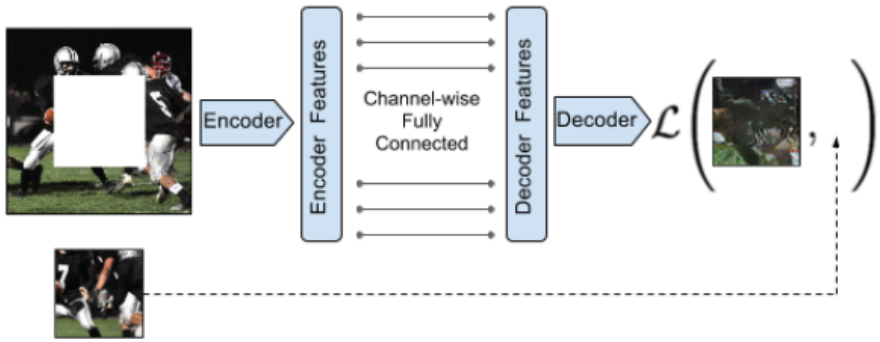


Figure 2: Context Encoder. The context image is passed through the encoder to obtain features which are connected to the decoder using channel-wise fully-connected layer as described in Section 3.1. The decoder then produces the missing regions in the image.

Architectures:
An encoder takes a part of an image, constructs a representation.

A decoder takes the representation, tries to reconstruct the missing part.

Trickier than NLP:

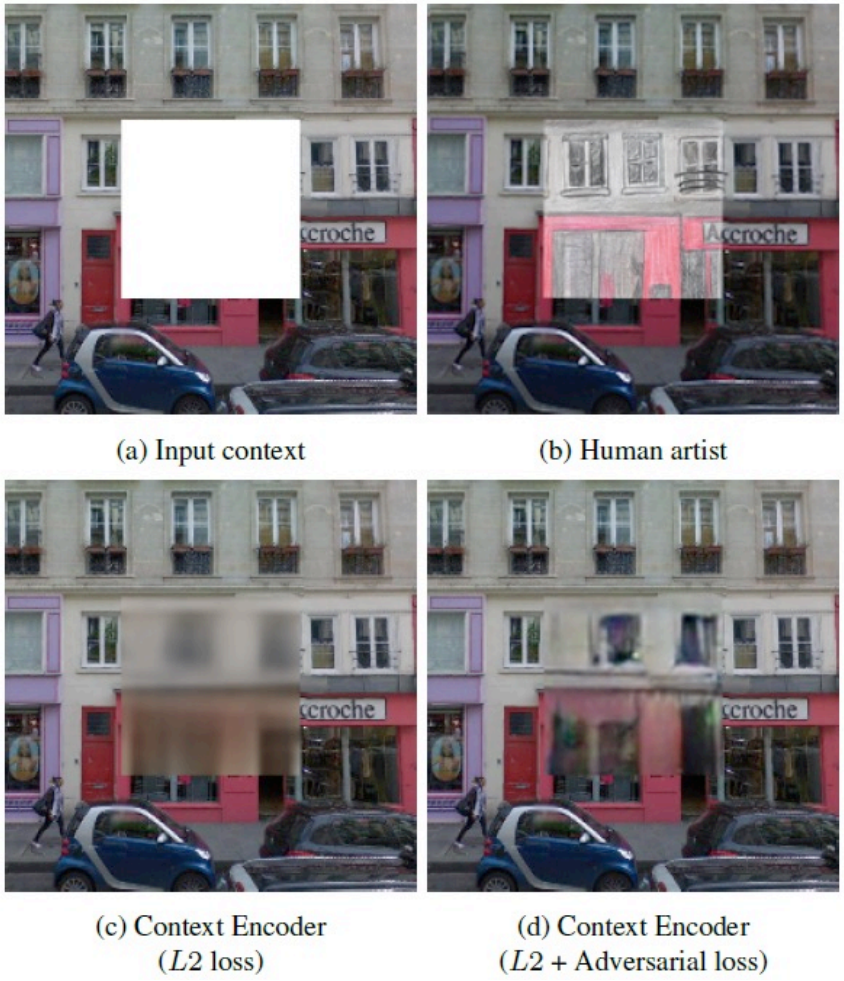
1. Meaningful losses for vision are more difficult to design.
2. Choice of region to mask out is important

Self-supervised learning in computer vision

- **Feature learning by Inpainting** (Pathak et al., '16)

L2:

$$\|I - \hat{I}\|_F^2$$



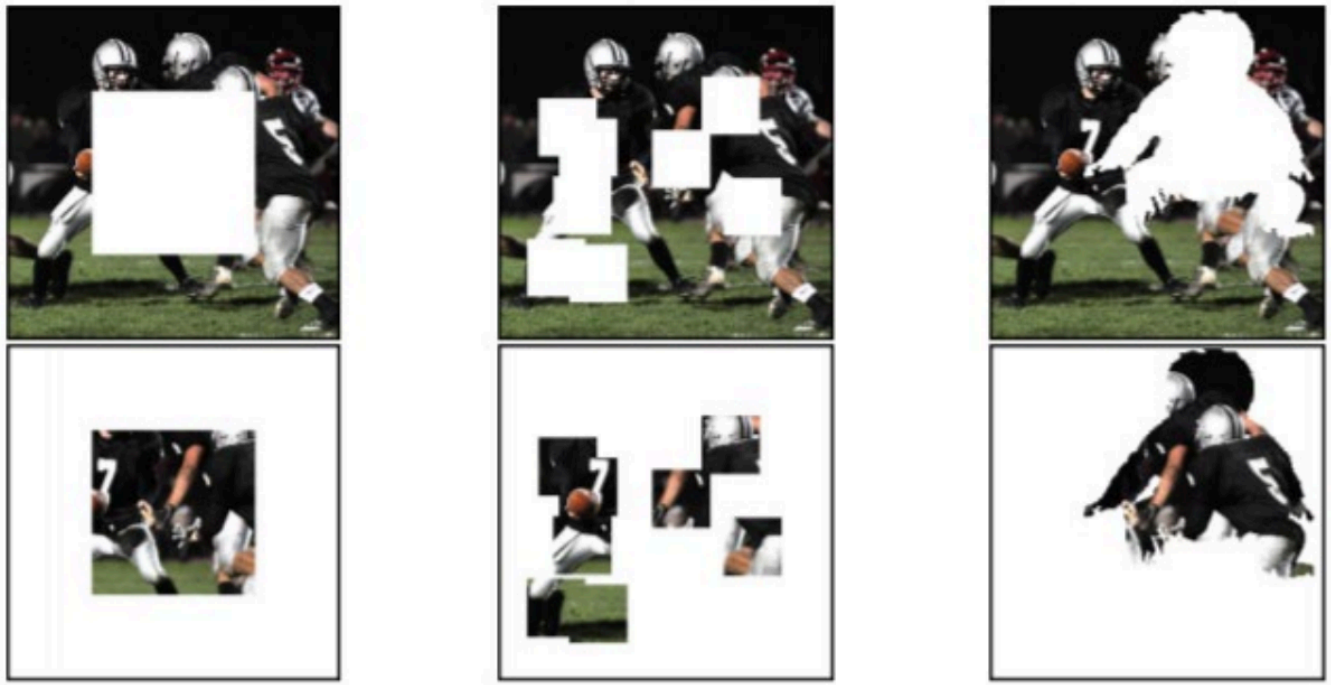
(a) Input context (b) Human artist

(c) Context Encoder ($L2$ loss) (d) Context Encoder ($L2$ + Adversarial loss)

L_2 vs. Adversarial loss from GAN

Self-supervised learning in computer vision

- **Feature learning by Inpainting** (Pathak et al., ’16)



(a) Central region (b) Random block (c) Random region

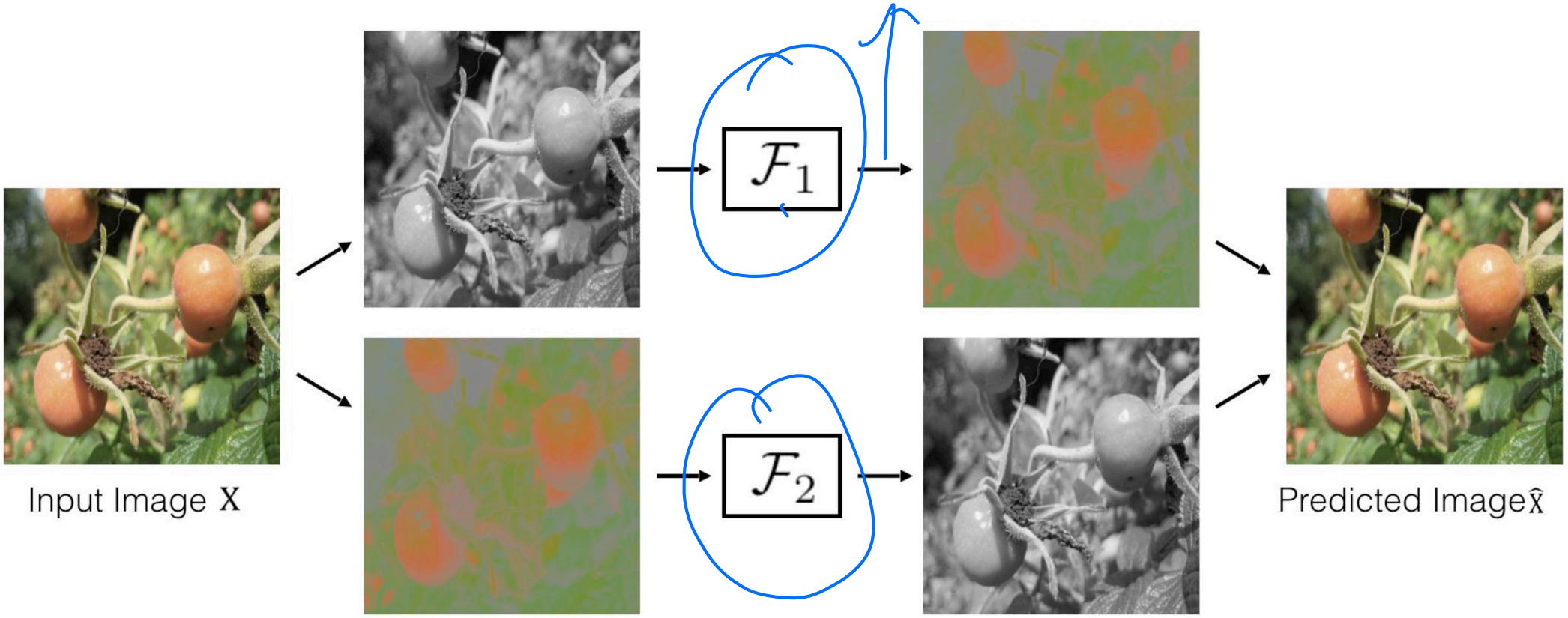
Figure 3: An example of image x with our different region masks $\hat{M}$ applied, as described in Section 3.3.

Fixed region vs. random square block vs. random region

Self-supervised learning in computer vision

- **Image Colorization** (Zhang et al. '16)

Encoder - decoder



Self-supervised learning in computer vision

- **Rotation Prediction** (Gidaris et al., ’18)

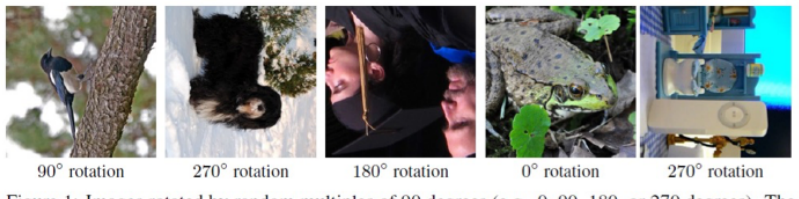
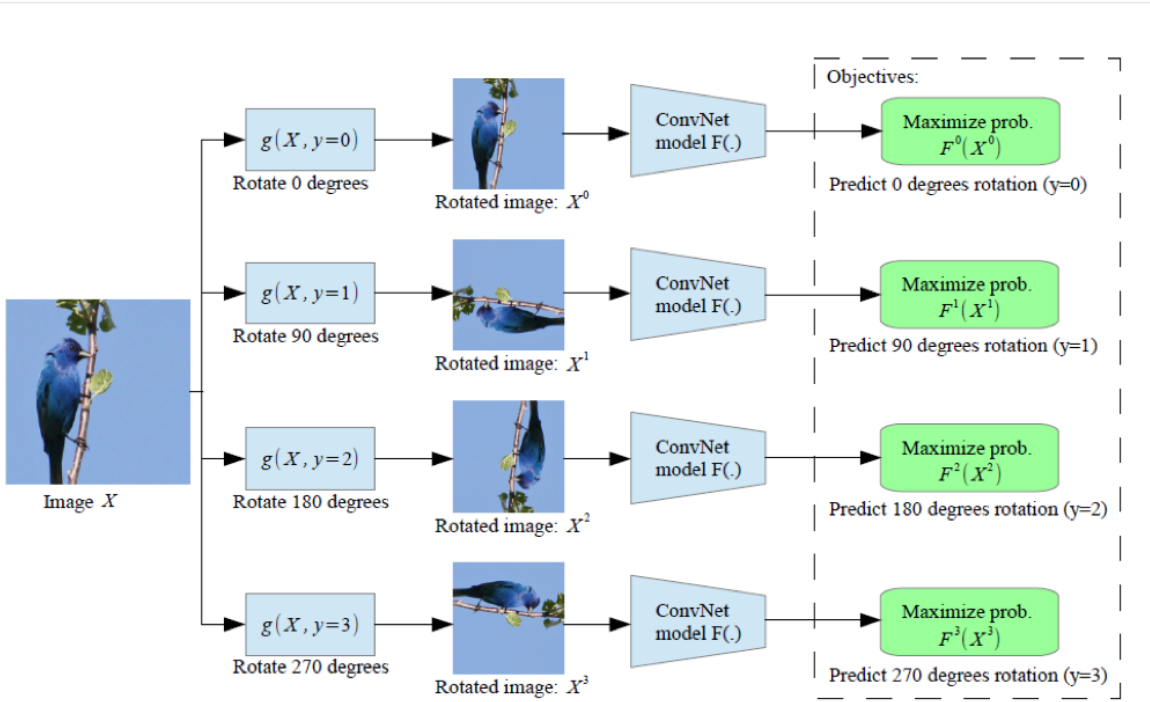


Figure 1: Images rotated by random multiples of 90 degrees (e.g., 0, 90, 180, or 270 degrees). The core intuition of our self-supervised feature learning approach is that if someone is not aware of the concepts of the objects depicted in the images, he cannot recognize the rotation that was applied to them.

(a) **Attention maps of supervised model** (b) **Attention maps of our self-supervised model**



Contrastive learning

representation

Idea: if features are "semantically" relevant, a "distortion" of an image should produce similar features.

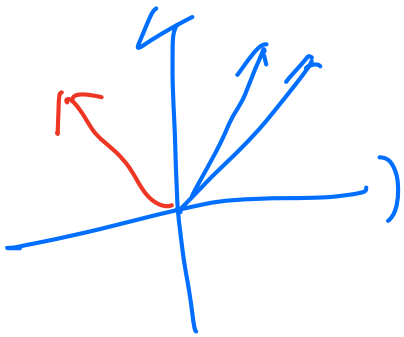
Framework:

- For every training sample, produce multiple *augmented* samples by applying various transformations.
- Train an encoder ***E*** to predict whether two samples are augmentations of the same base sample.
- A common way is train $\langle E(x), E(x') \rangle$ big if x, x' are two augmentations of the same sample:

$\|E(x)\|_2 = 1$

$$\ell_{x,x'} = -\log\left(\frac{\exp(\tau\langle E(x), E(x')\rangle)}{\sum_{\tilde{x}} \exp(\tau\langle E(x), E(\tilde{x})\rangle)}\right)$$

$$\min \sum_{x,x' \text{ augments of each other}} \ell_{x,x'}$$



τ: temperature

Contrastive learning

Contrastive Predictive Coding (Van den Oord et al., ’18)

- CPC: Original proposed on audio data
- Use context to predict futures
  - Random negative samples required

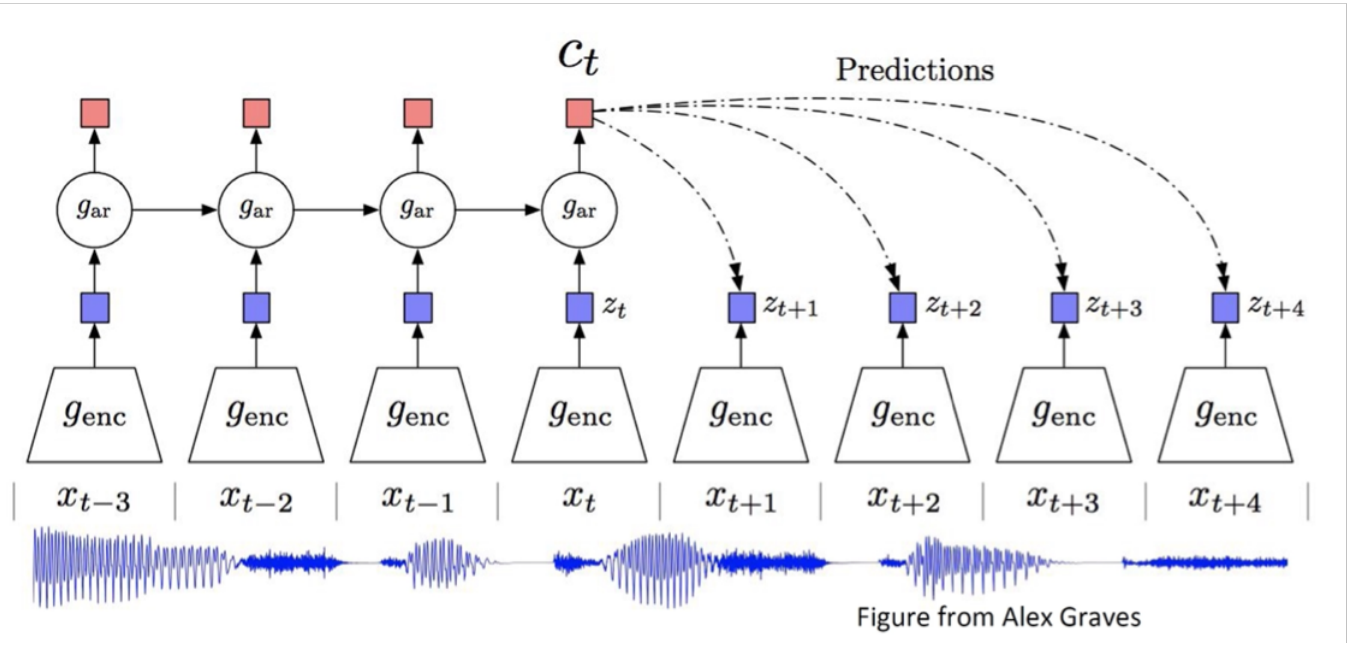


Figure from Alex Graves

$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

$$\mathcal{L}_N = -\mathop{\mathbb{E}}_X\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right]$$

Contrastive learning

Contrastive Predictive Coding (Van den Oord et al., ’18)

- CPC: Original proposed on audio data
- Use context to predict futures
  - Random negative samples required

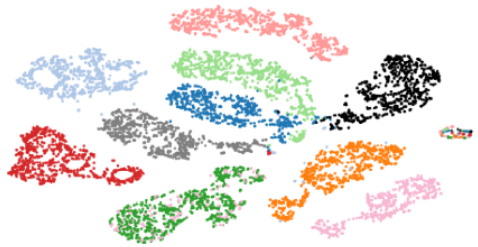


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

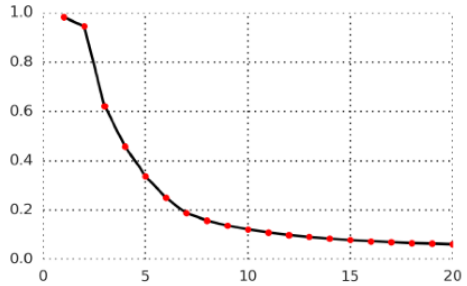


Figure 3: Average accuracy of predicting the positive sample in the contrastive loss for 1 to 20 latent steps in the future of a speech waveform. The model predicts up to 200ms in the future as every step consists of 10ms of audio.

Method	ACC
Phone classification	
Random initialization	27.6
MFCC features	39.7
CPC	64.6
Supervised	74.6
Speaker classification	
Random initialization	1.87
MFCC features	17.6
CPC	97.4
Supervised	98.5

Table 1: LibriSpeech phone and speaker classification results. For phone classification there are 41 possible classes and for speaker classification 251. All models used the same architecture and the same audio input sizes.

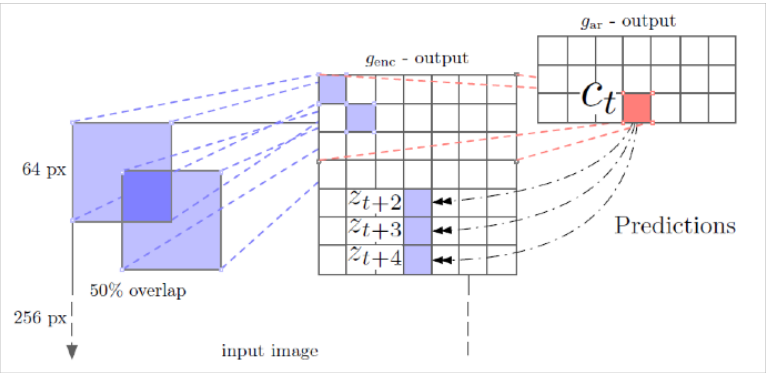
Method	ACC
#steps predicted	
2 steps	28.5
4 steps	57.6
8 steps	63.6
12 steps	64.6
16 steps	63.8
Negative samples from	
Mixed speaker	64.6
Same speaker	65.5
Mixed speaker (excl.)	57.3
Same speaker (excl.)	64.6
Current sequence only	65.2

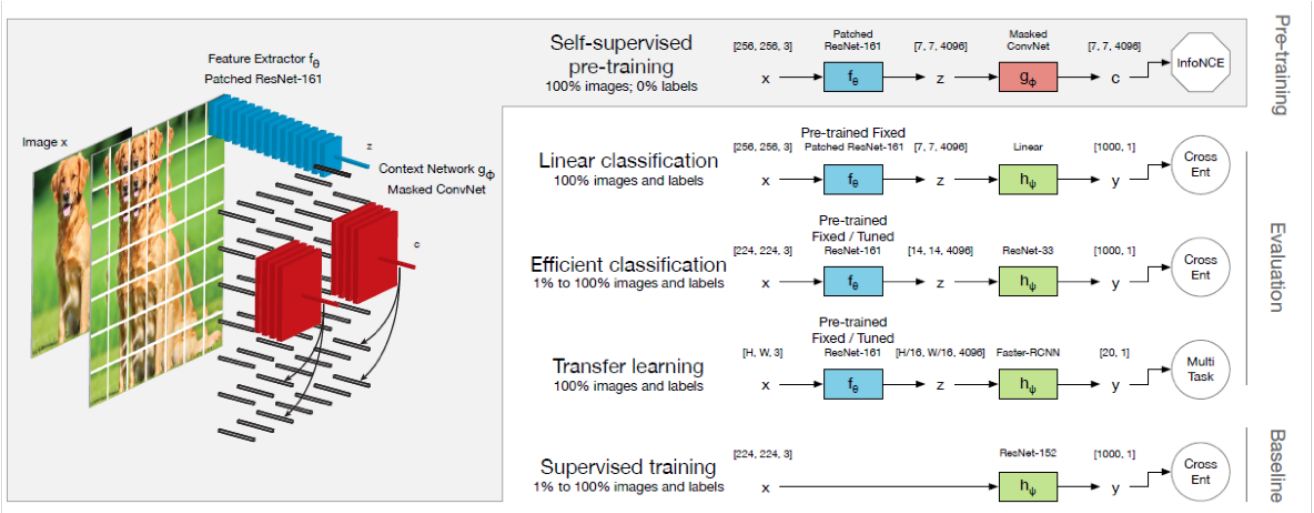
Table 2: LibriSpeech phone classification ablation experiments. More details can be found in Section 3.1.

Contrastive learning

Contrastive Predictive Coding (Van den Oord et al., ’18)

- CPCv2: improved version of CPC on images with large scale training
  - PixelCNN, more prediction directions, path augmentation, layer normalization





Contrastive learning

Contrastive Predictive Coding (Van den Oord et al., ’18)

- CPCv2: improved version of CPC on images with large scale training
  - PixelCNN, more prediction directions, path augmentation, layer normalization

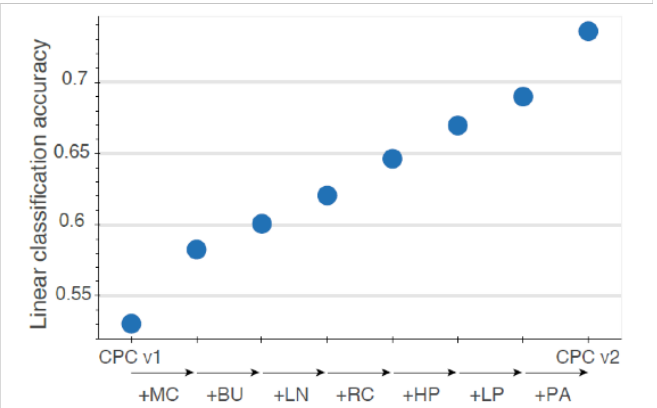
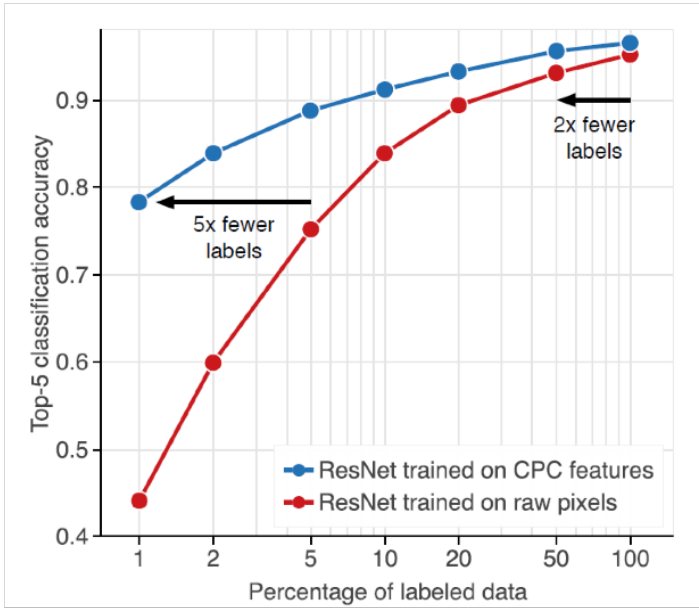


Figure 3. Linear classification performance of new variants of CPC, which incrementally add a series of modifications. MC: model capacity. BU: bottom-up spatial predictions. LN: layer normalization. RC: random color-dropping. HP: horizontal spatial predictions. LP: larger patches. PA: further patch-based augmentation. Note that these accuracies are evaluated on a custom validation set and are therefore not directly comparable to the results we report on the official validation set.

METHOD	PARAMS (M)	TOP-1	TOP-5
Methods using ResNet-50:			
INSTANCE DISCR. [1]	24	54.0	-
LOCAL AGGR. [2]	24	58.8	-
MOCO [3]	24	60.6	-
PIRL [4]	24	63.6	-
CPC V2 - RESNET-50	24	**63.8**	**85.3**
Methods using different architectures:			
MULTI-TASK [5]	28	-	69.3
ROTATION [6]	86	55.4	-
CPC V1 [7]	28	48.7	73.6
BIGBIGAN [8]	86	61.3	81.9
AMDIM [9]	626	68.1	-
CMC [10]	188	68.4	88.2
MOCO [2]	375	68.6	-
CPC V2 - RESNET-161	305	**71.5**	**90.1**



Contrastive learning

Contrastive Predictive Coding (Van den Oord et al., ’18)

- MoCo: Momentum Contrastive Learning (He et al., ’20)

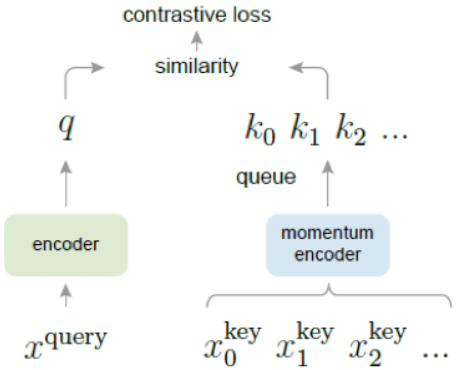
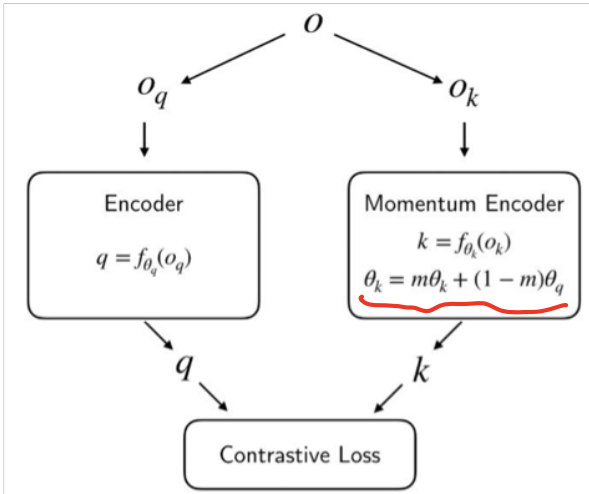


Figure 1. Momentum Contrast (MoCo) trains a visual representation encoder by matching an encoded query q to a dictionary of encoded keys using a contrastive loss. The dictionary keys $\{k_0, k_1, k_2, ...\}$ are defined on-the-fly by a set of data samples. The dictionary is built as a queue, with the current mini-batch enqueued and the oldest mini-batch dequeued, decoupling it from the mini-batch size. The keys are encoded by a slowly progressing encoder, driven by a momentum update with the query encoder. This method enables a large and consistent dictionary for learning visual representations.



$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+/\tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i/\tau)}$$

Contrastive learning

Contrastive Predictive Coding (Van den Oord et al., ’18)

- MoCo: Momentum Contrastive Learning (He et al., ’20)
  - Why momentum encoder?
    - Enable large and consistent buffer of negative samples
    - Ensure the encoding in buffer moves slowly via momentum
      - Which further ensures the feature extractor updates smoothly

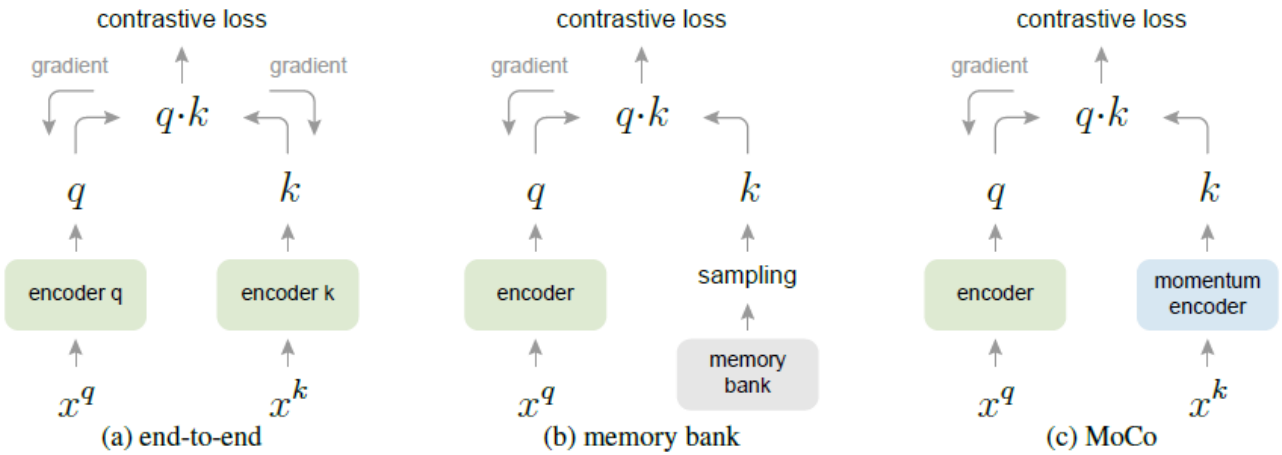


Figure 2. **Conceptual comparison of three contrastive loss mechanisms** (empirical comparisons are in Figure 3 and Table 3). Here we illustrate one pair of query and key. The three mechanisms differ in how the keys are maintained and how the key encoder is updated. **(a)**: The encoders for computing the query and key representations are updated ***end-to-end*** by back-propagation (the two encoders can be different). **(b)**: The key representations are sampled from a ***memory bank*** [61]. **(c)**: ***MoCo*** encodes the new keys on-the-fly by a momentum-updated encoder, and maintains a queue (not illustrated in this figure) of keys.

Contrastive learning

Contrastive Predictive Coding (Van den Oord et al., ’18)

- MoCo: Momentum Contrastive Learning (He et al., ’20)

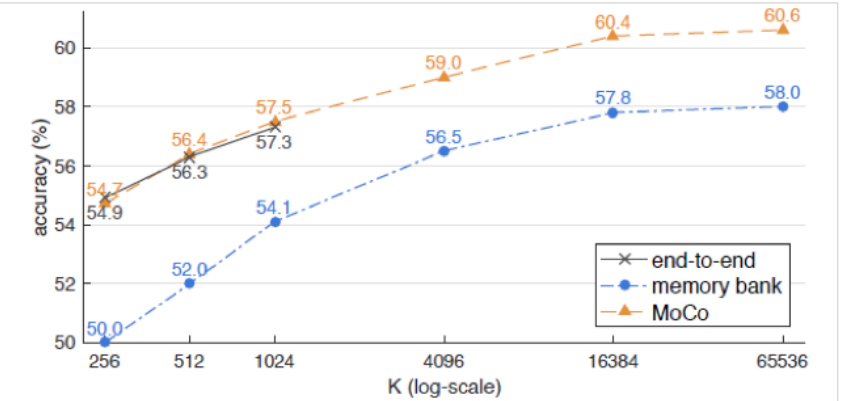
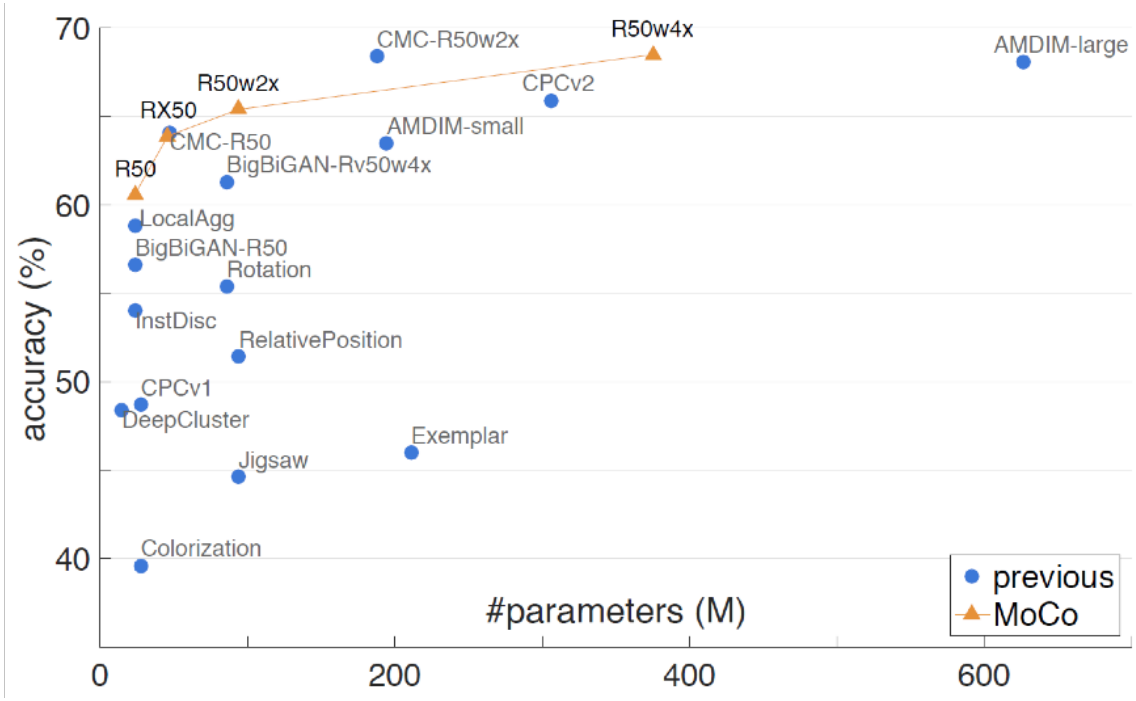


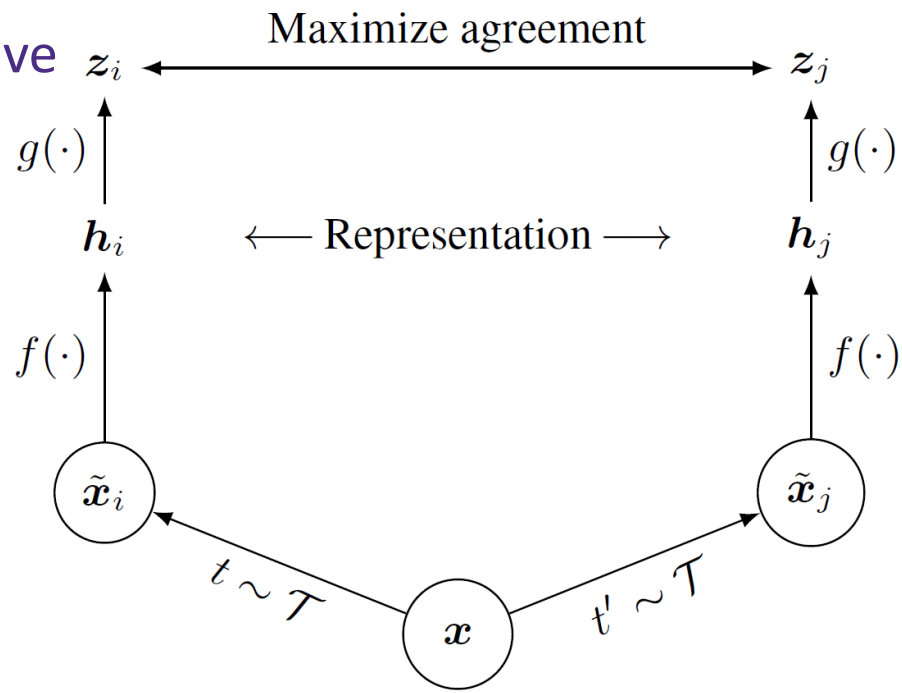
Figure 3. **Comparison of three contrastive loss mechanisms** under the ImageNet linear classification protocol. We adopt the same pretext task (Sec. 3.3) and only vary the contrastive loss mechanism (Figure 2). The number of negatives is K in memory bank and MoCo, and is $K-1$ in end-to-end (offset by one because the positive key is in the same mini-batch). The network is ResNet-50.



Contrastive learning

Contrastive Predictive Coding (Van den Oord et al., '18)

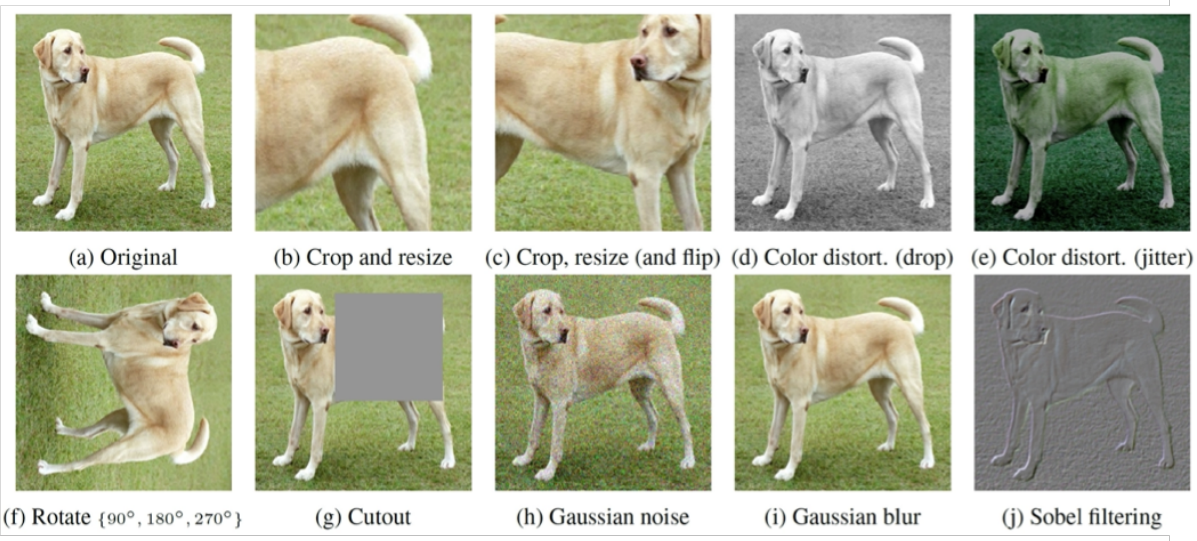
- SimCLR (Chen et al. '20)
  - A simple framework for contrastive learning of visual representations
    - Predefine a set of transformations
    - For a data, sample two transformations
    - Maximum agreement on representations
  - No negative pairs explicitly
    - Non-paired data in the batch are negative



Contrastive learning

Contrastive Predictive Coding (Van den Oord et al., '18)

- SimCLR (Chen et al. '20)



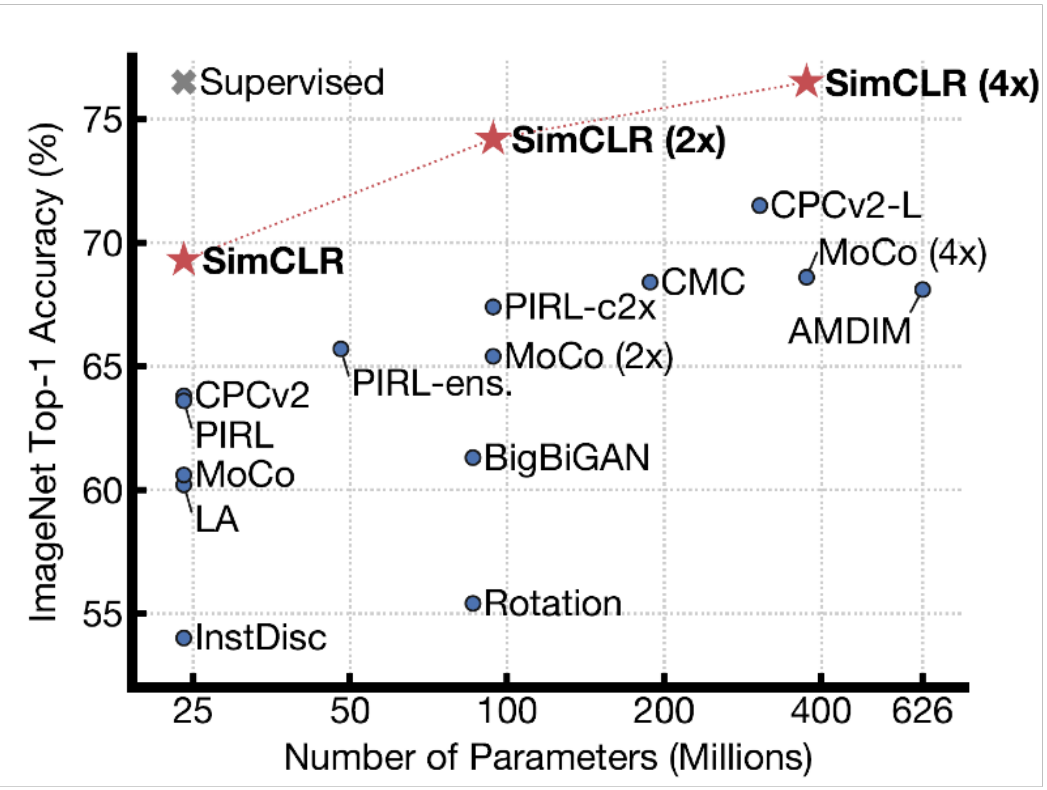
Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of f, g, $\mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

Contrastive learning

Contrastive Predictive Coding (Van den Oord et al., '18)

- SimCLR (Chen et al. '20)



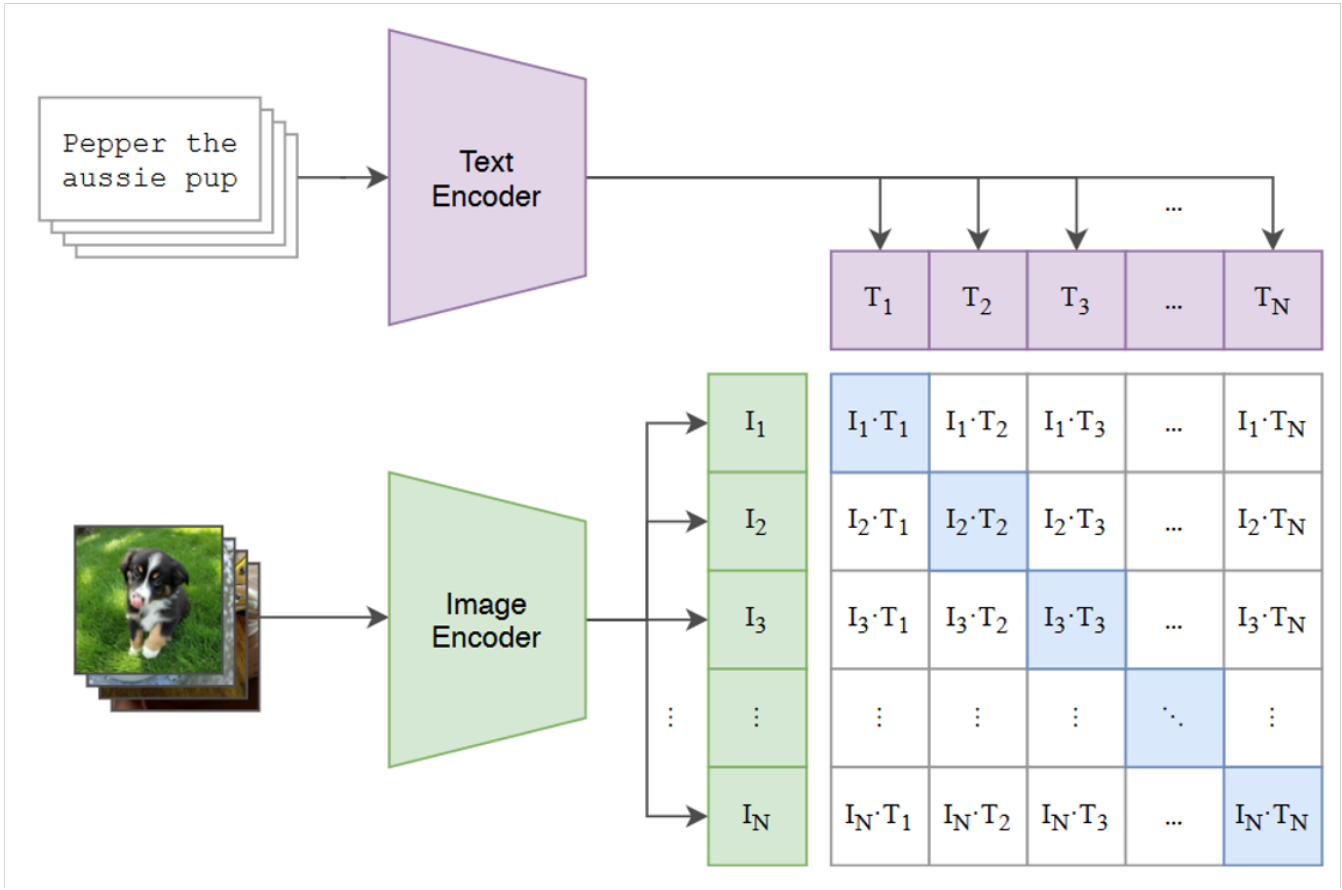
Method	Architecture	Label fraction 1%	Label fraction 10%
		Top 5	Top 5
Supervised baseline	ResNet-50	48.4	80.4
Methods using other label-propagation:			
Pseudo-label	ResNet-50	51.6	82.4
VAT+Entropy Min.	ResNet-50	47.0	83.4
UDA (w. RandAug)	ResNet-50	-	88.5
FixMatch (w. RandAug)	ResNet-50	-	89.1
S4L (Rot+VAT+En. M.)	ResNet-50 (4×)	-	91.2
Methods using representation learning only:			
InstDisc	ResNet-50	39.2	77.4
BigBiGAN	RevNet-50 (4×)	55.2	78.8
PIRL	ResNet-50	57.2	83.8
CPC v2	ResNet-161(*)	77.9	91.2
SimCLR (ours)	ResNet-50	75.5	87.8
SimCLR (ours)	ResNet-50 (2×)	83.0	91.2
SimCLR (ours)	ResNet-50 (4×)	**85.8**	**92.6**

Table 7. ImageNet accuracy of models trained with few labels.

Multimodal Contrastive Learning

CLIP

Contrastive Pretraining:
Train image and text representation together



Multimodal Contrastive Learning

(image, text)

Loss function

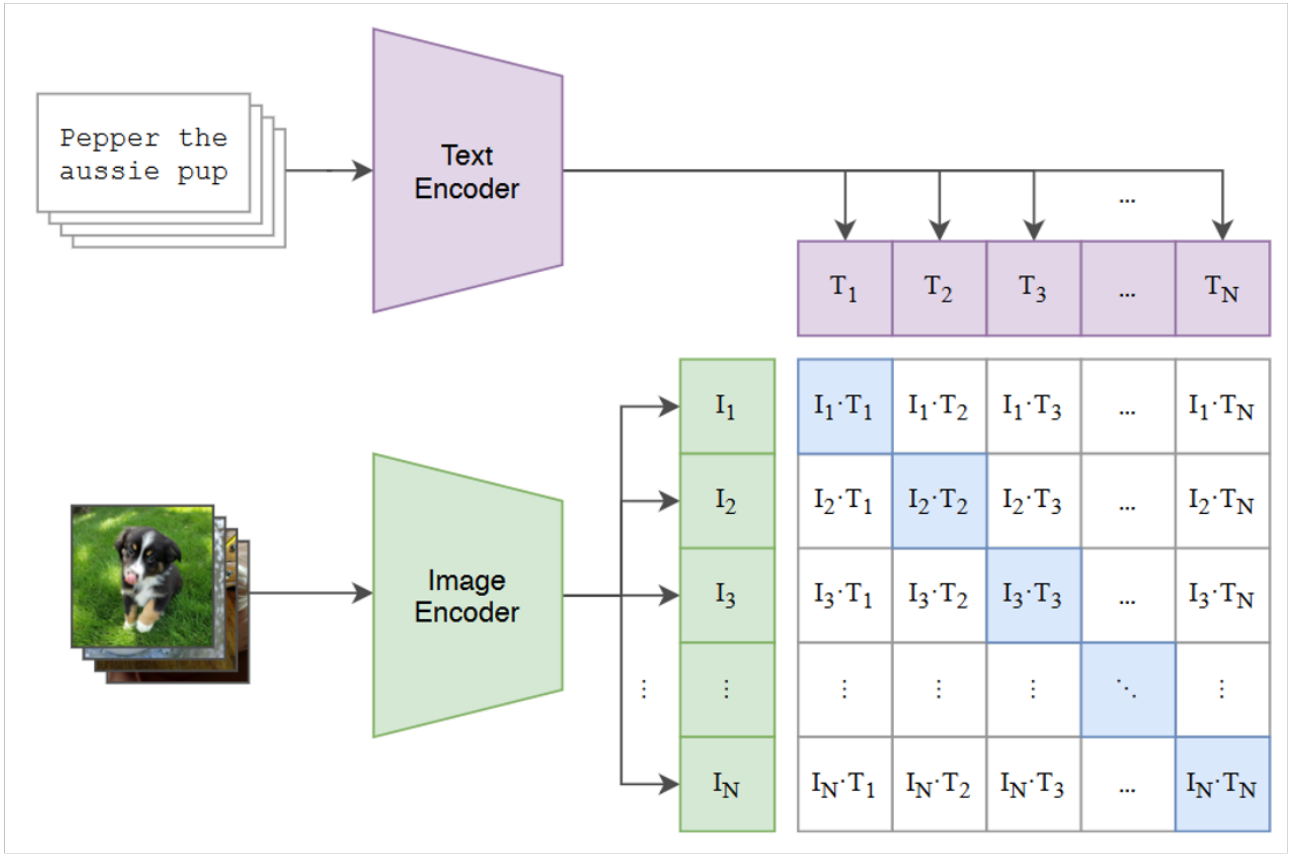
Let $q_{ij} := I_i^\top T_j$ (normalized embeddings: $||I_i||_2 = ||T_j||_2 = 1$),

$$\text{loss} = \frac{\text{loss}_I + \text{loss}_T}{2}$$

where,

$$\text{loss}_I = -\sum_{i=1}^{N} \log \frac{\exp(q_{ii})}{\sum_j \exp(q_{ij})}$$

$$\text{loss}_T = -\sum_{j=1}^{N} \log \frac{\exp(q_{jj})}{\sum_i \exp(q_{ij})}$$

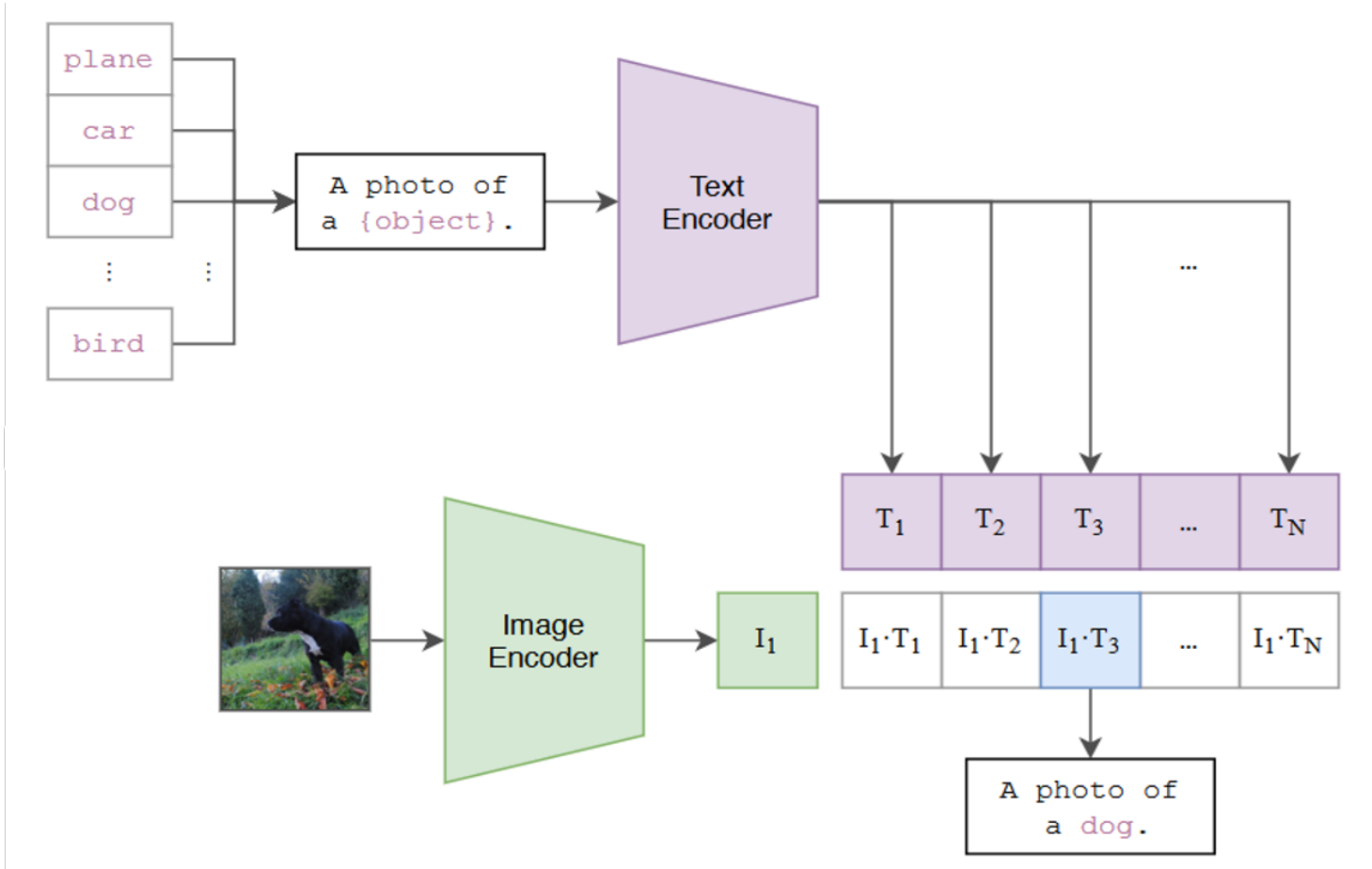


q_{ii} : inner product of same pair

Multimodal Contrastive Learning

Zero-Shot Classification:

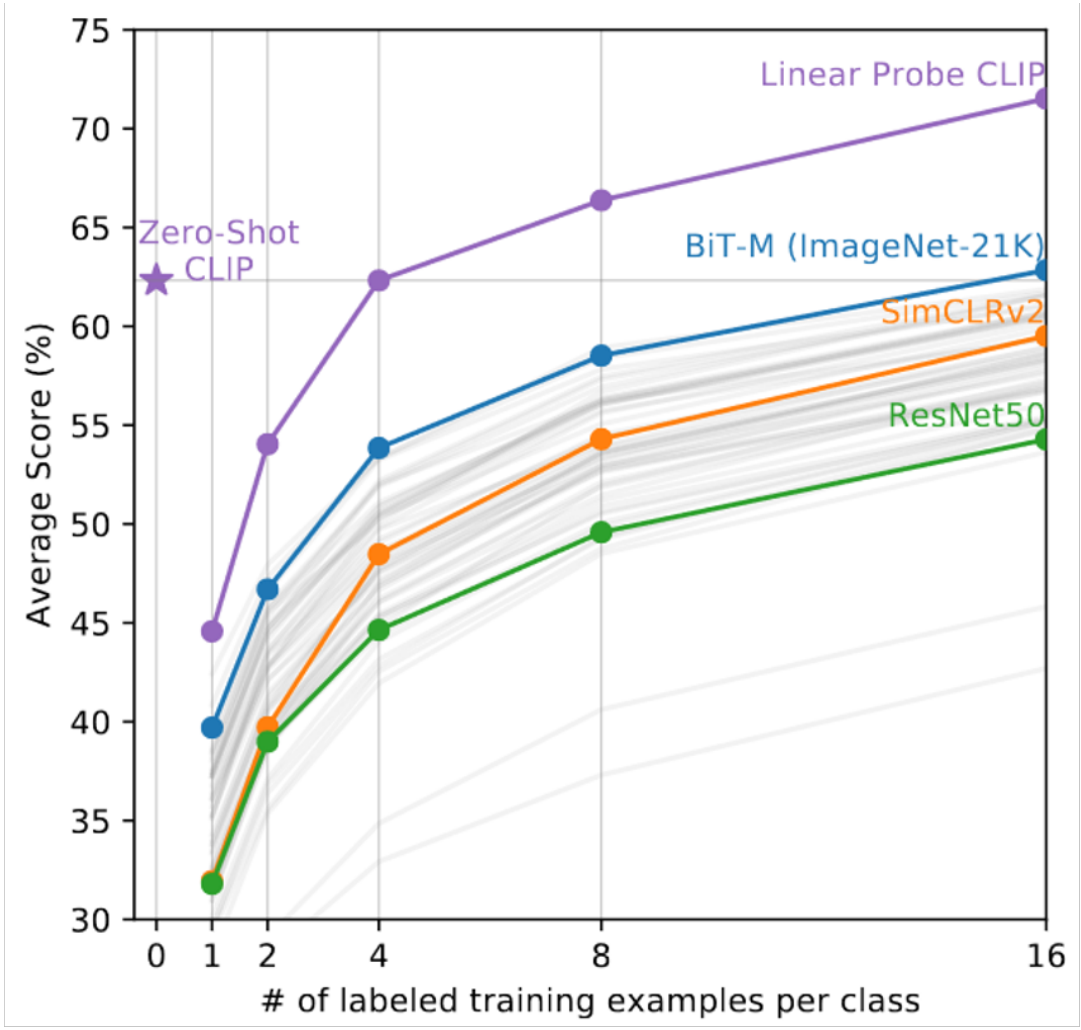
- Generate a prompt for each class



Multimodal Contrastive Learning

Results

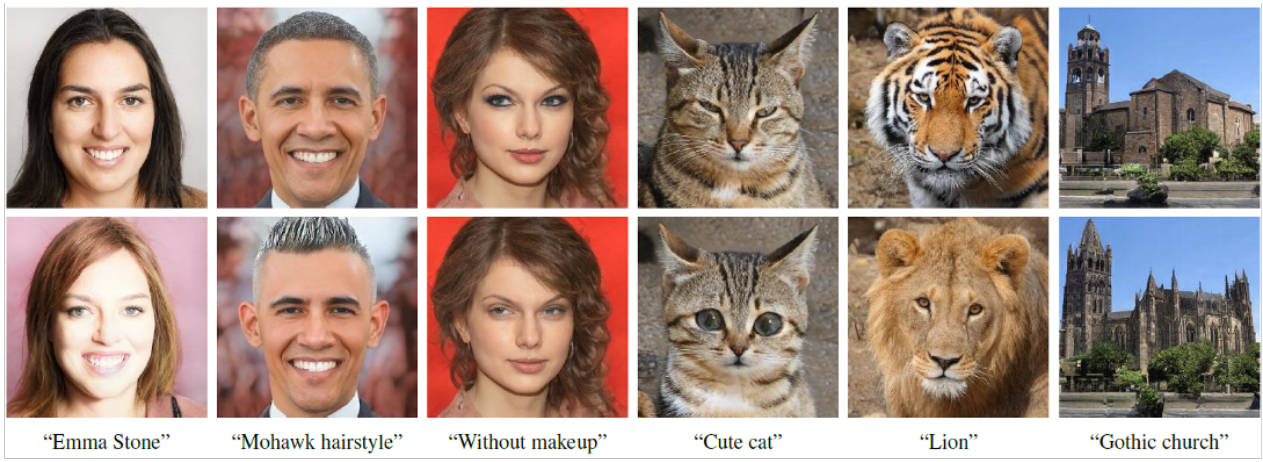
- Strong zero-shot and few-shot performance compared with other models.
- Zero-shot performance on **ImageNet**: CLIP ≈ fully supervised ResNet50!



Applications of CLIP

Image Generation

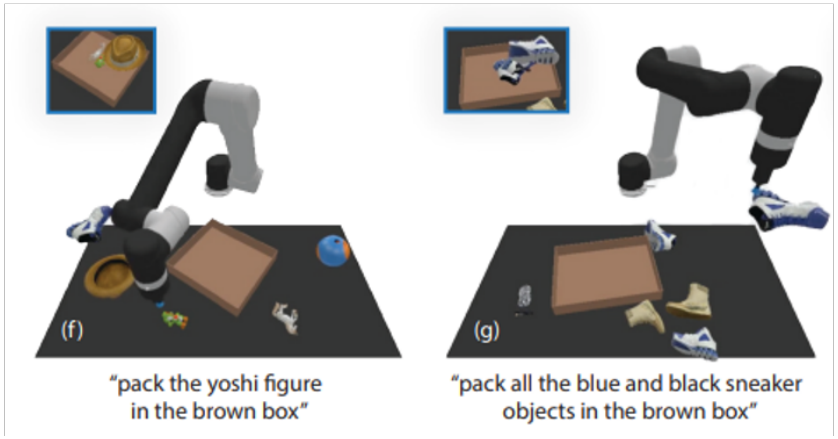
(StyleCLIP [Patashnik et al. 2021])



Robotics

(CLIPort [Shridhar et al. 2021])

...



Problems about Training CLIP

Require large amount of ***carefully curated*** image-text pairs
4 Billion closed-source data used for OpenAI's CLIP

Q: How to obtain lots of high-quality data?

One choice: Web-curated data pairs + data filtering

DataComp

(search query, image from search)

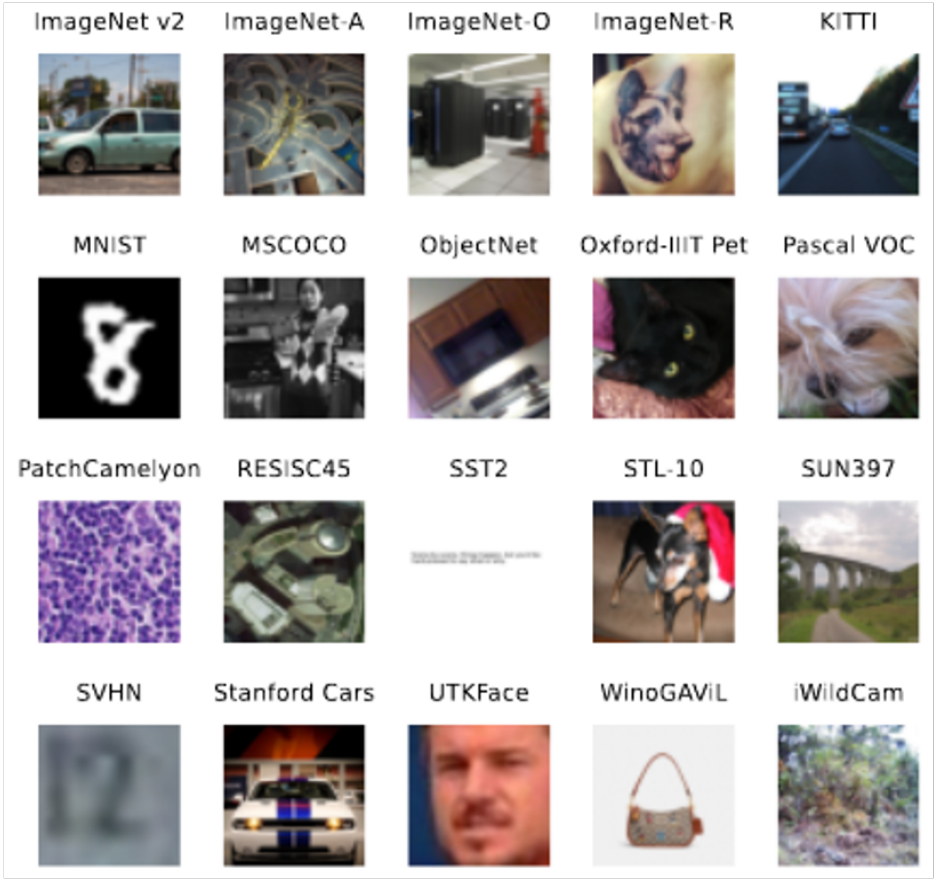
A benchmark standardize the training configuration

Training Process:

- Filtering data from a pool of ***low-quality*** data pairs
- Train a CLIP model with a **fixed architecture** and **hyperparameters**
- Fix **total number of training data seen** (1 pass of 4B data = 4 passes of 1B data)

Evaluation:

- 38 Zero-shot downstream tasks



Data Filtering

Distribution-agnostic methods

diversity

Image-based filtering

- Cluster the image embeddings (from a **pre-trained** CLIP model) of training data, and select the groups that contain at least one embedding from ImageNet-1k

use a teacher

CLIP score filtering

- Filter the data with low CLIP similarity assigned by a **pre-trained** CLIP model.

$$\text{CLIP score} = \bar{f}_{\text{image}}^{T} \bar{f}_{\text{text}}$$

Data Filtering

Setup:

Total number of training sample seen = 12.8M

Filtering Strategy	Dataset Size	ImageNet (1 sub-task)	ImageNet Dist. Shift (5)	VTAB (11)	Retrieval (3)	Average (38)
No filtering	12.8M	2.5	3.3	14.5	10.5	13.2
CLIP score (30%, reproduced)	3.8M	4.8	5.3	17.1	11.5	15.8
Image-based ∩ CLIP score (45%)	1.9M	4.2	4.6	17.4	10.8	15.5
$\mathbb{D}^2$ Pruning (image+text, reproduced)	3.8M	4.6	5.2	18.5	11.1	16.1
CLIP score (45%)	5.8M	4.5	5.1	17.9	**12.3**	16.1

Filtering significantly improves the performance!