Representation Learning
Pre-training

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:

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

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

τ: temperature

$\tau \to 0$

even trainable

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

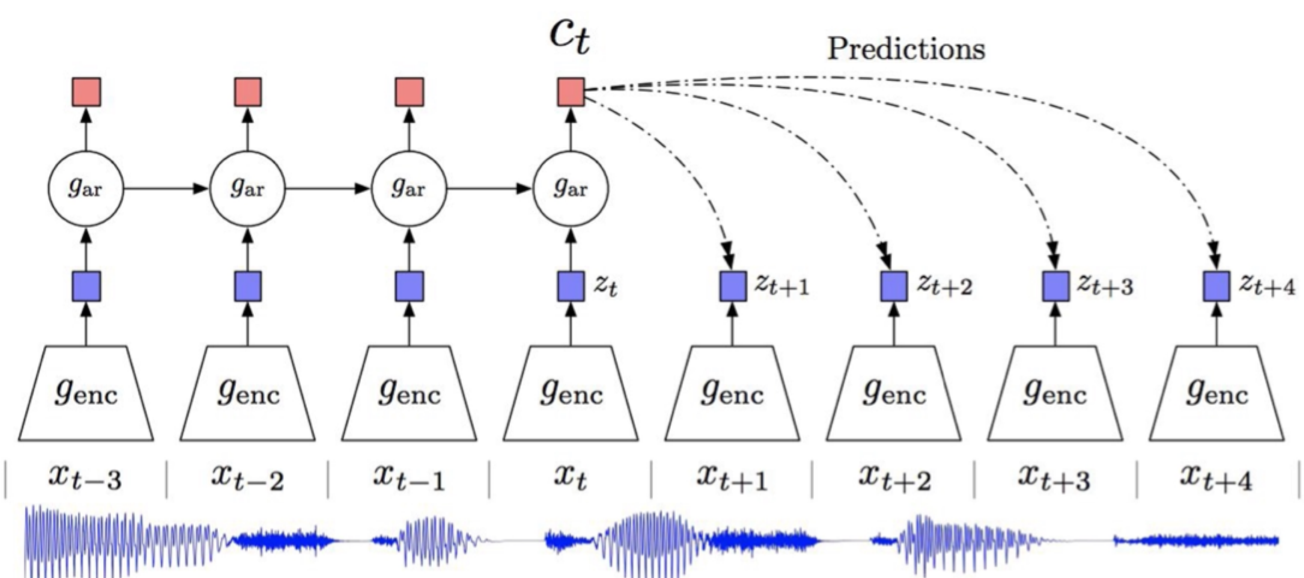


Figure from Alex Graves

hidden state after k steps from t

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

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

data set

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

negative mining

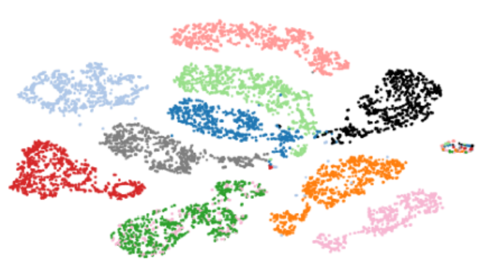


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

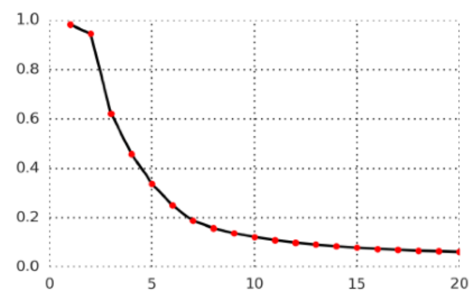


Figure 3: Average accuracy of predicting the positive sample in the contrastive loss for 1 to 20 latent steps in the future of a speech waveform. The model predicts up to 200ms in the future as every step consists of 10ms of audio.

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.

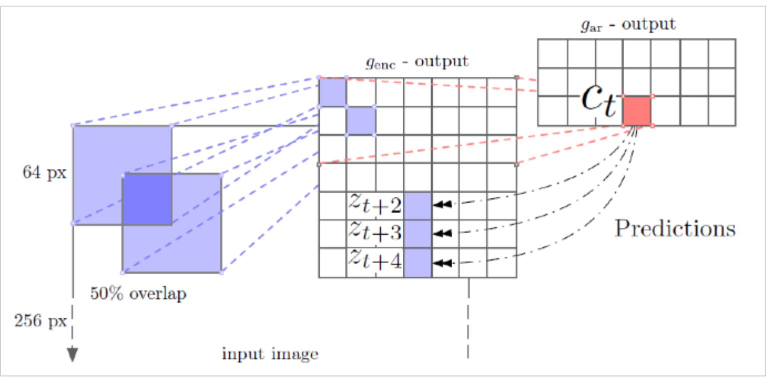
Method	ACC
#steps predicted	
2 steps	28.5
4 steps	57.6
8 steps	63.6
12 steps	64.6
16 steps	63.8
Negative samples from	
Mixed speaker	64.6
Same speaker	65.5
Mixed speaker (excl.)	57.3
Same speaker (excl.)	64.6
Current sequence only	65.2

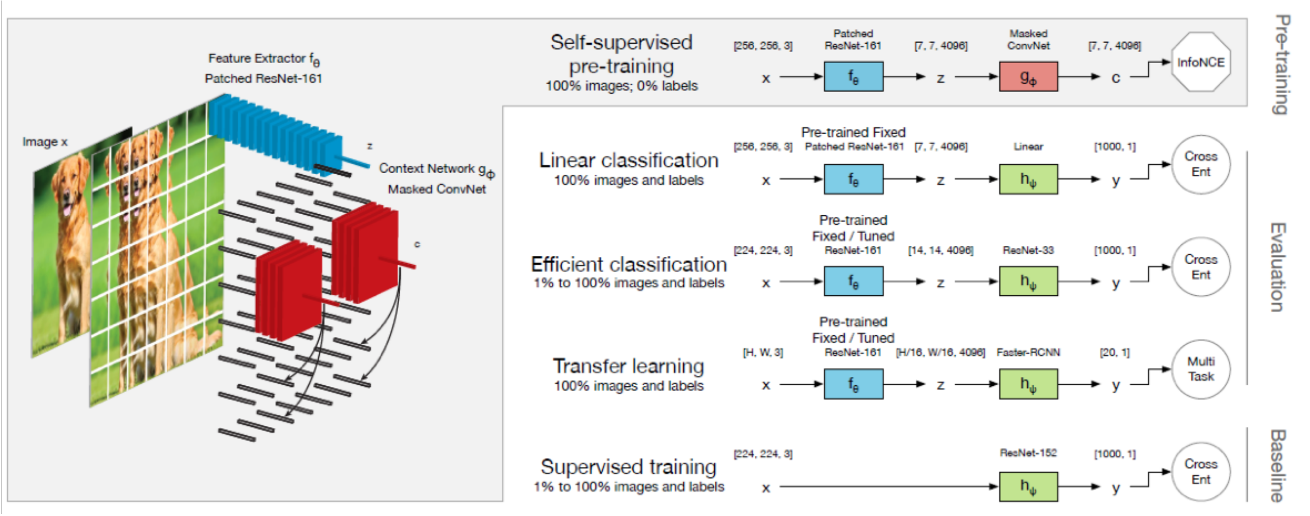
Table 2: LibriSpeech phone classification ablation experiments. More details can be found in Section 3.1.

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

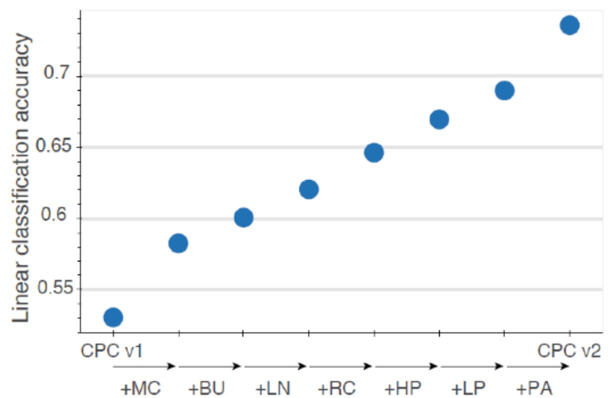
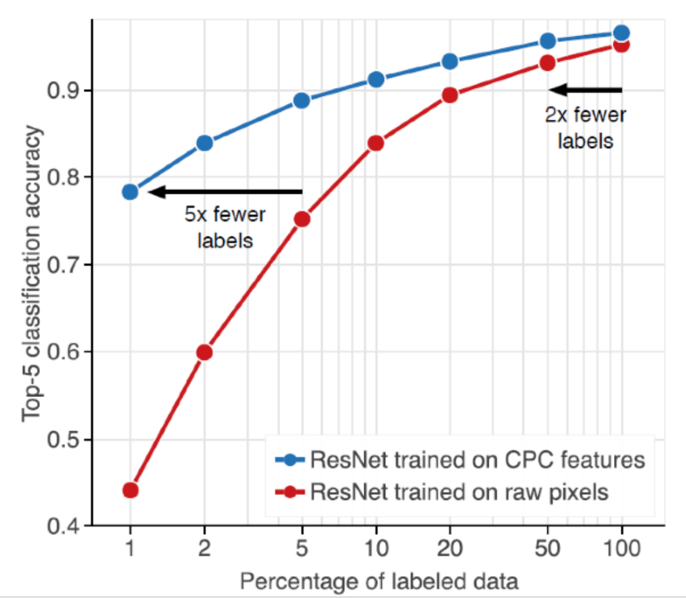


Figure 3. Linear classification performance of new variants of CPC, which incrementally add a series of modifications. MC: model capacity. BU: bottom-up spatial predictions. LN: layer normalization. RC: random color-dropping. HP: horizontal spatial predictions. LP: larger patches. PA: further patch-based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

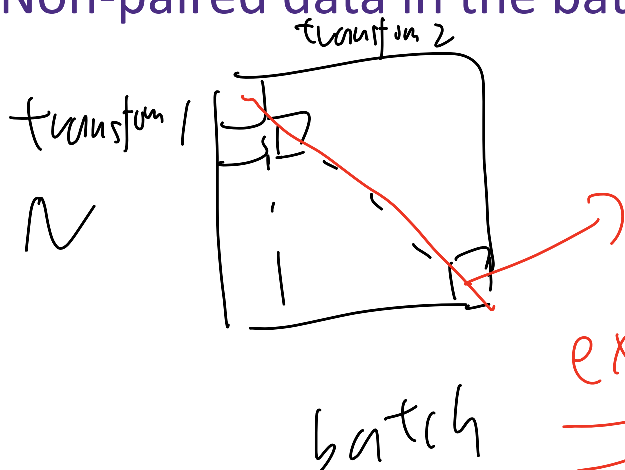
Method	Params (M)	Top-1	Top-5
Methods using ResNet-50:			
Instance Discr. [1]	24	54.0	-
Local Aggr. [2]	24	58.8	-
MoCo [3]	24	60.6	-
PIRL [4]	24	63.6	-
CPC v2 - ResNet-50	24	**63.8**	**85.3**
Methods using different architectures:			
Multi-task [5]	28	-	69.3
Rotation [6]	86	55.4	-
CPC v1 [7]	28	48.7	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

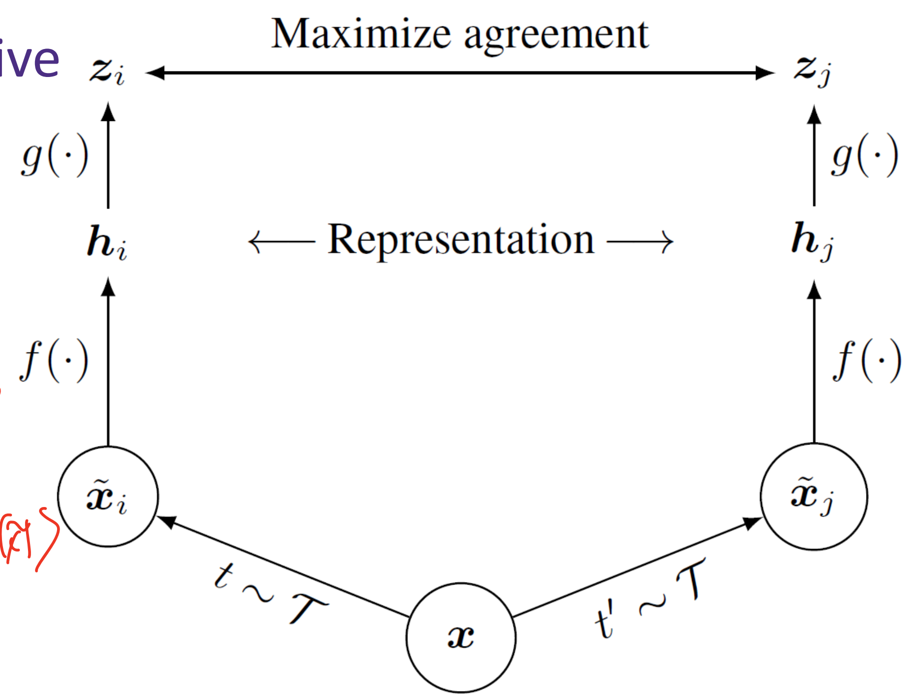
Contrastive Predictive Coding (Van den Oord et al., '18)

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



maximize

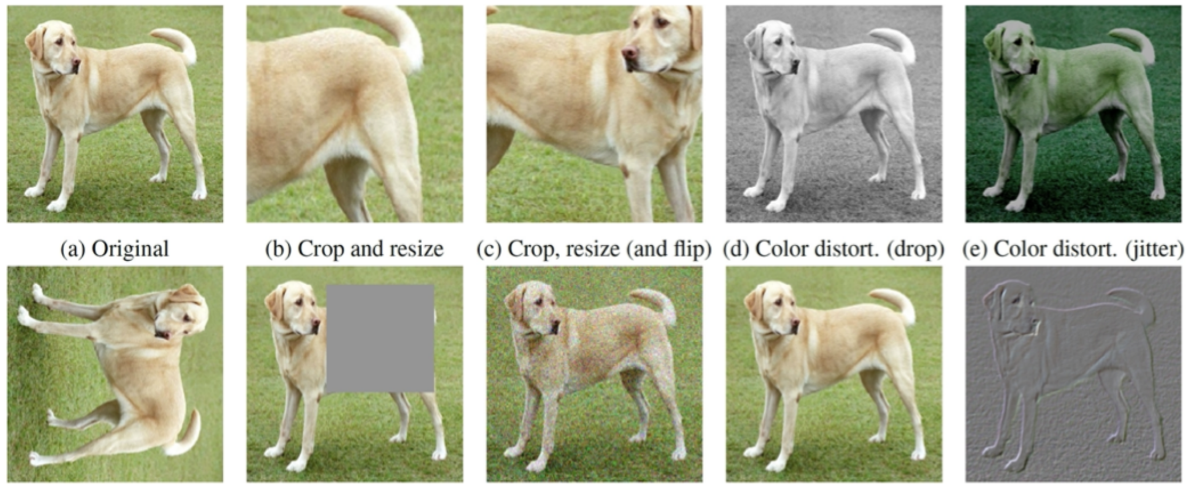
$$\frac{\exp(E(f(x), f(x')))}{\sum_{\tilde{x} \in batch} \exp(E(x, f(\tilde{x})))}$$



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

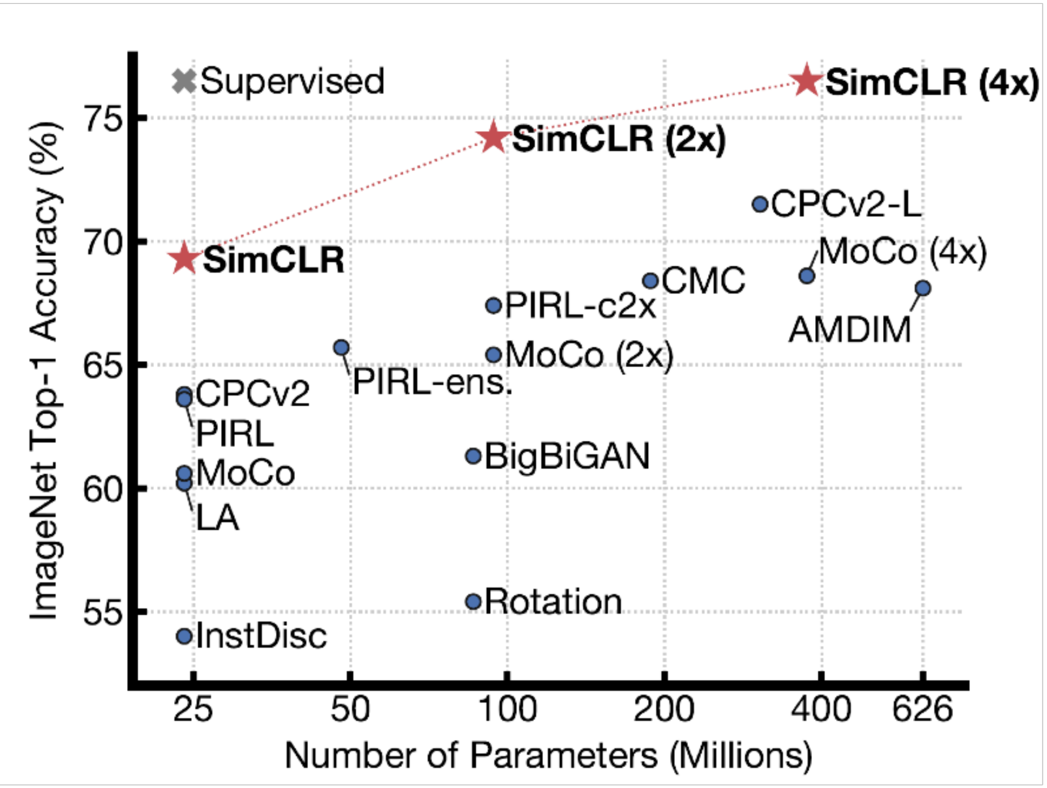
Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant 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)



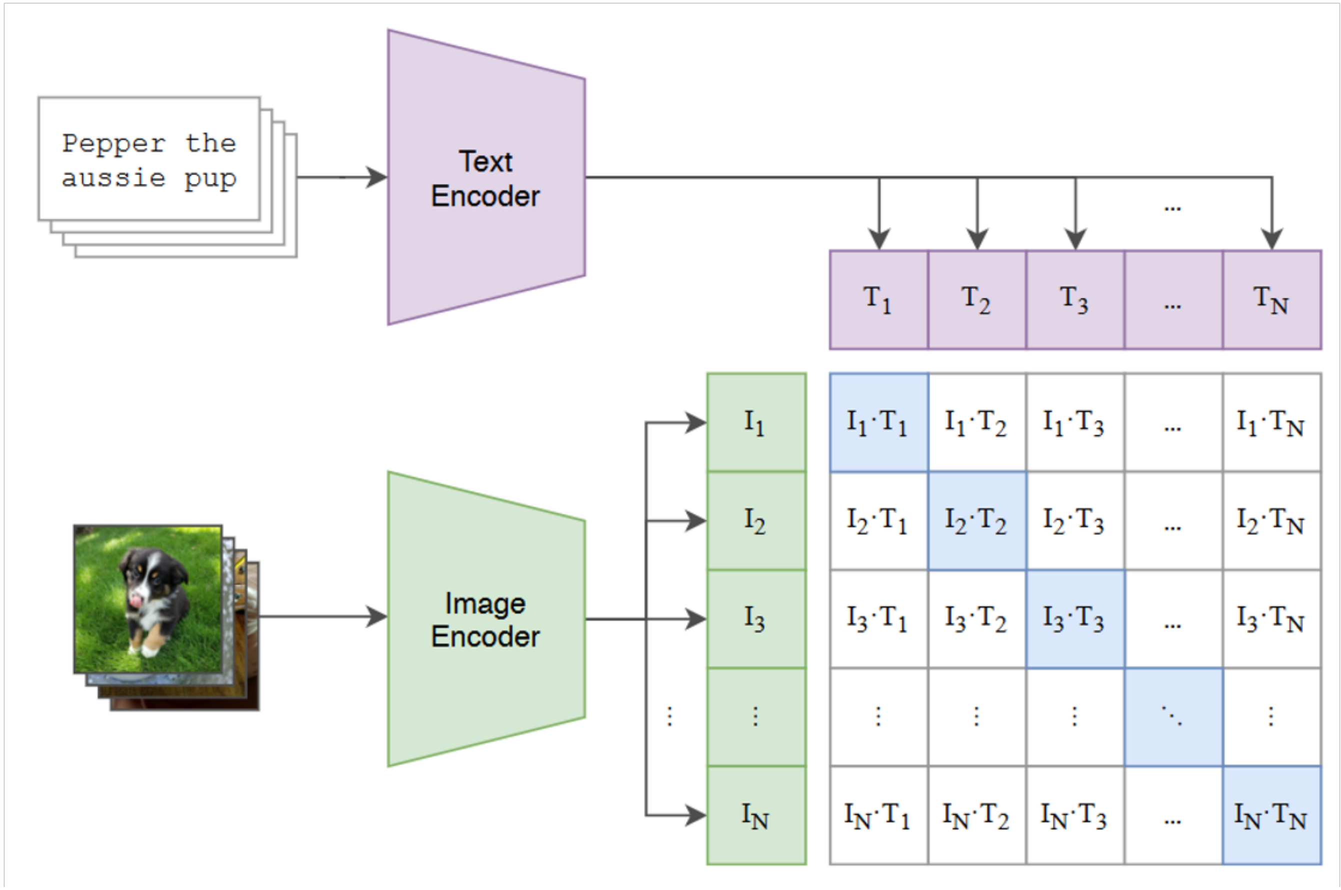
Method	Architecture	Label fraction 1%	10%
		Top 5	
Supervised baseline	ResNet-50	48.4	80.4
Methods using other label-propagation:			
Pseudo-label	ResNet-50	51.6	82.4
VAT+Entropy Min.	ResNet-50	47.0	83.4
UDA (w. RandAug)	ResNet-50	-	88.5
FixMatch (w. RandAug)	ResNet-50	-	89.1
S4L (Rot+VAT+En. M.)	ResNet-50 (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.

Multimodal Contrastive Learning

$\{(\text{image}_i, \text{text}_i)\}_{i=1}^{N}$

Contrastive Pretraining:
Train image and text representation together



Multimodal Contrastive Learning

I_i: image encoding for image i

Sample a batch size N

T_j: text encoding for text j

Loss function

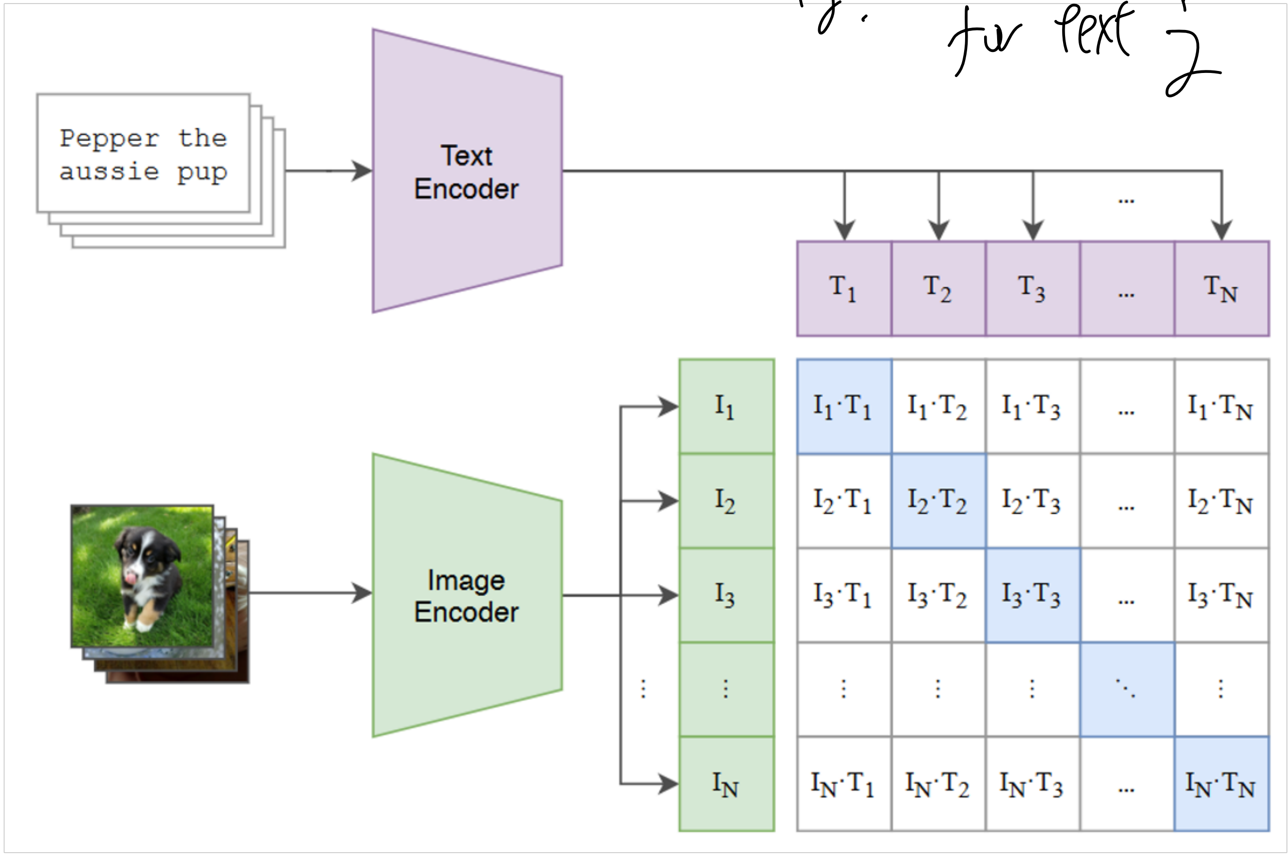
Let $q_{ij} := I_i^\top T_j$ (normalized embeddings: $||I_i||_2 = ||T_j||_2 = 1$),

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

where,

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

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



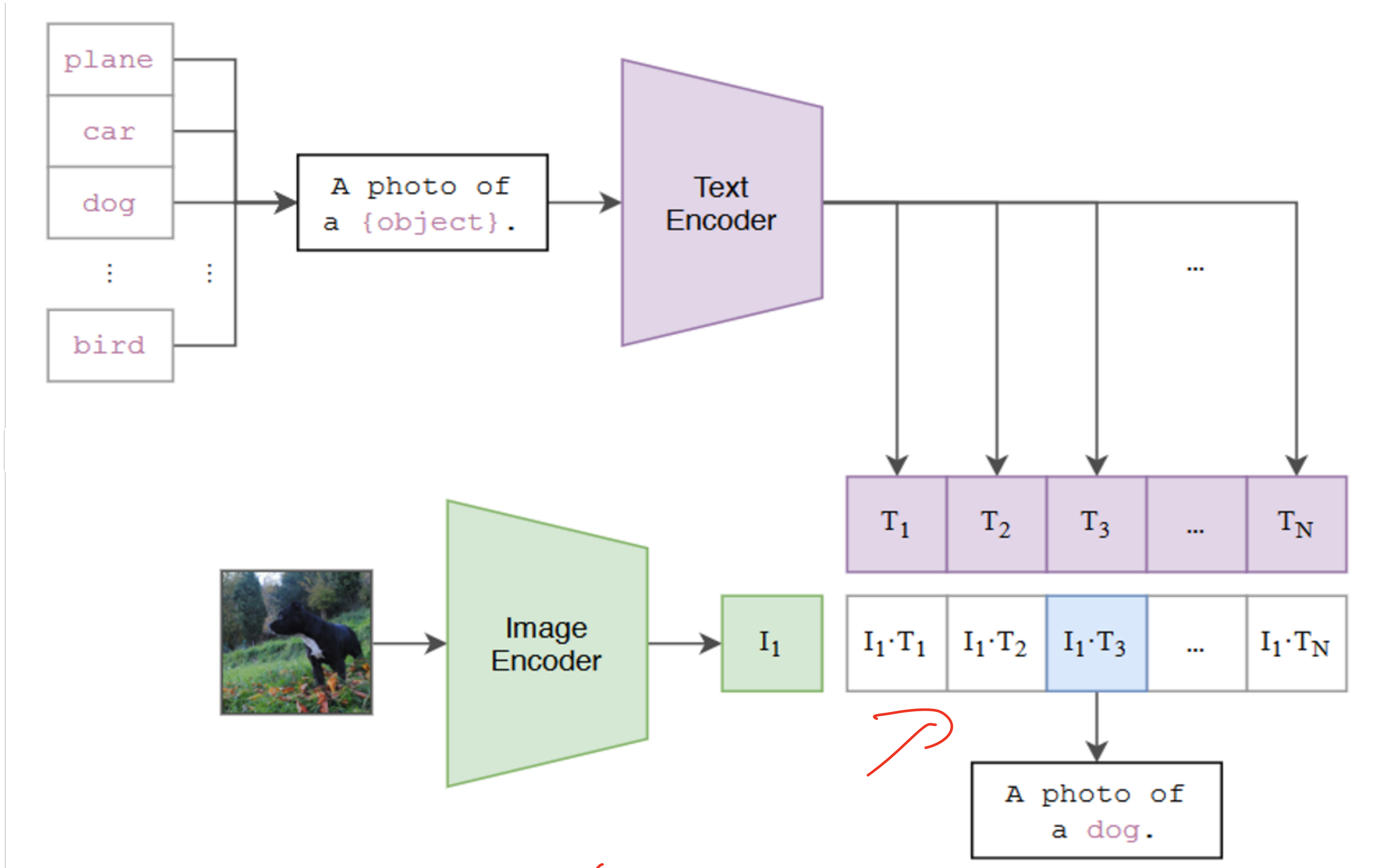
$\forall i$, (I_i, T_i) should have high similarity

$j \neq i$ (I_i, T_j) should have low similarity

Multimodal Contrastive Learning

Zero-Shot Classification:

- Generate a prompt for each class

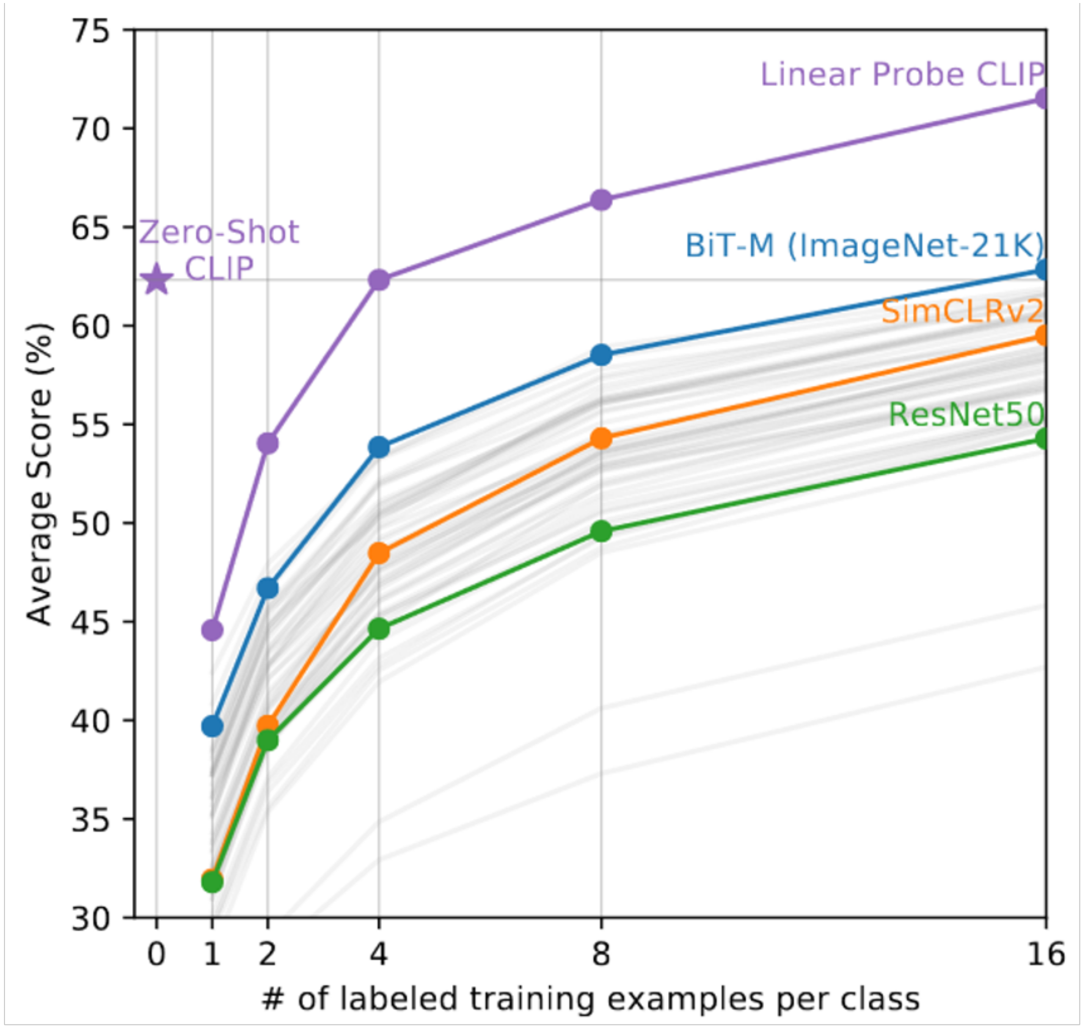


choose i with max $I_1 \cdot T_i$

Multimodal Contrastive Learning

Results

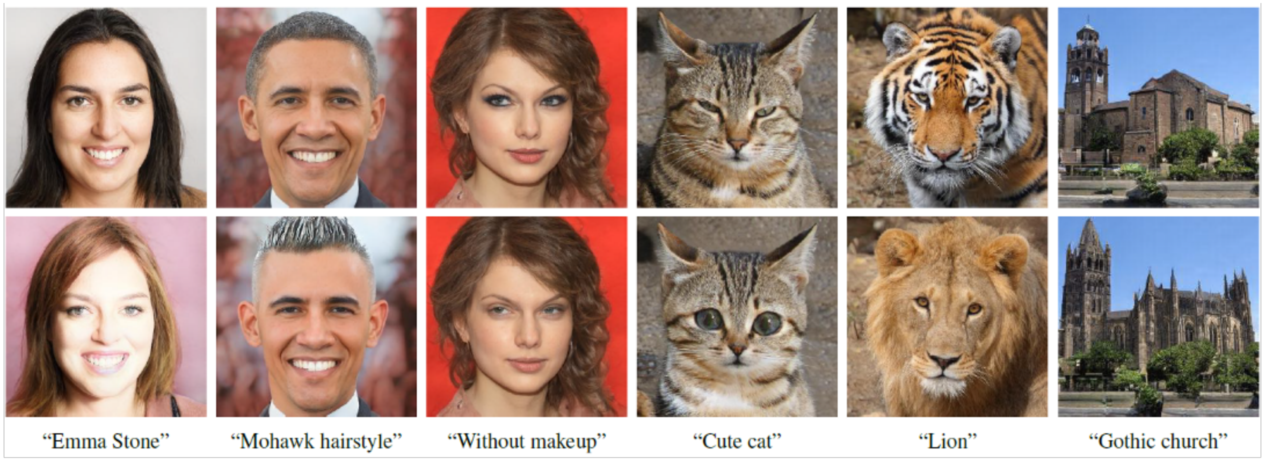
- Strong zero-shot and few-shot performance compared with other models.
- Zero-shot performance on **ImageNet**: CLIP ≈ fully supervised ResNet50!



Applications of CLIP

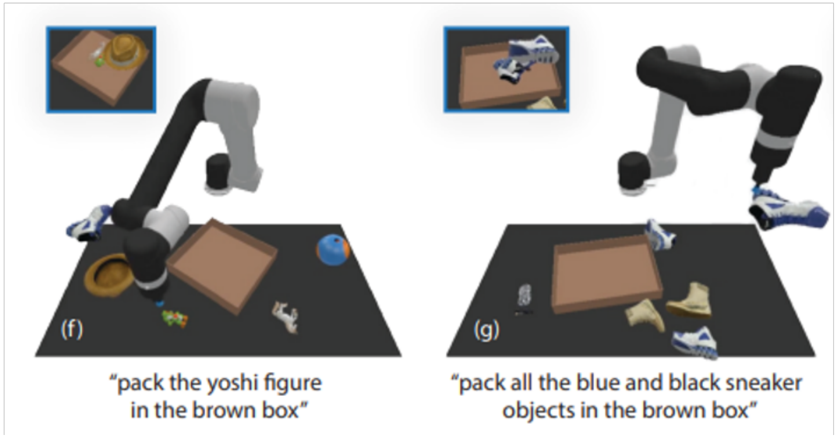
Image Generation

(StyleCLIP [Patashnik et al. 2021])



Robotics

(CLIPort [Shridhar et al. 2021])



...

Problems about Training CLIP

(image, text)

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

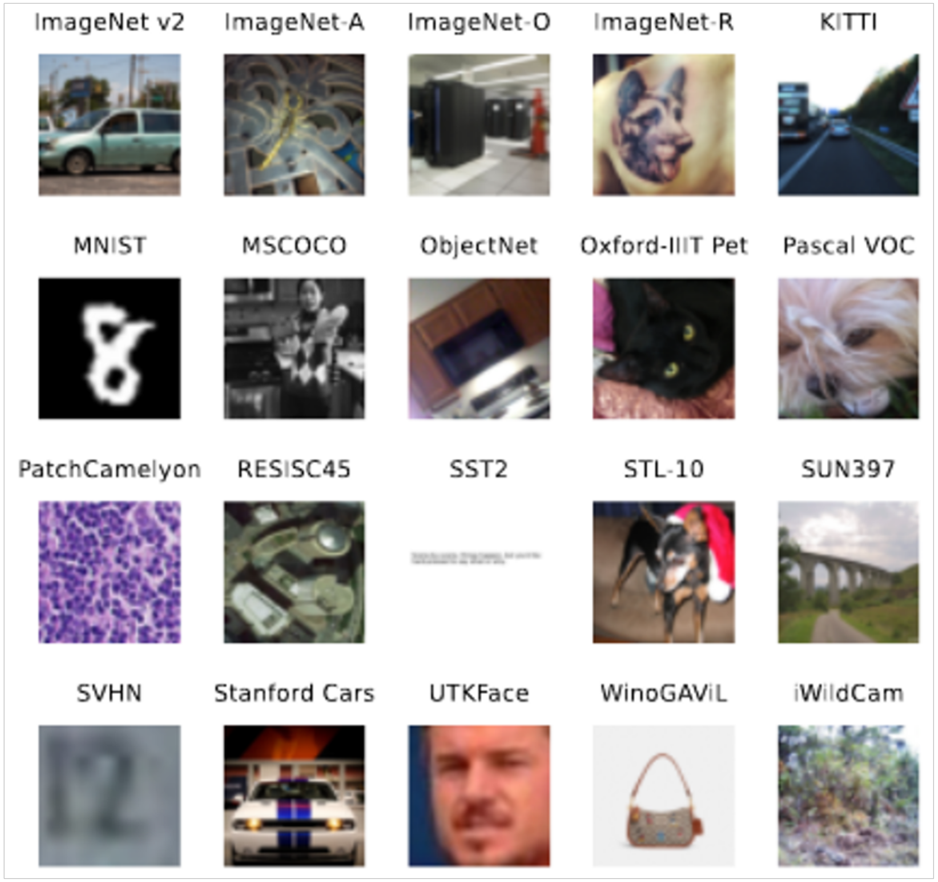
A benchmark standardize the training configuration

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

use a teacher: e.g. CLIP

Distribution-agnostic methods

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

CLIP score filtering

$\bar{f}$:

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

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

select large CLIP scores

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 $\cap$ 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!

Parameter-Efficient Fine-Tuning

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

GPT-3:
175 B

Fine-tune
all parameters
in model
=> too costly

pre-trained
W

Fine-tuned model
W + AB
=> do not
fine-tune W
only fine-tune
A & B
$A \in \mathbb{R}^{r \times d}$
$B: \mathbb{R}^{d \times r}$
$r \ll d$

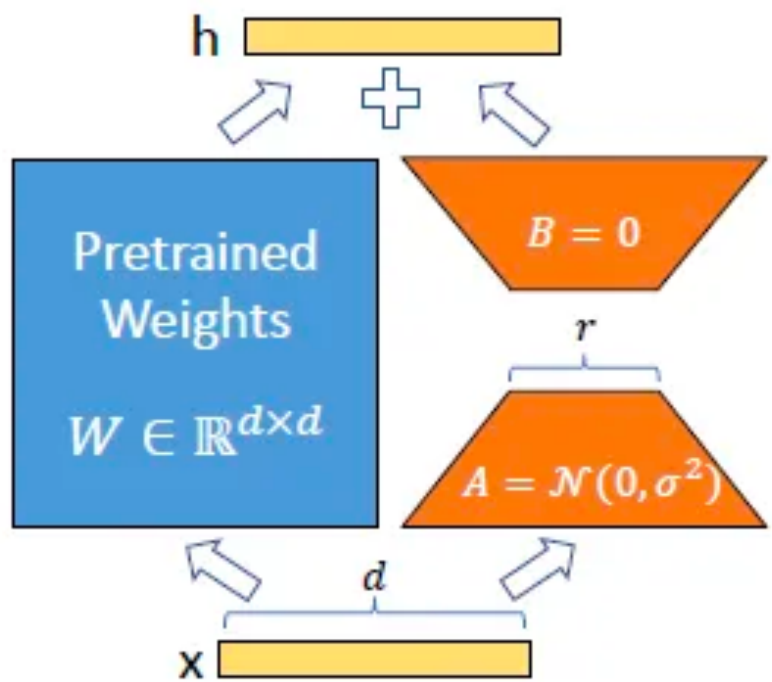


Figure 1: Our reparametrization. We only train A and B.

auto-regressive model

$\in$ Generative models

Generative Models

Distribution learning

Training Data(CelebA)

Model Samples (Karras et.al., 2018)

4 years of progression on Faces

Brundage et al., 2017

Image credits to Andrej Risteski

Distribution learning

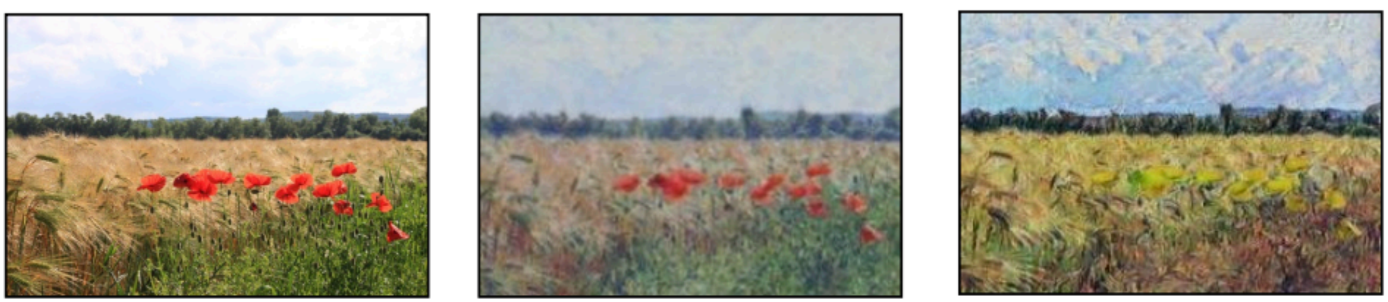
BigGAN, Brock et al '18

Distribution learning

generate an image from text prompt

Conditional generative model P(zebra images| horse images)

Style Transfer



Input Image Monet Van Gogh

Image credits to Andrej Risteski

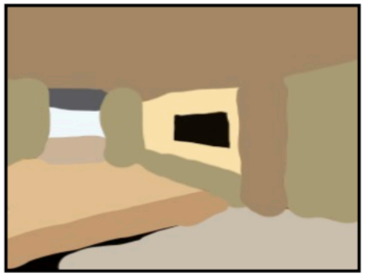
Distribution learning

Source actor

Target actor

Real-time reenactment

Generative model



Generate

Generative model
of realistic images

Stroke paintings to realistic images
[Meng, He, Song, et al., ICLR 2022]

"Ace of Pentacles"

Generate

Generative model
of paintings

Language-guided artwork creation
https://chainbreakers.kath.io @RiversHaveWings

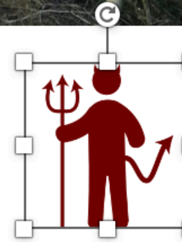
Slides credit to Yang Song

Generative model

High probability

Generative model of traffic signs

Low probability



Outlier detection

[Song et al., ICLR 2018]

Slide credit to Yang Song

Desiderata for generative models

for NLP: auto-regressive model satisfies all 3

p_θ

- **Probability evaluation**: given a sample, it is computationally efficient to evaluate the probability of this sample.

Given x, compute $p_\theta(x)$ efficiently

- **Flexible model family:** it is easy to incorporate any neural network models.

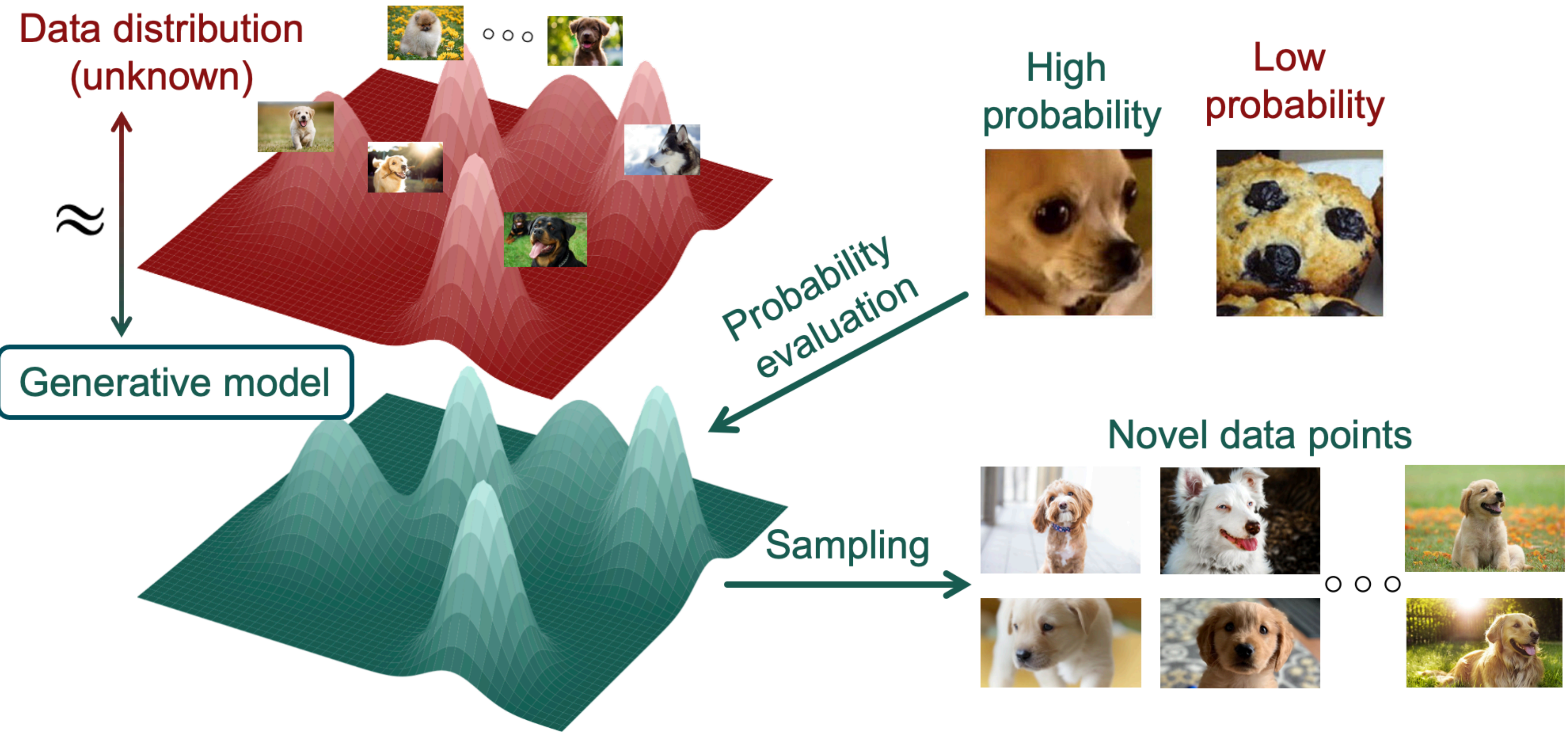
p_θ: CNN, transformer

- **Easy sampling:** it is computationally efficient to sample a data from the probabilistic model.

Given p_θ sample n data points $\Rightarrow \hat{p}_\theta$

$\hat{p}_\theta$ is close to p_θ

Desiderata for generative models



Slide credit to Yang Song

Taxonomy of generative models

given x, explicit expression $P_\theta(x)$

example:
$z \sim N(0, I)$
$x = G_\theta(z)$

- **Generative models**
  - Explicit density
    - Tractable density
      - Fully Visible Belief Nets
        - NADE
        - MADE
        - PixelRNN/CNN
      - Change of variables models:
        - (Nonlinear) ICA
        - Normalizing flows
    - Approximate density
      - Variational
        - Variational Autoencoder
      - Markov Chain
        - Energy models
        - (Restricted) Boltzmann machines
        - score-based
        - diffusion
  - Implicit density
    - Direct
      - GAN
    - Markov Chain
      - GSN

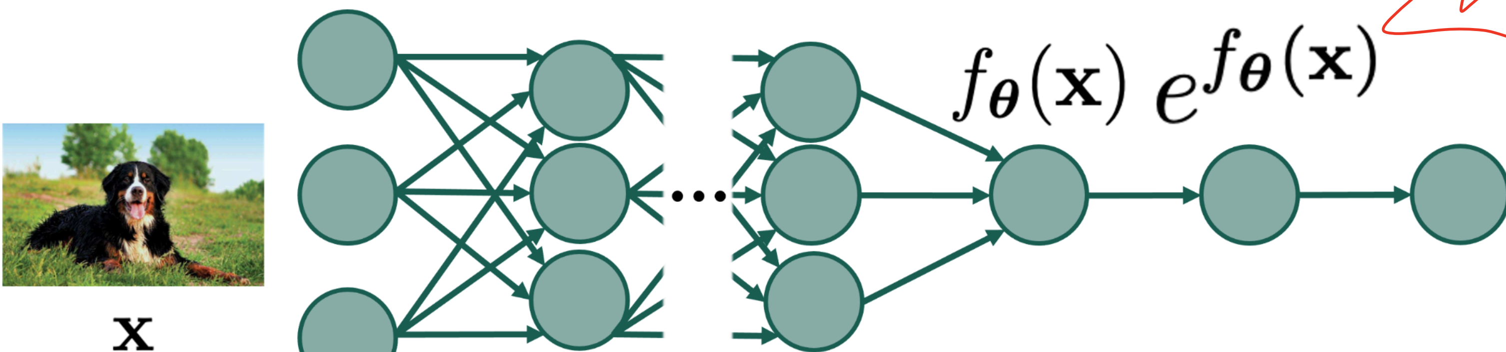
Image credits to Andrej Risteski

Key challenge for building generative models

distribution

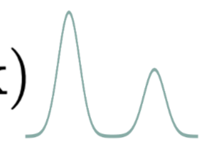
$p_\theta(x) \geq 0$

$\int_x p_\theta(x)\,dx = 1$



$f_\theta(\mathbf{x})$ $e^{f_\theta(\mathbf{x})}$

$$\frac{e^{f_\theta(\mathbf{x})}}{\boxed{Z_\theta}} = p_\theta(\mathbf{x})$$



Normalizing constant

$$Z_\theta = \sum_x e^{f_\theta(x)}$$

or $\int_x e^{f_\theta(x)}\,dx$

Slide credit to Yang Song

Key challenge for building generative models

Approximating the normalizing constant

- Variational auto-encoders [Kingma & Welling 2014, Rezende et al. 2014]
- Energy-based models [Ackley et al. 1985, LeCun et al. 2006]

✕ Inaccurate probability evaluation

Using restricted neural network models

- Autoregressive models [Bengio & Bengio 2000, van den Oord et al. 2016]
- Normalizing flow models [Dinh et al. 2014, Rezende & Mohamed 2015]

✕ Restricted model family

Generative adversarial networks (GANs)

- Model the generation process, not the probability distribution [Goodfellow et al. 2014]

✕ Cannot evaluate probabilities

Slide credit to Yang Song

Training generative models

- **Likelihood-based:** maximize the likelihood of the data under the model (possibly using advanced techniques such as variational method or MCMC):

$$\max_{\theta} \sum_{i=1}^{n} \log p_{\theta}(x_i)$$

- Pros:
  - **Easy training**: can just maximize via SGD.
  - **Evaluation**: evaluating the fit of the model can be done by evaluating the likelihood (on test data).

- Cons:
  - **Large models needed**: likelihood objectve is hard, to fit well need very big model.
  - **Likelihood entourages averaging:** produced samples tend to be blurrier, as likelihood encourages "coverage" of training data.

Training generative models

- **Likelihood-free:** use a **surrogate loss** (e.g., GAN) to train a discriminator to differentiate real and generated samples.

- Pros:
  - **Better objective, smaller models needed:** objective itself is learned - can result in visually better images with smaller models.

- Cons:
  - **Unstable training:** typically min-max (saddle point) problems.
  - **Evaluation:** no way to evaluate the quality of fit.