Natural Language Processing

Multimodal NLP: Contrastive Methods, CLIP

Prof. Iacopo Masi and Prof. Stefano Faralli

```
import matplotlib.pyplot as plt
import scipy
import random
import numpy as np
import pandas as pd
pd.set_option('display.colheader_justify', 'center')
```

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
#plt.style.use('seaborn-whitegrid')

font = {'family' : 'Times',
        'weight' : 'bold',
        'size'   : 12}

matplotlib.rc('font', **font)


# Aux functions

def plot_grid(Xs, Ys, ax=None):
    ''' Aux function to plot a grid'''
    t = np.arange(Xs.size) # define progression of int for indexing colormap
    if ax:
        ax.plot(0, 0, marker='+', color='r', linestyle='none') #plot origin
        ax.scatter(Xs,Ys, c=t, cmap='jet', marker='.') # scatter x vs y
        ax.axis('scaled') # axis scaled
    else:
        plt.plot(0, 0, marker='+', color='r', linestyle='none') #plot origin
        plt.scatter(Xs,Ys, c=t, cmap='jet', marker='.') # scatter x vs y
        plt.axis('scaled') # axis scaled

def linear_map(A, Xs, Ys):
    '''Map src points with A'''
    # [NxN,NxN] <- NxNx2 # add 3-rd axis, like adding another layer
    src = np.stack((Xs,Ys), axis=Xs.ndim)
    # flatten first two dimension
    # (NN)x2
    src_f = src.reshape(-1,src.shape[-1]) #ask reshape to keep last dimension and adjust the rest
    # 2x2 @ 2x(NN)
    dst = A @ src_f.T # 2xNN
    #(NN)x2 and then reshape as NxNx2
    dst = (dst.T).reshape(src.shape)
    # Access X and Y
    return dst[...,0], dst[...,1]


def plot_points(ax, Xs, Ys, col='red', unit=None, linestyle='solid'):
    '''Plots points'''
    ax.set_aspect('equal')
    ax.grid(True, which='both')
    ax.axhline(y=0, color='gray', linestyle='--')
    ax.axvline(x=0, color='gray',  linestyle='--')
    ax.plot(Xs, Ys, color=col)
    if unit is None:
        plotVectors(ax, [[0,1],[1,0]], ['gray']*2, alpha=1, linestyle=linestyle)
    else:
        plotVectors(ax, unit, [col]*2, alpha=1, linestyle=linestyle)

def plotVectors(ax, vecs, cols, alpha=1, linestyle='solid'):
    '''Plot set of vectors.'''
    for i in range(len(vecs)):
        x = np.concatenate([[0,0], vecs[i]])
        ax.quiver([x[0]],
                  [x[1]],
                  [x[2]],
                  [x[3]],
                  angles='xy', scale_units='xy', scale=1, color=cols[i],
                  alpha=alpha, linestyle=linestyle, linewidth=2)
```

My own latex definitions

This week lectures

- Generative vs Discriminative Models

- Contrastive Methods

- Multimodal NLP: NLP as supervision for the Visual domain

- CLIP

- unCLIP (Dall-E)

This lecture material is taken from

Mainly from research papers

Contrastive methods

- A Simple Framework for Contrastive Learning of Visual Representations **SimCLR** (vision)
- Sampling Matters in Deep Embedding Learning **Contrastive method: geometric(non-probabilistic** (vision)
- Learning Visual Features from Large Weakly Supervised Data **CLIP precursor (vision/nlp)**
- Learning Visual Features from Large Weakly Supervised Data (slides) (vision/nlp)
- Contrastive Learning of Medical Visual Representations from Paired Images and Text **CLIP idea is from this (medical field)** (vision/medical)
- Learning Transferable Visual Models From Natural Language Supervision **CLIP paper** (vision/nlp)
- Hierarchical Text-Conditional Image Generation with CLIP Latents **unCLIP Dalle-2** (vision/nlp)

Non-contrastive methods

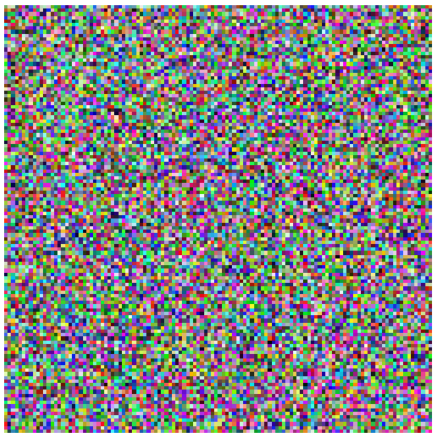
- Exploring Simple Siamese Representation Learning **SimSiam** (vision)
- Bootstrap Your Own Latent A New Approach to Self-Supervised Learning **BYOL** (vision)

Generative vs Discriminative Models in Machine Learning

Learning from complex distributions is in common to vision/NLP (curse of dimensionality)

Vision: high-dimensional, continuous data

$$p(\mathbf{x})$$



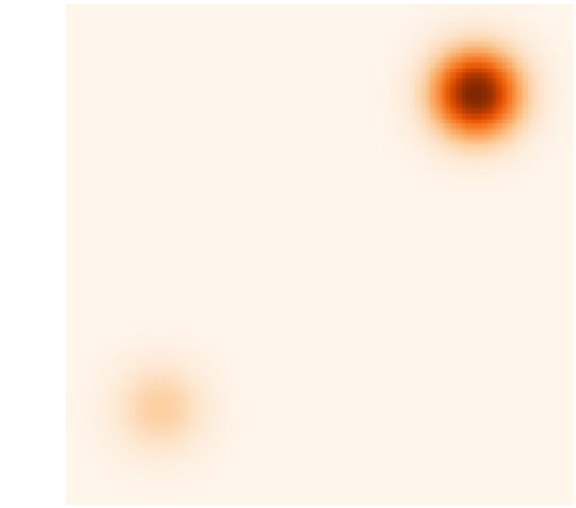
NLP: combinatorial/compositional problem of discrete symbols

$$p(w_1, \ldots, w_n)$$

```
pot and enough deep secret rat thunder black industry answer death material angle crime probable nation debt organization spade acid soup reward free circle west forward board bone
substance parcel south scissors move window hanging needle sticky pipe table old river attack design expert
```

Unknown density of the data

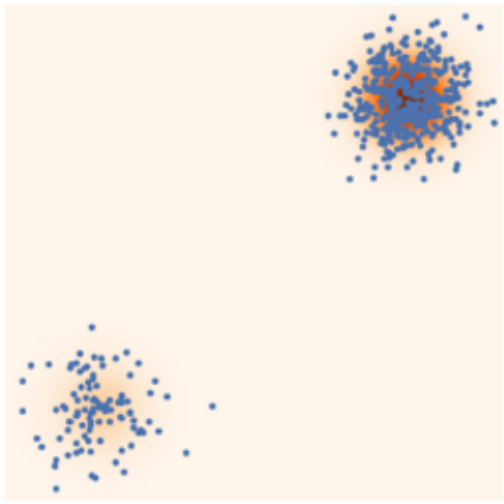
$p_{data(\mathbf{x})}$



Picture from [Yang-Song Blog](https://yang-song.net/blog/2021/score/)

Known data samples dataset

$\{\mathbf{x}_i\} \sim p_{data(\mathbf{x})}$



Picture from [Yang-Song Blog](https://yang-song.net/blog/2021/score/)

Generative vs Discriminative Models

Generative models objective is to learn an approximation of the data density given the data samples (dataset):

$$p_{data}(\mathbf{x}) \approx p_\theta(\mathbf{x})$$

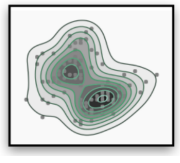
$$p_{data}(\mathbf{x}, y) \approx p_\theta(\mathbf{x}, y)$$

Discriminative models objective is to learn a decision boundary to "separate" data samples to perform classification.

$$p(y == \text{class } k|\mathbf{x})$$

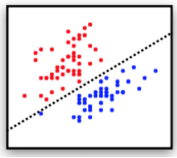
Generative

$$\log p(\mathbf{x})$$



Discriminative

$$\log p(y|\mathbf{x})$$



Generative and Discriminative are Connected

Think **"classes" as latent factors** in the data--the class label just "reveals" the latent factor. [There could be, there are others]. Think as "slicing the data density" using the latent factor ("coloring" the data density).

$$\log p(\mathbf{x}, y)$$

$$p_\theta(\mathbf{x}) = \sum_{y'} p_\theta(\mathbf{x}, y') \qquad \text{marginalization}$$

$$p_\theta(\mathbf{x}, y) = \underbrace{p(\mathbf{x}|y)}_{\text{class-cond. density}} \quad \underbrace{p(y)}_{} = \underbrace{p(y|\mathbf{x})}_{\text{discriminative}} \quad \underbrace{p(\mathbf{x})}_{} \qquad \text{product rule}$$

Generative and Discriminative are Connected

$$\underbrace{p(y|\mathbf{x})}_{\text{discriminative}} \quad \underbrace{p(\mathbf{x})}_{} = p_\theta(\mathbf{x}, y) = \underbrace{p(\mathbf{x}|y)}_{\text{generative}} \underbrace{p(y)}_{}$$

$$\underbrace{p(y|\mathbf{x})}_{\text{discriminative}} = \frac{p_\theta(\mathbf{x}, y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{\sum_{y'} p(\mathbf{x}|y')p(y')}$$

From Generative go Discriminative

If you need to do just classification (and do not need probabilities):

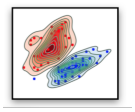
$$\arg\max_{y'} p(y'|\mathbf{x})$$

Then you can:

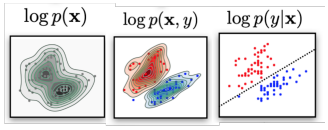
1. Model/Estimate the class conditional data density $p(\mathbf{x}|y)$ - *think of the data density "restricted to one latent factor, a class"*.
2. Estimate how much do you think a latent factor (class) is probable $p(y)$
3. Classify as:

$$\arg\max_{y'} p(\mathbf{x}|y')p(y') \qquad \text{Given that} \quad p(y|\mathbf{x}) \propto p(\mathbf{x}|y)p(y)$$

$$\log p(\mathbf{x}, y)$$



$$\log p(\mathbf{x}) \qquad \log p(\mathbf{x}, y) \qquad \log p(y|\mathbf{x})$$



Picture from [YOUR CLASSIFIER IS SECRETLY AN ENERGY BASED MODEL AND YOU SHOULD TREAT IT LIKE ONE] (https://openreview.net/pdf?id=Hkxzx0NtDB)

Where is the connection with NLP?

Where is the connection with NLP?

LM (language modeling) is about learning $p(w_1, \ldots, w_t)$ (thus **generative**) but it is implemented as **a discriminative classifier** that predicts a pmf of w_{t+1} given $w_1, \ldots, w_t$.

$$p(w_1, \ldots, w_t) \to \prod_{i=1}^{T} p(w_{i+1}|w_1, \ldots, w_i)$$

Do autoregressive models exist in vision?

Conditional Image Generation with PixelCNN Decoders

Aäron van den Oord
Google DeepMind
avdnoord@google.com

Nal Kalchbrenner
Google DeepMind
nalk@google.com

Oriol Vinyals
Google DeepMind
vinyals@google.com

Lasse Espeholt
Google DeepMind
espeholt@google.com

Alex Graves
Google DeepMind
gravesa@google.com

Koray Kavukcuoglu
Google DeepMind
korayk@google.com

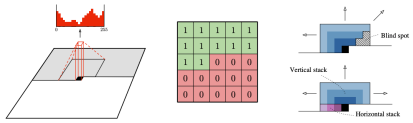


Figure 1: **Left**: A visualization of the PixelCNN that maps a neighborhood of pixels to prediction for the next pixel. To generate pixel x_i the model can only condition on the previously generated pixels $x_1, \ldots, x_{i-1}$. **Middle**: an example matrix that is used to mask the 5x5 filters to make sure the model cannot read pixels below (or strictly to the right) of the current pixel to make its predictions. **Right**: Top: PixelCNNs have a *blind spot* in the receptive field that can not be used to make predictions. Bottom: Two convolutional stacks (blue and purple) allow to capture the whole receptive field.

2.1 Gated Convolutional Layers

PixelRNNs, which use spatial LSTM layers instead of convolutional stacks, have previously been shown to outperform PixelCNNs as generative models [30]. One possible reason for the advantage is that the recurrent connections in LSTM allow every layer in the network to access the entire neighbourhood of previous pixels, while the region of the neighbourhood available to pixelCNN grows linearly with the depth of the convolutional stack. However this shortcoming can largely be alleviated by using sufficiently many layers. Another potential advantage is that PixelRNNs contain multiplicative units (in the form of the LSTM gates), which may help it to model more complex interactions. To amend this we replaced the rectified linear units between the masked convolutions in the original pixelCNN with the following gated activation unit:

$$\mathbf{y} = \tanh(W_{k,f} * \mathbf{x}) \odot \sigma(W_{k,g} * \mathbf{x}), \tag{2}$$

where σ is the sigmoid non-linearity, k is the number of the layer, $\odot$ is the element-wise product and $*$ is the convolution operator. We call the resulting model the Gated PixelCNN. Feed-forward neural networks with gates have been explored in previous works, such as highway networks [25], grid LSTM [13] and neural GPUs [12], and have generally proved beneficial to performance.

Contrastive Methods

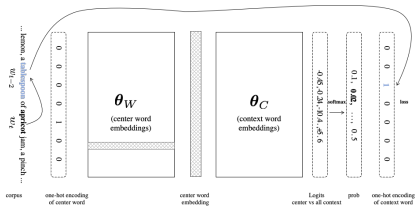
Can you remember in which part of the course we have seen a Contrastive method?

Scaling word2vec with....

Negative Sampling!

word2vec with Skip-Gram at a glance

... and why it can be seen as a tiny neural net.



Skip-gram

Instead of doing:

1. **Center word vs ground-truth context embedding** $\longrightarrow \theta_{c_i}[y] \cdot \theta_W[i]^T$
2. normalize as a distribution: **all context vs center word** $\longrightarrow \sum_{i=1}^{V} \exp\left(\theta_{c_i}[u] \cdot \theta_W[i]^T\right)$

$$\mathcal{L}(w_{t-1}, w_t; \theta) = \underbrace{-\theta_{c_i}[y] \cdot \theta_W[i]^T}_{\text{similarity center vs context}} + \underbrace{\log\left(\sum_{i=1}^{V} \exp\left(\theta_{c_i}[u] \cdot \theta_W[i]^T\right)\right)}_{\text{make sure it is a probability}}$$

Two solutions to approximate the denominator

1. **Negative sampling (Contrastive method)**
2. Hierarchical Softmax (Tree-based solution)

Scaling word2vec with Negative Sampling

positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

Push up positive and push down negatives

$$\min_\theta \underbrace{-\log\sigma\left(\theta_{c_i}[y]^T \theta_W[i]\right)}_{\text{push up positive pairs}} \underbrace{-\sum_{k=1}^{K} \log\left[\sigma\left(-\theta_{c_i}[k]^T \theta_W[i]\right)\right]}_{\text{push down negative pairs}}$$

Visualization

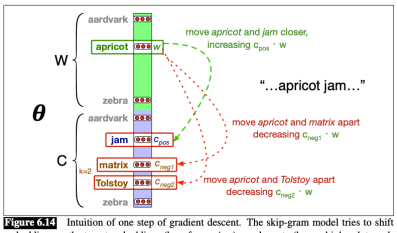
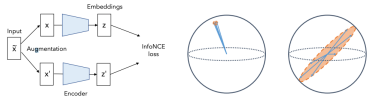


Figure 6.14 Intuition of one step of gradient descent. The skip-gram model tries to shift embeddings so the target embeddings (here for *apricot*) are closer to (have a higher dot product with) context embeddings for nearby words (here *jam*) and further from (lower dot product with) context embeddings for noise words that don't occur nearby (here *Tolstoy* and *matrix*).

Why we need the negatives?



(a) embedding space (b) complete collapse (c) dimensional collapse

Figure 1: Illustration of the collapsing problem. For complete collapse, the embedding vectors collapse to same point. For dimensional collapse, the embedding vectors only span a lower dimensional space.

Other forms of Contrastive Methods

Other forms of Contrastive Methods

1. --Contrastive loss implemented with logistic function and negative sampling [word2vec]-- (nlp)
2. Contrastive loss implemented with Softmax and large mini-batches (vision)
3. Contrastive loss with geometric interpretation and margin (vision)

Contrastive loss implemented with Softmax and large mini-batches

Method name: SimCLR

Vision

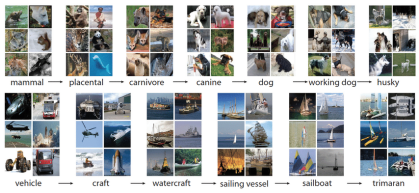
A Simple Framework for Contrastive Learning of Visual Representations

Ting Chen [1] **Simon Kornblith** [1] **Mohammad Norouzi** [1] **Geoffrey Hinton** [1]

ImageNet: A [computer] vision dataset

> ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. The project has been instrumental in advancing computer vision and deep learning research. The data is available for free to researchers for non-commercial use.

ImageNet: A [computer] vision dataset



Source: Ye, Tengqi. 2018. "Visual Object Detection from LifeLogs using Visual Non-lifelog Data." Researchgate, January. Accessed 2019-06-20.

Performance: Supervised vs Self-Supervised

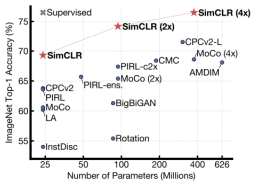


Figure 1. ImageNet Top-1 accuracy of linear classifiers trained on representations learned with different self-supervised methods (pretrained on ImageNet). Gray cross indicates supervised ResNet-50. Our method, SimCLR, is shown in bold.

Take away from the paper

- Composition of multiple data augmentation operations is crucial in defining the contrastive prediction tasks that yield effective representations. In addition, unsupervised contrastive learning benefits from stronger data augmentation than supervised learning.
- Introducing a learnable nonlinear transformation between the representation and the contrastive loss substantially improves the quality of the learned representations.
- Representation learning with contrastive cross entropy loss benefits from normalized embeddings and an appropriately adjusted temperature parameter.
- Contrastive learning benefits from larger batch sizes and longer training compared to its supervised counterpart. Like supervised learning, contrastive learning benefits from deeper and wider networks.

Main Idea

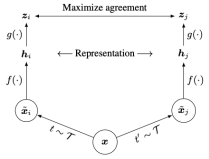


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation $\boldsymbol{h}$ for downstream tasks.

SimCLR Contrastive Loss

We randomly sample a minibatch of N examples and define the contrastive prediction task on pairs of augmented examples derived from the minibatch, resulting in $2N$ data points. We do not sample negative examples explicitly. Instead, given a positive pair, similar to (Chen et al., 2017), we treat the other $2(N-1)$ augmented examples within a minibatch as negative examples. Let $\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v} / \|\boldsymbol{u}\| \|\boldsymbol{v}\|$ denote the dot product between ℓ_2 normalized $\boldsymbol{u}$ and $\boldsymbol{v}$ (i.e. cosine similarity). Then the loss function for a positive pair of examples (i, j) is defined as

Normalized Temperature-scaled Cross-Entropy Loss

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

where $\mathbb{1}_{[k \neq i]} \in \{0, 1\}$ is an indicator function evaluating to 1 iff $k \neq i$ and τ denotes a temperature parameter. The final loss is computed across all positive pairs, both (i, j) and (j, i), in a mini-batch. This loss has been used in previous work (Sohn, 2016; Wu et al., 2018; Oord et al., 2018); for convenience, we term it *NT-Xent* (the normalized temperature-scaled cross entropy loss).

Data Augmentation

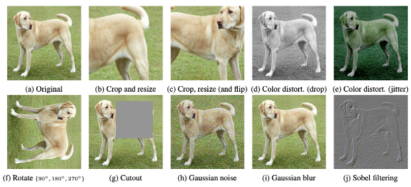


Figure 4. Illustrations of the studied data augmentation operators. Each augmentation can transform data stochastically with some internal parameters (e.g. rotation degree, noise level). Note that we *only* test these operators in ablation, the *augmentation policy used to train our models* only includes *random crop (with flip and resize)*, *color distortion*, and *Gaussian blur*. (Original image cc-by: Von.grzanka)

Data Augmentation: A way to perturb the original data

Create new, meaningful samples that you may find in the original data distribution

Red box = actually used for training

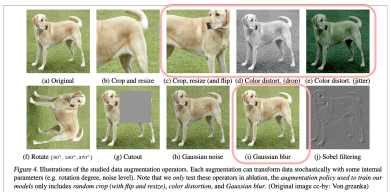


Figure 4. Illustrations of the studied data augmentation operators. Each augmentation can transform data stochastically with some internal parameters (e.g. rotation degree, noise level). Note that we *only* test these operators in ablation, the *augmentation policy used to train our models* only includes *random crop (with flip and resize)*, *color distortion*, and *Gaussian blur*. (Original image cc-by: Von.grzanka)

Comparison of loss functions

5. Loss Functions and Batch Size

5.1. Normalized cross entropy loss with adjustable temperature works better than alternatives

We compare the NT-Xent loss against other commonly used contrastive loss functions, such as logistic loss (Mikolov et al., 2013), and margin loss (Schroff et al., 2015). Table 2 shows the objective function as well as the gradient to the input of the loss function. Looking at the gradient, we observe 1) ℓ_2 normalization (i.e. cosine similarity) along with temperature effectively weights different examples, and an appropriate temperature can help the model learn from hard negatives; and 2) unlike cross-entropy, other objective functions do not weigh the negatives by their relative hardness. As a result, one must apply semi-hard negative mining (Schroff et al., 2015) for these loss functions: instead of computing the gradient over all loss terms, one can compute the gradient using semi-hard negative terms (*i.e.*, those that are within the loss margin and closest in distance, but farther than positive examples).

Comparison of loss functions

5. Loss Functions and Batch Size

5.1. Normalized cross entropy loss with adjustable temperature works better than alternatives

We compare the NT-Xent loss against other commonly used contrastive loss functions, such as logistic loss (Mikolov et al., 2013), and margin loss (Schroff et al., 2015). Table 2 shows the objective function as well as the gradient to the input of the loss function. Looking at the gradient, we observe 1) ℓ_2 normalization (i.e. cosine similarity) along with temperature effectively weights different examples, and an appropriate temperature can help the model learn from hard negatives; and 2) unlike cross-entropy, other objective functions do not weigh the negatives by their relative hardness. As a result, one must apply semi-hard negative mining (Schroff et al., 2015) for these loss functions: instead of computing the gradient over all loss terms, one can compute the gradient using semi-hard negative terms (*i.e.*, those that are within the loss margin and closest in distance, but farther than positive examples).

Comparison of loss functions

Name	Negative loss function	Gradient w.r.t. $\boldsymbol{u}$
NT-Xent	$\boldsymbol{u}^T\boldsymbol{v}^+/\tau - \log \sum_{\boldsymbol{v} \in \{\boldsymbol{v}^+, \boldsymbol{v}^-\}} \exp(\boldsymbol{u}^T\boldsymbol{v}/\tau)$	$(1 - \frac{\exp(\boldsymbol{u}^T\boldsymbol{v}^+/\tau)}{Z(\boldsymbol{u})})/\tau \boldsymbol{v}^+ - \sum_{\boldsymbol{v}^-} \frac{\exp(\boldsymbol{u}^T\boldsymbol{v}^-/\tau)}{Z(\boldsymbol{u})}/\tau \boldsymbol{v}^-$
NT-Logistic	$\log \sigma(\boldsymbol{u}^T\boldsymbol{v}^+/\tau) + \log \sigma(-\boldsymbol{u}^T\boldsymbol{v}^-/\tau)$	$(\sigma(-\boldsymbol{u}^T\boldsymbol{v}^+/\tau))/\tau \boldsymbol{v}^+ - \sigma(\boldsymbol{u}^T\boldsymbol{v}^-/\tau)/\tau \boldsymbol{v}^-$
Margin Triplet	$-\max(\boldsymbol{u}^T\boldsymbol{v}^- - \boldsymbol{u}^T\boldsymbol{v}^+ + m, 0)$	$\boldsymbol{v}^+ - \boldsymbol{v}^-$ if $\boldsymbol{u}^T\boldsymbol{v}^+ - \boldsymbol{u}^T\boldsymbol{v}^- < m$ else $\boldsymbol{0}$

Table 2. Negative loss functions and their gradients. All input vectors, i.e. $\boldsymbol{u}, \boldsymbol{v}^+, \boldsymbol{v}^-$, are ℓ_2 normalized. NT-Xent is an abbreviation for "Normalized Temperature-scaled Cross Entropy". Different loss functions impose different weightings of positive and negative examples.

SimCLR: main algorithm

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 $g(\cdot)$

Results: Linear classifier on top of SimCLR representation

Method	Architecture	Param (M)	Top 1	Top 5
Methods using ResNet-50:				
Local Agg.	ResNet-50	24	60.2	-
MoCo	ResNet-50	24	60.6	-
PIRL	ResNet-50	24	63.6	-
CPC v2	ResNet-50	24	63.8	85.3
SimCLR (ours)	ResNet-50	24	**69.3**	**89.0**
Methods using other architectures:				
Rotation	RevNet-50 (4×)	86	55.4	-
BigBiGAN	RevNet-50 (4×)	86	61.3	81.9
AMDIM	Custom-ResNet	626	68.1	-
CMC	ResNet-50 (2×)	188	68.4	88.2
MoCo	ResNet-50 (4×)	375	68.6	-
CPC v2	ResNet-161 (∗)	305	71.5	90.1
SimCLR (ours)	ResNet-50 (2×)	94	74.2	92.0
SimCLR (ours)	ResNet-50 (4×)	375	**76.5**	**93.2**

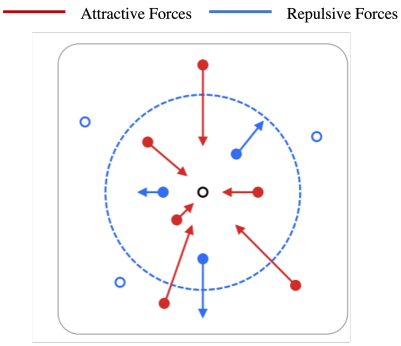
Table 6. ImageNet accuracies of linear classifiers trained on representations learned with different self-supervised methods.

×4 is a "depth width" multiplier of 4

Results: Models trained with few labels

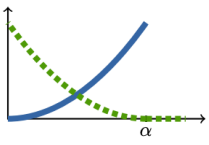
Method	Architecture	Label fraction 1%	10%
		Top 5	
Supervised baseline	ResNet-50	48.4	80.4
Methods using other label-propagation:			
Pseudo-label	ResNet-50	51.6	82.4
VAT+Entropy Min.	ResNet-50	47.0	83.4
UDA (w. RandAug)	ResNet-50	-	88.5
FixMatch (w. RandAug)	ResNet-50	-	89.1
S4L (Rot+VAT+En. M.)	ResNet-50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



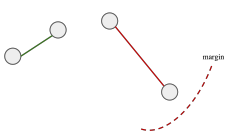
Contrastive loss graph (loss in function of distance)

The solid blue lines show the loss function for positive pairs, the dotted green for negative pairs.



(a) Contrastive loss [11]

Contrastive loss graph (in the embedding space)



Contrastive loss limitations

- One drawback of the contrastive loss is that **we have to select a constant margin.** <u>**for all pairs of negative samples.**</u> This implies that visually diverse classes are embedded in the same small space as visually similar ones. The embedding space does not allow for distortions.
- **Scale Quadratically** with the sampling of points.
  - this is why `SimCLR` works better with large batch size → you have more negative training samples to constrain the search space
  - `SimCLR` takes negative samples from the large mini-batch (**negative samples are the denominator in SimCLR loss**).

Triplet Loss

Idea: Take relative comparisons between pairs, not just consider the single pairs.

- So we have to have **two pairs (a positive pair, and a negative pair).**
- Keep **all positives closer to any negatives**

This formulation allows the embedding space to be arbitrarily distorted and does not impose a constant margin α.

$$\ell^{\text{triplet}}(a, p, n) := \left[D_{ap}^2 - D_{an}^2 + \alpha\right]_+ .$$

If not wrong, it was "invented" at USC

Results: Results across datasets

	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
Linear evaluation:												
SimCLR (ours)	**76.9**	**95.3**	80.2	48.4	**65.9**	60.0	61.2	**84.2**	**78.9**	89.2	**93.9**	**95.0**
Supervised	75.2	**95.7**	**81.2**	**56.4**	64.9	**68.8**	**63.8**	83.8	**78.7**	**92.3**	**94.1**	94.2
Fine-tuned:												
SimCLR (ours)	**89.4**	**98.6**	**89.0**	**78.2**	**68.1**	**92.1**	**87.0**	**86.6**	**77.8**	92.1	**94.1**	97.6
Supervised	88.7	98.3	**88.7**	**77.8**	67.0	91.4	**88.0**	86.5	**78.8**	**93.2**	**94.2**	**98.0**
Random init	88.3	96.0	81.9	**77.0**	53.7	91.3	84.8	69.4	64.1	82.7	72.5	92.5

Table 8. Comparison of transfer learning performance of our self-supervised approach with supervised baselines across 12 natural image classification datasets, for ResNet-50 (4×) models pretrained on ImageNet. Results not significantly worse than the best ($p > 0.05$, permutation test) are shown in bold. See Appendix B.8 for experimental details and results with standard ResNet-50.

Other forms of Contrastive Methods

1. –Contrastive loss implemented with logistic function and negative sampling [word2vec]– (nlp)
2. –Contrastive loss implemented with Softmax and large mini-batches (vision)–
3. Contrastive loss with geometric interpretation and margin (vision)

Contrastive loss with geometric interpretation and margins (vision)

Sampling Matters in Deep Embedding Learning

Chao-Yuan Wu* UT Austin cywu@cs.utexas.edu
R. Manmatha A9/Amazon manmatha@a9.com
Alexander J. Smola Amazon smola@amazon.com
Philipp Krähenbühl UT Austin philkr@cs.utexas.edu

Neural Net as an embedding function

Let $f_\theta(\mathbf{x}_i)$ be an embedding of a high-dimensional data point (in case of images) $\mathbf{x}_i \in \mathbb{R}^N$ where $f : \mathbb{R}^N \mapsto \mathbb{R}^D$ is a differentiable deep network with parameters θ.

Geometric idea

Our goal is to learn an embedding that **keeps similar data points close, while pushing dissimilar datapoints apart.** Note distance in measured in the embedding space.

$$D_{ij} = \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|_2$$

For any **positive pair** of datapoints:

- $y_{ij} = 1$ this distance should be small
- $y_{ij} = 0$ it should be large if negative pair

Contrastive Loss

The contrastive loss directly optimizes this distance by:

- encouraging **all positive distances to approach 0**
- while **keeping negative distances above a certain threshold** α

$$\ell^{\text{contrast}}(i, j) := y_{ij} D_{ij}^2 + (1 - y_{ij}) \left[\alpha - D_{ij}\right]_+^2 .$$

where $[\cdot]_+ = \max(0, \cdot)$

Right part of the math reads as if you are already far away of at least α then **do not pay a penalty.**

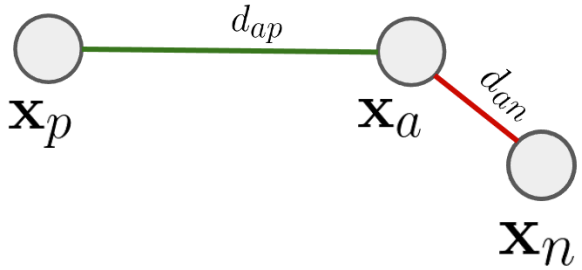
Two Pairs

Although we have two pairs, we have "**3 actors**" in total (**3 embeddings**):

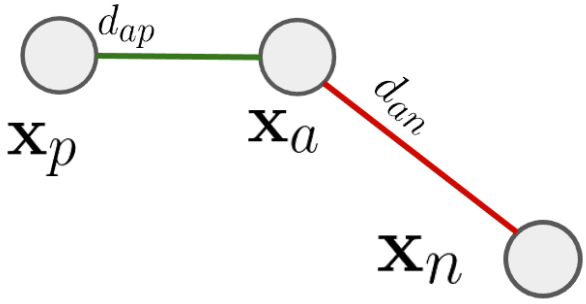
$\mathbf{x}_p$ positive $\mathbf{x}_a$ anchor $\mathbf{x}_n$ negative

- A $\mathbf{x}_a$ an embedding paired with another positive embedding $\mathbf{x}_p$
  - Call p the positive and a (which is another positive), **the anchor.**
  - The assumption is that the latent factor of $\mathbf{x}_p$ is equal latent factor of $\mathbf{x}_a$.
- Then select a negative as the pair $\mathbf{x}_n$ wrt to another negative point $\mathbf{x}_n$.

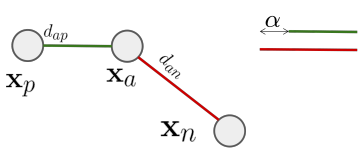
Triplet Loss



Triplet Loss



Triplet loss (relative margin)



Triplet loss Limitations

- Now you have to sample **Triplet! Cubic sampling complexity**
- Moreover, once the network converges, most samples contribute in a minor way as very few of the negative margins are violated

Hacks:

For the contrastive loss, hard negative mining usually offers faster convergence. For the triplet loss, it is less obvious, as hard negative mining often leads to collapsed models, i.e. all images have the same embedding

Distance distribution in high-dimension

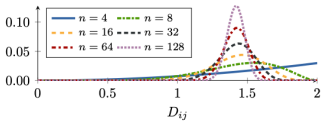


Figure 2: Density of datapoints on the D-dimensional unit sphere. Note the concentration of measure as the dimensionality increases — most points are almost equidistant.

4. Distance Weighted Margin-Based Loss

To understand what happens when sampling negative uniformly, recall that our embeddings are typically constrained to the n-dimensional unit sphere $\mathbb{S}^{n-1}$ for large $n \geq 128$. Consider the situation where the points are uniformly distributed on the sphere. In this case, the distribution of pairwise distances follows

$$q(d) \propto d^{n-2}\left[1 - \tfrac{1}{4}d^2\right]^{\frac{n-3}{2}}.$$

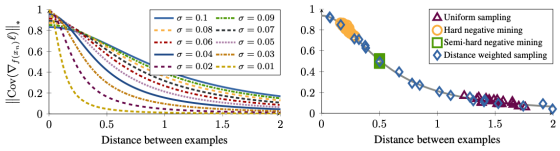
See [1] for a derivation. Figure 2 shows concentration of measure occurring. In fact, in high dimensional space, $q(d)$ approaches $\mathcal{N}(\sqrt{2}, \frac{1}{2n})$. In other words, if negative examples are scattered uniformly, and we sample them randomly, we are likely to obtain examples that are $\sqrt{2}$-away. For thresholds less than $\sqrt{2}$, this induces no loss, and thus no progress for learning. Learned embeddings follow a very similar distribution, and thus the same reasoning applies. See supplementary material for details.

We do inverse transform sampling on q^{-1}

Instead of sampling uniformly and be likely to get $\sqrt{2}$-away points, we sample uniformly but according to the distance.

Distance weighted sampling. We thus propose a new sampling distribution that corrects the bias while controlling the variance. Specifically, we sample uniformly according to distance, i.e. sampling with weights $q(d)^{-1}$. This gives us examples which are spread out instead of being clustered around a small region. To avoid noisy samples, we clip the weighted sampling. Formally, given an anchor example a, distance weighted sampling samples negative pair (a, n^*) with

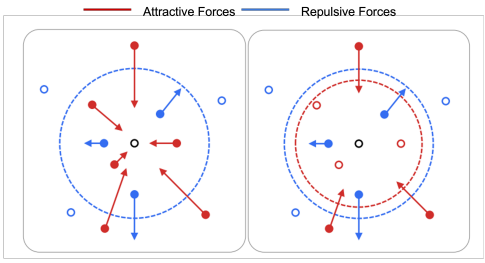
$$\Pr(n^* = n|a) \propto \min\left(\Lambda, q^{-1}(D_{an})\right).$$



(a) Variance of gradient at different noise levels. (b) Sample distribution for different strategies.

Figure 3: (a) shows the nuclear norm of a noisy gradient estimate for various levels of noise. High variance means the gradient is close to random, while low variance implies a deterministic gradient estimate. Lower is better. Note that higher noise levels have a lower variance at distance 0. This is due to the spherical projection imposed by the normalization. (b) shows the empirical distribution of samples drawn for different strategies. Distance weighted sampling selects a wide range of samples, while all other approaches are biased towards certain distances.

Double-margin Contrastive Loss



Margin-based Contrastive loss

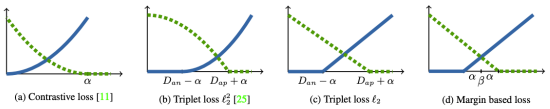
Margin based loss. These observations motivate our design of a loss function which enjoys the flexibility of the triplet loss, has a shape suitable for examples from all distances, while offering the computational efficiency of a contrastive loss. The basic idea can be traced back to the insight that in ordinal regression only the relative order of scores matters [17]. That is, we only need to know the crossover between both sets. Isotonic regression exploits this by estimating such a threshold separately and then penalizes scores relative to the threshold. We use the same trick, now applied to pairwise distances rather than score functions. The adaptive margin based loss is defined as

$$\ell^{\text{margin}}(i,j) := (\alpha + y_{ij}(D_{ij} - \beta))_+ .$$

Here β is a variable that determines the boundary between positive and negative pairs, α controls the margin of separation, and $y_{ij} \in \{-1, 1\}$. Figure 4d visualizes this new loss

Note: They are learning β!

Loss at a glance



(a) Contrastive loss [11] (b) Triplet loss ℓ_2^2 [25] (c) Triplet loss ℓ_2 (d) Margin based loss

Figure 4: Loss vs. pairwise distance. The solid blue lines show the loss function for positive pairs, the dotted green for negative pairs. Our loss finds an optimal boundary β between positive and negative pairs, and α ensures that they are separated by a large margin.

Break

Toward CLIP (Contrastive Language-Image Pre-Training)

Learning Visual Features from Large Weakly Supervised Data

Armand Joulin* ajoulin@fb.com
Laurens van der Maaten* lvdmaaten@fb.com
Allan Jabri ajabri@fb.com
Nicolas Vasilache ntv@fb.com
Facebook AI Research
770 Broadway, New York NY 10003

Note: this paper is before Transformers. Circa 2015.

Idea: Instead of Self-Supervised..... "Text" Supervised

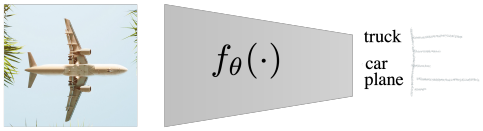
> In this paper, we explore the potential of leveraging massive, weakly labeled image collections for learning good visual features.

- We train convolutional networks on a dataset of 100 million Flickr photos and captions, and show that these networks produce features that **perform well in a range of vision problems**
- We also show that the networks appropriately capture **word similarity**, and learn correspondences between different languages

Image Classification

Note the discriminative model: $p(y|x)$

Given x, compute a distribution over the labels $p(y|x)$.



Problem: Who is going to label this image as "plane"?

Use Supervision from the Web [Weak Supervision]

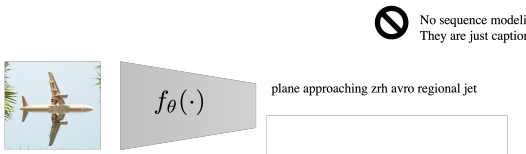
Research question:

> Can we learn high quality visual features from scratch without using any fully supervised data?

Figure 1: Six randomly picked photos from the Flickr 100M dataset and the corresponding descriptions.

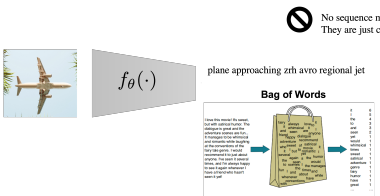
Flickr 100M dataset contains ~100M photos with associated "captions"

Weakly Supervised Image classification



Weakly Supervised Image classification

- We treat each individual word in a photo's caption as **a target for that photo**
- That is: a **multi-label learning problem with extremely noise labels**
- We map an image **x** to multiple labels, not a single **y**.



Text preprocessing

Preprocessing. We preprocessed the text by removing all numbers and punctuation (*e.g.*, the # character for hashtags), removing all accents and special characters, and lower-casing. We then used the Penn Treebank tokenizer[3] to tokenize the titles and captions into words, and used all hashtags and words as targets for the photos. We remove the 500 most common words (*e.g.*, "the", "of", and "and") and because the tail of the word distribution is very long [1], we restrict ourselves to predicting only the $K = \{1,000; 10,000; 100,000\}$ most common words. For these dictionary sizes, the average number of targets per photo is 3.72, 5.62, and 6.81, respectively. The target for each image is a bag of all the words in the dictionary associated with that image, *i.e.*, a multi-label vector $\mathbf{y} \in \{0,1\}^K$. The

plane 1
car 0
...
truck 0

K

plane 1 0 0
car 0 0 0
...
truck 0 0 0
approaching 0 1 0
regional 0 0 1

"plane approaching zrh avro regional jet"

Multi-label Classification problem

Note: each word token now is **NOT mutually exclusive** as in standard classification. i.e. if you sum the label vector you do not get unit mass (it is not a probability anymore). Yet you get the number of word token in the caption.

K

plane 1
car 0
...
truck 0
approaching 1
regional 1

"plane approaching zrh avro regional jet"

Loss Function

Note N is the number of samples and K is how many words you have in the bag.

Loss functions. We denote the training set by $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1,\ldots,N}$ with the D-dimensional observation $\mathbf{x} \in \mathbb{R}^D$ and the multi-label vector $\mathbf{y} \in \{0,1\}^K$. We parametrize the mapping $f(\mathbf{x};\theta)$ from observation $\mathbf{x} \in \mathbb{R}^D$ to some intermediate embedding $\mathbf{e} \in \mathbb{R}^E$ by a convolutional network with parameters θ; and the mapping from that embedding $\mathbf{e}$ to a label $\mathbf{y} \in \{0,1\}^K$ by $\text{sign}(\mathbf{W}^\top \mathbf{e})$, where $\mathbf{W}$ is an $E \times K$ matrix. The parameters θ and $\mathbf{W}$ are optimized jointly to minimize a one-versus-all or multi-class logistic loss. The one-versus-all logistic loss sums binary classifier losses over all classes:

$$\sum_{n=1}^{N}\sum_{k=1}^{K} \frac{y_{nk}}{N_k} \log \sigma(f(\mathbf{x}_n;\theta)) + \frac{1-y_{nk}}{N-N_k} \log(1-\sigma(f(\mathbf{x}_n,\theta))),$$

where $\sigma(x) = 1/(1+\exp(-x))$ is the sigmoid function and N_k is the number of positive examples for the class k. The multi-class logistic loss minimizes the negative sum

Loss Function

Note N is the number of samples and K is how many words you have in the bag.

k. The multi-class logistic loss minimizes the negative sum of the log-probabilities over all positive labels. Herein, the probabilities are computed using a softmax layer:

$$\ell(\theta, \mathbf{W}; \mathcal{D}) = \frac{-1}{N} \sum_{n=1}^{N}\sum_{k=1}^{K} y_{nk} \log \left[\frac{\exp(\mathbf{w}_k^\top f(\mathbf{x}_n;\theta))}{\sum_{k'=1}^{K} \exp(\mathbf{w}_{k'}^\top f(\mathbf{x}_n;\theta))} \right].$$

In preliminary experiments, we also considered a pairwise ranking loss [50, 53]. This loss only updates two columns of $\mathbf{W}$ per training example (corresponding to a positive and a negative label). We found that when training convolutional networks end-to-end, these sparse updates significantly slow down training, which is why we did not consider ranking loss further in this study.

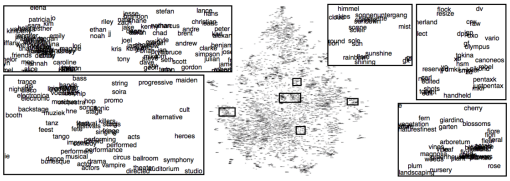


Figure 6. t-SNE map of 10,000 words based on their embeddings as learned by a weakly supervised convolutional network trained on the Flickr dataset. Note that all the semantic information represented in the word embeddings is the result of observing that these words are assigned to images with similar visual content (the model did not observe word co-occurrences during training). A full-resolution version of the map is provided in the supplemental material.

Contrastive Learning of Medical Visual Representations from Paired Images and Text

Yuhao Zhang* YUHAO@CS.STANFORD.EDU
Biomedical Informatics Training Program, Stanford University

Hang Jiang* HJIAN42@STANFORD.EDU
Symbolic Systems Program, Stanford University

Yasuhide Miura† YSMIURA@STANFORD.EDU
Computer Science Department, Stanford University

Christopher D. Manning MANNING@STANFORD.EDU
Computer Science and Linguistics Departments, Stanford University

Curtis P. Langlotz LANGLOTZ@STANFORD.EDU
Department of Radiology, Stanford University

Training pairs

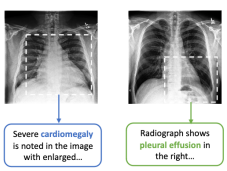


Figure 1: Two example chest X-ray images with different abnormality categories, along with sentences from their paired textual report and example views indicative of their characteristics.

ConVIRT: Contrastive Visual Representation Learning from Text

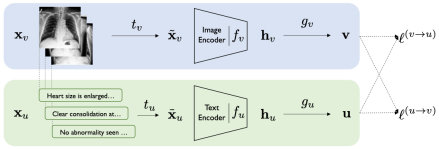


Figure 2: Overview of our ConVIRT framework. The blue and green shades represent the image and text encoding pipelines, respectively. Our method relies on maximizing the agreement between the true image-text representation pairs with bidirectional losses $\ell^{(v\to u)}$ and $\ell^{(u\to v)}$.

ConVIRT: Contrastive Visual Representation Learning from Text

At training time, we sample a minibatch of N input pairs $(\mathbf{x}_v, \mathbf{x}_u)$ from training data, and calculate their representation pairs $(\mathbf{v}, \mathbf{u})$. We use $(\mathbf{v}_i, \mathbf{u}_i)$ to denote the i-th pair. The training objective of ConVIRT involves two loss functions. The first loss function is an image-to-text contrastive loss for the i-th pair:

$$\ell_i^{(v\to u)} = -\log \frac{\exp(\langle \mathbf{v}_i, \mathbf{u}_i \rangle / \tau)}{\sum_{k=1}^{N} \exp(\langle \mathbf{v}_i, \mathbf{u}_k \rangle / \tau)}, \tag{2}$$

where $\langle \mathbf{v}_i, \mathbf{u}_i \rangle$ represents the cosine similarity, i.e., $\langle \mathbf{v}, \mathbf{u} \rangle = \mathbf{v}^\top \mathbf{u} / \|\mathbf{v}\| \|\mathbf{u}\|$; and $\tau \in \mathbb{R}^+$ represents a temperature parameter. This loss takes the same form as the InfoNCE loss (Oord et al., 2018), and minimizing it leads to encoders that maximally preserve the mutual information between the true pairs under the representation functions. Intuitively, it is the log loss of an N-way classifier that tries to predict $\langle \mathbf{v}_i, \mathbf{u}_i \rangle$ as the true pair. Note that unlike previous work which use a contrastive loss between inputs of the same modality (Chen et al., 2020a; He et al., 2020), our image-to-text contrastive loss is asymmetric for each input modality. We therefore define a similar text-to-image contrastive loss as:

$$\ell_i^{(u\to v)} = -\log \frac{\exp(\langle \mathbf{u}_i, \mathbf{v}_i \rangle / \tau)}{\sum_{k=1}^{N} \exp(\langle \mathbf{u}_i, \mathbf{v}_k \rangle / \tau)}. \tag{3}$$

ConVIRT: Contrastive Visual Representation Learning from Text

Our final training loss is then computed as a weighted combination of the two losses averaged over all positive image-text pairs in each minibatch:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left(\lambda \ell_i^{(v\to u)} + (1-\lambda) \ell_i^{(u\to v)} \right), \tag{4}$$

where $\lambda \in [0,1]$ is a scalar weight.

ConVIRT: Realization

Visual part: ResNet-50 Encoder

Sequential applications of five random transformations: cropping, horizontal flipping, affine transformation, color jittering and Gaussian blur.

NLP part: BERT encoder

BERT Encoder followed by a max-pooling layer over all output vectors. We initialize our encoder with the **ClinicalBERT weights** (Alsentzer et al., 2019) pretrained on the MIMIC clinical notes, which achieved state-of-the-art performance on a suite of clinical NLP tasks.

We apply a simple uniform sampling of a sentence from the input document $\mathbf{u}$ (i.e., $\mathbf{u}$ is a randomly sampled sentence from $\mathbf{u}$ for each minibatch).

The CLIP paper

Learning Transferable Visual Models From Natural Language Supervision

Alec Radford [*1] Jong Wook Kim [*1] Chris Hallacy [1] Aditya Ramesh [1] Gabriel Goh [1] Sandhini Agarwal [1]
Girish Sastry [1] Amanda Askell [1] Pamela Mishkin [1] Jack Clark [1] Gretchen Krueger [1] Ilya Sutskever [1]

Claim

> We demonstrate that a simplified version of ConVIRT trained from scratch, which we call CLIP, for Contrastive Language- Image Pre-training, is an efficient and scalable method of learning from natural language supervision

Results

> CLIP learns to perform a wide set of tasks during pre- training including OCR, geo-localization, action recognition, and outperforms the best publicly available ImageNet model while being more computationally efficient. We also find that zero-shot CLIP models are much more robust than equivalent accuracy supervised ImageNet models.

Approach

Learning perception from the **supervision contained in natural language paired with images.**

Huge dataset

large quantities of data of this form available publicly on the internet. To test this we constructed a new dataset of 400 million (image, text) pairs collected form a variety of publicly available sources on the Internet. To attempt to cover as broad a set of visual concepts as possible, we search for (image, text) pairs as part of the construction process whose text includes one of a set of 500,000 queries. We approximately class balance the results by including up to 20,000 (image, text) pairs per query. The resulting dataset has a similar total word count as the WebText dataset used to train GPT-2. We refer to this dataset as WIT for WebImageText. [1]

Caption Prediction does not work

> Our initial approach, similar to VirTex, jointly trained an image CNN and text transformer from scratch to predict the caption of an image

Note: BoW is the 2015 paper we reviewed so far.

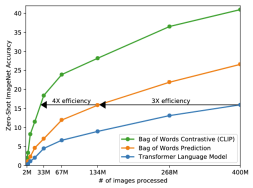


Figure 2. **CLIP is much more efficient at zero-shot transfer than our image caption baseline.** Although highly expressive, we found that transformer-based language models are relatively weak at zero-shot ImageNet classification. Here, we see that it learns 3x slower than a baseline which predicts a bag-of-words (BoW) encoding of the text (Joulin et al., 2016). Swapping the prediction objective for the contrastive objective of CLIP further improves efficiency another 4x.

CLIP Classification: Text encoder generates classification "weights"

Text encoder acts as an **on-the-fly generator of classification weights**

$$p(y|x) \propto \mathbf{E}_{y_i} \mathbf{e}_x$$

The image embedding is given $\mathbf{e}_x$. $\forall$ class y_i we encode the class in the text y_i and "fill" a column matrix in $\mathbf{E}_y$. Then we classify as:

$$\arg\max_i \mathbf{E}_{y_i} \mathbf{e}_x$$

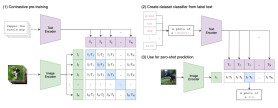


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

CLIP Training Code

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

Figure 3. Numpy-like pseudocode for the core of an implementation of CLIP.

CLIP Results

Zero Short Performance

CLIP with transformers to "encode" the classifier vs logistic regression trained on ResNet-features.

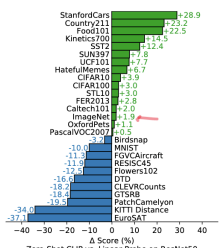


Figure 4. **Zero-shot CLIP is competitive with a fully supervised baseline.** Across a 27 dataset eval suite, a zero-shot CLIP classifier outperforms a fully supervised linear classifier fitted on ResNet50 features on 16 datasets, including ImageNet.

Details

Since over-fitting is not a major concern, the details of training CLIP are simplified compared to Zhang et al. (2020). We train CLIP from scratch instead of initializing with pre-trained weights. We remove the non-linear projection between the representation and the contrastive embedding space. We use only a linear projection to map from each encoder's representation to the multi-modal embedding space.

We also remove the text transformation function t_u which samples a single sentence at uniform from the text since many of the (image, text) pairs in CLIP's pre-training dataset are only a single sentence. We also simplify the image transformation function t_v. A random square crop from resized images is the only data augmentation used during training. Finally, the temperature parameter which controls the range of the logits in the softmax, τ, is directly optimized during training as a log-parameterized multiplicative scalar to avoid turning as a hyper-parameter.

Relax JM

CLIP Training and Inference

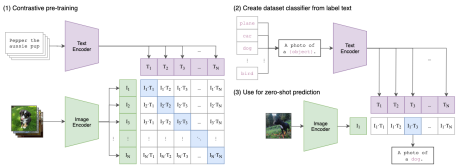


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

Zero Short Performance

First, CLIP's zero-shot classifier is **generated via natural language which allows for visual concepts to be directly specified ("communicated").** By contrast, "normal" supervised learn- ing must infer concepts **indirectly from training examples**. Context-less example-based learning has the drawback that **many different hypotheses can be consistent with the data**, especially in the one-shot case. A single image often con- tains many different visual concepts

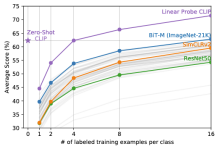


Figure 5. **Zero-shot CLIP outperforms few-shot linear probes.** Zero-shot CLIP matches the average performance of a 4-shot linear classifier trained on the same feature space and nearly matches the best results of a 16-shot linear classifier across publicly available models. For both BiT-M and SimCLRv2, the best performing model is highlighted. Light gray lines are other models in the eval suite. The 20 datasets with at least 16 examples per class were used in this analysis.

Representation Learning: Distributional Shifts

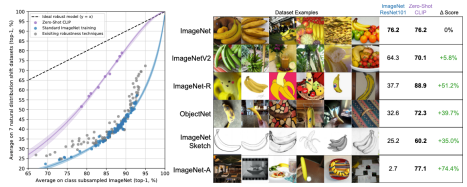
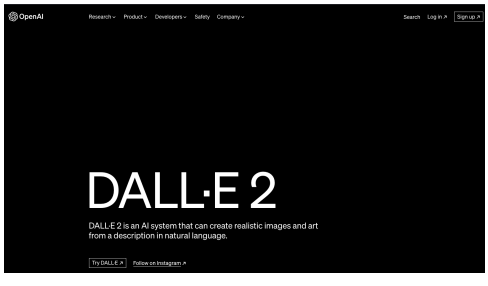


Figure 7. **Zero-shot CLIP is much more robust to distribution shift than standard ImageNet models.** (Left) An ideal robust model (dashed line) performs equally well on the ImageNet distribution and on other natural image distributions. Zero-shot CLIP models shrink this "robustness gap" by up to 75%. Linear fits on logit transformed values are shown with bootstrap estimated 95% confidence intervals. (Right) Visualizing distribution shift for bananas, a class shared across 5 of the 7 natural distribution shift datasets. The performance of the best zero-shot CLIP model is compared with a model that has the same performance on the ImageNet validation set, ResNet101.

unCLIP (DALL·E 2)



Hierarchical Text-Conditional Image Generation with CLIP Latents

Aditya Ramesh* OpenAI aramesh@openai.com

Prafulla Dhariwal* OpenAI prafulla@openai.com

Alex Nichol* OpenAI alex@openai.com

Casey Chu* OpenAI casey@openai.com

Mark Chen OpenAI mark@openai.com

Method

Our training dataset consists of pairs (x, y) of images x and their corresponding captions y. Given an image x, let z_i and z_t be its CLIP image and text embeddings, respectively. We design our generative stack to produce images from captions using two components:

- A *prior* $P(z_i|y)$ that produces CLIP image embeddings z_i conditioned on captions y.
- A *decoder* $P(x|z_i, y)$ that produces images x conditioned on CLIP image embeddings z_i (and optionally text captions y).

Method

The decoder allows us to invert images given their CLIP image embeddings, while the prior allows us to learn a generative model of the image embeddings themselves. Stacking these two components yields a generative model $P(x|y)$ of images x given captions y:

$$P(x|y) = P(x, z_i|y) = P(x|z_i, y)P(z_i|y).$$

The first equality holds because z_i is a deterministic function of x. The second equality holds because of the chain rule. Thus, we can sample from the true conditional distribution $P(x|y)$ by first sampling z_i using the

prior, and then sampling x using the decoder. In the following sections, we describe our decoder and prior stacks. For training details and hyperparameters, refer to Appendix C.

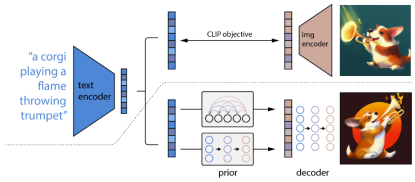


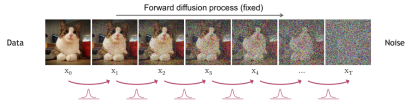
Figure 2: A high-level overview of unCLIP. Above the dotted line, we depict the CLIP training process, through which we learn a joint representation space for text and images. Below the dotted line, we depict our text-to-image generation process: a CLIP text embedding is first fed to an autoregressive or diffusion prior to produce an image embedding, and then this embedding is used to condition a diffusion decoder which produces a final image. Note that the CLIP model is frozen during training of the prior and decoder.

Joint Distribution is the marginal times the conditional

$$q(\mathbf{x}_0, \dots, \mathbf{x}_T) = q(\mathbf{x}_0) \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

Forward Diffusion Process - Image Domain

$$q(\mathbf{x}_0, \dots, \mathbf{x}_T) = q(\mathbf{x}_0) \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$



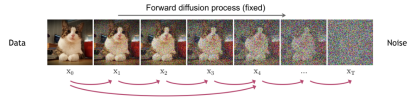
Forward Diffusion Process - Diffusion Kernel

We can "bypass" time steps and write:

$$\alpha_t = \prod_{s=1}^{t} (1 - \beta_t).$$

Note $\beta_t \approx 1\mathbf{e}-2$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_t}\,\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I}) \quad \text{diffusion kernel}$$

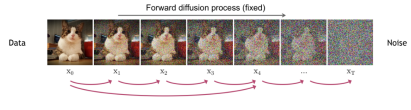


Forward Diffusion Process - Sampling

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_t}\,\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I}) \quad \text{diffusion kernel}$$

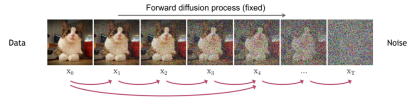
Sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ and then

$$\mathbf{x}_t = \sqrt{\alpha_t}\,\mathbf{x}_{t-1} + (1-\alpha_t)\epsilon$$



Forward Diffusion Process - $q(\mathbf{x}_t|\mathbf{x}_0) \approx \mathcal{N}(0, \mathbf{I})$

The β_t value is designed such that $\alpha_T \mapsto 0$ and then $q(\mathbf{x}_t|\mathbf{x}_0) \approx \mathcal{N}(0, \mathbf{I})$



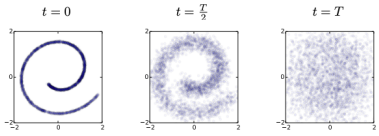
Diffusion Models

Diffusion

Based on https://cvpr2022-tutorial-diffusion-models.github.io/

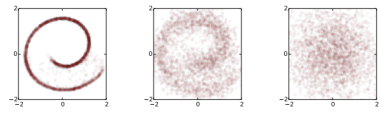
Map structured data to white noise

$q\left(\mathbf{x}^{(0\cdots T)}\right)$



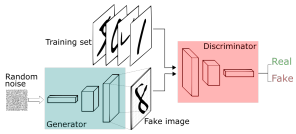
From white noise learn θ to go back to structured data

$p\left(\mathbf{x}^{(0\cdots T)}\right)$



Learning to transform noise to "meaningful data" is generative modeling

Even GANs map noise to data



Data distribution (unknown)

$q(\mathbf{x}_0)$ but we have samples from the training set $\mathbf{x}_0 \sim q(\mathbf{x})$

$q(\mathbf{x}_0) \longrightarrow \mathcal{N}(0, \mathbf{I})$

Data perturbation process parametrized by β

Given what we have $\mathbf{x}_0$, we define a distribution over the perturbed output at the next step:

$$\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{1-\beta_t}\,\mathbf{x}_0, \beta_t \mathbf{I})$$

Read it as generate data centered on $\mathbf{x}_0$ scaled by $\sqrt{1-\beta_t}$ with variance β_t.

Go recursive

$$\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

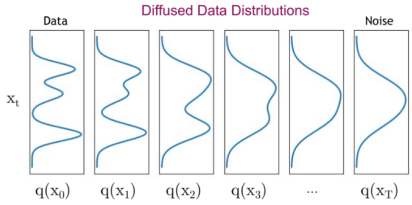
This means we applied the chain with T time steps:

$$\mathbf{x}_0 \to \beta_1 \dots \to \beta_T \to q(\mathbf{x}_0, \dots, \mathbf{x}_T)$$

What happens to data distribution in the forward process

$$\underbrace{q(\mathbf{x}_t)}_{\substack{\text{Diffused}\\ \text{data dist.}}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\substack{\text{Joint}\\ \text{dist.}}} d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\substack{\text{Input}\\ \text{data dist.}}} \underbrace{q(\mathbf{x}_t \mid \mathbf{x}_0)}_{\substack{\text{Diffusion}\\ \text{kernel}}} d\mathbf{x}_0$$

Gaussian Smoothing of the unknown data distribution



Forward pass is ancestor sampling

Similar to sampling in GMM (i.e. sample which gaussian and then sample from the gaussian). Here we aim at sampling from $\mathbf{x}_t \sim q(\mathbf{x}_t)$:

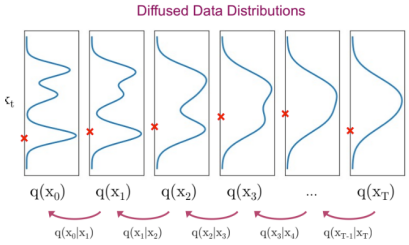
1. We sample $\mathbf{x}_0 \sim q(\mathbf{x})$ (take a data point from training set)
2. Given $\mathbf{x}_0$, we sample $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$

Generative Learning by Denoising

Reversing the forward diffusion process

The generative distribution will be trained to describe the same trajectory, but in reverse:

$$p(\mathbf{x}_0 \cdots, \mathbf{x}_T) = p(\mathbf{x}_T) \prod_{t=1}^{T} p(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$

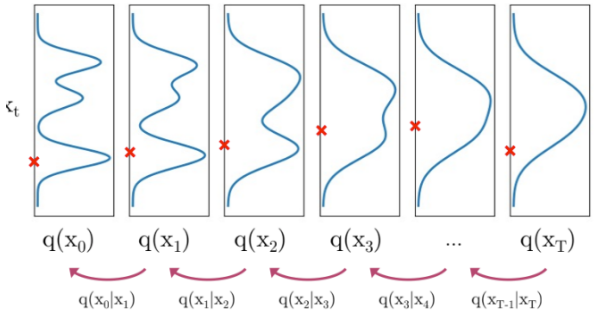


Reverse Generation

- Sample $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$
- Iteratively sample $\mathbf{x}_{t-1} \sim q(\mathbf{x}_{t-1}|\mathbf{x}_t)$

$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ true denoising distribution is intractable unless β_t is very small.

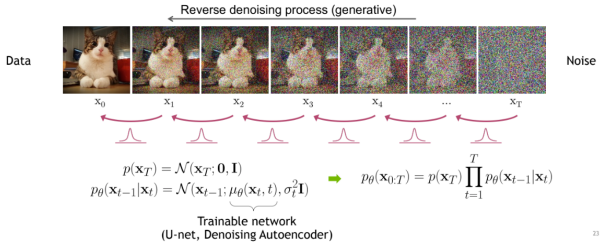
Diffused Data Distributions



We train a model to approximate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$

If we assume $\beta_t \approx 0$ (very small), then the reverse process is also **Gaussian** where the mean is parametrized by the weights of a neural networks.

Reverse Denoising Process



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}, \sigma_t^2\mathbf{I})$$

Trainable network
(U-net, Denoising Autoencoder)

$$\Rightarrow \quad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$