Unsupervised Learning

Luisa Fernanda Borbón y María Fernanda Roa
Guillaume Jeanneret y Felipe Torres

AML course

Content

- Introduction
- History
- Traditional Methods
- Autoencoders
- Other Deep UL Methods
- Papers
- Tutorial & Homework

Introduction

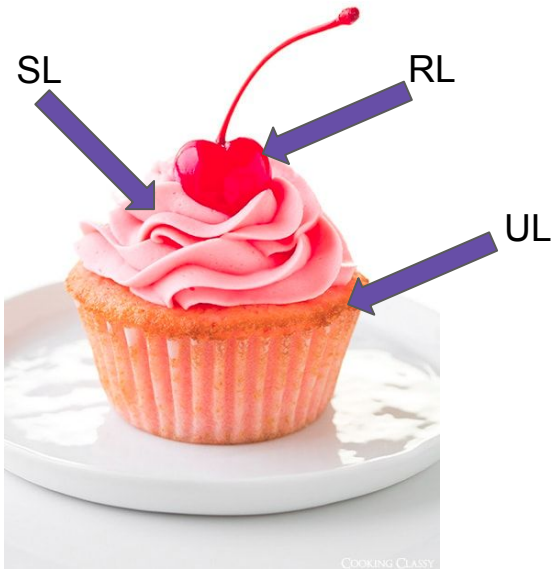
Definition, Key Concepts

What is Unsupervised Learning?

(x1, x2, …, xN) ∈ X with joint density P(X)

Goal: infer properties of P(X) without the help of a supervisor: Clustering and/or feature extraction

X represents all variables → Inference not related to change respect other set



What is Unsupervised Learning?

Possible approaches:

- Global models → GMM
- Various simple descriptive statistics
  - Identify low-dimensional manifolds in X-space → represent high data density
  - Example: Clustering → find convex regions containing modes of P(X)

Association rules → simple descriptions of high density regions

UL Vs SL

- Typically in classification or regression problems (SL)
- Effectiveness reduced by noisy ground truth (SL)
- Depends on availability of annotations (SL)
- <u>Well established metrics (SL)</u>

- Useful in exploratory analysis (UL)
- Greater amount of data, more complex properties (UL)

UL Vs RL

- Goal: maximize the long-term benefits that the agent receives (RL)
- Goal: infer data properties (UL)
- Interacts with its environment (RL)
- Based on Markov’s decision process (RL)
- <u>Performance evaluation based on reward (RL)</u>

History

Motivation and Timeline

Early Motivation

History

1932

Tolma and Craik

Unsupervised Learning ideas based on cognitive processes.

1949

Hebb

Hebbian Learning Rule, links statistical methods to neurophysiological plasticity experiments

1954

Attneave

Importance of redundancy in perception

1959

Rosenblatt

Explain automatic separation and classification of clustered sensory stimuli
.

1960

Watanabe

Pattern recognition based on the principle of minimum entropy

1970

David Marr

Early UL postulate for neocortex model.

Uttley

The *informon*, separate frequently occuring patterns from a background of randomly associated elements.

History

1980

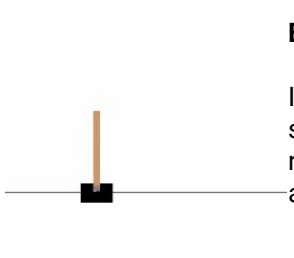
Grossberg

Emphasized the interaction between supervised and unsupervised learning.

1983

Barto

Improved RL pole-balancing scheme by observing pattern of recurring correction-movements and their outcomes.



1985

Linsker

Hierarchical mapping scheme that shows spontaneous self-organization and develops on his infomax principle ideas from Uttley

1986

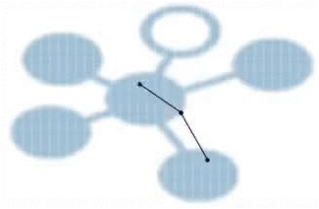
Pearlmutter and Hinton

Unsupervised procedures for discovering regularities.

1989

Rummelhart and Zupser

Competitive Learning



1992

Barlow

Characterized Neural Codes by constructing cognitive maps and working models

Traditional UL Methods

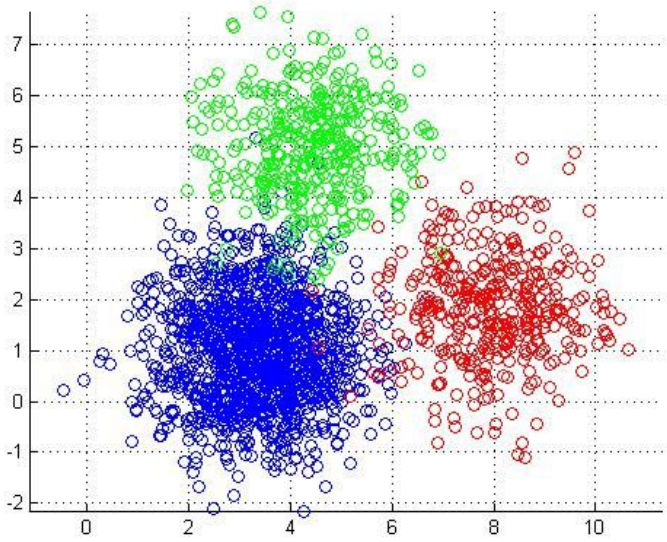
Clustering and Dimensionality Reduction

Machine Learning Problems

	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Clustering

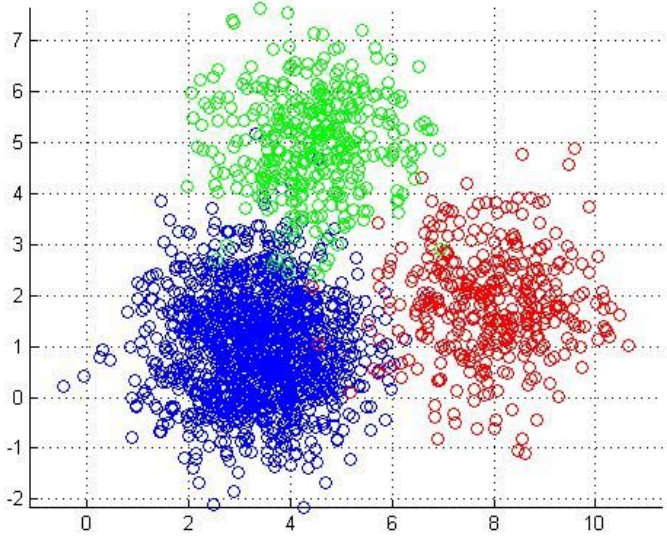
- Intent to organize the data set into subgroups or clusters
- A cluster is a collection of data items which are “similar” between them, and “dissimilar” to data items in other clusters.
- The result depends on the notion of similarity being considered.

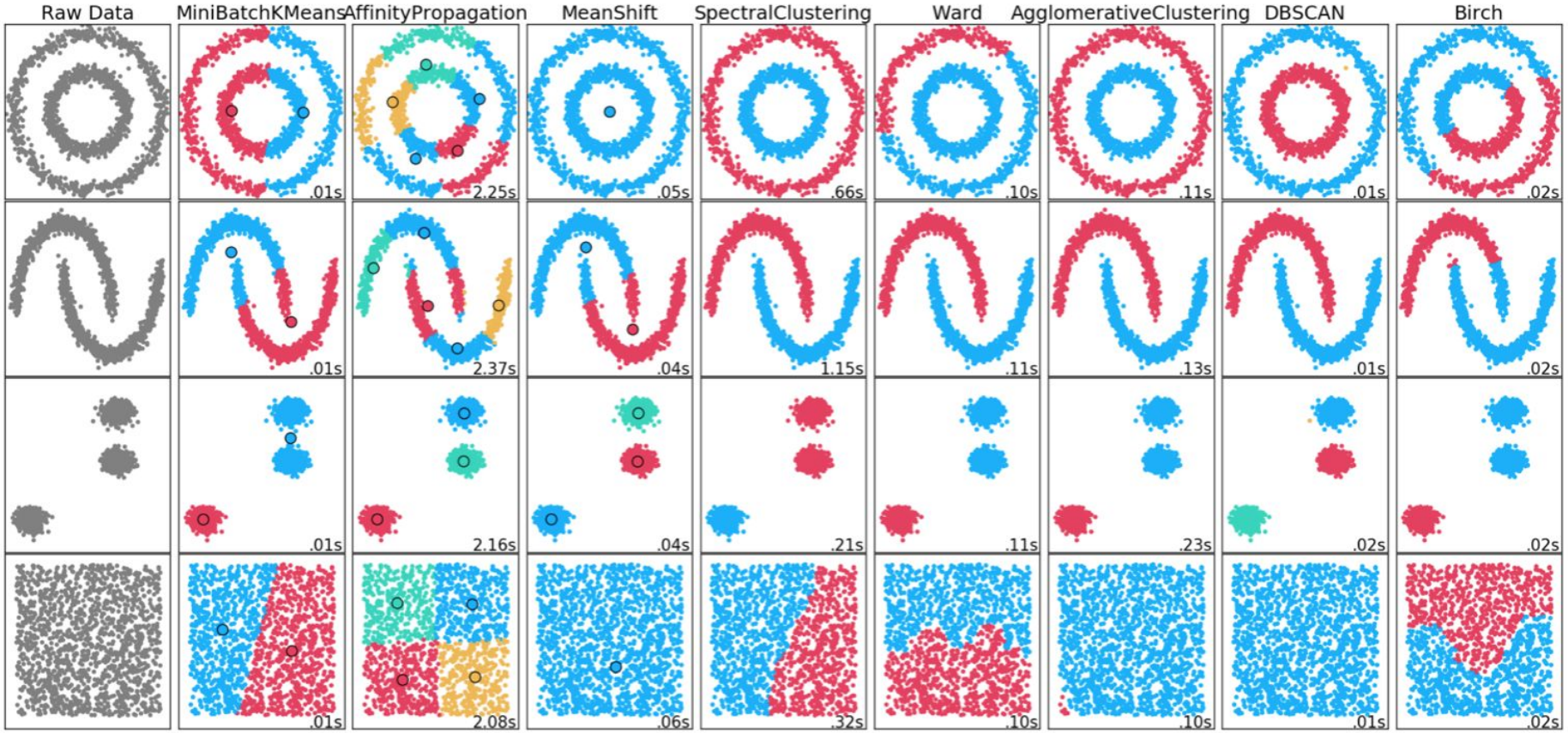


Clustering

- Optimization Problem
- Need for Constraints
  - Minimum distance between clusters
  - Number of clusters
- Evaluation Metrics
- Popular Methods

- K-means
- Mean-Shift
- GMM
- Hierarchical Clusters




Raw Data
MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
AgglomerativeClustering
DBSCAN
Birch
.01s
2.25s
.05s
.66s
.10s
.11s
.01s
.02s
.01s
2.37s
.04s
1.15s
.11s
.13s
.01s
.02s
.01s
2.16s
.04s
.21s
.11s
.23s
.02s
.02s
.01s
2.08s
.06s
.32s
.10s
.10s
.01s
.02s

K-means

- Need to specify the number k of clusters.
- Similarity given by the Euclidean distance.

Pros

- Simple, linear and efficient
- Always terminate in a local optimum

Cons

- Depend on initialization
- Sensitive to outliers

Algorithm 10.1 *K-Means Clustering*

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
   (a) For each of the K clusters, compute the cluster *centroid*. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
   (b) Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf

Gaussian Mixture Model

- K-means generalization with soft assignment
- Each cluster is described as a Gaussian density, using a centroid and covariance matrix.

Pros

- Allows distance to adapt to the data
- Can converge to a local minimum

Cons

- Depend on initialization
- Assumes normal distribution in clusters

E step: estimate ***responsibilities*** (*soft-assignment*) given the model parameters

$$z_{ik} = \frac{1}{Z_i}\pi_k N(x|\mu_k, \Sigma_k), \qquad \sum_k z_{ik} = 1,$$

M step: estimate ***parameters*** $\{\pi_k, \mu_k, \Sigma_k\}$, given the responsibilities

$$\begin{aligned}
\boldsymbol{\mu}_k &= \frac{1}{N_k}\sum_i z_{ik}\boldsymbol{x}_i, \\
\boldsymbol{\Sigma}_k &= \frac{1}{N_k}\sum_i z_{ik}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T, \\
\pi_k &= \frac{N_k}{N},
\end{aligned}$$

Mixing coefficient: Estimation of the fraction of points in each component

where

$$N_k = \sum_i z_{ik}.$$

Requires knowledge of the number of models k.

Mean-Shift

1. Initialize random seed, and window W
2. Calculate center of gravity (the "mean") of W: $\sum_{x \in W} x H(x)$
3. Shift the search window to the mean
4. Repeat Step 2 until convergence

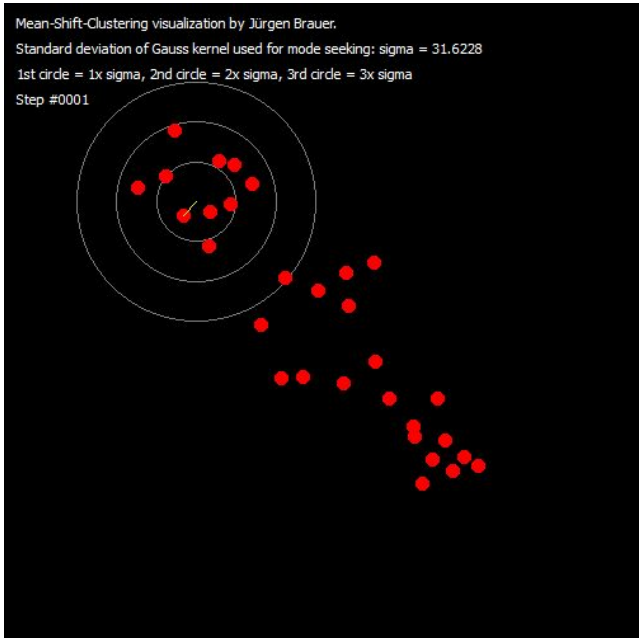
- Aims to discover blobs in a smooth density of samples.
- Update candidates for centroids to be the mean of the points within a given region.

Pros

- Robust to outliers
- Finds variable number of modes

Cons

- Computationally expensive
- Depends on the window size



https://courses.csail.mit.edu/6.869/handouts/PAMIMeanshift.pdf
http://vision.stanford.edu/teaching/cs131_fall1314_nope/lectures/lecture13_kmeans_cs131.pdf

Hierarchical Clustering

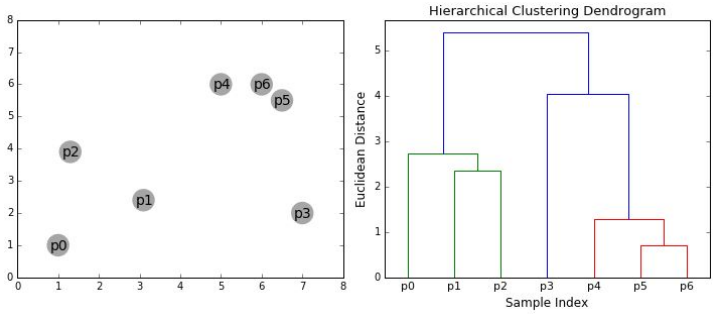
- No need to specify a particular number of clusters K
- Divisive or agglomerative
- Dendrogram: Tree-based representation
- Linkage: New dissimilarity measure between a pair of observations

Pros

- Adaptable with flexible distance metrics
- Richer representation than a fixed partition

Cons

- Slow algorithm for large scale problems



Algorithm 10.2 *Hierarchical Clustering*

1. Begin with n observations and a measure (such as Euclidean distance) of all the $\binom{n}{2} = n(n-1)/2$ pairwise dissimilarities. Treat each observation as its own cluster.

2. For $i = n, n-1, \ldots, 2$:

   (a) Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.

   (b) Compute the new pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.

http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf

Machine Learning Problems

	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

ICA

- Separates mixed signals and find independent components
- We are looking for independent and orthonormal basis for the data:

  basis $(\phi_1, ..., \phi_n)$ such that $\phi_i \cdot \phi_j = 0$ if $i \neq j$ and 1 if $i = j$

- Objective function is:

$$J(W) = \|W_x\|_1$$

  Where there is a constraint: $WW^T = I$

Taken from http://ufldl.stanford.edu/tutorial/unsupervised/ICA/

ICA

How to optimize objective:

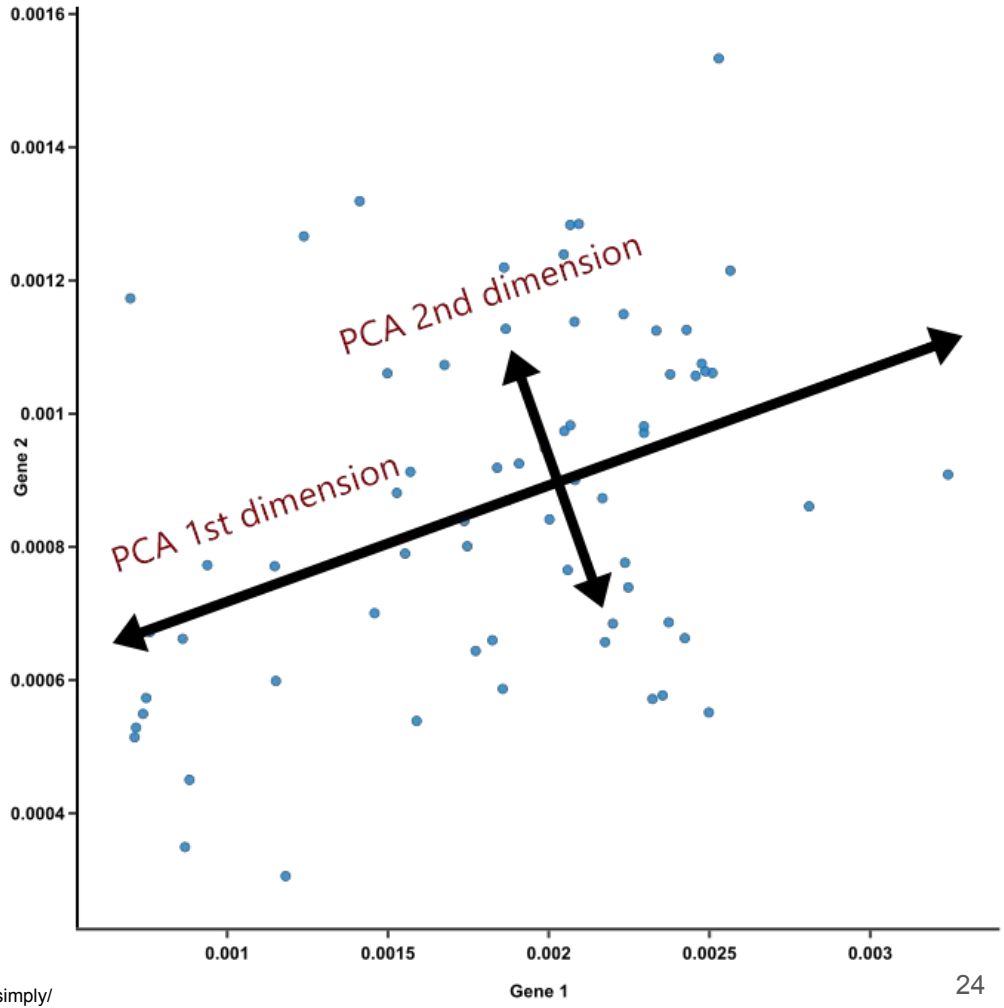
$$W \leftarrow W - \alpha \bigtriangledown_W \|W_x\|_1$$

$W \leftarrow proj_U$ Where *U* is the space of matrices satisfying $WW^T = I$

Taken from http://ufldl.stanford.edu/tutorial/unsupervised/ICA/

PCA

- Finds a low-dimensional representation of a data set that contains as much as possible of the variation
- Each dimension is a linear combination of the p features



Taken from https://blog.bioturing.com/2018/06/14/principal-component-analysis-explained-simply/

PCA

How we find these components?

First principal component of a set of features *X1, X2, …, Xp* is the normalized linear combination of the features:

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \ldots + \phi_{p1} X_p$$

It is normalized for $\sum_{i=1}^{p} \phi_{i1}^2 = 1$ where ϕ correspond to the loadings

The optimization problem solved by the first component is:

$$\max_{\phi_{11},\ldots,\phi_{p1}} \left\{ \frac{1}{n} \sum_{j=1}^{n} \left(\sum_{j=1}^{n} \phi_{j1} x_{ij} \right)^2 \right\} subject\ to \sum_{j=1}^{n} \phi_{j1}^2 = 1$$

Taken from hhttp://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf

PCA

Second principal component: linear combination of *X1, …, Xp* with maximal variance

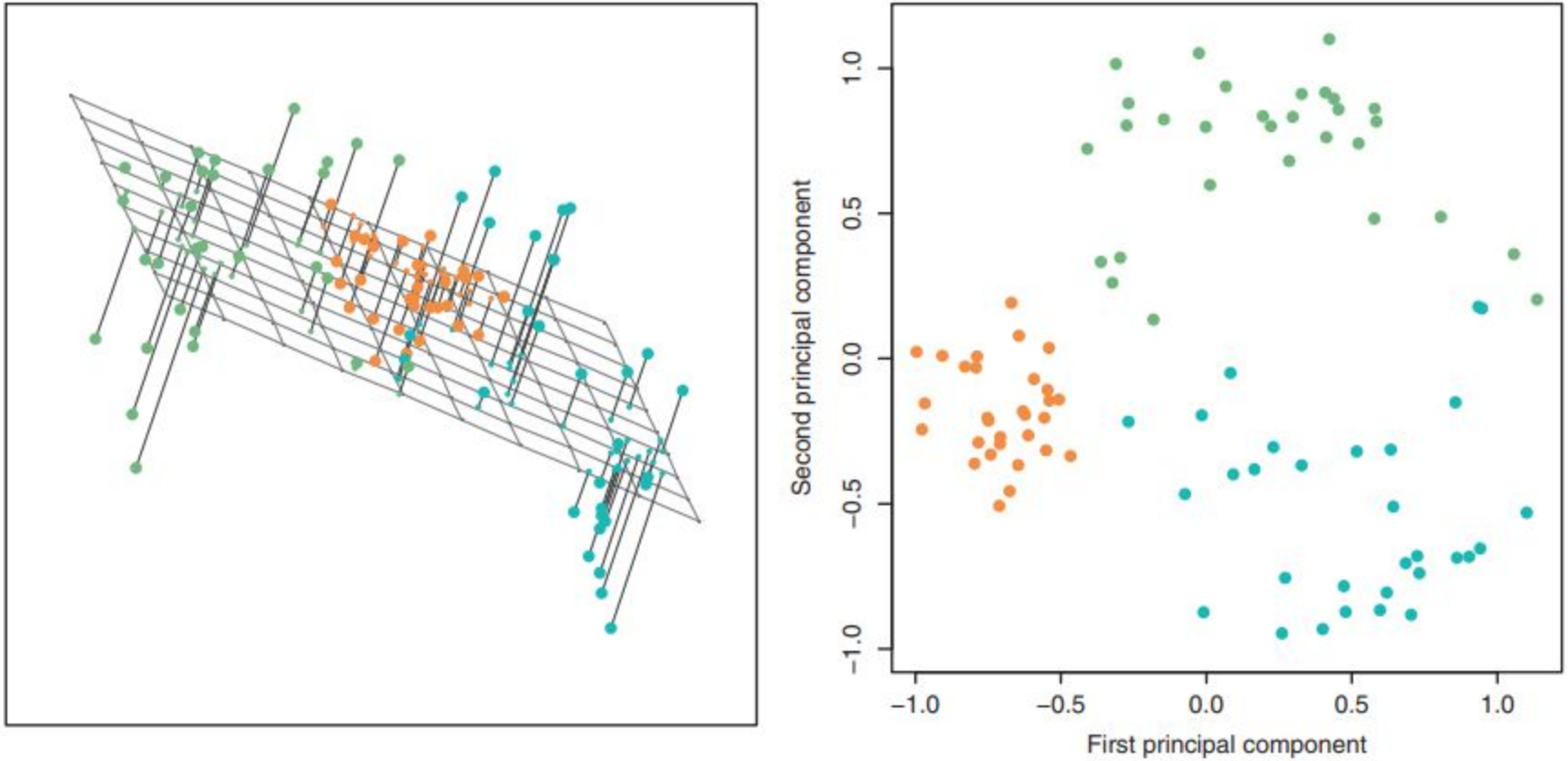
Scores with the form:

$$z_{i2} = \phi_{12} x_{i1} + \phi_{22} x_{i1} + \ldots + \phi_{p2} x_{ip}$$

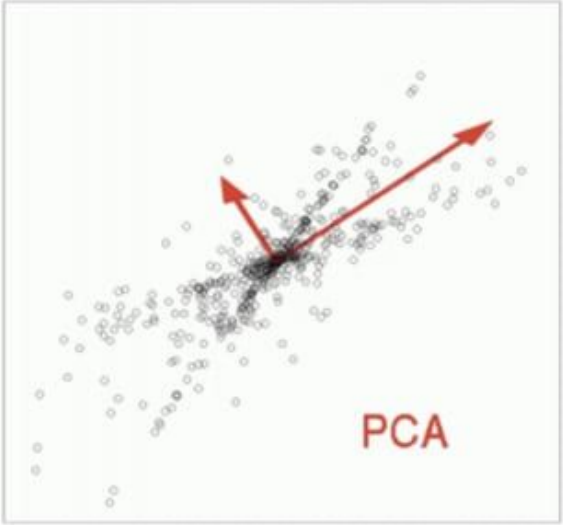
It is uncorrelated with *Z1*, is orthogonal

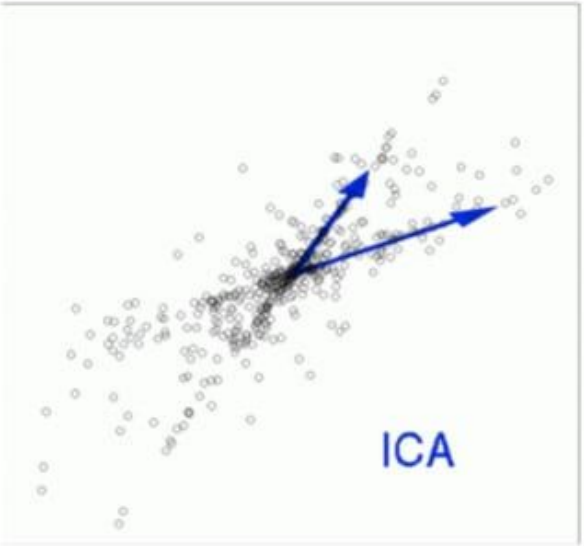
If we are in a 2D space, there is only two possible principal components. In high dimensional space, there are multiple principal components calculated similarly

Taken from hhttp://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf


Second principal component
First principal component
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

PCA vs ICA





Taken from https://dev.to/swyx/unsupervised-learning-feature-transformation-pcf

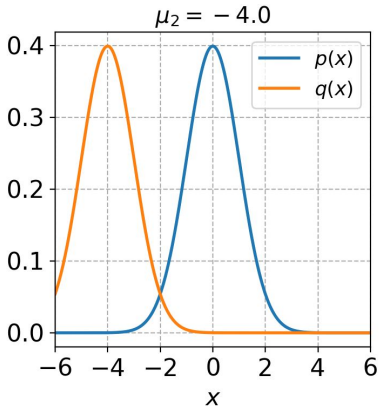
Metrics

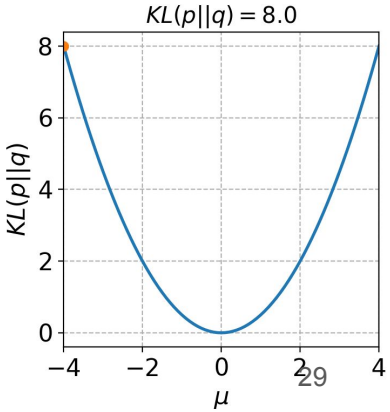
- Not a direct measure of success
- Heuristic arguments for motivating algorithms and judge quality of results

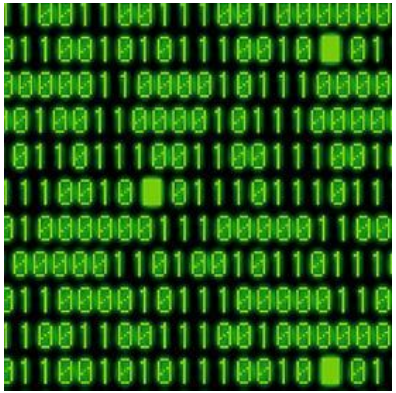
Example: KL divergence

$$KL(P||Q) = \sum p_i(x) log(\frac{p_i(x)}{q_i(x)})$$

Measure of relative entropy



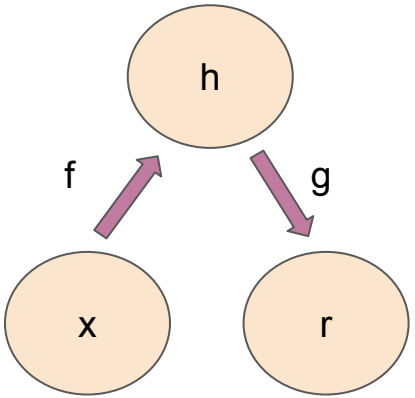




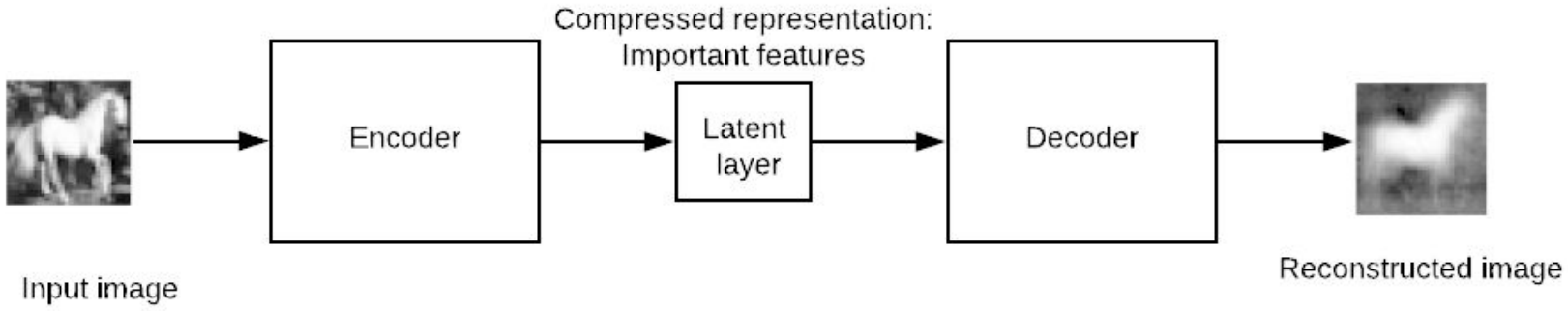
Autoencoders

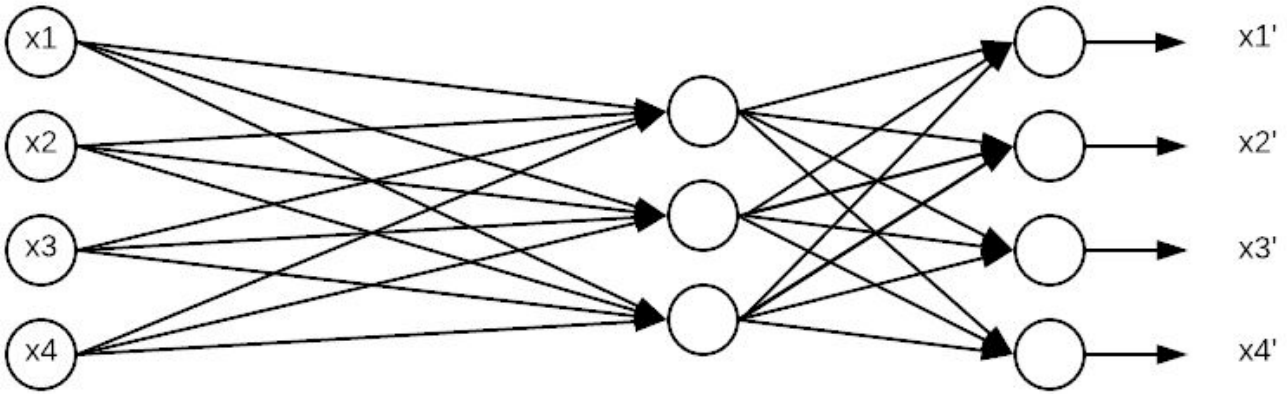
Autoencoders

- Neural network attempt to copy input to output
- Two parts:
  - Encoder h = f(x)
  - Decoder r = g(h)
- Traditionally used for:
  - Dimensionality reduction
  - Feature learning



Taken from https://www.deeplearningbook.org/contents/autoencoders.html





Taken from https://www.guru99.com/autoencoder-deep-learning.html

Undercomplete Autoencoders

- h has a smaller dimension than x
- Code dimension is less than input dimension
- Captures most important features of training data
- Not useful if given to much capacity
- Learning process described as minimizing a loss function L:

$$L(x, g(f(x)))$$

If L = MSE and the decoder is linear → Same subspace as PCA

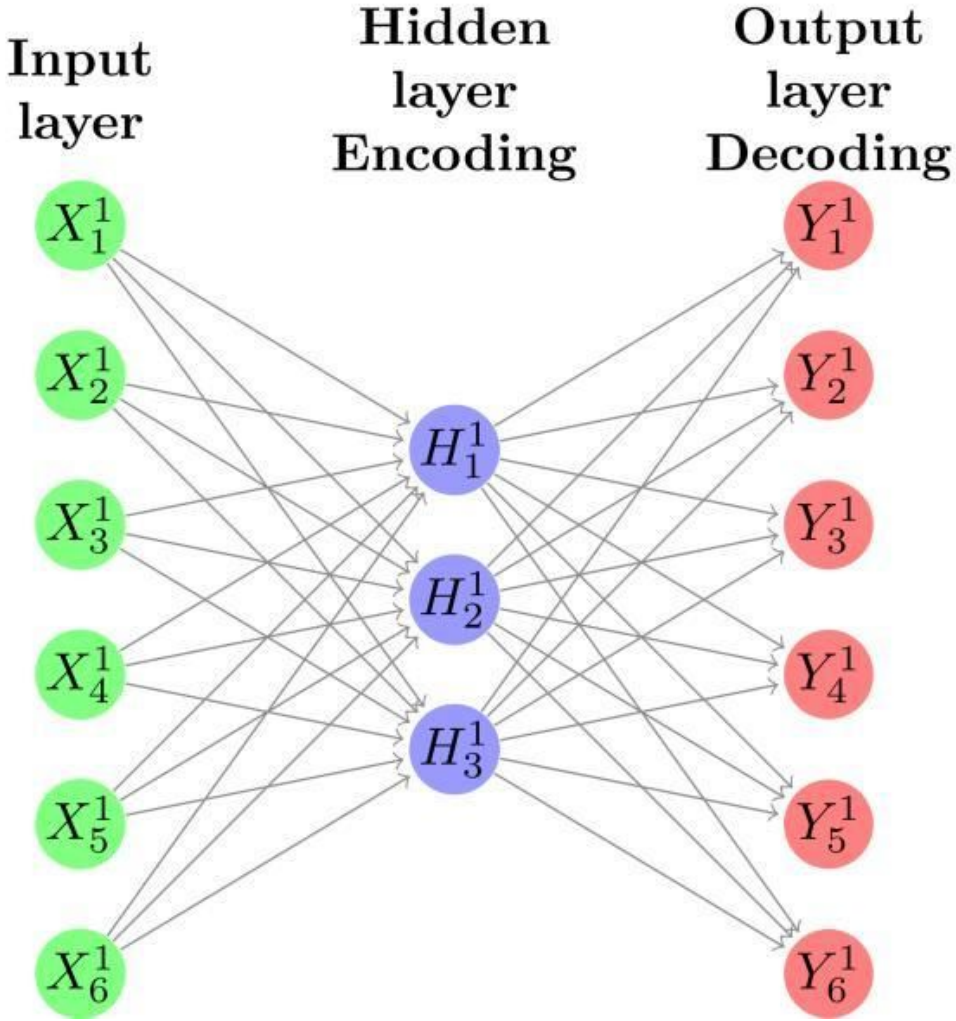
Non-linear encoder and decoder → generalization on PCA

Regularizing
Sparse Autoencoders

- Training criterion includes a sparsity penalty $\Omega(h)$ on the h layer
- Has a reconstruction error:

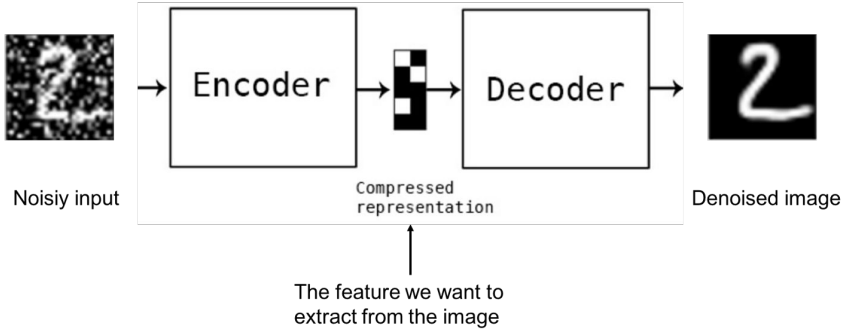
$$L(x, g(f(x))) + \Omega(h)$$

- Respond to unique statistical features of train dataset
- Typically used to learn features for another task
- $\Omega(h)$ acts like a regularizer


Input layer
Hidden layer Encoding
Output layer Decoding
X_1^1
X_2^1
X_3^1
X_4^1
X_5^1
X_6^1
H_1^1
H_2^1
H_3^1
Y_1^1
Y_2^1
Y_3^1
Y_4^1
Y_5^1
Y_6^1

Regularizing
Denoising Autoencoders

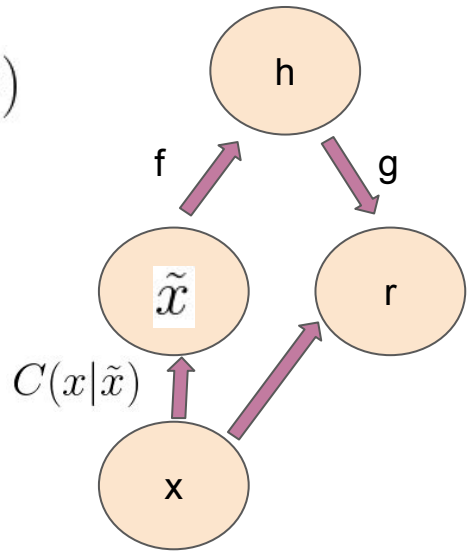
- Changes the reconstruction error term of the cost function
- It minimizes a function: $L(x, g(f(\tilde{x})))$ where $\tilde{x}$ is a copy of x corrupted by some noise
- Receives corrupted data and predicts the original



Regularizing
Denoising Autoencoders

Learns a reconstruction distribution $p_{rec}(x|\tilde{x})$ for $(x, \tilde{x})$

1. Sample a training example x from training data
2. Sample corrupted version $\tilde{x}$ from $C(\tilde{x})$
3. Use $(x, \tilde{x})$ as a training example for estimating the autoencoder reconstruction distribution



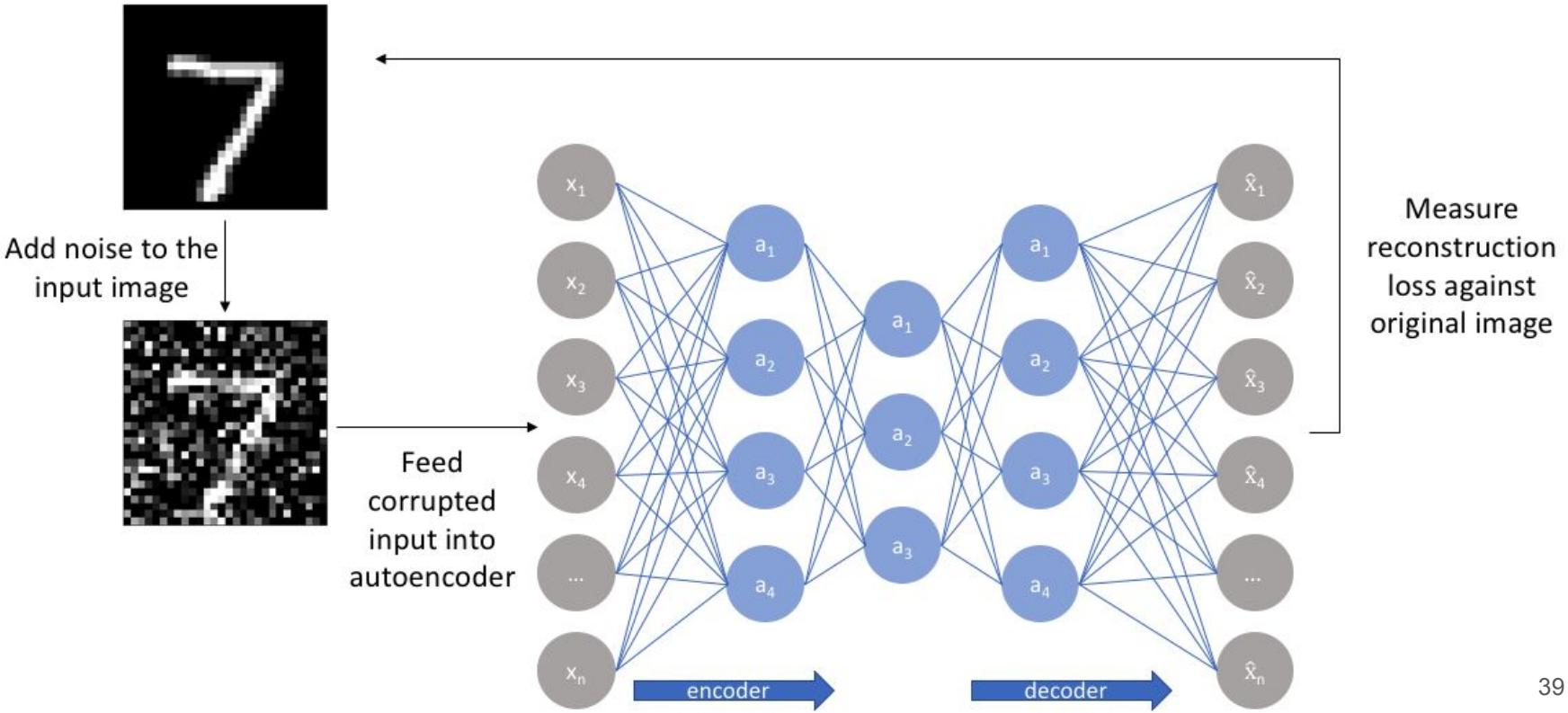
Regularizing
Contractive Autoencoders

- Introduces an explicit regularizer on the code h = f(x)
- Encourages derivatives to be as small as possible
- Penalty is the squared Frobenius norm of the Jacobian matrix of partial derivatives associated with the encoder function:

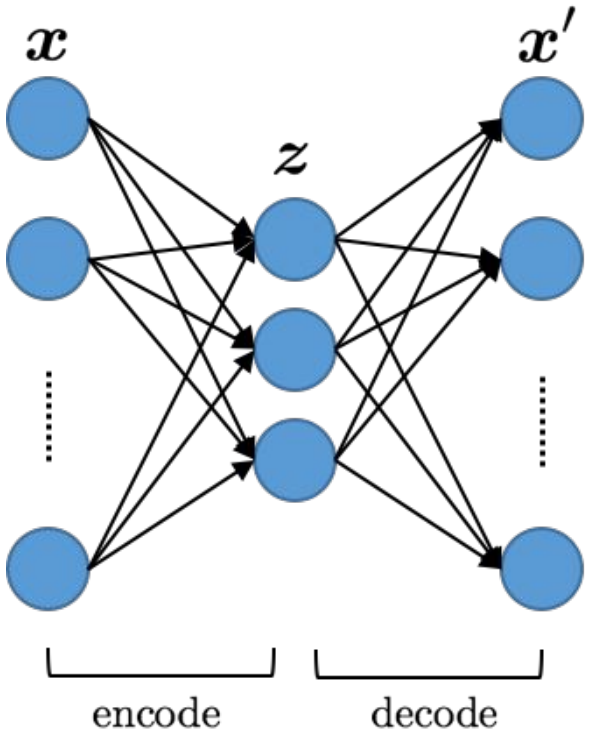
$$\Omega(h) = \lambda \left\| \frac{\partial f(x)}{\partial x} \right\|_F^2$$

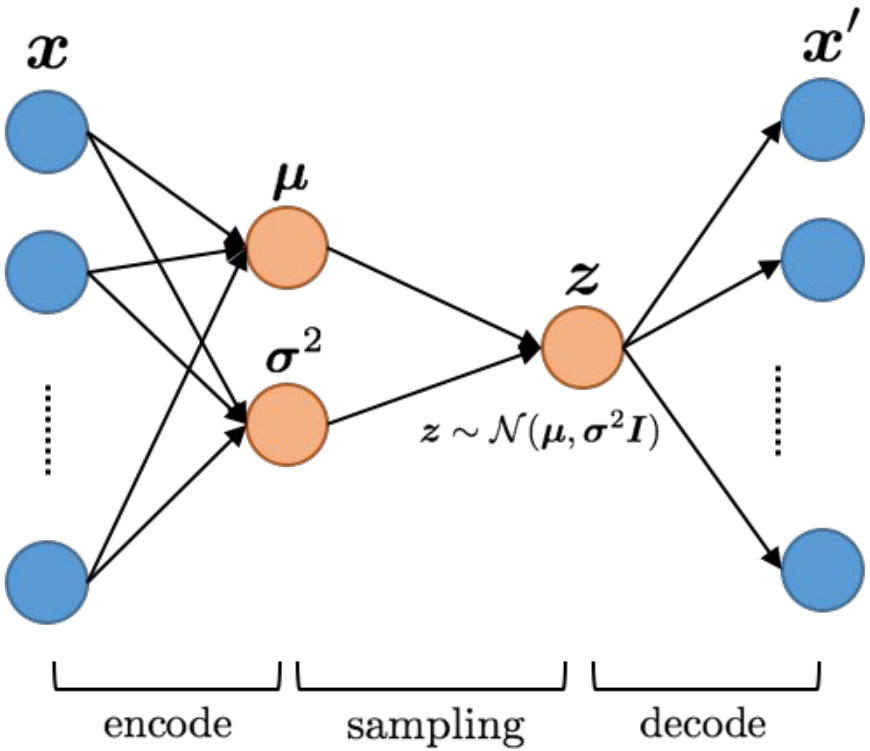
- Resists perturbations of its input

Contractive autoencoders



Variational Autoencoders





Variational Autoencoders

- Assume an underlying distribution of source data
- Search the parameters of that distribution
- Reparametrization trick: $z = \mu + \sigma \odot \varepsilon$

$$\varepsilon \quad \text{Normal}(0, 1)$$

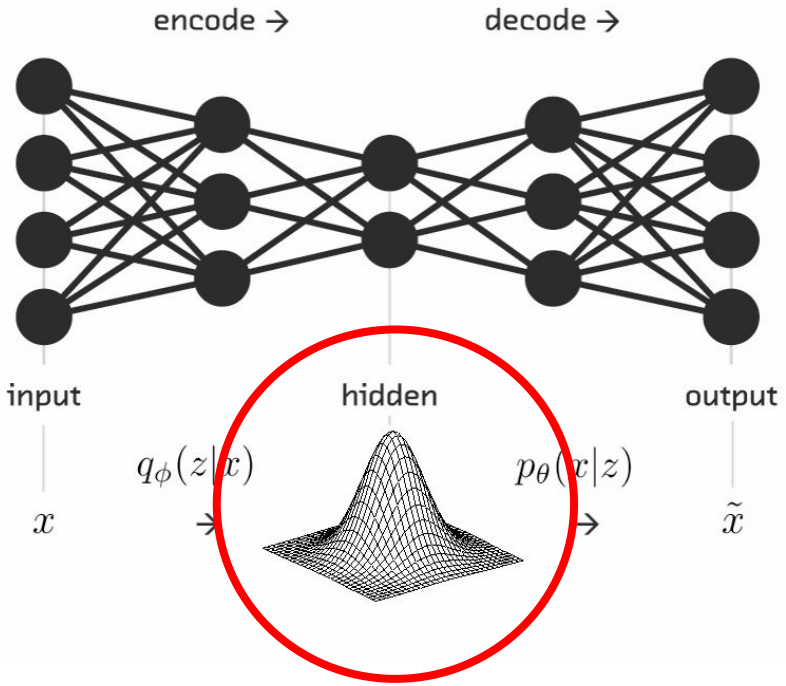
$$L(\theta, \phi;\ \mathbf{x}, \mathbf{z}) = \mathbb{E}_{q_\phi(\mathbf{x}|\mathbf{z})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{x}|\mathbf{z})||p(z))$$

Deep Unsupervised Learning

Generative Models and Self-Supervised Learning

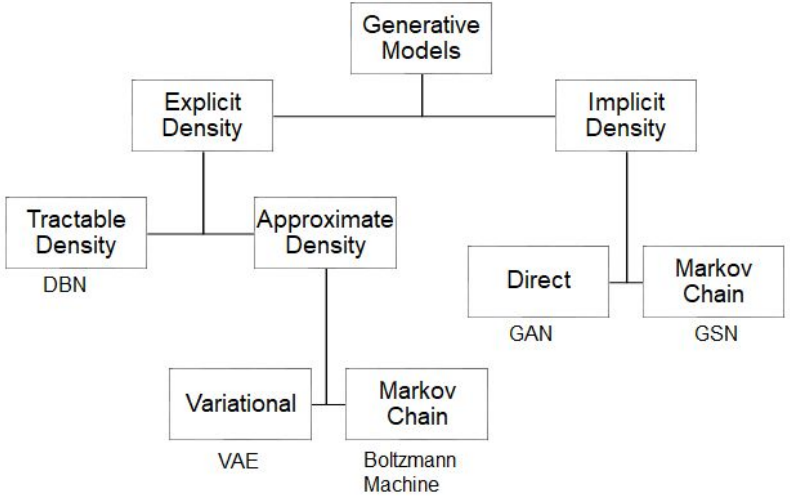
VAE are Generative Models

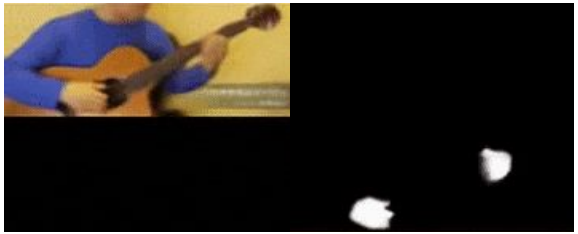
- Variational Autoencoders give a probabilistic spin to autoencoders.
- Map the input into a probability distribution at the bottleneck.
- Sample from the distribution to decode.



Generative Models

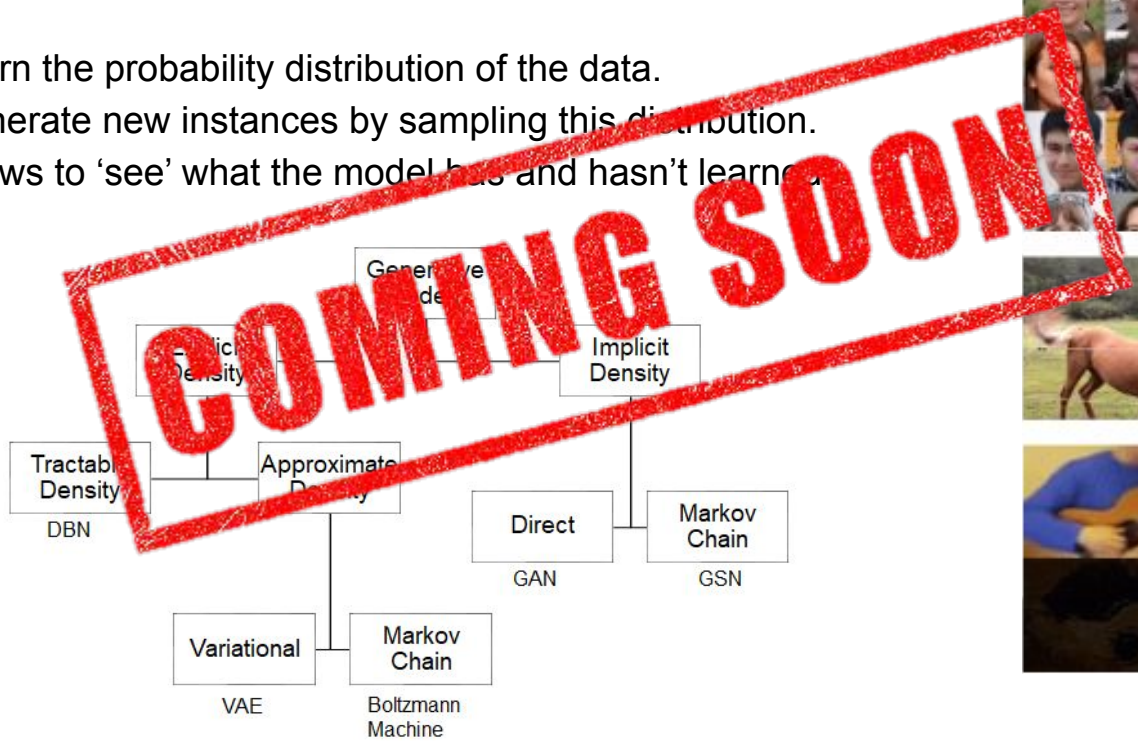
- Learn the probability distribution of the data.
- Generate new instances by sampling this distribution.
- Allows to ‘see’ what the model has and hasn’t learned.

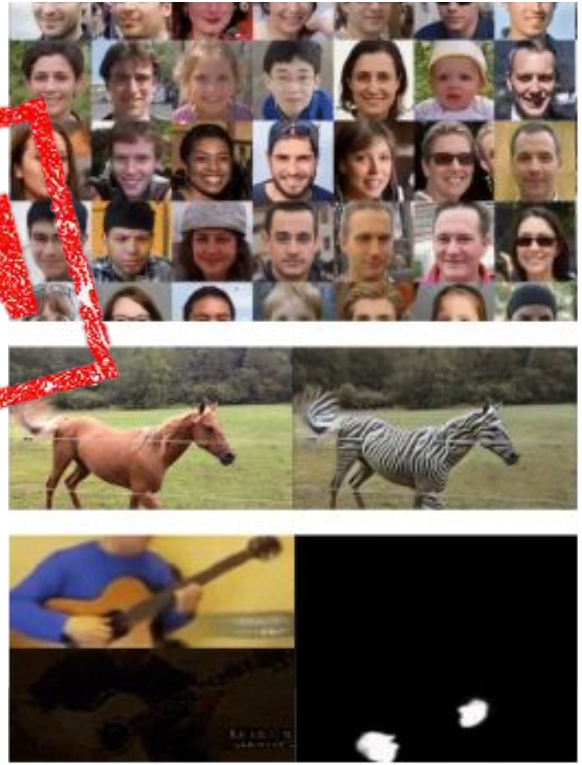




Generative Models

- Learn the probability distribution of the data.
- Generate new instances by sampling this distribution.
- Allows to ‘see’ what the model has and hasn’t learned.





Self-Supervised Learning SSL

We learn models of the world by prediction

- Predict any part of the input from other part.
- Learn by training from the data itself.
- Fill in the blanks,

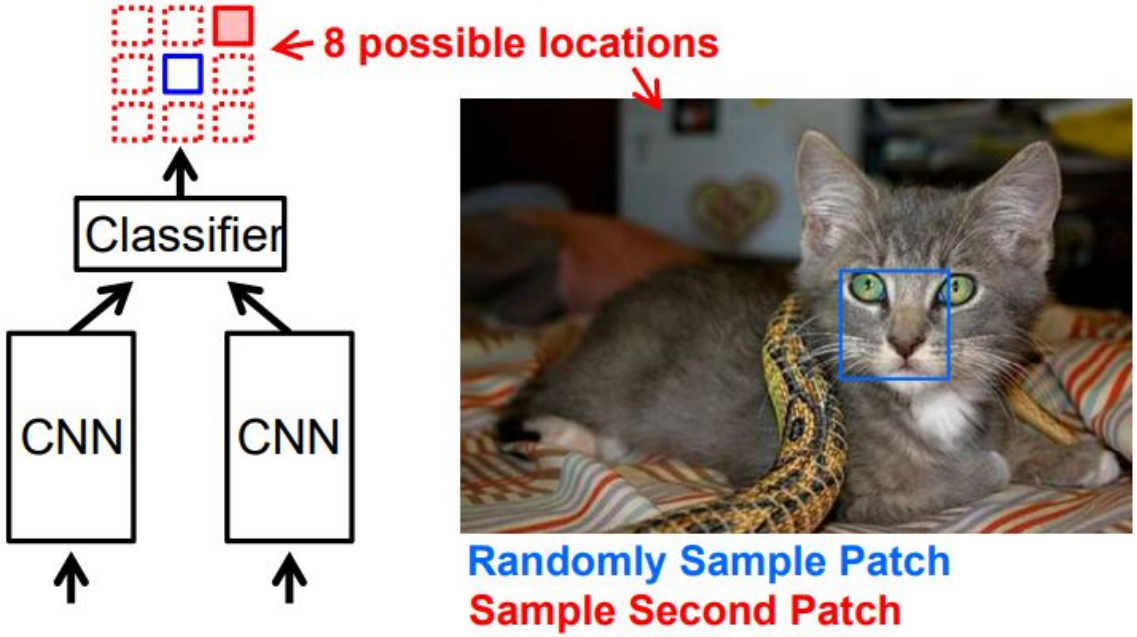
“Is just a form of supervised learning but the supervisory signal is the input itself, at a different location of time”

-LeCun, 2018

https://www.youtube.com/watch?v=7I0Qt7GALVk

Example: Relative Positioning

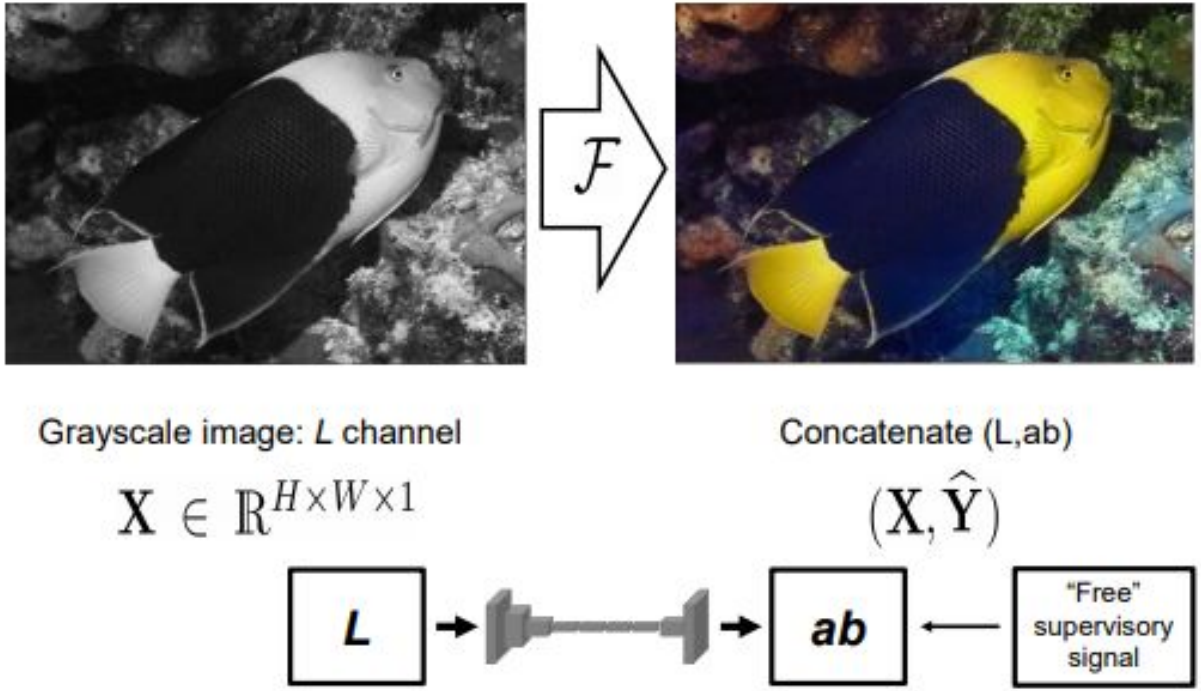
Train network to predict relative position of two regions in the same image



Unsupervised visual representation learning by context prediction,
Carl Doersch, Abhinav Gupta, Alexei A. Efros, ICCV 2015

https://arxiv.org/pdf/1505.05192.pdf

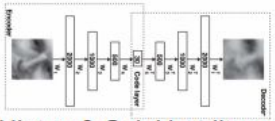
Example: Image Colourization



https://arxiv.org/pdf/1603.08511.pdf
Try it Here: https://demos.algorithmia.com/colorize-photos/

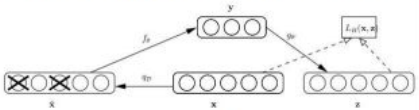
Other examples

Autoencoders



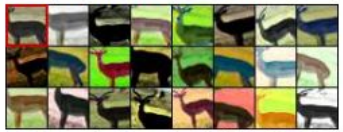
Hinton & Salakhutdinov. Science 2006.

Denoising Autoencoders



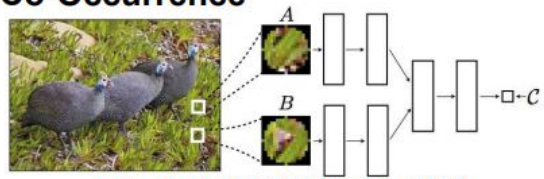
Vincent *et al.* ICML 2008.

Exemplar networks



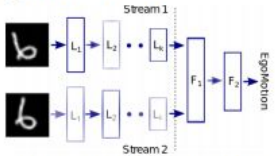
Dosovitskiy *et al.*, NIPS 2014

Co-Occurrence



Isola *et al.* ICLR Workshop 2016.

Egomotion



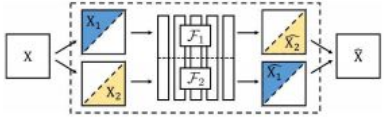
Agrawal *et al.* ICCV 2015 Jayaraman *et al.* ICCV 2015

Context

Noroozi et al 2016

Pathak *et al.* CVPR 2016

Split-brain auto-encoders



Zhang *et al.* CVPR 2017

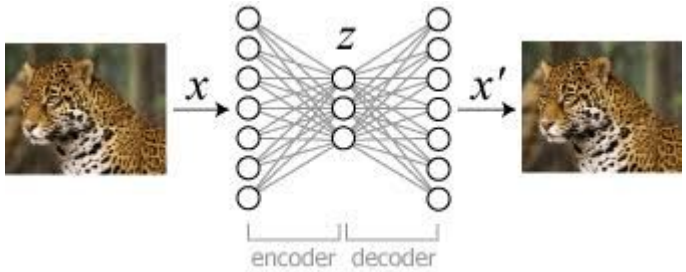
https://project.inria.fr/paiss/files/2018/07/zisserman-self-supervised.pdf

Types of Self-Supervised Learning

Selection of the Labels

Local Approach

- Use data as labels directly
- Ex. Autoencoders



Global Approach

- Takes into account relationships between data to make the labels
- Ex. Deep Clustering

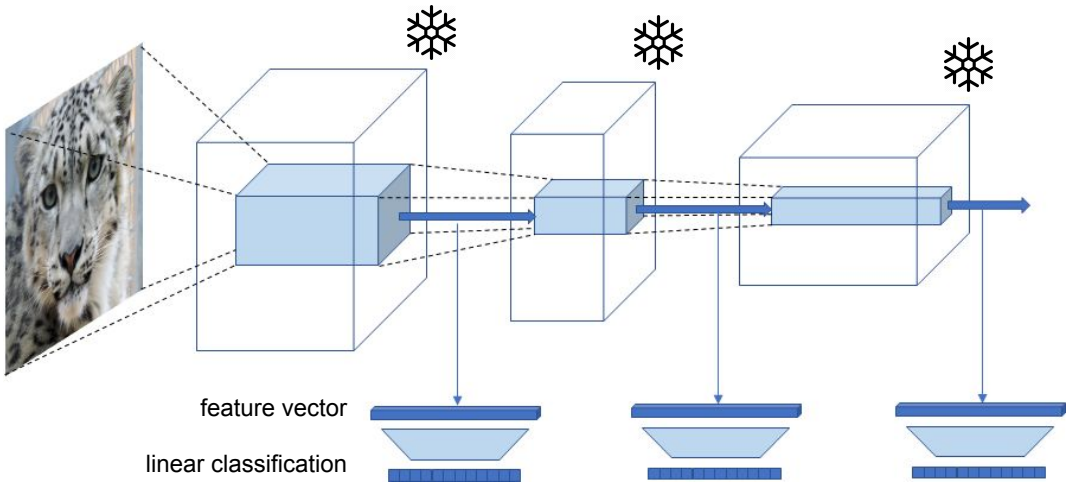
(a) Raw data (b) ConvAE (c) Our method

https://arxiv.org/pdf/1807.05520.pdf

https://arxiv.org/pdf/1904.13113.pdf

Unsupervised Feature Representation

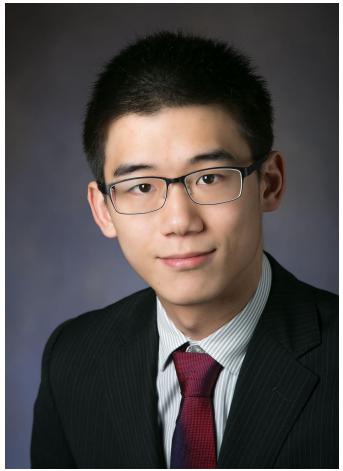
- Test the task generalization capability of the features
- Fixing the learning representation with frozen weights
- Train linear classifiers on top of each layer with supervision
- Compare accuracy between layers, randomly assigned filters and supervised approach



https://arxiv.org/pdf/1807.05520.pdf
https://arxiv.org/pdf/1611.09842.pdf
https://arxiv.org/pdf/1603.08511.pdf

Papers

Unsupervised Textual Grounding: Linking Words to Image Concepts



Raymond A. Yeh

Minh N. Do

Alexander G. Schwing

Textual grounding

I will give a prize to the first one who finds the dinosaur with a man costume

Textual grounding

- Linking words to objects
- Important task for robotics, human-computer interaction
- Computer find the task complex due to object variability and ambiguity

A woman giving a presentation about a Chevrolet car.

Rocky spot on the left, bottom of the picture.

Useful progress

- Deep net based object detection
- Object classification
- Semantic segmentation

Relies on availability of bounding boxes

Completely unsupervised mechanism based on hypothesis testing

Related Work
Textual Grounding

- First was related to image retrieval
- Classic approaches:
  - Recurrent neural networks
  - Metric learning
  - Correlation analysis
  - Neural net embeddings
- Others: based on region proposal
- This work: Usage of efficient subwindow search

there is a cat sitting on a shelf .

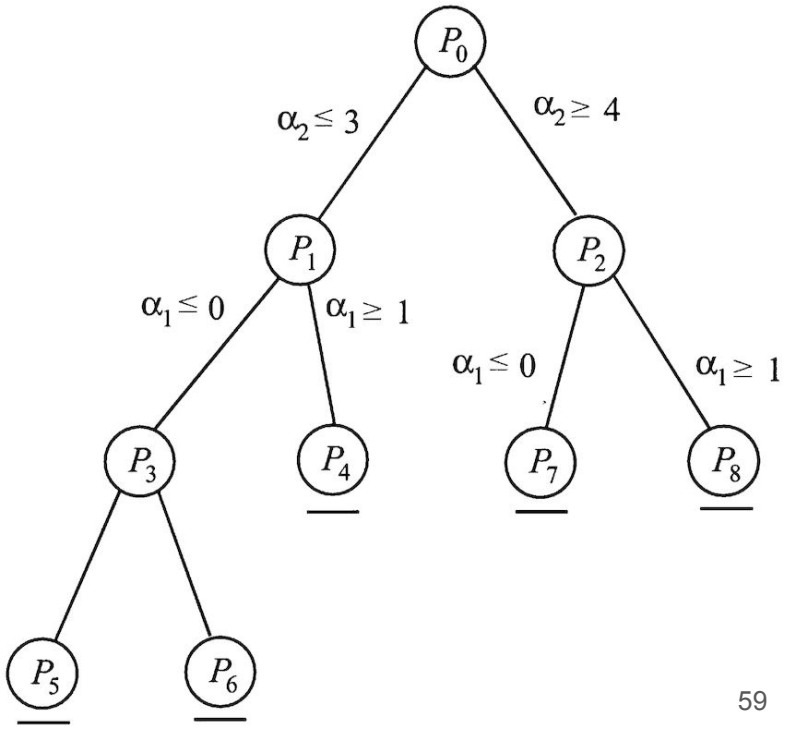
a kitchen with stainless steel appliances .

Taken form: https://arxiv.org/pdf/1411.2539.pdf

Related Work

Efficient subwindow search (Lampert *et al*, 2009)

- Method for object localization
- Branch and bound → Find the global optimum of a quality function
- Analize all possible subimages in the possible candidate image
- Same object locations that an exhaustive sliding window
- Fewer evaluations → Faster



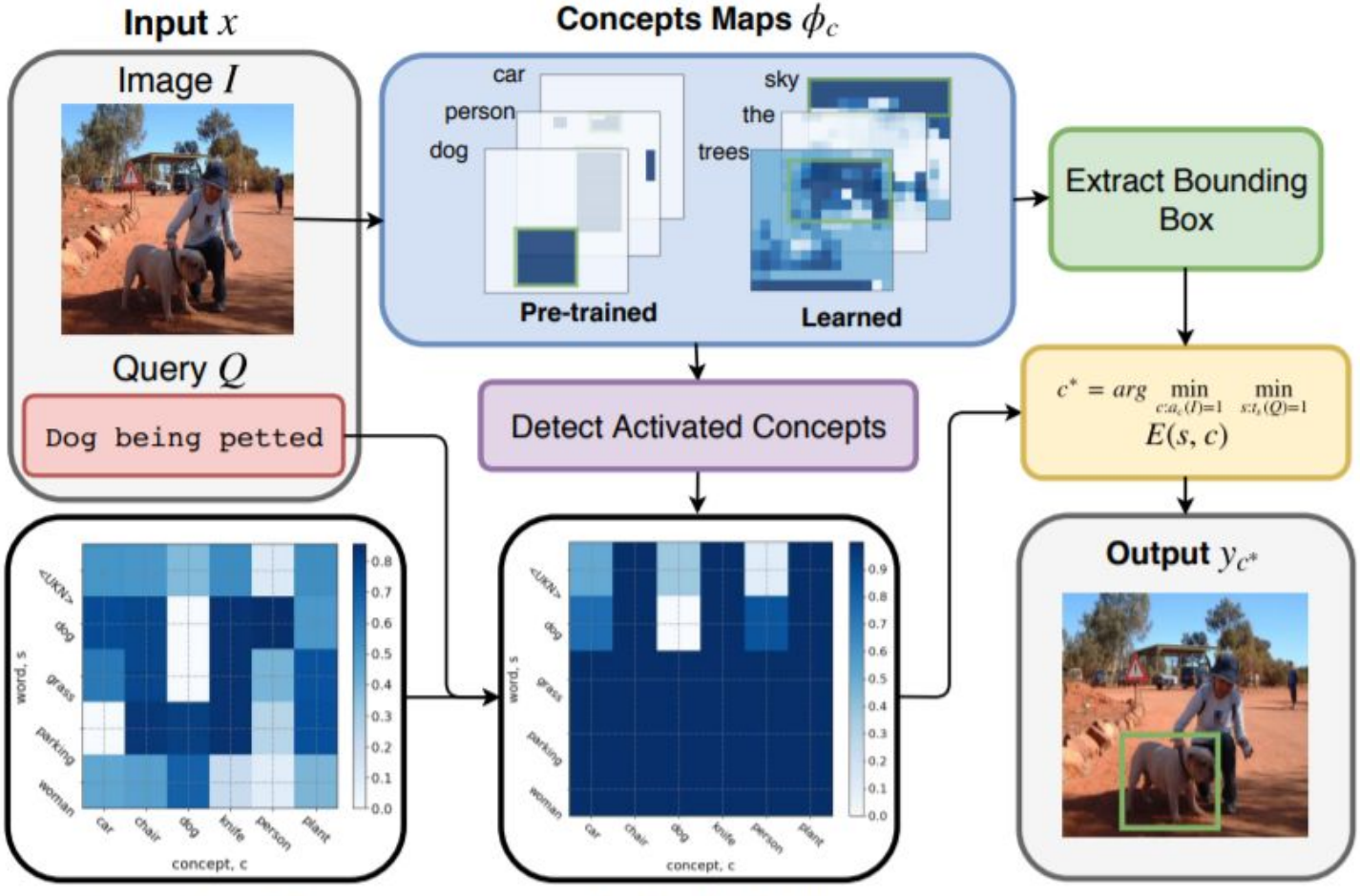
Taken form: https://www.researchgate.net/figure/An-example-of-a-branch-and-bound-tree_fig2_3325941

Related Work
Visual attention

- Image attention modules → VQA
- Others: approach model attention as a single network
- This work: takes into account multiple modalities, easy to extend

Taken form: https://arxiv.org/pdf/1612.00837.pdf

Model



Concept-word relevance $E(s, c)$

Input activated $E(s, c)$

Notation

$x = (Q,\ I) \longrightarrow$ Input

$y = (y_1,\ y_2,\ y_3,\ y_4)\ \ \longrightarrow$ Parametrization of bounding box

$$y \in \mathcal{Y} = \prod_{i=1}^{4} \{0, \ldots, y_{i,\max}\}$$

$\mathcal{D} = \{(x)\}$

$E(s,\ c) \longrightarrow$ ‘image concept’-word relevance

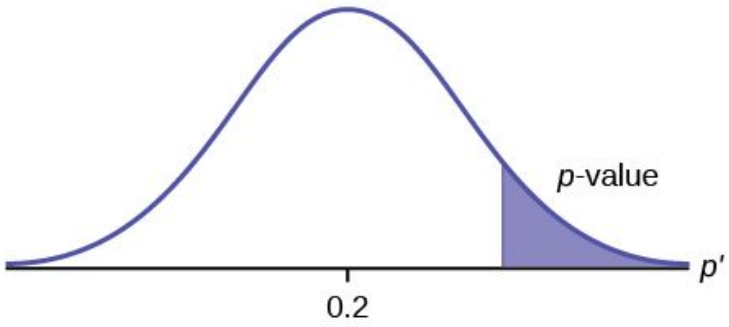
$a_c(I) \in \{0, 1\} \sim \textit{Bernoulli}\ \ \longrightarrow$ image concept is activated in image

Learning

- Statistical hypothesis testing:

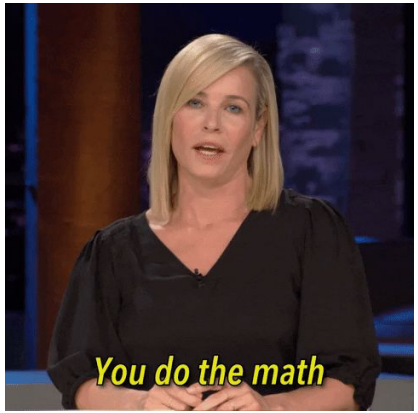
$$H_0(s,\ c):\ P\left(a_c = 1 \mid t_s = 1\right) = P\left(a_c = 1\right),$$

$$H_1(s,\ c):\ P\left(a_c = 1 \mid t_s = 1\right) > P\left(a_c = 1\right).$$



- Probability of an activated concept is independent of observing an activated word
- One-sided hypothesis test

Learning



- $N(s) = \sum_{Q \in D} t_s(Q) \rightarrow$ count of word s
- $N(s,c) = \sum_{(Q,I) \in D} a_c(I) \bullet t_s(Q) \rightarrow$ count of s and c co-occur
- $N(s,\ c) \leq N(s)$
- $n_{s,c} \rightarrow\ P\left(n_{s,c}\right) = Bin\left(N(s),\ P\left(a_c = 1 \mid t_s = 1\right)\right)$

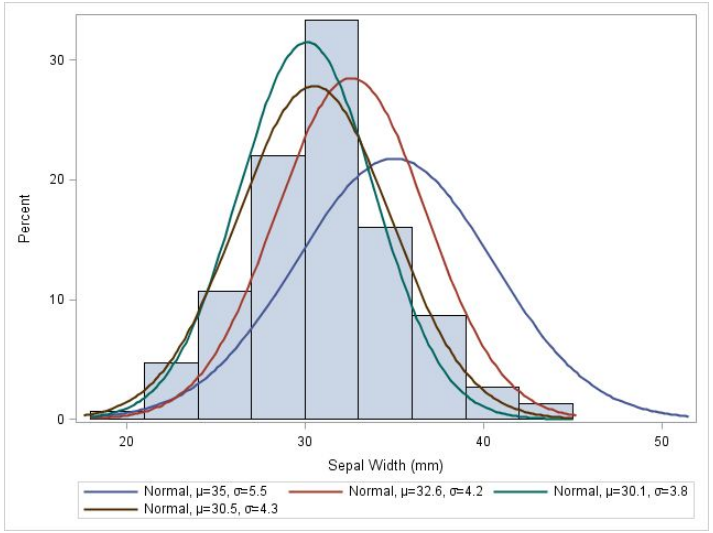
Because of the Central Limit Theory, $n_{s,c}$ approximates to a normal distribution

$$\left(\mu :\ N(s)P\left(a_c = 1 \mid t_s = 1\right),\ \sigma^2 :\ N(s)P\left(a_c = 1 \mid t_s = 1\right)\left(1 - P\left(a_c = 1 \mid t_s = 1\right)\right)\right)$$

Learning

$$E(s,\ c) = P\left(n_{s,\ c} > N(s,\ c) |\ H_0(s,\ c)\ True\right) = \frac{1}{2} - \frac{1}{2} erf\left(\frac{N(s,c) - \mu}{\sigma\sqrt{2}}\right)$$

We need $P(a_c = 1)$ so we use the maximum likelihood estimator:



$$N(c)\,/|\mathcal{D}|\ while\ N(c) \sum_{(Q,I)\in\mathcal{D}} a_c(I)$$

Inference

Each image concept c is represented by a score map $\phi_c(I) \in \mathbb{R}^{W \times H}$

1. Detect a bounding box for each score map with efficient subwindow search
   - It covers at least 5% of mage and has confidence > 0.5
   - Use token activations from query Q
2. Find concept c*, has the lowest evidence of H_0 = True: $c* = \arg \min_{c \in C: a_c(I)=1} \min_{s \in S: t_s(q)=1} E(s,\ c)$
3. Obtain bounding box estimate for concept ϕ_{c*} via efficient subwindow search
   - It covers at least 5% of mage and has confidence > 0.5
   - $E(s,\ c)$ >>, bounding box is the entire image

Image concepts

Semantic segmentation

- Estimated class probability by pixel
- Spatial and concept information

- Threshold probability at 0.5
- $P(x) > 0.5 \rightarrow x = 1$
- $P(x) < 0.5 \rightarrow x = -1$

Object detection

- Set of boxes with class label
- Estimated probability

- Drop boxes with confidence less than 0.5
- Score map → assign 1 to pixels in box, -1 otherwise

Learned image classification network

- Probability of each class
- No spatial information

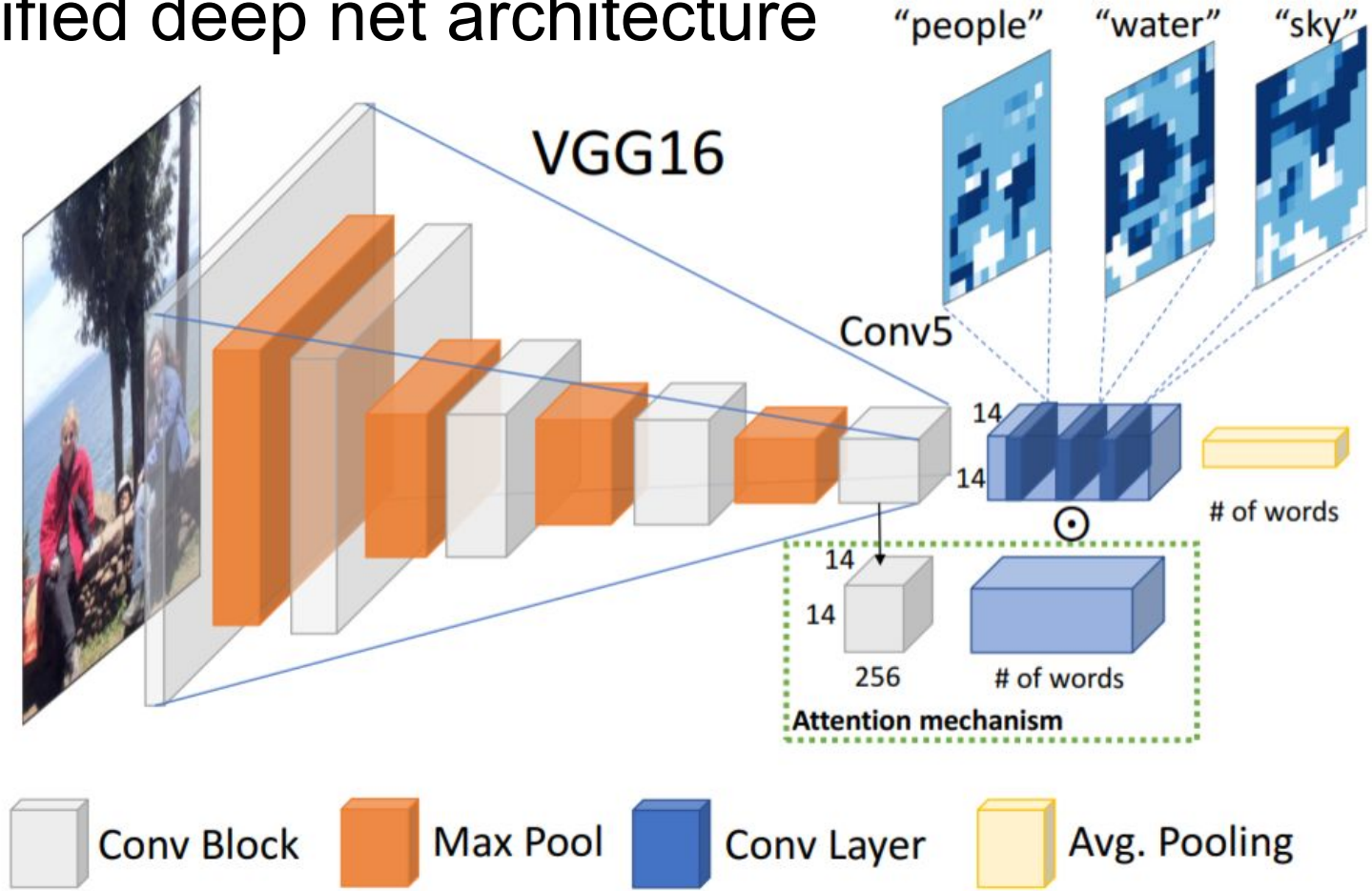
- Proposed a modified deep net architecture
- Retain spatial information

Modified deep net architecture

- Minimizes negative cross-entropy between ground truth $t_s(Q)$ and estimated probability $p_\theta\left(t_s(Q)|I\right)$
- Uses negative binary cross-entropy loss

$$\mathcal{L} = \sum_{(I,Q)\in\mathcal{D},s\in\mathcal{S}} t_s(Q)\ln p_\theta\left(\hat{t}_s|I\right)+(1-t_s(Q))\left(1-\ln p_\theta\left(\hat{t}_s|I\right)\right)$$

Modified deep net architecture



Results

Ablation study in ReferItGame

Approach	Image Features	Accuracy (%)
Supervised		
SCRC (2016) [13]	VGG-cls	17.93
GroundeR (2016) [43]	VGG-cls	26.93
Unsupervised		
GroundeR (2016) [43]	VGG-cls	10.70
GroundeR (2016) [43]	VGG-det	-
Entire image	None	14.62
Largest proposal	None	14.73
Mutual Info.	VGG-det	16.00
Ours	VGG-cls	**18.68**
Ours	VGG-det	**17.88**
Ours	Deeplab-seg	**16.83**
Ours	YOLO-det	**17.96**
Ours	VGG-cls + VGG-det	**20.10**
Ours	VGG-cls + YOLO-det	**20.91**

Ablation study in Flickr30k

Approach	Image Features	Accuracy (%)
Supervised		
CCA (2015) [40]	VGG-cls	27.42
SCRC (2016) [13]	VGG-cls	27.80
CCA (2016) [41]	VGG-det	43.84
GroundeR (2016) [43]	VGG-det	47.81
Unsupervised		
GroundeR (2016) [43]	VGG-cls	24.66
GroundeR (2016) [43]	VGG-det	28.94
Entire image	None	21.99
Largest proposal	None	24.34
Mutual Info.	VGG-det	31.19
Ours	VGG-cls	22.31
Ours	VGG-det	**35.90**
Ours	Deeplab-seg	**30.72**
Ours	YOLO-det	**36.93**

Performance over types on Flickr30k

	people	clothing	body parts	animals	vehicles	instruments	scene	other
# Instances	5,656	2,306	523	518	400	162	1,619	3,374
Entire Image	27.83	5.24	0.76	17.56	25.50	15.43	**45.77**	16.56
Largest proposal	31.80	7.58	2.10	30.11	34.50	17.28	41.21	17.21
GroundeR, VGG-det (2016)	44.32	9.02	0.96	46.91	46.00	19.14	28.23	16.98
Ours, VGG-det	**61.93**	**16.86**	**2.48**	64.28	54.0	9.87	16.66	14.25
Ours, YOLO-det	58.37	14.87	2.29	**68.91**	**55.00**	**22.22**	24.87	**20.77**

Table 3. Unsupervised phrase localization performance over types on Flickr 30k Entities (accuracy in %).

Top five relevant concepts trained on Flickr30k

Word	Most relevant concepts
man	person, surfboard, toilet, hot dog, tie
woman	person, broccoli, handbag, scissors, knife
boy	skateboard, bed, person, sports ball, toothbrush
girl	bed, person, fork, sofa, toothbrush
police	bus, motorbike, horse, traffic light, car
chef	knife, spoon, bowl, apple, sink

Table 4. The top five relevant concepts for each word based on $E(s, c)$ trained on Flickr30k using YOLO-det image features.

Qualitative results

Successful results

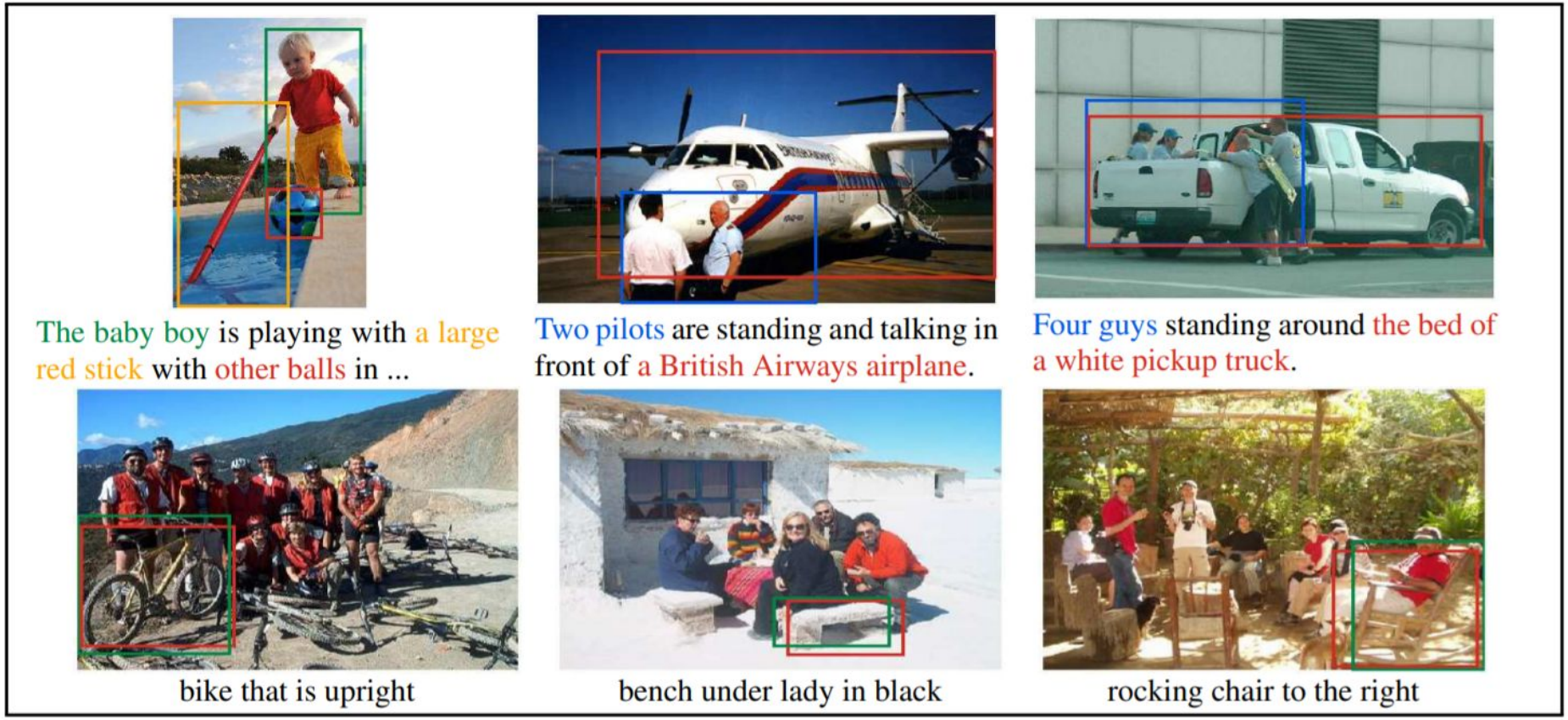
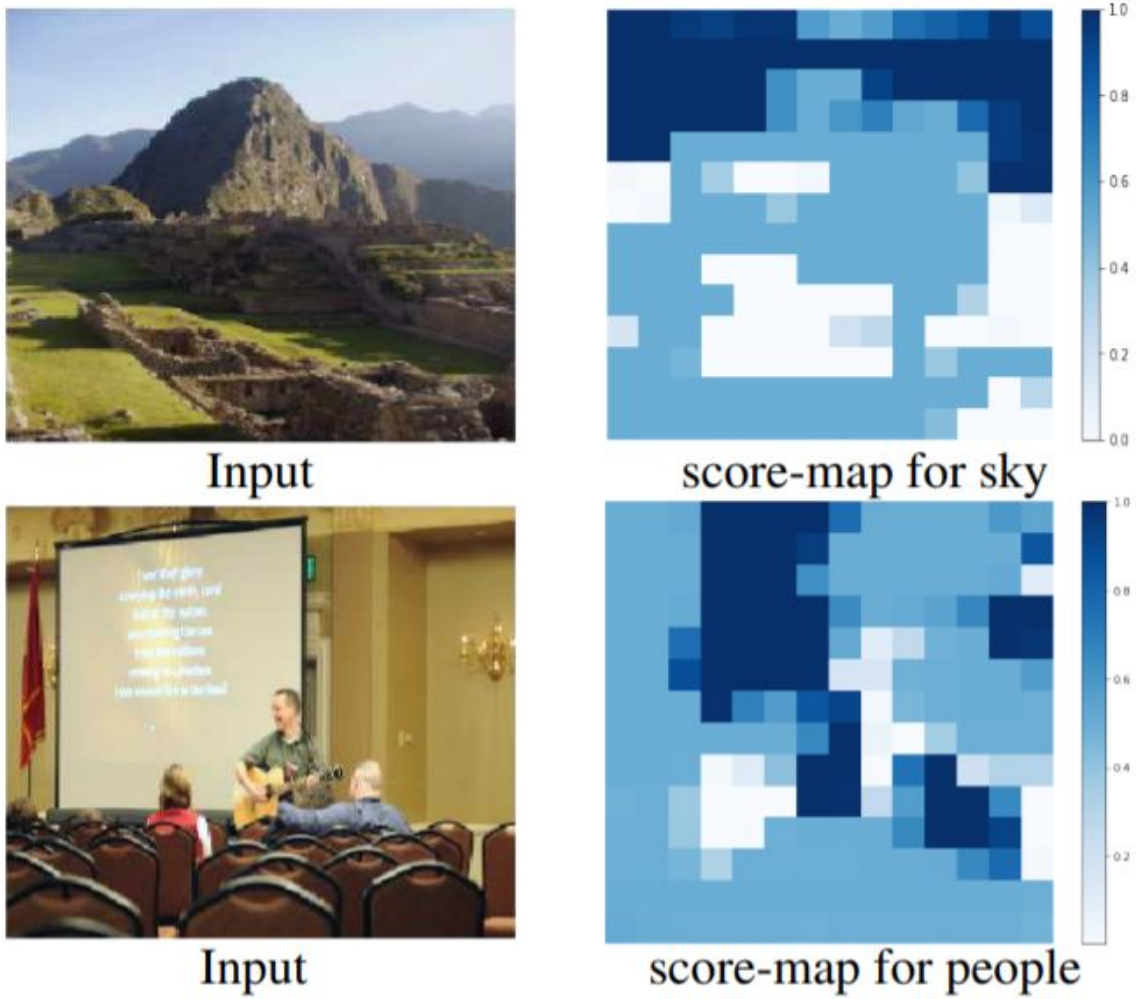


Figure 4. Results from our approach on the test set for grounding of textual phrases. **Top Row:** Flickr 30k Entities Dataset. **Bottom Row:** ReferItGame Dataset (Groundtruth box in green and predicted box in red).

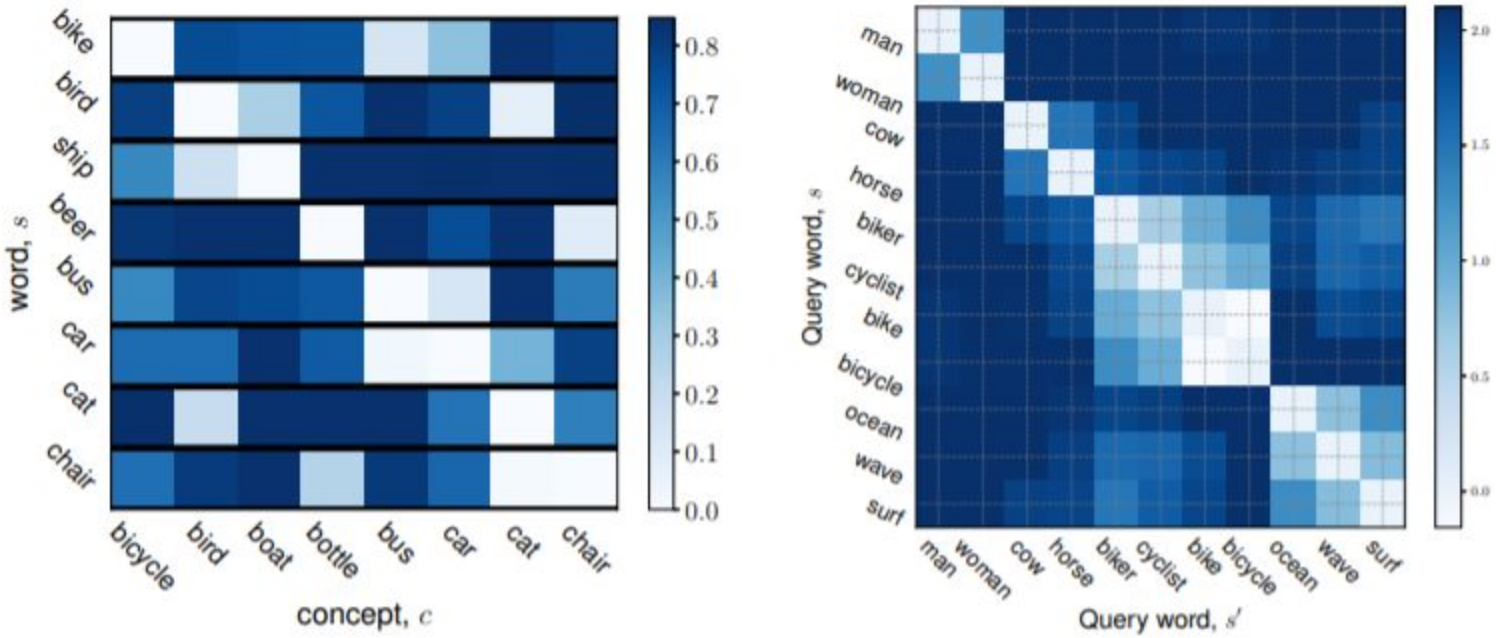
Failure cases

Figure 5. Flickr30k Failure Cases, (Green box: ground-truth, Red box: predicted).

Success and failure cases for learned image concepts



E(s, c) subset



(a) (b)

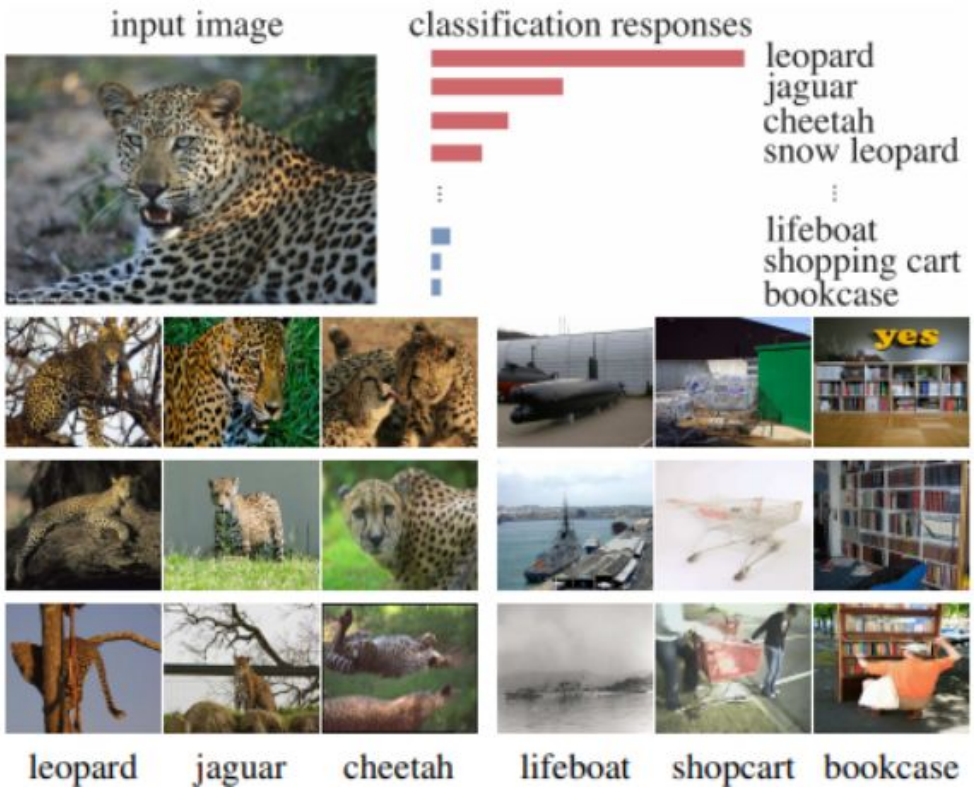
Figure 7. (a) Trained $E(s, c)$ visualized on word, s, and detection concepts c, (b) Euclidean distance visualized between the vector $E(s, :)$ and $E(s', :)$. Both visualizations are trained on Flickr30k.

Unsupervised Feature Learning via Non-Parametric Instance Discrimination

Zhirong Wu[⋆†] Yuanjun Xiong[†‡] Stella X. Yu[⋆] Dahua Lin[†]
[⋆]UC Berkeley / ICSI [†]Chinese University of Hong Kong [‡]Amazon Rekognition

Motivation

“It is not the semantic labeling, but the apparent similarity in the data themselves that brings some classes closer than others.”



Instance Discrimination

- Instace-wise supervision
- Non-Parametric Approach
- Metric Learning Problem

Learn an **embedding function** without supervision. This embedding **induces a metric** over the image space that should map visually similar images closer to each other.

$$\mathbf{v} = f_{\boldsymbol{\theta}}(x)$$

$$d_{\boldsymbol{\theta}}(x, y) = \|f_{\boldsymbol{\theta}}(x) - f_{\boldsymbol{\theta}}(y)\|$$

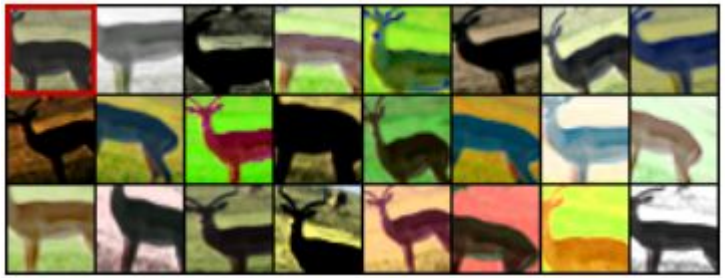
Related Work

- Metric Learning

Derived from feature learning, automatically learn a distance metric over the representation.

- Exemplar CNN

Automatically generated surrogate task and taking data augmentation to the extreme.



https://ai.stanford.edu/~ang/papers/nips02-metric.pdf

https://papers.nips.cc/paper/5548-discriminative-unsupervised-feature-learning-with-convolutional-neural-networks.pdf

Approach

Non-Parametric Softmax Classifier

$$P(i|\mathbf{v}) = \frac{\exp\left(\mathbf{w}_i^T \mathbf{v}\right)}{\sum_{j=1}^{n} \exp\left(\mathbf{w}_j^T \mathbf{v}\right)}$$

$$\downarrow$$

$$\mathbf{w}_j^T \mathbf{v} \text{ with } \mathbf{v}_j^T \mathbf{v}$$

$$\downarrow$$

$$\boxed{P(i|\mathbf{v}) = \frac{\exp\left(\mathbf{v}_i^T \mathbf{v} / \tau\right)}{\sum_{j=1}^{n} \exp\left(\mathbf{v}_j^T \mathbf{v} / \tau\right)}}$$

Temperature

Learning Objective

Minimize

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \log P(i|f_{\boldsymbol{\theta}}(x_i))$$

Approach

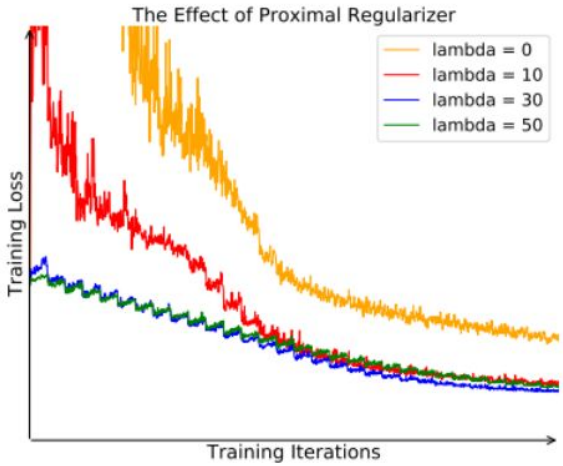
Noise-Contrastive Estimation (NCE)

Reduce computational complexity from O(n) to O(1) per sample.

Convert multi-class classification into a set of binary classification problems that discriminate between data samples and noise samples.

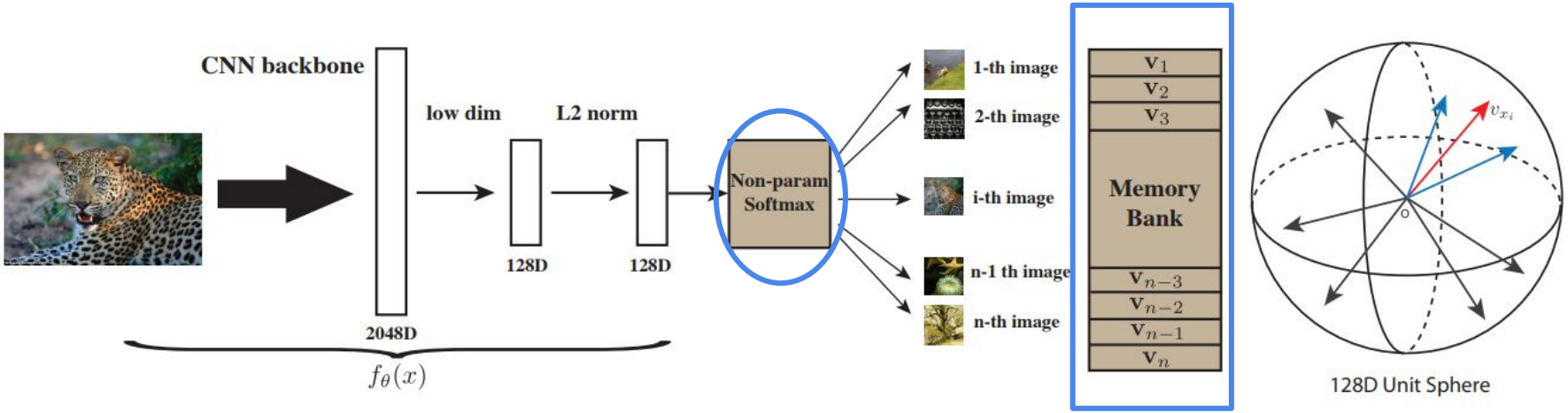
Proximal Regularization

Reduce random sampling fluctuation by introducing an additional term to encourage smoothness of training dynamics.



http://proceedings.mlr.press/v9/gutmann10a/gutmann10a.pdf

Model


CNN backbone
low dim
L2 norm
128D
128D
2048D
$f_\theta(x)$
Non-param Softmax
1-th image
2-th image
i-th image
n-1 th image
n-th image
$\mathbf{v}_1$
$\mathbf{v}_2$
$\mathbf{v}_3$
Memory Bank
$\mathbf{v}_{n-3}$
$\mathbf{v}_{n-2}$
$\mathbf{v}_{n-1}$
$\mathbf{v}_n$
v_{x_i}
128D Unit Sphere

Results

Parametric vs Non-parametric Softmax

Training / Testing	Linear SVM	Nearest Neighbor
Param Softmax	60.3	63.0
Non-Param Softmax	75.4	**80.8**
NCE $m = 1$	44.3	42.5
NCE $m = 10$	60.2	63.4
NCE $m = 512$	64.3	78.4
NCE $m = 4096$	70.2	**80.4**

Table 1: Top-1 accuracies on CIFAR10, by applying linear SVM or kNN classifiers on the learned features. Our non-parametric softmax outperforms parametric softmax, and NCE provides close approximation as m increases.

Image Classification Accuracy on ImageNet							
method	conv1	conv2	conv3	conv4	conv5	kNN	#dim
Random	11.6	17.1	16.9	16.3	14.1	3.5	10K
Data-Init [16]	17.5	23.0	24.5	23.2	20.6	-	10K
Context [2]	16.2	23.3	30.2	31.7	29.6	-	10K
Adversarial [4]	17.7	24.5	31.0	29.9	28.0	-	10K
Color [47]	13.1	24.8	31.0	32.6	31.8	-	10K
Jigsaw [27]	19.2	30.1	34.7	33.9	28.3	-	10K
Count [28]	18.0	30.6	34.3	32.5	25.7	-	10K
SplitBrain [48]	17.7	29.3	35.4	35.2	32.8	11.8	10K
Exemplar[3]			31.5			-	4.5K
Ours Alexnet	16.8	26.5	31.8	34.1	**35.6**	31.3	128
Ours VGG16	16.5	21.4	27.6	33.1	**37.2**	33.9	128
Ours Resnet18	16.0	19.9	26.3	35.7	**42.1**	**40.5**	128
Ours Resnet50	15.3	18.8	24.4	35.3	**43.9**	**42.5**	128

Table 2: Top-1 classification accuracies on ImageNet.

Results

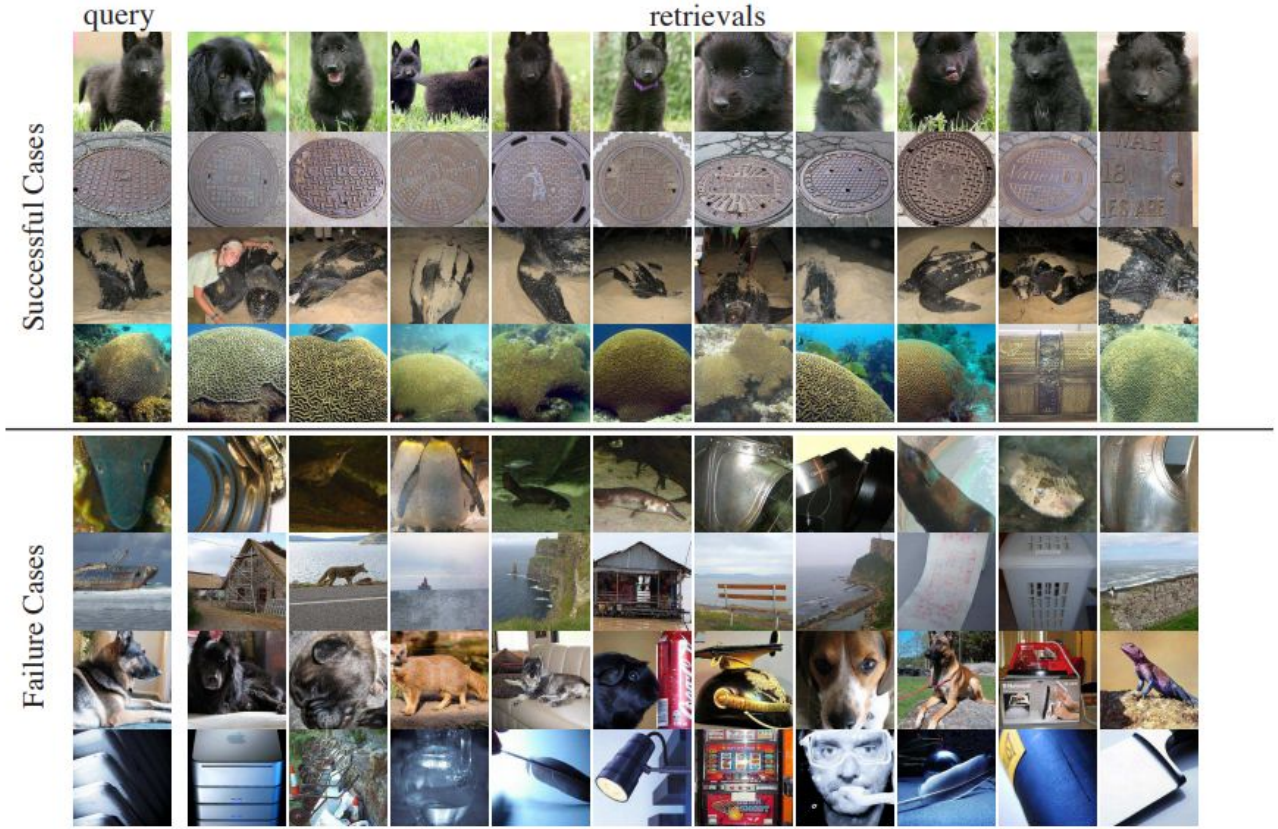


Figure 5: Retrieval results for example queries. The left column are queries from the validation set, while the right columns show the 10 closest instances from the training set. The upper half shows the best cases. The lower half shows the worst cases.

Results- Other Tasks

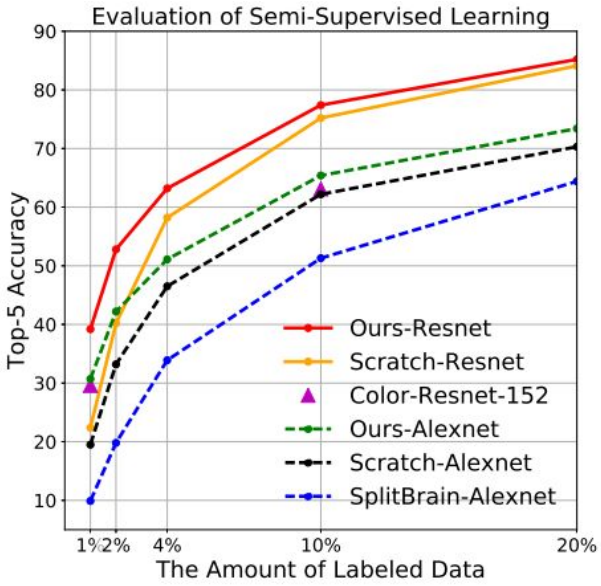


Figure 6: Semi-supervised learning results on ImageNet with an increasing fraction of labeled data (x axis). Ours are consistently and significantly better. Note that the results for colorization-based pretraining are from a deeper ResNet-152 network [19].

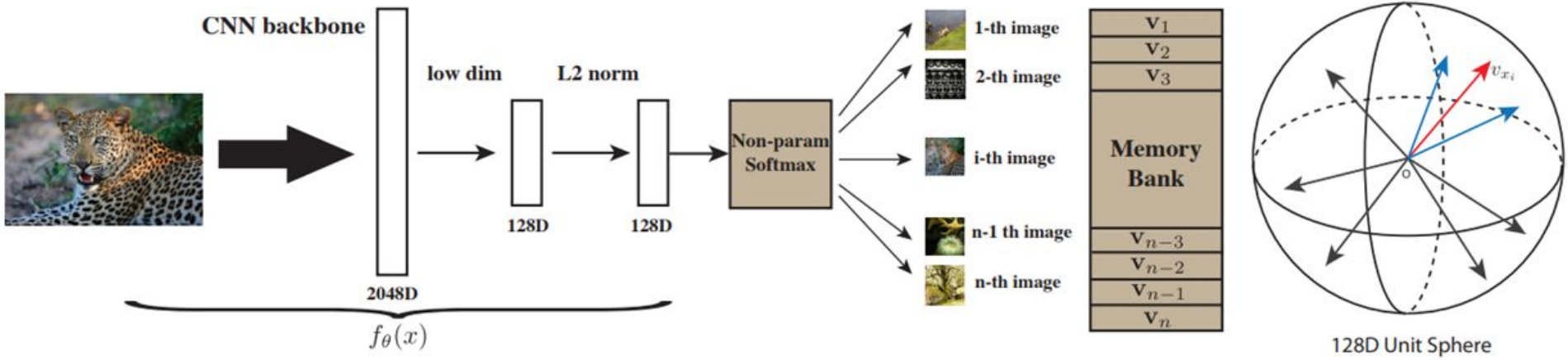
Method	mAP
AlexNet Labels†	56.8
Gaussian	43.4
Data-Init [16]	45.6
Context [2]	**51.1**
Adversarial [4]	46.9
Color [47]	46.9
Video [44]	47.4
Ours Alexnet	48.1

Method	mAP
VGG Labels†	67.3
Gaussian	39.7
Video [44]	60.2
Context [2]	61.5
Transitivity [45]	**63.2**
Ours VGG	60.5
ResNet Labels†	76.2
Ours ResNet	**65.4**

Table 6: Object detection performance on PASCAL VOC 2007 test, in terms of mean average precision (mAP), for supervised pretraining methods (marked by †), existing unsupervised methods, and our method.

Tutorial

Model


CNN backbone
low dim
L2 norm
128D
128D
2048D
$f_\theta(x)$
Non-param Softmax
1-th image
2-th image
i-th image
n-1 th image
n-th image
$\mathbf{v}_1$
$\mathbf{v}_2$
$\mathbf{v}_3$
Memory Bank
$\mathbf{v}_{n-3}$
$\mathbf{v}_{n-2}$
$\mathbf{v}_{n-1}$
$\mathbf{v}_n$
v_{x_i}
o
128D Unit Sphere

Hyperparameters

- Learning rate
- Feature vector dimension
- Number of noise samples for NCE (k)
- Temperature parameter for softmax (t)
- Momentum for non-parametric updates (m)
- Number of total epochs to run
- Training batch size
- Test batch size

```
lr=0.03
low-dim=128
nce-k=4096
nce-t=0.1
nce-m=0.5
epochs=20
batch_size=20
batch_size_test=100
```

Data Preparation

```
if __name__ == '__main__':

    # Data
    print('==> Preparing data..')
    transform_train = transforms.Compose([
        transforms.RandomResizedCrop(size=32, scale=(0.2,1.)),
        transforms.ColorJitter(0.4, 0.4, 0.4, 0.4),
        transforms.RandomGrayscale(p=0.2),
        #transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
    ])

    transform_test = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
    ])
```

```
classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
ndata = trainset.__len__()

def ResNet18(low_dim=128):
    return ResNet(BasicBlock, [2,2,2,2], low_dim)

if args.nce_k > 0:
    lemniscate = NCEAverage(args.low_dim, ndata, args.nce_k, args.nce_t, args.nce_m)
else:
    lemniscate = LinearAverage(args.low_dim, ndata, args.nce_t, args.nce_m)
```

```python
class ResNet(nn.Module):

    def __init__(self, block, layers, low_dim=128):
        self.inplanes = 64
        super(ResNet, self).__init__()
        self.conv1 = nn.Conv2d(3, 64, kernel_size=7, stride=2, padding=3,
                               bias=False)
        self.bn1 = nn.BatchNorm2d(64)
        self.relu = nn.ReLU(inplace=True)
        self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2, padding=1)
        self.layer1 = self._make_layer(block, 64, layers[0])
        self.layer2 = self._make_layer(block, 128, layers[1], stride=2)
        self.layer3 = self._make_layer(block, 256, layers[2], stride=2)
        self.layer4 = self._make_layer(block, 512, layers[3], stride=2)
        self.avgpool = nn.AvgPool2d(7, stride=1)
        self.fc = nn.Linear(512 * block.expansion, low_dim)
        self.l2norm = Normalize(2)

        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                n = m.kernel_size[0] * m.kernel_size[1] * m.out_channels
                m.weight.data.normal_(0, math.sqrt(2. / n))
            elif isinstance(m, nn.BatchNorm2d):
                m.weight.data.fill_(1)
                m.bias.data.zero_()
```

```python
def _make_layer(self, block, planes, blocks, stride=1):
    downsample = None
    if stride != 1 or self.inplanes != planes * block.expansion:
        downsample = nn.Sequential(
            nn.Conv2d(self.inplanes, planes * block.expansion,
                      kernel_size=1, stride=stride, bias=False),
            nn.BatchNorm2d(planes * block.expansion),
        )

    layers = []
    layers.append(block(self.inplanes, planes, stride, downsample))
    self.inplanes = planes * block.expansion
    for i in range(1, blocks):
        layers.append(block(self.inplanes, planes))

    return nn.Sequential(*layers)
```

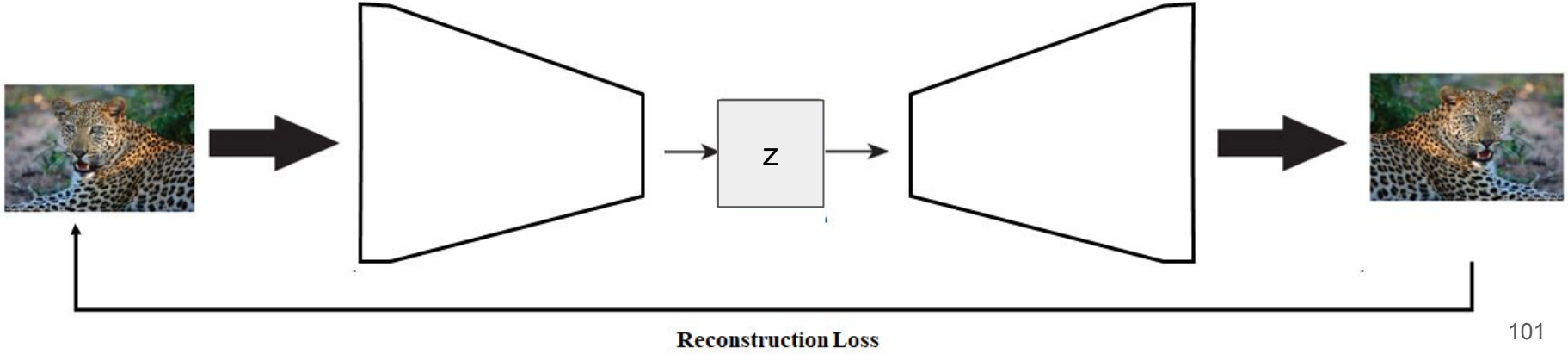
```python
def forward(self, x):
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.maxpool(x)

    x = self.layer1(x)
    x = self.layer2(x)
    x = self.layer3(x)
    x = self.layer4(x)

    x = self.avgpool(x)
    x = x.view(x.size(0), -1)
    x = self.fc(x)
    x = self.l2norm(x)

    return x
```

```python
class BasicBlock(nn.Module):
    expansion = 1

    def __init__(self, inplanes, planes, stride=1, downsample=None):
        super(BasicBlock, self).__init__()
        self.conv1 = conv3x3(inplanes, planes, stride)
        self.bn1 = nn.BatchNorm2d(planes)
        self.relu = nn.ReLU(inplace=True)
        self.conv2 = conv3x3(planes, planes)
        self.bn2 = nn.BatchNorm2d(planes)
        self.downsample = downsample
        self.stride = stride

    def forward(self, x):
        residual = x

        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)

        out = self.conv2(out)
        out = self.bn2(out)

        if self.downsample is not None:
            residual = self.downsample(x)

        out += residual
        out = self.relu(out)

        return out
```

```python
class BasicBlock(nn.Module):
    expansion = 1

    def __init__(self, in_planes, planes, stride=1):
        super(BasicBlock, self).__init__()
        self.conv1 = nn.Conv2d(in_planes, planes, kernel_size=3, stride=stride, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(planes)
        self.conv2 = nn.Conv2d(planes, planes, kernel_size=3, stride=1, padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(planes)

        self.shortcut = nn.Sequential()
        if stride != 1 or in_planes != self.expansion*planes:
            self.shortcut = nn.Sequential(
                nn.Conv2d(in_planes, self.expansion*planes, kernel_size=1, stride=stride, bias=False),
                nn.BatchNorm2d(self.expansion*planes)
            )

    def forward(self, x):
        out = F.relu(self.bn1(self.conv1(x)))
        out = self.bn2(self.conv2(out))
        out += self.shortcut(x)
        out = F.relu(out)
        return out
```

```python
# define loss function
if hasattr(lemniscate, 'K'):
    criterion = NCECriterion(ndata)
else:
    criterion = nn.CrossEntropyLoss()

net.to(device)
lemniscate.to(device)
criterion.to(device)

if args.test_only:
    acc = kNN(0, net, lemniscate, trainloader, testloader, 200, args.nce_t, 1)
    sys.exit(0)

optimizer = optim.SGD(net.parameters(), lr=args.lr, momentum=0.9, weight_decay=5e-4)

def adjust_learning_rate(optimizer, epoch):
    """Sets the learning rate to the initial LR decayed by 10 every 30 epochs"""
    lr = args.lr
    if epoch >= 80:
        lr = args.lr * (0.1 ** ((epoch-80) // 40))
    print(lr)
    for param_group in optimizer.param_groups:
        param_group['lr'] = lr
```

```python
# Training
def train(epoch):
    print('\nEpoch: %d' % epoch)
    adjust_learning_rate(optimizer, epoch)
    train_loss = AverageMeter()
    data_time = AverageMeter()
    batch_time = AverageMeter()
    correct = 0
    total = 0

    # switch to train mode
    net.train()

    end = time.time()
    for batch_idx, (inputs, targets, indexes) in enumerate(trainloader):
        data_time.update(time.time() - end)
        inputs, targets, indexes = inputs.to(device), targets.to(device), indexes.to(device)
        optimizer.zero_grad()

        features = net(inputs)
        outputs = lemniscate(features, indexes)
        loss = criterion(outputs, indexes)

        loss.backward()
        optimizer.step()

        train_loss.update(loss.item(), inputs.size(0))

        # measure elapsed time
        batch_time.update(time.time() - end)
        end = time.time()

        print('Epoch: [{}][{}/{}]'
            'Time: {batch_time.val:.3f} ({batch_time.avg:.3f}) '
            'Data: {data_time.val:.3f} ({data_time.avg:.3f}) '
            'Loss: {train_loss.val:.4f} ({train_loss.avg:.4f})'.format(
            epoch, batch_idx, len(trainloader), batch_time=batch_time, data_time=data_time, train_loss=train_loss))
```

Homework

- Use the MNIST dataset
- Follow the instance-wise approach of the tutorial but adapt the code to create an autoencoder:
  - You may modify the encoder architecture and need to program the decoding layers.
  - Modify the loss function, visualize the results and discuss.
- (**Bonus**) Change the latent space representation and turn it into a Variational Autoencoder.