Lecture 12:
Activity Recognition and Unsupervised Learning

Tuesday April 4, 2017

Announcements!

- International Max Planck Research School for Intelligent Systems with director Michael Black, applications open for 100 new PhD students

- Final Project milestones due today

- Vote for Final Day and Location on Doodle, if you didn't get a Doodle link let me know

- Complain about AWS availability to t-staff

Activity Recognition

Classic Video Segmentation: **Optical Flow**

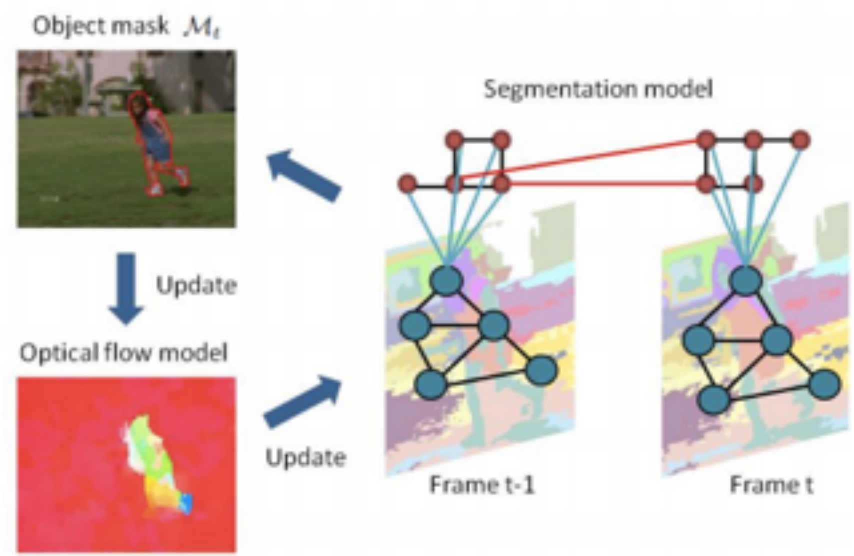


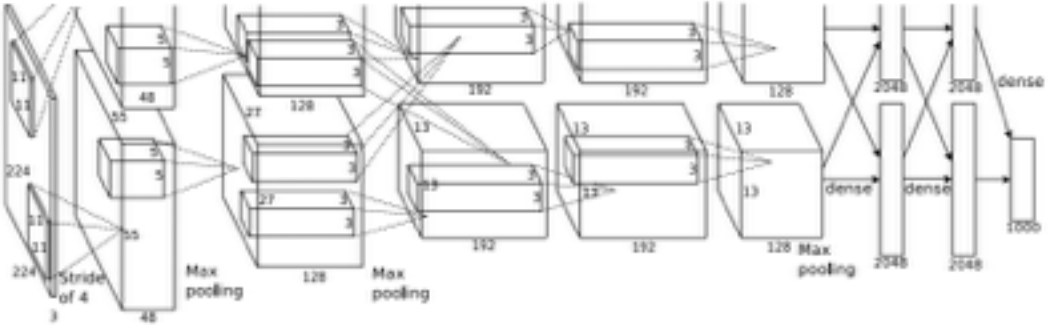
Figure 2. Overview of the proposed model. For segmentation, we consider a multi-level spatial-temporal model. Red circles denote pixels, which belong to the superpixel marked by the turquoise circles. The black and the red lines denote the spatial and temporal relationships, respectively. The relationships between the pixels and the superpixel are denoted by the turquoise lines. After obtaining the object mask, $\mathcal{M}_t$, we use this mask to re-estimate the optical flow, and update both models iteratively.

Latest Iteration:
Video Segmentation via object flow
Tsai et al., 2016

[G. Farnebäck, “Two-frame motion estimation based on polynomial expansion,” 2003]
[T. Brox and J. Malik, “Large displacement optical flow: Descriptor matching in variational motion estimation,” 2011]

Case Study: AlexNet

[Krizhevsky et al. 2012]

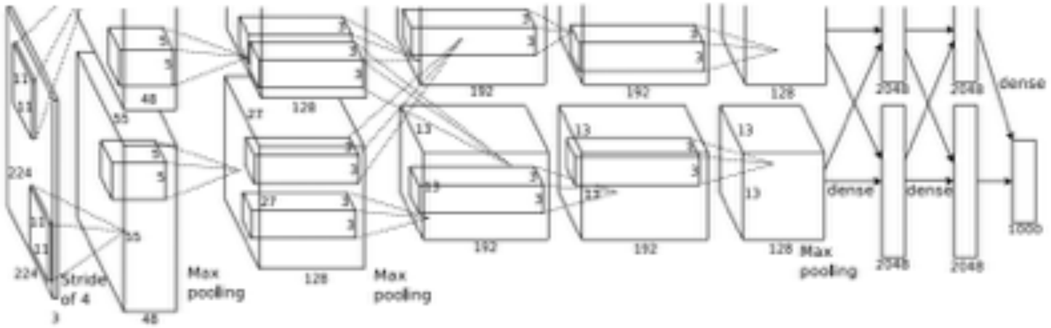


Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**
Q: What if the input is now a small chunk of video? E.g. [227x227x3x15] ?

Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**
Q: What if the input is now a small chunk of video? E.g. [227x227x3x15] ?
A: Extend the convolutional filters in time, perform spatio-temporal convolutions!
E.g. can have 11x11xT filters, where T = 2..15.

Spatio-Temporal ConvNets

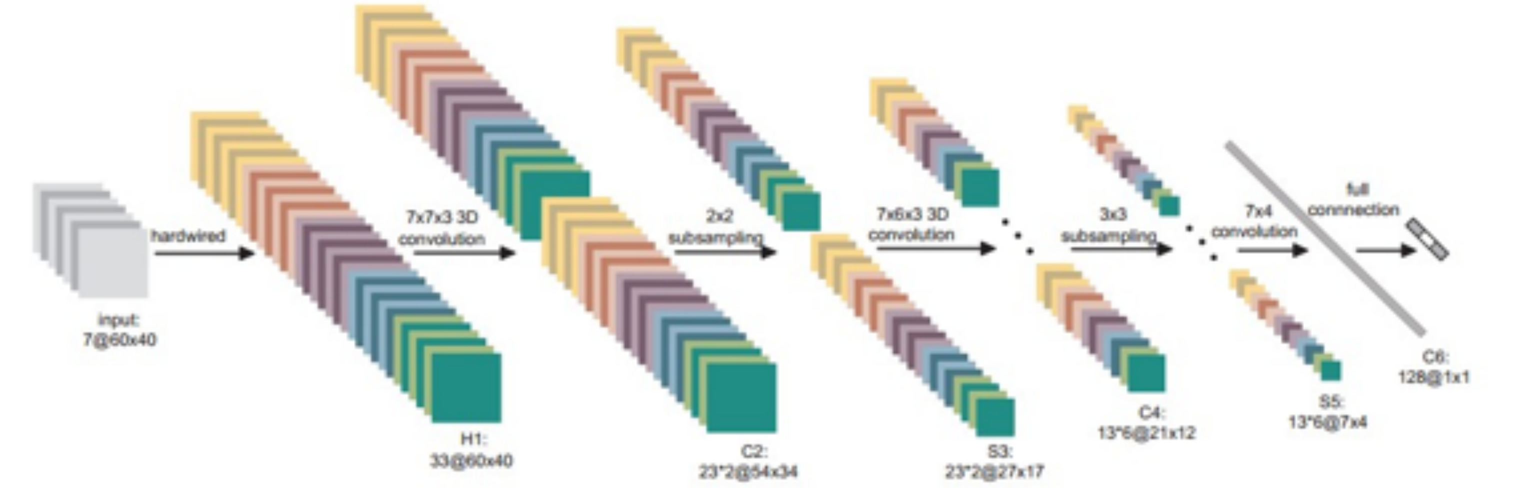


Figure 3. A 3D CNN architecture for human action recognition. This architecture consists of 1 hardwired layer, 3 convolution layers, 2 subsampling layers, and 1 full connection layer. Detailed descriptions are given in the text.

[3D Convolutional Neural Networks for Human Action Recognition, Ji et al., 2010]

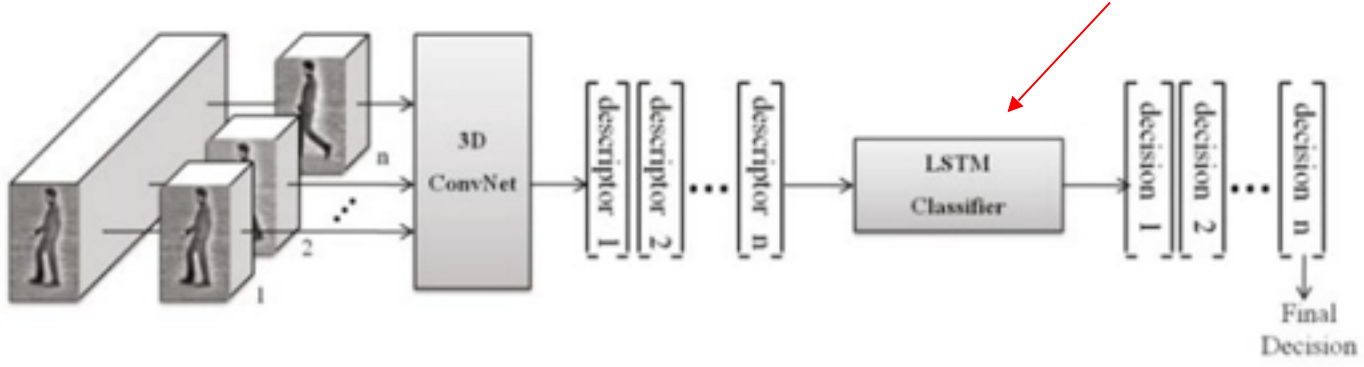
Spatio-Temporal ConvNets

Learned filters on the first layer

[Large-scale Video Classification with Convolutional Neural Networks, Karpathy et al., 2014]

Long-time Spatio-Temporal ConvNets

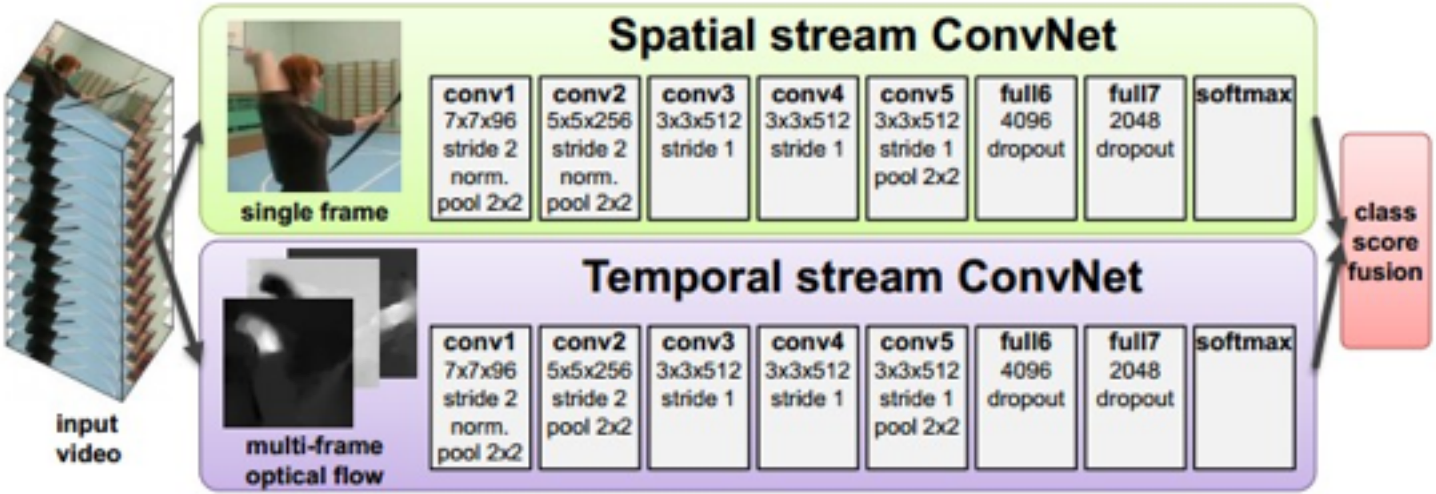
LSTM way before it was cool



(This paper was ahead of its time. Cited 65 times.)

Sequential Deep Learning for Human Action Recognition, Baccouche et al., **2011**

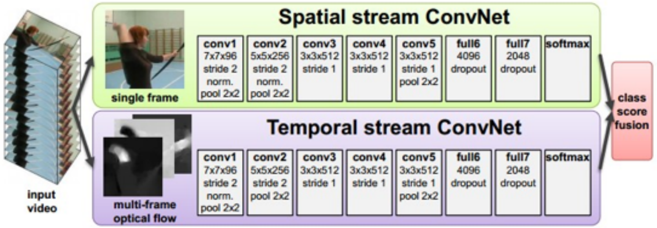
Spatio-Temporal ConvNets



[Two-Stream Convolutional Networks for Action Recognition in Videos, Simonyan and Zisserman 2014]

[T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," 2011]

Spatio-Temporal ConvNets



Spatial stream ConvNet	73.0%	40.5%
Temporal stream ConvNet	83.7%	54.6%
Two-stream model (fusion by averaging)	86.9%	58.0%
Two-stream model (fusion by SVM)	**88.0%**	**59.4%**

Two-stream version works much better than either alone.

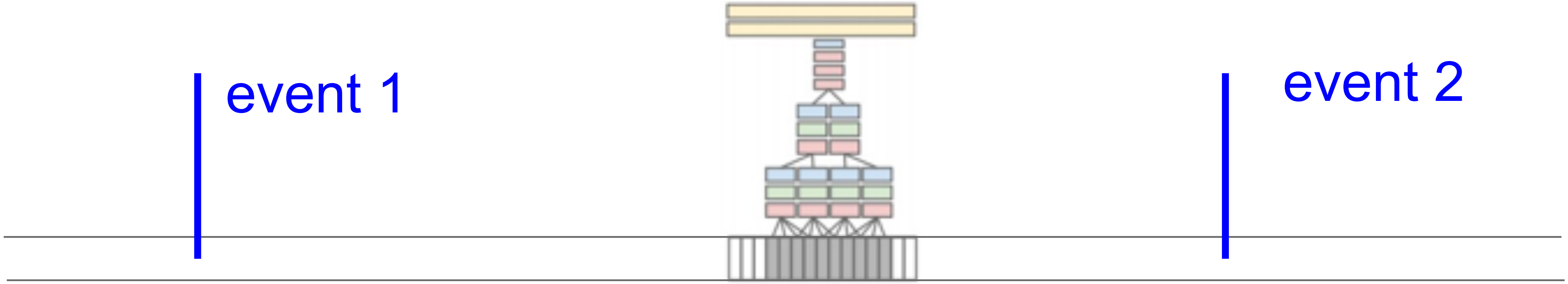
[Two-Stream Convolutional Networks for Action Recognition in Videos, **Simonyan** and Zisserman 2014]

[T. Brox and J. Malik, “Large displacement optical flow: Descriptor matching in variational motion estimation,” 2011]

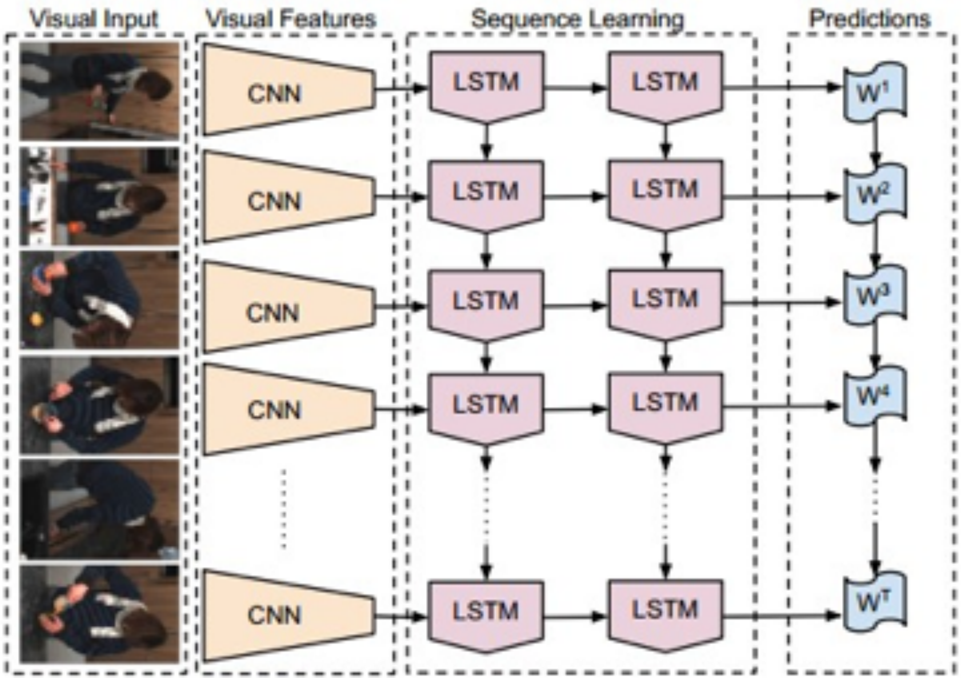
Long-time Spatio-Temporal ConvNets

All 3D ConvNets so far used local motion cues to get extra accuracy (e.g. half a second or so)

Q: what if the temporal dependencies of interest are much much longer? E.g. several seconds?

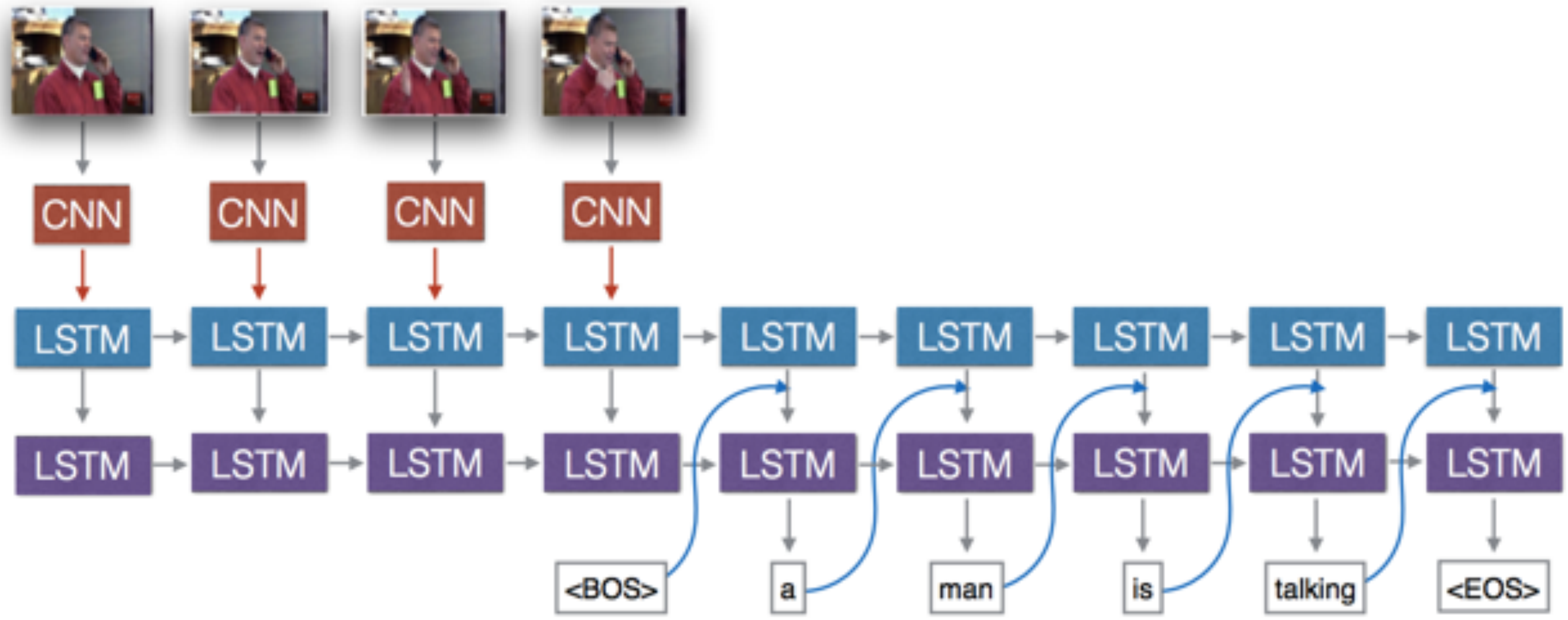


Long-time Spatio-Temporal ConvNets



[Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al., 2015]

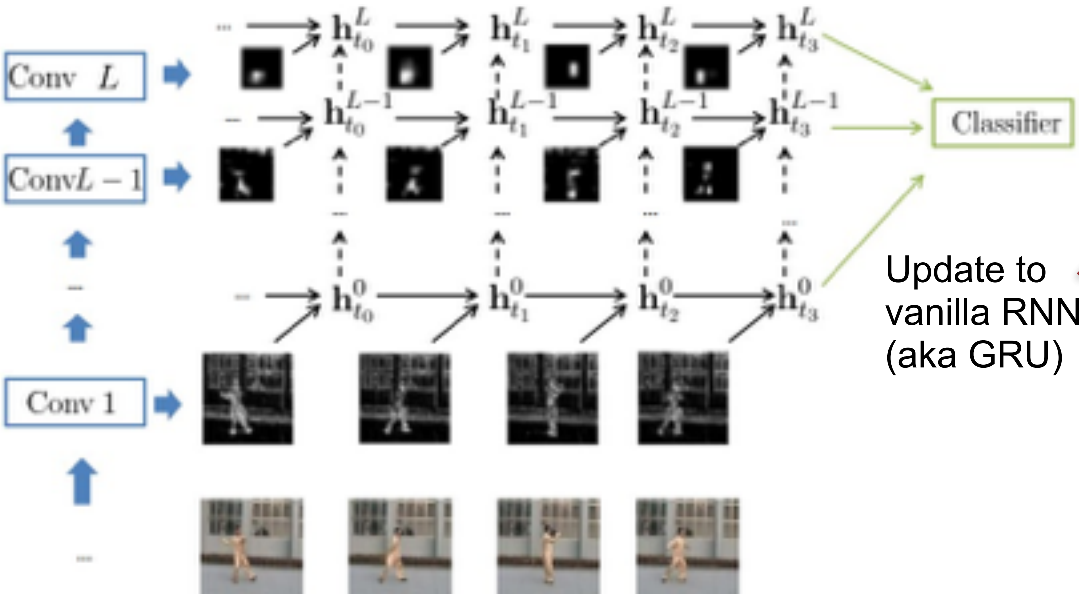
Video Description



Venugopalan et al., "Sequence to Sequence -- Video to Text," 2015.

sequential input & output

Long-time Spatio-Temporal ConvNets



All neurons in the ConvNet are recurrent.

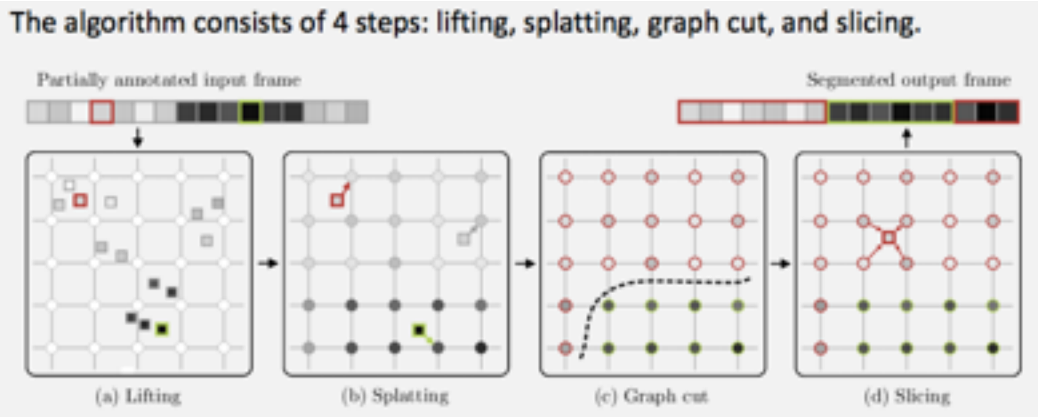
Update to vanilla RNN (aka GRU)

$$\begin{aligned}
\mathbf{z}_t^l &= \sigma(\mathbf{W}_z^l * \mathbf{x}_t^l + \mathbf{U}_z^l * \mathbf{h}_{t-1}^l), && \text{update gate} \\
\mathbf{r}_t^l &= \sigma(\mathbf{W}_r^l * \mathbf{x}_t^l + \mathbf{U}_r^l * \mathbf{h}_{t-1}^l), && \text{reset gate} \\
\tilde{\mathbf{h}}_t^l &= \tanh(\mathbf{W}^l * \mathbf{x}_t^l + \mathbf{U} * (\mathbf{r}_t^l \odot \mathbf{h}_{t-1}^l)), \\
\mathbf{h}_t^l &= (1 - \mathbf{z}_t^l)\mathbf{h}_{t-1}^l + \mathbf{z}_t^l \tilde{\mathbf{h}}_t^l,
\end{aligned}$$

Only requires (existing) 2D CONV routines. No need for 3D spatio-temporal CONV.

[Delving Deeper into Convolutional Networks for Learning Video Representations, Ballas et al., 2016]

Propagation



Graph Cut for Video:

Bilateral Space Video Segmentation
Marki et al., 2016

Unsupervised Learning

Unsupervised Learning Overview

- Autoencoders
    - Vanilla
    - Variational
- Adversarial Networks

Supervised vs Unsupervised

- **Supervised Learning**
- **Data**: (x, y)
  - x is data, y is label
- **Goal**: Learn a *function* to map x -> y
- **Examples**: Classification, regression, object detection, semantic segmentation, image captioning, etc

Supervised vs Unsupervised

- **Supervised Learning**
- **Data**: (x, y)
  - x is data, y is label
- **Goal**: Learn a *function* to map x -> y
- **Examples**: Classification, regression, object detection, semantic segmentation, image captioning, etc

Unsupervised Learning

Data: x
Just data, no labels!

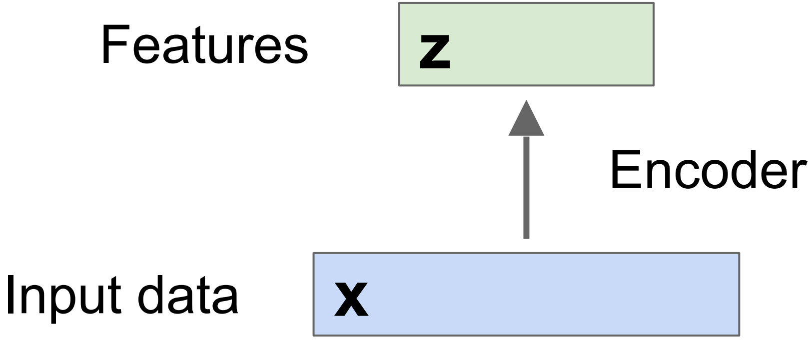
Goal: Learn some *structure* of the data

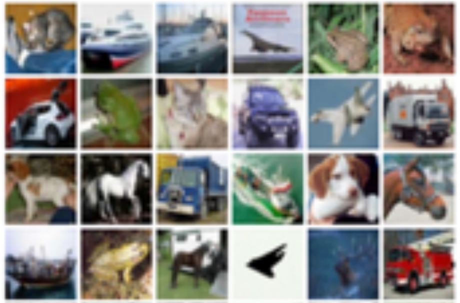
Examples: Clustering, dimensionality reduction, feature learning, generative models, etc.

Unsupervised Learning

- Autoencoders
  - Traditional: feature learning
  - Variational: generate samples
- Generative Adversarial Networks: Generate samples

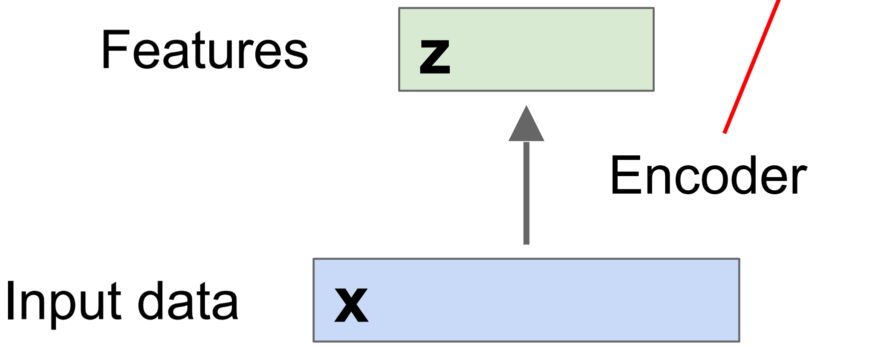
Autoencoders

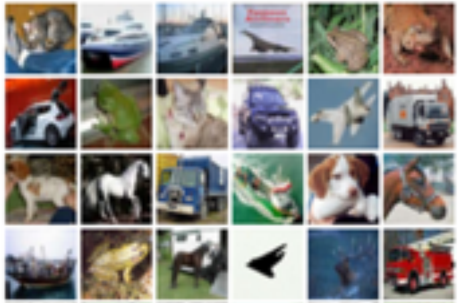




Autoencoders

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN





* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Autoencoders

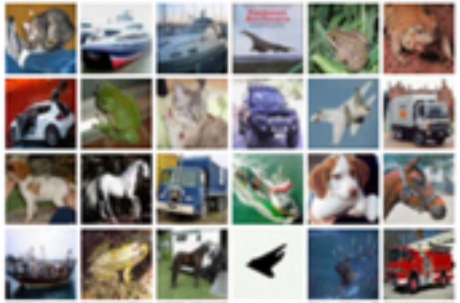
z usually smaller than **x**
(dimensionality reduction)
Prevents trivial solution

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN

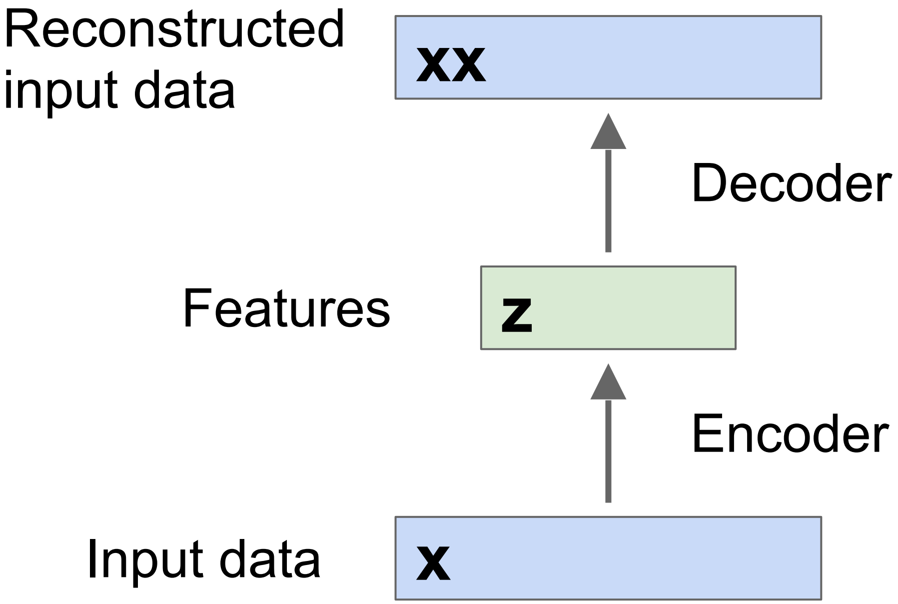
Features **z**

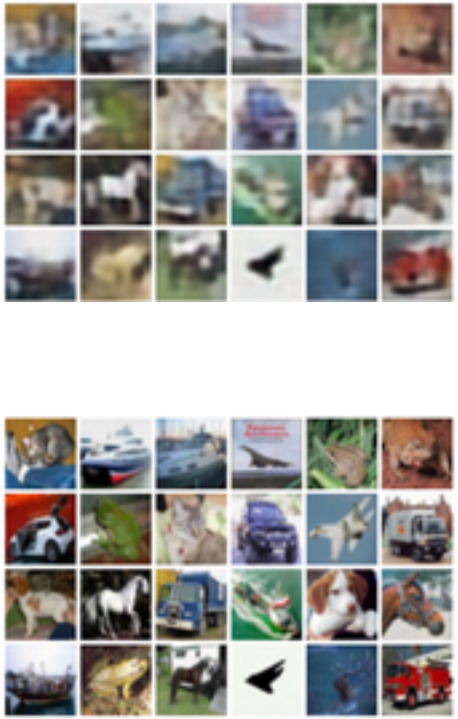
Encoder

Input data **x**



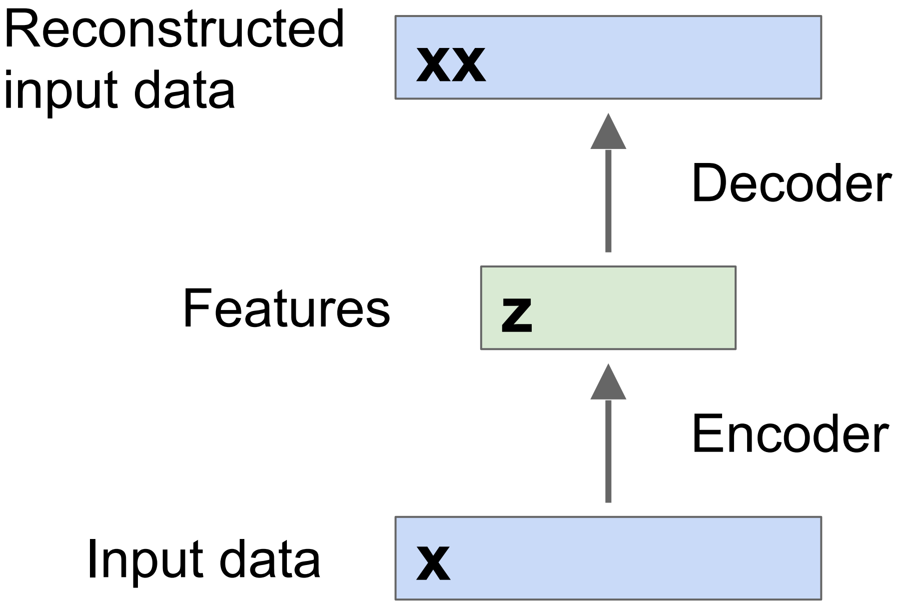
Autoencoders

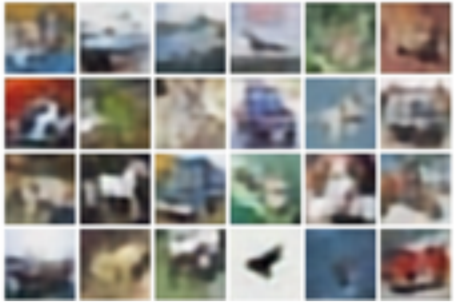




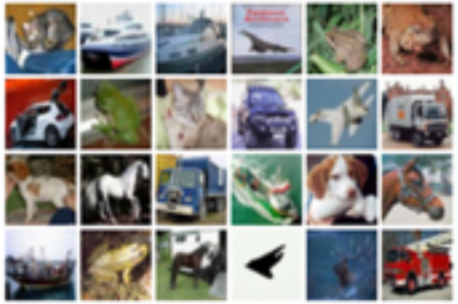
* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Autoencoders



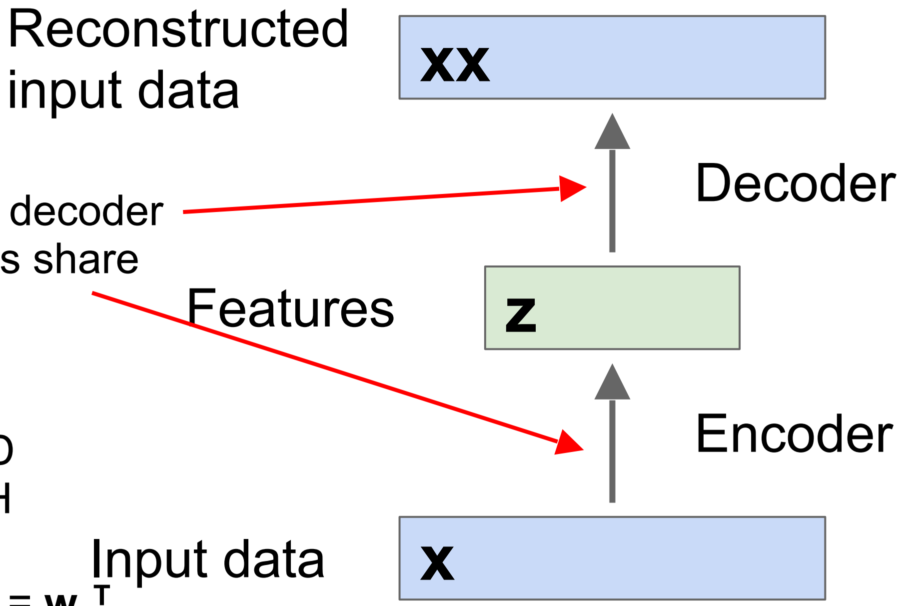


Encoder: 4-layer conv
Decoder: 4-layer upconv



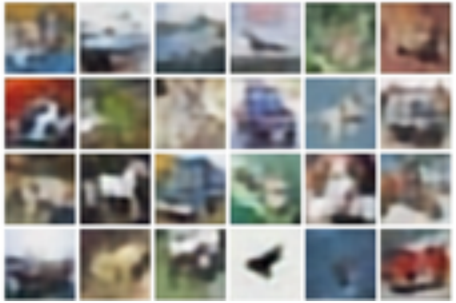
Autoencoders

Goal: Train for reconstruction with no labels!

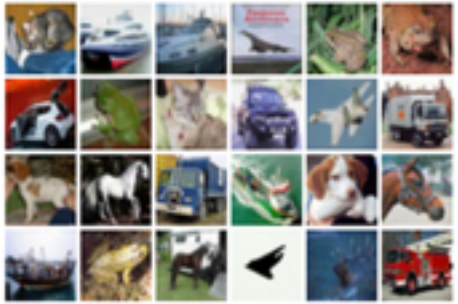


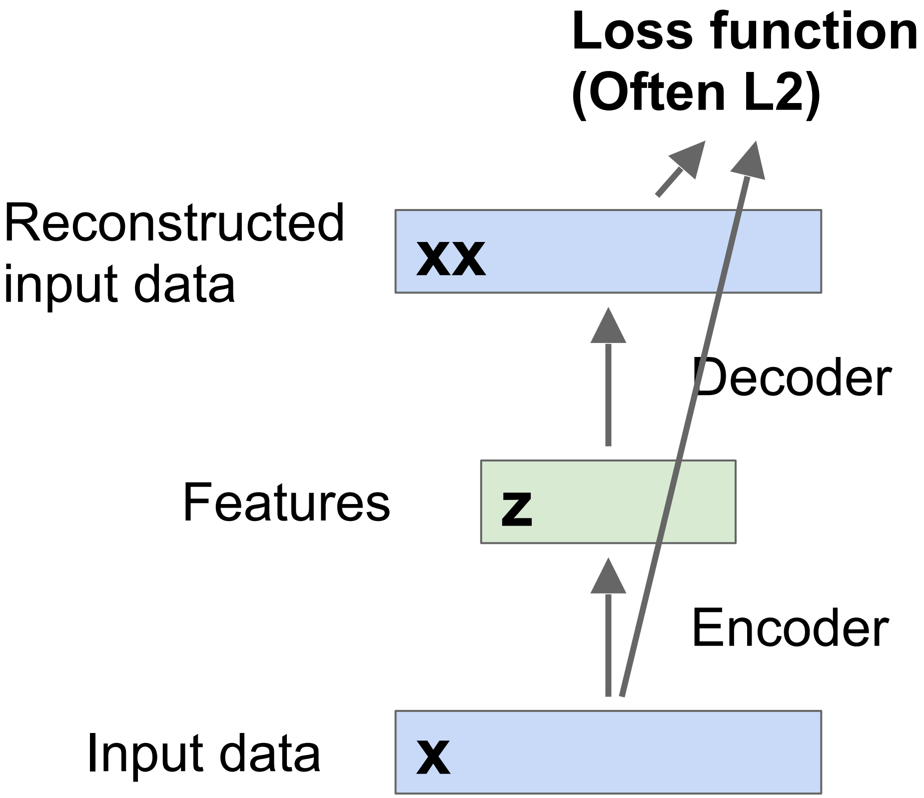
Encoder / decoder sometimes share weights

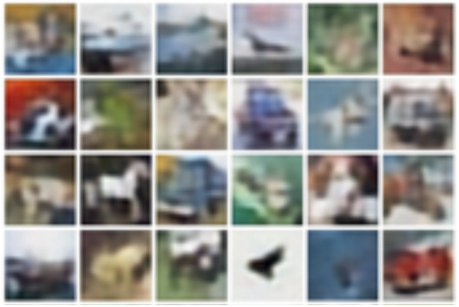
Example:
dim(**x**) = D
dim(**z**) = H
$\mathbf{w}_e$: H x D
$\mathbf{w}_d$: D x H = $\mathbf{w}_e^T$



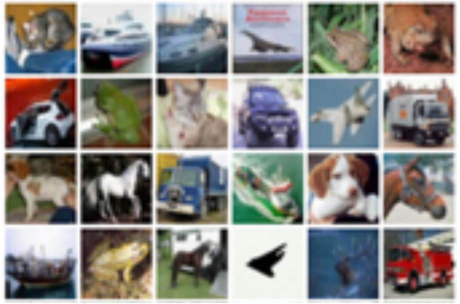
Encoder: 4-layer conv
Decoder: 4-layer upconv



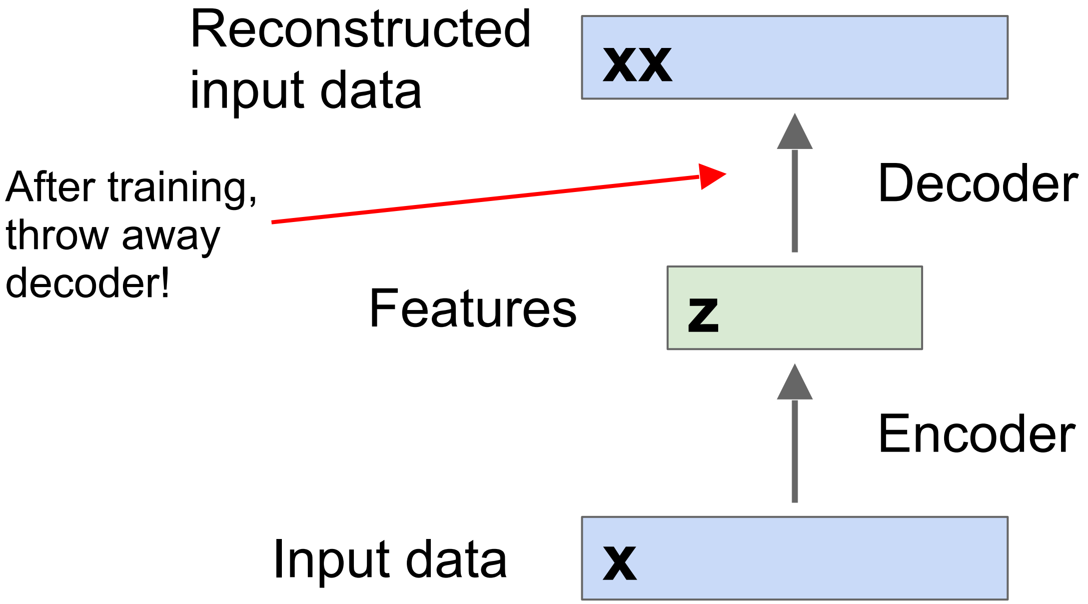


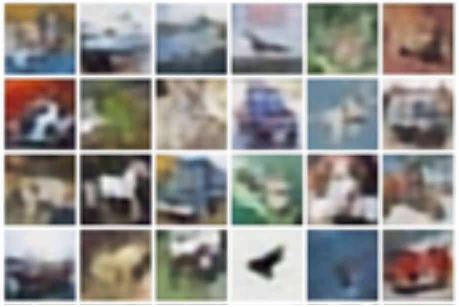


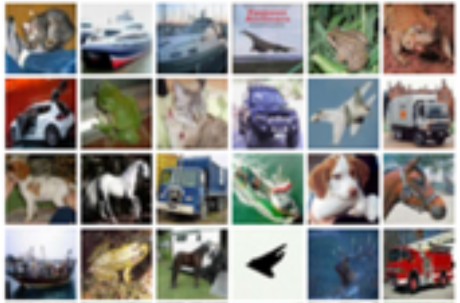
Train for reconstruction with no labels!



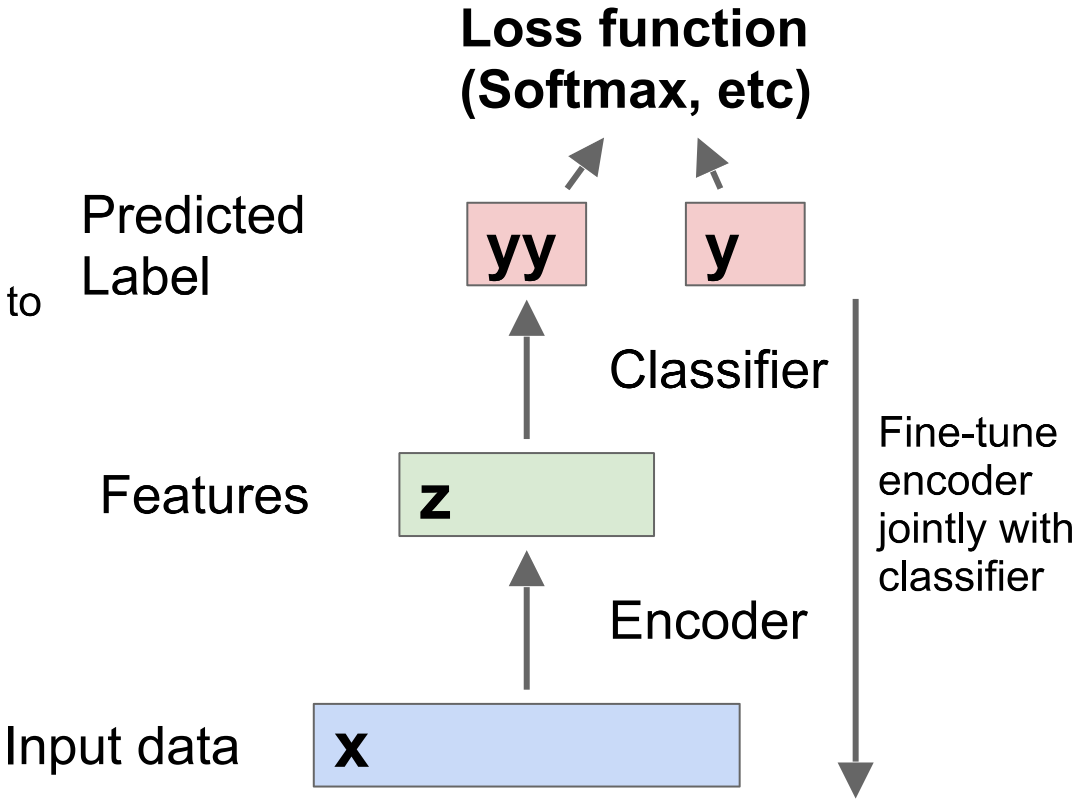
Autoencoders







Use encoder to initialize a **supervised** model



bird plane
dog deer truck

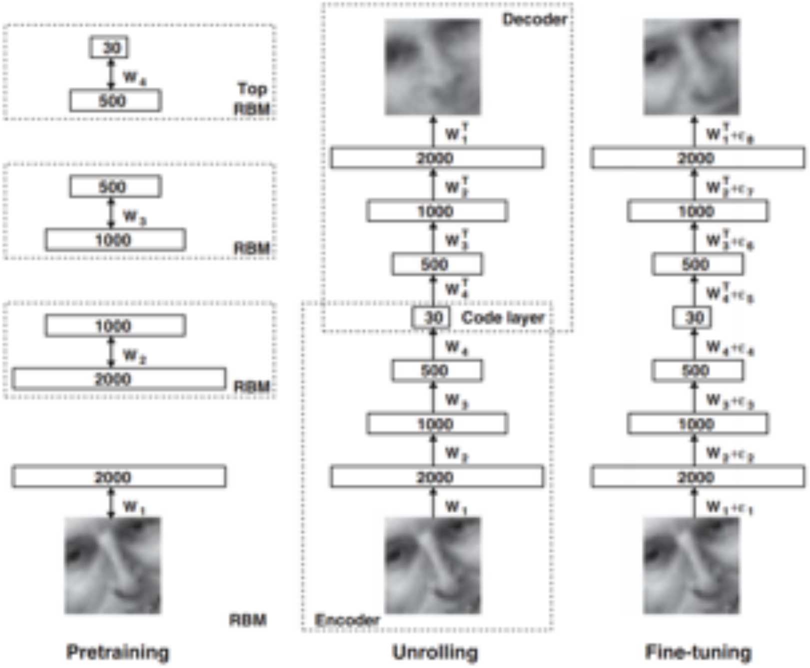
Train for final task (sometimes with small data)

Autoencoders: Greedy Training

In mid 2000s layer-wise pretraining with Restricted Boltzmann Machines (RBM) was common

Training deep nets was hard in 2006!

It is difficult to optimize the weights in nonlinear autoencoders that have multiple hidden layers (2–4). With large initial weights, autoencoders typically find poor local minima; with small initial weights, the gradients in the early layers are tiny, making it infeasible to train autoencoders with many hidden layers. If



Not common anymore

With ReLU, proper initialization, batchnorm, Adam, etc easily train from scratch

Hinton and Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks”, Science 2006

Alternatives

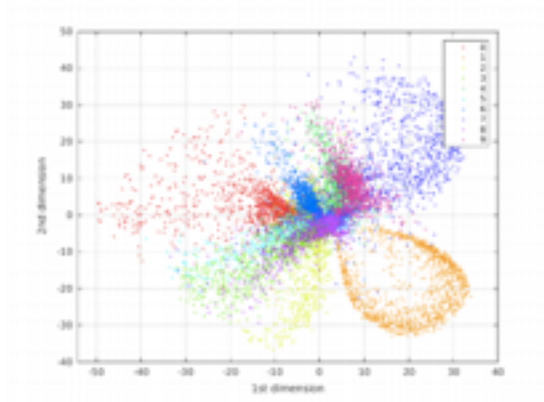


Fig. 1. Visualization of MNIST test set in a 2D space by classic 2-dimensional auto-encoder

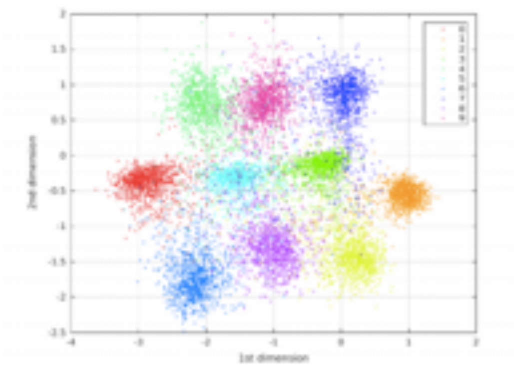


Fig. 2. Visualization of MNIST test set in a 2D space by Siamese network

- Siamese Networks
- Triplet Networks
- Pretraining on unrelated supervised task (aka Transfer Learning)

Creation of a Deep Convolutional Auto-Encoder in Caffe
Volodymyr Turchenko, Artur Luczak. arXiv 2015

Generating Samples

- What if you want to make new examples?
- Need Generative Model
- MCMC?
    - too slow, hard to scale
- MAP / Maximization?
    - Strong overfitting of high dimensional data — won't generate a large variety of interesting things

Variational Autoencoder
a Generative Method

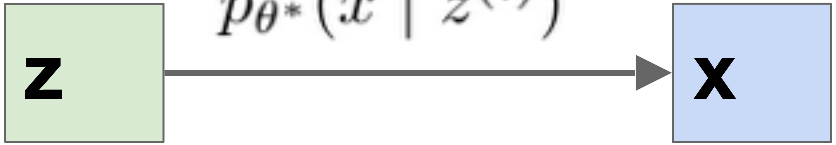
- A Bayesian spin on an autoencoder - lets us generate data!
- Assume our data $\{x^{(i)}\}_{i=1}^{N}$ is generated like this:

Sample from *true* *conditional*

$p_{\theta^*}(x \mid z^{(i)})$

Sample from *true prior*

$p_{\theta^*}(z)$



Intuition: **x** is an image, **z** gives class, orientation, attributes, etc

Problem: Estimate θ without access to latent states $z^{(i)}$!

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoder: Encoder

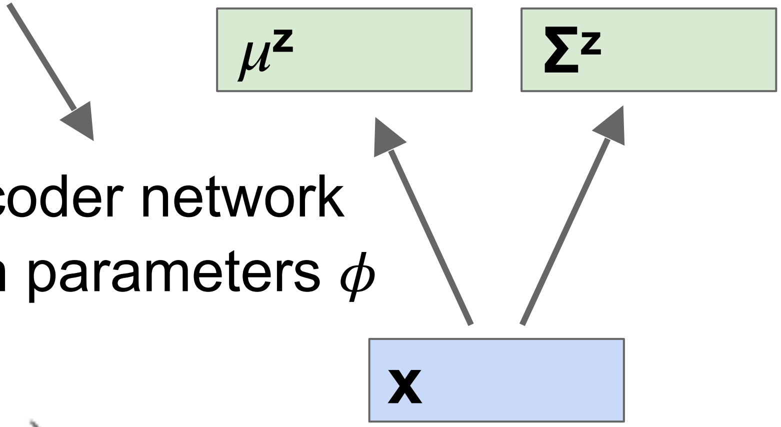
- By Bayes Rule the posterior is:

$$p_\theta(z \mid x) = \frac{\boxed{p_\theta(x \mid z)}\,\boxed{p_\theta(z)}}{\boxed{p_\theta(x)}}$$

Use decoder network =)
Gaussian =)
Intractible integral =(

Fully-connected
or convolutional

Mean and (diagonal)
covariance of
$q_\phi(z \mid x)$



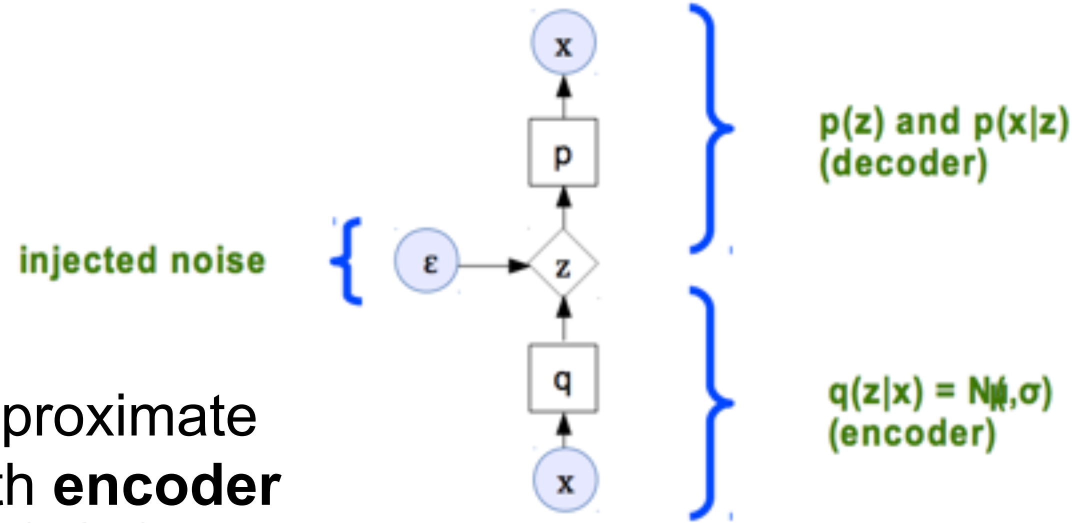
Encoder network
with parameters ϕ

Data point

Approximate posterior with
encoder network $q_\phi(z \mid x)$

Kingma and Welling,
ICLR 2014

Variational auto-encoder



Solution: Approximate posterior with **encoder network** $q_\phi(z \mid x)$

Variational Autoencoder
a Generative Method

Key reparameterization trick

Construct samples $z \sim q_\varphi(z|x)$ in two steps:

1. $\varepsilon \sim p(\varepsilon)$ *(random seed independent of φ)*
2. $z = g(\varphi, \varepsilon, x)$ (differentiable perturbation)

such that $z \sim q_\varphi(z|x)$ (the correct distribution)

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Auto-Encoding Variational Bayes
Online algorithm

repeat

$$\mathbf{x} \leftarrow \text{random datapoint or minibatch}$$
$$\boldsymbol{\epsilon} \leftarrow \text{sample from } p(\boldsymbol{\epsilon})$$
$$g_{\boldsymbol{\theta}}, g_{\boldsymbol{\phi}} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, g(\boldsymbol{\epsilon}, \boldsymbol{\phi}))$$

Backprop (Torch7 / Theano)

Decoder Network Parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot g_{\boldsymbol{\theta}}$

Encoder Network Parameters $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha \cdot g_{\boldsymbol{\phi}}$

e.g. Adagrad

until convergence

Scales to very large datasets!

Variational Autoencoder

Reconstructed **xx**

Sample from $p_\theta(x \mid z)$

$\mu^{\mathbf{x}}$ $\Sigma^{\mathbf{x}}$

Decoder network

z

Sample from $q_\phi(z \mid x)$

$\mu^{\mathbf{z}}$ $\Sigma^{\mathbf{z}}$

Encoder network

Data point **x**

Training like a normal autoencoder: **reconstruction loss** at the end, **regularization toward prior** in middle

Mean and (diagonal) covariance of $p_\theta(x \mid z)$ **(should be close to data x)**

Mean and (diagonal) covariance of $q_\phi(z \mid x)$ **(should be close to prior $p_\theta(z)$)**

Kingma and Welling, ICLR 2014

Autoencoder Overview

- Traditional Autoencoders
  - Try to reconstruct input
  - Used to learn features, initialize supervised model
  - Not used much anymore
- Variational Autoencoders
  - Bayesian meets deep learning
  - Sample from model to generate images

Generative Adversarial Networks

Generative Adversarial Nets

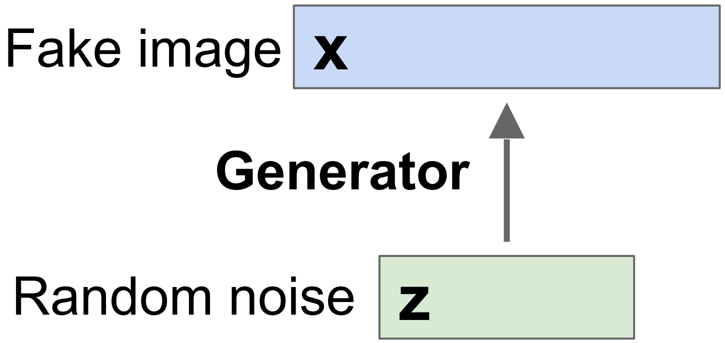
Can we generate images with less math?

Random noise $\mathbf{z}$

Goodfellow et al, "Generative Adversarial Nets", NIPS 2014

Generative Adversarial Nets

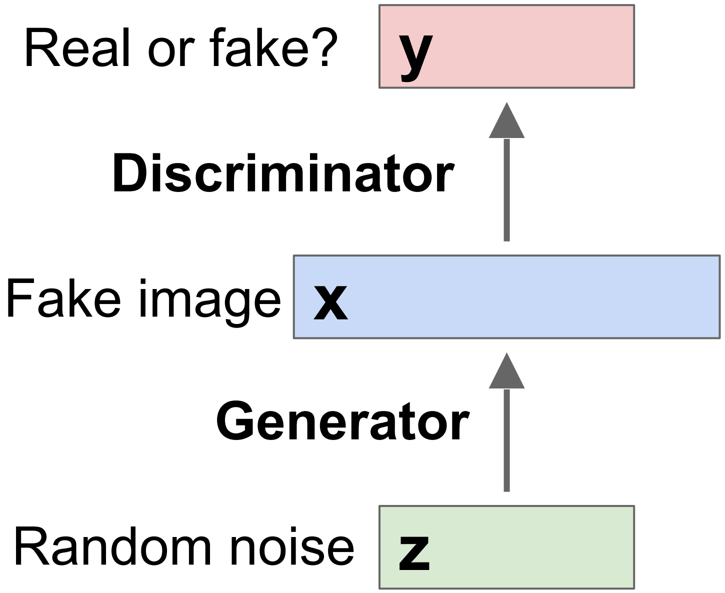
Can we generate images with less math?



Goodfellow et al, “Generative Adversarial Nets”, NIPS 2014

Generative Adversarial Nets

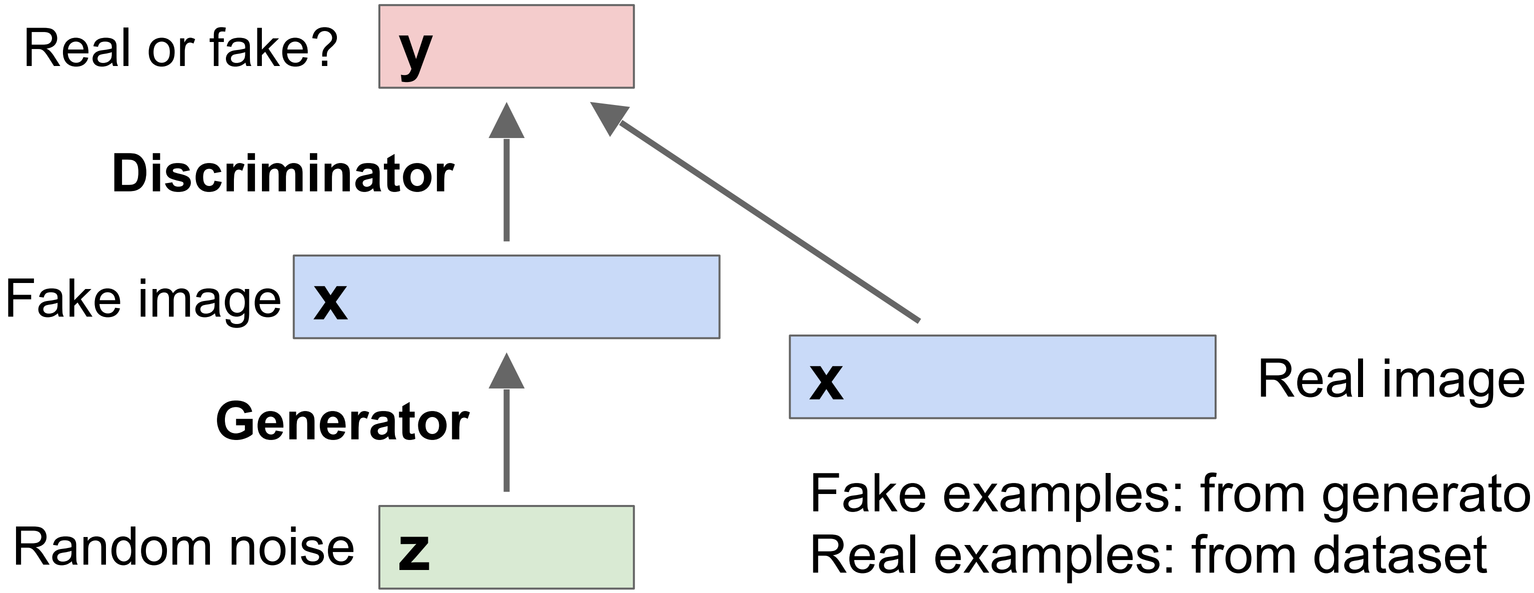
Can we generate images with less math?



Goodfellow et al, "Generative Adversarial Nets", NIPS 2014

Generative Adversarial Nets

Can we generate images with less math?



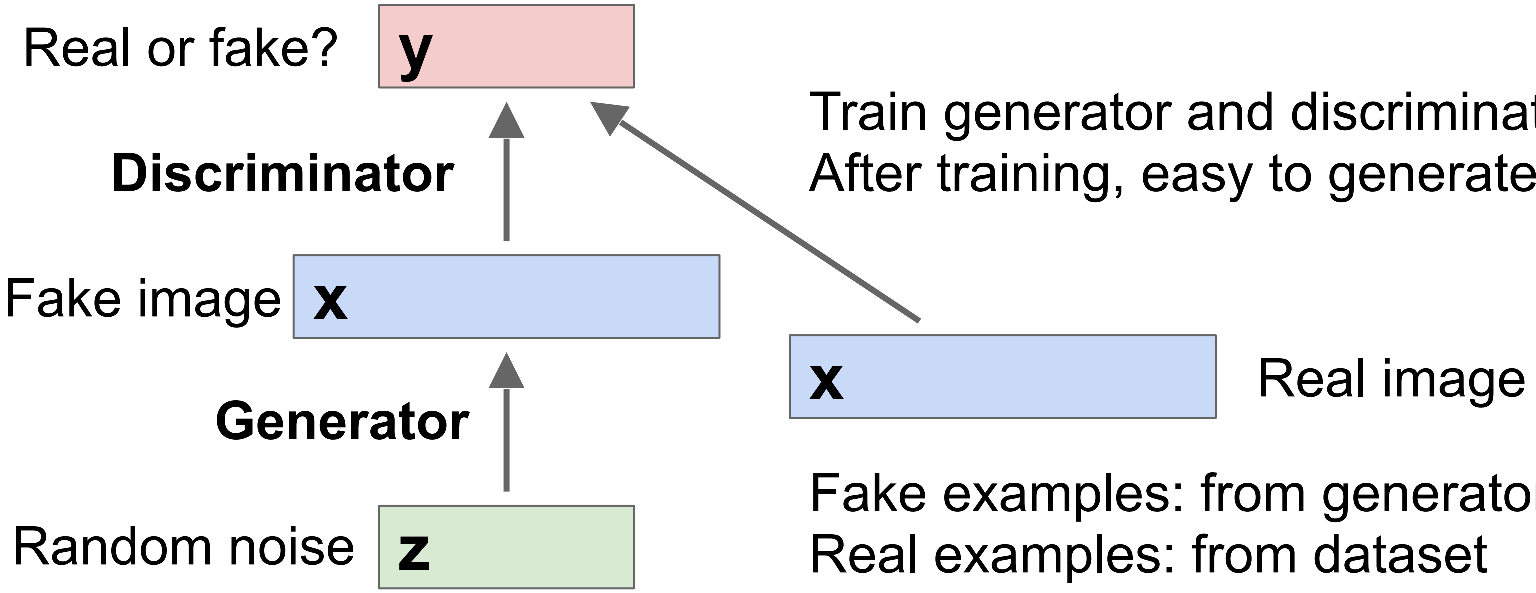
Fake examples: from generator
Real examples: from dataset

Goodfellow et al, "Generative Adversarial Nets", NIPS 2014

Generative Adversarial Nets

Can we generate images with less math?

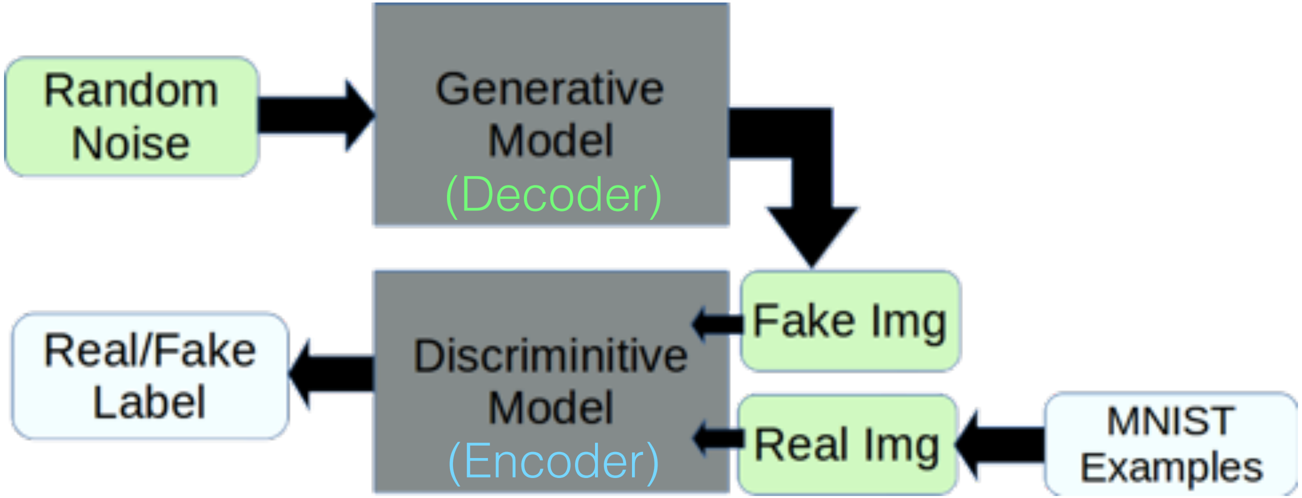
Train generator and discriminator jointly
After training, easy to generate images



Fake examples: from generator
Real examples: from dataset

Goodfellow et al, "Generative Adversarial Nets", NIPS 2014

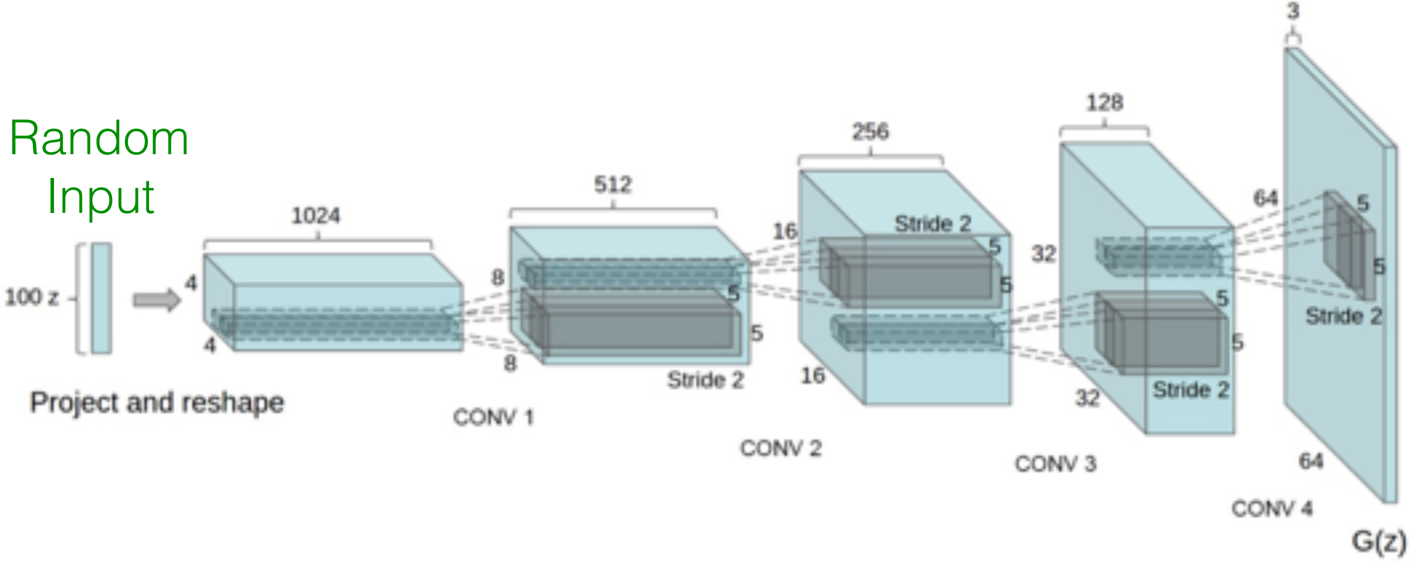
Generative Adversarial Nets



Generative Network

Generated Image

Random Input

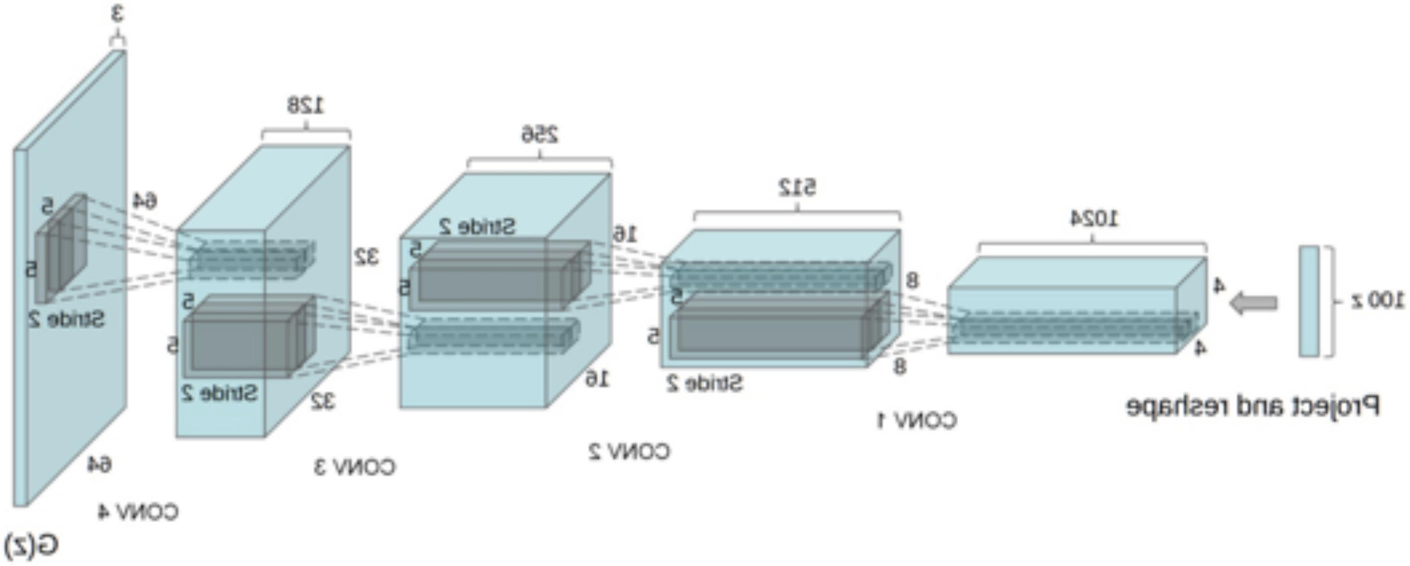


Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

Discriminative Network

Real Training Image

Classified Label Vector



This is just a CNN!

Generative Adversarial Nets: Simplifying

Samples from the model look amazing!

Radford et al, ICLR 2016

Generative Adversarial Nets: Simplifying

Interpolating between random points in latent space

Radford et al, ICLR 2016

Generative Adversarial Nets: Vector Math

Radford et al, ICLR 2016

Smiling woman   Neutral woman   Neutral man

Smiling Man

Samples from the model

Average Z vectors, do arithmetic

Generative Adversarial Nets: Vector Math

Glasses man No glasses man No glasses woman

Woman with glasses

Radford et al,
ICLR 2016

Learning what to Ignore

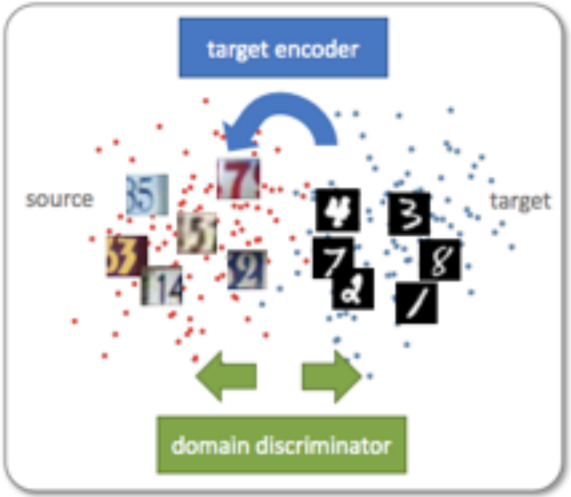


Figure 1: We propose an improved unsupervised domain adaptation method that combines adversarial learning with discriminative feature learning. Specifically, we learn a discriminative mapping of target images to the source feature space (target encoder) by fooling a domain discriminator that tries to distinguish the encoded target images from source examples.

Tzeng et al, “Adversarial Discriminative Domain Adaptation”, arXiv 2017.

Interaction

Figure 9. Interactive image generation and editing. The user can incrementally modify the sketch to change the eyes, hair, and head decorations.

Sangkloy et al, “Scribbler: Controlling Deep Image Synthesis with Sketch and Color”, Siggraph 2017.

Deep Learning and Generalization

(super short) primer on generalization

- given samples $z_1, z_2, \ldots, z_n \sim D$, optimize the cost/loss over hypotheses $h \in \mathcal{H}$:

$$\mathbb{E}\{ f(z, h)\}$$

- want $h_*(z_1, \ldots, z_n)$ such that $\mathbb{E}\{ f(z, h_*)\} \leq \inf_{\mathrm{h}} \mathbb{E}\{ f(z, h)\} + \epsilon(n)$

Central finding of Zhang et al (2017):

deep neural nets are able to fit random labels and data

So how are Deep Nets achieving good generalization?

datasets and models

- CIFAR10 dataset:
  60000 images (50000 train, 10000 validation), 10 categories

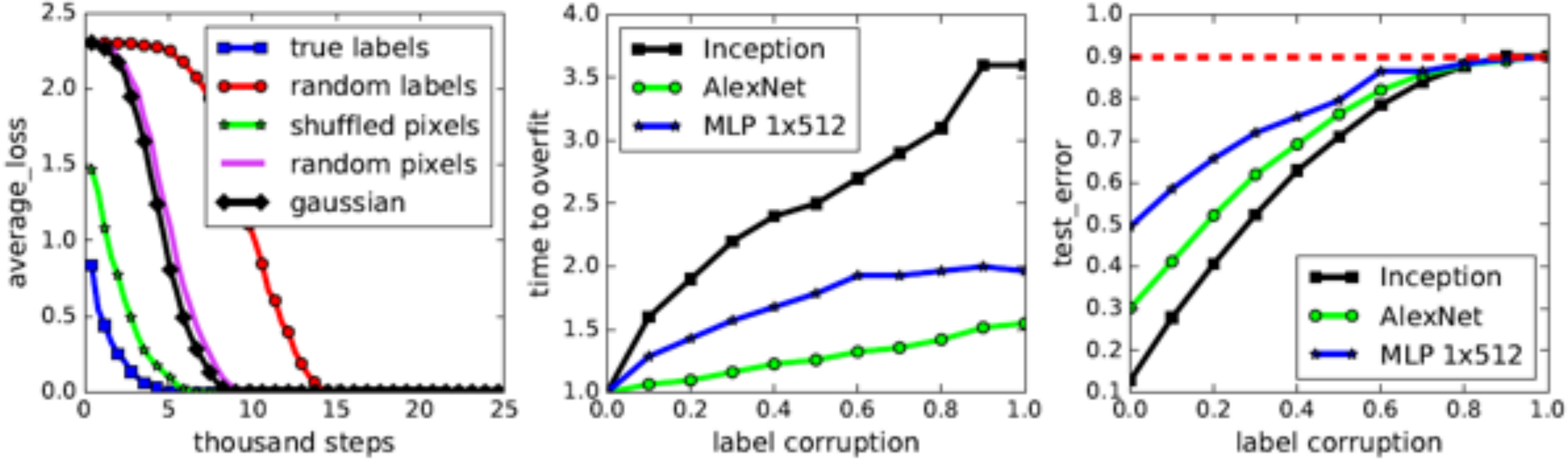
- ImageNet dataset:
  1,281,167 training images, 50000 validation images, 1000 categories

- alexnet, inception, multilayer perceptrons

randomization tests:

- true labels: original dataset
- partially corrupted labels: randomize p fraction of labels
- random labels: randomize all labels
- shuffled pixels: uniform random permutation applied to all images
- random pixels: different random permutation
- gaussian: pixels replaced by i.i.d. gaussian r.v.

performance on randomized tests



(a) learning curves (b) convergence slowdown (c) generalization error growth

explicit regularization does not help much

Table 1: The training and test accuracy (in percentage) of various models on the CIFAR10 dataset. Performance with and without data augmentation and weight decay are compared. The results of fitting random labels are also included.

model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random labels)		no	no	100.0	10.12
Alexnet	1,387,786	yes	yes	99.90	81.22
		yes	no	99.82	79.66
		no	yes	100.0	77.36
		no	no	100.0	76.07
(fitting random labels)		no	no	99.82	9.86
MLP 3x512	1,735,178	no	yes	100.0	53.35
		no	no	100.0	52.39
(fitting random labels)		no	no	100.0	10.48
MLP 1x512	1,209,866	no	yes	99.80	50.39
		no	no	100.0	50.51
(fitting random labels)		no	no	99.34	10.61