Large Scale Deep Learning and Self-Supervision in Vision and NLP

Thomas Breuel

NVIDIA

INTRODUCTION

Central Question

What can you do with large amounts of unlabeled training data?

Different Kinds of Learning

- supervised learning
  - inputs and outputs are given
- unsupervised learning
  - only inputs are given
- semi-supervised learning
  - combine supervised and unsupervised data
- self-supervised learning
  - algorithm derives a supervised problem from unsupervised data

Different Kinds of Learning

- transfer learning
  - use a model trained on one problem to solve another problem
- active learning
  - the learning algorithm requests transcribed data from an oracle
- metric learning
  - learn a distance measure that helps with clustering / classification
- representation learning
  - transform input vectors into another space that, removes noise, expresses invariances, places "similar" samples closer together

Different Kinds of Models

- discriminative vs generative
- linear models
- non-linear models
  - parametric probabilistic
  - support vector machines
  - deep learning
    - convolutional
    - recurrent
    - transformer

Overview

Motivation:

- OCR example

Classical Techniques:

- statistical basis of machine learning (mostly review)
- linear methods (mostly review?)

Deep Learning:

- language modeling and sequence learning
- self-supervised learning for images
- generative modeling (VAE, flows, GANs)

Background

I'm assuming you have

- done some supervised deep learning, e.g., trained MNIST, ImageNet, etc.
- understand the basics of SGD, loss functions, etc.

Focus

We will focus on the unsupervised / self-supervised aspects of methods and papers we will be discussing, discussing other aspects of papers only as needed.

OPTICAL CHARACTER RECOGNITION

Motivating Example: OCR

Optical Character Recognition:

- image → text
- closely related to object recognition, autonomous driving, speech recognition, ...
- much better statistical understanding of...
  - class structure
  - noise
  - ground truth
  - syntactic structure / language modeling

Motivating Example: OCR

NEW MEMBERS Qualified

OBITUARIES

Philip G. Johnson

Lorraine J. D. Frazer

Daniel C. Green

Archie Frank Fanger

DO YOU USE WIRE CLOTH STRAINERS?

Michigan Wire Cloth Company can assist you in selecting the correct grade of wire cloth for any type of strainer, and can help in the development of proper strainer design.

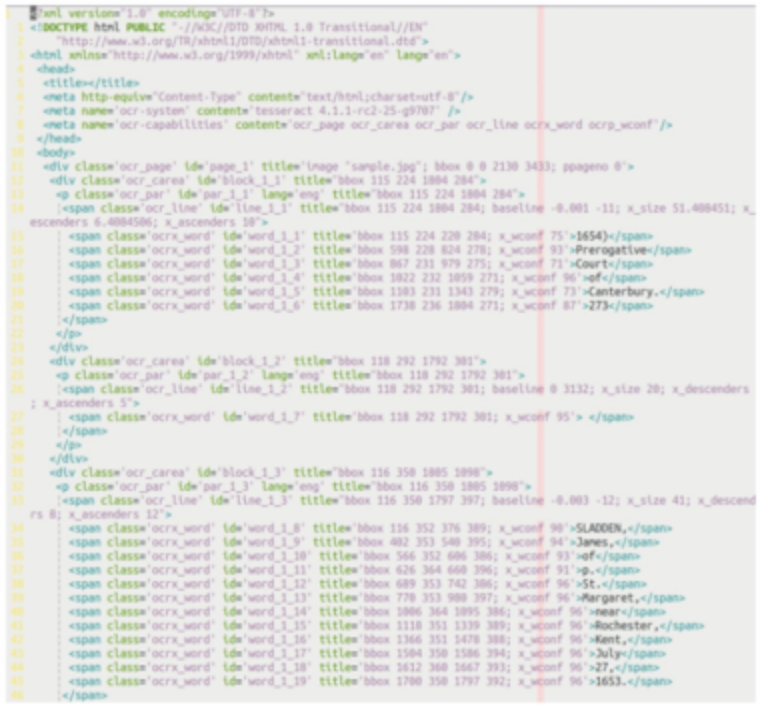
Complete manufacturing facilities for drawing the wire - weaving the cloth - and fabricating the assembly all in one continuous production line.

Send Michigan Your Inquiries

MICHIGAN WIRE CLOTH CO.

October, 1944

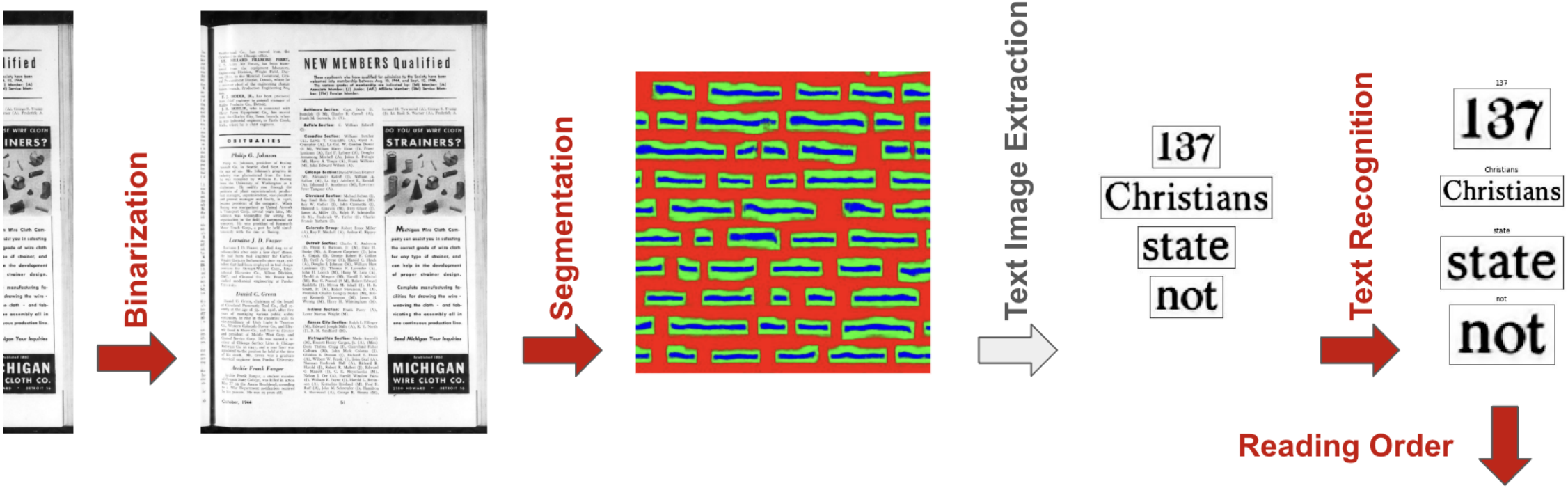
51



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
  <title></title>
  <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  <meta name='ocr-system' content='tesseract 4.1.1-rc2-25-g9707' />
  <meta name='ocr-capabilities' content='ocr_page ocr_carea ocr_par ocr_line ocrx_word ocrp_wconf' />
 </head>
 <body>
  <div class='ocr_page' id='page_1' title='image "sample.jpg"; bbox 0 0 2130 3433; ppageno 0'>
   <div class='ocr_carea' id='block_1_1' title='bbox 115 224 1804 284'>
    <p class='ocr_par' id='par_1_1' lang='eng' title='bbox 115 224 1804 284'>
     <span class='ocr_line' id='line_1_1' title='bbox 115 224 1804 284; baseline -0.001 -11; x_size 51.408451; x_descenders 6.4084506; x_ascenders 10'>
      <span class='ocrx_word' id='word_1_1' title='bbox 115 224 220 284; x_wconf 75'>1654)</span>
      <span class='ocrx_word' id='word_1_2' title='bbox 598 228 824 278; x_wconf 93'>Prerogative</span>
      <span class='ocrx_word' id='word_1_3' title='bbox 867 231 979 275; x_wconf 71'>Court</span>
      <span class='ocrx_word' id='word_1_4' title='bbox 1022 232 1059 271; x_wconf 96'>of</span>
      <span class='ocrx_word' id='word_1_5' title='bbox 1103 231 1343 279; x_wconf 73'>Canterbury.</span>
      <span class='ocrx_word' id='word_1_6' title='bbox 1738 236 1804 271; x_wconf 87'>273</span>
     </span>
    </p>
   </div>
   <div class='ocr_carea' id='block_1_2' title='bbox 118 292 1792 301'>
    <p class='ocr_par' id='par_1_2' lang='eng' title='bbox 118 292 1792 301'>
     <span class='ocr_line' id='line_1_2' title='bbox 118 292 1792 301; baseline 0 3132; x_size 20; x_descenders 5; x_ascenders 5'>
      <span class='ocrx_word' id='word_1_7' title='bbox 118 292 1792 301; x_wconf 95'> </span>
     </span>
    </p>
   </div>
   <div class='ocr_carea' id='block_1_3' title='bbox 116 350 1805 1098'>
    <p class='ocr_par' id='par_1_3' lang='eng' title='bbox 116 350 1805 1098'>
     <span class='ocr_line' id='line_1_3' title='bbox 116 350 1797 397; baseline -0.003 -12; x_size 41; x_descenders 8; x_ascenders 12'>
      <span class='ocrx_word' id='word_1_8' title='bbox 116 352 376 389; x_wconf 90'>SLADDEN,</span>
      <span class='ocrx_word' id='word_1_9' title='bbox 402 353 540 395; x_wconf 94'>James,</span>
      <span class='ocrx_word' id='word_1_10' title='bbox 566 352 606 386; x_wconf 93'>of</span>
      <span class='ocrx_word' id='word_1_11' title='bbox 626 364 668 396; x_wconf 91'>p.</span>
      <span class='ocrx_word' id='word_1_12' title='bbox 689 353 742 386; x_wconf 96'>St.</span>
      <span class='ocrx_word' id='word_1_13' title='bbox 770 353 980 397; x_wconf 96'>Margaret,</span>
      <span class='ocrx_word' id='word_1_14' title='bbox 1006 364 1095 386; x_wconf 96'>near</span>
      <span class='ocrx_word' id='word_1_15' title='bbox 1116 351 1339 389; x_wconf 96'>Rochester,</span>
      <span class='ocrx_word' id='word_1_16' title='bbox 1366 351 1478 388; x_wconf 96'>Kent,</span>
      <span class='ocrx_word' id='word_1_17' title='bbox 1504 350 1586 394; x_wconf 96'>July</span>
      <span class='ocrx_word' id='word_1_18' title='bbox 1612 360 1667 393; x_wconf 96'>27,</span>
      <span class='ocrx_word' id='word_1_19' title='bbox 1700 350 1797 392; x_wconf 96'>1653.</span>
     </span>
```

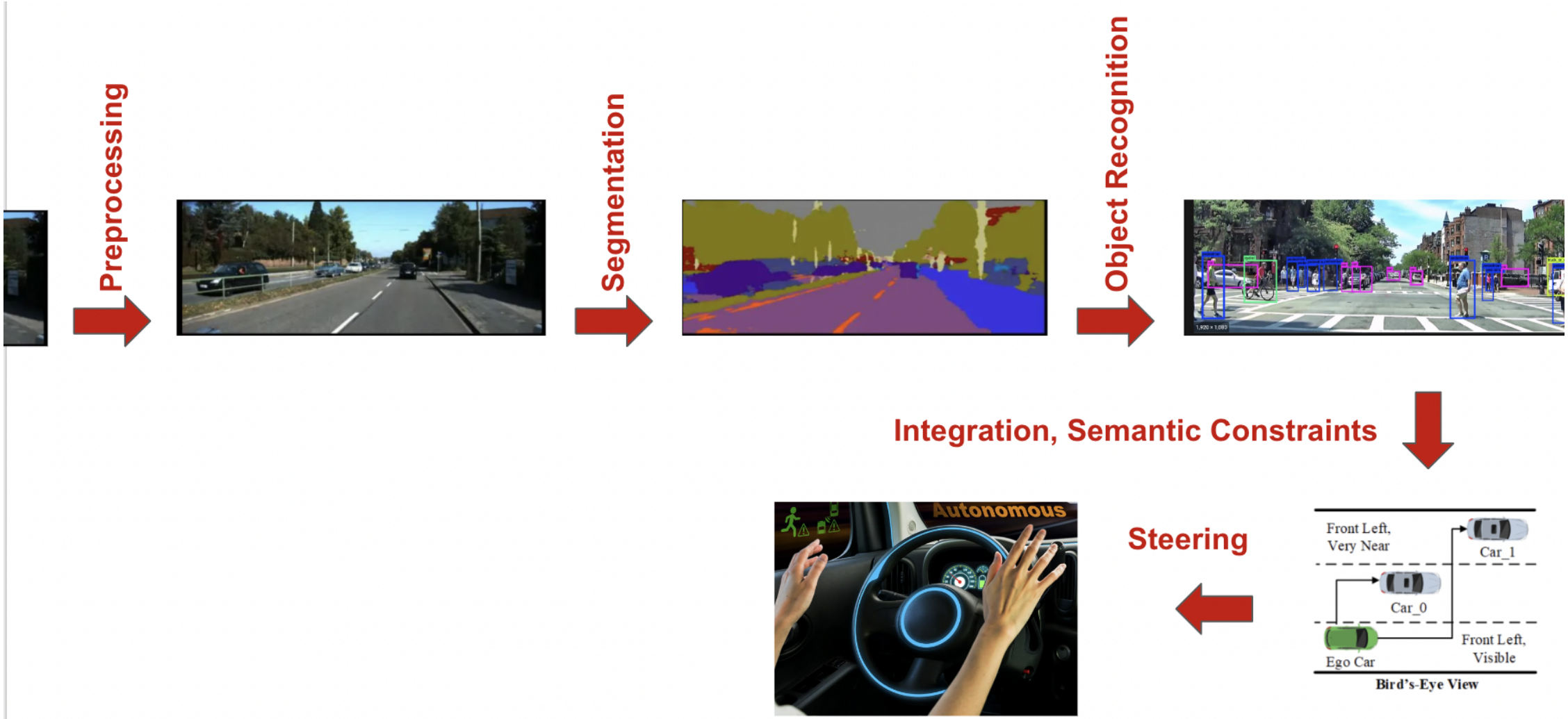
image $\rightarrow$ text, markup

Motivating Example: OCR



"1500s, when an unknown printer took a galley of type and scrambled it to make a type…"

OCR vs AV Pipeline



Motivating Example: OCR

We need to train multiple models:

- preprocessing
- segmentation
- recognition
- syntactic model (language model)
- geometric model (reading order model)

This is common to many computer vision tasks using deep learning.

OCR and Semi-Supervised Learning

Available data:

- thousands of scanned pages with manual segmentation and text
- tens of thousands of scanned pages with approximate text
- millions of scanned pages without text
- large amounts of text without document images
- the ability to generate new texts and printed documents

Typical unsupervised / semi-supervised learning problem.

First Step: Supervised Training

Train supervised models on manually labeled / transcribed data:

- segmentation model
- image $\rightarrow$ text model

Yields good performance *on data similar to training data*.

Unlabeled training data primarily helps with generalization to new datasets.

Using Millions of Untranscribed Pages

Idea:

- Use our or other OCR systems to transcribe those pages and use the output as training data ("pseudolabel")

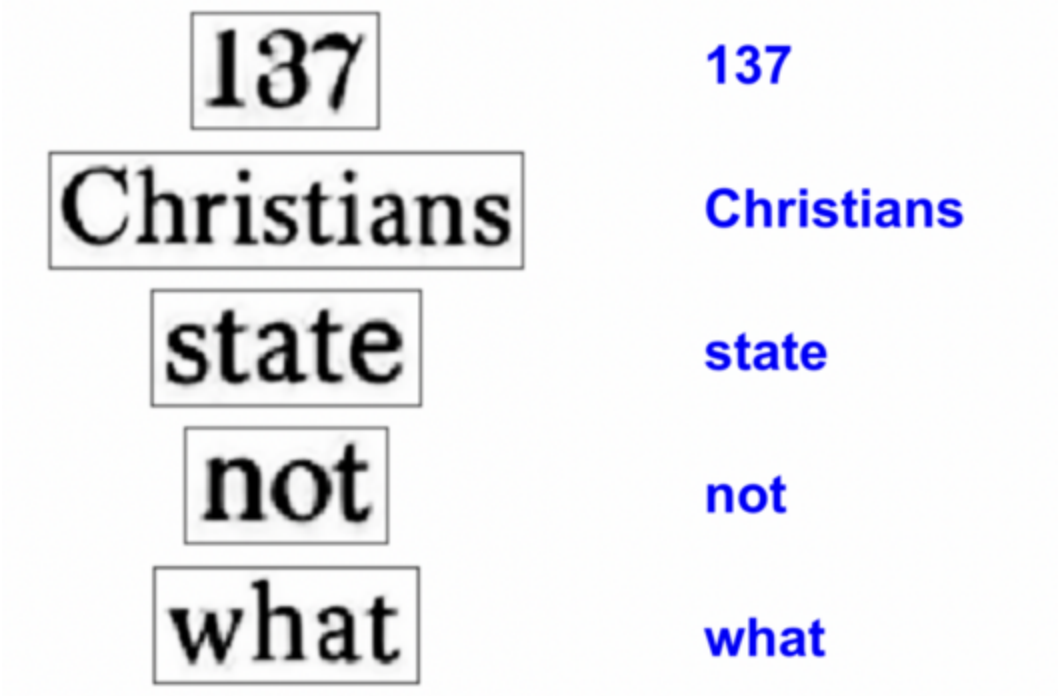
Questions:

- How can using a worse OCR system work for training a better OCR system?
- Is there anything we can do to improve this?

SEMI-SUPERVISED TEXT RECOGNITION

Example: Word Recognition Problem

Let's focus just on recognizing words for the following examples (forget about the rest of the OCR system):



Using Millions of Untranscribed Pages

Idea:

- run the existing OCR system, giving word images and corresponding text
- construct a new training set by...
  - rejecting any word that the OCR system has a low confidence in
  - rejecting any word that is not found in the dictionary
- this way, we obtain a new training set that contains "mostly good" training samples
- iterate this multiple times

Why does this work?

Network estimates $P(c|x)$ (x: image, c: class)

For OCR, we know:

- true $P(c|x)$ is approximately 0 or 1 (no ambiguities)
- any uncertainty in classifier output is due to mislabeled training data
- e.g. if 20% of training data mislabeled: $\tilde{P}(c|x) = 0.8$ for true class c
- if we use pseudolabel $\arg\max_c \tilde{P}(c|x)$, many accidentally mislabled training labels will actually be correctly labeled
- as a result, the posterior probability will be estimated higher on the next training round and the model improves

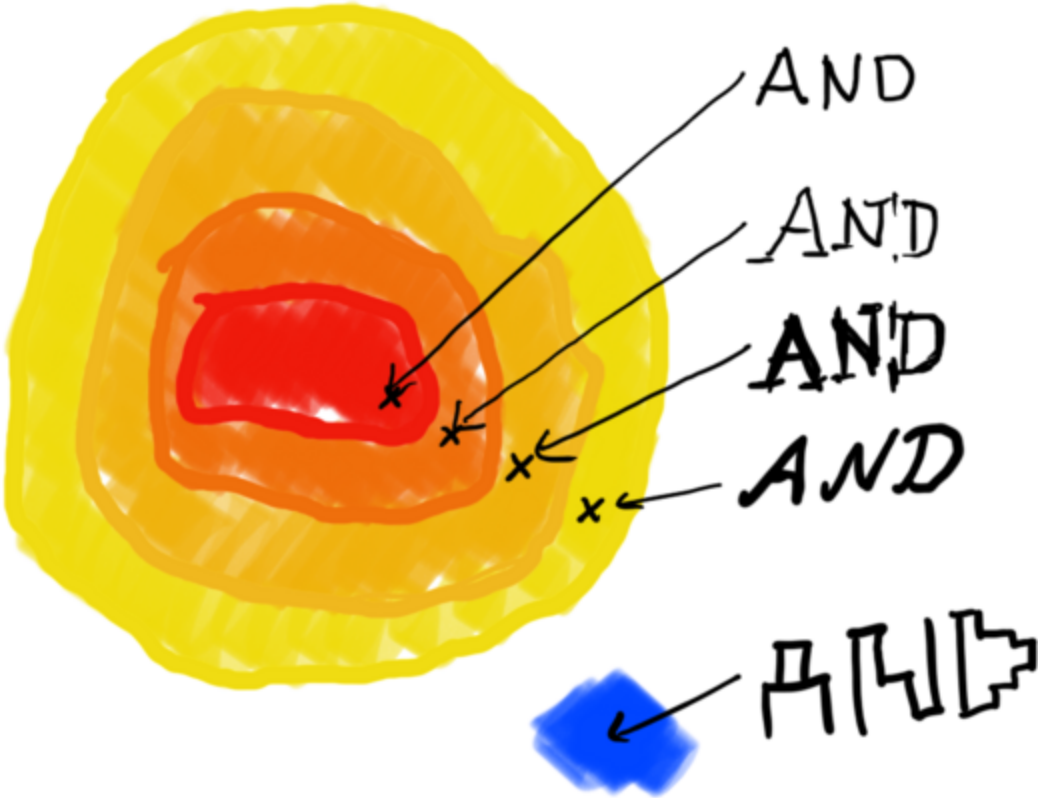
EM Algorithms

- latent variables in semi-supervised OCR
  - labels for unlabeled portion of training set
  - outlier status for unlabeled portion of training set
- EM algorithms recover latent variables by...
  - "making a best guess" given the current model
  - retraining the model as if that guess is correct

Unsupervised Learning

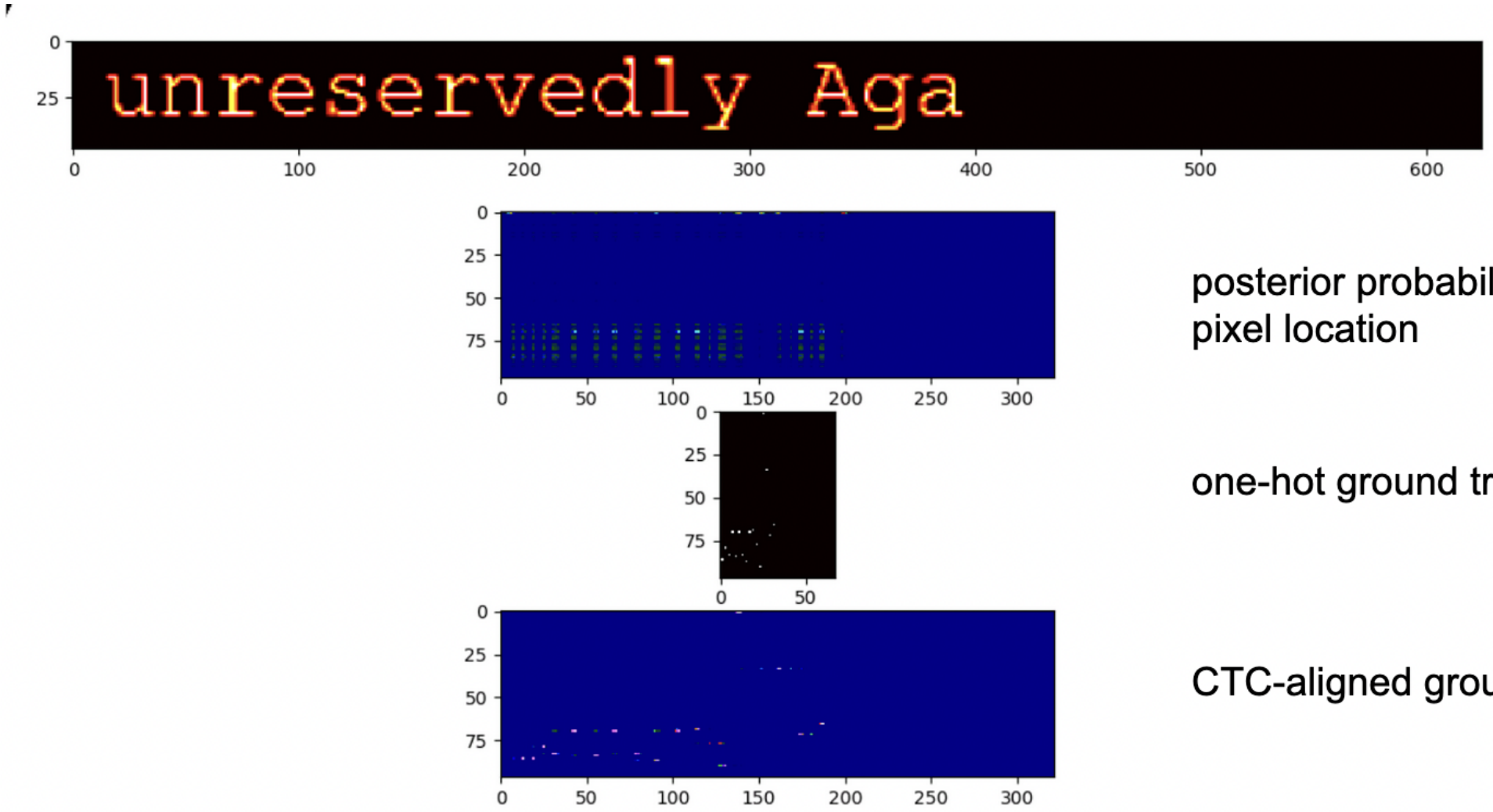
- identify the *latent variables* that are being recovered
- identify the *prior assumptions / inductive biases* in the model
- identify the *EM algorithm implementation*

Iterated Recognition / Dataset Construction



WEAKLY SUPERVISED TEXT RECOGNITION

EM Training with Language Models (aka CTC)



EM Training with LSTM/Conv + CTC

- ideally, the LSTM/Conv model outputs the correct class only where the character occurs
- outputs ϵ (no character) everywhere else
- our transcript does not contain the character positions
- CTC performs *alignment* between classifier output and transcript
- CTC estimates the most likely positions of characters given the current model

EM Training without Any Transcripts

Normal EM-Training:

- image + transcript, EM-training recovers alignment

Language Model-Based EM-Training:

- image + language model, EM-training recovers text + alignment

EM Training, Information Theory

There always has to be *some* source of information:

- unigram perplexity: 1000 ⟵ what the classifier outputs
- bigram perplexity: 200
- trigram perplexity: 100 ⟵ what the language model imposes
- full transcript perplexity: 1 ⟵ fully supervised training

Additionally: several bits per character for character location.

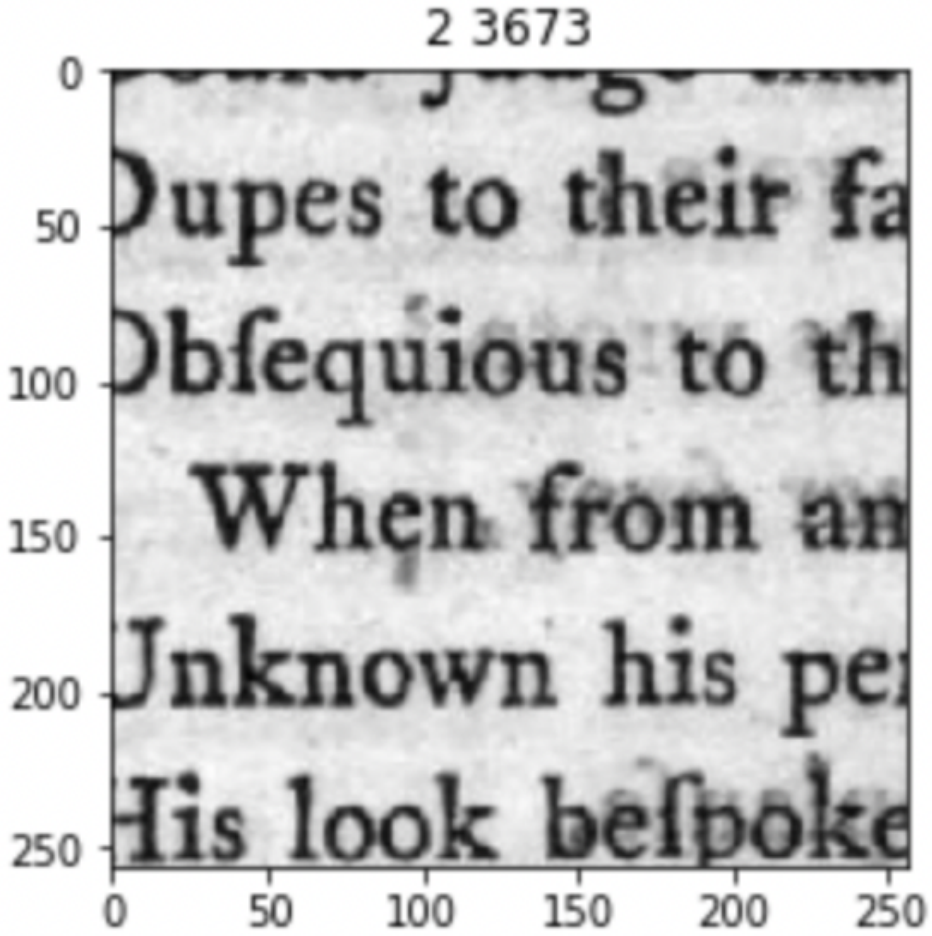
Different Forms of Weak-/Unsupervised Learning

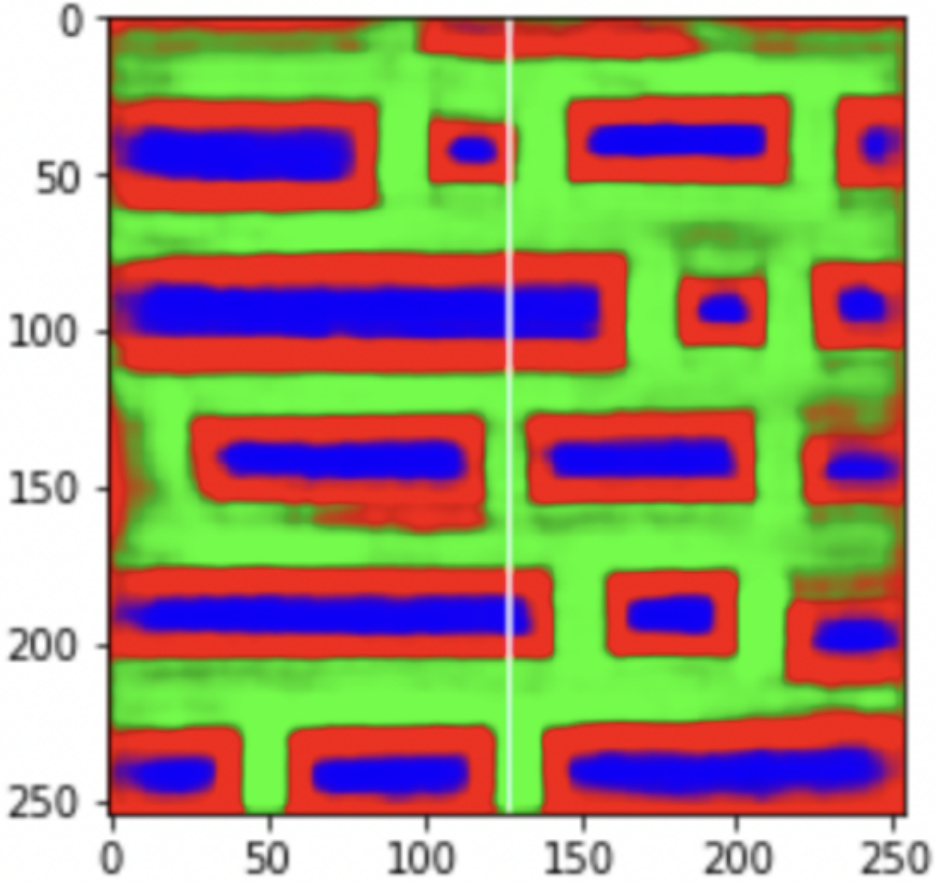
We have seen three different forms of weak/unsupervised training for text → image training:

- using image + transcript, lacking just the alignment/segmentation
- using image + language model only
- using just pseudolabels from weak classifier

SELF-SUPERVISED PAGE SEGMENTATION

Self-Supervision of Page Segmentation





Self-Supervision of Page Segmentation

Assume we start with errorful segmenter from small labeled training data.

Two kinds of errors:

- segmenter returns word for non-words
- segmenter misses actual words

Approach:

- validate each returned word using OCR (no OCR = not a word)
- mark everything that is not identified as a word as "don't know" and exclude from training

Self-Supervision

Approaches to self-supervision usually incorporate prior knowledge and require some manual design:

- text recognizer
  - reject non-words from "soft labels"
  - use language models as part of EM training
- page segmentation
  - validate segmentation via OCR
  - introduce "don't know" mask during training

Self-Supervision

Prior knowledge about the task is required to choose meaningful self-supervision tasks.

- word recognition: pseudolabels, language-model based rejection
- segmentation: verification via OCR, introduction of "don't care" regions
- object reconition, natural image segmentation: *later*

ACTIVE LEARNING

Active Learning

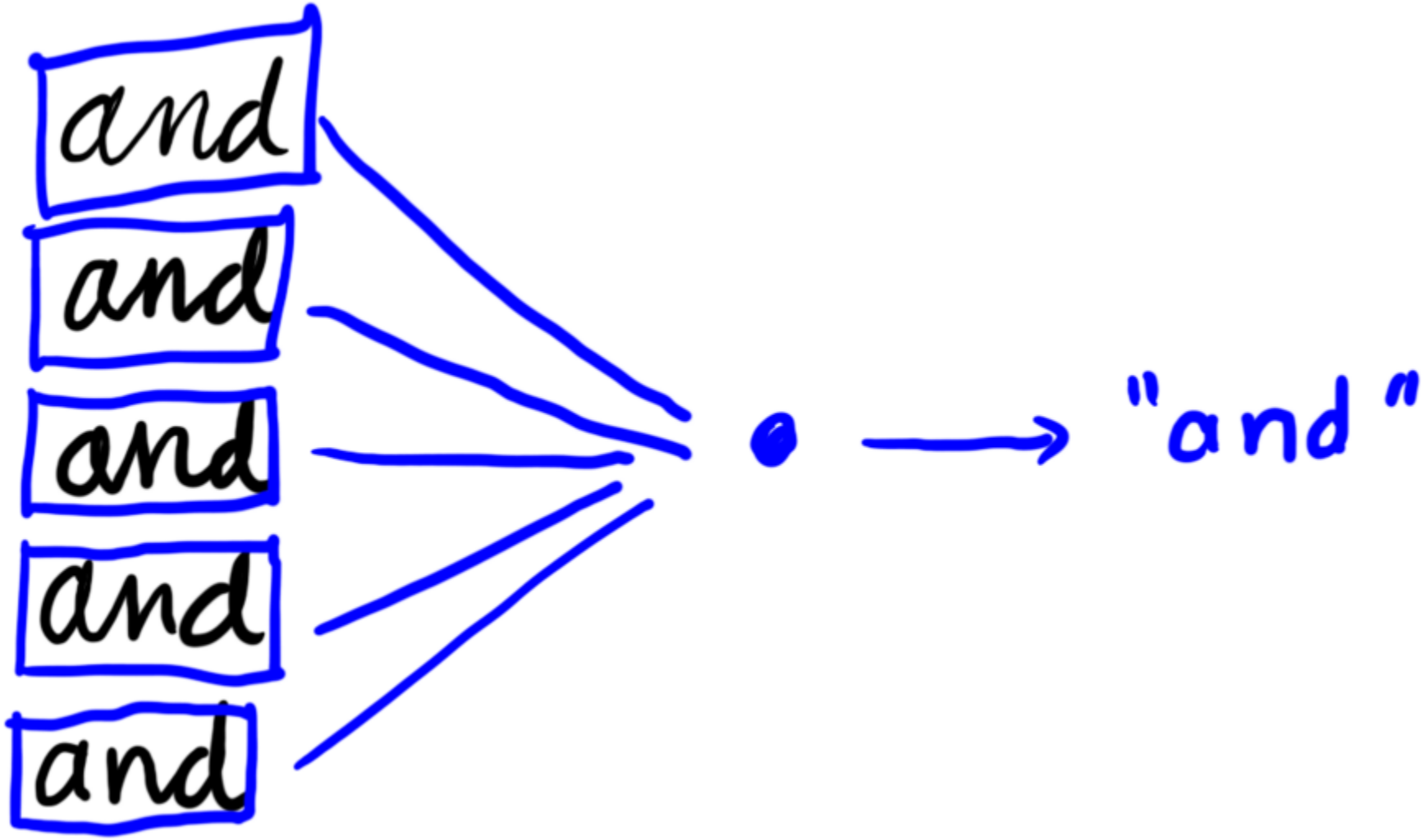
Request Help from Oracle:

- run OCR as above and send low confidence outputs to manual transcribers
  - output near decision boundaries
  - output that can maximally help the classifier improve

Clustering:

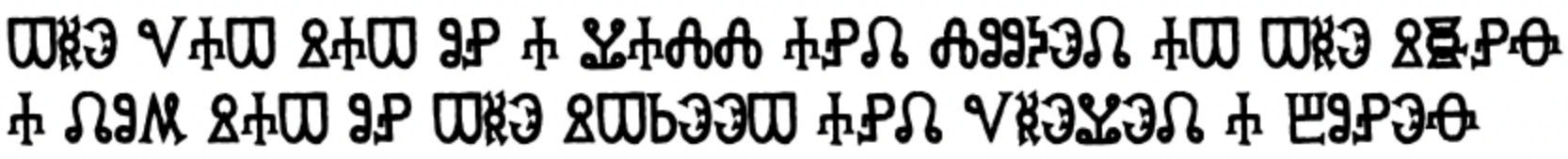
- perform clustering on character or word images and manually transcribe each cluster

Transcription with Clustering



Question: what similarity measure do we use for clustering?

OCR by Solving a Cryptogram



- cluster characters by shape
- use frequency and patterns to infer character identity
  - e.g.: first word is likely "THE"
    - letter frequencies
    - word frequencies
  - single letter word is likely "A"
- explicit form of many EM-based recognition algorithms

OCR by Solving a Cryptogram

THE CAT SAT ON A WALL AND LOOKED AT THE SUN.
A DOG SAT ON THE STREET AND CHEWED A BONE.

DATA SOURCE MODELS

Hierarchical Bayes

- clustering is based on a simple hierarchical Bayesian source model
- latent variables can be recovered by EM or Bayesian methods
- simple clustering assumes $\Sigma = 1$
- both c and μ are latent; need oracle to recover actual class labels

Channel View

$c \in \mathbb{Z}_k \xrightarrow{\text{render}} \xi \xrightarrow{\text{noisy channel}} x \xrightarrow{\text{classifier}} P(c|x)$

OCR: render = digital typeset; noisy channel: printing + scanning

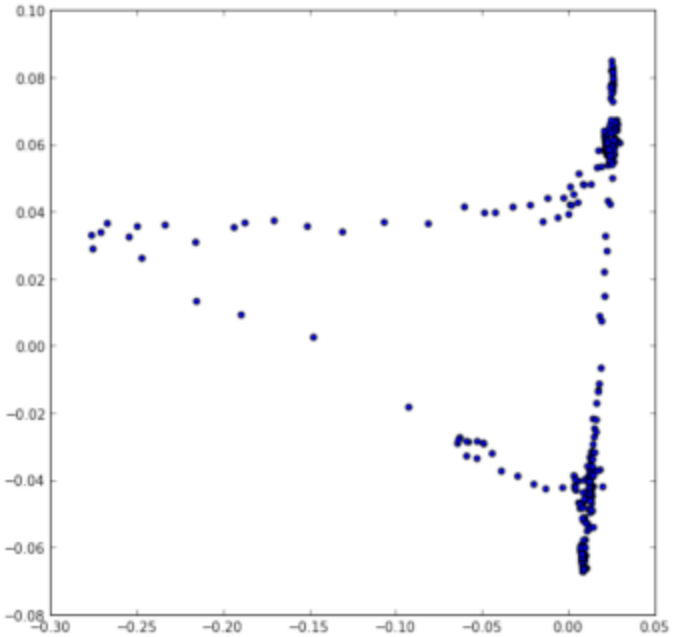
Vision: render = choice of pose ; noisy channel: lighting, camera

Rendering can involve transformation parameters, giving rise to *view manifolds*.

Real View Manifold

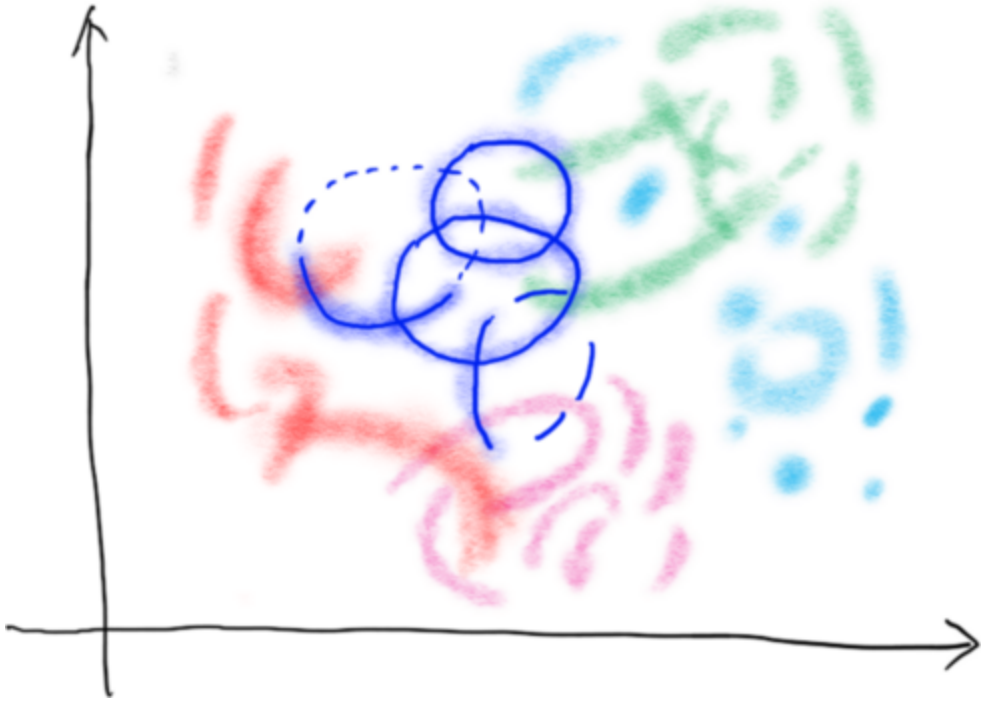
5 second motion sequence of object rotating back and forth

```
from sklearn.decomposition import RandomizedPCA
pca = RandomizedPCA(20)
lo = pca.fit_transform(data)
mds = manifold.LocallyLinearEmbedding()
vl = mds.fit_transform(lo)
scatter(vl[:,0],vl[:,1])
```



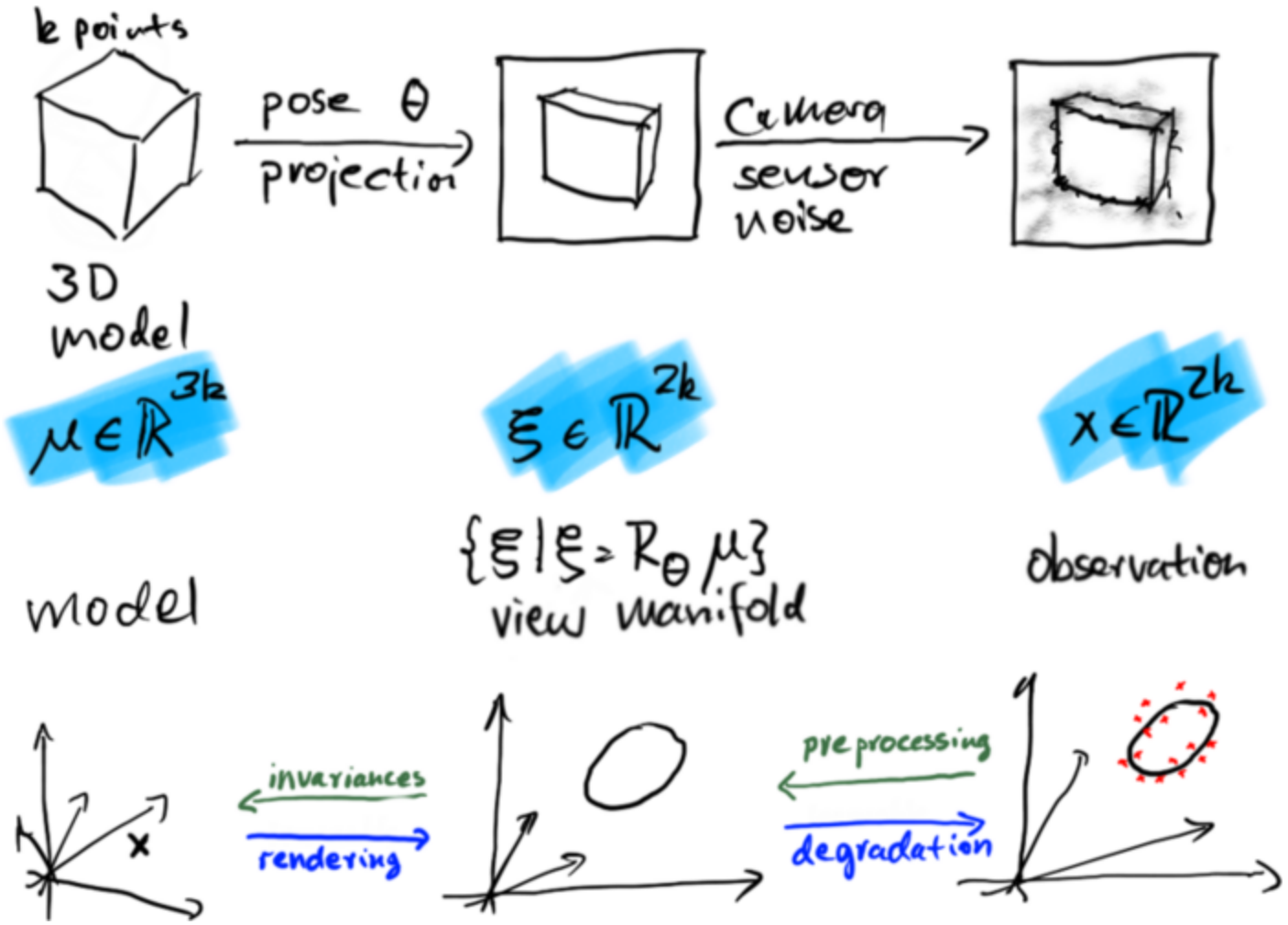
Dimensionality reduction in pixel space: $\mathbb{R}^{256} \rightarrow \mathbb{R}^2$. Video: rotating.mp4

Manifolds Learning



- think of real data as clouds surrounding...
- union of manifolds with boundary (only shown for blue class)
- "manifold learning" = union-of-manifold-s-with-boundary learning

Model, View Manifold, and Noise



Unsupervised Learning

Given or Learned:

- rendering
- degradation

Constructed or Learned (inverse problems):

- invariances
- preprocessing / image cleanup

CycleGAN can learn both directions simultaneously with no supervision.

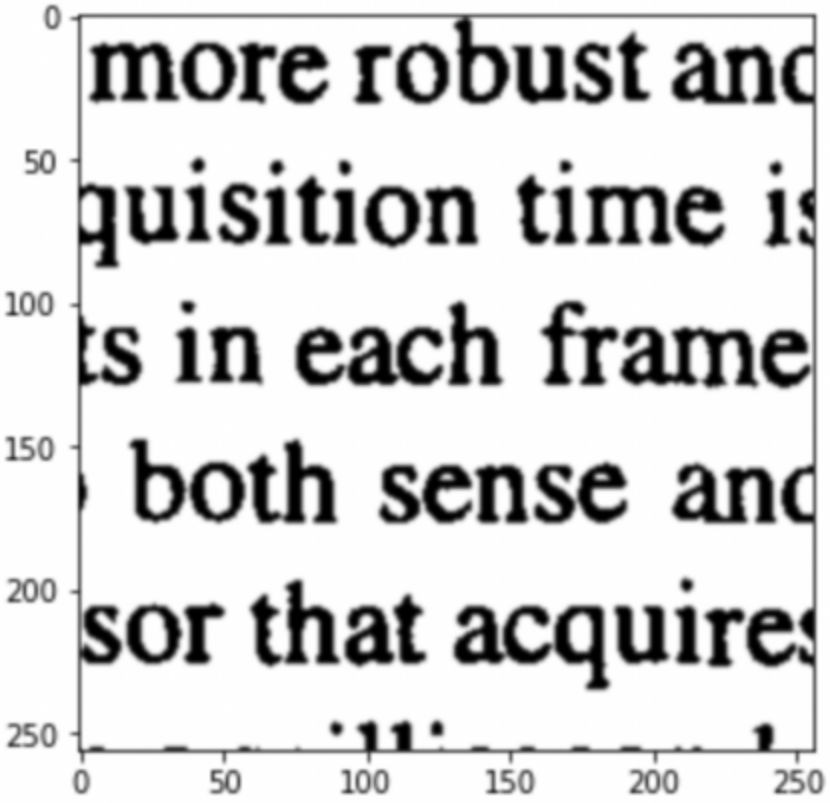
TRAINING DATA GENERATION / DEGRADATION

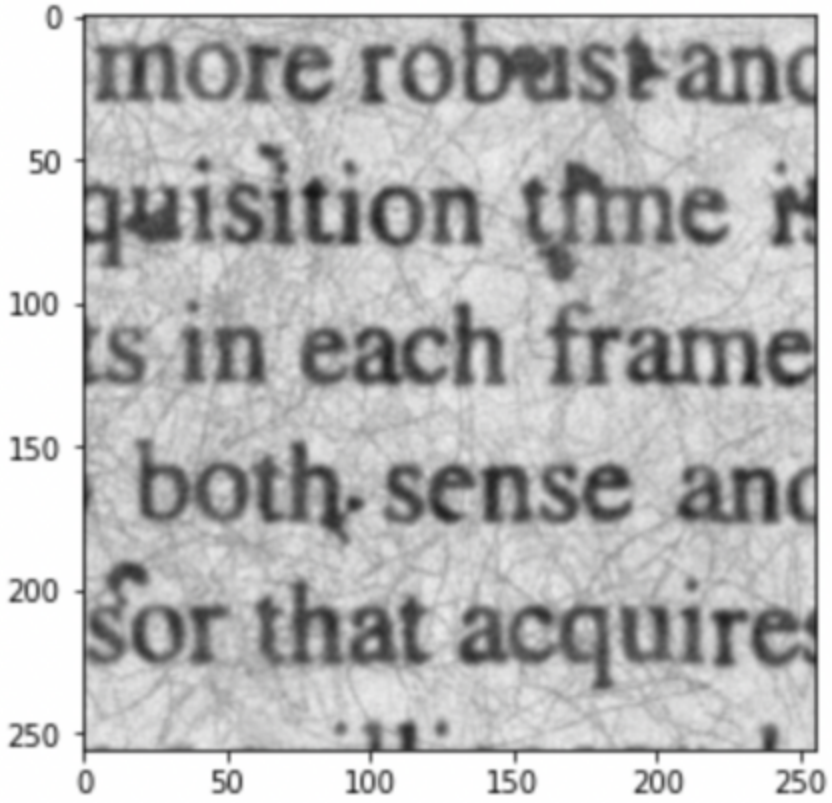
Artificial Data Generation

This is the "forward path" in the channel view of recognition:

- digital typesetting = perfect "artificial" document generation
- take any text, generate pages of perfect text
- for OCR, we need "degraded images"
  - printing, scanning, photographing, photocopying, ...
- physical processes for document image degradataion are well known

Artificial Data Generation





Document Image Restoration

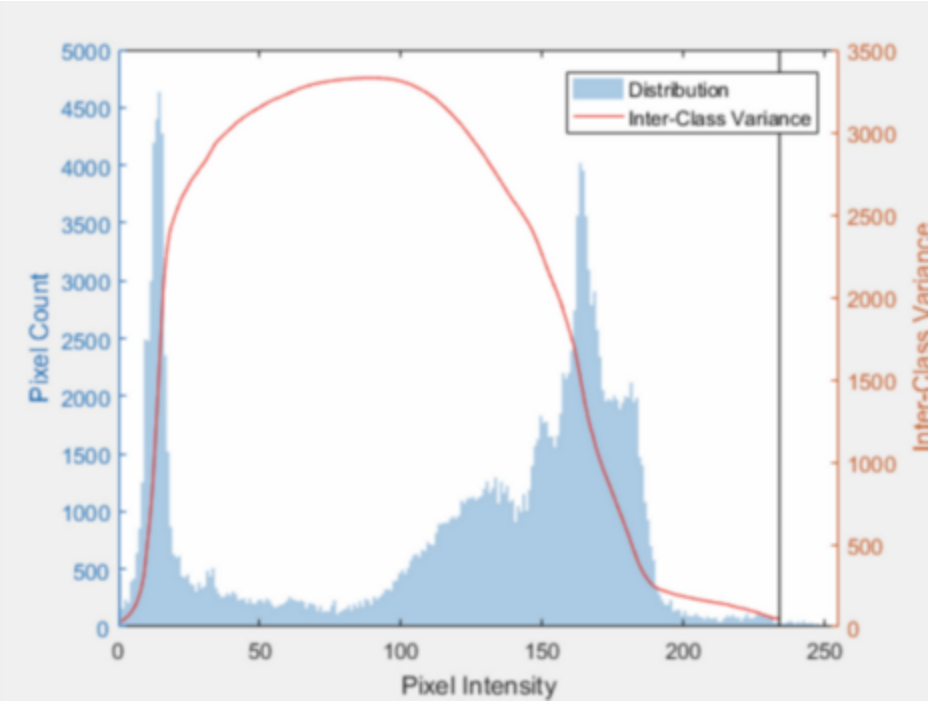
- recognition is easier if documents are not degraded
- can we restore the "clean" image via unsupervised learning?

AV/object recognition: image translation prior to recognition

Classical Document Image Restoration

- binarization
  - optimal thresholding
  - optimal linear filtering (deconvolution, etc.)
  - clustering
  - performance-based dynamic thresholding
- deep learning
  - supervised restoration (LSTM, pix2pix)
  - unsupervised restoration via CycleGAN

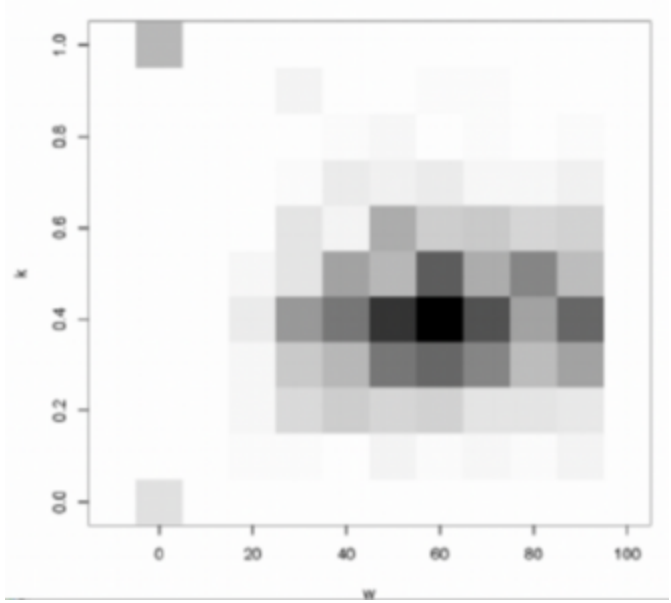
Otsu's Method: "Optimal" Thresholding



Otsu's method:

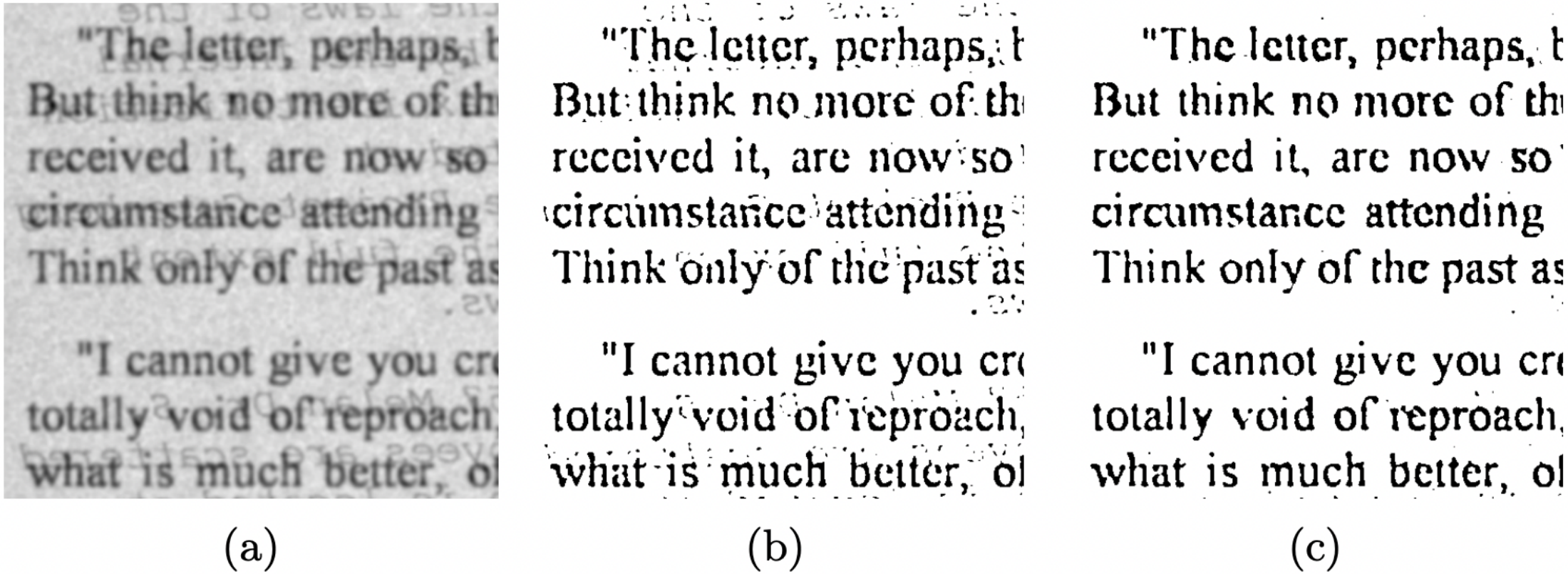
- manually constructed inverse degradation model based on prior knowledge/assumptions
- assume some kind of mixture model of image pixel generation
- maximize inter-class variance

Performance-Based Thresholding



- run thresholding with many parameters
- pick the parameters that yield the best OCR output
- no transcript, so use proxy
  - use statistics/classifier, or #words in dictionary
  - closely related to GAN methods
- assumption/prior: local thresholding = good way of inverting degradation model

Binarization by Supervised Deep Learning



(a) original, (b) Sauvola, (c) LSTM-based binarization

Self-Supervised Training for Binarization

1. generate clean images, degrade with degradation model
    - takes advantage of prior knowledge of degradataion models
2. take degraded images, use performance-based thresholding
    - takes advantage of knowledge of statistical properties of output

CycleGAN for Document Preprocessing

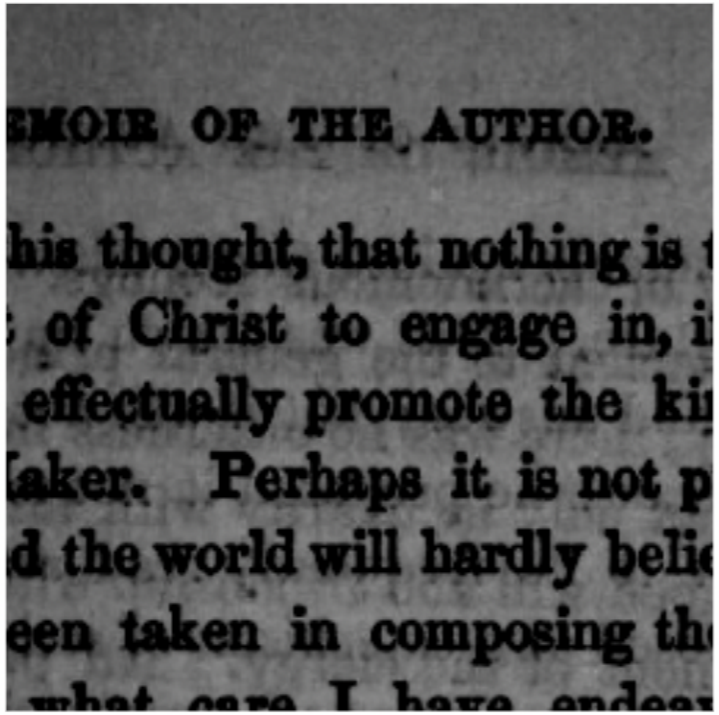
CycleGAN replaces all those components with trainable networks:

- clean image $\rightarrow$ degraded image
- degraded image $\rightarrow$ clean image
- clean image detector
- degraded image detector

These correspond to the forward and backward arrows in our channel model.

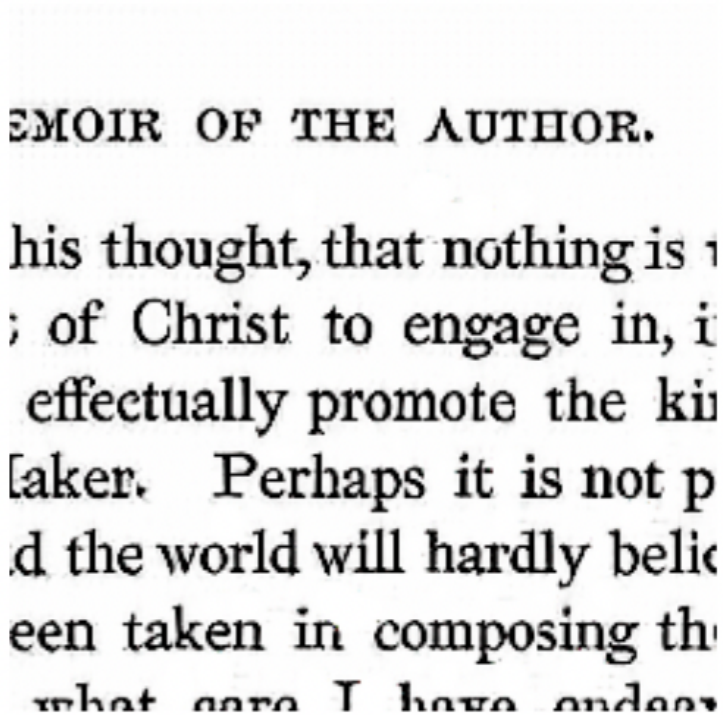
CycleGAN can be trained end-to-end without any labeled data or (significant) prior assumptions.

CycleGAN for Document Preprocessing



g_{AB}

g_{BA}



d_A d_B

End-to-End OCR

NEW MEMBERS Qualified

These applicants who have qualified for admission to the Society have been welcomed into membership between Aug. 15, 1944, and Sept. 15, 1944. The various grades of membership are indicated by: (M) Member; (A) Associate Member; (J) Junior; (Aff.) Affiliate Member; (SM) Service Member; (FM) Foreign Member.

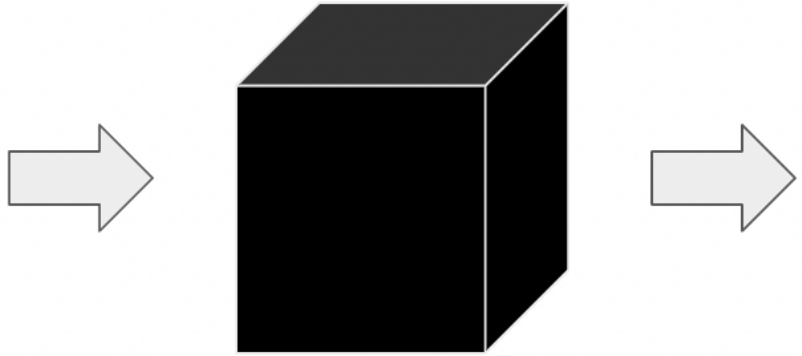
OBITUARIES

Philip G. Johnson

Lorraine J. D. Frazer

Daniel C. Green

Archie Frank Fanger



Weatherhead Co., has moved from the Cleveland to the Chicago office.
LT. MILLARD FILLMORE PERRY, U. S. Army Air Forces, has been transferred from the equipment laboratory, Engineering Division, Wright Field, Dayson, Ohio, to the Materiel Command, Central Procurement Division, Detroit, where he is assistant chief of the engineering change branch
…

End-to-End OCR

Recent Developments

- transformer models permit full end-to-end training for image-to-text transcriptions
- do not need intermediate segmentation, text-line recognition

Self-Supervised Training

- apply the same principles
- train an initial model using supervised data
- compute output on unlabeled data ("soft labels")
- correct the output using language models
- retrain
- possibly use auxiliary tasks for pretraining (later)

Summary

In the OCR example, we have seen most of the major concepts of unsupervised and semi-supervised training:

- EM training (aka soft labeling)
- use of language modeling as data source
- clustering
- unsupervised preprocessing / image enhancement

Coming Next

- review some classical statistics and machine learning
- look at "bag of tricks" for language models
- look at "bag of tricks" for object recognition
- look at modeling densities and GANs

STATISTICAL FOUNDATIONS

Supervised Learning Problem

Given: samples x_i with labels y_i drawn from some distribution $p(x, y)$

Goal: predict the label y for a new sample x.

Assume:

- $x \in \mathbb{R}^n$ - continuous measurements (e.g., images, speech, etc.)
- $y \in \{1 \ldots N_c\}$ - discrete class labels (classification problem)

Bayes Optimal Classifier

Assume y is discrete.

To minimize the expected 0-1 loss, classify according to the posterior probability of the class given the data.

This is called the Bayes optimal classifier.

$$D(x) = \arg\max_y P(y|x)$$

NB:

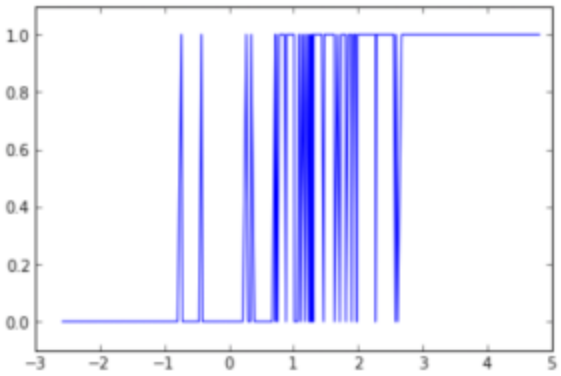
- $D(x)$ is the *decision function*.
- $\arg\max_y P(y|x)$ is really shorthand for $\arg\max_i P(y = i|x)$
- any function monotonically related to posterior gives the same decision function

Neural Networks and Bayes Optimal Classifiers

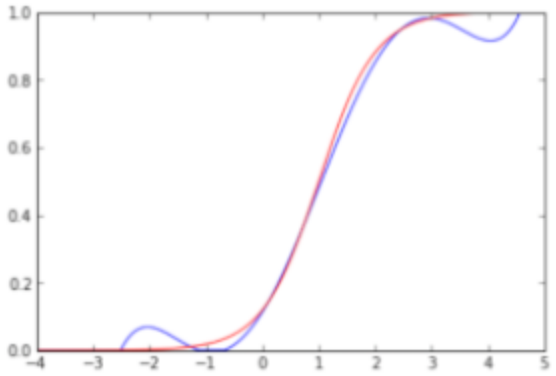
Neural networks are (universal) approximators for discriminant functions.

- neural networks $f_\theta(x)$ trained with one-hot softmax or MSE approximate the posterior $P(y|x)$
- we use the same $\arg\max_y$ approach to deriving a decision function from discriminant functions
- this approximates the Bayes optimal classifier by approximating $P(y|x)$

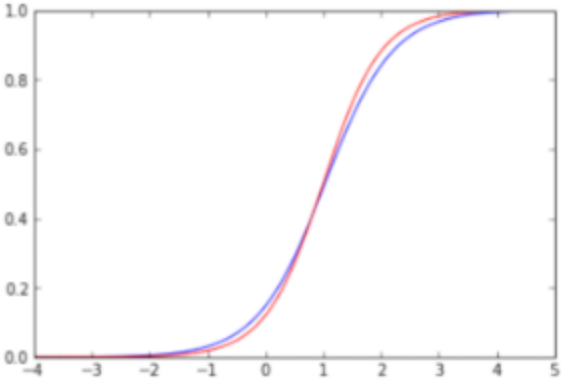
Function Approximation for $P(y|x)$



function to be approximated



lsq fit with polynomial



lsq fit with sigmoidal MLP

- lsq fit of any smooth function approximates posteriors
- meaningful posterior probabilities = approximation is not "good"

Bayes Formula

$$P(y|x) = \frac{p(x|y)P(y)}{p(x)} = \frac{p(x,y)}{p(x)}$$

Relationship between:

- prior class probability $P(y)$
- sample distribution, "evidence" $p(x)$
- posterior probability $P(y|x)$
- class conditional density $p(x|y)$

(I will use P for discrete distributions and p for densities.)

Bayes Formula: Learning Tasks

Any of the four factors in Bayes formula can be learned:

- posterior probability $P(y|x)$ - usually learned directly in supervised learning
- prior class probability $P(y)$ - easy to estimate by counting, useful for some classifier retargeting
- sample distribution, "evidence" $p(x)$ - a density estimation problem, useful for outlier detection, structure discovery, domain adaptation, sample generation
- class conditional density $p(x|y)$ - a density estimation problem, useful for classification, sample generation, structure discovery, etc.

We can also apply Bayes formula to learn one factor in terms of others, e.g., $p(x|y) = P(y|x)p(x)/P(y)$

Estimating Densities

Construct new densities from:

- parametric densities: normal, Cauchy, uniform, ...
- Cartesian product
- mixture densities
- linear and non-linear transformations of densities
- Bayes formula

Carefully maintain normalization.

NB: This is the basis of deep learning density estimators we will cover later.

Not Estimating Densities

In many cases, we don't need an explicit way of computing $p(x)$, we may simply need to be able to...

- only sample from $p(x)$ but not evaluate (e.g. generative models)
- determine whether $p(x) > p(x')$

Some deep learning models take advantage of this.

Parametric Densities

Uniform Density

- $p(x) = \quad_{x \in [0,1]^n}$

Normal Density

- $p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$
- $p(x) = \det(2\pi\Sigma)^{-1/2} \exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$
- maximum likelihood estimates of parameters either closed form or by gradient descent

NB: multivariate normal is Cartesian product followed by linear transformations

Mixture Densities

Any convex combination of densities is another density.

Let $p_i(x)$, $i = 1 \ldots k$, be densities.

Let $\lambda_i \in [0, 1]$ and $\sum \lambda_i = 1$, then

- $p(x) = \sum_i \lambda_i p_i(x)$

is another density.

Transformations of Densities

If $x \sim p(x)$, and $y = f(x)$, then obviously there is some $p(y)$ such that $y \sim p(y)$

We can use this to construct complex densities from simple ones, in particular if f is represented by a trainable DNN.
We will cover later what the form of $p(y)$ is in terms of $p(x)$

Simple example, linear transformation:

$$y = f(x) = M \cdot x + b$$

If $x \sim \mathcal{N}(0, 1)$, then $y \sim \mathcal{N}(\mu, \Sigma)$

Exercise: how do (M, b) relate to (μ, Σ)?

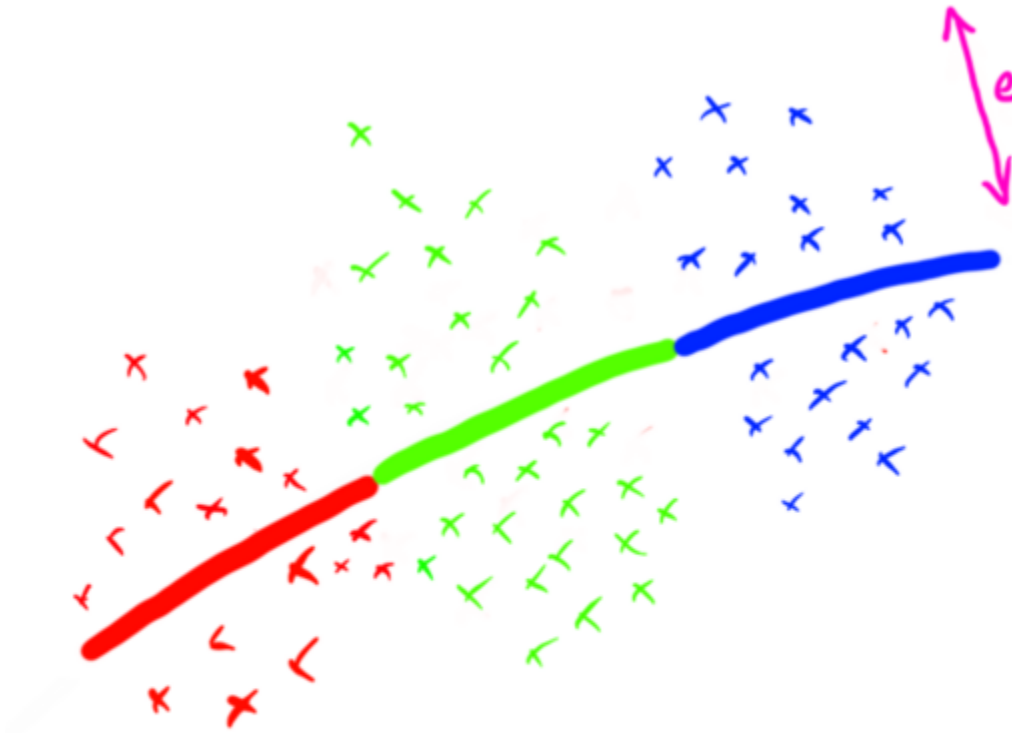
Nearest Neighbor Classification

- classify each sample based on the class of its nearest (k-nearest) neighbor(s)
- if e^* is the Bayes error rate, then k-nearest neighbor error e_k asymptotically achieves $e^* \leq e_k \leq e^*(2 - \frac{k}{k-1}e^*)$
- this limit works only for very large numbers of "training samples"
- this assumes samples come from a metric space, but assumes little about the metric
- the limit simply assumes that we have enough samples that the class conditional densities around a sample x up to the k-nearest neighbor are approximate constant

Learned Distance Functions and Nearest Neighbor Classification

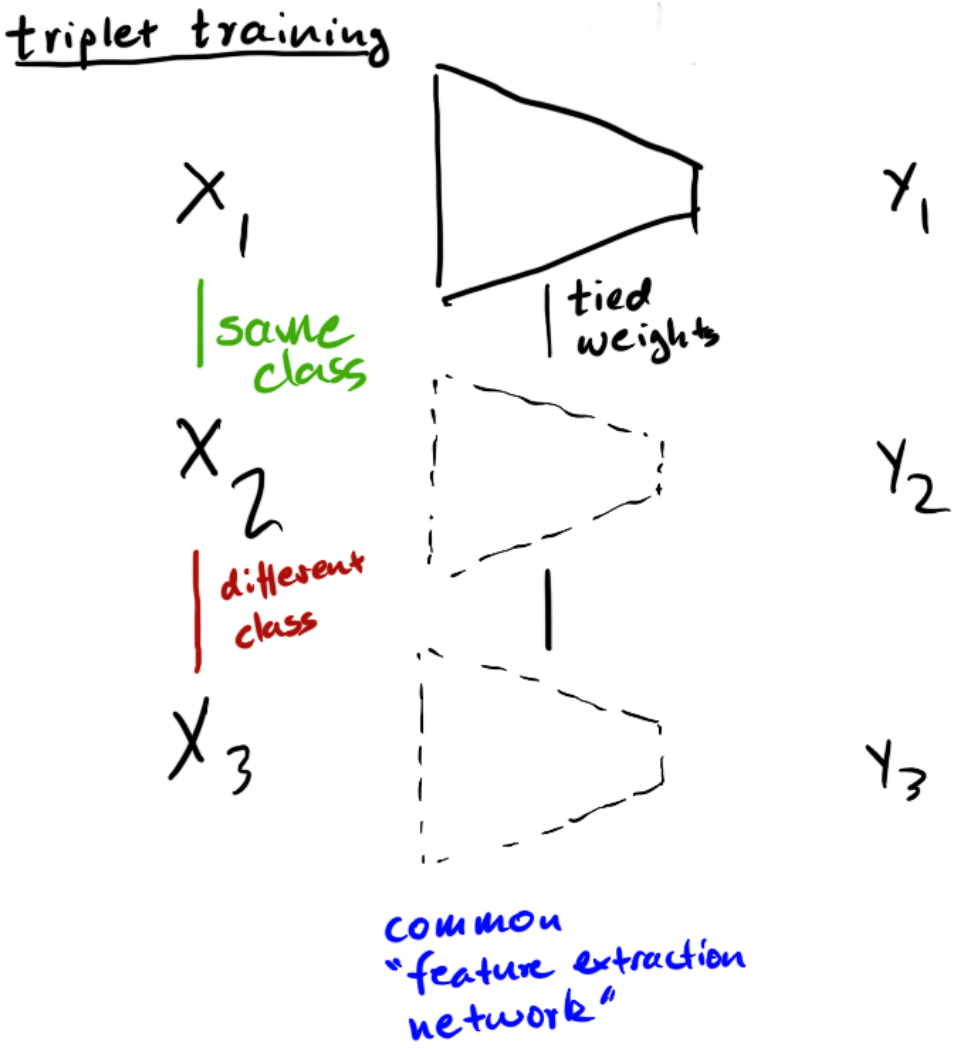
- nearest neighbor classification relies on some distance measure $d(x, y)$ to find the k nearest neighbors of a sample
- some distance measures work may work better than others
- e.g., a distance measure may measure only in the direction of meaningful sample variation, ignoring "noise dimensions"
- Examples:
  - Mahalanobis distance (based on covariance matrix)
  - Siamese networks (deep learning)
- distance measures can be trained on a small labeled training set and then be used to generalize to a large unlabeled training set; for clustering; etc.

Learned Distance Functions



- Measure distance parallel to data surface
- measure "geodesic distance" in data surface
- learn parameterized distance function $d_\theta(x,y)$ so that NN-rule gives low classification error

Siamese Networks



A

$$\tilde{P}_\theta(\text{same} \mid y_1, y_2) = 1$$
$$\tilde{P}_\theta(\text{same} \mid y_1, y_2) = 0$$

B

$$\|y_1 - y_2\| < \|y_2 - y_3\|$$

$$\text{loss} = \max(0, \|y_1 - y_2\| - \|y_2 - y_3\|)$$

C

$$\frac{y_1 \cdot y_2}{\|y_1\| \|y_2\|} = 1 \qquad \frac{y_2 \cdot y_3}{\|y_2\| \|y_3\|} = -1$$

"Linear Siamese Networks"

What if $y = f(x)$ is linear?

- we obtain *some* Mahalanobis distance
- this matrix differs from both PCA and ICA in general
- even if the class conditional densities are Gaussian, this may not be an optimal distance measure, since decision boundaries can be quaddratic
- c.f. Fisher linear discriminant analysis
- c.f. decision boundaries for Gaussian class conditional densities

Summary

- optimal classification is governed by posterior probabilities and loss functions
- Bayes rules identifies the different probability functions we can learn: $P(y)$, $p(x)$, $P(y|x)$, $p(x|y)$
- we can construct complicated densities from simple densities by representing them as non-linear transformations of simple densities
- learning distance functions can also be helpful for unsupervised or semi-supervised learning
- we can use Siamese networks to learn useful feature extractors and distance functions

LINEAR METHODS

Linear Methods

Examples:

- PCA, ICA, k-means, GMM
- can often be trained directly w/o gradient descent

Why?

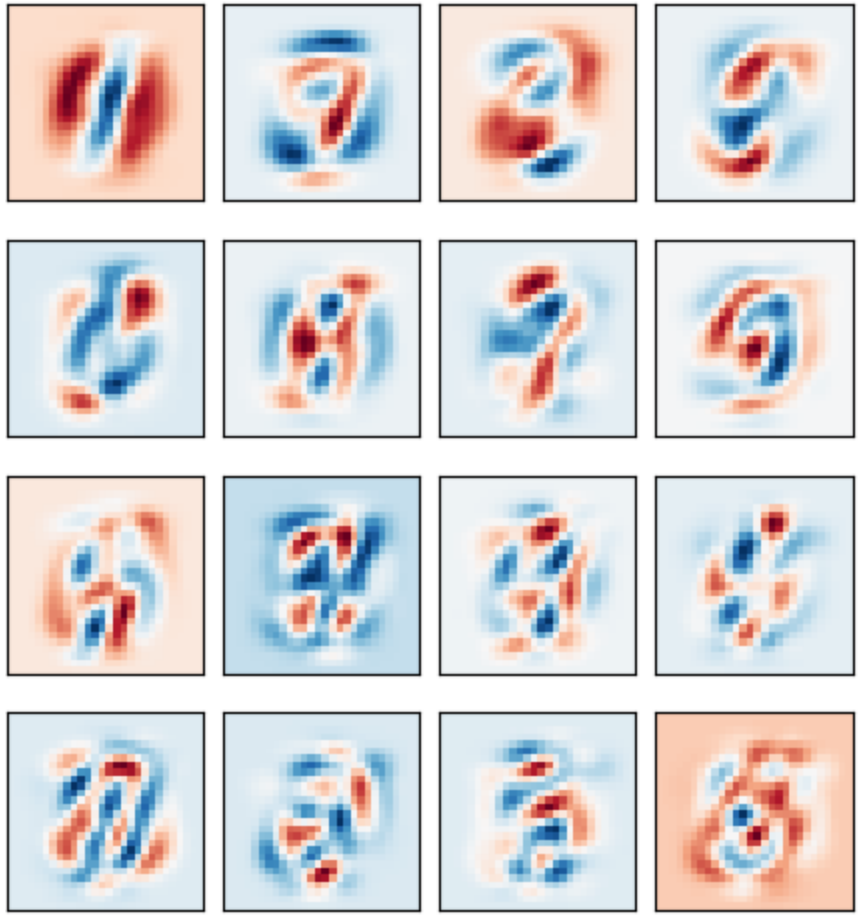
- many deep learning methods are generalizations of linear methods
  - replace $y = M \cdot x + b$ with $y = f_\theta(x)$
- deep neural networks are usually locally linear
  - linear model is basis for analysis of deep networks

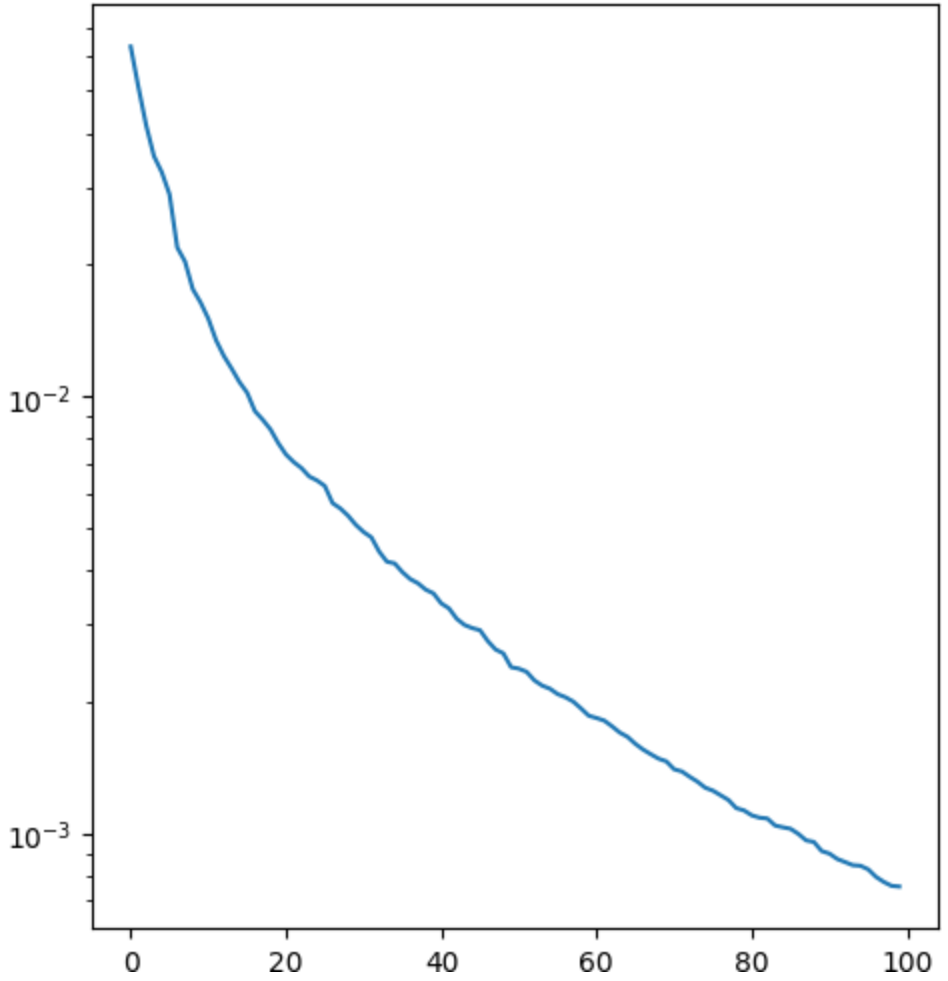
PCA

Example: Eigenfaces

Turk and Pentland, JCN 1991

Example: MNIST

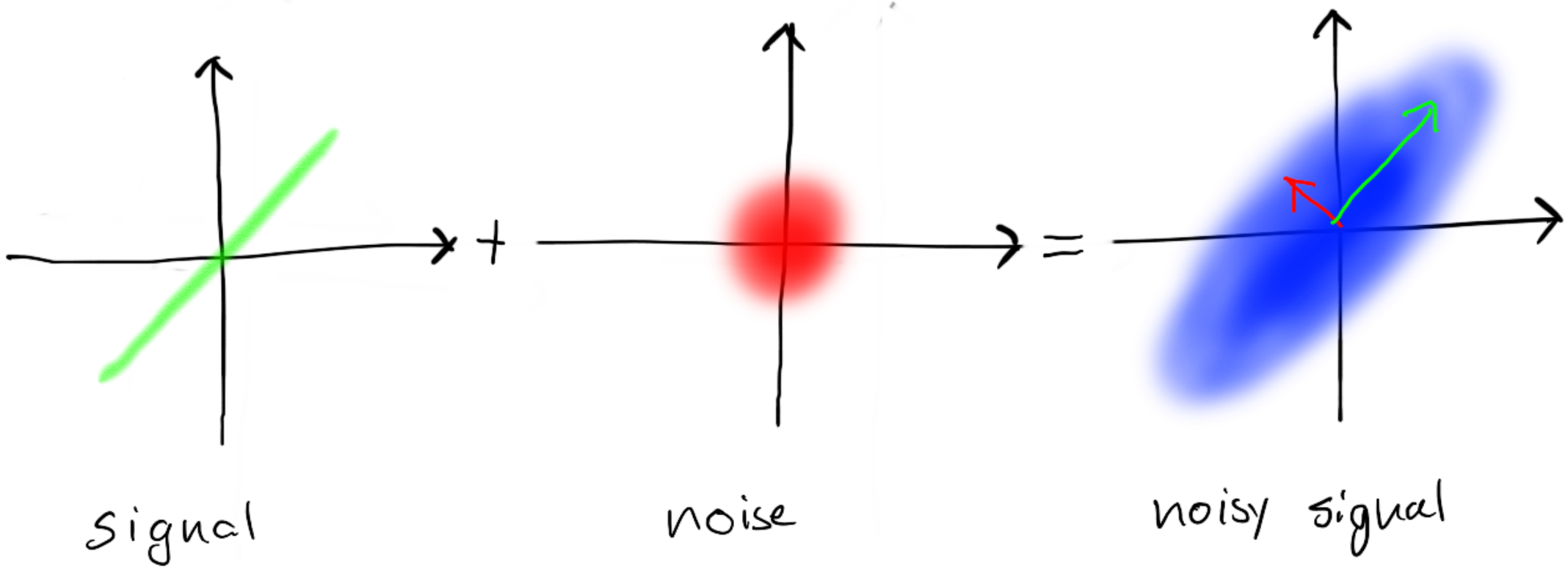




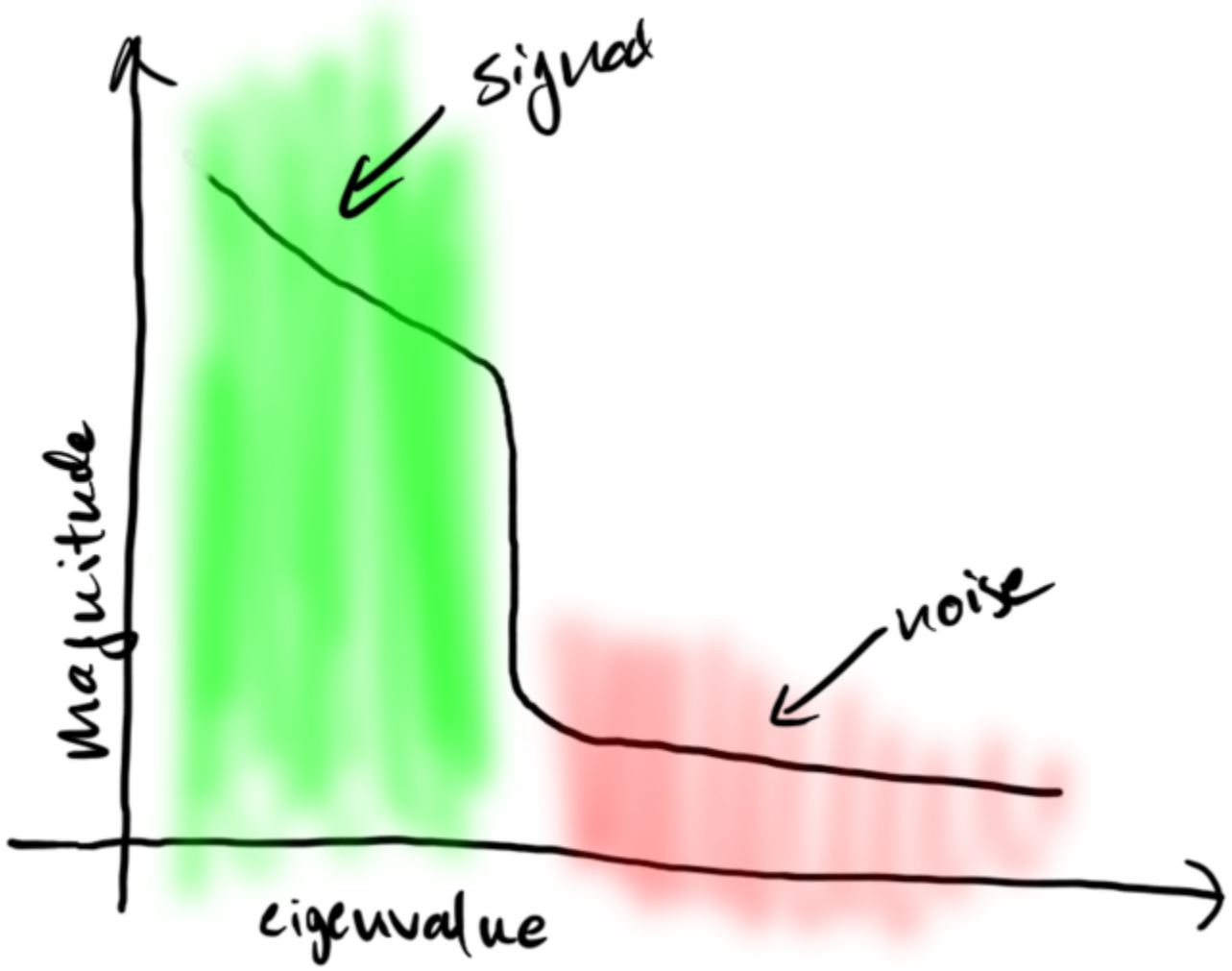
PCA

- take original data
- perform PCA
  - center and compute covariance matrix
  - project onto subspace with high variances
- perform nearest neighbor classification after projection
- generally results in better performance than nearest neighbor in original space
- why?

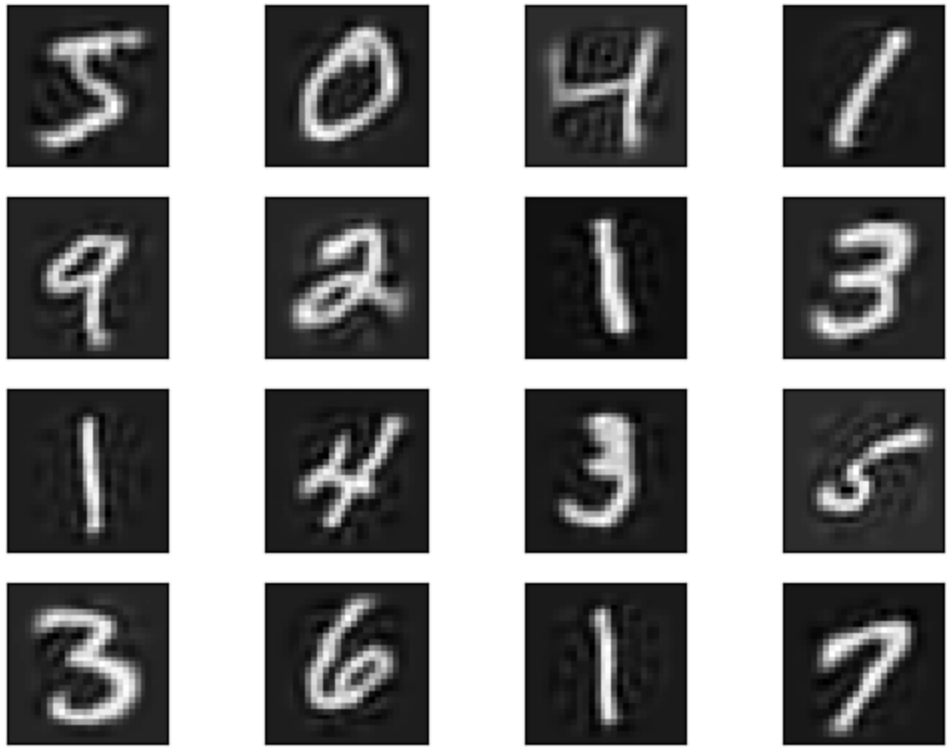
Separation of Signal and Noise using PCA



Separation of Signal and Noise using PCA



Example: PCA Reconstruction of MNIST



Separation of Signal and Noise using PCA

Assume:

- $x = \xi + \nu$ - signal ξ and additive noise ν
- variation in the signal ξ is in a linear subspace
- noise is Gaussian and small relative to variation in signal

Then:

- "large PCA components" correspond to the signal
- "small PCA components" correspond to noise

"Projection onto data manifold."

PCA and Compression

Notice that we can use PCA for compression:

- keep only the "largest PCA componenets"
- project the original signal onto those components
- code the projected values

This is a simple example of the relationship between unsupervised learning and compression.

Nearest Neighbor and PCA

We can perform nearest neighbor with PCA in several ways:

- transform the original data with $y = Mx + b$ and perform nearest neighbor classification on the y
  - "latent representation"
- project the original data into subspace $x' = M^T Mx$ and perform nearest neighbor classification on the projected x'
  - "projection onto data manifold", "denoising"
- define a new distance function using the Mahalanobis distance
  - "metric learning", "Siamese networks"

ICA

Prior Assumptions

- PCA
  - meaningful signal is large
  - noise is small
- ICA
  - meaningful signal(s) is/are non-Gaussian
  - noise is Gaussian

Both are approximately satisfied for "view manifolds" of 3D objects and in many other cases.

Unsupervised Separation of Signal and Noise using ICA

Assume:

- $x = \sum \xi_i v_i = M \cdot \xi$
  - the individual ξ_i are not Gaussian and independent
  - the v_i are non-colinear unit vectors

Then:

- ICA will recover the v_i and lets us recover the underlying ξ_i

Therefore: unsupervised recovery of ξ from samples drawn according to $p(x)$

ICA Implementation

Find a matrix $W \in \mathbb{R}^{k \times n}$ that is orthonormal $ W W^T = 1 $ that

- the components of $W \cdot x$ have minimal mutual information
- the components of $W \cdot x$ are maximally non-Gaussian
- that minimizes $\mathbb{E}||W \cdot x||_1$

NB:

- any of these can be used as objective functions
- no reference to reconstruction error; does not minimize reconstruction error
- full ICA spans full input space

"Neural Network" ICA = RICA

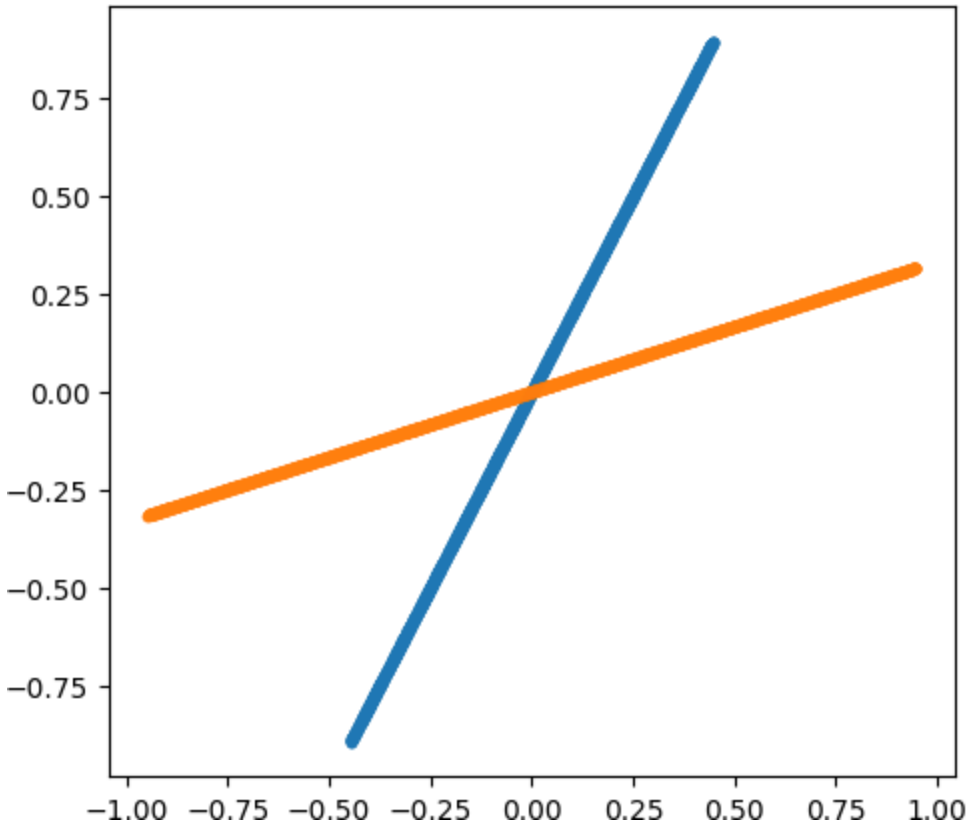
Optimize W via gradient descent over the data matrix X:

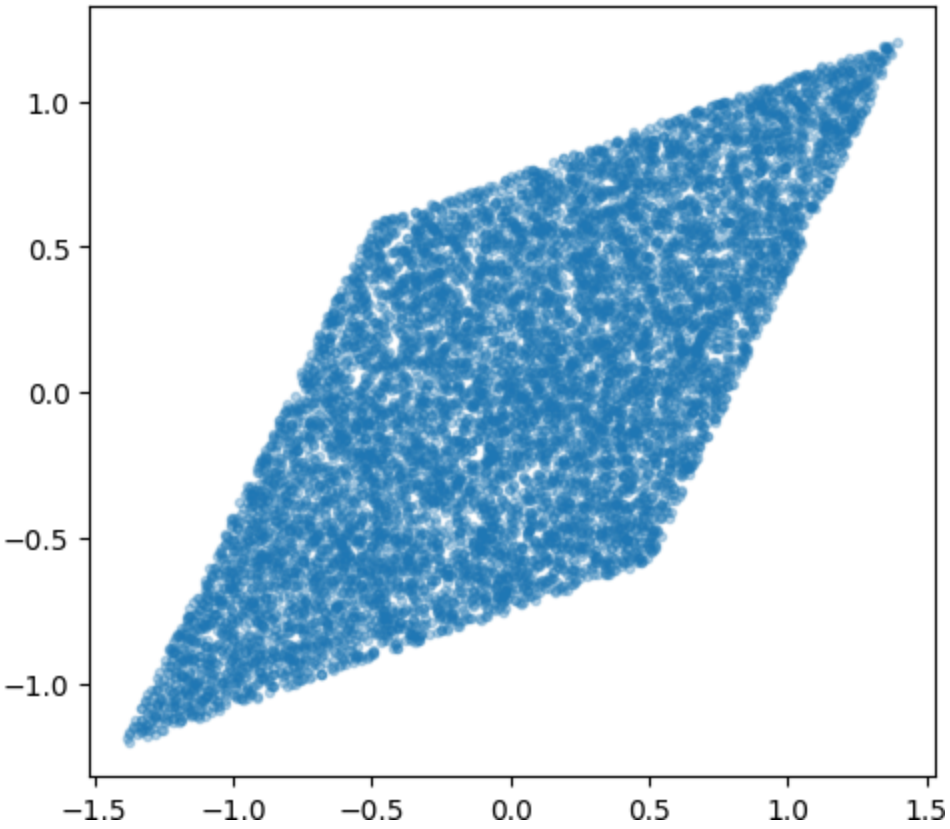
$$\hat{W} = \arg\min_w \lambda||WX||_1 + ||W^T Wx - x||_2^2$$

We may additionally explicitly constrain $W^T W =$.

- For $\lambda \to \infty$ and whitened data, this is equivalent to ICA.
- For $\lambda = 0$, this is simply PCA or a linear autoencoder.

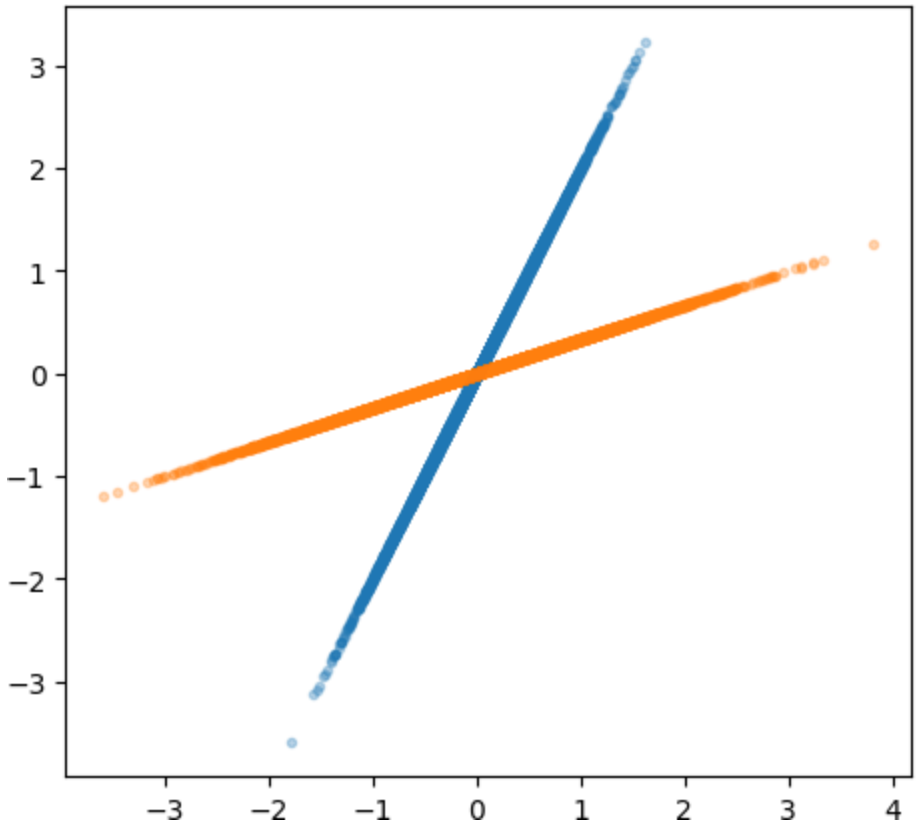
ICA and Non-Gaussian Densities

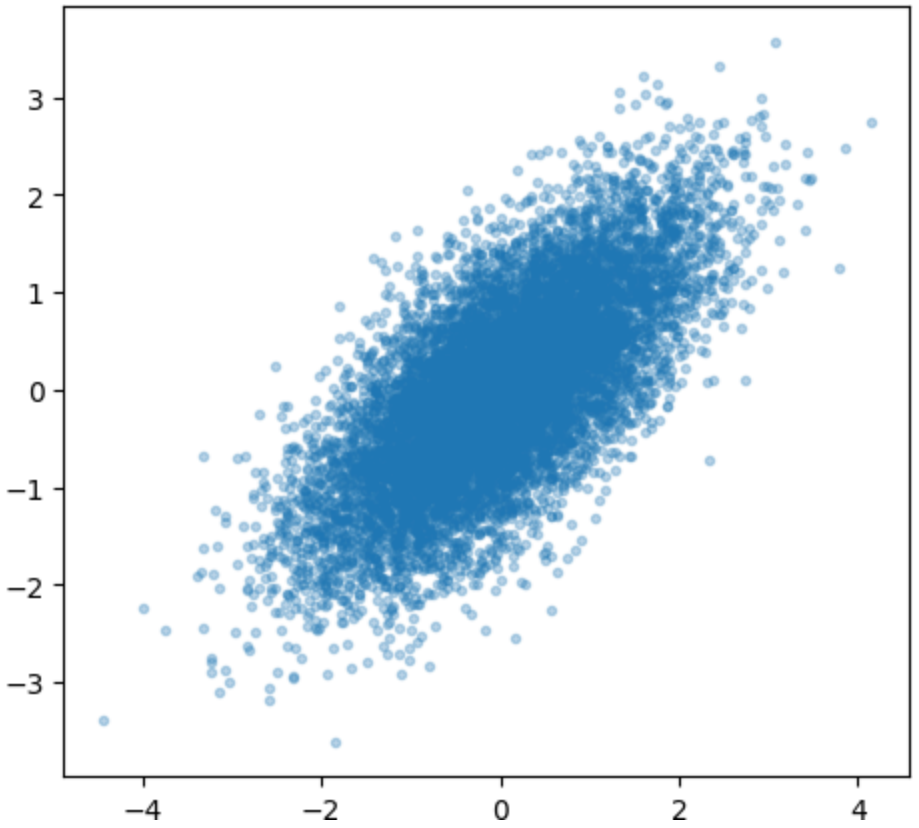




Projections along the two original directions give rise to maximal non-Gaussian distributions (uniform, in this case).

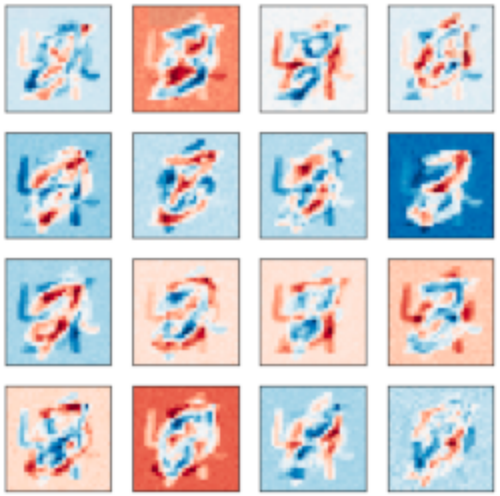
ICA and Gaussian Densities



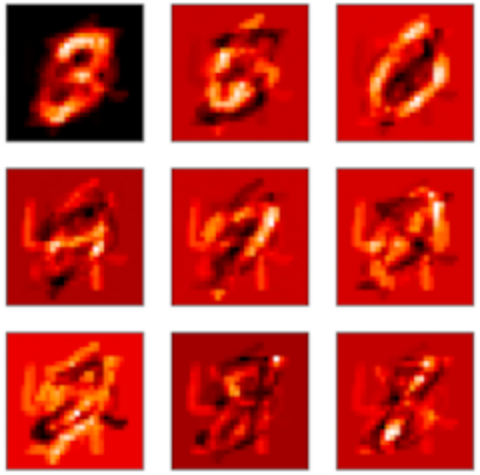


If the component densities are Gaussian, we cannot recover their directions.

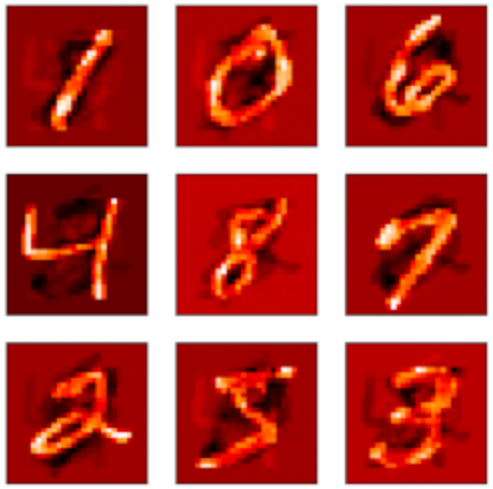
PCA vs ICA on Mixtures of MNIST Characters



mix of 9 characters



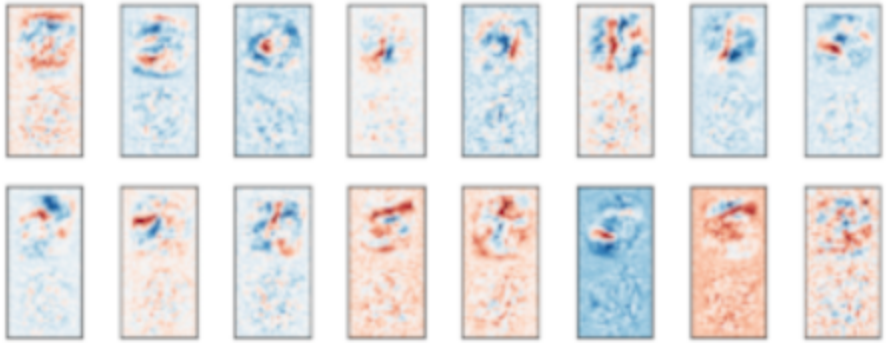
PCA



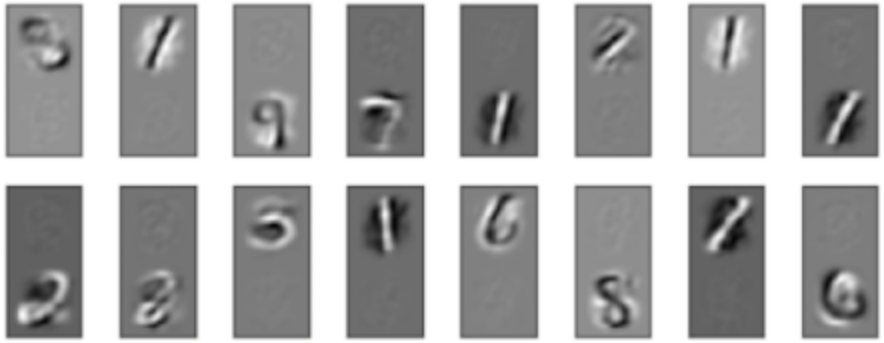
ICA

Irrelevant Features, ICA, and Neural Networks

MNIST digit pairs; bottom = distractor



weights in simple NN reflect relevance of top, irrelevance of bottom of image

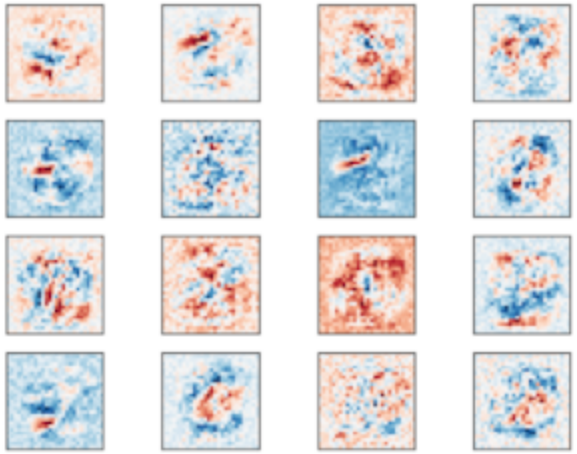


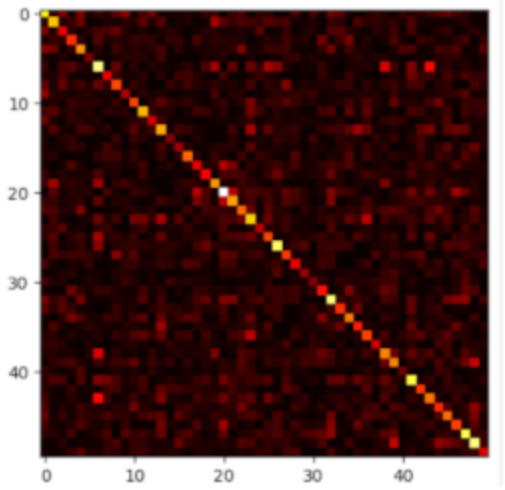
ICA separates top and bottom; cannot determine relevance

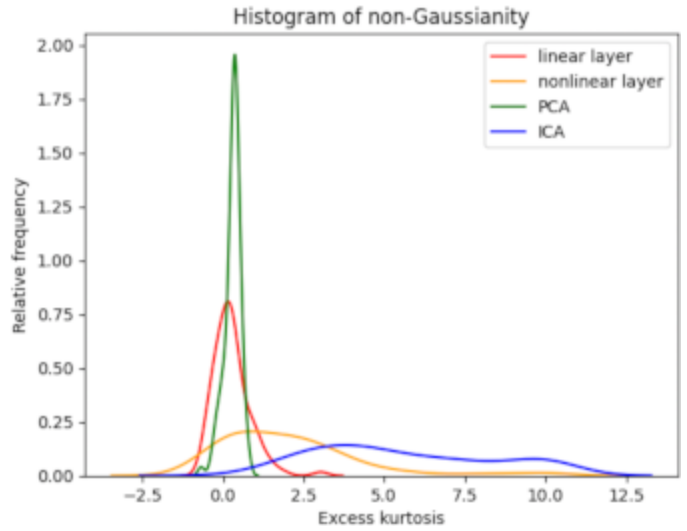
linear reconstruction of input from hidden layer activations; more information is lost about bottom

Do Neural Network Layers perform ICA / RICA?

```
Sequential(
  (0): Linear(in_features=784, out_features=100, bias=True)
  (1): ReLU()
  (2): Linear(in_features=100, out_features=10, bias=True)
)
```







GAUSSIAN MIXTURE MODELS

Gaussian Mixture Models

Why?

- good approximation to many real densities
- implicit or explicit in some deep learning models
- introduction to latent variables and the EM algorithm

Gaussian Mixture Models and k-Means

$p(x) = \sum_j \lambda_j \mathcal{N}(x; \mu_j, \Sigma_j)$

where

$\sum \lambda_j = 1, \lambda_j \in [0, 1]$

Special case: k-means ($\Sigma_j = 1$)

EM Algorithm and Latent Variables

For each sample x_i, the latent variable is cluster membership.

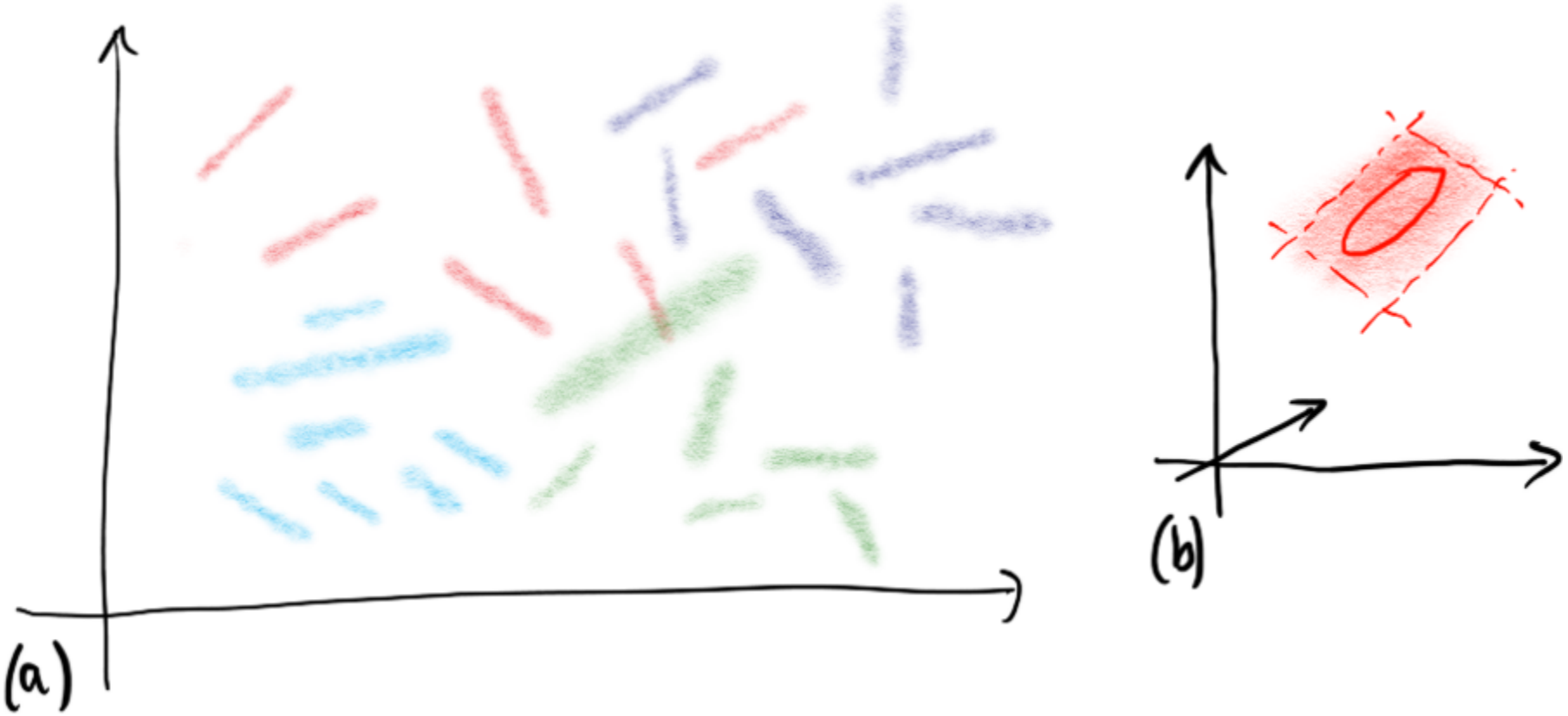
1. initialize parameters randomly
2. maximize $\prod p(x_i)$ by assigning x_i to best cluster j
    - assignment is latent variable
3. recompute μ_j and Σ_j using this assignment
4. repeat at step 2

A kind of "gradient descent" or "coordinate descent"; may get trapped in local minima.

The Meaning of Mixture Components

- many classification problems are mixture density problems
  - each class consists of multiple prototypes
  - prototypes can undergo transformations
  - after transformation there is additive Gaussian noise
- Gaussian mixture models can represent both...
  - (linear) transformations (small variance directions)
  - additive noise (large variance directions)

Linear Manifolds Learning in Vision



(a) Gaussian mixtures are collections of "linear manifolds with errors" (in $\mathbb{R}^{65536}$)

(b) $P \cdot R \cdot \xi + p$, true: $R \in SO(3)$, approx: $R \in \mathbb{R}^{3\times3}$

Identify mixture components, then just assign a class label to each component.

Relationships between Linear Models

All these models represent samples x as a linear combination z of basis vectors.

$x = M \cdot z$

Different constraints on z and different objectives:

- PCA - MSE objective, z has smaller dimension
- ICA - components of z are independent / non-Gaussian
- k-Means, VQ - MSE objective, z are basis vectors ("one hot")
- NNMF - all components of z are positive

Summary

"Traditional" Unsupervised Learning Techniques:

- PCA - local decomposition into signal/noise components
- Mahalanobis distance - automatic distance learning
- ICA - identify independent components
- mixture density estimation - identify mixture components, often corresponding to class conditional densities

Many of the idea for unsupervised deep learning are already present here in the linear case.

SEQUENCE MODELING

Sequence Modeling

- classic
  - probabilistic language models (n-grams, backoff, WFST)
  - LPC
  - HMM
  - images: HMM-OCR, turbo coding, inpainting
- deep learning
  - word2vec
  - character level LSTM
  - GPT
  - BERT
  - MAE

Type of Sequences

- time series
  - climate data
  - stock market
  - scientific measurements
  - health data
- text
  - sequence of characters
  - sequence of words
- images and videos
  - various ways of turning into sequences

CLASSICAL THEORY

Self Supervised Learning and Sequences

Discrete time sequence:

- $x_1, x_2, \ldots, x_t, x_{t+1}$

Sequence learning is naturally unsupervised/self-supervised:

- Predict x_{t+1} given $\{x_t \ldots x_1\}$

The sequence itself serves as a source of supervised training data.

Language Models

For language models, the x_t come from a finite alphabet $\Sigma \cong \mathbb{Z}_k$

A *string* is a finite sequence of symbols: $s = s_1 \dots s_l \in \Sigma^*$

A *language model* is an estimate of the true distribution $P(s)$ over strings.

Language Models by Counting

If the language model has strings of length less than some L, there is a finite number of tototal possible strings.

We can sample a training set $S = s_1, \ldots, s_N$ from $p(s)$ and estimate:

$$\hat{P}(s) = \frac{1}{N} \#\{s_i \in S : s_i = s\}$$

This is impractical for large L.

Factoring the Language Model

Simple conditional expansion:

$$P(s) = P(s_l|s_{l-1}\dots s_1)P(s_{l-1}\dots s_1)$$

Apply recursively:

$$P(s) = P(s_l|s_{l-1}\dots s_1)P(s_{l-1}|s_{l-2}\dots s_1)\dots P(s_1)$$

Assume: $P(s_i|s_{i-1}\dots s_1) = P(s_i|s_{i-1})$

$$P(s) = P(s_l|s_{l-1})P(s_{l-1}|s_{l-2})\dots P(s_1)$$

This is the bigram model. It is a simple kind of Bayesian network.

Why autoregressive?

There are lots of probability estimates we could compute, e.g. using left and right context.

$$P(s_l | s_{l+2}, s_{l+1}, s_{l-1}, s_{l-2})$$

We factor left to right (causally) because:

- we need a directed acyclic graph for efficient belief propagation / inference
- we often operate "causally" and "in real time"

NB: once $P(s)$ has been computed, it includes both left and right context, regardless of factorization

Easy Sampling

Sampling from $P(s)$:

- sample s_1 according to the prior $P(s_1)$
- sample s_2 according to $P(s_2|s_1)$
- sample s_t according to $P(s_t|s_{t-1}\dots s_1) \approx P(s_t|s_{t-1})$

Deal with finite sequences and positional dependencies by having start and end tokens.

Estimating Parameters for the Bigram Model

$$P(s) = P(s_l|s_{l-1})P(s_{l-1}|s_{l-2})\dots P(s_1)$$

How do we estimate the $P(s_t|s_{t-1})$?

Counting: "What is the distribution of words following the word 'giraffe'?"

Estimating Parameters for the Bigram Model

```
bigram = {}
for i in range(len(shakespeare)-1):
    word = shakespeare[i]
    bigram.setdefault(shakespeare[i], Counter())[shakespeare[i+1]] += 1

for k, v in bigram["king"].most_common(10):
    print("king %10s %.4f" % (k[:10], v/n))
```

```
king      henry 0.1325
king    richard 0.0914
king        and 0.0562
king     edward 0.0500
king       john 0.0408
king         of 0.0345
king          i 0.0191
king         is 0.0178
king     philip 0.0171
king         my 0.0128
```

n-Grams

```
ngrams = {}
for i in range(len(shakespeare)-n):
    context = tuple(shakespeare[i:i+n-1])
    ngrams.setdefault(context, Counter())[shakespeare[i+n-1]] += 1Python

ngrams[("king", "and")].most_common(10)
```

```
[('queen', 15),
 ('his', 10),
 ('all', 5),
 ('not', 5),
 ('the', 4),
 ('commonweal', 4),
 ('who', 3),
 ('if', 3),
 ('i', 2),
 ('polonius', 2)]
```

Backoff and Smoothing

For large n, many strings may not occur in the training set:

- `ngrams[("king", "and", "many", "other")].most_common(10)`

yields no results, yet probability isn't zero. What do we do?

Try shorter n-grams:

- `ngrams[("and", "many", "other")].most_common(10)`
- `ngrams[("many", "other")].most_common(10)`
- `ngrams[("other")].most_common(10)`

Back-off: approximate $P(x_t|longcontext)$ using $P(x_t|shortercontext)$

Sampling from Shakespeare

Training: complete works of Shakespeare, tokens=words

n=1

- Can, ., a to see out BERTRAM eyes I . . And true sphere shall see?
- Not . judgment, experience The, daily, and,, and soundly . BY in we hair
- books- laid like .' To How shepherd me here fray les . and . thy, smelt, me

n=2

- friend doth still we send us'd from whom shouldst thou art just.
- whom, no bush suppos'd to see thy youtli and fearful of duty.
- ignoble traitor; and had your evils.

Sampling from Shakespeare

n=3

- wak'd, As thou lov'st me, here he comes- one of the realm shall not hedge us out
- clear of thee thine uncles and myself Have travail'd in Rouen.
- and his man, sir; namely, the dancing banners of the Emperal's men.

n=5

- Then shall we hear their 'larum, and they ours.
- want, Taste grief, need friends: subjected thus, How can you say to this?
- a quiet sword; by the means whereof' a faces it out, but fights not.

Deep Learning and Sequences

Deep learning models are really good at modeling conditional probabilities:

$$P(s_t|s_{t-1}\dots s_1) = f_\theta(s_{t-1}\dots s_1)$$

where f_θ is some kind of neural network.

This means:

- sequentially estimate conditional probabilities
- multiply the conditional probabilities to obtain the final distribution
- "autoregressive models"

Sequence Models for Classification

Classification:

$$P(c|s_t \dots s_1) = \frac{P(s_t \dots s_1|c)P(c)}{P(s_t \dots s_1)}$$

Translation:

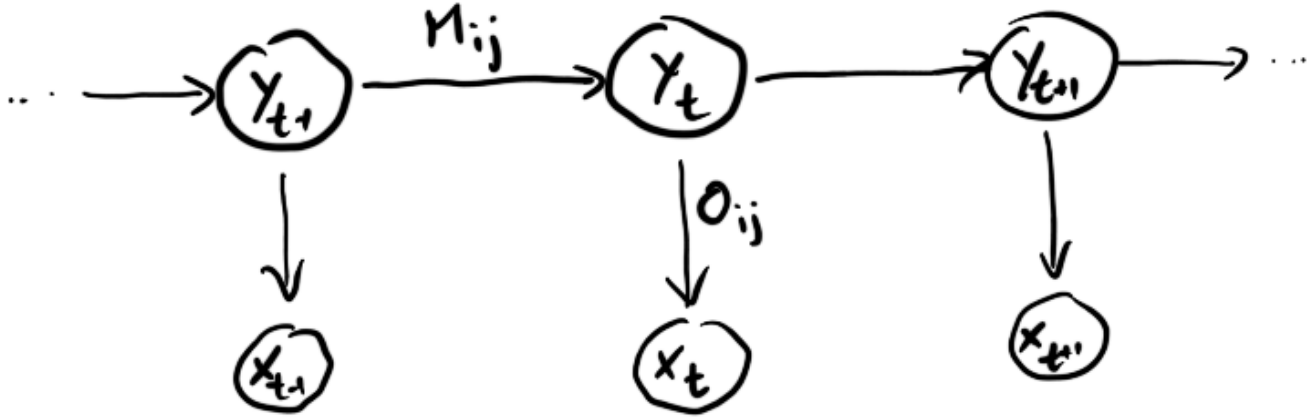
$$P(o_t \dots o_1|s_t \dots s_1) = P(o_t|o_{t-1} \dots o_1, s_t \dots s_1) = \dots$$

Common classication tasks for text: sentiment, genre, topic

Evaluation of Sequence Models

- perplexity
- BLEU score
- task specific score:
  - speech recognition error rate
  - classification error rate
- user studies

Hidden Markov Models



Simple Case:

X_t, Y_t discrete

$P(Y_{t+1}=i \mid Y_t=j) = M_{ij}$

$P(X_t=i \mid Y_t=j) = O_{ij}$

- Y_t is a Markov process (not observable)
- X_t are observations
- X_t is probabilistically determined by Y_t

Popular b/c of efficient and scalable algorithms for training and inference.

Generalizations of HMMs

- sequence-to-sequence (transducers)
  - c.f. transformer models
- replace inputs and outputs with vectors in $\mathbb{R}^n$
  - e.g. via vector quantization
- replace transition probabilities with potentials (CRFs)
  - c.f. energy-based learning

"Traditional" speech recognition with HMMs combines many/all these approaches.

We find many of the same ideas in deep learning now.

UNSUPERVISED DEEP LEARNING FOR NLP

Language Modeling for Its Own Sake

Predict the next word in a sequence:

- generate new text (sampling)
- assign probabilities to strings (language modeling)

Examples:

- n-gram models, finite state transducers
- TDNN, LSTM models (e.g. Graves 2013, Arxiv 1308.0850)

Language Modeling for Transfer Learning

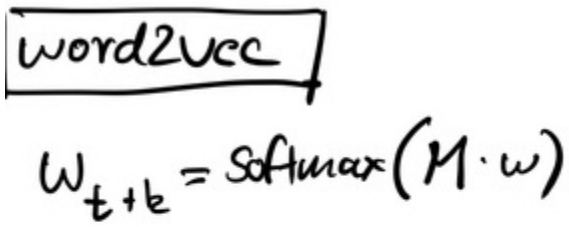
Task: predict words in sequences

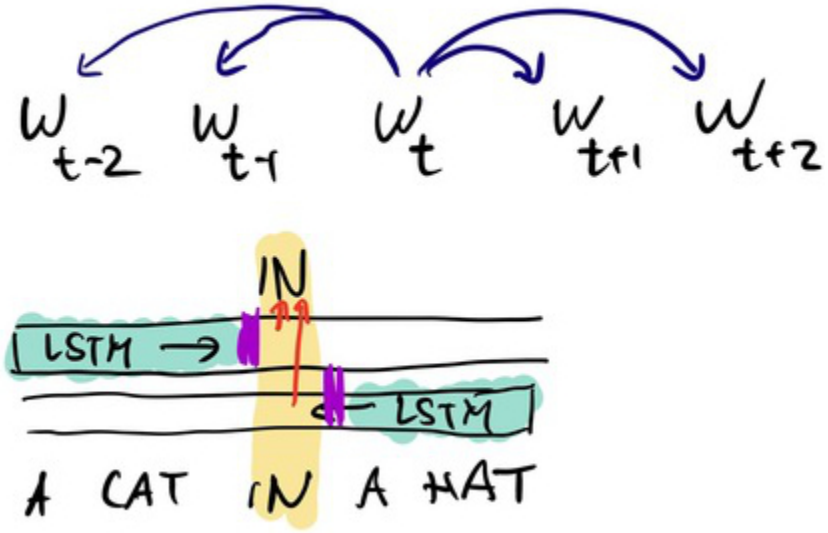
Purpose: obtain a useful *embedding*; transfer learning; generation

Examples:

- word2vec
- ELMO
- BERT, GPT, etc.

word2vec and ELMO





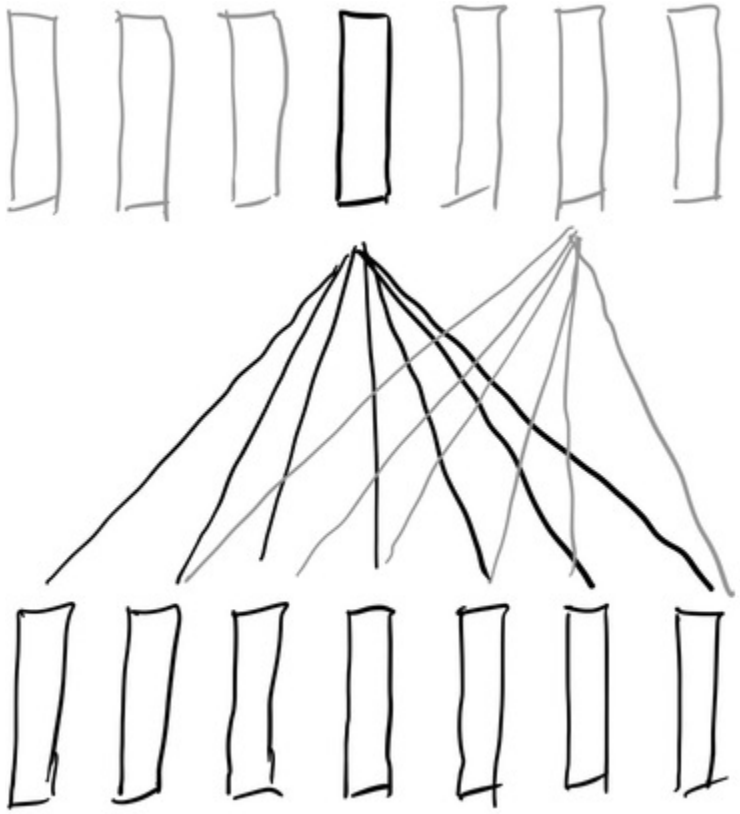
- self-supervised task: predict word from context
- bidirectional (since we're not using it for autoregressive decoding)
- embeddings used as inputs to other systems

TRANSFORMERS (QUICK REVIEW)

Transformer Architecture

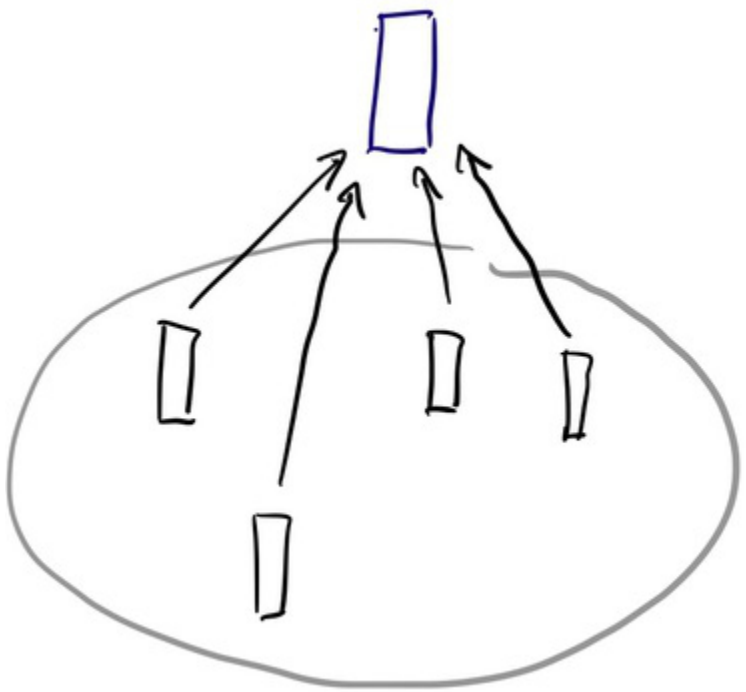
- fundamental change in sequence modeling
  - arbitrarily long term dependencies through content-addressability
  - all time steps trainable in parallel
- history
  - added attention mechanisms to LSTM to improve performance
  - later, eliminated the LSTM altogether and retained just attention

Transformers are Trained in Parallel



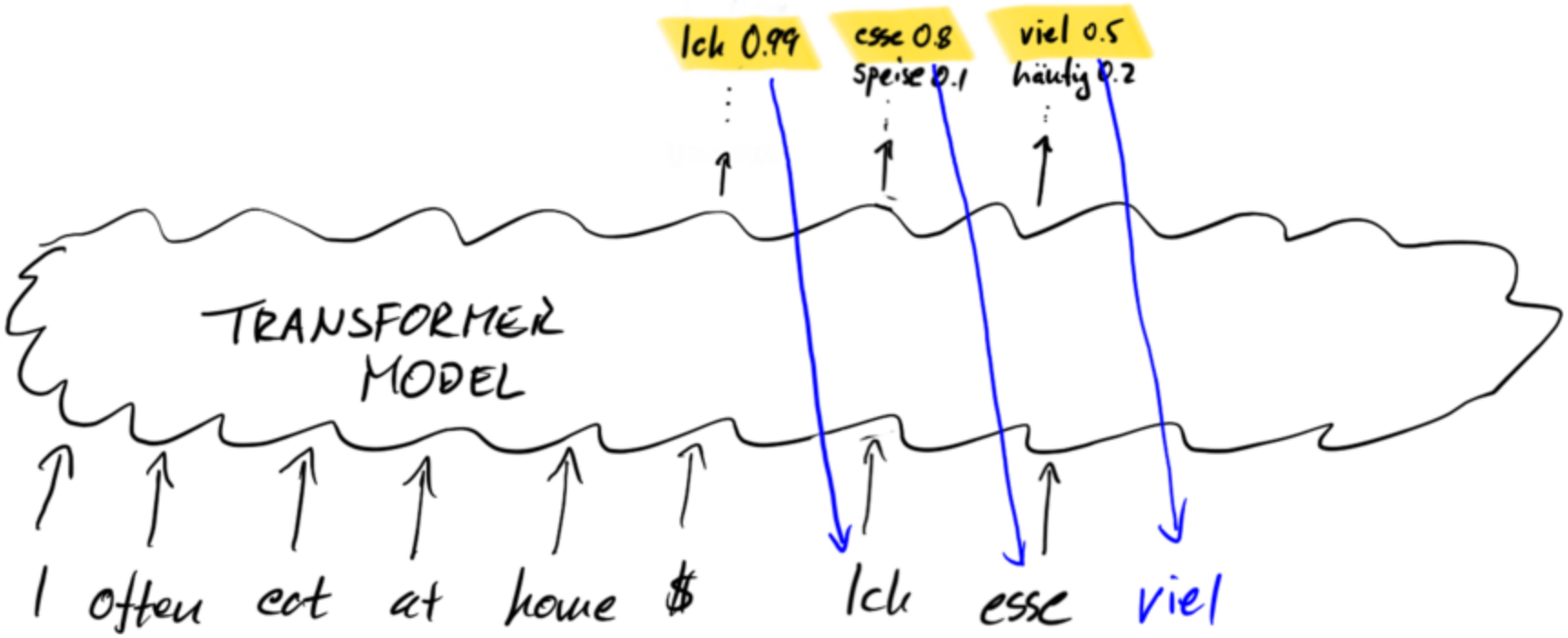
(Like convolutional networks.)

Transformers are Set Learners

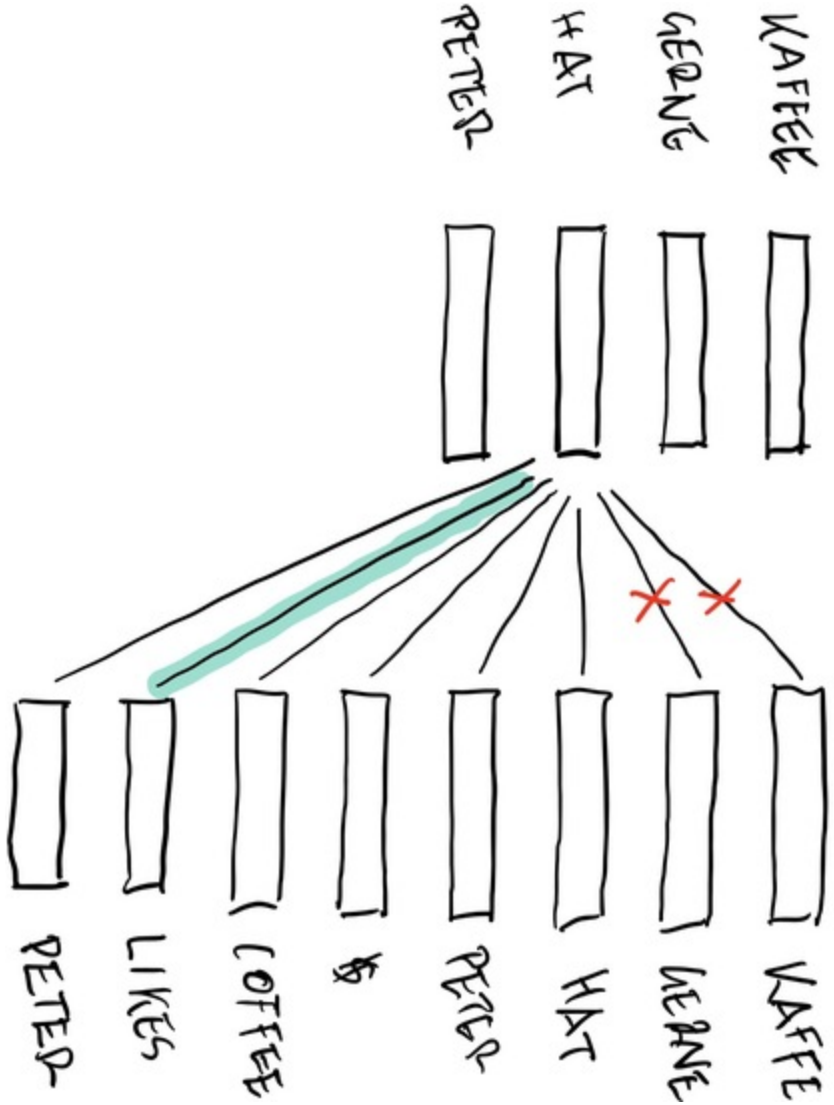


The order of inputs do a transformer doesn't matter as far as the model is concerned.

For Sequence Tasks, we use Autoregressive Decoding



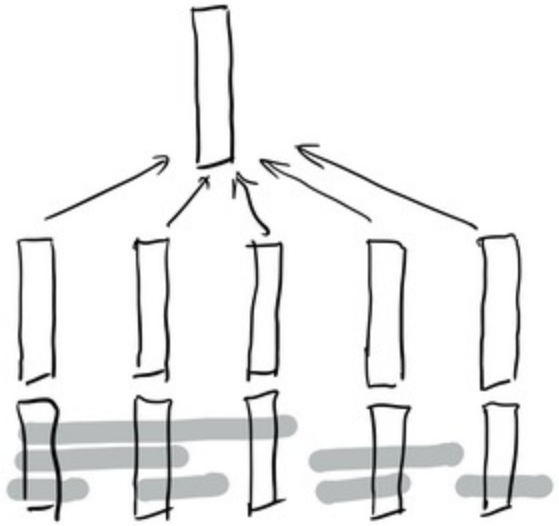
Attention in Transformers



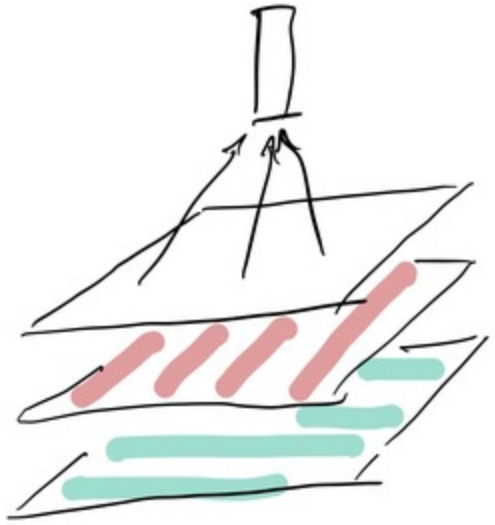
ATTENTION
USED FOR TWO TASKS:

- SELECT RELEVANT PARTS OF INPUT (VIA KEY-VALUE QUERY)
- MASK PARTS OF THE INPUT ALLOWING AUTOREGRESSIVE DECODING

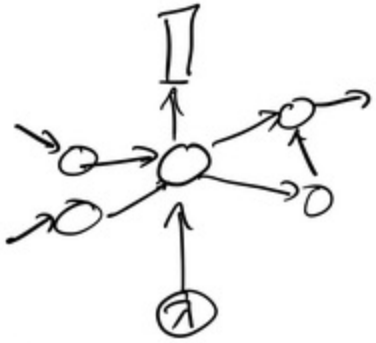
Sequences are Modeled via Positional Encodings



TRANSFORMERS USE POSITIONAL INFORMATION BY ADDING A "POSITIONAL BARCODE TO THE INPUTS



2D POSITIONAL ENCODING FOR IMAGES



GRAPH LAPLACIAN AS POSITIONAL ENC.

SELF-SUPERVISED LANGUAGE LEARNING WITH TRANSFORMERS

BERT

Basics:

- architecture: multi-layer transformer blocks
- dataset: books (800M words), Wikipedia (250M words)

Intended Use:

- transfer learning for NLP tasks (more general than word embeddings)

BERT - Self-Supervised Tasks

"Cats like playing with string."

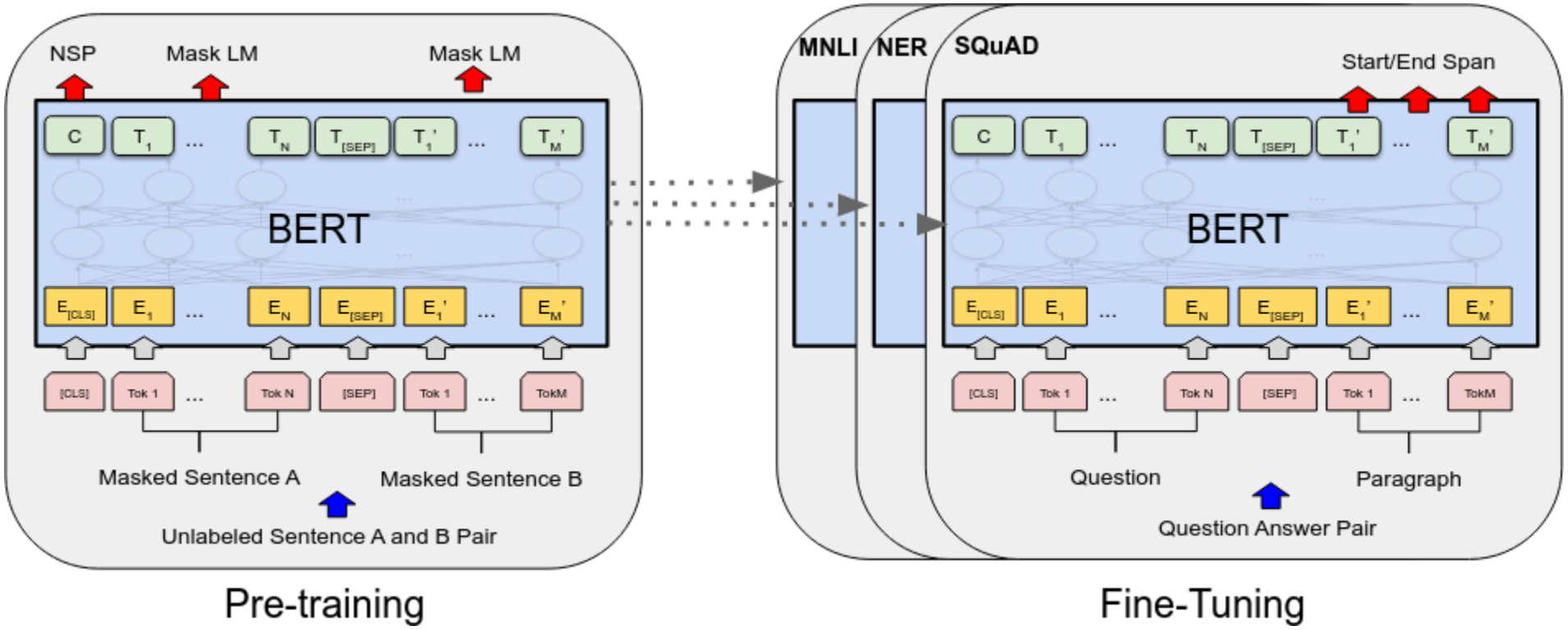
Task 1:

- corrupt 15% of the words: "Cats like [MASK] with string."
- restore original string

Task 2:

- sentence entailment (a classificaiton task)
- "My cat likes [MASK] with string. [SEP] He [MASK] likes milk." -- True

BERT Pretraining vs Fine Tuning



Devlin et al., 2019, Arxiv 1810.04805v2

BERT - Results on GLUE Benchmarks

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

Devlin et al., 2019, Arxiv 1810.04805v2

GPT-Style Models (Generative Pre-Training)

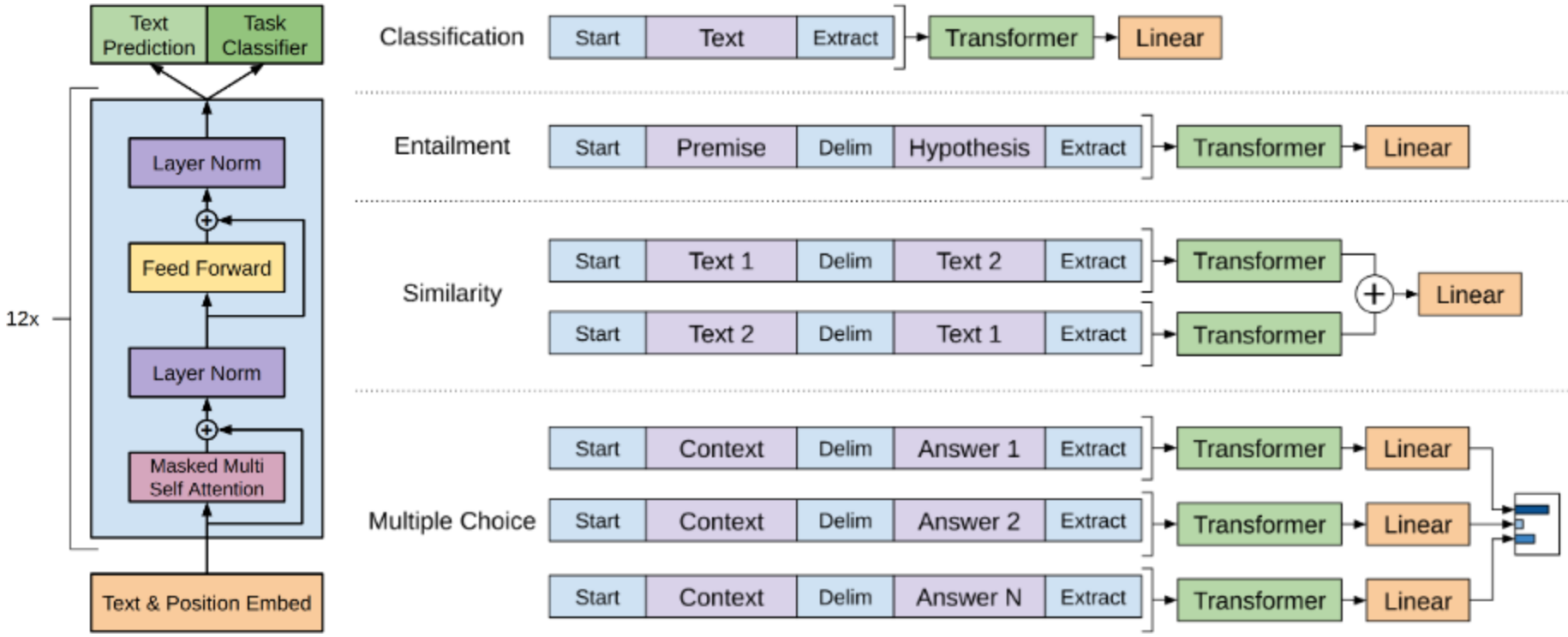
Basics:

- transformers with autoregressive language modeling
- model size: 117 million (GPT) TO 175 billion (GPT-3) parameters
- data: up to 400 billion words (GPT-3)
    - books, WebText, huge crawl cleaned with WebText Model

Intended Use:

- text generation, direct question answering
- transfer learning (fine-tuning on labeled data)
- few-shot learning via prompting

GPT Fine Tuning

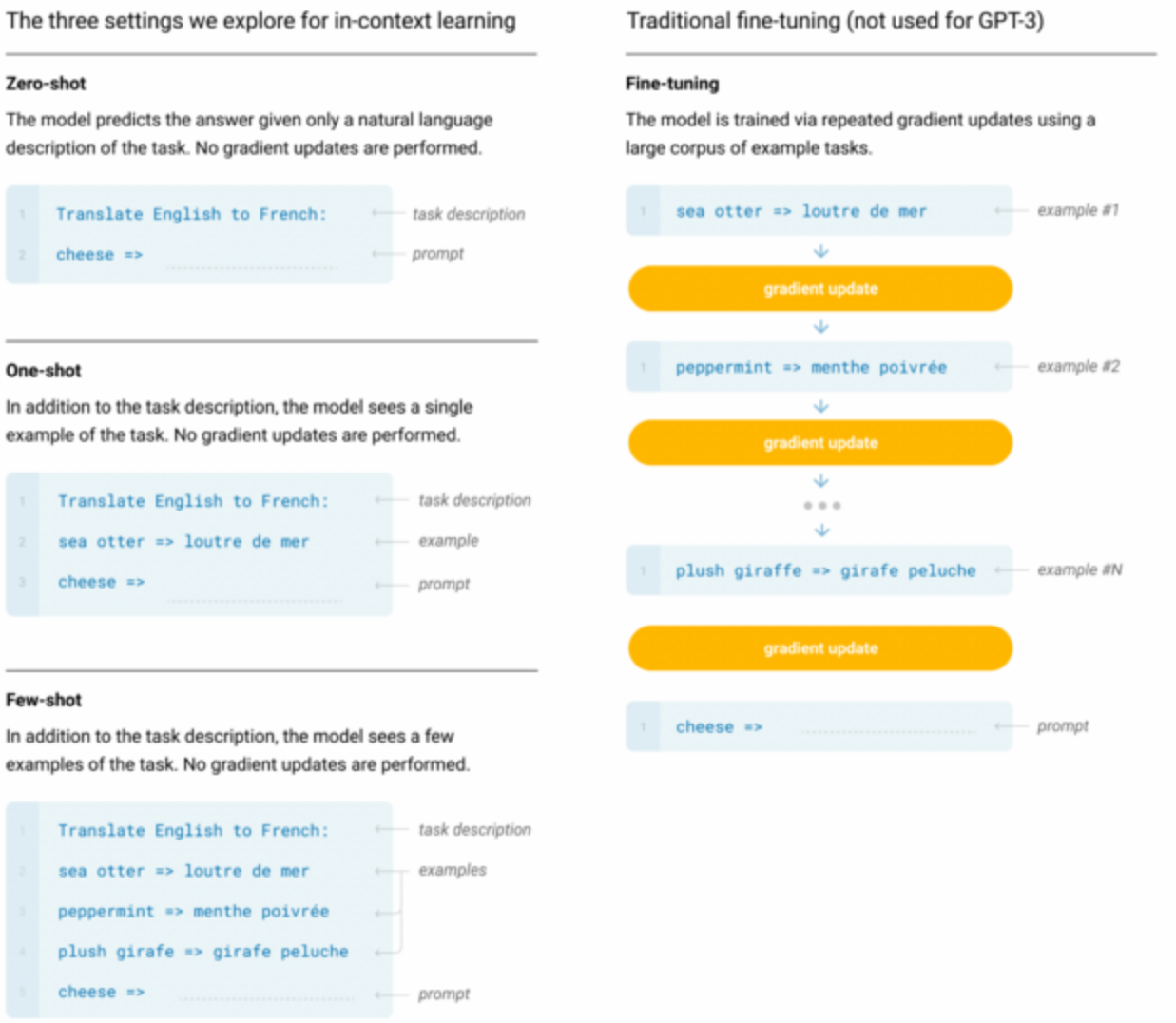


pretraining: next-word-prediction

fine-tuning: task-only OR next-word + task

Radford et al. 2018 (OpenAI)

GPT-3 and Few-Shot "Learning"



Example: GPT-3 Use of Definitions

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:
We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:
One day when I was playing tag with my little sister, she got really excited and she started doing these crazy farduddles.
A "yalubalu" is a type of vegetable that looks like a big pumpkin. An example of a sentence that uses the word yalubalu is:
I was on a trip to Africa and I tried this yalubalu vegetable that was grown in a garden there. It was delicious.

How? Sources of Translations in large Corpora

”I’m not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile [I’m not a fool].**
In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: ”**Mentez mentez, il en restera toujours quelque chose**,” which translates as, ”**Lie lie and something will always remain.**”
“I hate the word ‘**perfume**,”’ Burr says. ‘It’s somewhat better in French: ‘**parfum**.’
If listened carefully at 29:55, a conversation can be heard between two guys in French: “**-Comment on fait pour aller de l’autre coté? -Quel autre coté?**”, which means “**- How do you get to the other side? - What side?**”.
If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?
“**Brevet Sans Garantie Du Gouvernement**”, translated to English: “**Patented without government warranty**”.

Table 1. Examples of naturally occurring demonstrations of English to French and French to English translation found throughout the WebText training set.

BERT vs Prompting

Approaches:

- BERT solves novel tasks by training *heads* on top of its embeddings
- GPT-3 solves novel tasks through prompting

Characteristics:

- BERT+trained head gives better results than GPT-3 given the same compute resources
- GPT-3 requires no retraining for novel tasks
- GPT-3 performance no novel tasks is hard to predict

T5 and exT5

- T5 and ExT5 are sequence-to-sequence models
- they are trained both unsupervised and supervised
- task prompting is incorporated into the training data
- supervised training can be done completely in terms of natural language

ExT5

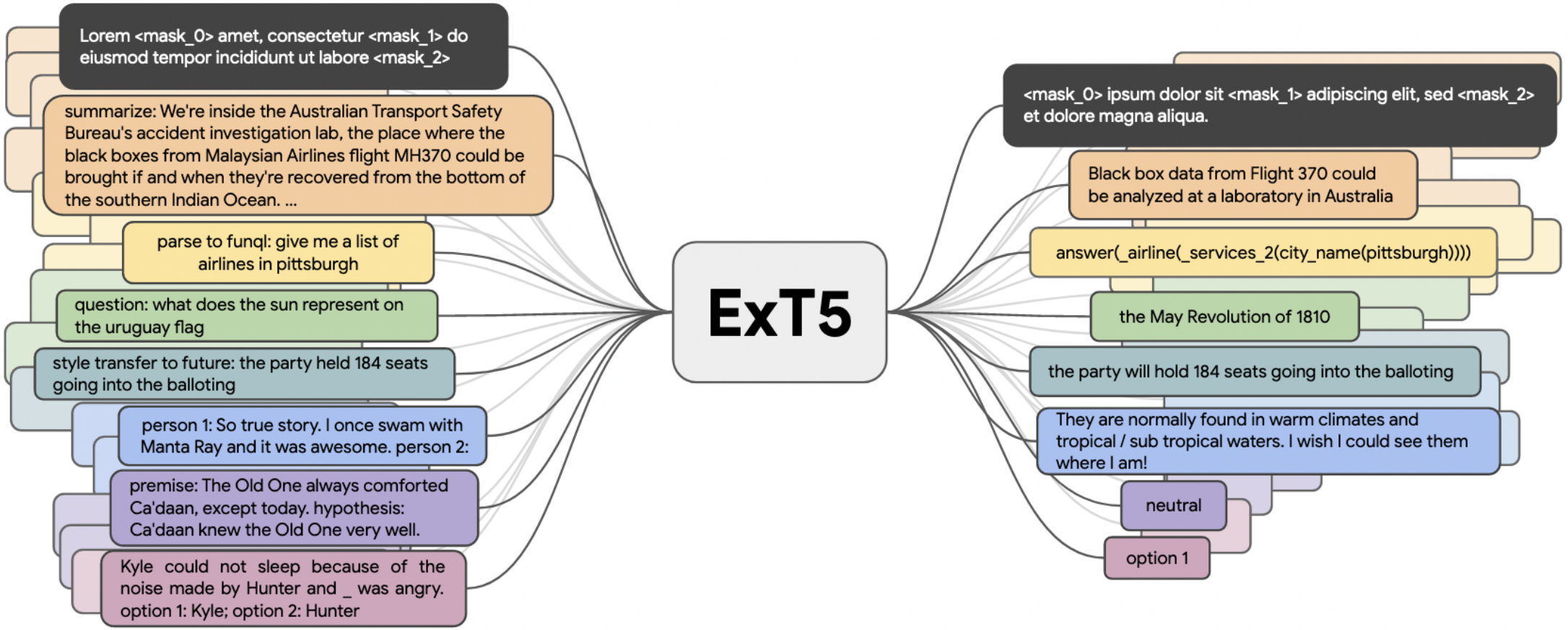


Figure 1: ExT5 pre-trains on self-supervised C4 and supervised ExMix: a massive multi-task mixture.

GENERAL DISCUSSION

Perplexity of GPT-3

model	perplexity
unigram	962
bigram	170
trigram	109
GPT-3	20

This low perplexity appears to require understanding more and more context / semantics.

Unsupervised / Self-Supervised Sequence Learning

Why does it work so well?

- large number of choices, most of them wrong = many bits of unsupervised information from each word prediction
  - c.f. using language models as speech/OCR ground truth
- captures semantic information because it helps reduce perplexity

Sequence Learning vs Intelligence

Large scale semi-supervised training of language models are NOT reasonable models of human cognition:

- humans acquire syntax + semantics with exposure to less than 200M words
- humans derive meaning of language from semantic world models
- semi-supervised language models = Chinese room experiment

However:

- deep language models statistically combined with other sensory modalities might be!

Take-Home Messages

- transformer models + GPUs + huge amounts of text now allow us to extract high quality syntactic and semantic information
- most of the traditional work on syntax, semantics, knowledge representation, etc. has been made practically obsolete by such models
- but... these models are largely black boxes
- for solving novel problems, we have a choice between...
  - prompt engineering for pre-trained models
  - fine-tuning by traditional transfer learning
  - fine-tuning by text-to-text training

SEMI-SUPERVISED LEARNING IN VISION

Already covered...

We have already covered (in the OCR case):

- pseudo-labels
- active learning
- data augmentation and prior knowledge of invariances
- EM algorithms

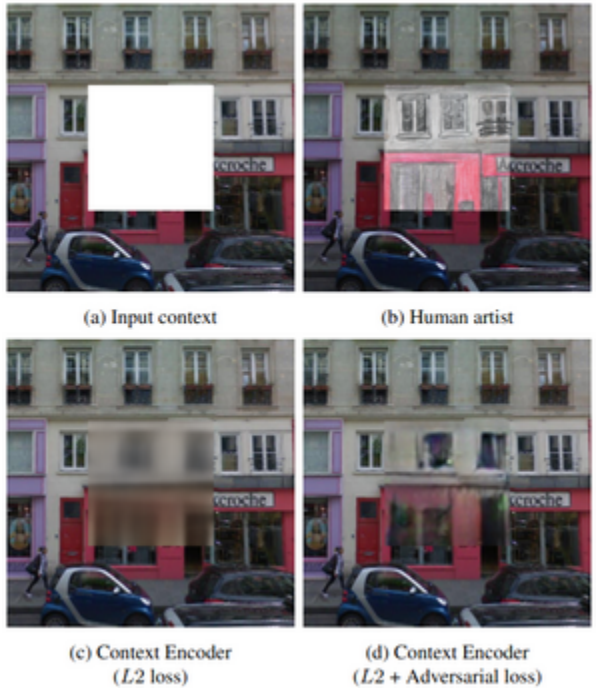
Pre-Transformer Approaches

Prior attempts to carry over principles for language models and HMMs to the image domain:

- linearize images and apply HMM or LSTM models
- apply VQ to patches and apply syntactic models to the resulting "visual words"
- corrupt images with noise and training a network to restore them
- predict color images from grayscale
- mask parts of images and predict the masked parts (like BERT)
- determine the spatial relations between patches (like entailment)

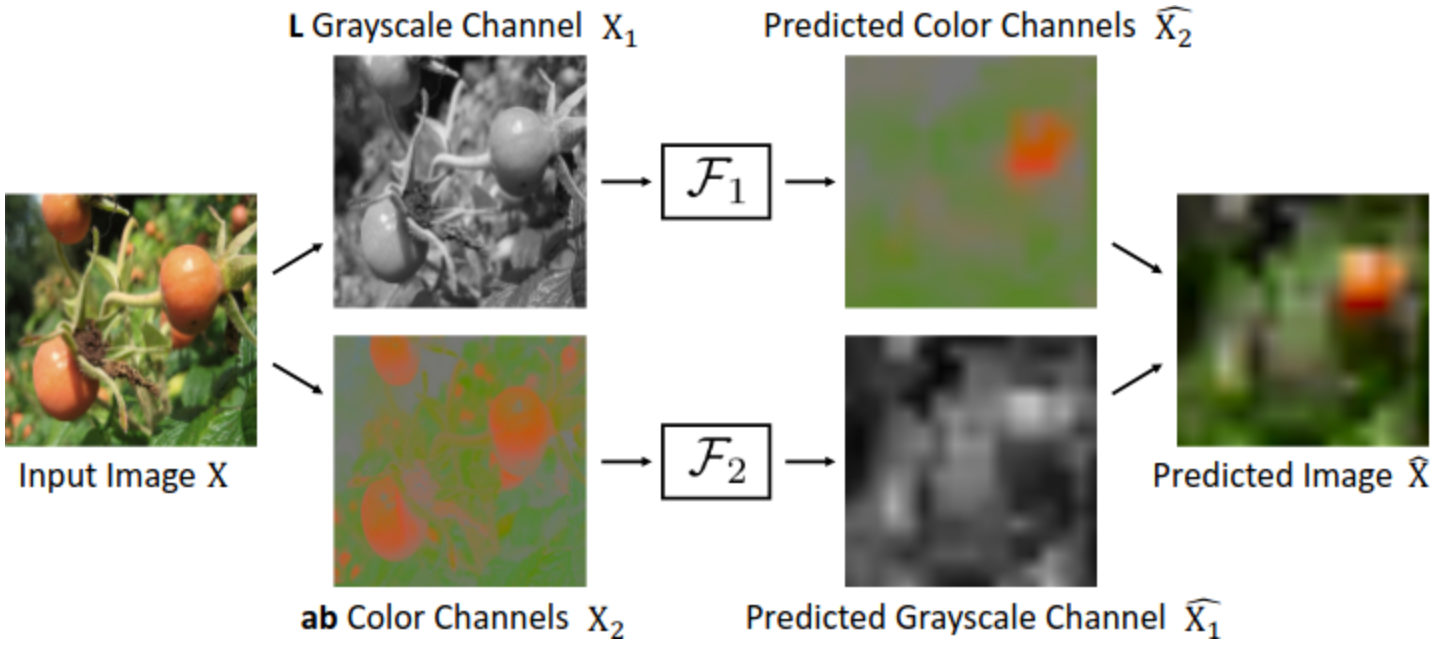
All of these yield deep learning architectures that are potentially useful for transfer learning, but never beat supervised models.

Masked Predictions



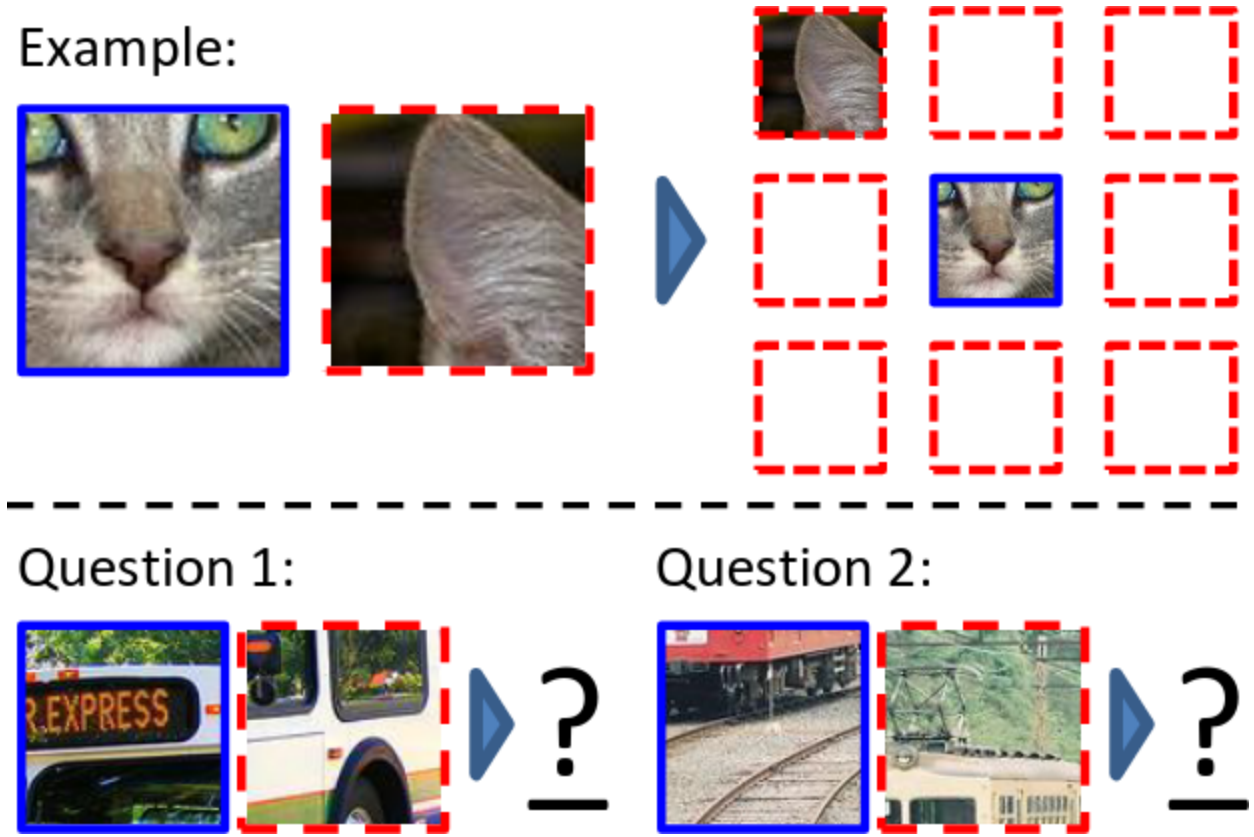
Pathak et al. 2016

Split-Brain Autoencoder



Zhang et al. 2016

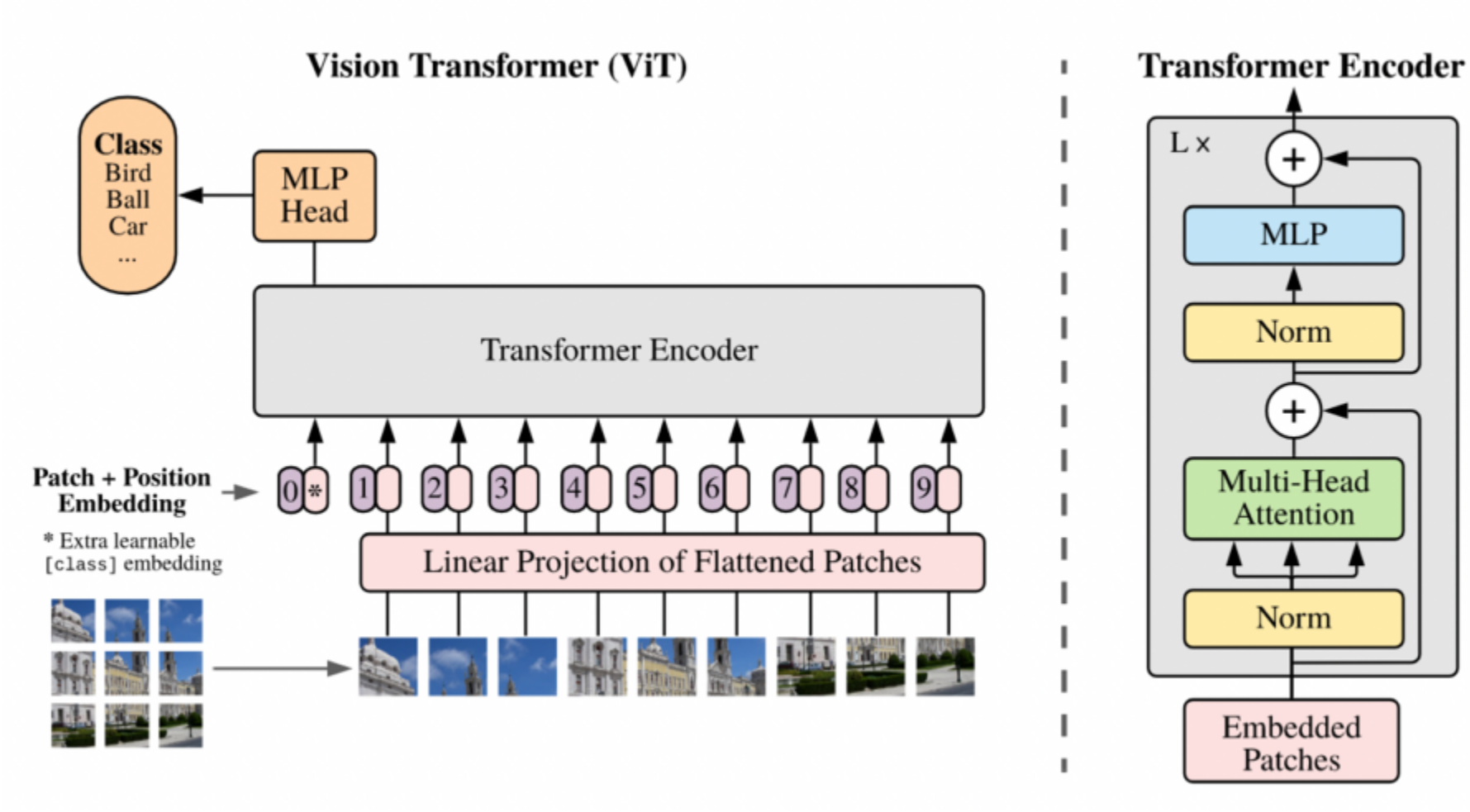
Context Encoding



Doersch et al. 2016

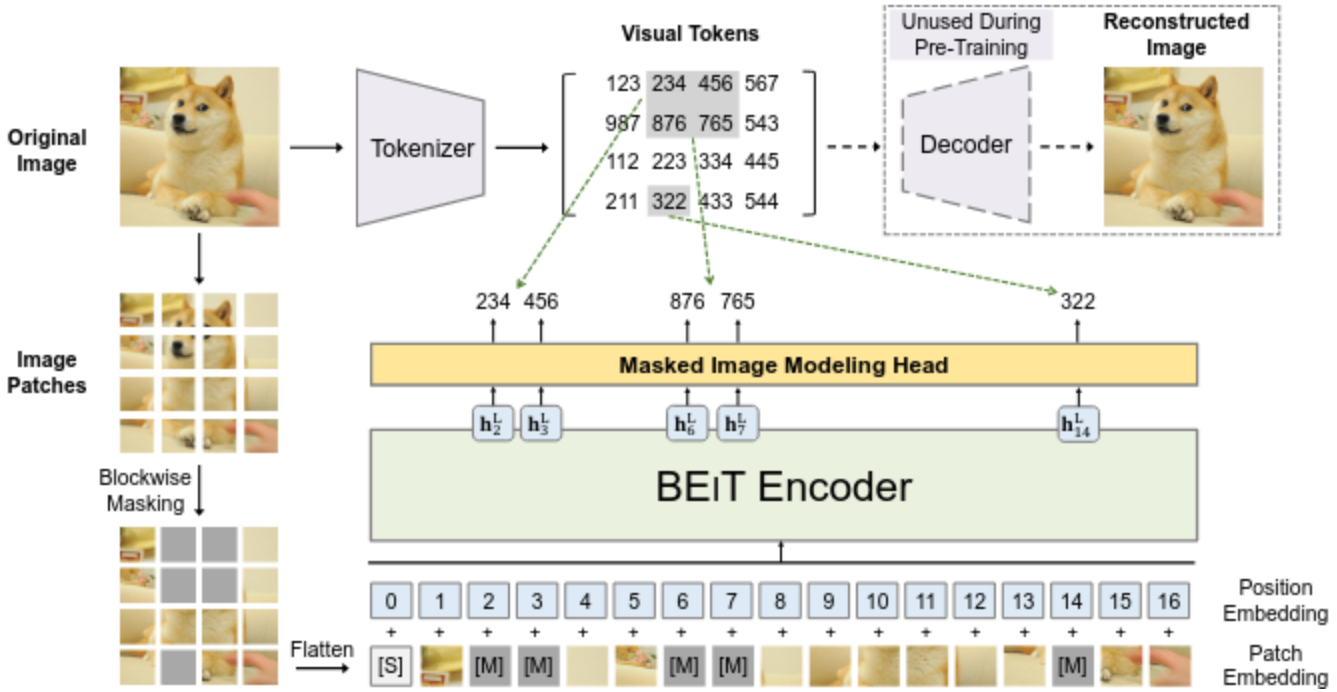
VISION TRANSFORMERS

Vision Transformers



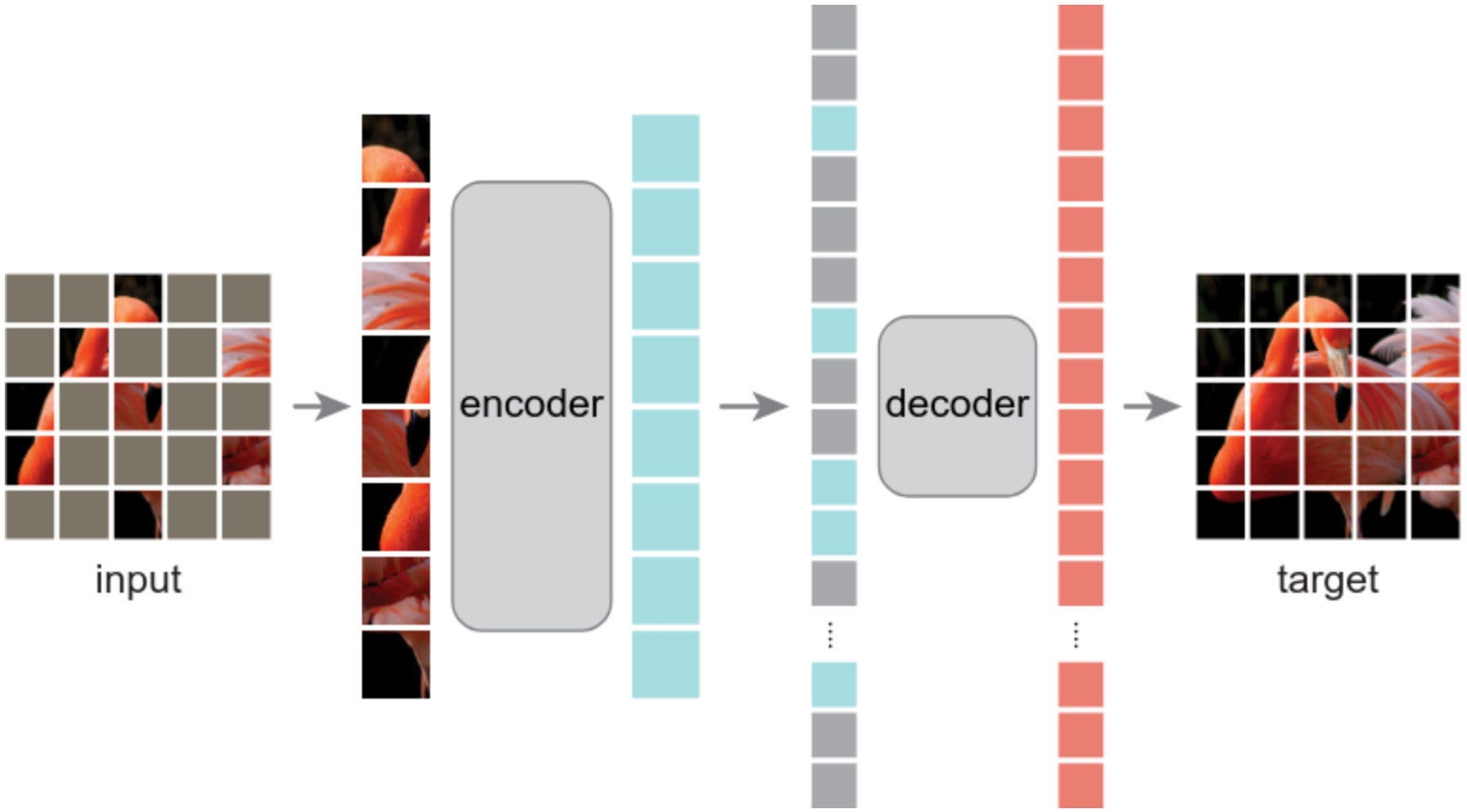
The NLP transformer architecture carries over directly to images: just change the positional embedding. Dosovitskiy et al. 2020, arXiv:2010.11929

BEiT - BERT Pre-Training of Image Transformers



BERT-like pretraining carries over directly. Bao et al., 2022

Masked Autoencoder (MAE)



MAE uses a simpler architecture and no tokenization. He et al. 2021

Masked Autoencoder - Reconstructions

He et al. 2021

Masked Autoencoder - Transfer Learning

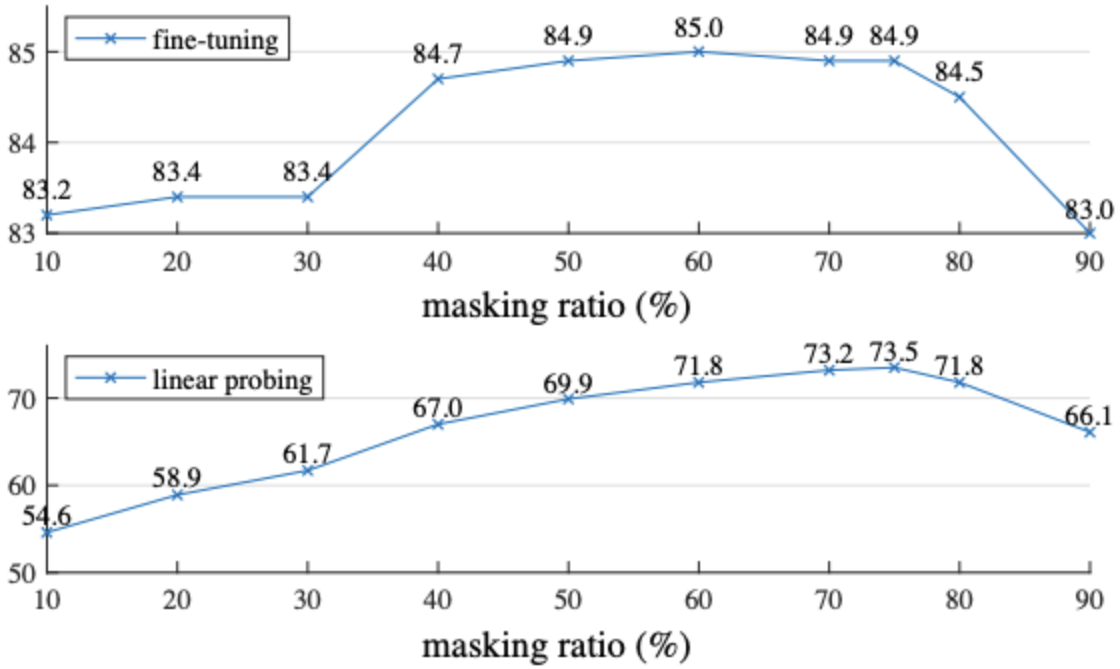


Figure 5. **Masking ratio**. A high masking ratio (75%) works well for both fine-tuning (top) and linear probing (bottom). The y-axes are ImageNet-1K validation accuracy (%) in all plots in this paper.

Unsupervised Training with Transformers

Transformer architectures make BERT-like masking useful unsupervised pre-training for image-related tasks.

OTHER APPROACHES

SimCLR - Contrastive Learning

Basic idea:

- generate two differently augmented versions of the same image
- train a representation that is as similar as possible for the same image, different for different images

Details:

- carefully choose augmentations, watch out for trivial solutions (e.g. color)
- separate representation from scoring
- compute "softmax over cosine similarity" over very large batches
- implemented with ResNet

SimCLR - Contrastive Learning

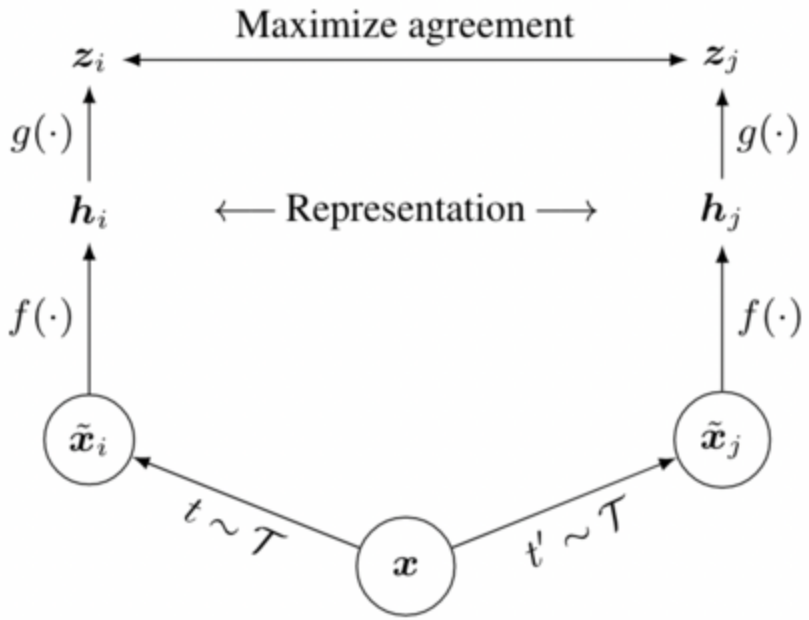


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder 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.

Chen et al., 2020, arXiv:2002.05709

SimCLR - Augmentations

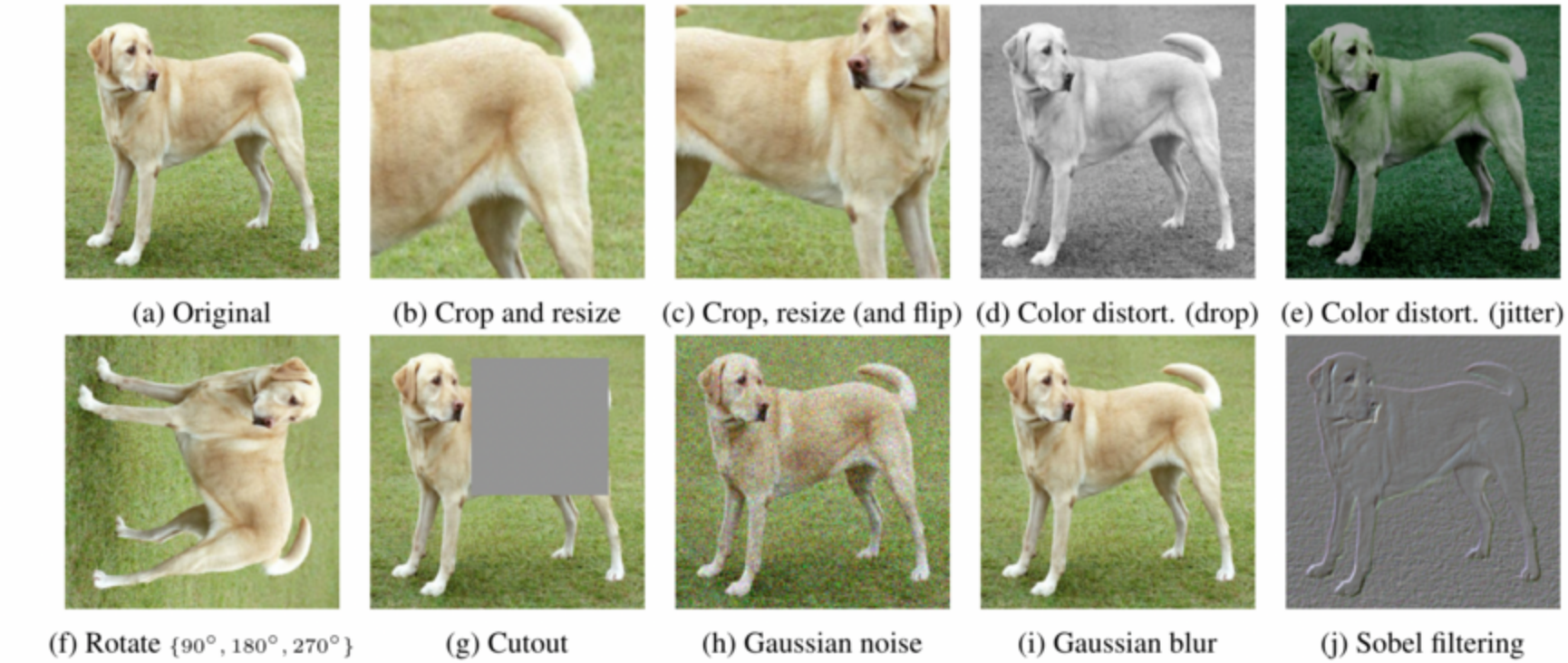


Figure 4. Illustrations of the studied data augmentation operators. Each augmentation can transform data stochastically with some internal parameters (e.g. rotation degree, noise level). Note that we *only* test these operators in ablation, the *augmentation policy used to train our models* only includes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

SimCLR - Transfer Learning Performance

	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 $(4\times)$ 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.

DINO

- self-DIstillation with NO labels
- discovers labels / class structure by itself
- unsupervised representation learning for images
- impressive semantic segmentation results
- attention map = segmentation map
- vision transformer or ResNet 50 based

DINO Architecture

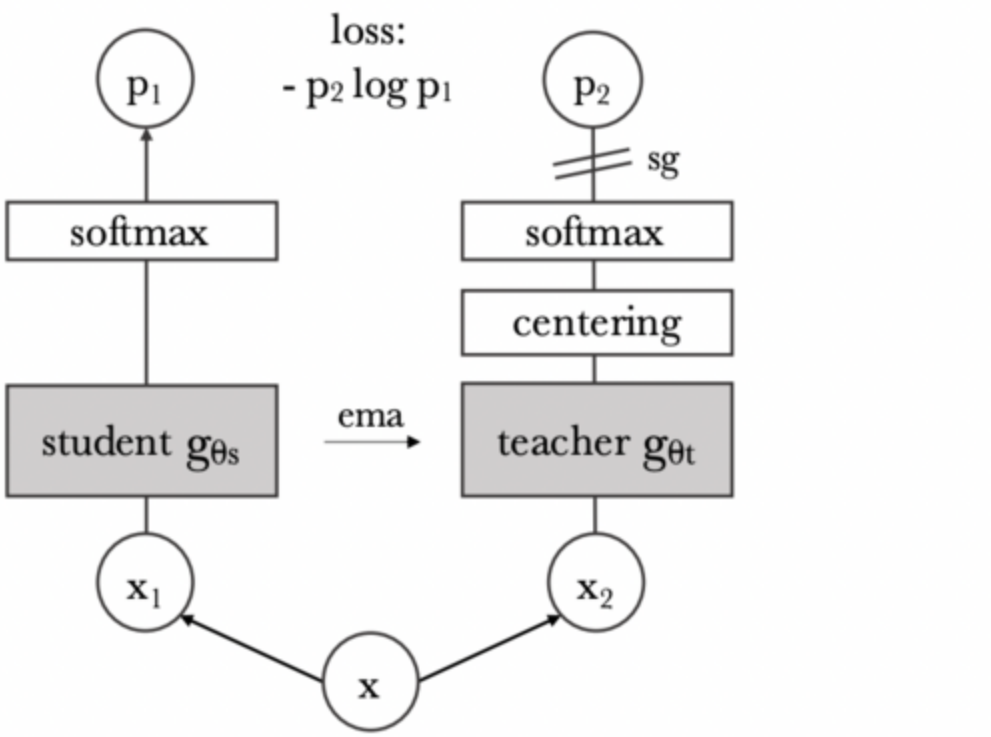
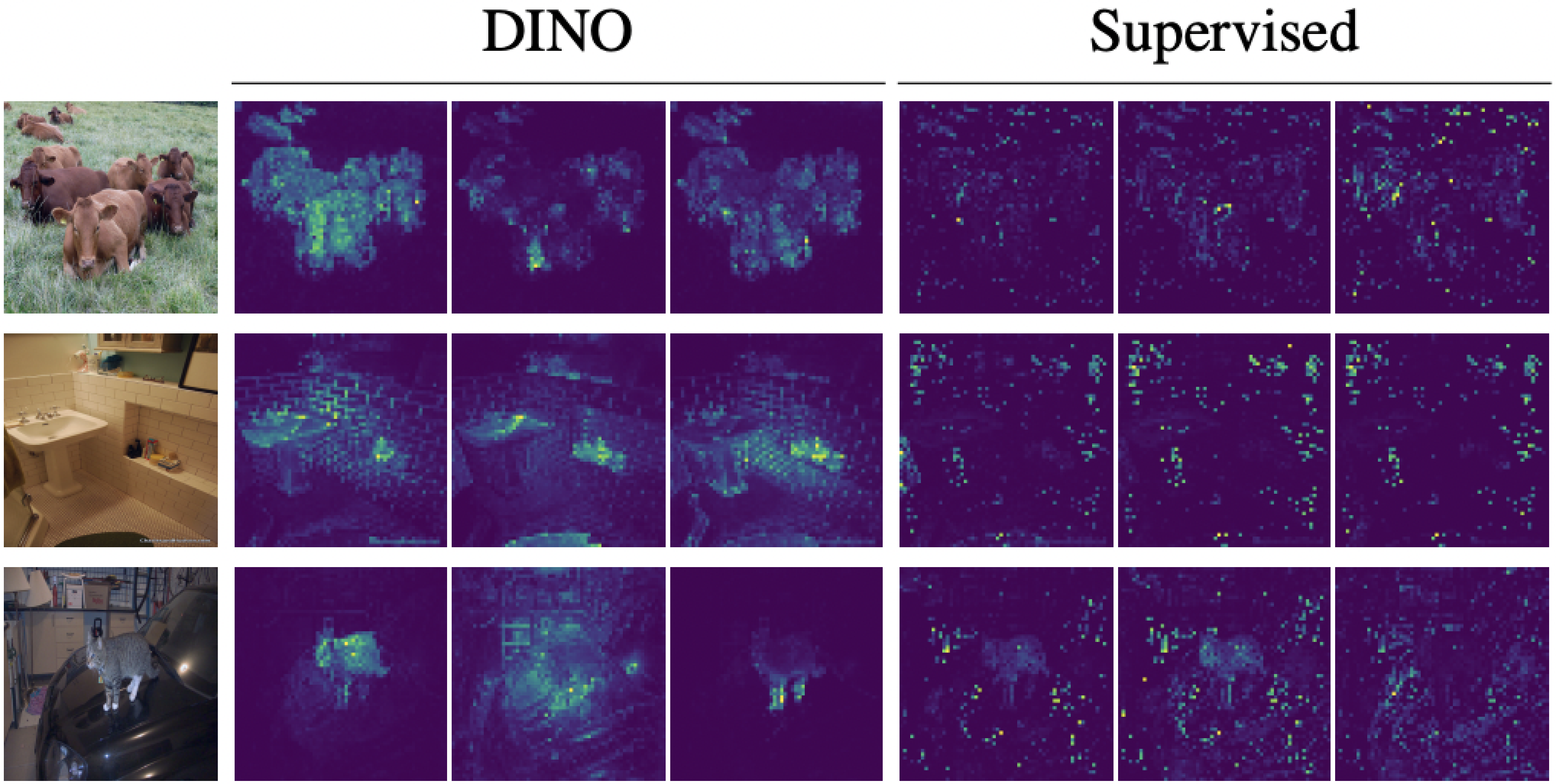


Figure 2: **Self-distillation with no labels.** We illustrate DINO in the case of one single pair of views (x_1, x_2) for simplicity. The

DINO Results

Method	Arch.	Param.	im/s	Linear	k-NN
Supervised	RN50	23	1237	79.3	79.3
SCLR [12]	RN50	23	1237	69.1	60.7
MoCov2 [15]	RN50	23	1237	71.1	61.9
InfoMin [67]	RN50	23	1237	73.0	65.3
BarlowT [81]	RN50	23	1237	73.2	66.0
OBoW [27]	RN50	23	1237	73.8	61.9
BYOL [30]	RN50	23	1237	74.4	64.8
DCv2 [10]	RN50	23	1237	75.2	67.1
SwAV [10]	RN50	23	1237	**75.3**	65.7
DINO	RN50	23	1237	**75.3**	**67.5**
Supervised	ViT-S	21	1007	79.8	79.8
BYOL* [30]	ViT-S	21	1007	71.4	66.6
MoCov2* [15]	ViT-S	21	1007	72.7	64.4
SwAV* [10]	ViT-S	21	1007	73.5	66.3
DINO	ViT-S	21	1007	**77.0**	**74.5**

DINO Segmentation



- semantic segmentation by attention looks better than from supervised models

Summary: SimCLR and DINO

High Level:

- SimCLR is a kind of *representation* or *metric* learning
- DINO is a kind of *clustering*
- implementations are complex and with lots of hyperparameters

Conclusions:

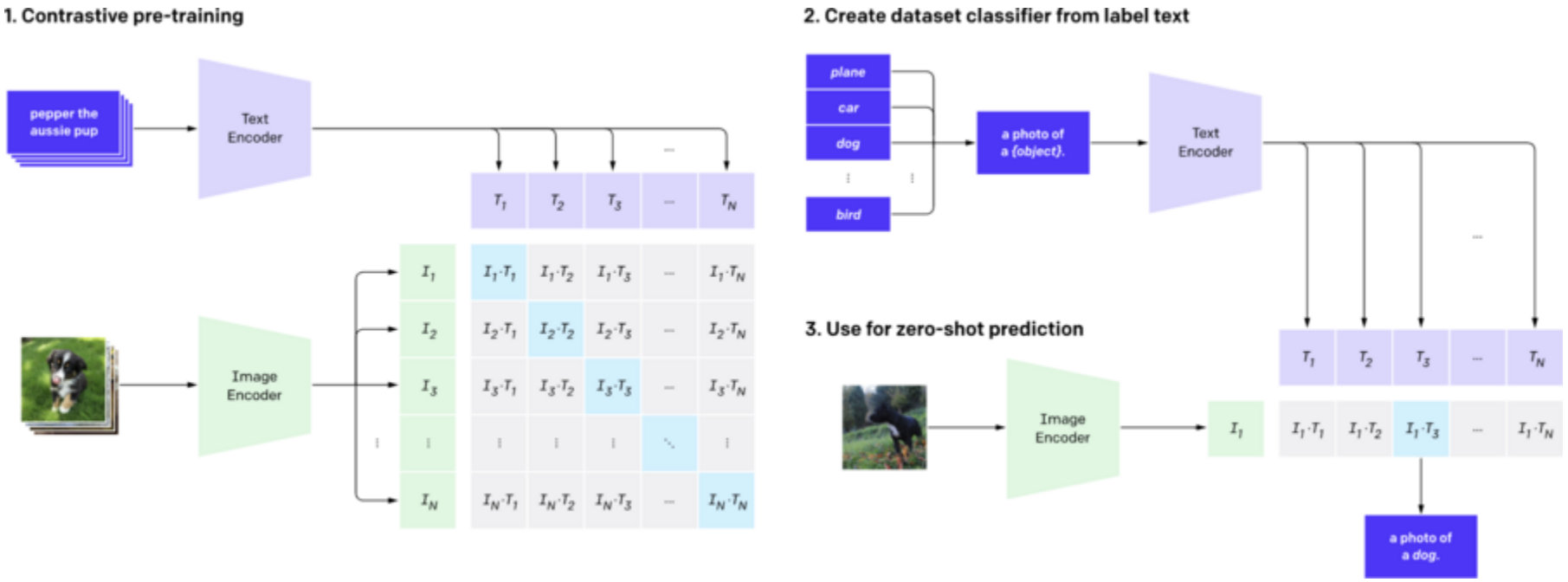
- tasks like these may be most useful as additional tasks combined with masking (just like BERT)

COMBINING TEXT AND IMAGE MODELS

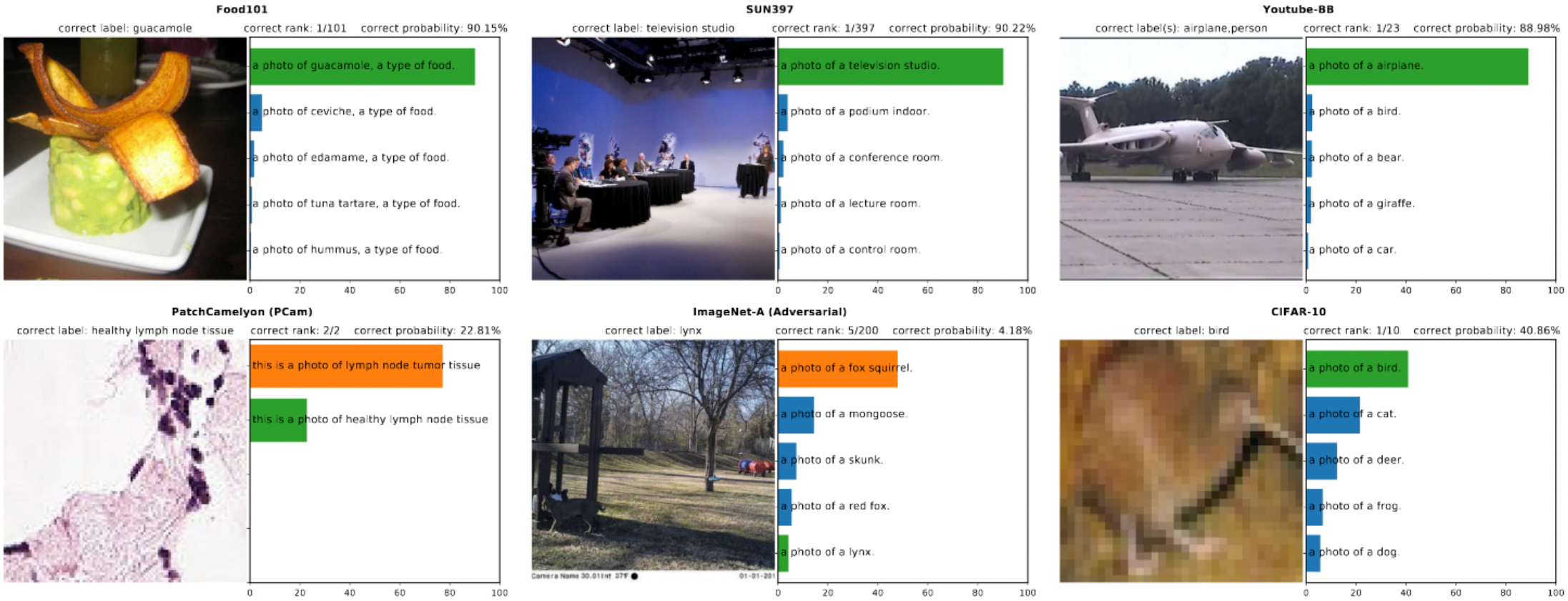
Combining Image and Text Models

- GPT-3, ExT5, etc. show how natural language models can be used for zero short learning
- CLIP
  - use natural language supervision for image recognition (weak supervision)
  - vision: transformer or ResNet, language: transformer
  - permit "prompt engineering" to allow different kinds of NLP tasks
  - uses contrastive pretraining (rather than, say, captioning)

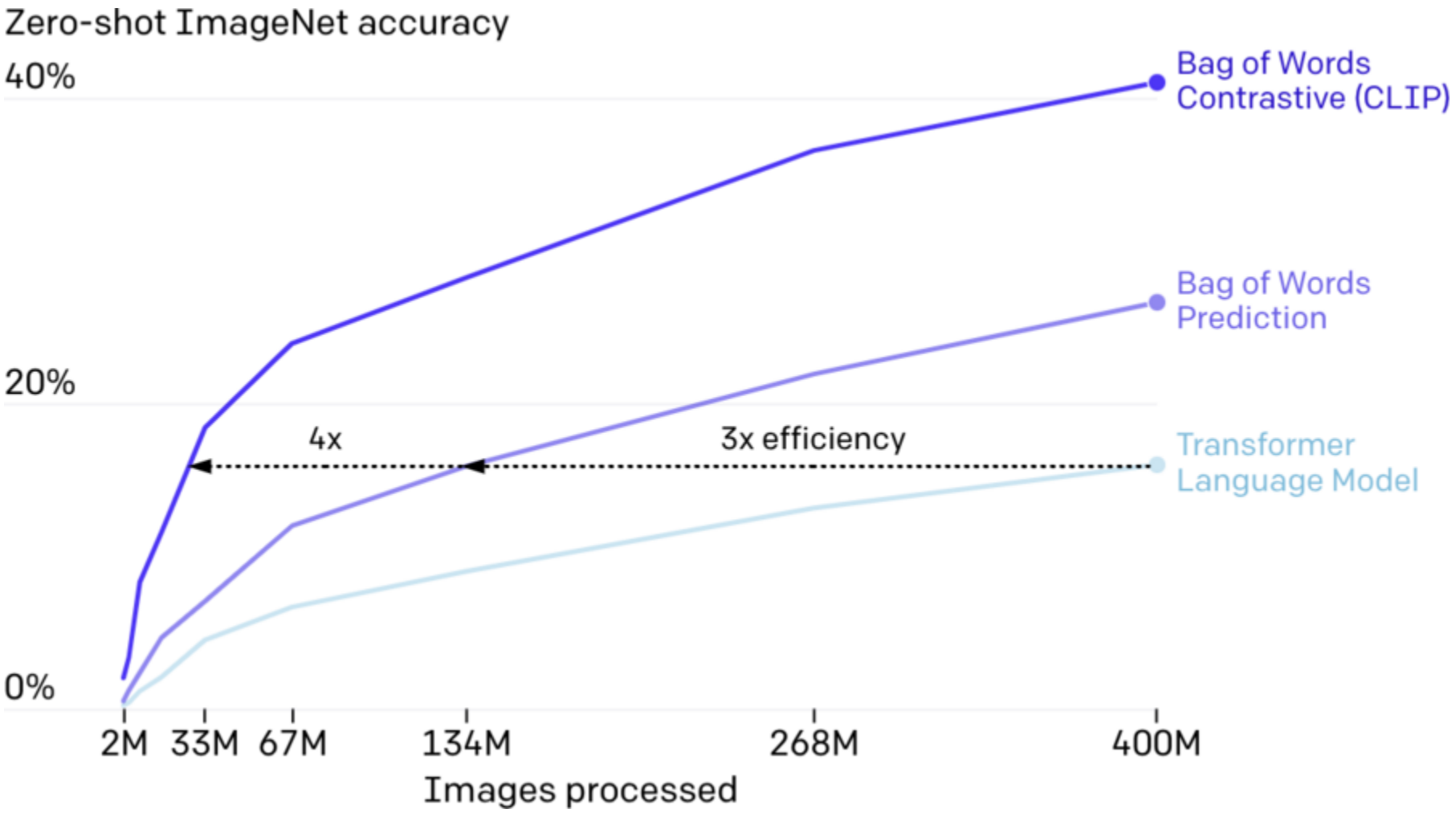
CLIP Architecture



CLIP Results



CLIP Results



Discussion

Current and future directions:

- combining text and image
- integrating unsupervised pretrained vision and language models
- integrating video and audio and identifying good self-supervised tasks for these (e.g., VideoMAE, Audio-MAE)
- cross-modal combinations likely also reduce the amount of training data required within each modality

GENERATIVE AND DENSITY MODELS

Traditional Approaches

- parametric density estimation
- mixture models
- Bayesian networks
- causal models
- Boltzmann machine
- restricted Boltzmann machine
- Energy Based Models

Generative and Density Models

Important Applications:

- speech generation, game content generation: conditional generation of content based on some inputs
- image and video enhancement: upscaling, sharpening, interpolation, ray tracing, etc.

Largely Unfulfilled Potential:

- unsupervised learning for transfer learning
- training dataset generation and augmentation
- domain transfer (winter->summer, etc.)

FLOW MODELS

Flow Models Idea

- model some complicated $p(\xi)$
- given $x \sim p(x) = \mathcal{N}(0, \mathbf{1})$
- learn an invertible function $x = f_\theta(\xi)$

Flow Models - Mapping Densities

Example:

- linear mapping of a standard normal density $\mathcal{N}(0, 1)$ to a general normal density $\mathcal{N}(\mu, \Sigma)$
- requires changing the normalization factor b/c linear map is not volume preserving

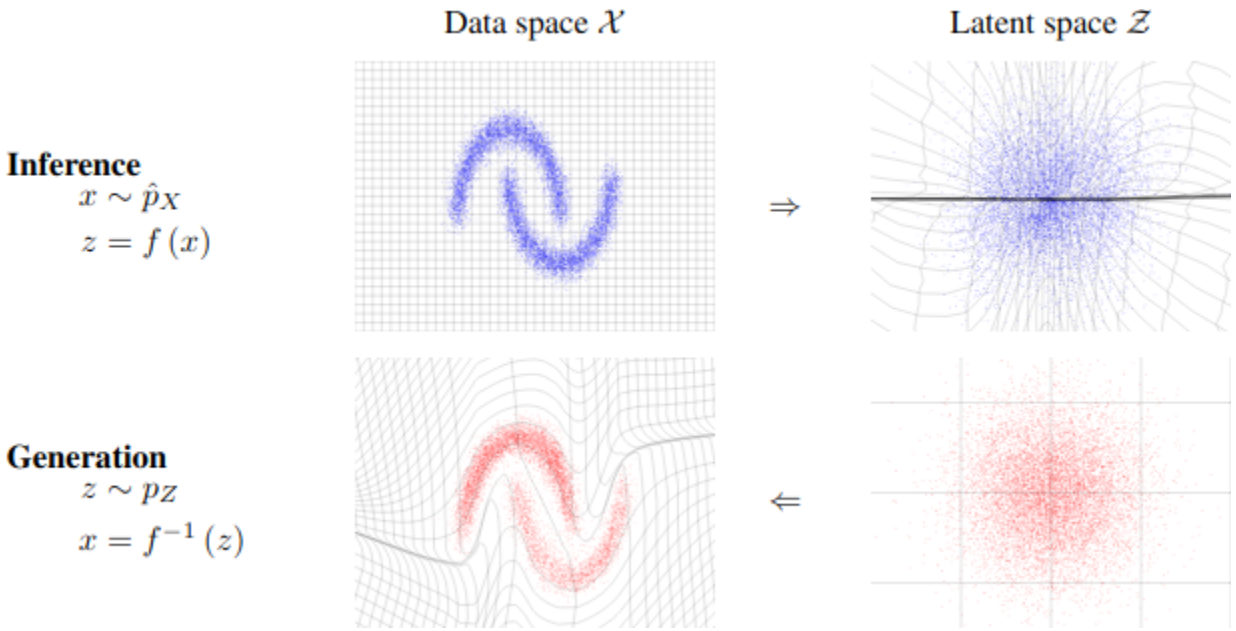
Generalization:

- the local linear relationship between x and ξ is given by the Jacobian J_{f_θ}
- we account for the volume change using the determinant of the Jacobian
- $p(x) = p(\xi) \cdot \det(J_{f_\theta}(\xi))$

Updating the loss function:

- $\arg\min_\theta \mathbb{E}_x[-\log p(x)] = \arg\min_\theta \mathbb{E}_x[-\log p(f_\theta(x)) - \log\det(J_{f_\theta}(\xi))]$

Flow Models - Illustration

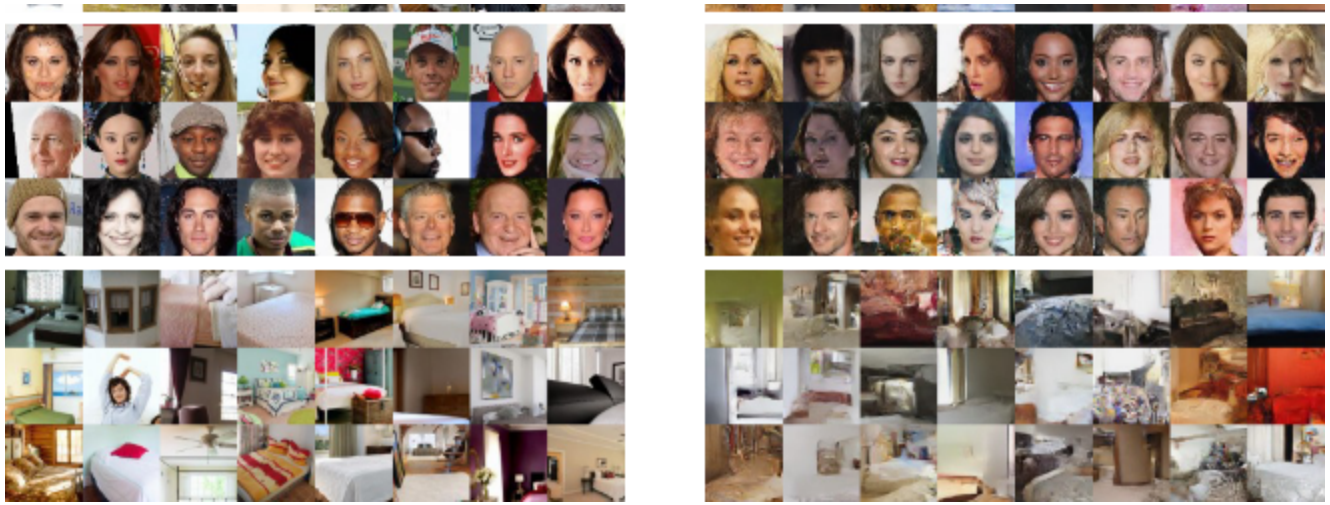


Dinh et al. ICLR 2017

Flow Models - Technicalities

- mappings compose: if $f = g \circ h$
  - then $p(x) = p(\xi) \cdot \det(J_g(h(\xi))) \det(J_h(\xi))$
- regular linear layers are too costly to invert, use affine coupling layers instead (NICE/RealNVP)
  - split input to layer in two halves $x = (x_l, x_h)$
  - $y_l = x_l$
  - $y_h = x_h \odot f_\theta(x_l) + g_\theta(x_l)$ (elementwise affine)
  - the Jacobian is triangular, so its determinant is the product of the diagonal elements
- need to "dequantize" images
- other options: mixtures of logistics, continuous time flows, ...

Flow Models - Examples



Dinh et al. ICLR 2017

VAE

Variational Autoencoder

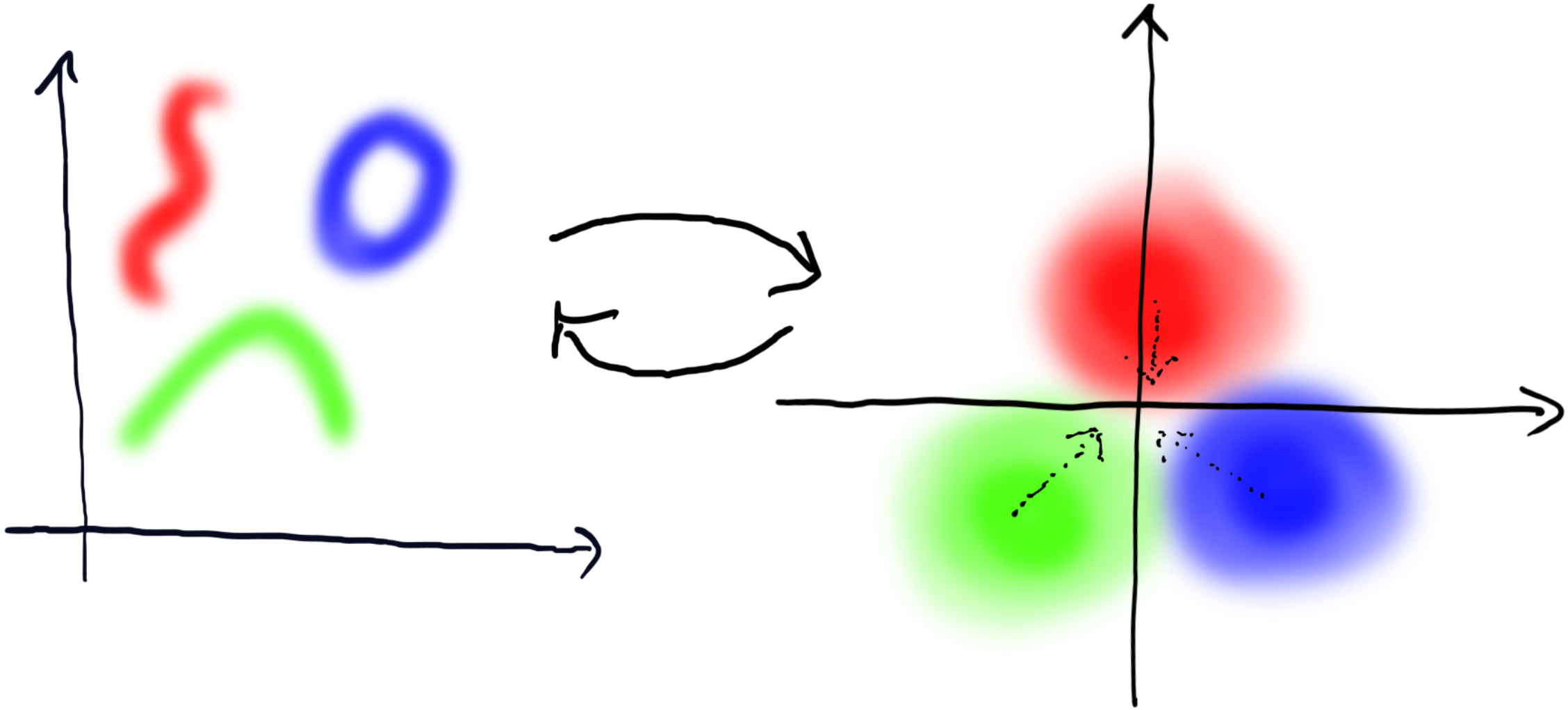
Problem using Autoencoders for Generative Models:

- tends not to generate useful latent space representations
- small changes in latent space not guaranteed to correspond to small changes in output

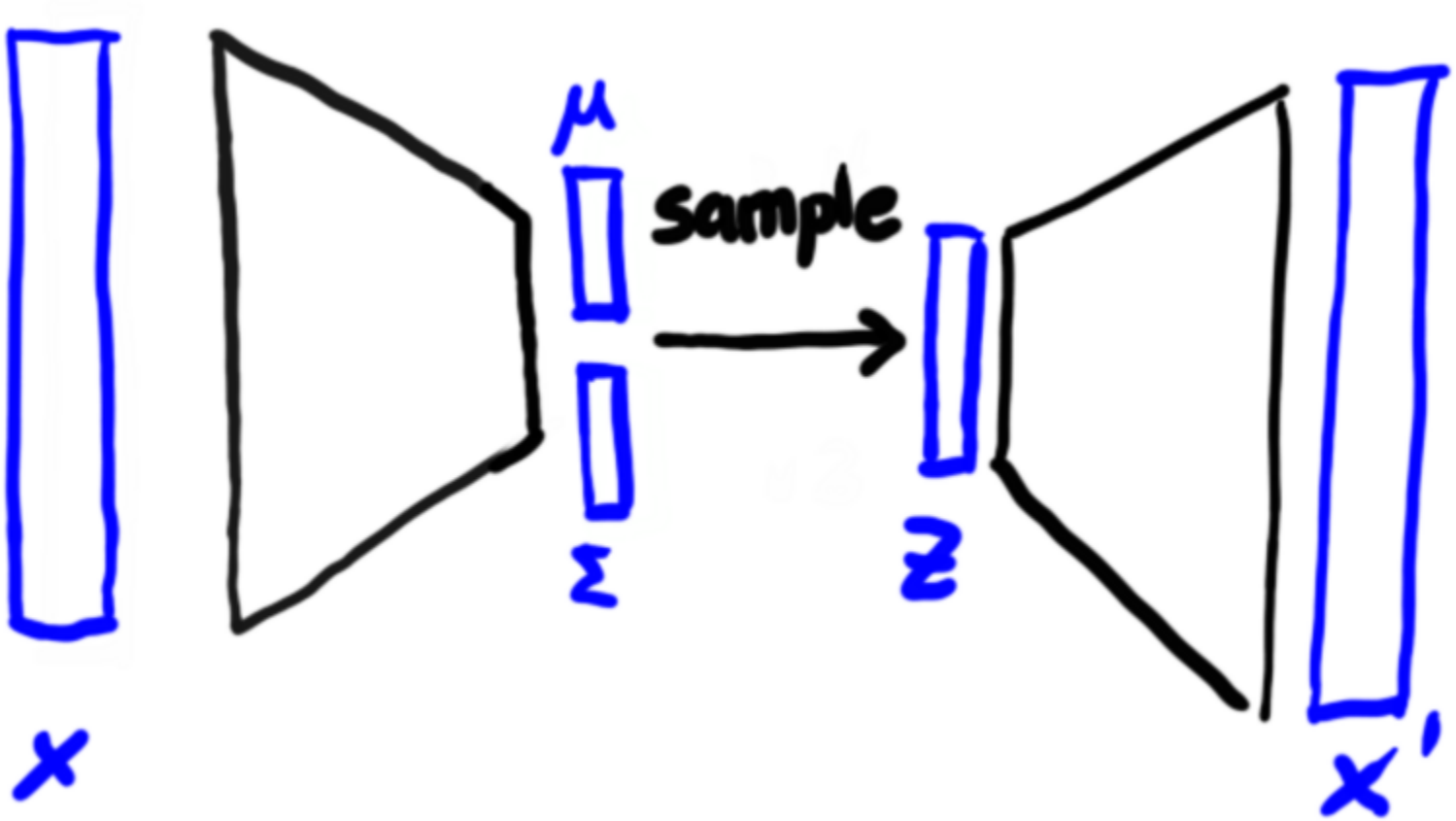
Solution:

- map each input vector x to a conditional distribution over y
- specifically
  - output parameters for a parametric distribution $p_\phi(y|x)$
  - choose $p_\phi(y|x) = \mathcal{N}(y, \mu_x, \Sigma_x)$
  - move $p_\phi(y|x)$ towards $\mathcal{N}(0, \mathbf{1})$
- reconstruct x by sampling from that conditional distribution

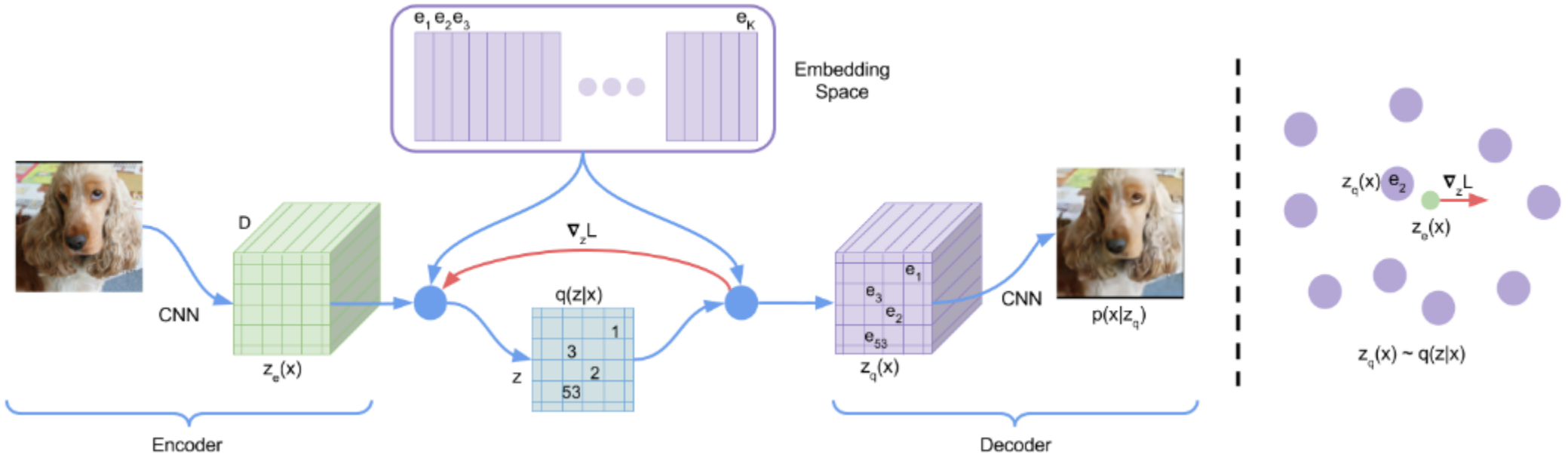
VAE Objective



VAE Structure



VQ-VAE



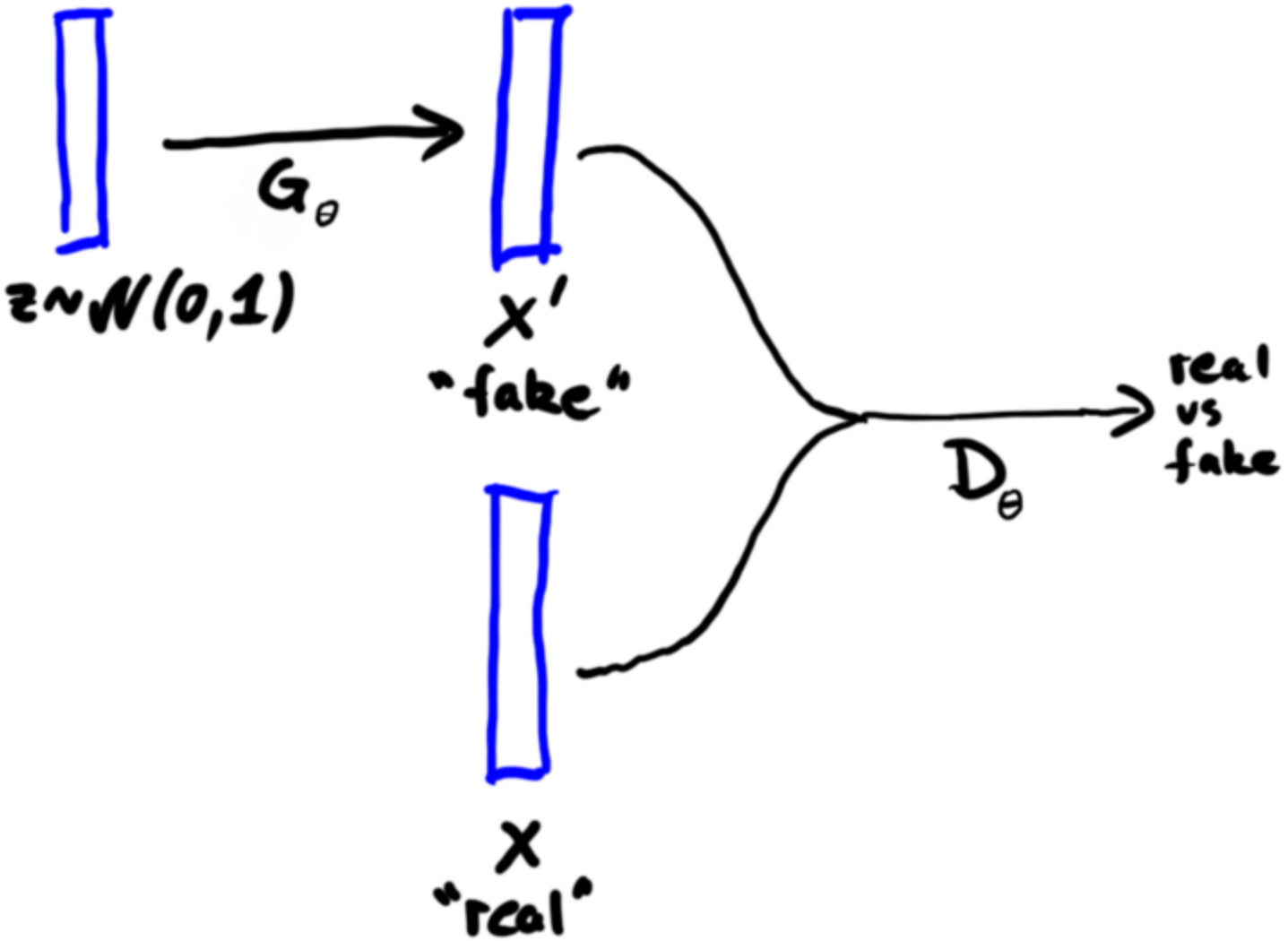
VQ-VAE uses a discrete latent space with a "language model", giving rise to a rather different kind of model; van den Oord et al., Arxiv 2018

GENERATIVE ADVERSARIAL NETWORKS

Generative Adversarial Models

- sampling-only models
- sample from an underlying distribution $z \sim p(z)$
- compute output as $f_\theta(z)$, inducing some $p_\theta(x)$
- for other models, we tried to make $p_\theta(x)$ close to the data
- with GANs, we cannot access $p_\theta(x)$, only sample from it
- a kind of "game" in which a generator tries to fool a discriminator

GAN

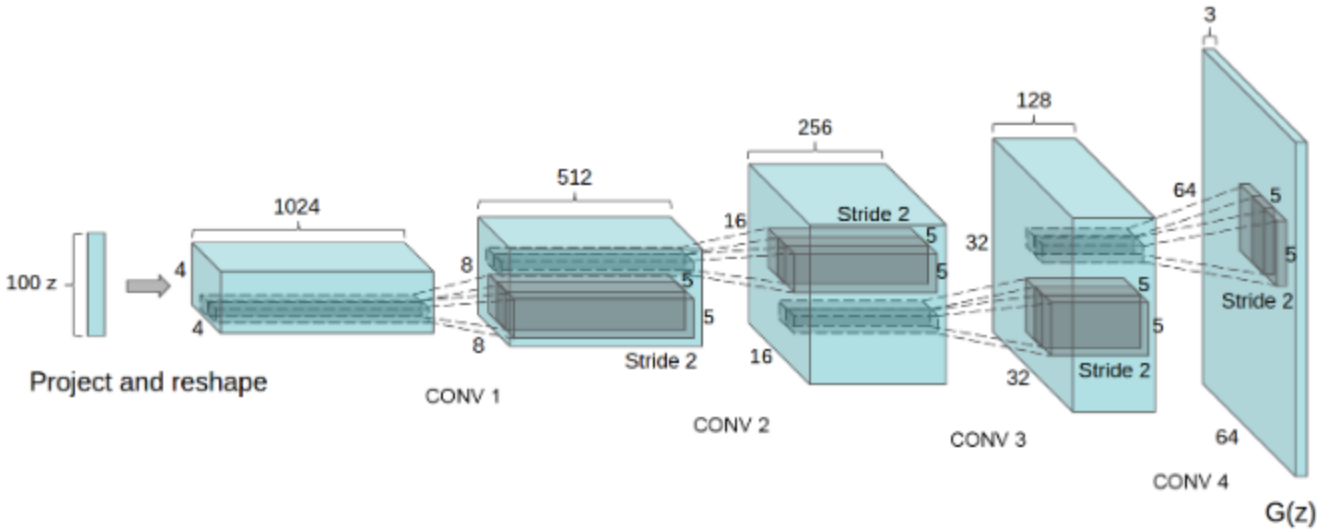


Major Difficulties Implementing GAN

- low signal from a single global discriminator
  - solution: use patchwise/pixel-wise discrimination
- discriminator easily saturates and generates no usable deltas
  - solution: carefully choose discriminator architecture and limited training
- mode collapse: if $p(x)$ is multimodal, model may sample a single mode
  - solution: use Wasserstein GAN (roughly: discriminators w/bounded Lipschitz coefficients)

Original GAN Results

Deep Convolutional GAN (DCGAN)



Radford et al., 2016; Arxiv 1511.06434

DCGAN - Results

Radford et al., 2016; Arxiv 1511.06434

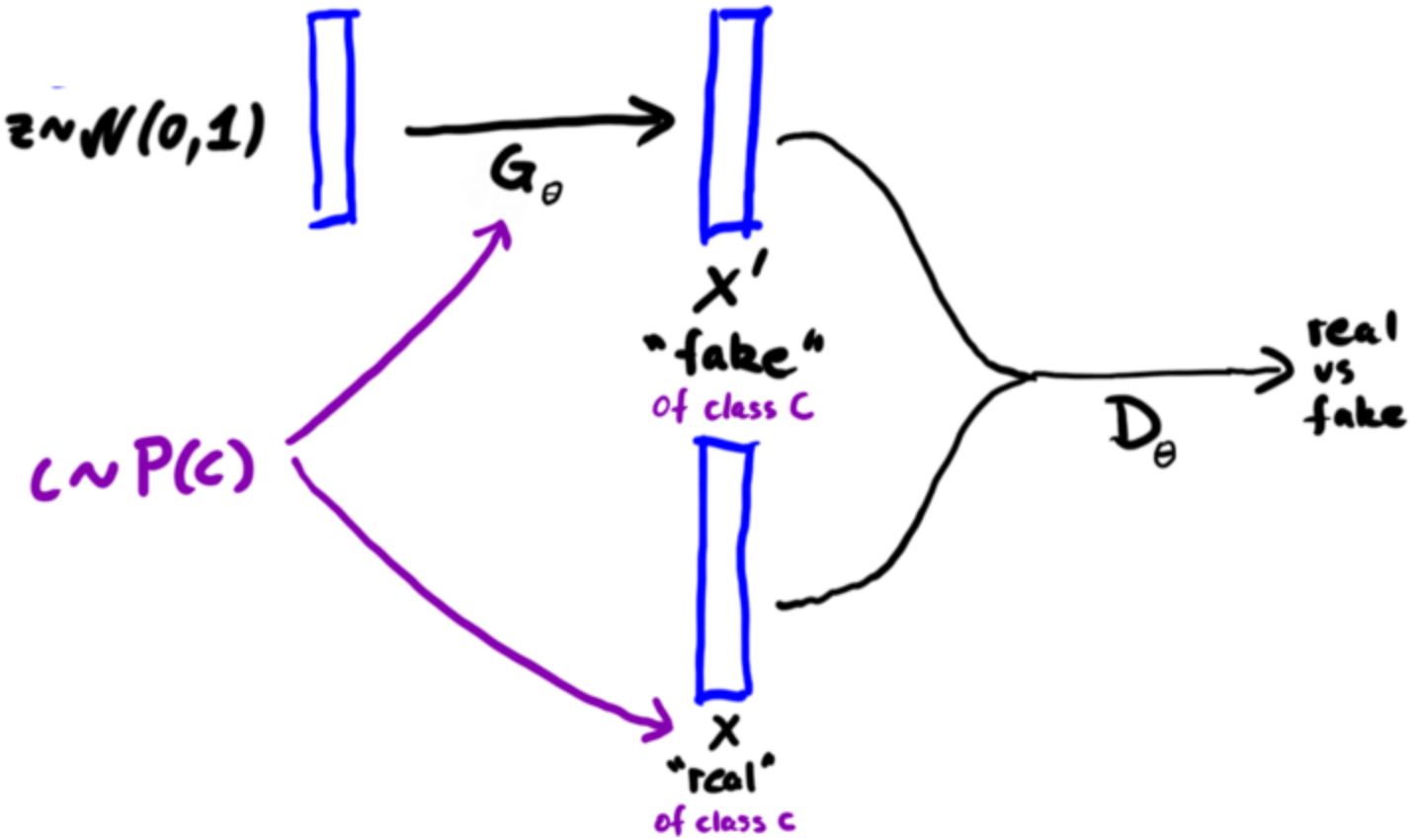
DCGAN - Representation Learning / Features

Table 1: CIFAR-10 classification results using our pre-trained model. Our DCGAN is not pre-trained on CIFAR-10, but on Imagenet-1k, and the features are used to classify CIFAR-10 images.

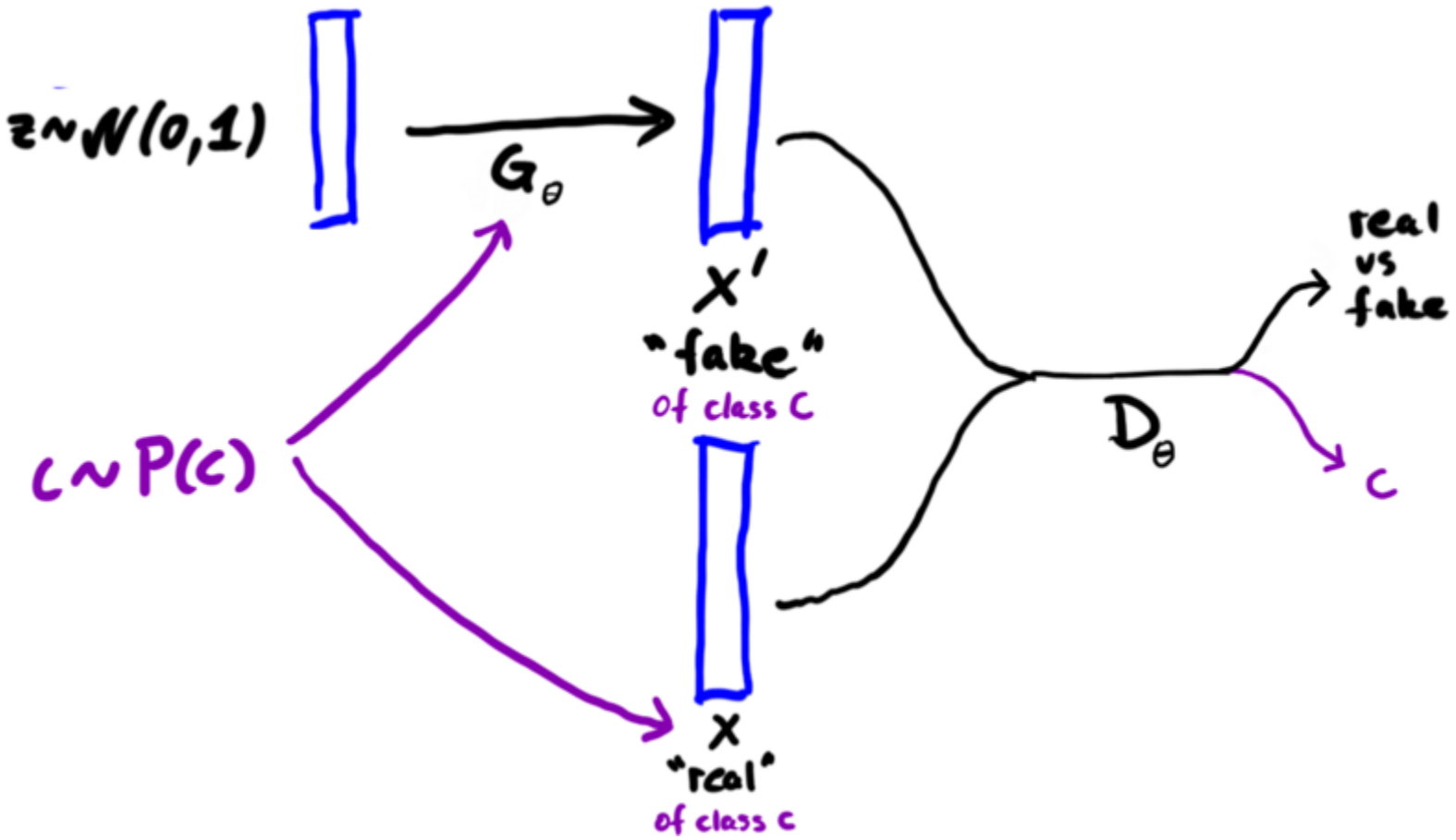
Model	Accuracy	Accuracy (400 per class)	max # of features units
1 Layer K-means	80.6%	63.7% (±0.7%)	4800
3 Layer K-means Learned RF	82.0%	70.7% (±0.7%)	3200
View Invariant K-means	81.9%	72.6% (±0.7%)	6400
Exemplar CNN	84.3%	77.4% (±0.2%)	1024
DCGAN (ours) + L2-SVM	82.8%	73.8% (±0.4%)	512

Radford et al., 2016; Arxiv 1511.06434

Conditional GAN

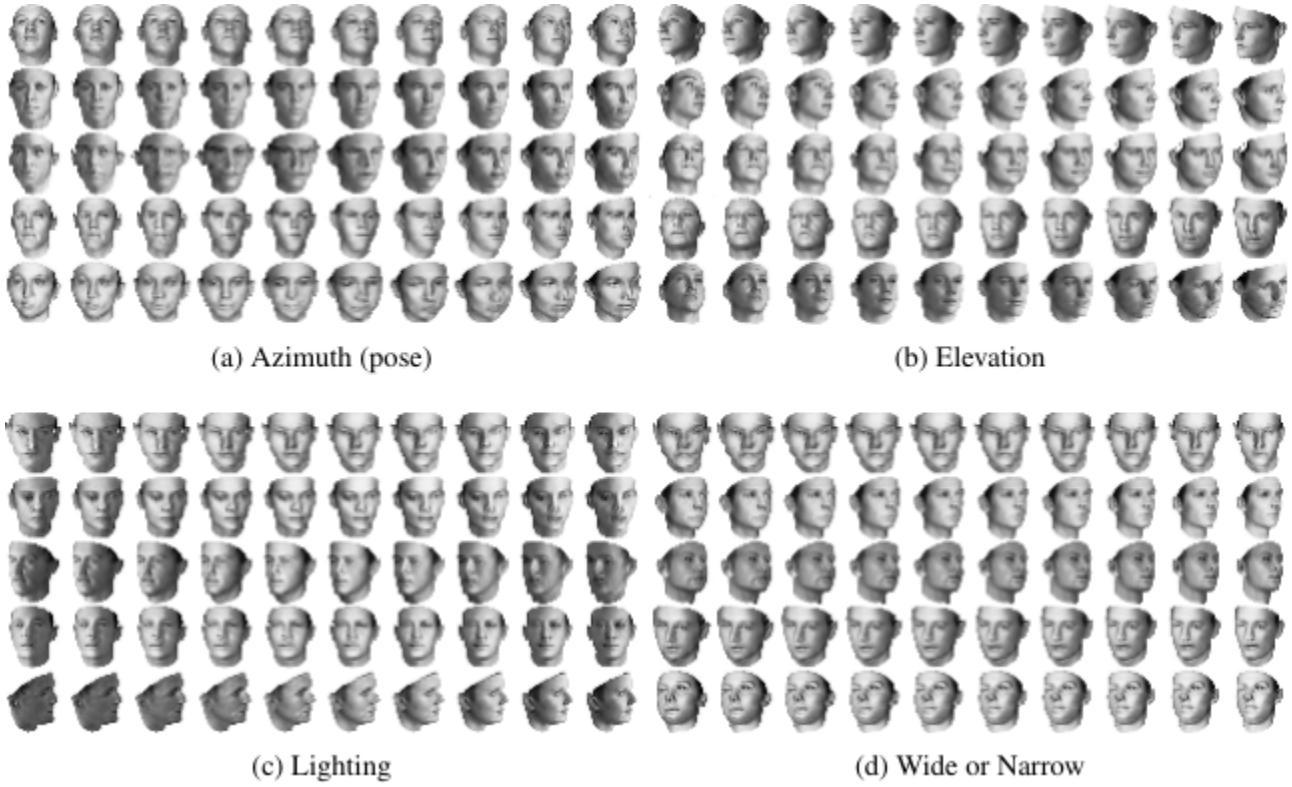


InfoGAN



Chen et al., 2016; arXiv:1606.03657.

InfoGAN - Disentangling Latent Codes



Chen et al., 2016; arXiv:1606.03657.

BigGAN Results

Large bag of tricks. Brock et al. 2018.

DISTRIBUTION ALIGNMENT

Problem

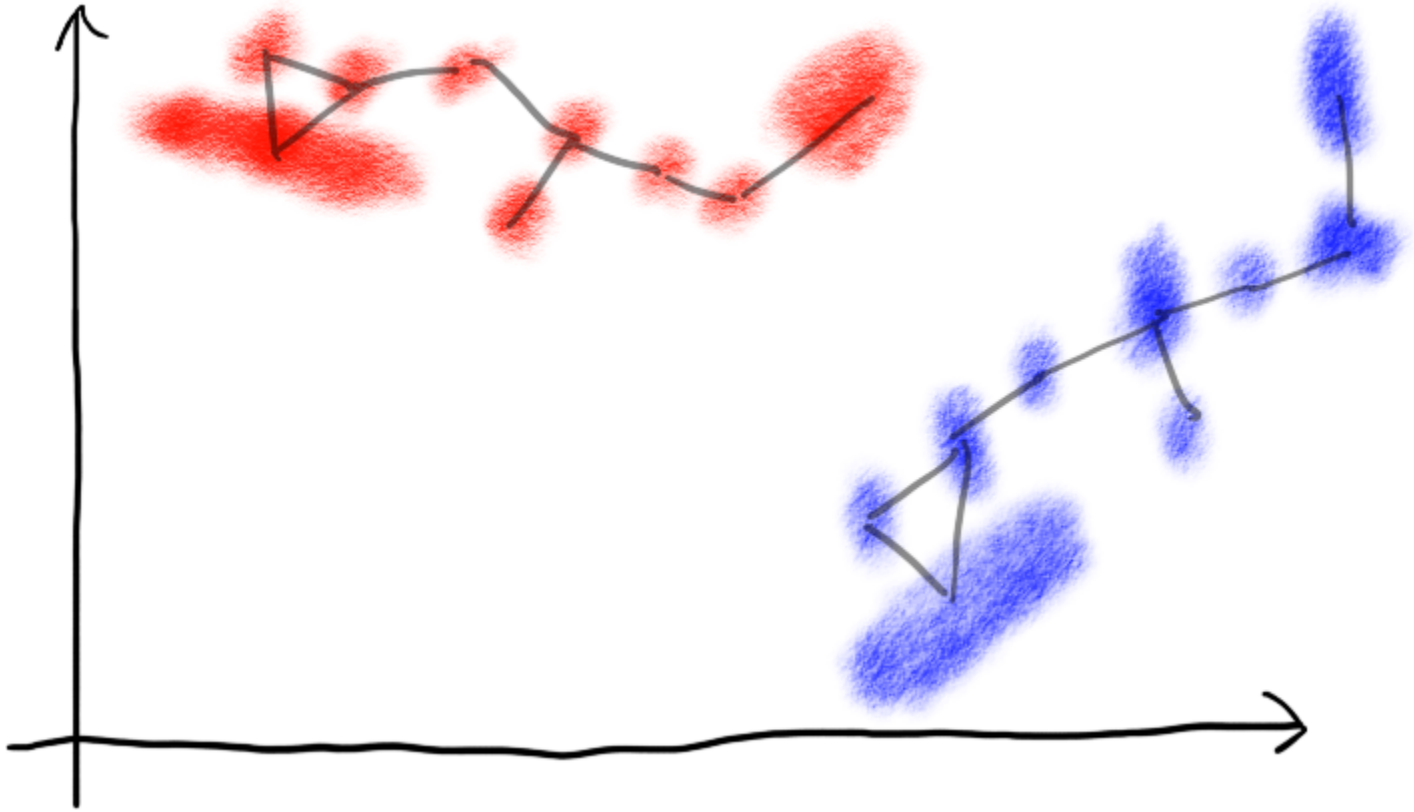
Assume we have two different but related distributions $p(x)$ and $p'(x)$ but no corresponding samples.

Examples:

- daytime vs nightime images
- outdoors scenes in sunshine vs rain
- photographic images vs oil paintings
- clean binary document image vs photographic capture

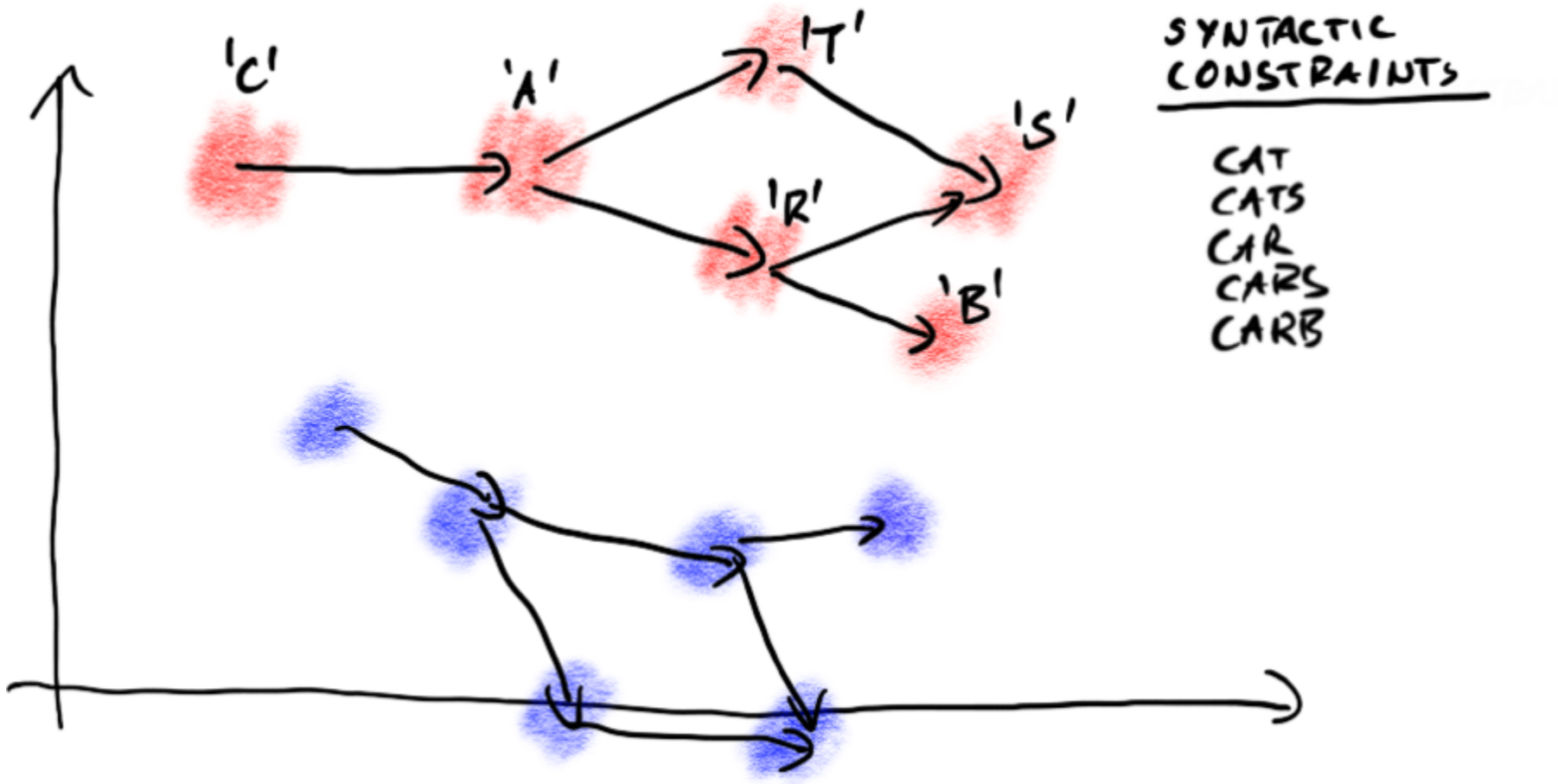
Can we map between these two domains given just independent samples from each? Sometimes.

Alignment of Mixture Densities (no other info)



- distribution alignment is possible for distributions with a lot of structure
- ... if the "general shape" of the distribution is preserved

Alignment of Mixture Densities (with sequences)



- syntactic constraints (e.g. sequences of samples) help with distribution alignment
- this is what EM training for OCR, speech, etc. tends to do

Uses of Distribution Alignment for Training

E.g. AV images: summer and winter, lots of summer training data, little winter training data

Approach 1:

- train model on summer + translate winter to summer

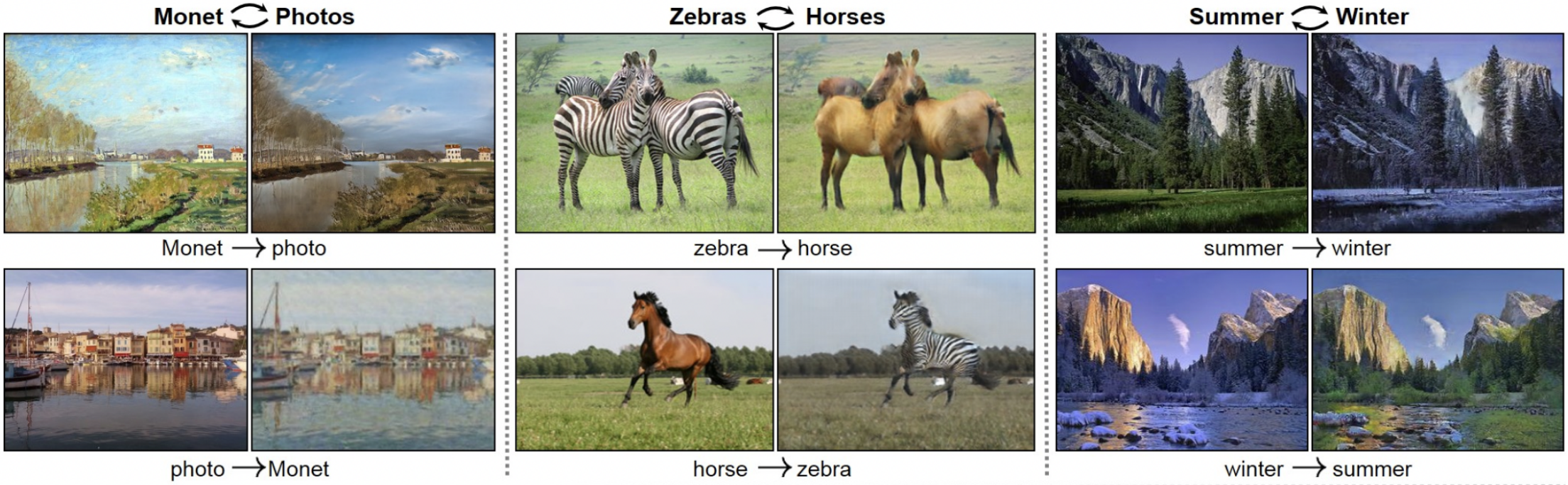
Approach 2:

- use distribution alignment to generate lots of winter images from summer images, then augment the training set with that
- train model on summer images and artificially generated winter images

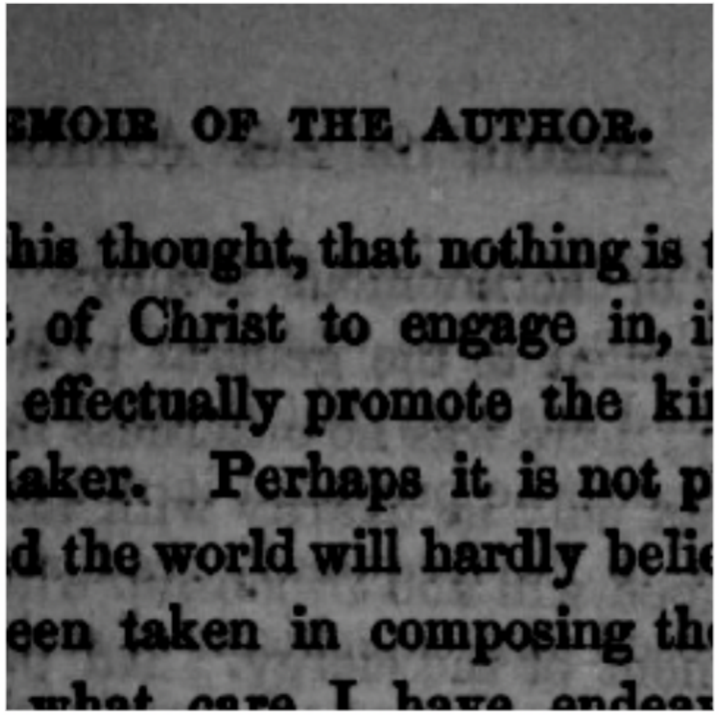
CycleGAN

- two generators $G' : x \rightarrow x'$, $G : x' \rightarrow x$
- two discriminators: D' (discriminates real/fake x'), D (discriminates real/fake x)
- train G, D and G', D' like a regular GAN
- require that $G' \circ G$ and $G \circ G'$ behave like identities

CycleGAN Examples

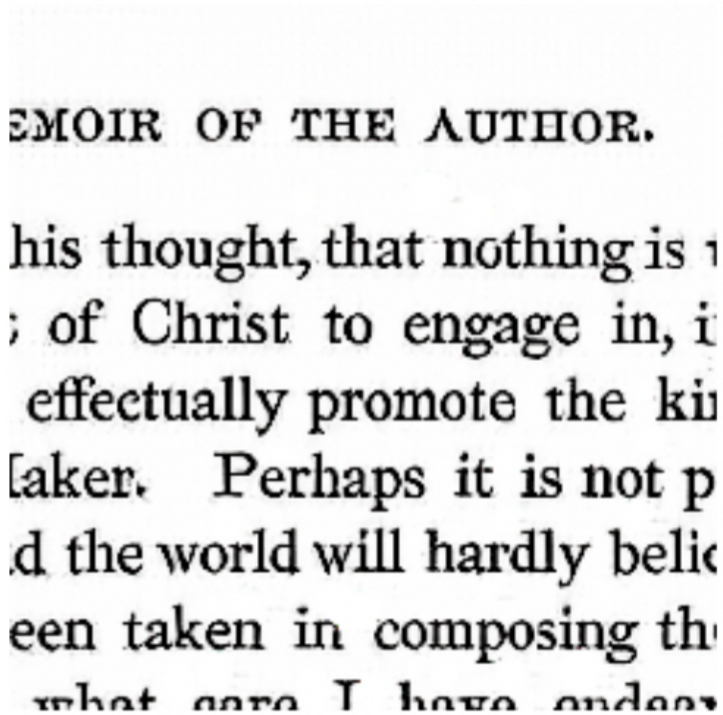


CycleGAN Examples



g_{AB}

g_{BA}



d_A d_B

Ongoing and Future Research

- extensions to text, speech, music, video, etc.
- condition on language (of course, already lots of news coverage)
- engineering work to remove artifacts and make output indistinguishable from real
- create forensic tools that can detect fakes
- combine transformers with each of the different generative approaches
- use non-generative unsupervised pretraining to help improve generative models

Limitations

It is an open question whether generative models will ever be useful unsupervised models for transfer learning to discriminative or other generative models.

High quality generative output requires modeling a large amount of detail that is irrelevant to most tasks.

Almahairi, Amjad, Sai Rajeshwar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. “Augmented CycleGAN: Learning Many-to-Many Mappings from Unpaired Data.” In *Proceedings of the 35th International Conference on Machine Learning*, 195–204. PMLR, 2018. https://proceedings.mlr.press/v80/almahairi18a.html.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. “Wasserstein GAN.” arXiv, December 6, 2017. https://doi.org/10.48550/arXiv.1701.07875.

Bao, Hangbo, Li Dong, Songhao Piao, and Furu Wei. “BEiT: BERT Pre-Training of Image Transformers.” arXiv, September 3, 2022. https://doi.org/10.48550/arXiv.2106.08254.

Bell, Anthony J., and Terrence J. Sejnowski. “An Information-Maximization Approach to Blind Separation and Blind Deconvolution.” *Neural Computation* 7 (1995): 1129–59.

BELL, ANTHONY J., and TERRENCE J. SEJNOWSKI. “The ‘Independent Components’ of Natural Scenes Are Edge Filters.” *Vision Research* 37, no. 23 (December 1997): 3327–38.

Brock, Andrew, Jeff Donahue, and Karen Simonyan. “Large Scale GAN Training for High Fidelity Natural Image Synthesis.” arXiv, February 25, 2019. https://doi.org/10.48550/arXiv.1809.11096.

Bromley, Jane, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. “Signature Verification Using a ‘Siamese’ Time Delay Neural Network,” n.d., 8.

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. “Language Models Are Few-Shot Learners.” arXiv, July 22, 2020. https://doi.org/10.48550/arXiv.2005.14165.

———. “Language Models Are Few-Shot Learners.” arXiv, July 22, 2020. https://doi.org/10.48550/arXiv.2005.14165.

Caron, Mathilde, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. “Emerging Properties in Self-Supervised Vision Transformers.” arXiv, May 24, 2021. https://doi.org/10.48550/arXiv.2104.14294.

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. “A Simple Framework for Contrastive Learning of Visual Representations.” arXiv, June 30, 2020. https://doi.org/10.48550/arXiv.2002.05709.

Chen, Ting, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. “Big Self-Supervised Models Are Strong Semi-Supervised Learners.” arXiv, October 25, 2020. https://doi.org/10.48550/arXiv.2006.10029.

Chen, Xi, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets.” arXiv, June 11, 2016. https://doi.org/10.48550/arXiv.1606.03657.

Creswell, Antonia, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. “Generative Adversarial Networks: An Overview.” *IEEE Signal Processing Magazine* 35, no. 1 (January 2018): 53–65. https://doi.org/10.1109/MSP.2017.2765202.

Croitoru, Florinel-Alin, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. “Diffusion Models in Vision: A Survey.” arXiv, September 10, 2022. https://doi.org/10.48550/arXiv.2209.04747.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.” arXiv, May 24, 2019. https://doi.org/10.48550/arXiv.1810.04805.

Dinh, Laurent, David Krueger, and Yoshua Bengio. “NICE: Non-Linear Independent Components Estimation.” arXiv, April 10, 2015. https://doi.org/10.48550/arXiv.1410.8516.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. “Density Estimation Using Real NVP.” arXiv, February 27, 2017. https://doi.org/10.48550/arXiv.1605.08803.

Doersch, Carl, Abhinav Gupta, and Alexei A. Efros. “Unsupervised Visual Representation Learning by Context Prediction.” arXiv, January 16, 2016. https://doi.org/10.48550/arXiv.1505.05192.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, et al. “An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale.” arXiv, June 3, 2021. https://doi.org/10.48550/arXiv.2010.11929.

Du, Yilun, and Igor Mordatch. “Implicit Generation and Generalization in Energy-Based Models.” arXiv, June 29, 2020. https://doi.org/10.48550/arXiv.1903.08689.

Goldberg, Yoav, and Omer Levy. “Word2vec Explained: Deriving Mikolov et al.’s Negative-Sampling Word-Embedding Method.” arXiv, February 15, 2014. https://doi.org/10.48550/arXiv.1402.3722.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative Adversarial Networks.” arXiv, June 10, 2014. https://doi.org/10.48550/arXiv.1406.2661.

Graves, Alex. “Generating Sequences With Recurrent Neural Networks.” arXiv, June 5, 2014. http://arxiv.org/abs/1308.0850.

Grill, Jean-Bastien, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, et al. “Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning.” arXiv, September 10, 2020. https://doi.org/10.48550/arXiv.2006.07733.

He, Kaiming, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. “Masked Autoencoders Are Scalable Vision Learners.” arXiv, December 19, 2021. https://doi.org/10.48550/arXiv.2111.06377.

OpenAI. “Implicit Generation and Generalization Methods for Energy-Based Models,” March 21, 2019. https://openai.com/blog/energy-based-models/.

Jang, Eric, Shixiang Gu, and Ben Poole. “Categorical Reparameterization with Gumbel-Softmax.” arXiv, August 5, 2017. https://doi.org/10.48550/arXiv.1611.01144.

Kingma, Diederik P., and Max Welling. “An Introduction to Variational Autoencoders.” *Foundations and Trends® in Machine Learning* 12, no. 4 (2019): 307–92. https://doi.org/10.1561/2200000056.

———. “An Introduction to Variational Autoencoders.” *Foundations and Trends® in Machine Learning* 12, no. 4 (2019): 307–92. https://doi.org/10.1561/2200000056.

———. “Auto-Encoding Variational Bayes.” arXiv, May 1, 2014. https://doi.org/10.48550/arXiv.1312.6114.

Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. “Siamese Neural Networks for One-Shot Image Recognition,”

n.d., 8.
LeCun, Yann. “Learning Invariant Feature Hierarchies.” In *Computer Vision – ECCV 2012. Workshops and Demonstrations*, edited by Andrea Fusiello, Vittorio Murino, and Rita Cucchiara, 7583:496–505. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. https://doi.org/10.1007/978-3-642-33863-2_51.
Makansi, Osama, Eddy Ilg, Özgün Cicek, and Thomas Brox. “Overcoming Limitations of Mixture Density Networks: A Sampling and Fitting Framework for Multimodal Future Prediction.” In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7137–46, 2019. https://doi.org/10.1109/CVPR.2019.00731.
McCann, Bryan, James Bradbury, Caiming Xiong, and Richard Socher. “Learned in Translation: Contextualized Word Vectors.” arXiv, June 20, 2018. https://doi.org/10.48550/arXiv.1708.00107.
Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. “Efficient Estimation of Word Representations in Vector Space.” arXiv, September 6, 2013. https://doi.org/10.48550/arXiv.1301.3781.
Oord, Aaron van den, Oriol Vinyals, and Koray Kavukcuoglu. “Neural Discrete Representation Learning.” arXiv, May 30, 2018. http://arxiv.org/abs/1711.00937.
Papamakarios, George, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. “Normalizing Flows for Probabilistic Modeling and Inference.” arXiv, April 8, 2021. https://doi.org/10.48550/arXiv.1912.02762.
Park, Taesung, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. “Contrastive Learning for Unpaired Image-to-Image Translation.” arXiv, August 20, 2020. https://doi.org/10.48550/arXiv.2007.15651.
Pathak, Deepak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. “Context Encoders: Feature Learning by Inpainting.” arXiv, November 21, 2016. https://doi.org/10.48550/arXiv.1604.07379.
Pennington, Jeffrey, Richard Socher, and Christopher Manning. “Glove: Global Vectors for Word Representation.” In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–43. Doha, Qatar: Association for Computational Linguistics, 2014. https://doi.org/10.3115/v1/D14-1162.
Pentland, Alex. “E i g e d c e s for Recognition,” n.d., 16.
Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. “Deep Contextualized Word Representations.” arXiv, March 22, 2018. http://arxiv.org/abs/1802.05365.
Principe, Jose C., Dongxin Xu, and John W. Fisher Iii. “Information-Theoretic Learning,” 1999.
Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. “Improving Language Understanding by Generative Pre-Training,” n.d., 12.
Razavi, Ali, Aaron van den Oord, and Oriol Vinyals. “Generating Diverse High-Fidelity Images with VQ-VAE-2.” arXiv, June 2, 2019. https://doi.org/10.48550/arXiv.1906.00446.
Rocca, Joseph. “Understanding Variational Autoencoders (VAEs).” Medium, March 21, 2021. https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73.
Shree, Priya. “The Journey of Open AI GPT Models.” *Walmart Global Tech Blog* (blog), November 10, 2020. https://medium.com/walmartglobaltech/the-journey-of-open-ai-gpt-models-32d95b7b7fb2.
Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention Is All You Need.” arXiv, December 5, 2017. https://doi.org/10.48550/arXiv.1706.03762.
Vincent, Pascal, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion,” n.d., 38.
Zhang, Han, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. “Self-Attention Generative Adversarial Networks.” arXiv, June 14, 2019. https://doi.org/10.48550/arXiv.1805.08318.
Zhang, Richard, Phillip Isola, and Alexei A. Efros. “Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction.” arXiv, April 20, 2017. https://doi.org/10.48550/arXiv.1611.09842.
Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks.” arXiv, August 24, 2020. https://doi.org/10.48550/arXiv.1703.10593.