Representation Learning Methods

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.

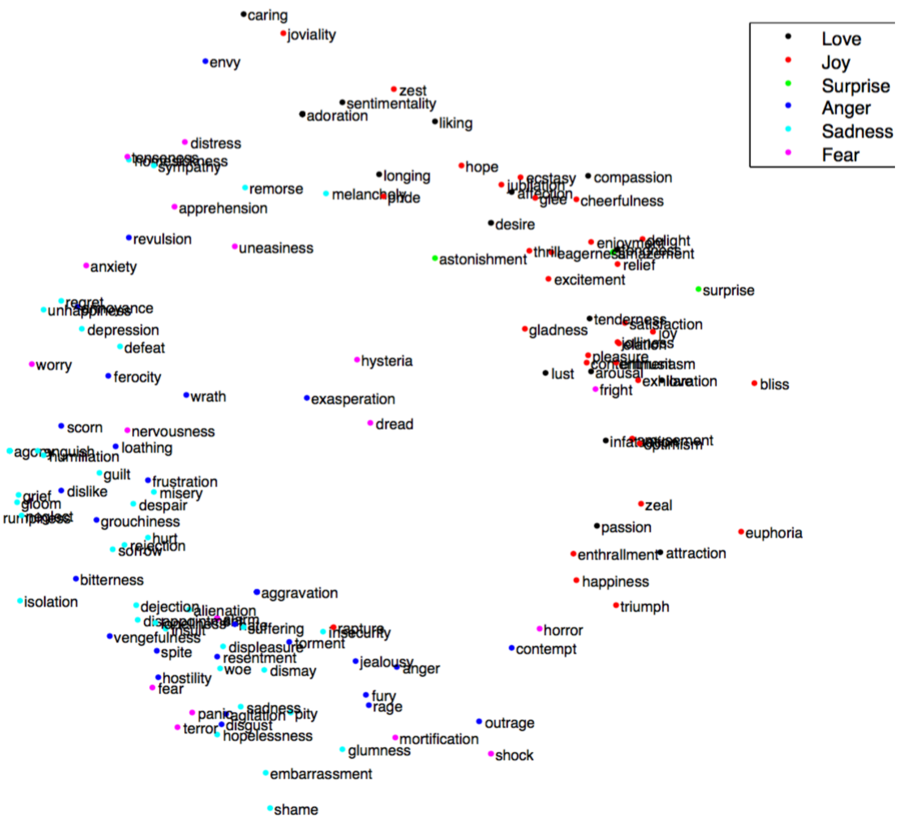
Word embeddings, word2vec

basic unit

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)

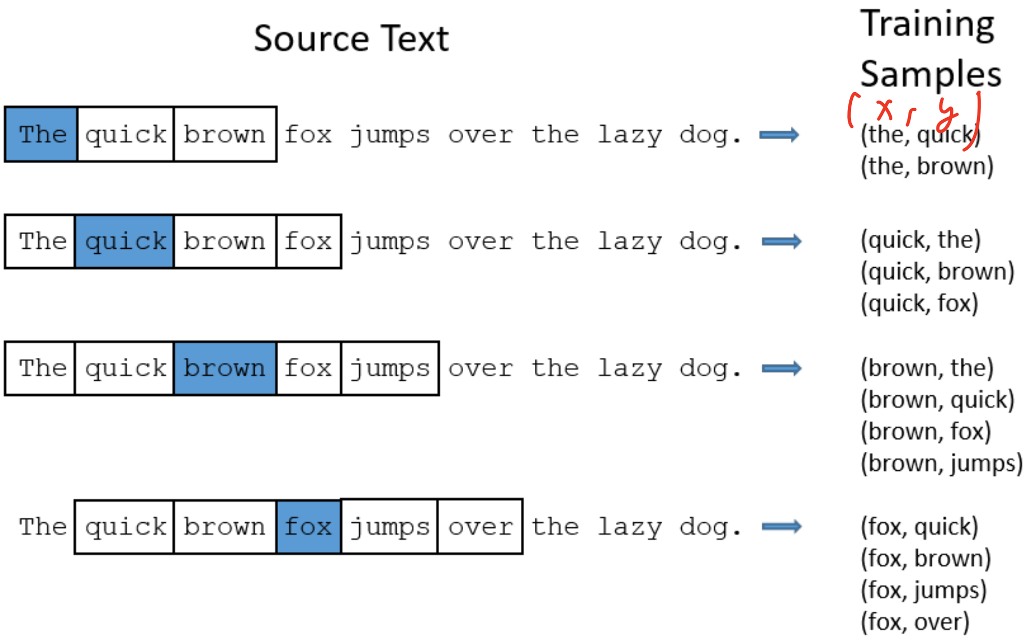


Word embeddings, word2vec

$p(\text{word } 2 \mid \text{word } 1)$

wiki, news, books

each word



Source Text	Training Samples (x, y)
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)

one-hot

$$\begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \in \mathbb{R}^k$$

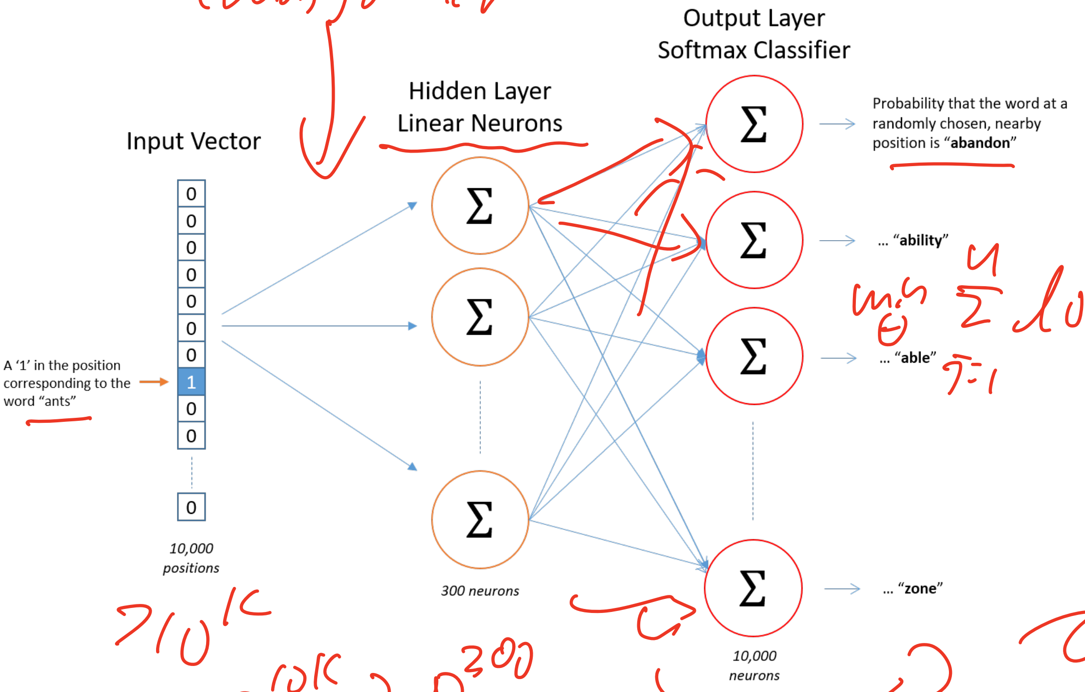
slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word embeddings, word2vec

$P(\text{word2} \mid \text{word1}$

multi-layer FC

transformer


Input Vector
0
0
0
0
0
0
0
1
0
0
0
10,000 positions
A '1' in the position corresponding to the word "ants"
Hidden Layer Linear Neurons
Σ
Σ
Σ
300 neurons
Output Layer Softmax Classifier
Σ
Σ
Σ
Σ
10,000 neurons
Probability that the word at a randomly chosen, nearby position is "**abandon**"
... "**ability**"
... "**able**"
... "**zone**"

$P(\text{"abandon"} \mid \text{"ant"})$

$\min_{\theta} \sum_{i=1}^{n} \text{loss}(x_i, y_i)$

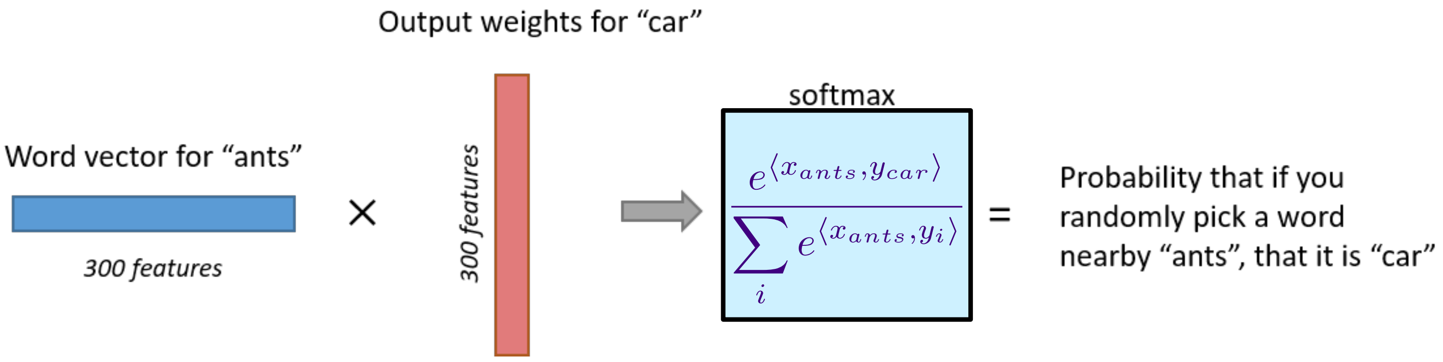
$>10^{k}$

$\mathbb{R}^{10k} \to \mathbb{R}^{300}$

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

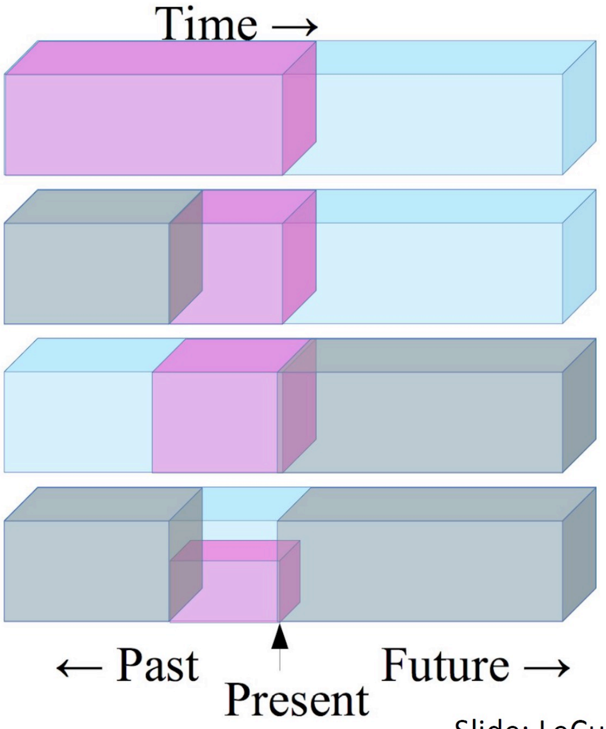
slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

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.

image

- Predict the occluded from the visible
- Pretend there is a part of the input you don't know and predict that.



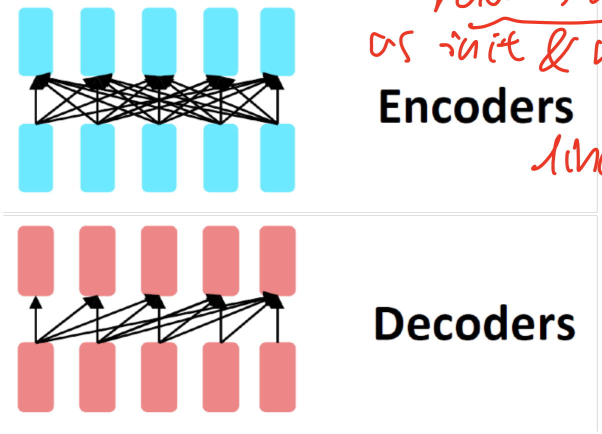
Slide: LeCun

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

(1) fix raw → rep
just re train last linear layer

(2) use learned raw → rep as init & retrain last linear layer



X_t only depends on $X_{i<t}$

Pre-training Transformer Encoder

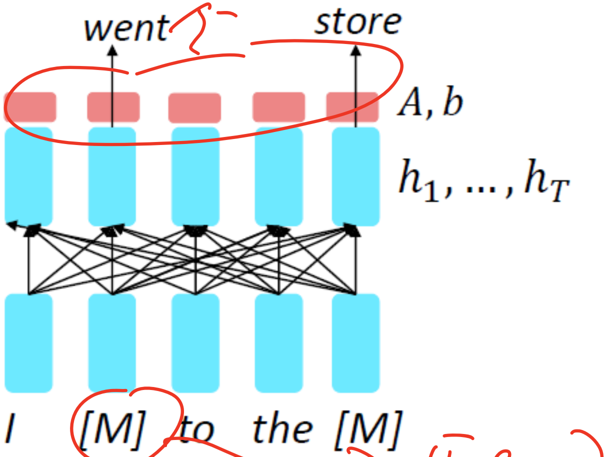
- Pre-training a bi-directional encoder
  - Cannot directly adopt language modeling ← auto-regressive
  - **Idea:** word prediction given contexts (similar to word2vec)

word | rest of the sentence

- 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

$$\arg\min_{\theta} \sum_{i=1}^{n} loss(f(x_i), y_i)$$

classification



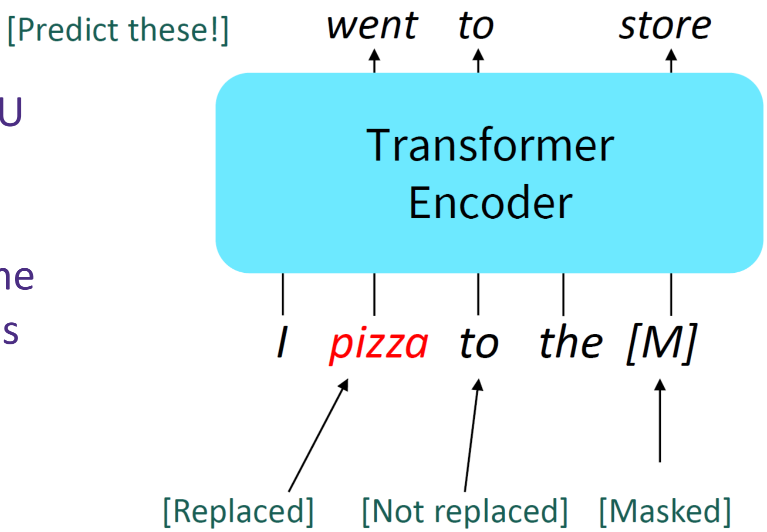
$x = \begin{pmatrix} I \\ [M] \\ to \\ the \\ [M] \end{pmatrix} \rightarrow$ (went) y

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

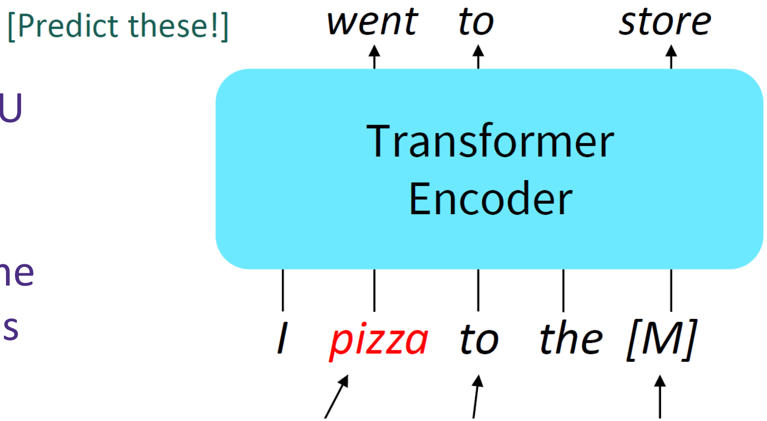


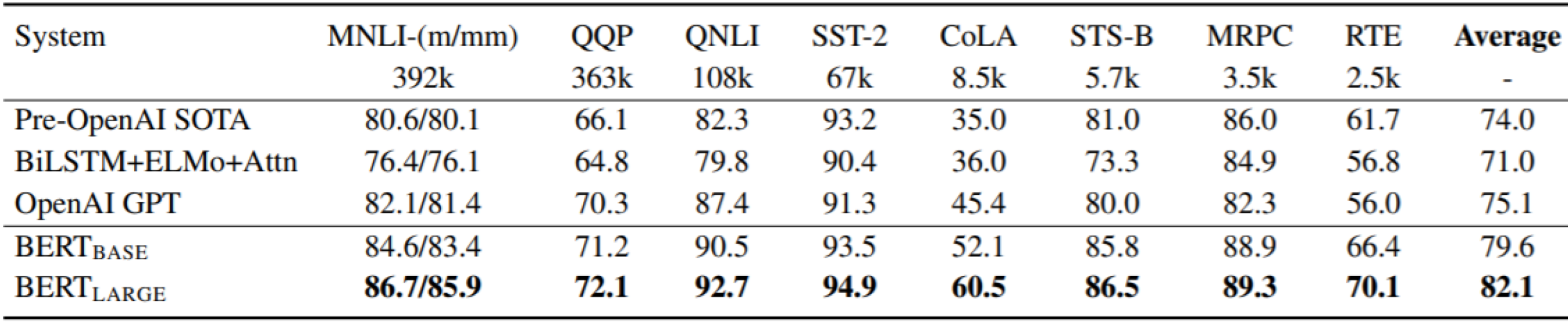
Pre-training Transformer Encoder

- **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



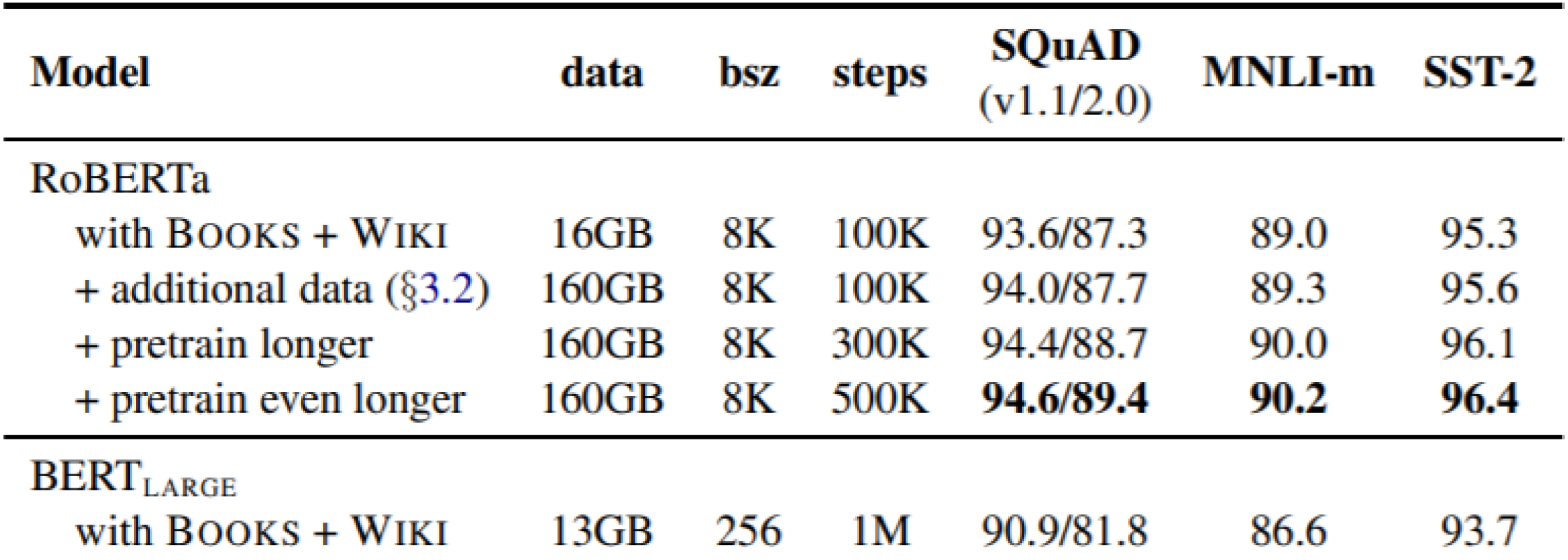


System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1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

Pre-training Decoder

learn $P(X_t | X_{i<t})$

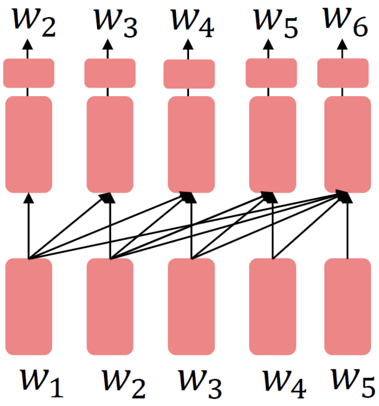
- Decoder Pretraining
  - Just train a language model over corpus.
  - Good for generative task (e.g., text generation)

- 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

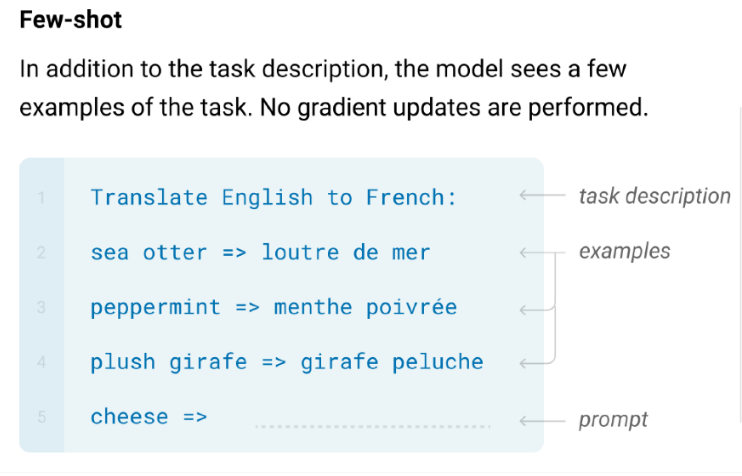
- GPT-3 (OpenAI ’20)
  - Language models are few-shot learners
  - 175B parameters

- 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

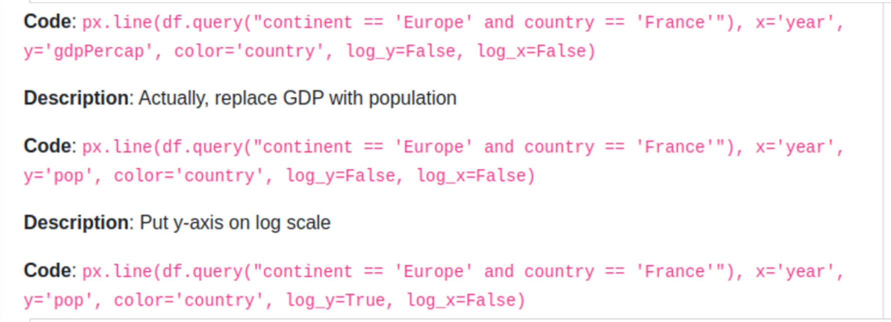


Few-shot

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 Generatio



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

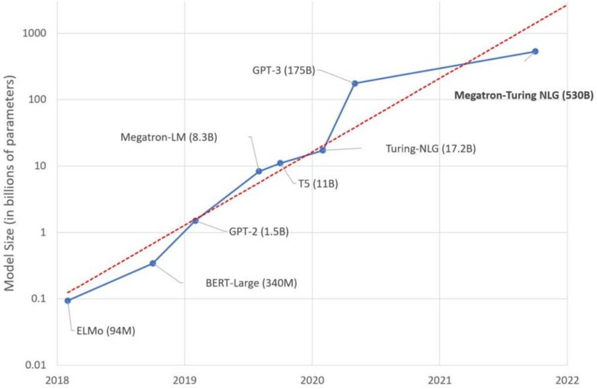
Code: px.line(df.query("continent == 'Europe' and country == 'France'"), x='year', y='pop', color='country', log_y=False, log_x=False)

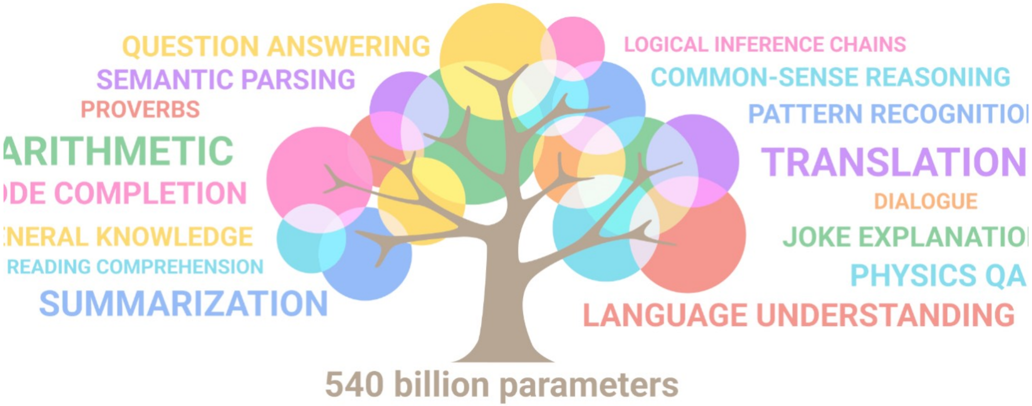
Description: Put y-axis on log scale

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)



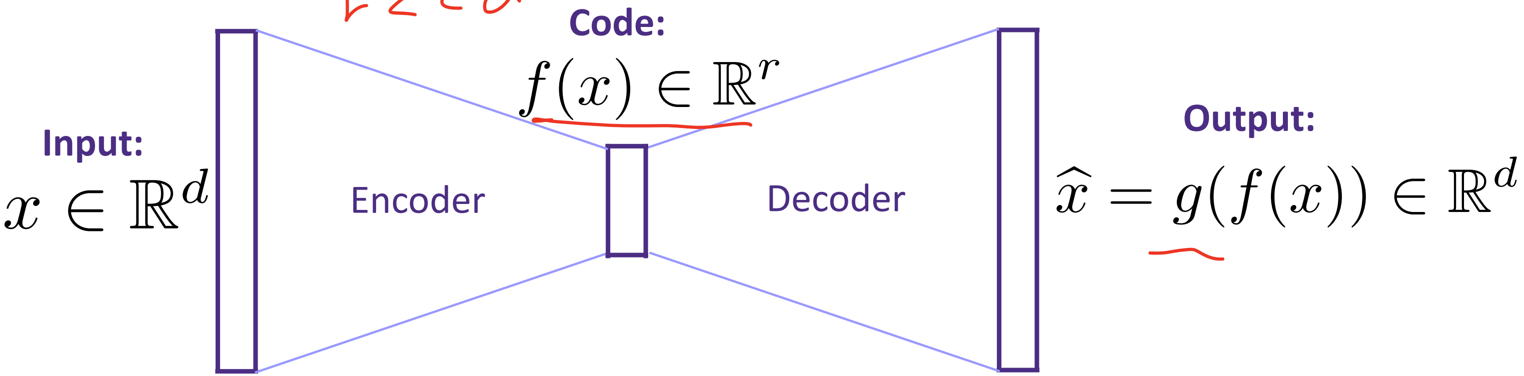


Autoencoders

$g = f^{-1}$

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

$r \ll d$



Code:

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

Input:

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

Encoder

Decoder

Output:

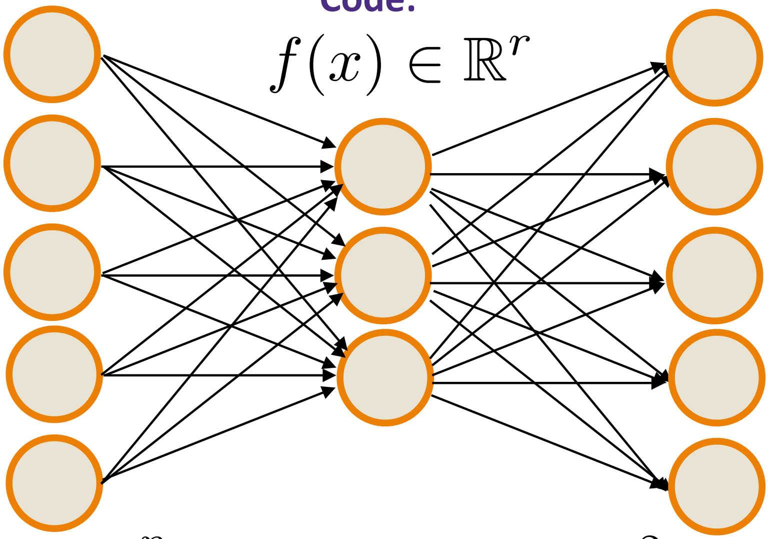
$\widehat{x} = g(f(x)) \in \mathbb{R}^d$

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

f, g NN

Autoencoders

$X \in \mathbb{R}^{d \times n}$



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

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

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

what f, g are linear

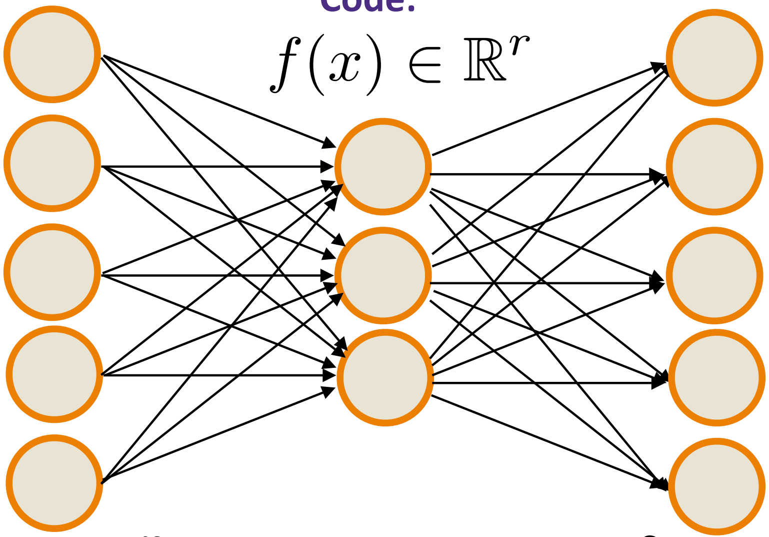
$$\underset{f,g}{\text{minimize}} \sum_{i=1}^{n} \|x_i - g(f(x_i))\|_2^2$$

$A : \mathbb{R}^{d \times r}$, $B : \mathbb{R}^{r \times d}$

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

$\min_{A,B} \|BAX - X\|_F^2$, $B = A^+$

Autoencoders



Input:

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

Code:

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

Output:

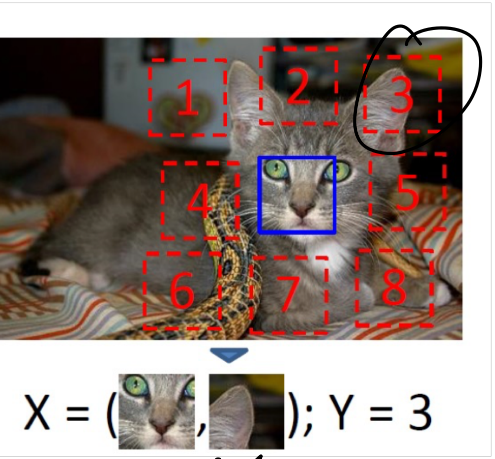
$\widehat{x} = g(f(x)) \in \mathbb{R}^d$

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

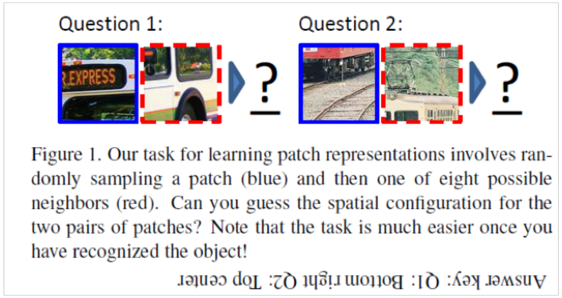
What if $f(X) = Ax$ and $g(y) = By$?

Self-supervised learning in computer vision

Context Prediction (Pathak et al., '15)



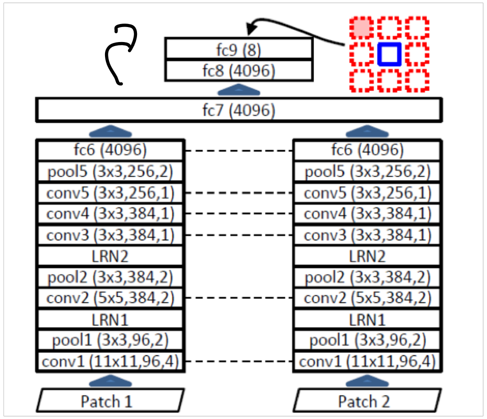
x_1 x_2

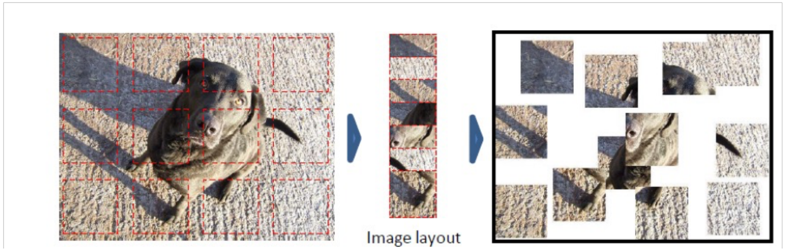

Question 1:

Question 2:

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

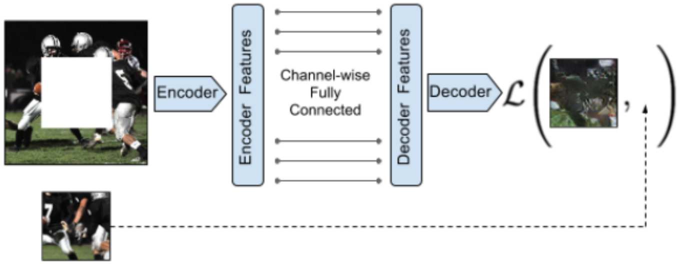


Figure 2: Context Encoder. The context image is passed through the encoder to obtain features which are connected to the decoder using channel-wise fully-connected layer as described in Section 3.1. The decoder then produces the missing regions in the image.

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

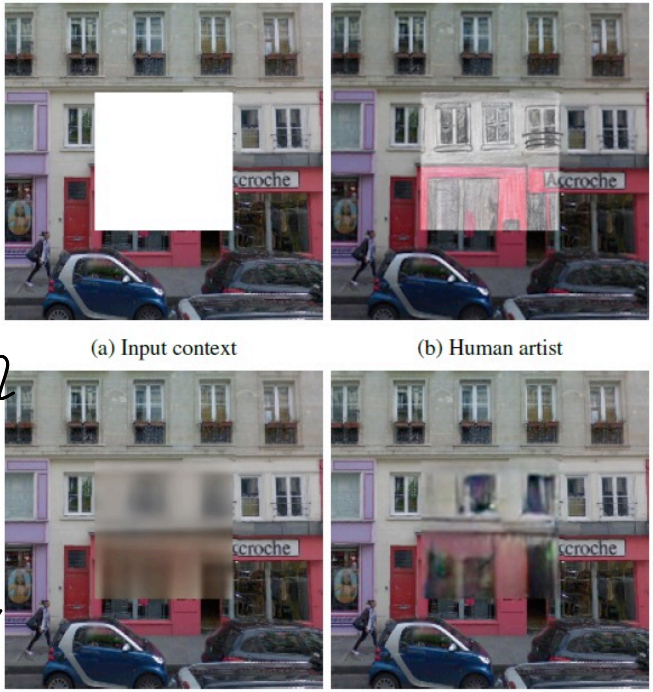
- **Feature learning by Inpainting** (Pathak et al., ’16)

L2 : $\hat{m}$: mask

$\hat{m} \odot X$: miss part

$\| \hat{m} \odot X - \hat{m} \odot F[(1-m) \odot X] \|_F^2$

F : encoder + decoder



(a) Input context

(b) Human artist

(c) Context Encoder (*L*2 loss)

(d) Context Encoder (*L*2 + Adversarial loss)

Adv loss

$\max \mathbb{E}_{x \sim X} [\log D(x)]$

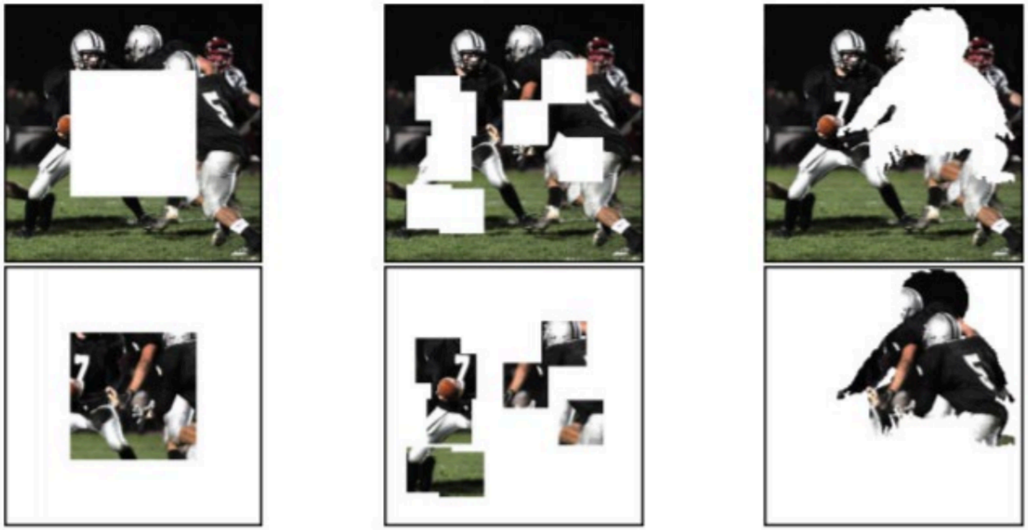
$D \leftarrow NN$

$+ \log(1 - D(F((1-m) \odot x)))$

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

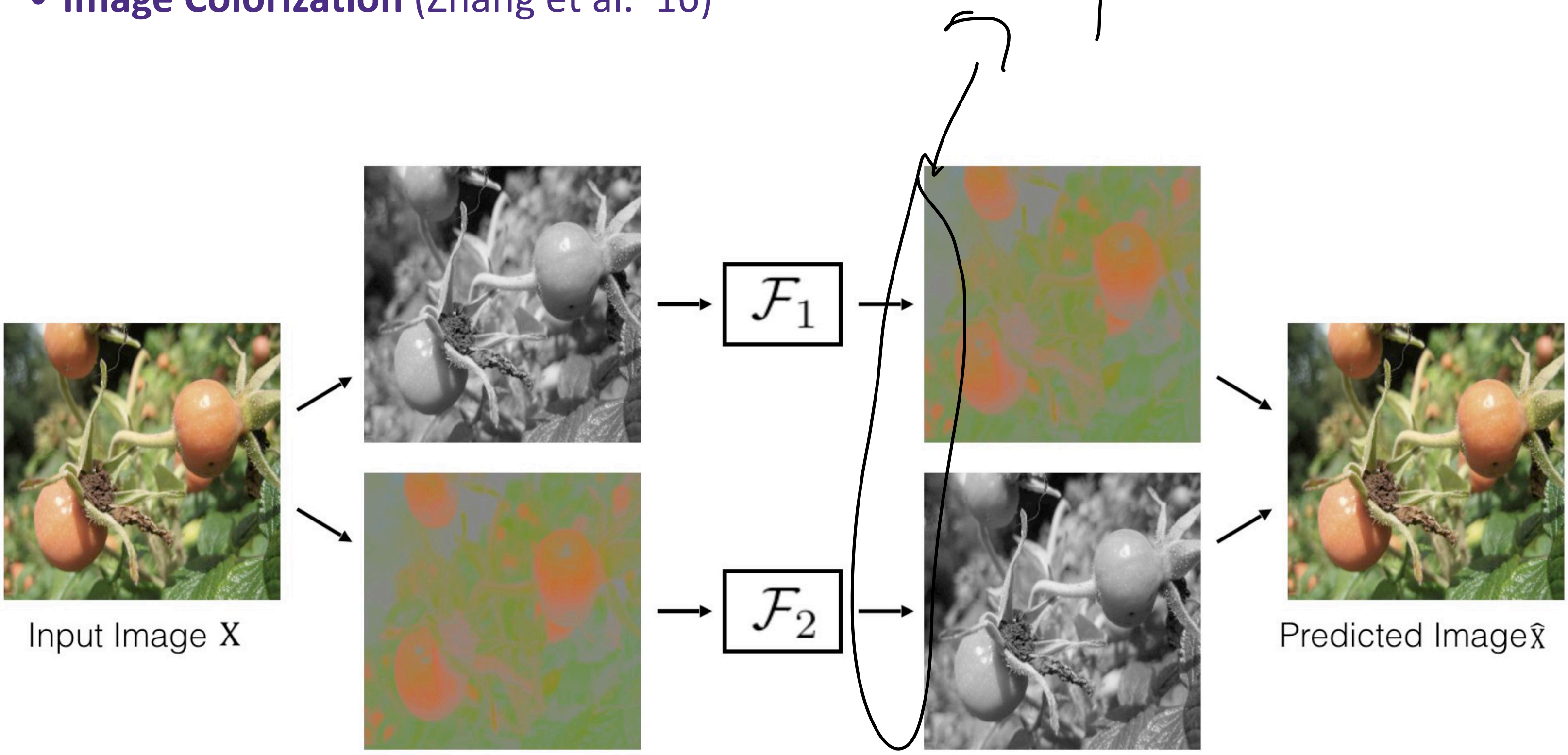
Figure 3: An example of image x with our different region masks $\hat{M}$ applied, as described in Section 3.3.

Fixed region vs. random square block vs. random region

Self-supervised learning in computer vision

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

representation



Self-supervised learning in computer vision

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

vep

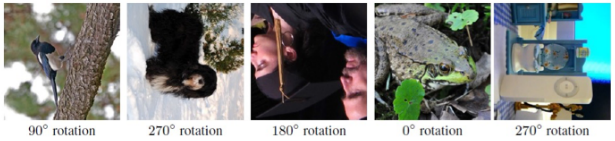
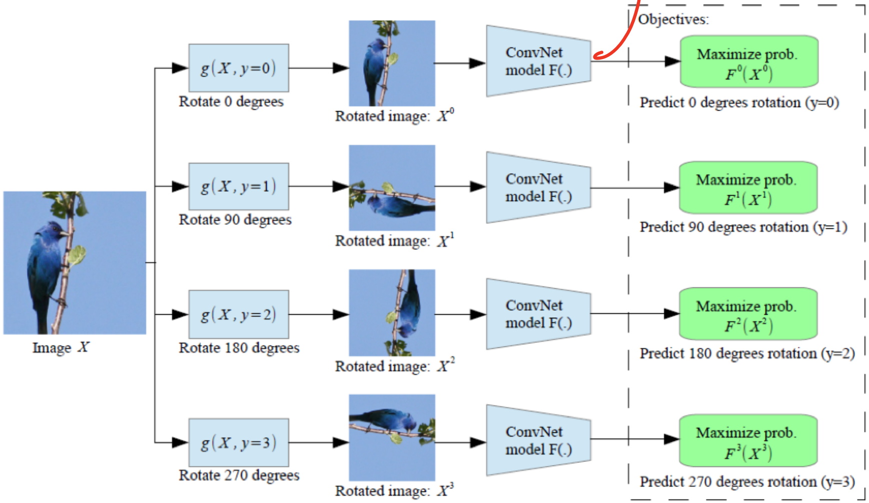


Figure 1: Images rotated by random multiples of 90 degrees (e.g., 0, 90, 180, or 270 degrees). The core intuition of our self-supervised feature learning approach is that if someone is not aware of the concepts of the objects depicted in the images, he cannot recognize the rotation that was applied to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

(a) **Attention maps of supervised model**

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



Contrastive learning

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. rotation,
- 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:

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

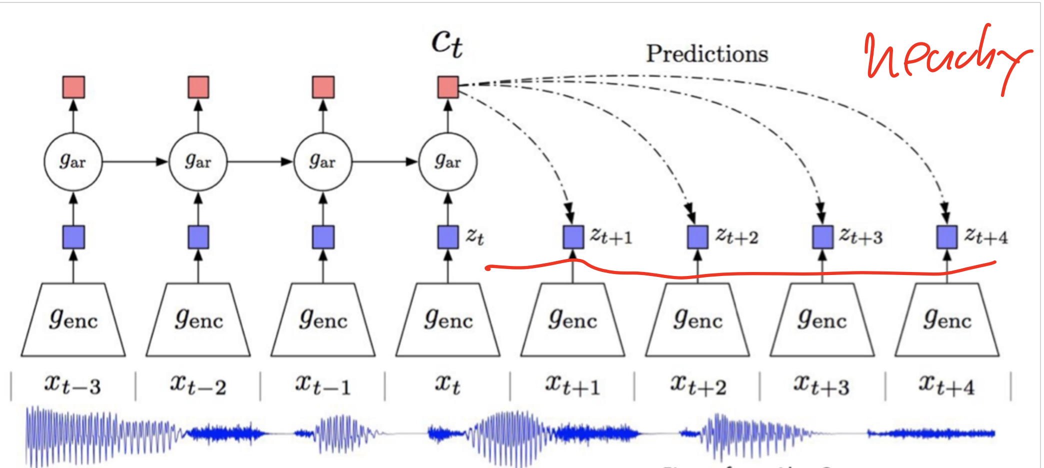
$\in$ all

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

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



nearby

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

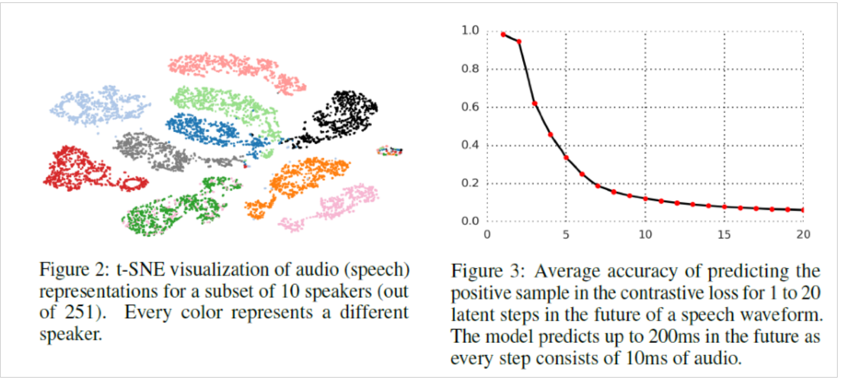
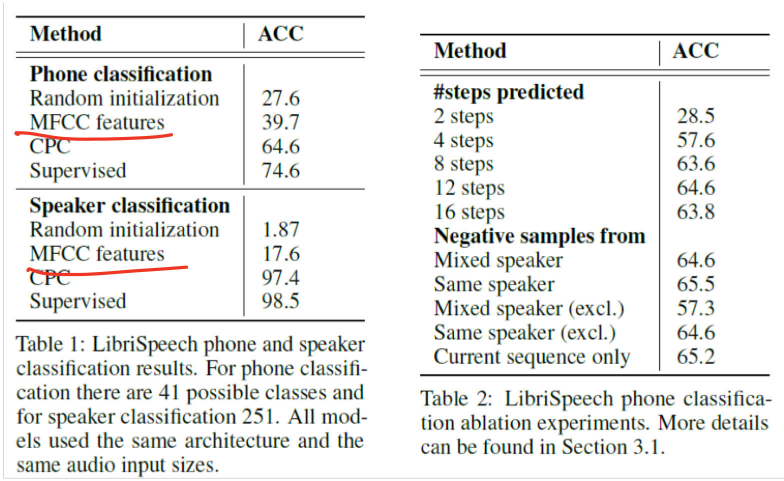


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

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.

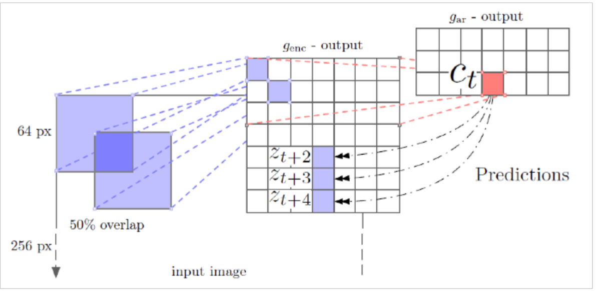
Method	ACC
#steps predicted	
2 steps	28.5
4 steps	57.6
8 steps	63.6
12 steps	64.6
16 steps	63.8
Negative samples from	
Mixed speaker	64.6
Same speaker	65.5
Mixed speaker (excl.)	57.3
Same speaker (excl.)	64.6
Current sequence only	65.2

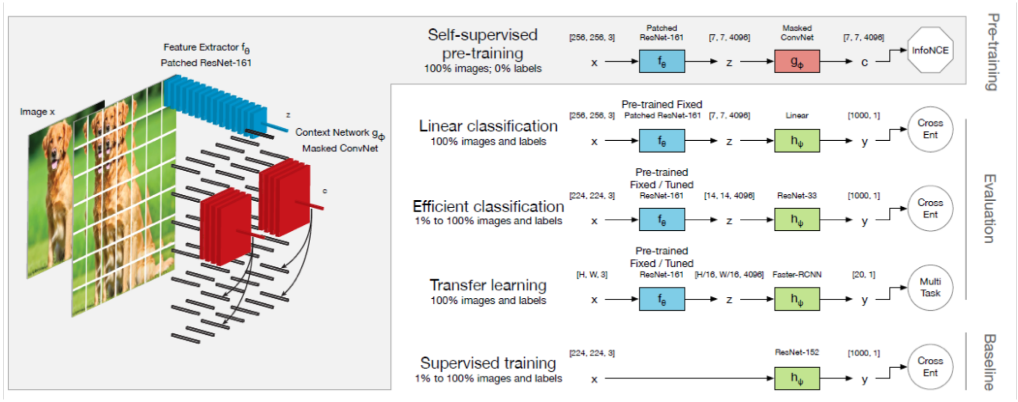
Table 2: LibriSpeech phone classification ablation experiments. More details can be found in Section 3.1.

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

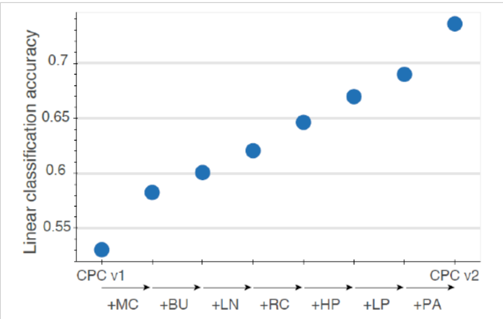
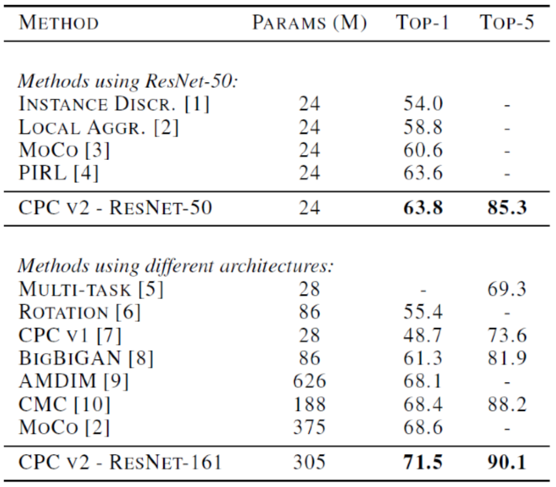
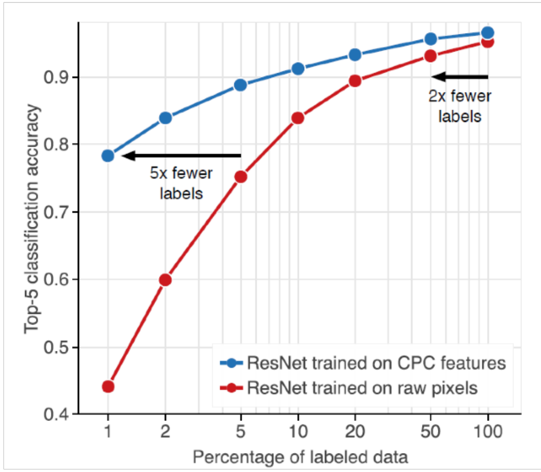


Figure 3. Linear classification performance of new variants of CPC, which incrementally add a series of modifications. MC: model capacity. BU: bottom-up spatial predictions. LN: layer normalization. RC: random color-dropping. HP: horizontal spatial predictions. LP: larger patches. PA: further patch-based 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)

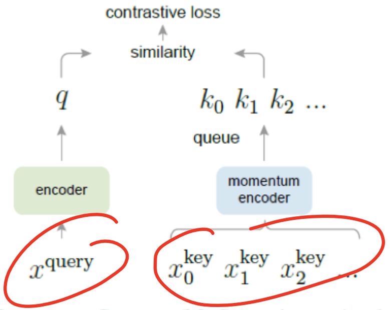
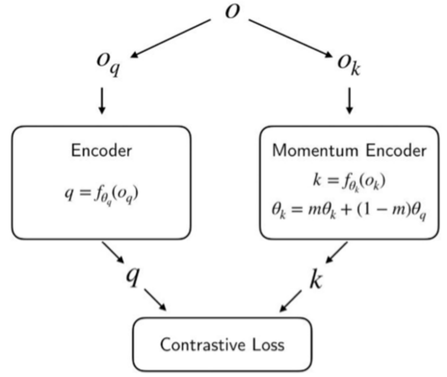


Figure 1. Momentum Contrast (MoCo) trains a visual representation encoder by matching an encoded query q to a dictionary of encoded keys using a contrastive loss. The dictionary keys $\{k_0, k_1, k_2, \ldots\}$ are defined on-the-fly by a set of data samples. The dictionary is built as a queue, with 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.



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

θ_q: query para

θ_k: key para

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

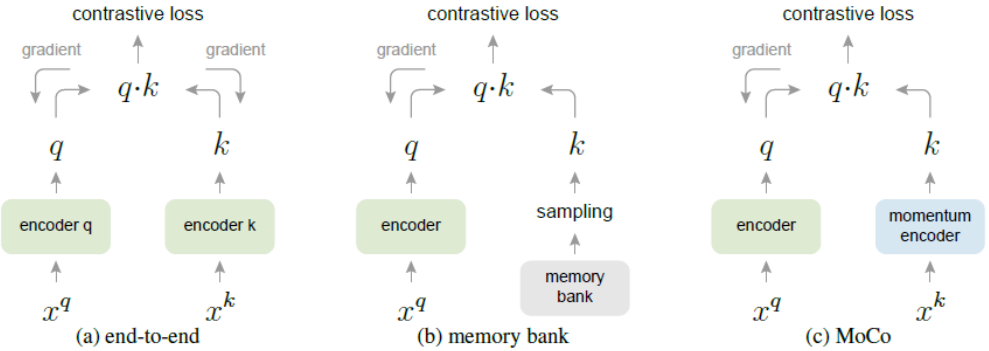


Figure 2. **Conceptual comparison of three contrastive loss mechanisms** (empirical comparisons are in Figure 3 and Table 3). Here we illustrate one pair of query and key. The three mechanisms differ in how the keys are maintained and how the key encoder is updated. **(a)**: The encoders for computing 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)

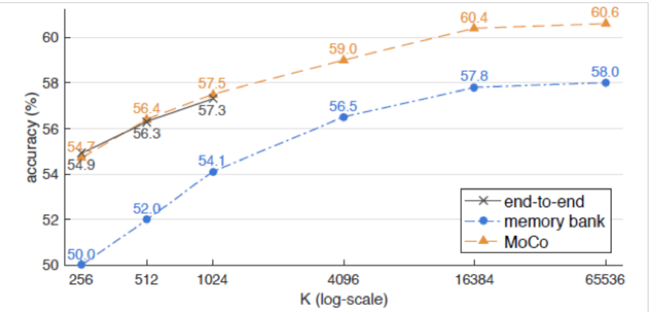
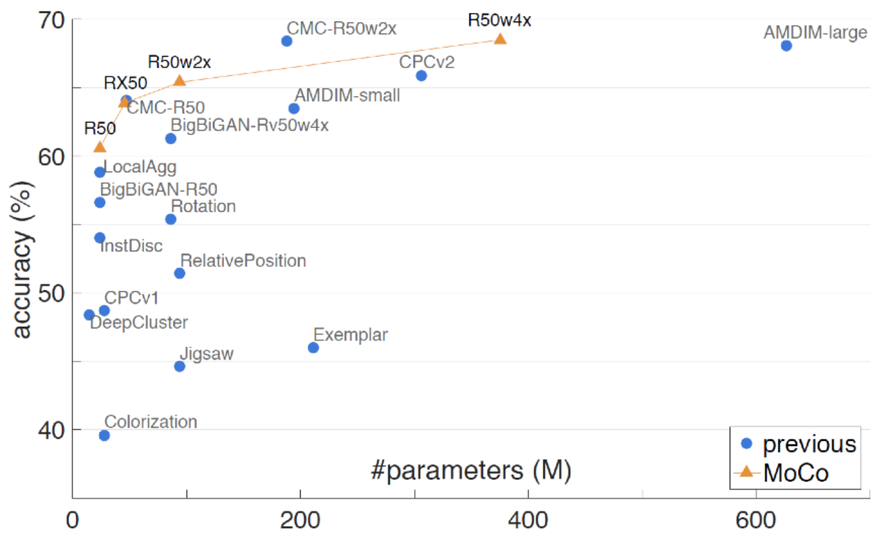


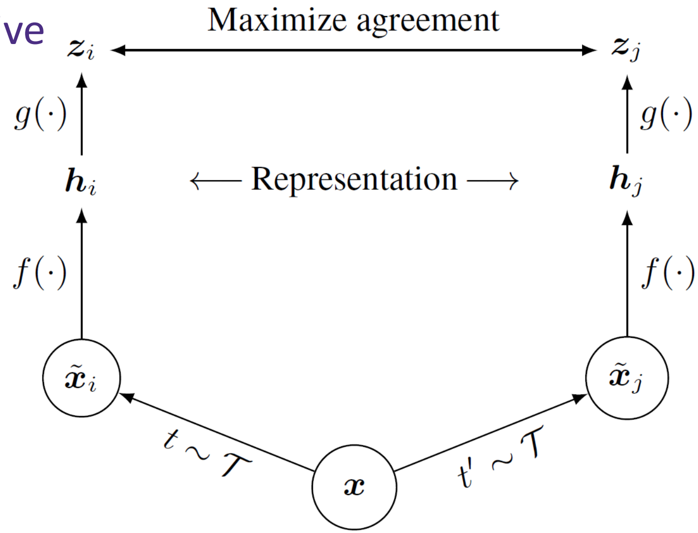
Figure 3. **Comparison of three contrastive loss mechanisms** under the ImageNet linear classification protocol. We adopt the same pretext task (Sec. 3.3) and only vary the contrastive loss mechanism (Figure 2). The number of negatives is K in memory bank and MoCo, and is $K-1$ in end-to-end (offset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

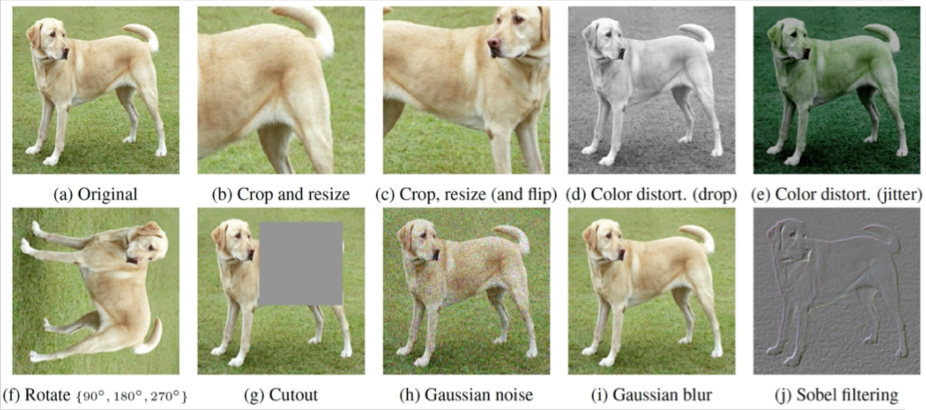
- SimCLR (Chen et al. ’20)
  - A simple framework for contrastive learning of visual representations
    - Predefine a set of transformations rotation, flips
    - For a data, sample two transformations
    - Maximum agreement on representations
  - No negative pairs explicitly
    - Non-paired data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



Contrastive learning

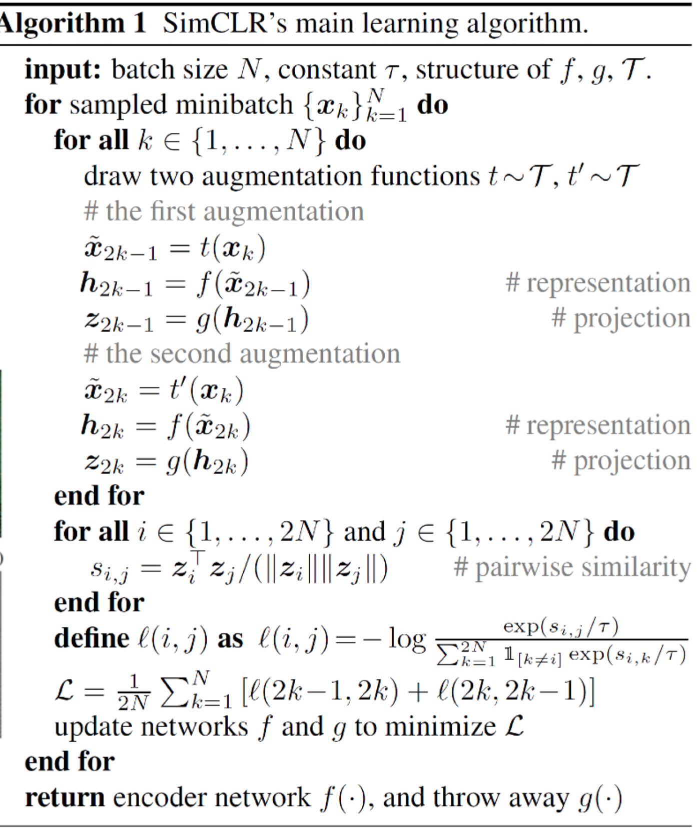
Contrastive Predictive Coding (Van den Oord et al., ’18)

- SimCLR (Chen et al. ’20)



(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



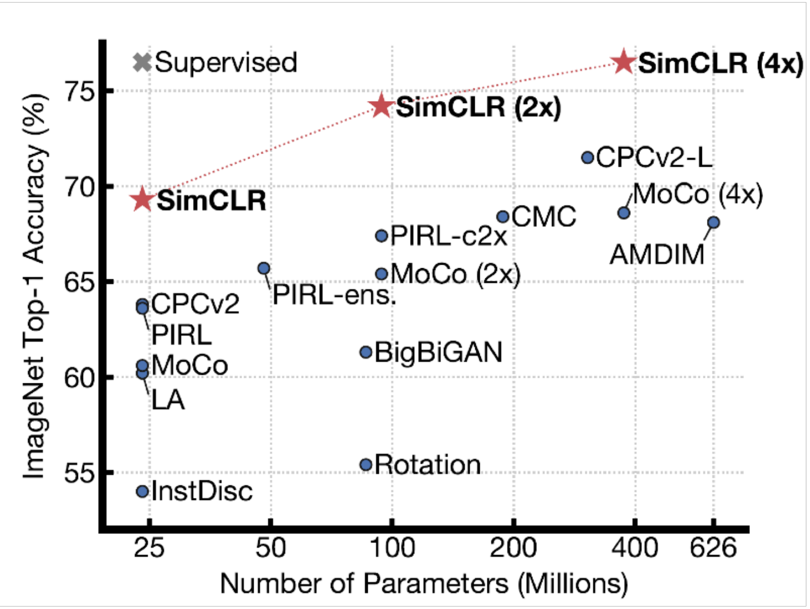
Algorithm 1 SimCLR’s main learning algorithm.

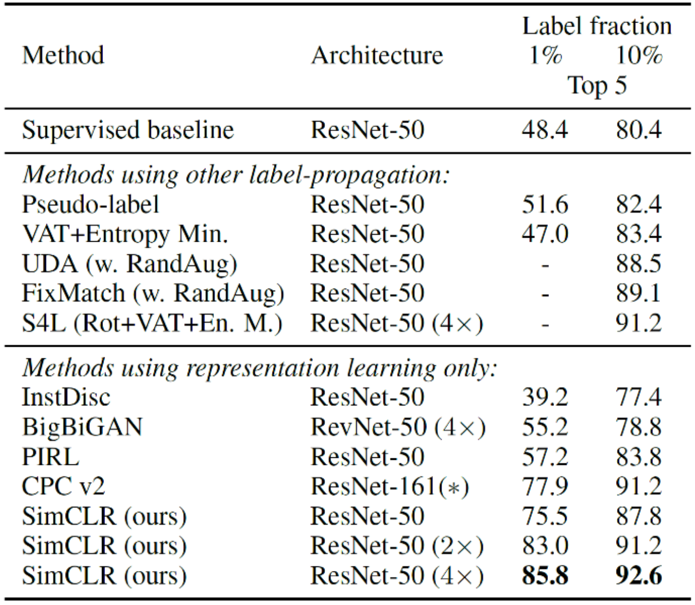
```
input: batch size $N$, constant $\tau$, structure of $f$, $g$, $\mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^N$ do
    for all $k \in \{1, \ldots, N\}$ do
        draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
        # the first augmentation
        $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
        $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$                # 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 $g(\cdot)$
```

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.

Summary

- 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
  - …