Representation Learning
Pre-training

Example in image representation

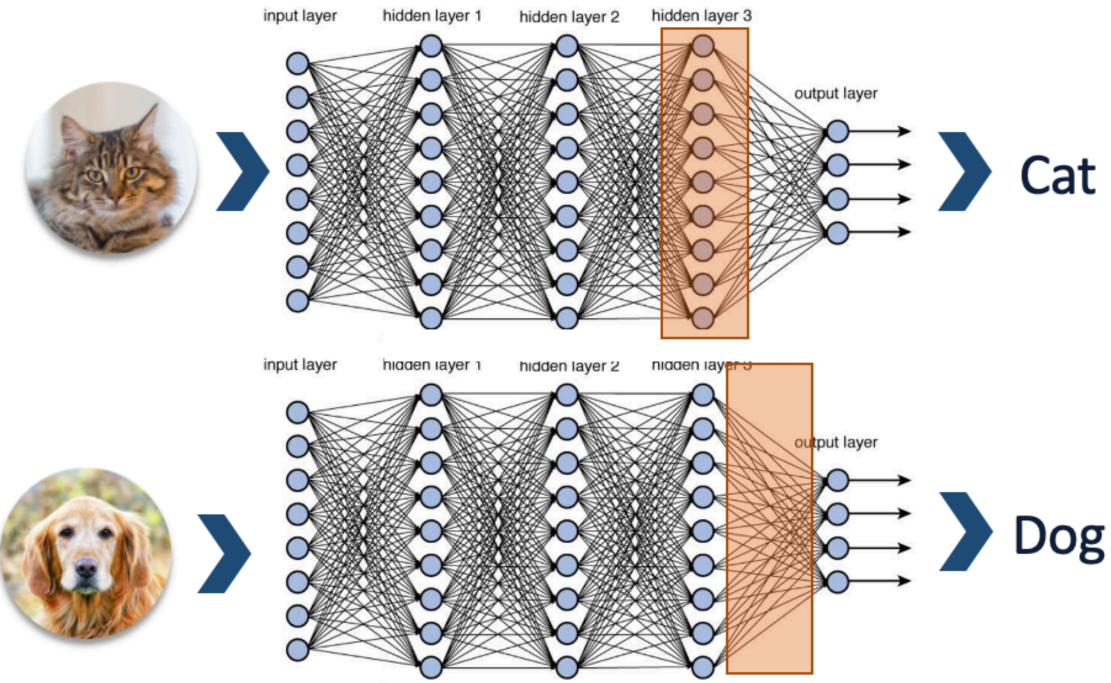


Image Representation

Cat

Dog

New linear classifier

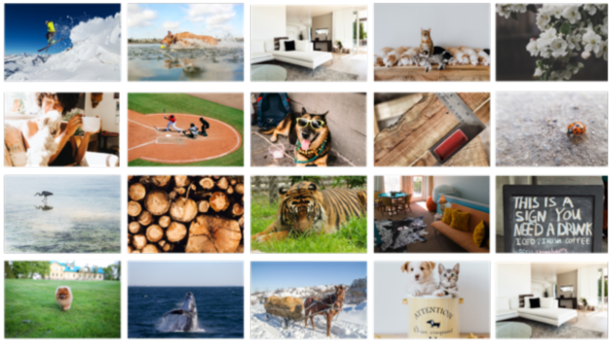
Train a neural network (ResNet) on ImageNet (1M data, 1000 classes)

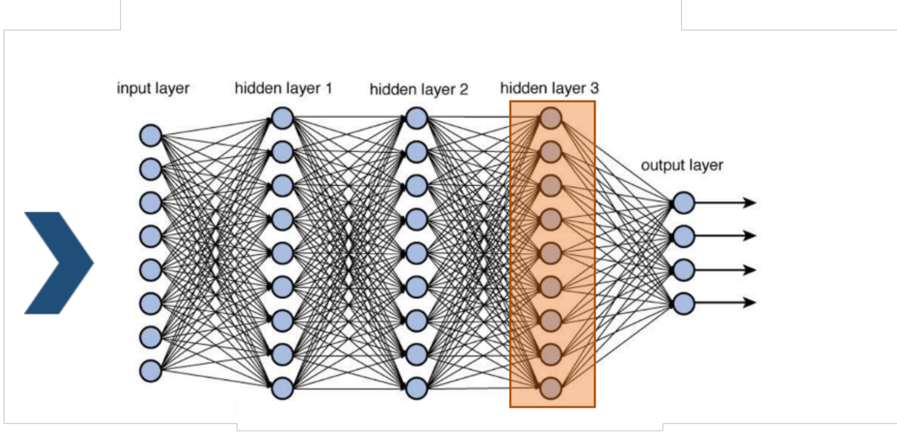
Representation (feature extractor): The mapping from image to the second-to-the-last layer.

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

Example in image representation

Source tasks (for training representation): ImageNet



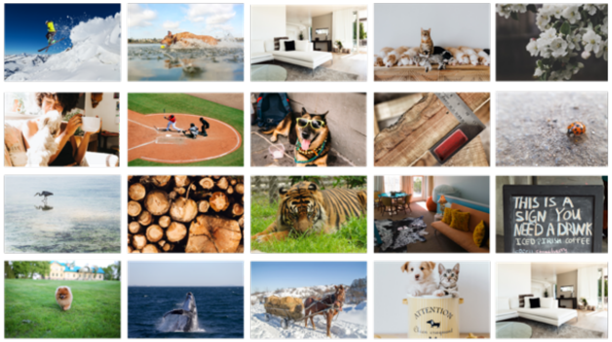


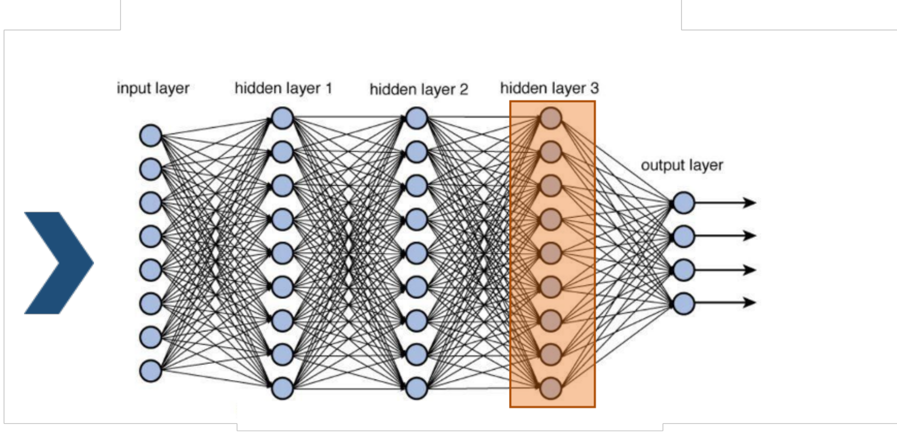
Target task: Few-shot Learning on VOC07 dataset (20 classes, 1-8 examples per class)

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

Example in image representation

Source tasks (for training representation): ImageNet





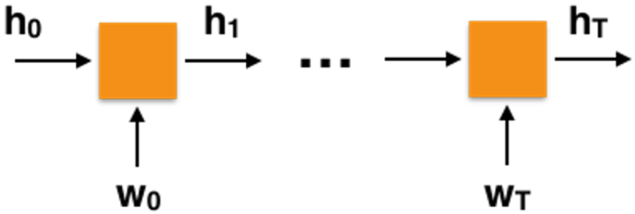
Target task: Few-shot Learning on VOC07 dataset (20 classes, 1-8 examples per class)

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

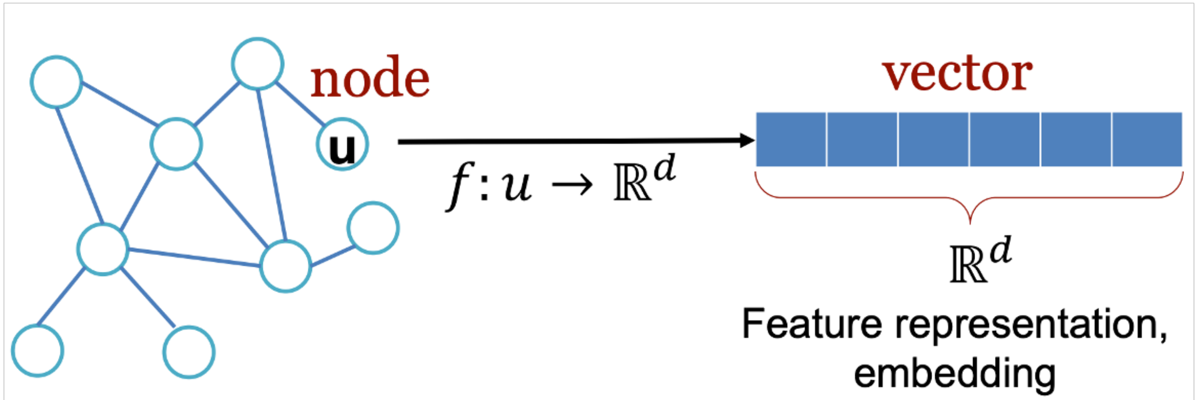
Examples

Natural Language Processing

Final hidden state: Sentence representation



Graph Representation Learning



Feature representation, embedding

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.

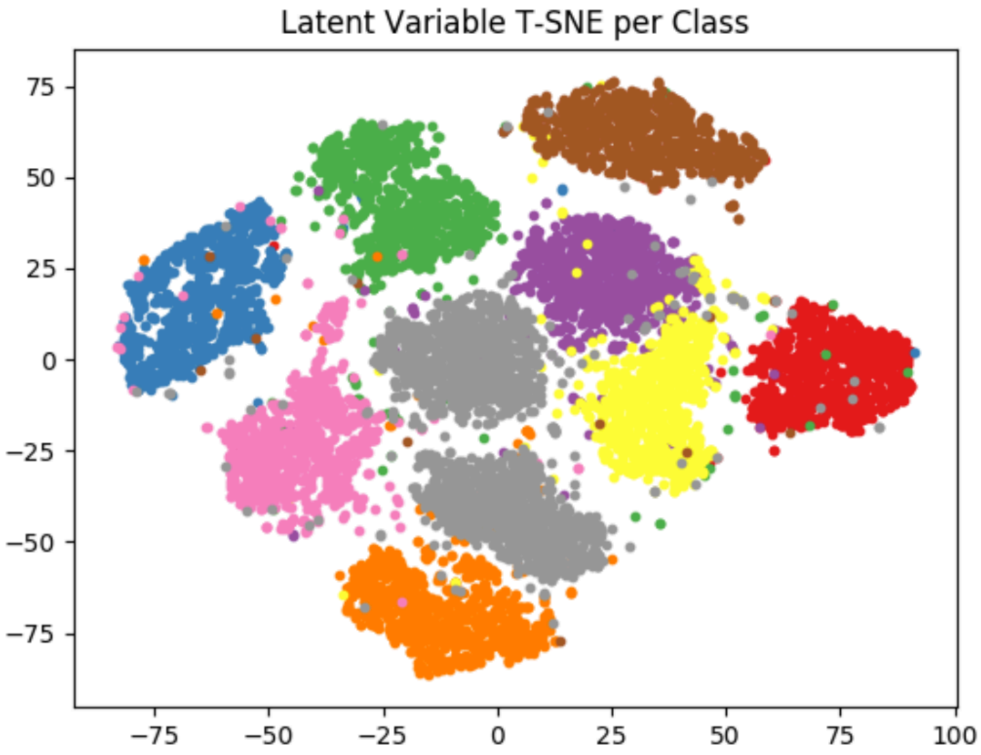
- **Semantic clusterability**: features of the same “semantic class” (e.g. images in the same class) are clustered together.

- **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.

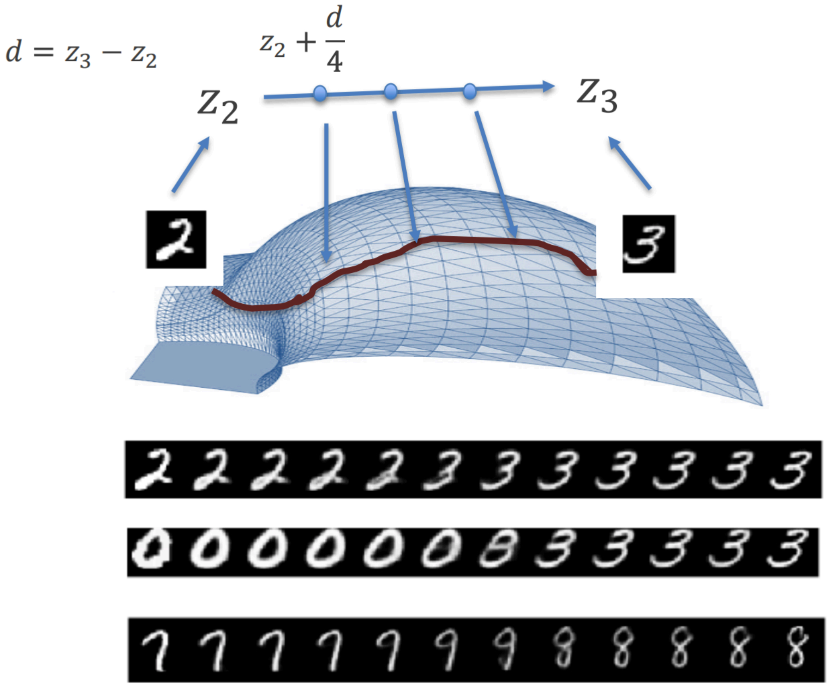


Intuition: If semantic classes are linearly separable, and labels on downstreams tasks depend linearly on semantic classes: we only need to learn a simple classifer.

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

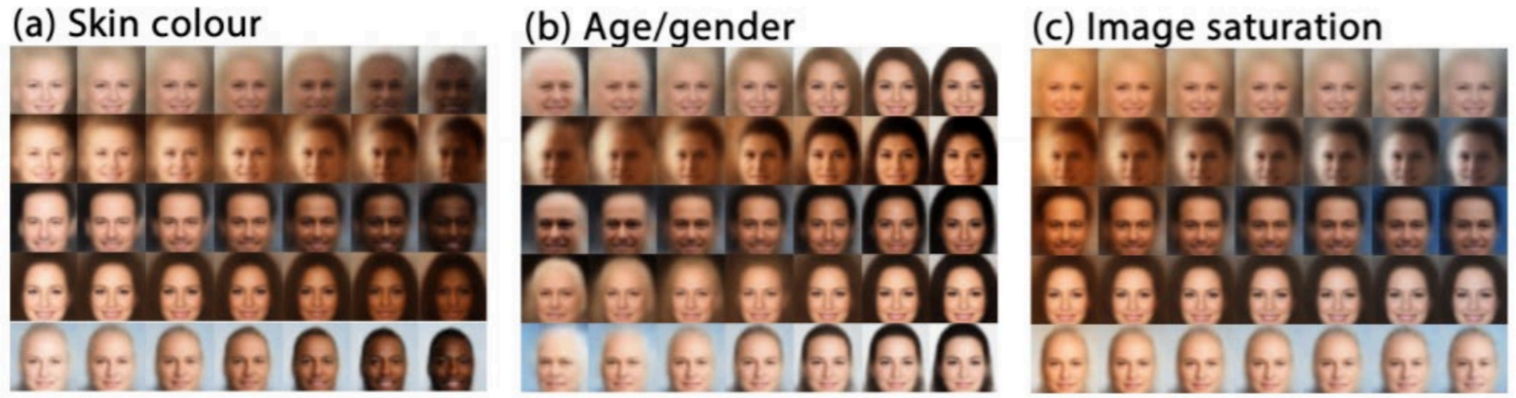
Linear interpolation: in the representation space, linear interpolations produce meaningful data points (latent space is convex).

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: $p_\theta(z) = \Pi_i p_\theta(z_i)$.



(a) Skin colour (b) Age/gender (c) Image saturation

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.

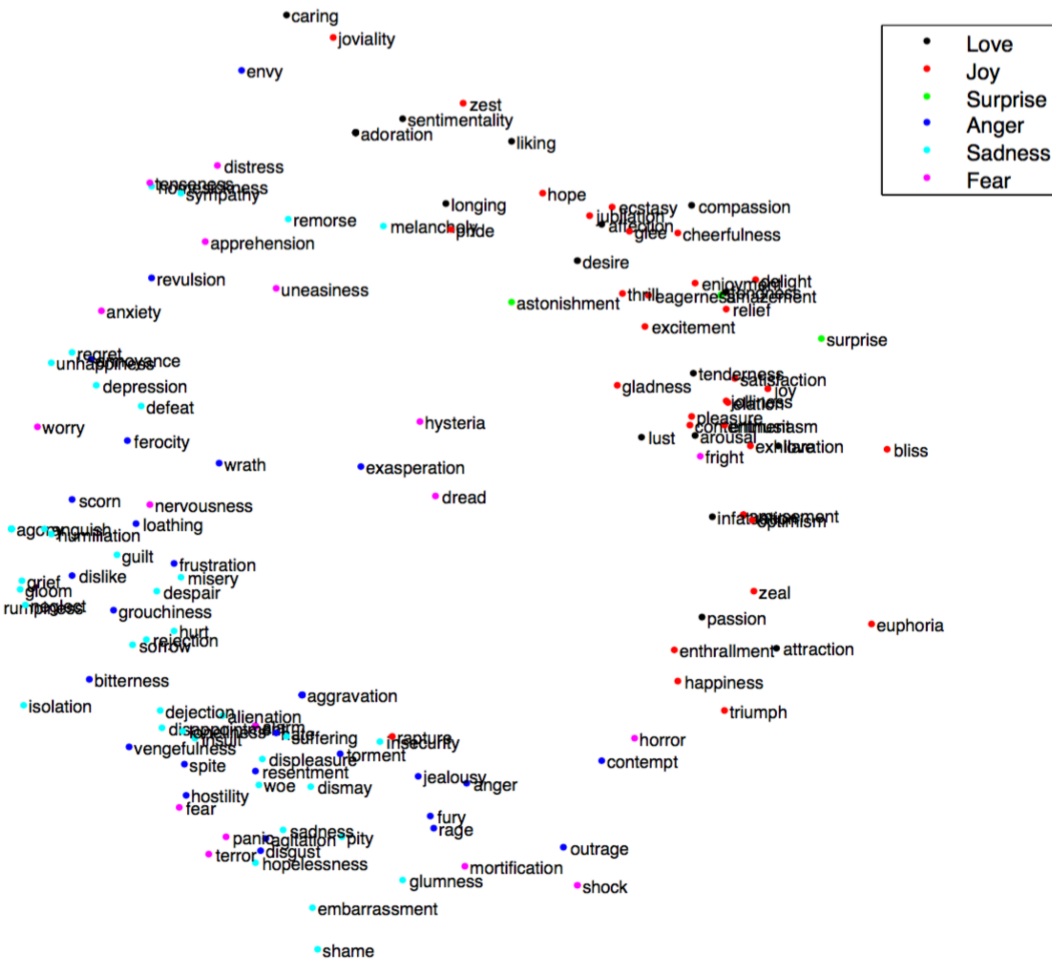
Representation Learning Methods

Word embeddings, word2vec

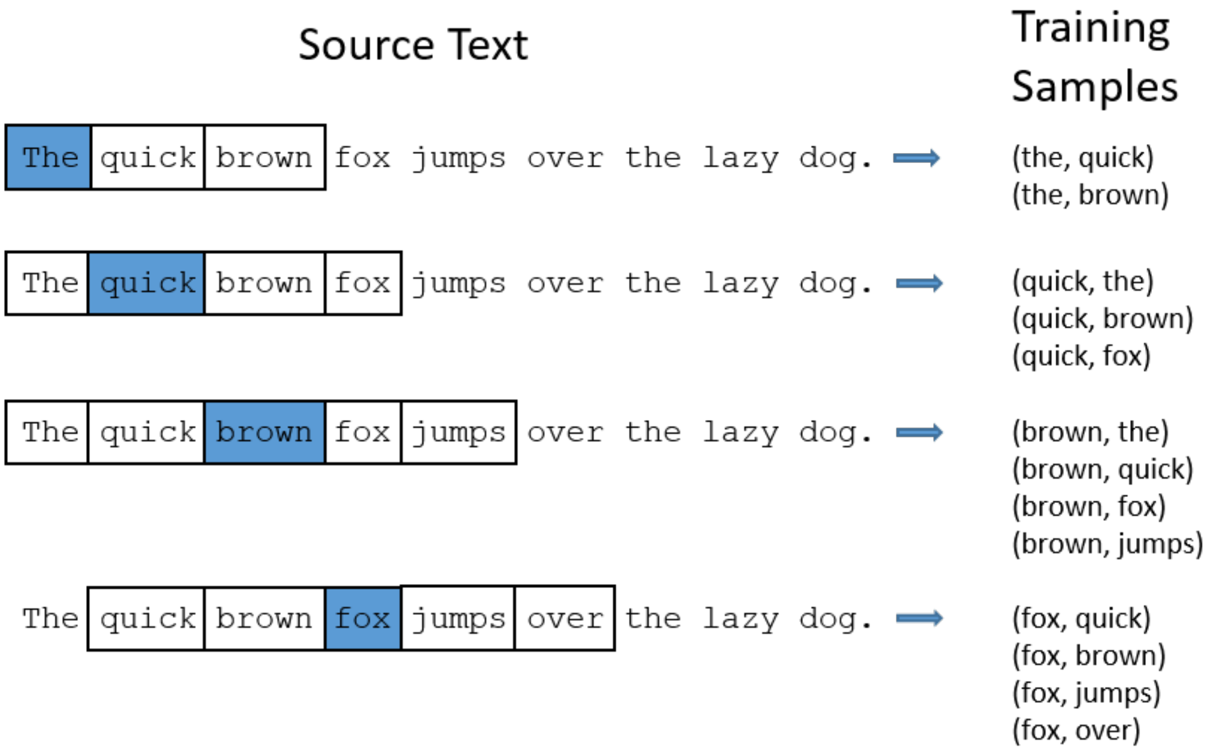
Can we **embed words** into a latent space?

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)



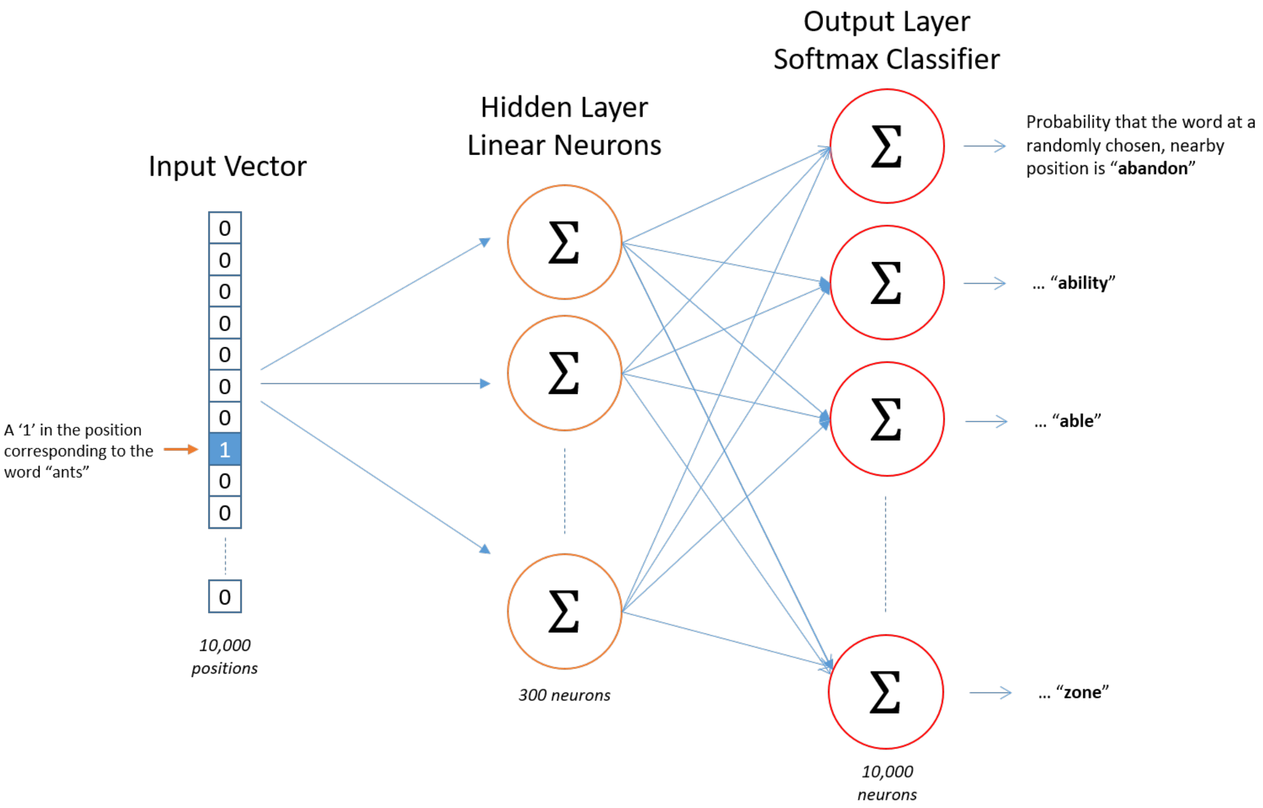
Word embeddings, word2vec



Source Text		Training Samples
The quick brown fox jumps over the lazy dog.	⟹	(the, quick) (the, brown)
The quick brown fox jumps over the lazy dog.	⟹	(quick, the) (quick, brown) (quick, fox)
The quick brown fox jumps over the lazy dog.	⟹	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown fox jumps over the lazy dog.	⟹	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

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

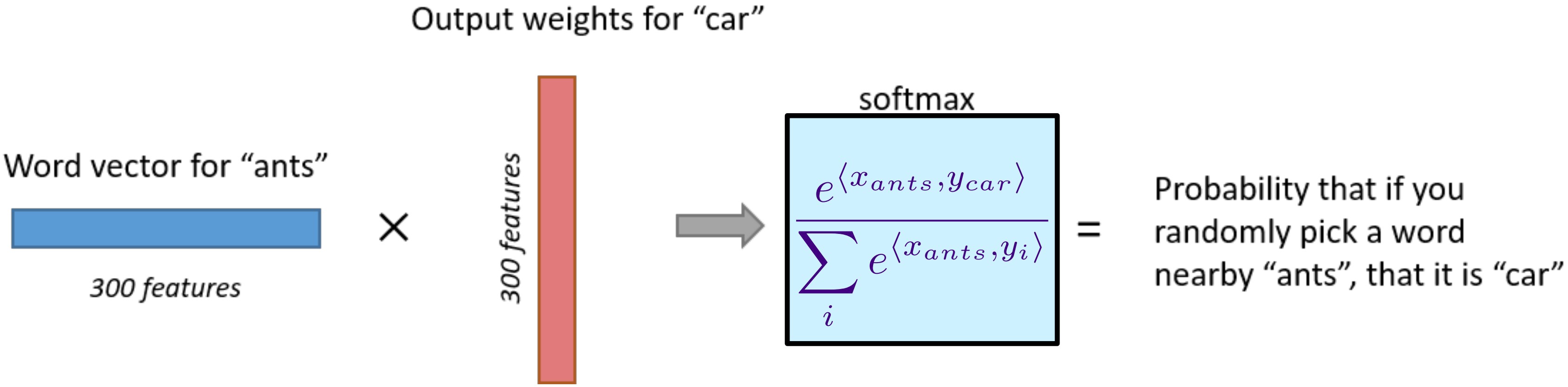
Word embeddings, word2vec



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



Output weights for “car”

Word vector for “ants”

300 features

×

300 features

softmax

$$\frac{e^{\langle x_{ants}, y_{car}\rangle}}{\sum_{i} e^{\langle x_{ants}, y_i\rangle}}$$

=

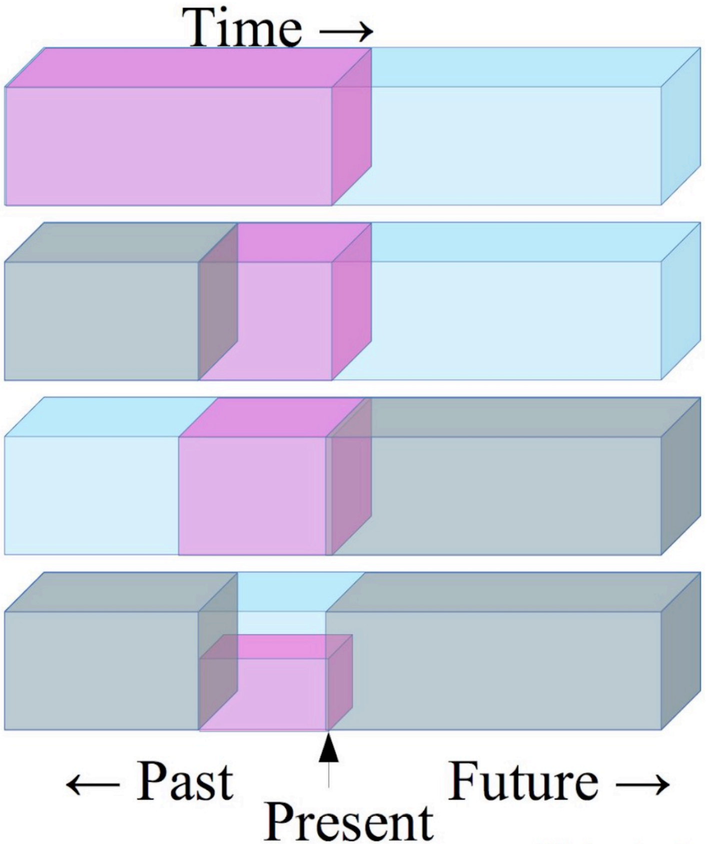
Probability that if you randomly pick a word nearby “ants”, that it is “car”

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

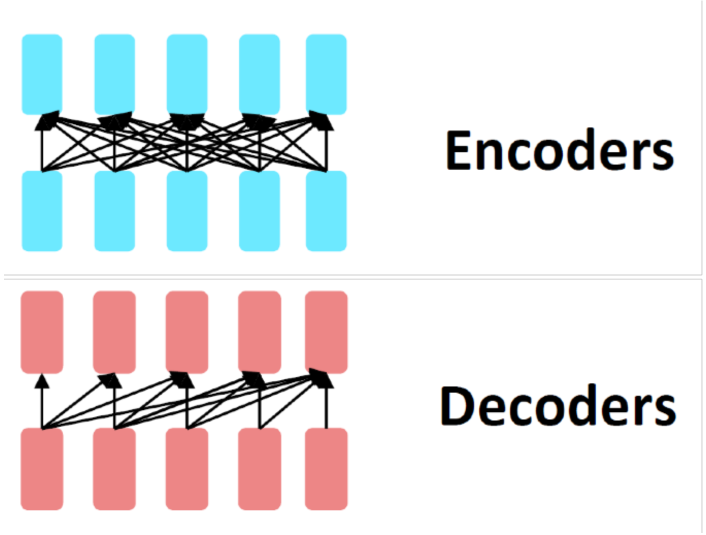
Transformer Pretraining

- Collect a large amount of corpus (wiki) and pretrain 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

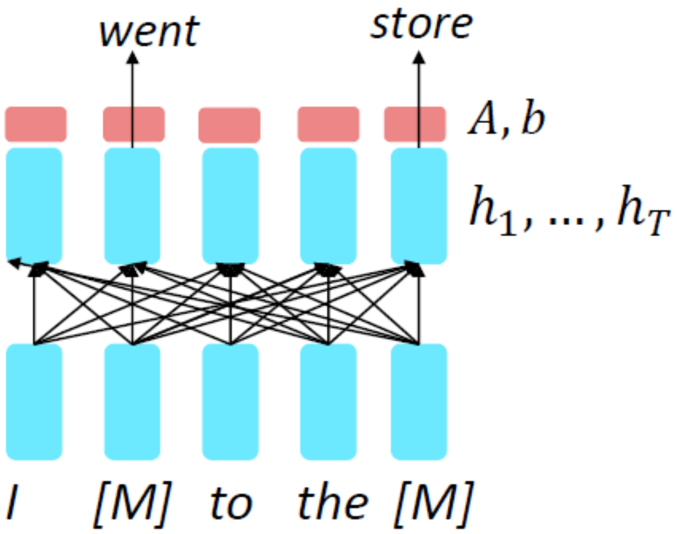


Pre-training Transformer Encoder

- 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

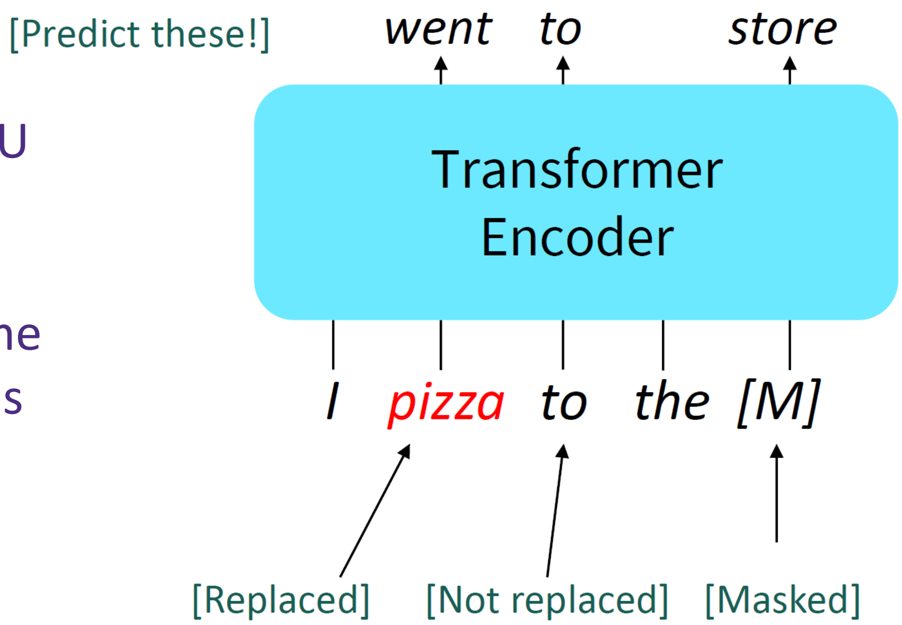


Pre-training Transformer Encoder

- **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

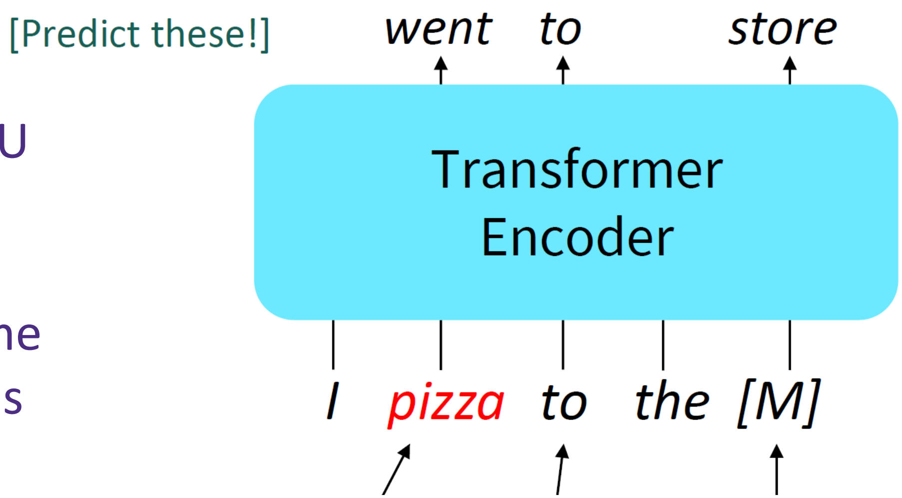


Pre-training Transformer Encoder

- **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

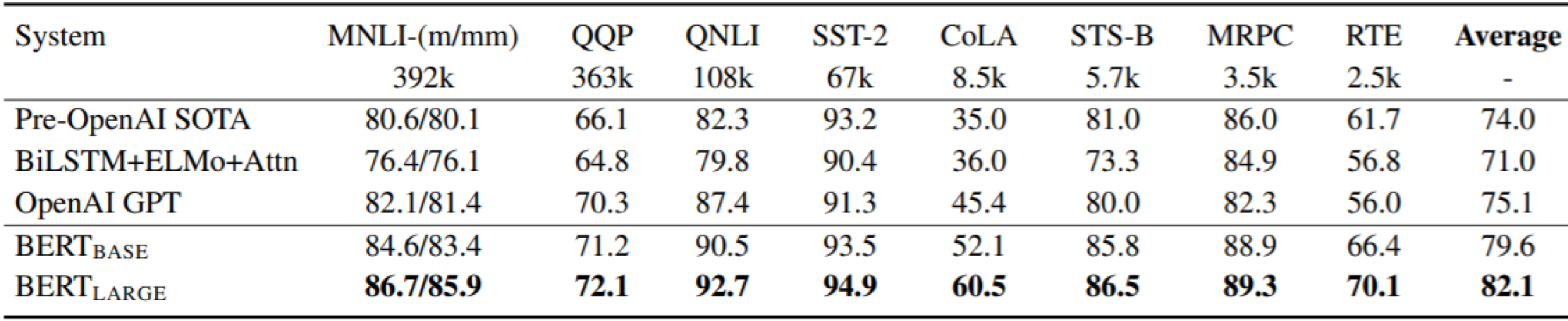


[Predict these!]

went *to* *store*

Transformer Encoder

I *pizza* *to* *the* *[M]*

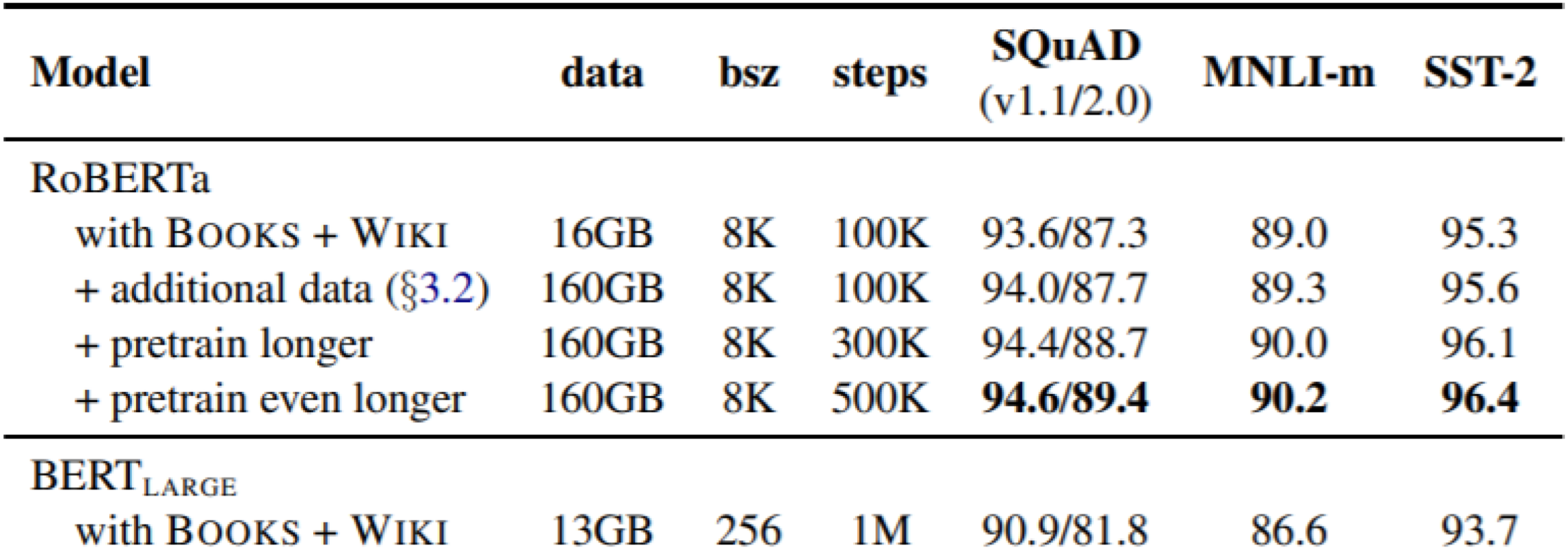


System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT$_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
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**
BERT$_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

Pre-training Decoder

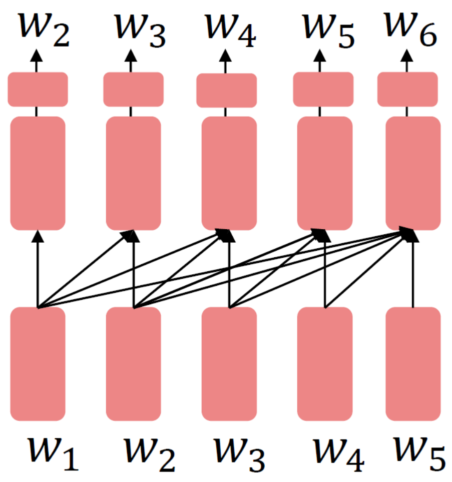
- Decoder Pretraining
  - Just train a language model over corpus.
  - Good for generative task (e.g., text generation)

- 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

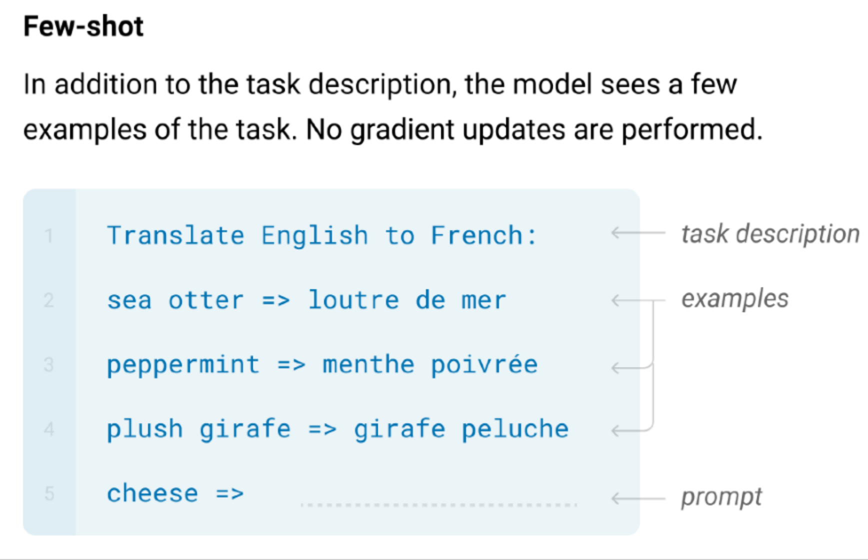
- GPT-3 (OpenAI ’20)
  - Language models are few-shot learners
  - 175B parameters

- Also Image GPT (OpenAI ’20)



Pre-training Decoder

- GPT-3 (OpenAI ’20)
  - You may not need to fine-tune the model parameters for downstrea mtasks.
  - 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.

```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

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

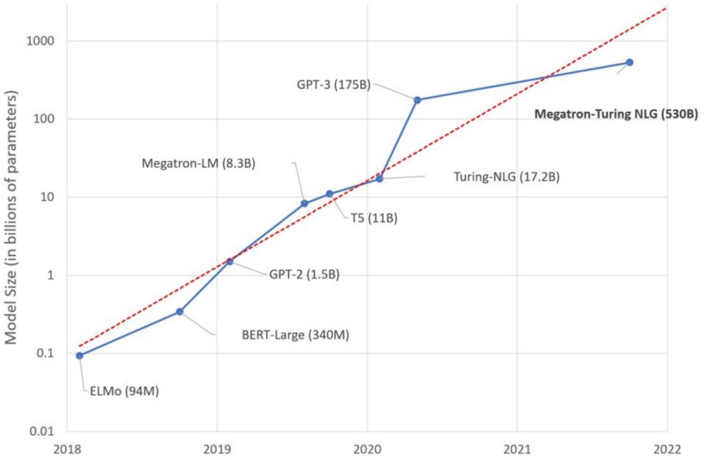
Code: `px.line(df.query("continent == 'Europe' and country == 'France'"), x='year', y='pop', color='country', log_y=False, log_x=False)`

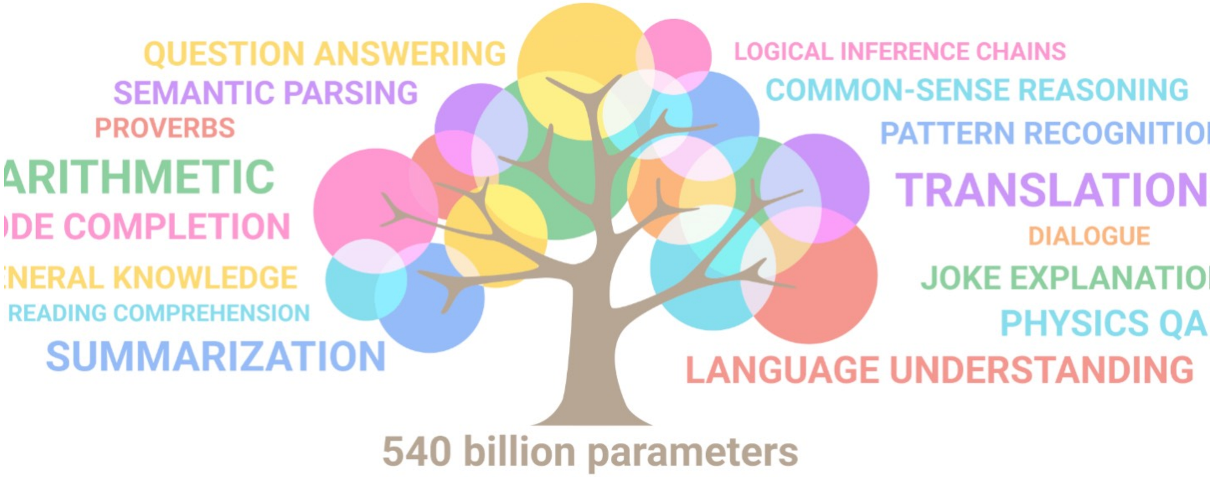
Description: Put y-axis on log scale

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

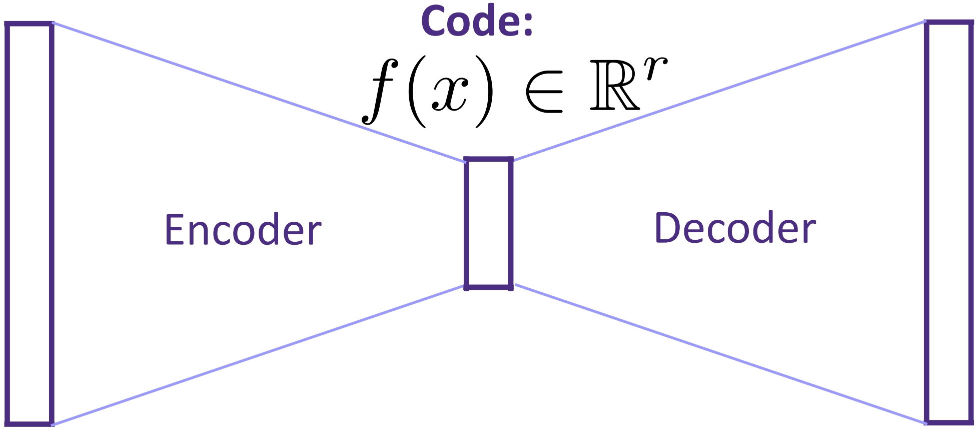
- A big ongoing race on training large language models
  - Megatron-Turing NLG (530B, Microsoft, ’22)
  - Pathways Language Model (540B, Google, ’22)





Autoencoders

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



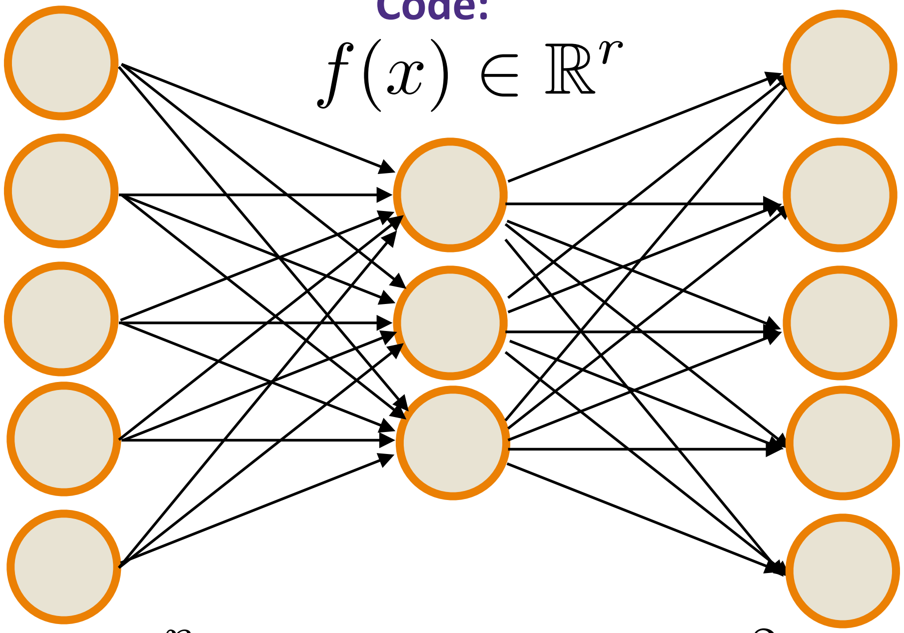
Code:
$f(x) \in \mathbb{R}^r$

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

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

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

Autoencoders



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

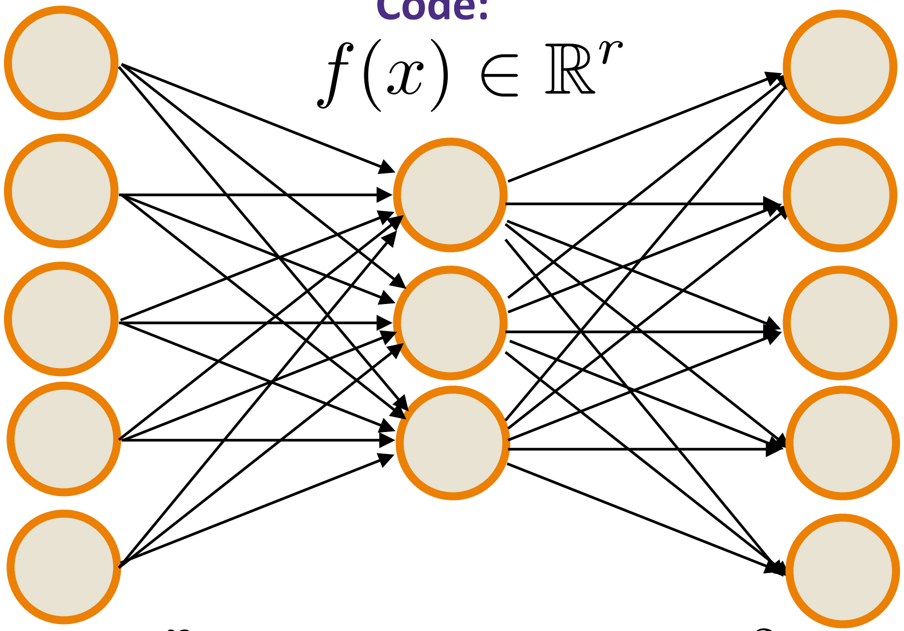
Code:
$f(x) \in \mathbb{R}^r$

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 $g(y) = By$?

Autoencoders



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

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

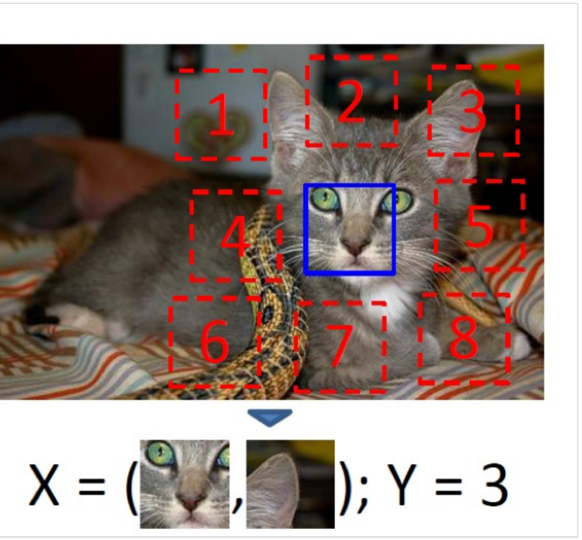
Output:
$\widehat{x} = g(f(x)) \in \mathbb{R}^d$

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

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

Self-supervised learning in computer vision

Context Prediction (Pathak et al., ‘15)



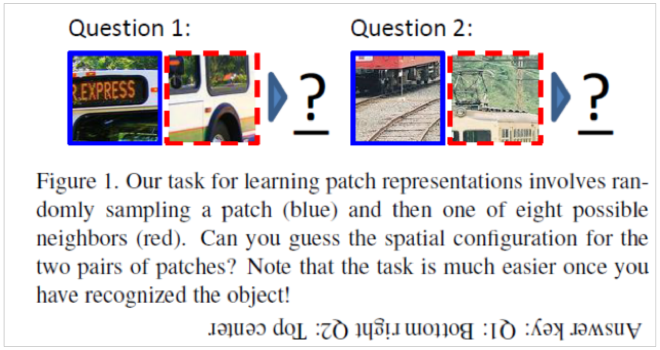
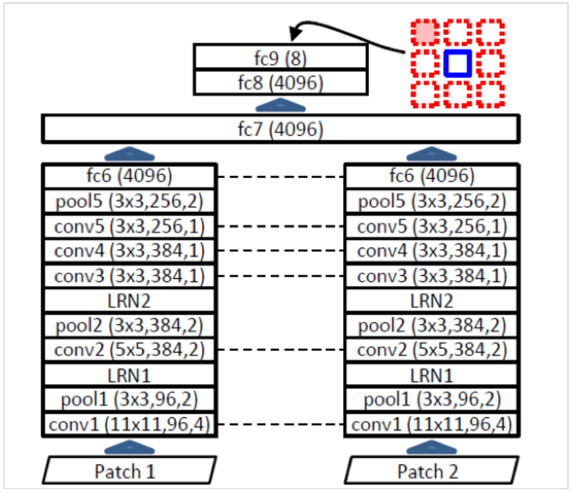
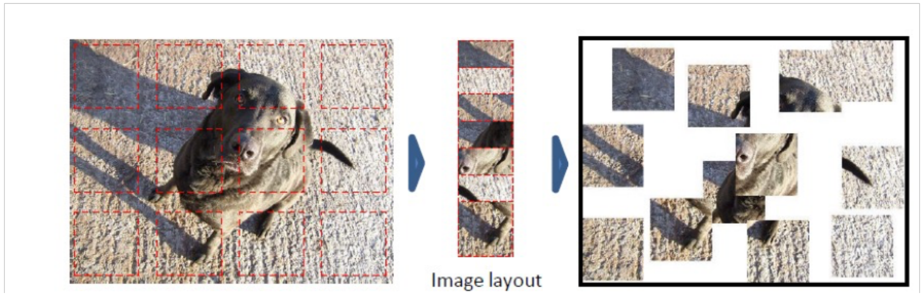


Figure 1. Our task for learning patch representations involves randomly sampling a patch (blue) and then one of eight possible neighbors (red). Can you guess the spatial configuration for the two pairs of patches? Note that the task is much easier once you have recognized the object!

Answer key: Q1: Bottom right Q2: Top center





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

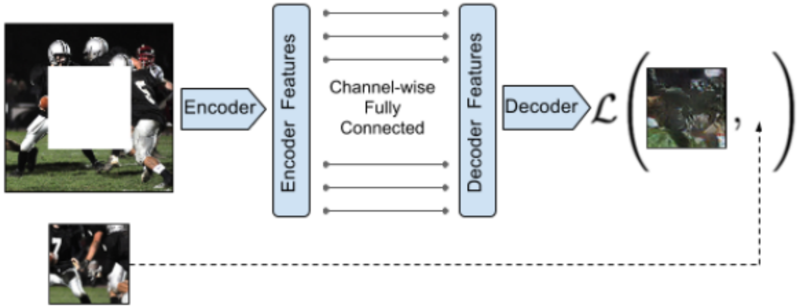


Figure 2: Context Encoder. The context image is passed through the encoder to obtain features which are connected to the decoder using channel-wise fully-connected layer as described in Section 3.1. The decoder then produces the missing regions in the image.

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)

(a) Input context

(b) Human artist

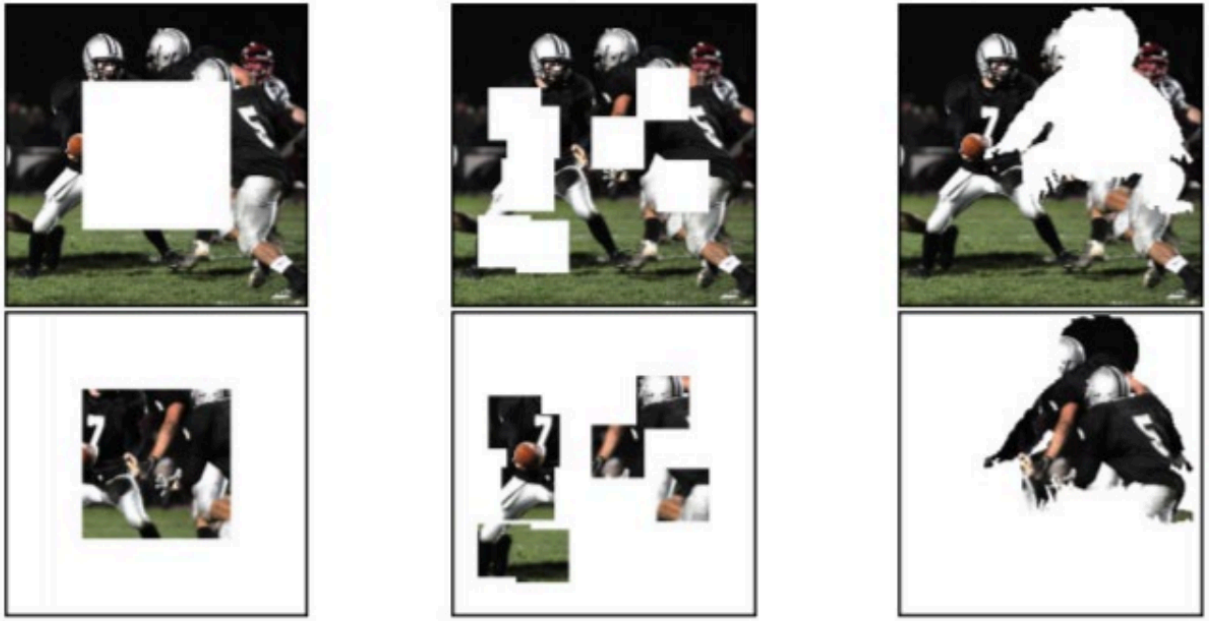
(c) Context Encoder ($L2$ loss)

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

L_2 vs. Adversarial loss

Self-supervised learning in computer vision

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



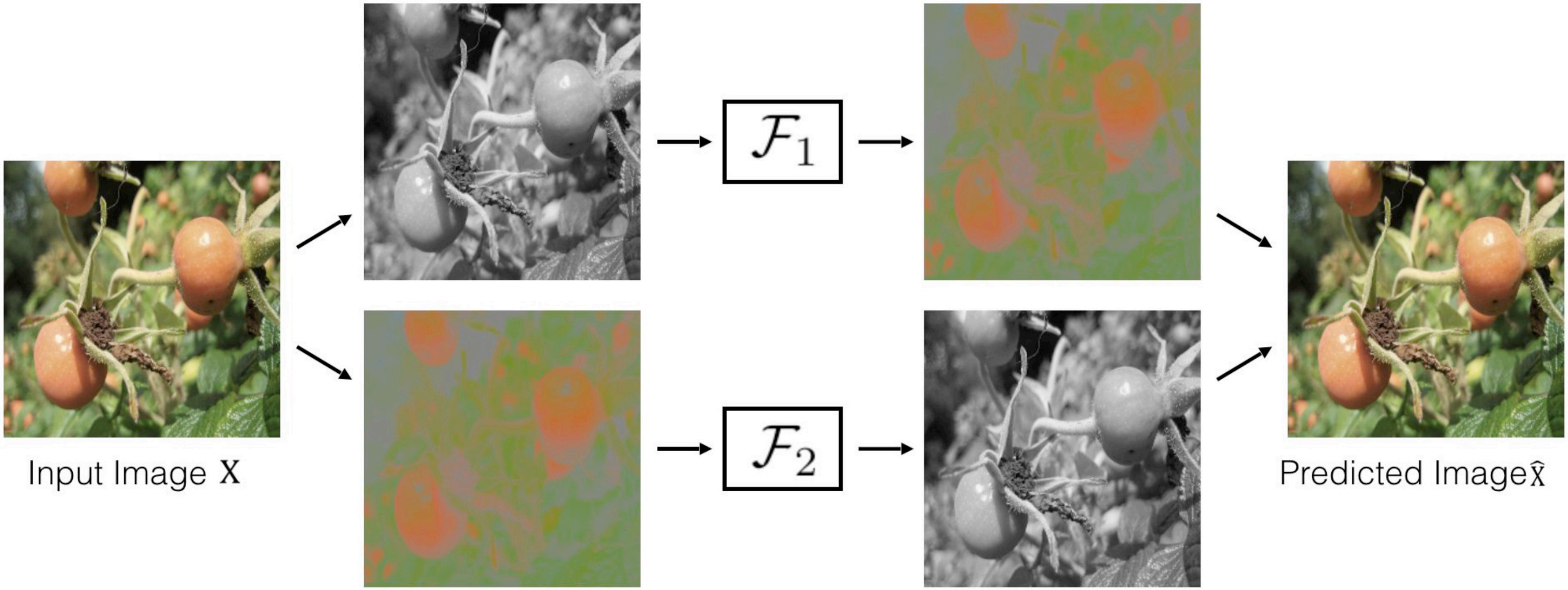
(a) Central region (b) Random block (c) Random region

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

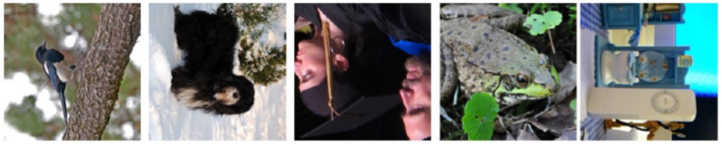
Self-supervised learning in computer vision

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



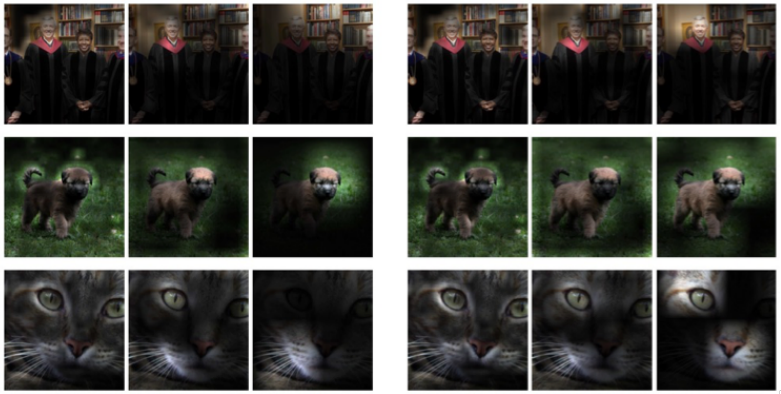
Self-supervised learning in computer vision

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



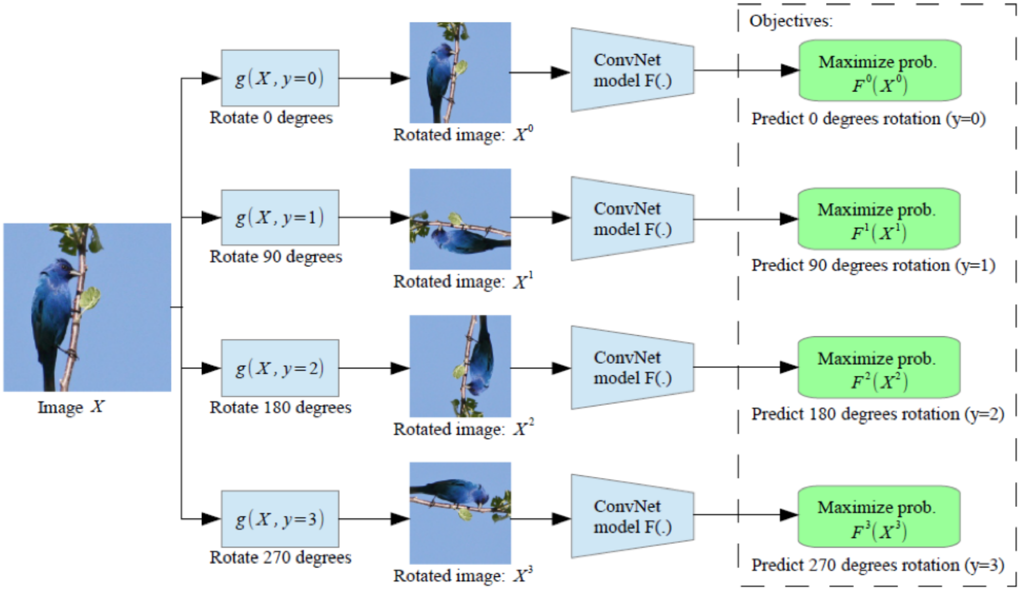
90° rotation 270° rotation 180° rotation 0° rotation 270° rotation

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.



Conv1 27 × 27 Conv3 13 × 13 Conv5 6 × 6 Conv1 27 × 27 Conv3 13 × 13 Conv5 6 × 6

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



Contrastive learning

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:

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

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

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

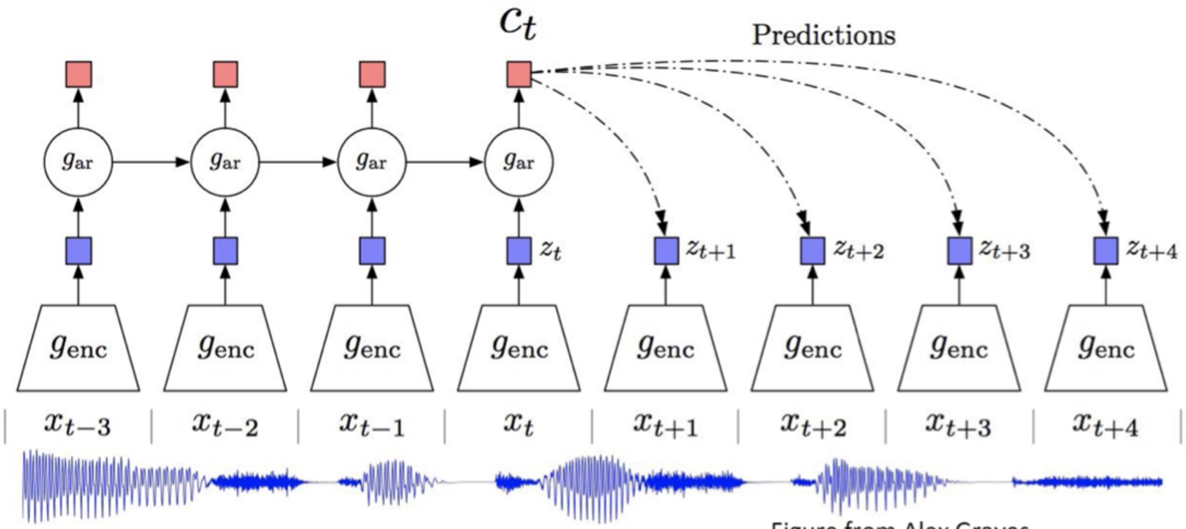


Figure from Alex Graves

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

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

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

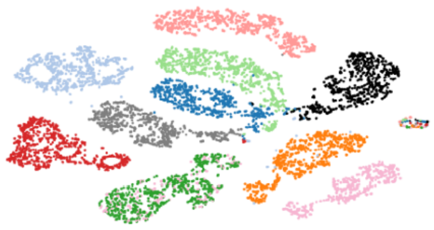


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

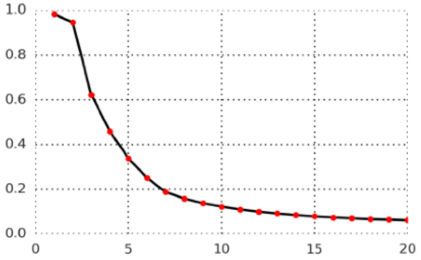
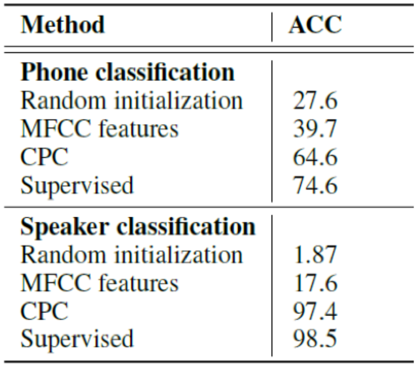
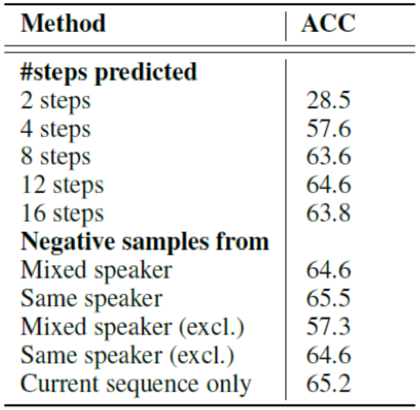


Figure 3: Average accuracy of predicting the positive sample in the contrastive loss for 1 to 20 latent steps in the future of a speech waveform. The model predicts up to 200ms in the future as every step consists of 10ms of audio.



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.



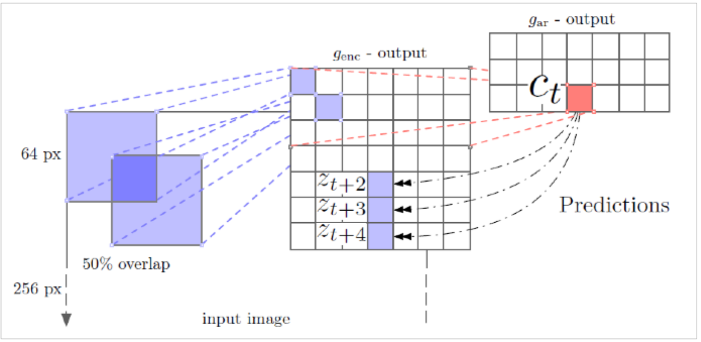
Method	**ACC**
#steps predicted	
2 steps	28.5
4 steps	57.6
8 steps	63.6
12 steps	64.6
16 steps	63.8
Negative samples from	
Mixed speaker	64.6
Same speaker	65.5
Mixed speaker (excl.)	57.3
Same speaker (excl.)	64.6
Current sequence only	65.2

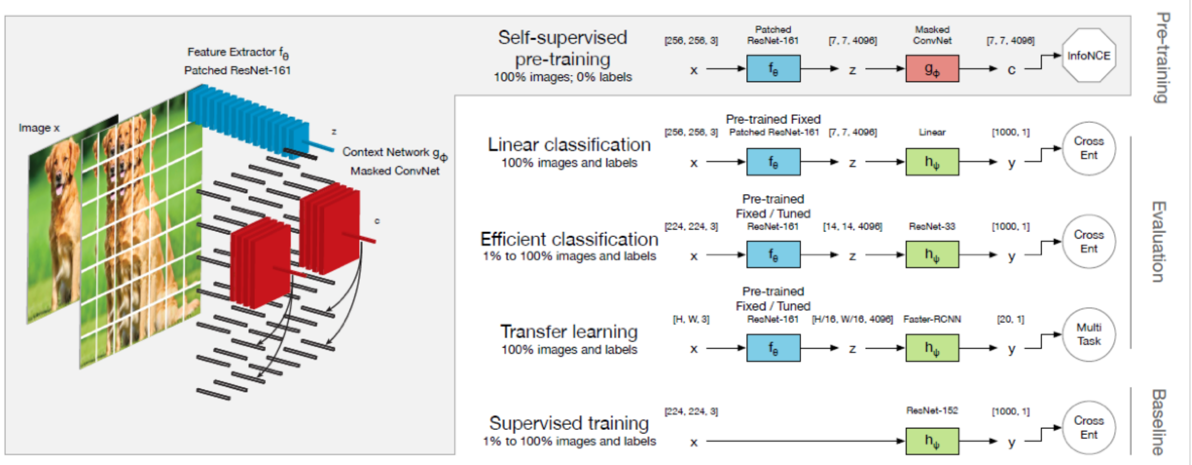
Table 2: LibriSpeech phone classification ablation experiments. More details can be found in Section 3.1.

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

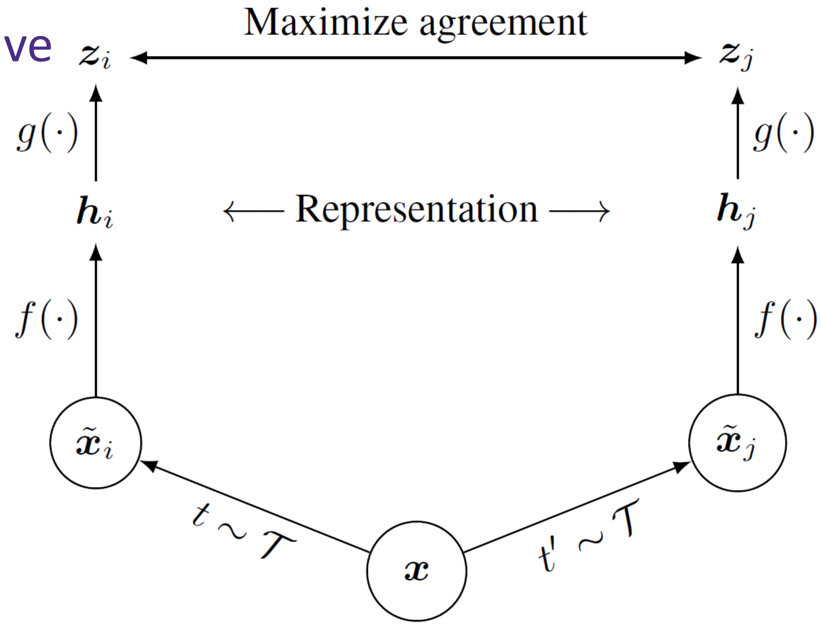




Contrastive learning

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

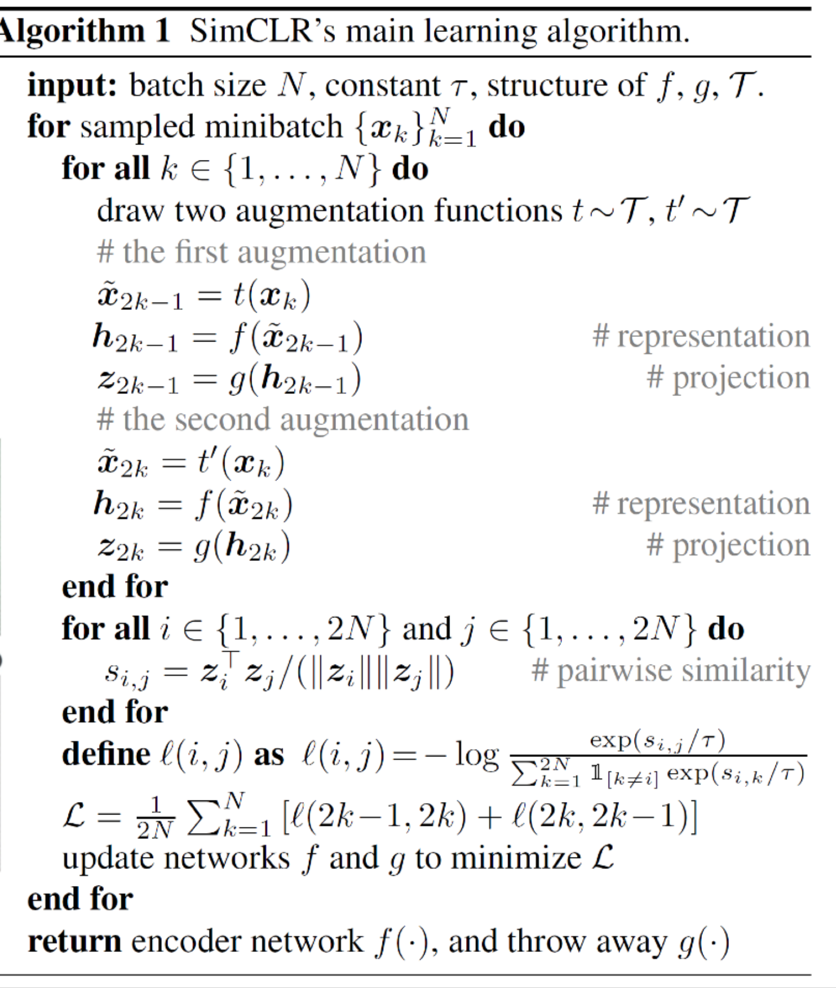
- SimCLR (Chen et al. ’20)
  - A simple framework for contrastive learning of visual representations
    - Predefine a set of transformations
    - For a data, sample two transformations
    - Maximum agreement on representations
  - No negative pairs explicitly
    - Non-paired data in the batch are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



Contrastive learning

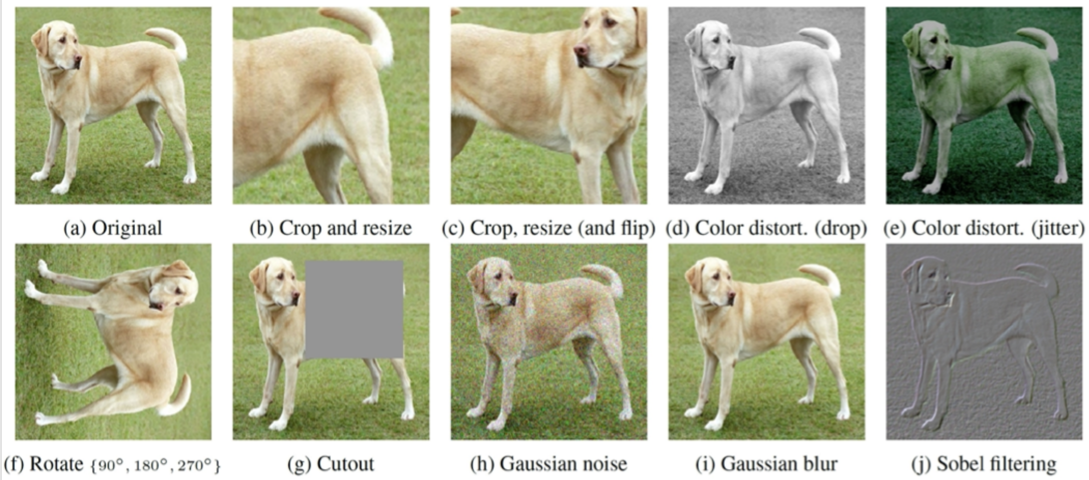
Contrastive Predictive Coding (Van den Oord et al., ’18)

- SimCLR (Chen et al. ’20)



Algorithm 1 SimCLR’s main learning algorithm.

```
input: batch size $N$, constant $\tau$, 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 $g(\cdot)$
```



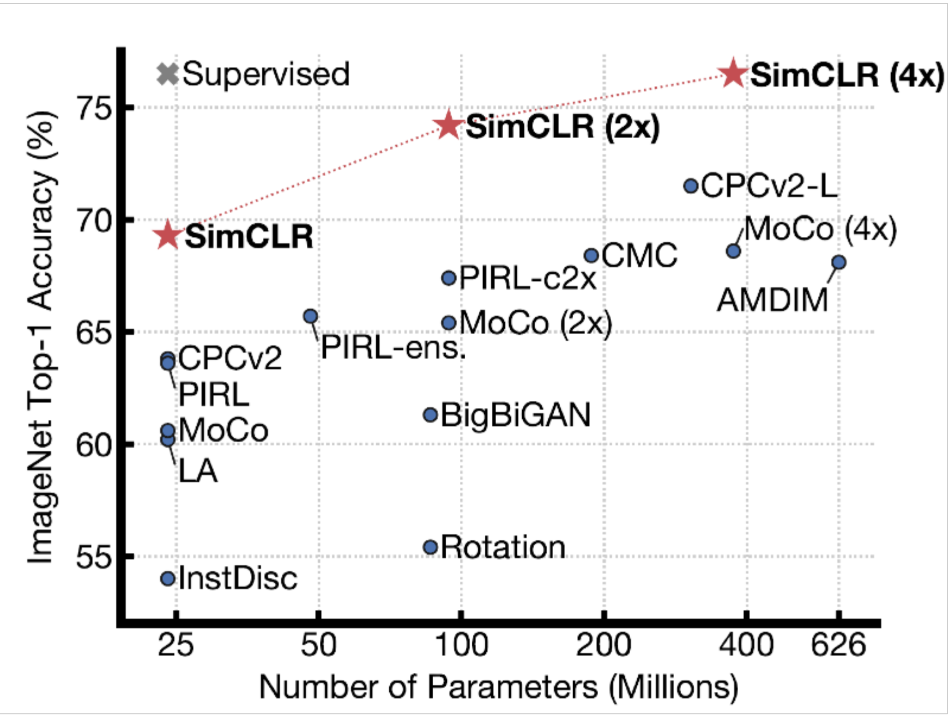
(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

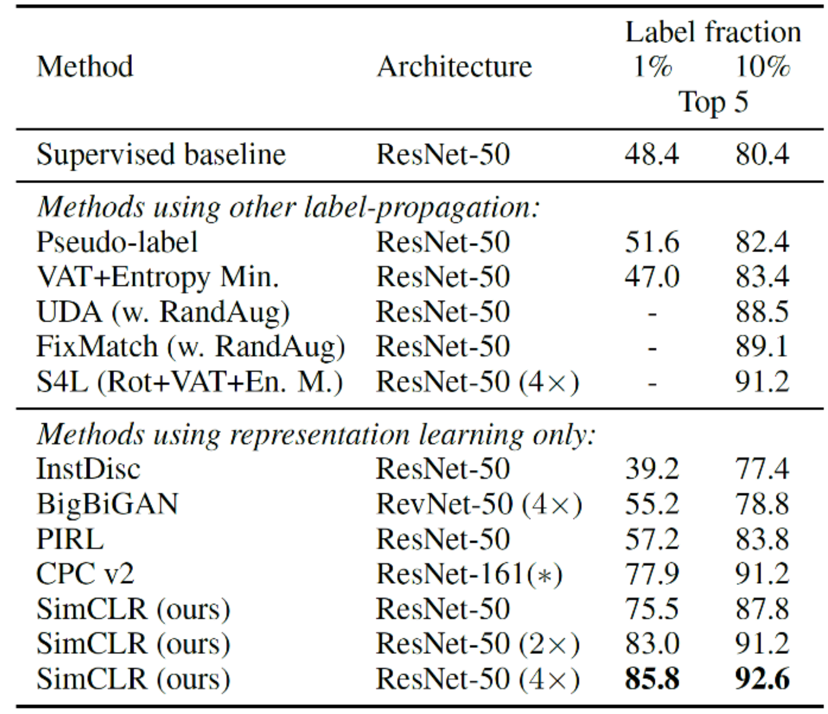
(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

Contrastive learning

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

- SimCLR (Chen et al. ’20)





Method	Architecture	Label fraction 1%	 10%
		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.

Parameter-Efficient Fine-Tuning

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

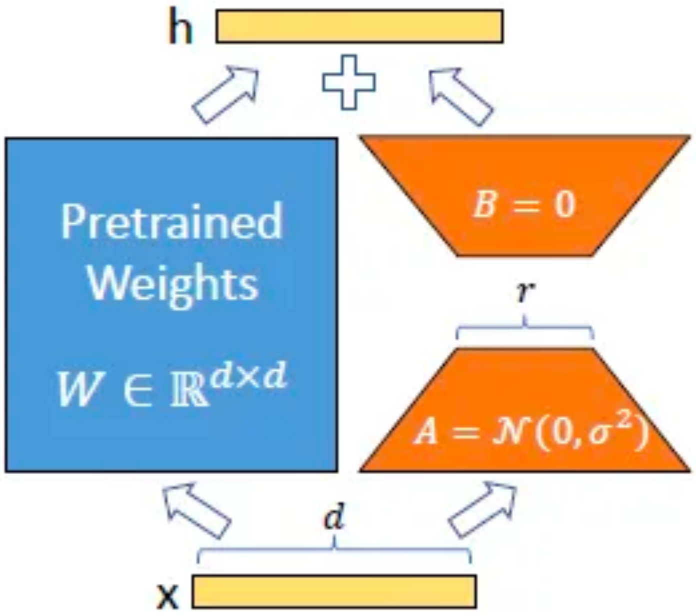


Figure 1: Our reparametrization. We only train A and B.