

Lecture 14:

Videos

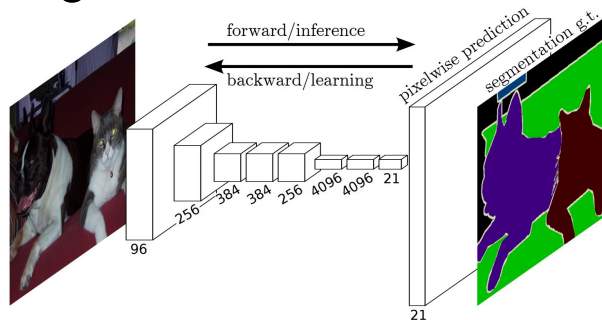
Unsupervised Learning

Administrative

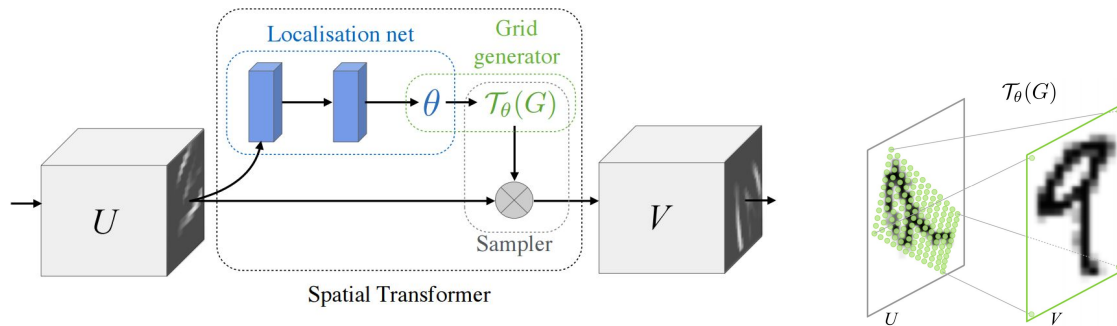
- Everyone should be done with Assignment 3 now
- Milestone grades will go out soon

Last class

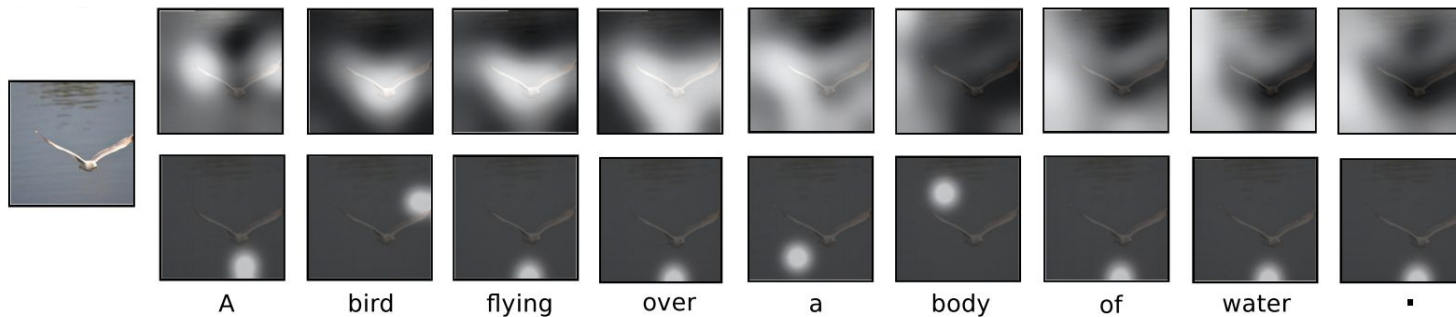
Segmentation



Spatial Transformer

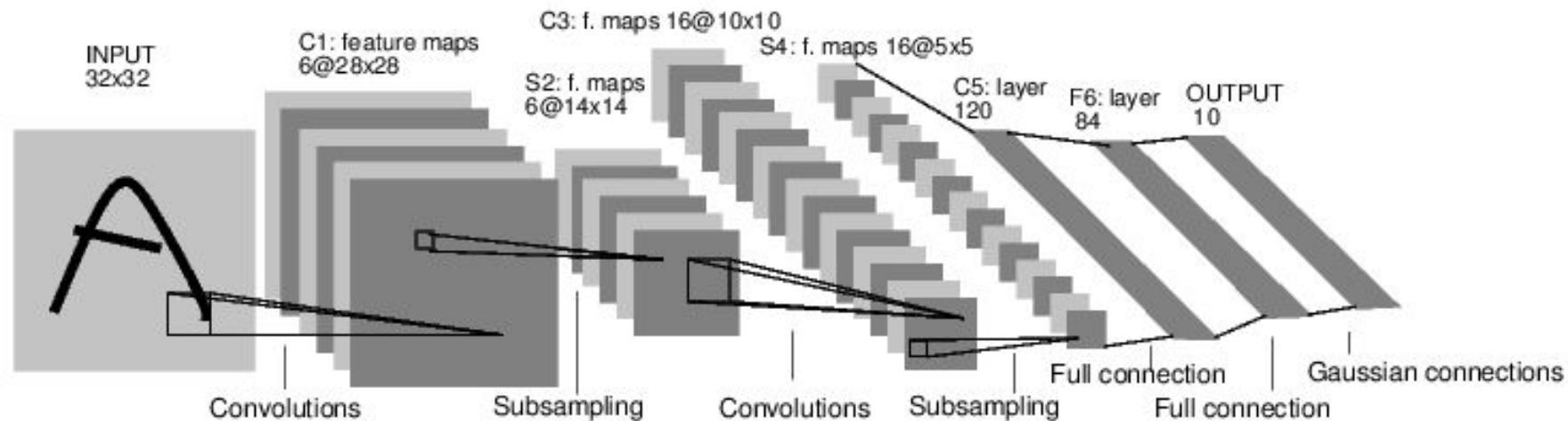


Soft Attention



Videos

ConvNets for images



Feature-based approaches to Activity Recognition

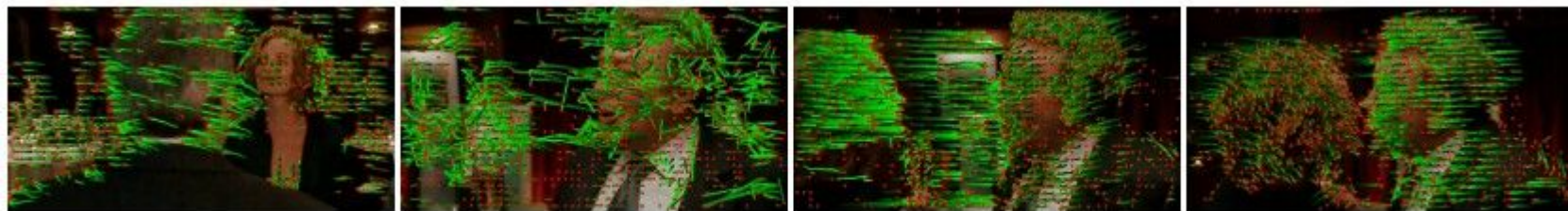
Dense trajectories and motion boundary descriptors for action recognition

Wang et al., 2013

Action Recognition with Improved Trajectories

Wang and Schmid, 2013

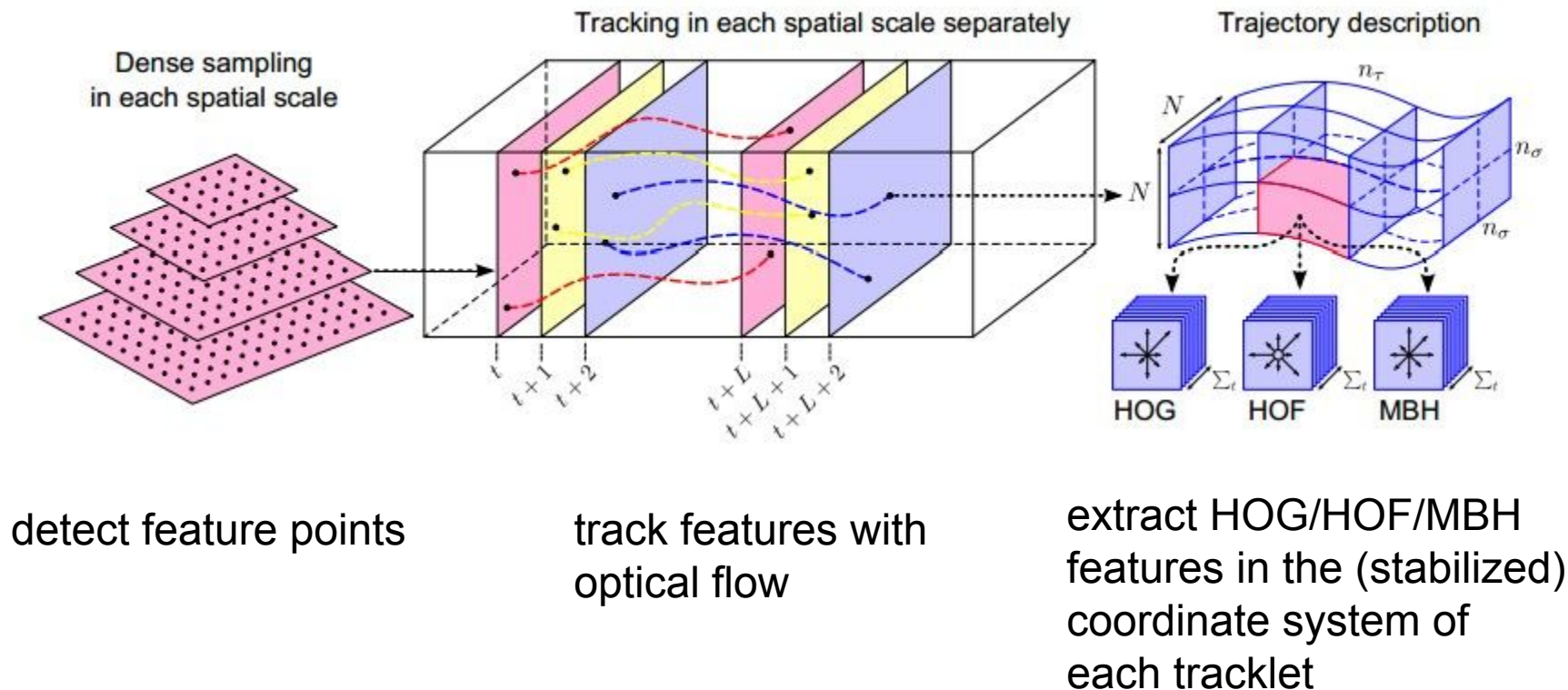
(code available!)



Dense trajectories

Dense trajectories and motion boundary descriptors for action recognition

Wang et al., 2013



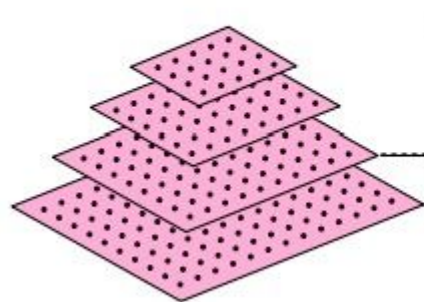
Dense trajectories and motion boundary descriptors for action recognition

Wang et al., 2013



detected feature points

Dense sampling
in each spatial scale



[J. Shi and C. Tomasi, "Good features to track," CVPR 1994]

[Ivan Laptev 2005]

Dense trajectories and motion boundary descriptors for action recognition

Wang et al., 2013



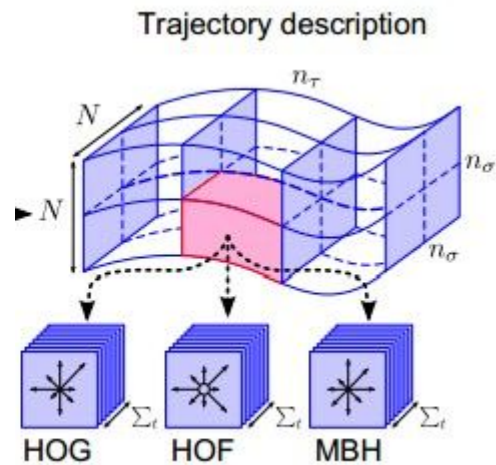
track each keypoint using **optical flow**.

[G. Farneback, "Two-frame motion estimation based on polynomial expansion," 2003]

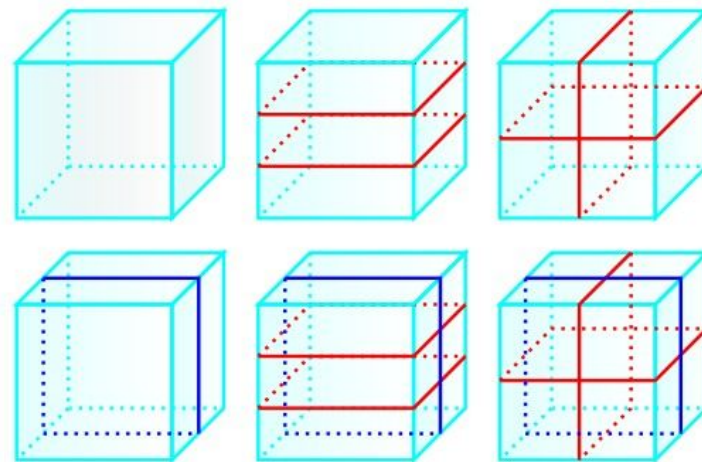
[T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," 2011]

Dense trajectories and motion boundary descriptors for action recognition

Wang et al., 2013



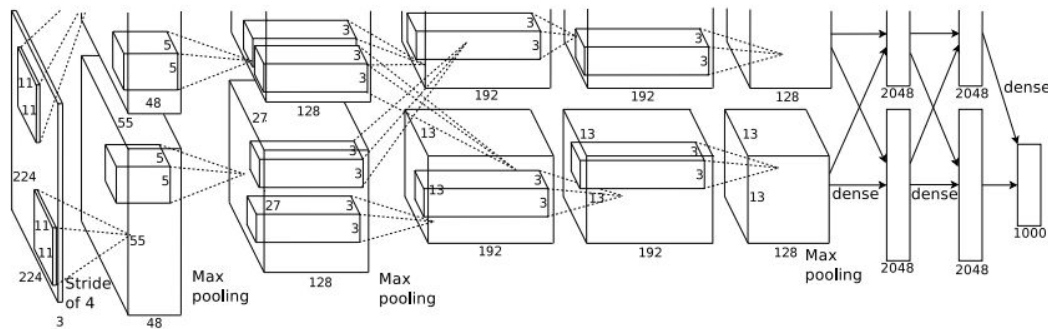
Extract features in the local coordinate system of each tracklet.



Accumulate into histograms, separately according to multiple spatio-temporal layouts.

Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4

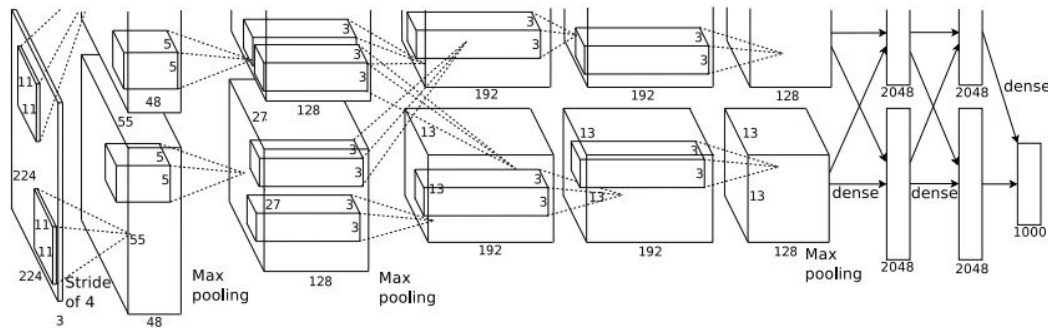
=>

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

=>

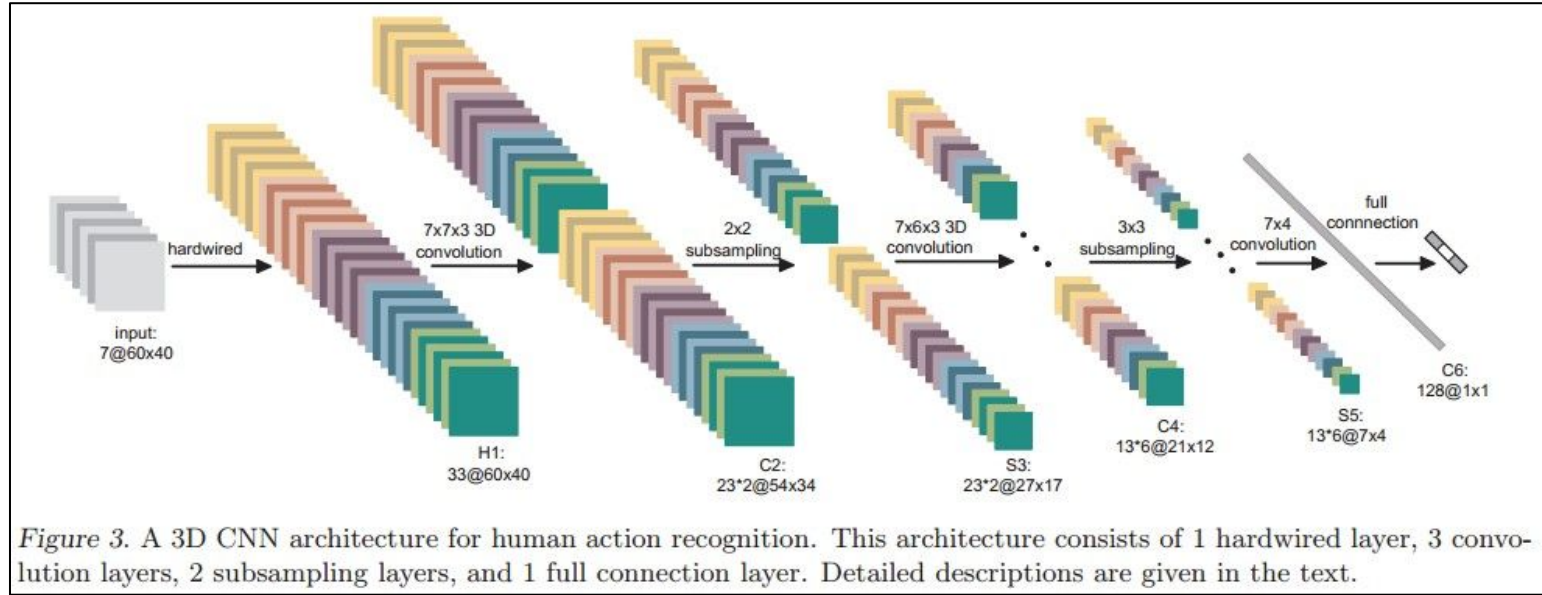
Output volume **[55x55x96]**

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!

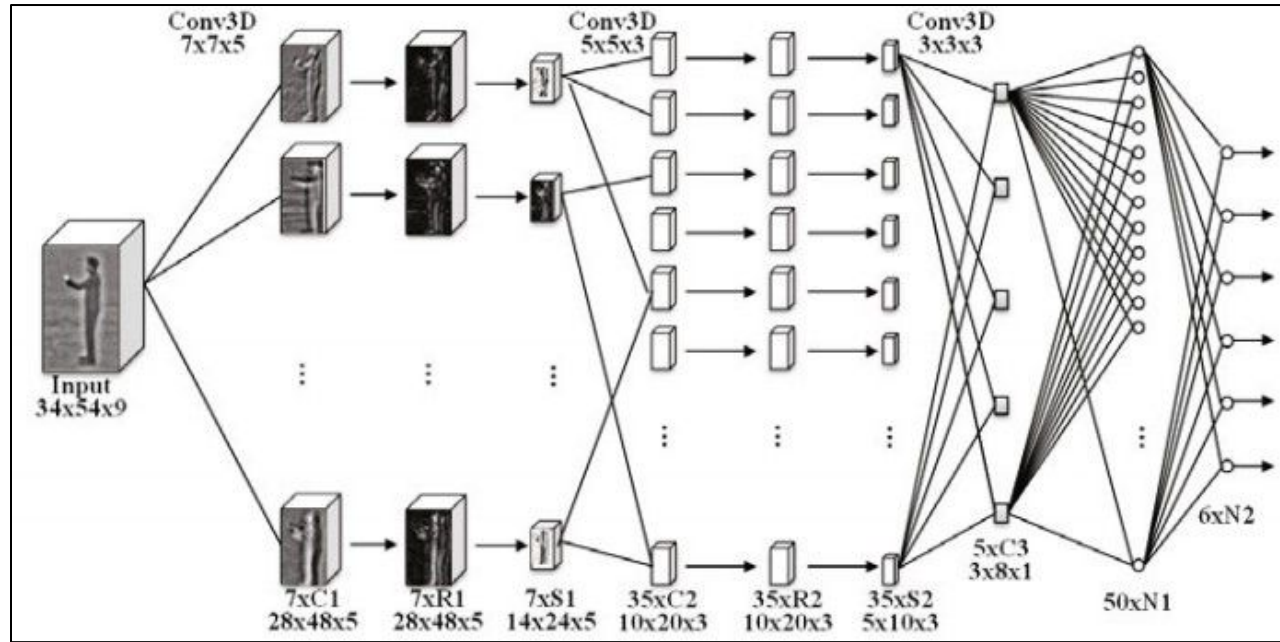
E.g. can have 11x11xT filters, where T = 2..15.

Spatio-Temporal ConvNets



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

Spatio-Temporal ConvNets

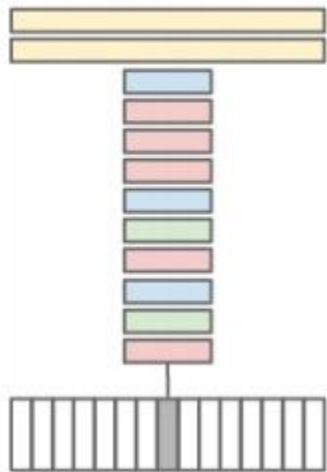


Sequential Deep Learning for Human Action Recognition, Baccouche et al., 2011

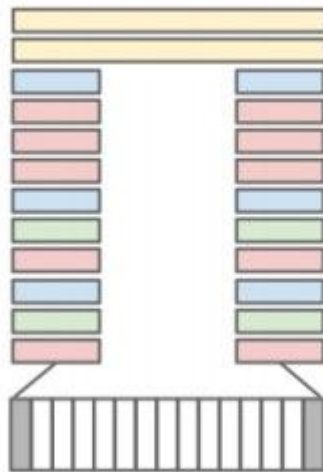
Spatio-Temporal ConvNets

spatio-temporal convolutions;
worked best.

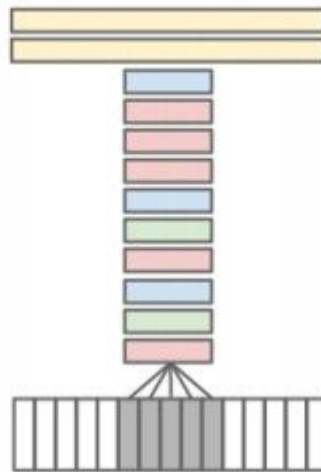
Single Frame



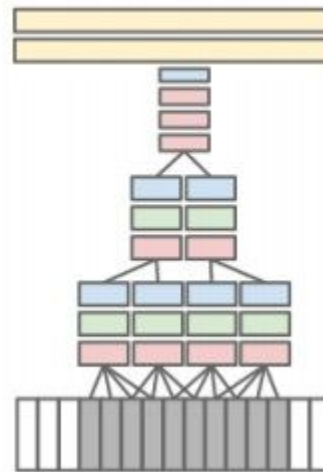
Late Fusion



Early Fusion



Slow Fusion



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

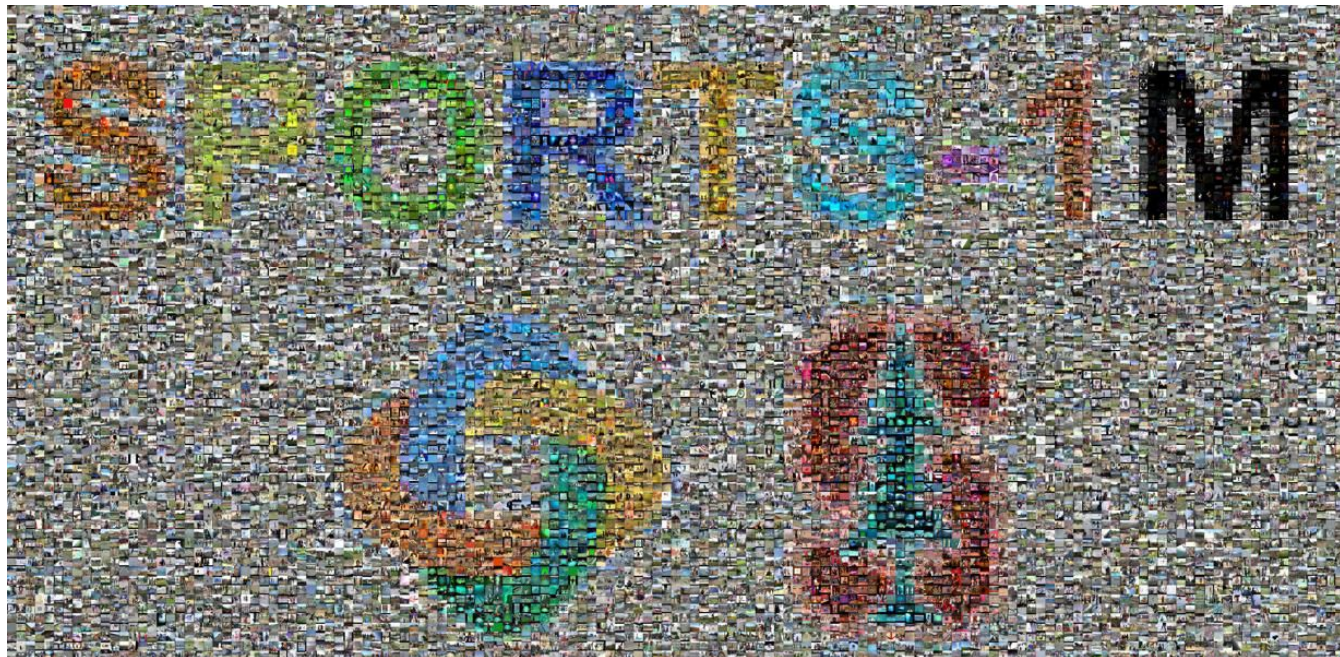
Spatio-Temporal ConvNets

Learned filters on
the first layer



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

Spatio-Temporal ConvNets



1 million videos
487 sports classes

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

Spatio-Temporal ConvNets

Model	Clip Hit@1	Video Hit@1	Video Hit@5
Feature Histograms + Neural Net	-	55.3	-
Single-Frame	41.1	59.3	77.7
Single-Frame + Multires	42.4	60.0	78.5
Single-Frame Fovea Only	30.0	49.9	72.8
Single-Frame Context Only	38.1	56.0	77.2
Early Fusion	38.9	57.7	76.8
Late Fusion	40.7	59.3	78.7
Slow Fusion	41.9	60.9	80.2
CNN Average (Single+Early+Late+Slow)	41.4	63.9	82.4

The motion information didn't add all that much...

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

Spatio-Temporal ConvNets



track cycling
cycling
track cycling
road bicycle racing
marathon
ultramarathon



ultramarathon
ultramarathon
half marathon
running
marathon
inline speed skating



heptathlon
heptathlon
decathlon
hurdles
pentathlon
sprint (running)



bikejoring
mushing
bikejoring
harness racing
skijoring
carting



longboarding
longboarding
aggressive inline skating
freestyle scootering
freeboard (skateboard)
sandboarding



ultimate (sport)
ultimate (sport)
hurling
flag football
association football
rugby sevens



demolition derby
demolition derby
monster truck
mud bogging
motocross
grand prix motorcycle racing



telemark skiing
snowboarding
telemark skiing
nordic skiing
ski touring
skijoring



whitewater kayaking
whitewater kayaking
rafting
kayaking
canoeing
adventure racing



arena football
indoor american football
arena football
canadian football
american football
women's lacrosse



reining
barrel racing
rodeo
reining
cowboy action shooting
bull riding



eight-ball
nine-ball
blackball (pool)
trick shot
eight-ball
straight pool

Spatio-Temporal ConvNets

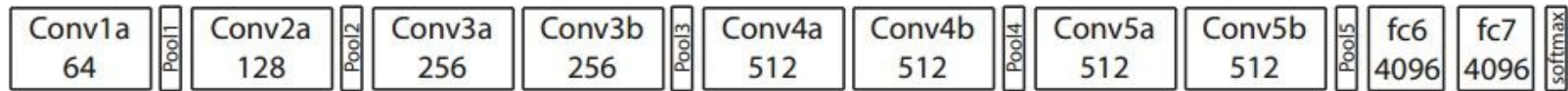
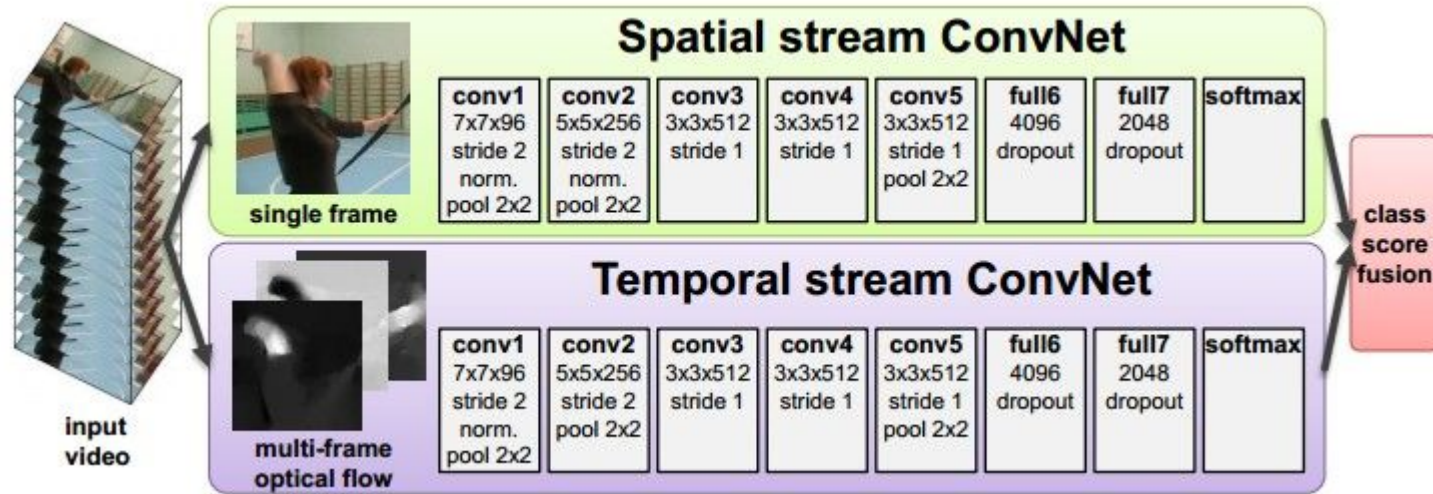


Figure 3. **C3D architecture.** C3D net has 8 convolution, 5 max-pooling, and 2 fully connected layers, followed by a softmax output layer. All 3D convolution kernels are $3 \times 3 \times 3$ with stride 1 in both spatial and temporal dimensions. Number of filters are denoted in each box. The 3D pooling layers are denoted from pool1 to pool5. All pooling kernels are $2 \times 2 \times 2$, except for pool1 is $1 \times 2 \times 2$. Each fully connected layer has 4096 output units.

3D VGGNet, basically.

[Learning Spatiotemporal Features with 3D Convolutional Networks, Tran et al. 2015]

Spatio-Temporal ConvNets

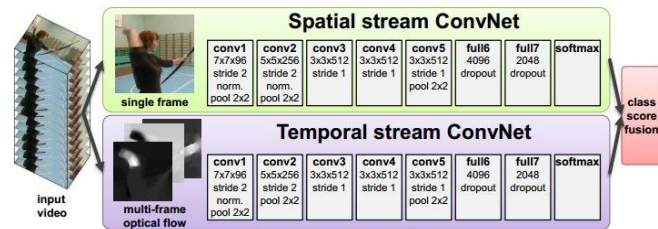


(of VGGNet fame)

[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.

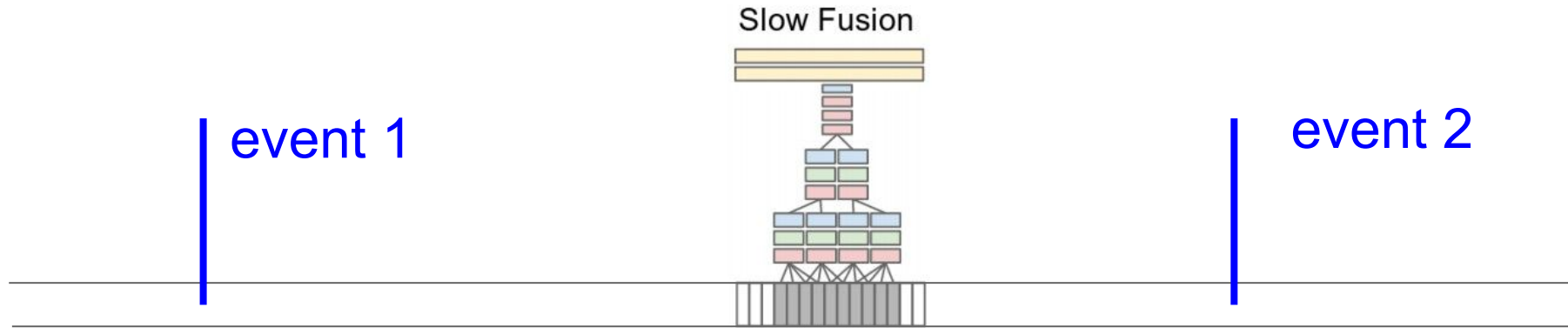
[Two-Stream Convolutional Networks for Action Recognition in Videos, **Simonyan** and Zisserman 2014]

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

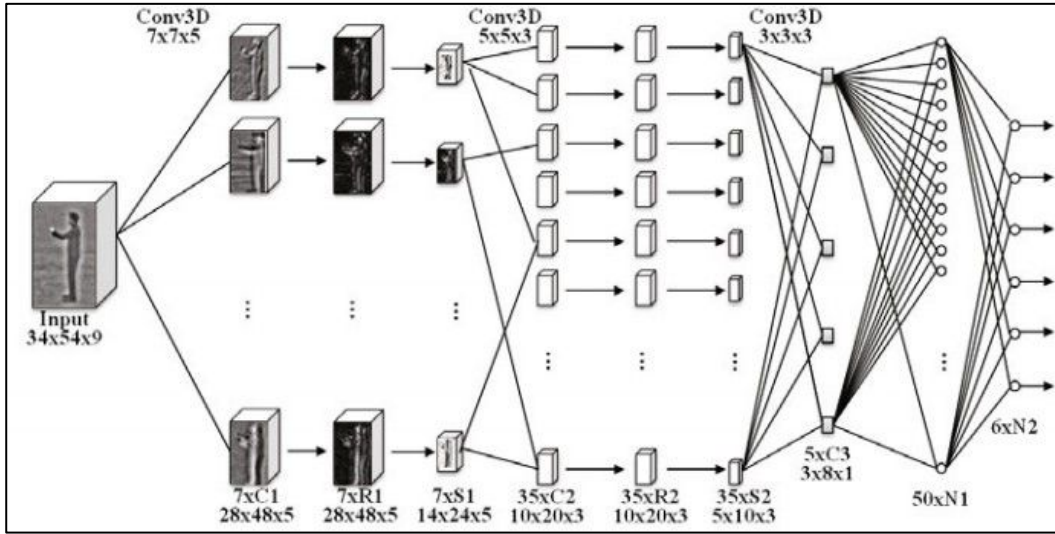
Long-time Spatio-Temporal ConvNets

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

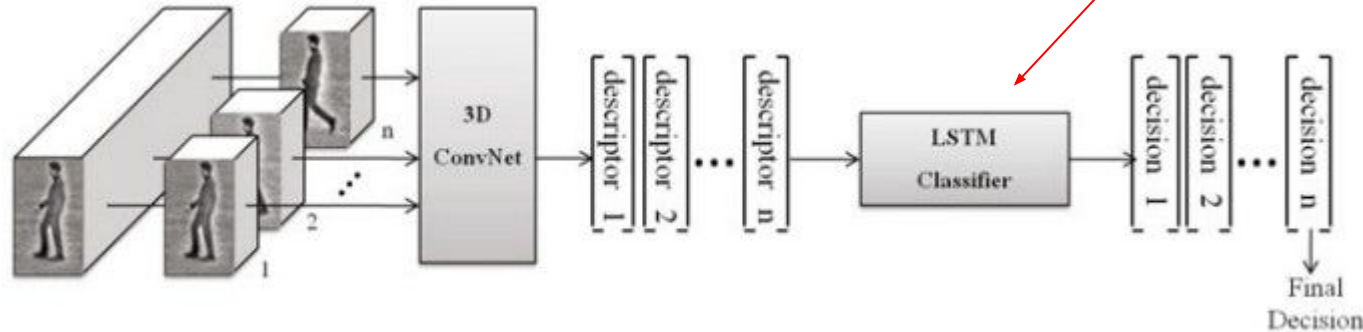


(This paper was way ahead of its time. Cited 65 times.)

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

Long-time Spatio-Temporal ConvNets

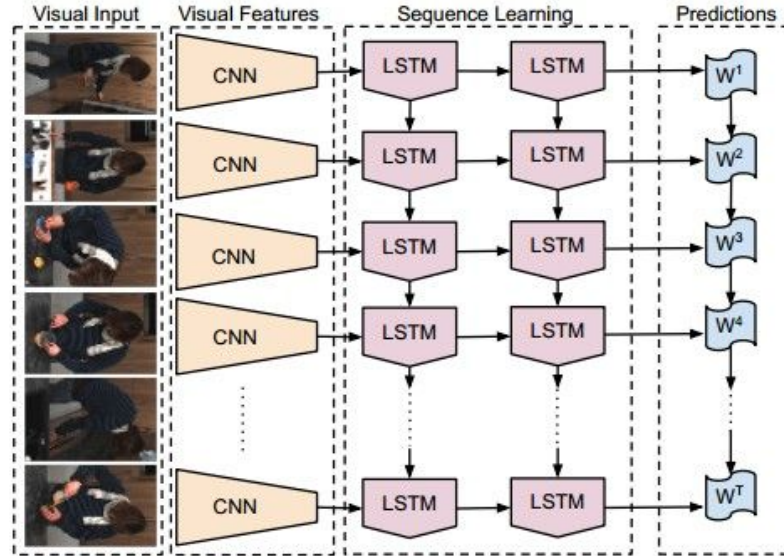
LSTM way before it was cool



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

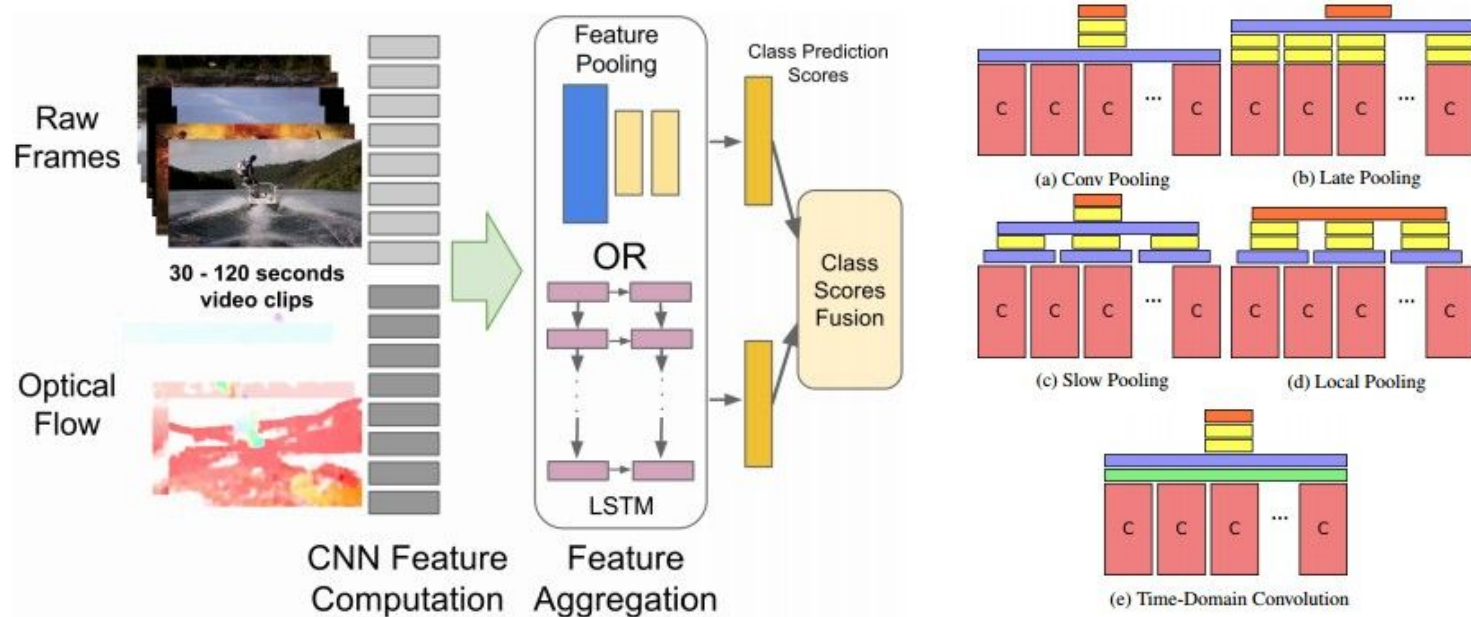
Sequential Deep Learning for Human Action Recognition, Baccouche et al., 2011

Long-time Spatio-Temporal ConvNets



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

Long-time Spatio-Temporal ConvNets



[Beyond Short Snippets: Deep Networks for Video Classification, Ng et al., 2015]

Summary so far

We looked at two types of architectural patterns:

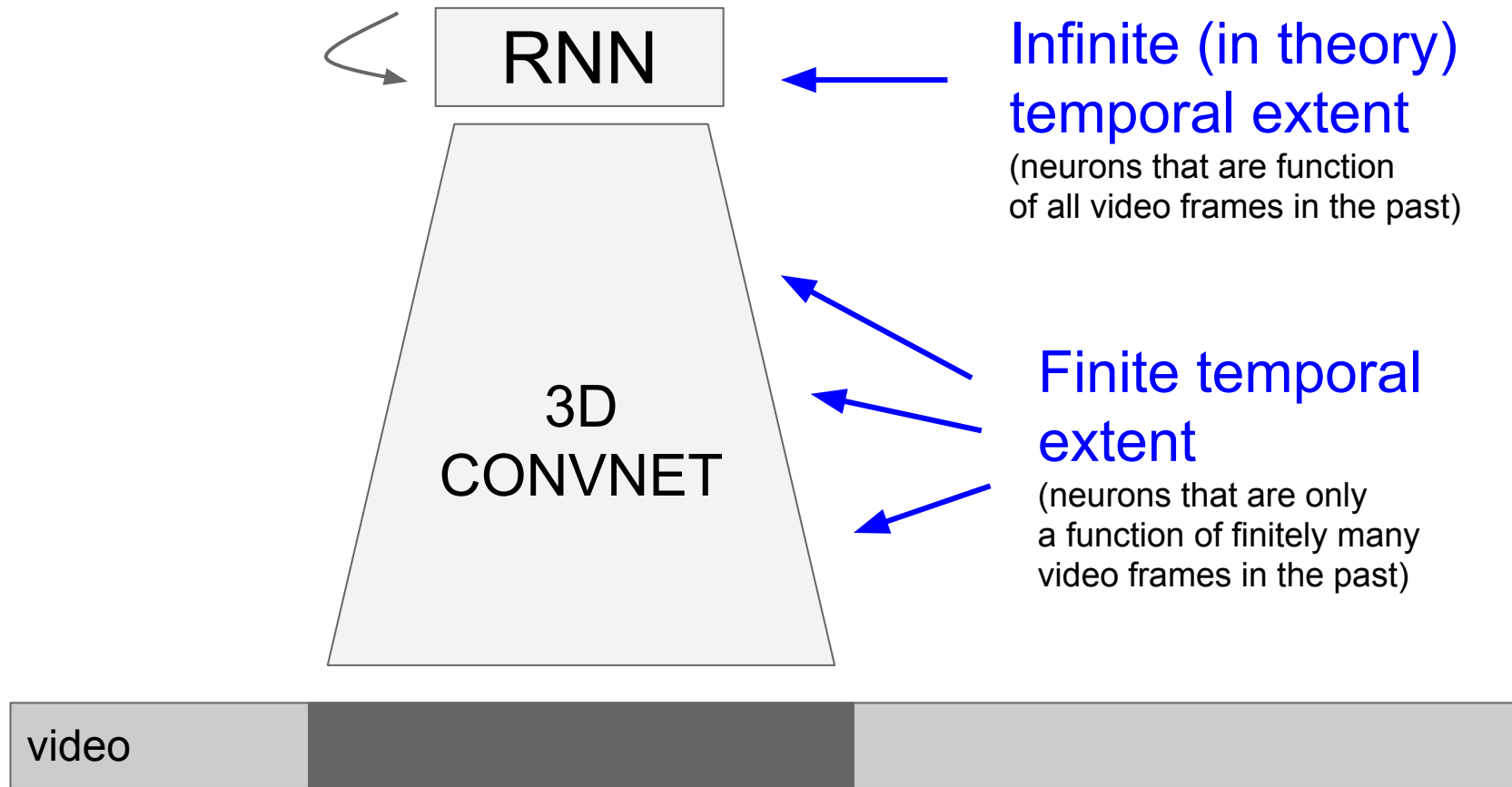
1. Model temporal motion locally (3D CONV)
 2. Model temporal motion globally (LSTM / RNN)
- + Fusions of both approaches at the same time.

Summary so far

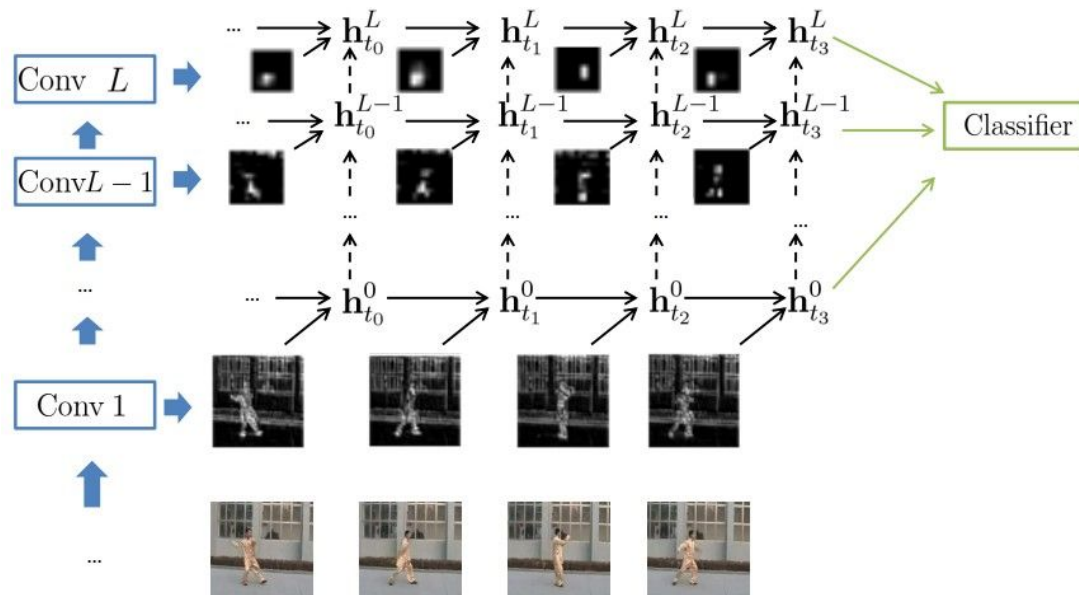
We looked at two types of architectural patterns:

1. Model temporal motion locally (3D CONV)
 2. Model temporal motion globally (LSTM / RNN)
- + Fusions of both approaches at the same time.

There is another (cleaner) way!



Long-time Spatio-Temporal ConvNets



Beautiful:

All neurons in the ConvNet are recurrent.

$$\mathbf{z}_t^l = \sigma(\mathbf{W}_z^l * \mathbf{x}_t^l + \mathbf{U}_z^l * \mathbf{h}_{t-1}^l),$$

$$\mathbf{r}_t^l = \sigma(\mathbf{W}_r^l * \mathbf{x}_t^l + \mathbf{U}_r^l * \mathbf{h}_{t-1}^l),$$

$$\tilde{\mathbf{h}}_t^l = \tanh(\mathbf{W}^l * \mathbf{x}_t^l + \mathbf{U} * (\mathbf{r}_t^l \odot \mathbf{h}_{t-1}^l)),$$

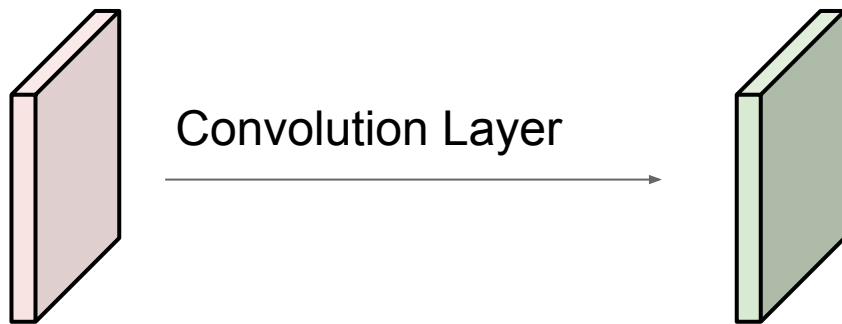
$$\mathbf{h}_t^l = (1 - \mathbf{z}_t^l) \mathbf{h}_{t-1}^l + \mathbf{z}_t^l \tilde{\mathbf{h}}_t^l,$$

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]

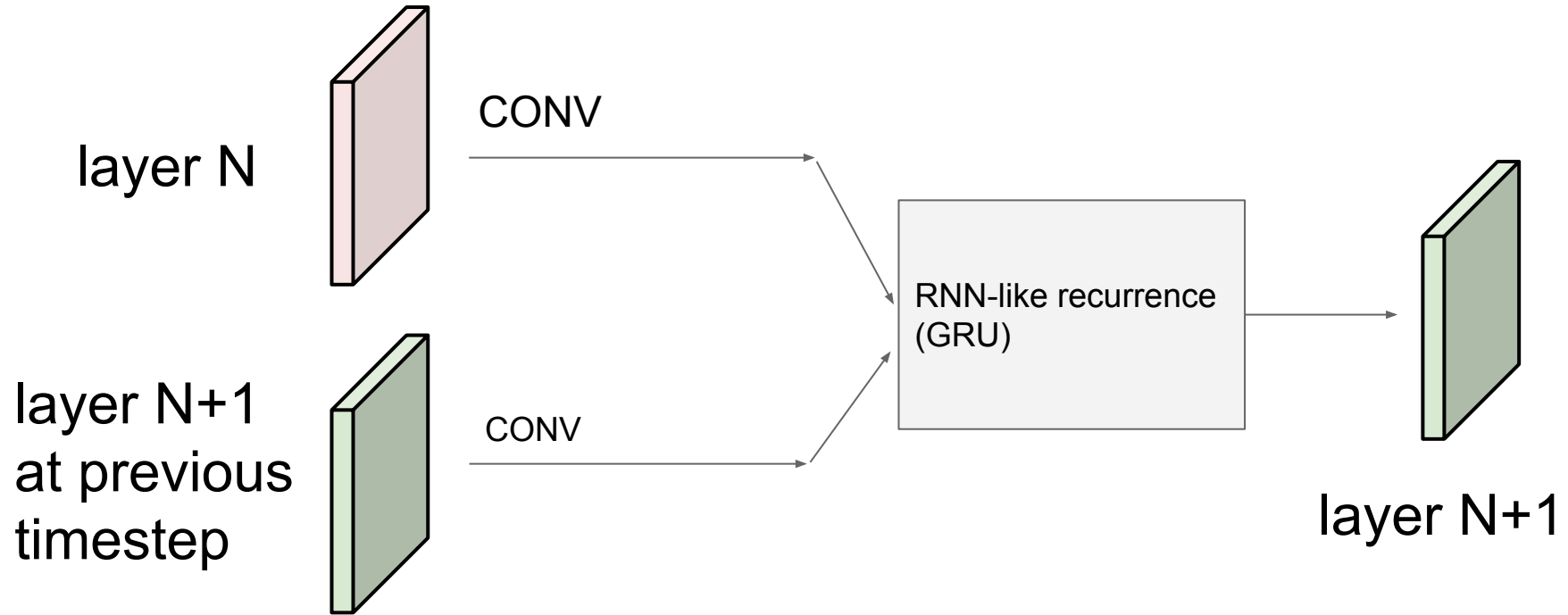
Long-time Spatio-Temporal ConvNets

Normal ConvNet:



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

Long-time Spatio-Temporal ConvNets



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

Long-time Spatio-Temporal ConvNets

Recall: RNNs

$$h_t = f_W(h_{t-1}, x_t)$$

Vanilla RNN

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

GRU

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

LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$
$$\begin{aligned} c_t^l &= f \odot c_{t-1}^l + i \odot g \\ h_t^l &= o \odot \tanh(c_t^l) \end{aligned}$$

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

Long-time Spatio-Temporal ConvNets

Recall: RNNs

$$h_t = f_W(h_{t-1}, x_t)$$

Matrix multiply

=>

CONV



GRU

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}),$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}),$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W} \mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \mathbf{h}_{t-1} + \mathbf{z}_t \tilde{\mathbf{h}}_t,$$

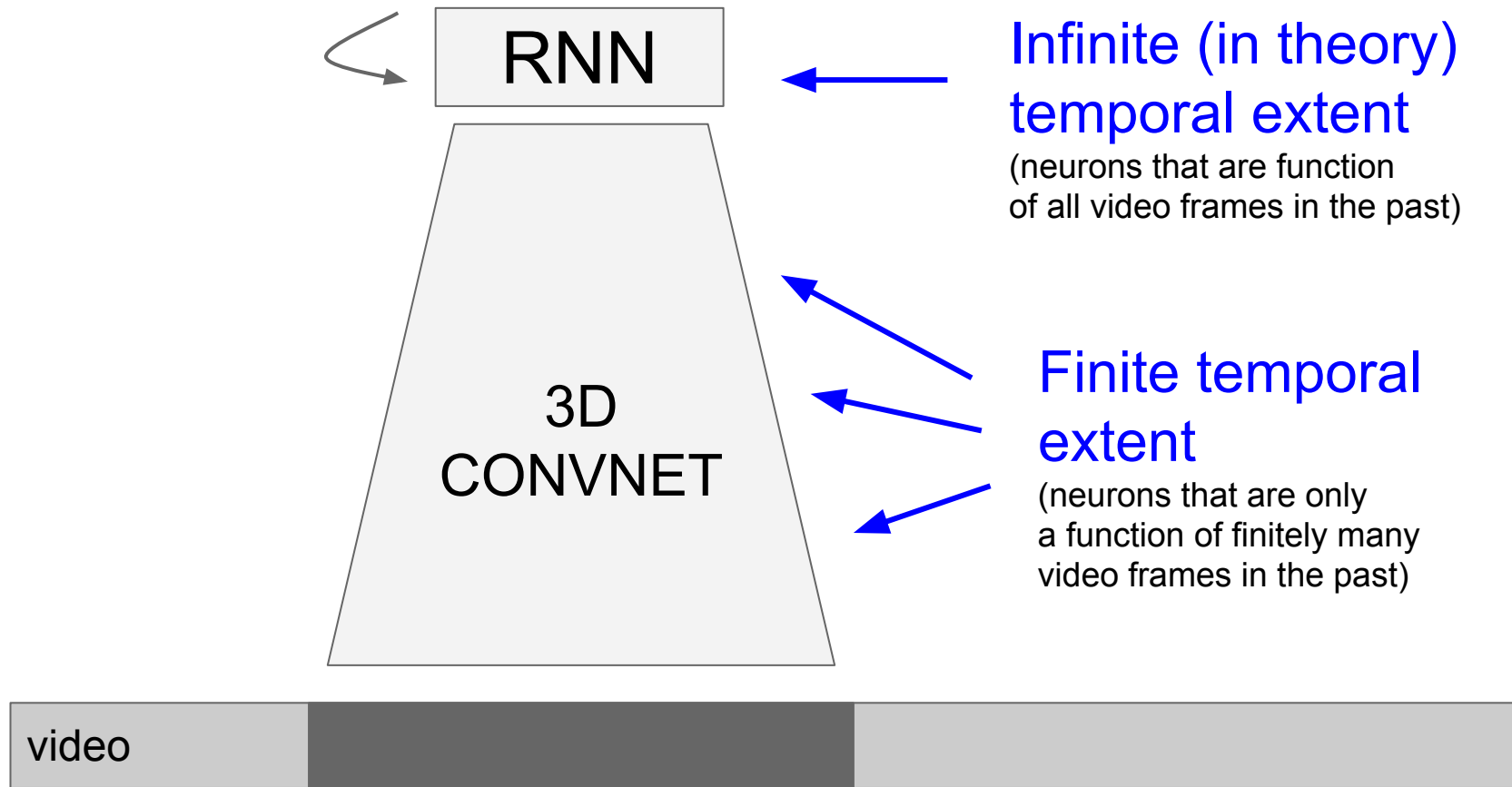
$$\mathbf{z}_t^l = \sigma(\mathbf{W}_z^l * \mathbf{x}_t^l + \mathbf{U}_z^l * \mathbf{h}_{t-1}^l),$$

$$\mathbf{r}_t^l = \sigma(\mathbf{W}_r^l * \mathbf{x}_t^l + \mathbf{U}_r^l * \mathbf{h}_{t-1}^l),$$

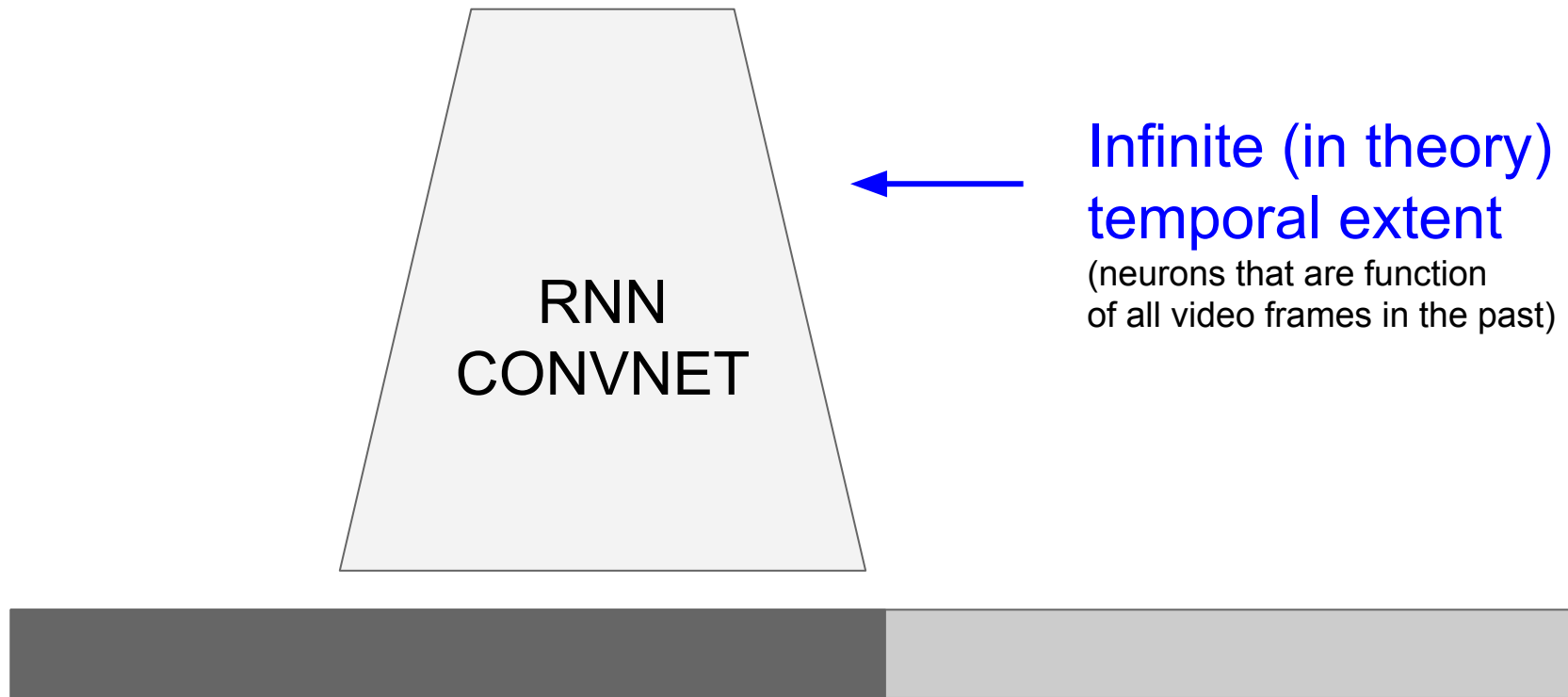
$$\tilde{\mathbf{h}}_t^l = \tanh(\mathbf{W}^l * \mathbf{x}_t^l + \mathbf{U} * (\mathbf{r}_t^l \odot \mathbf{h}_{t-1}^l)),$$

$$\mathbf{h}_t^l = (1 - \mathbf{z}_t^l) \mathbf{h}_{t-1}^l + \mathbf{z}_t^l \tilde{\mathbf{h}}_t^l,$$

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



i.e. we obtain:



Summary

- You think you need a Spatio-Temporal Fancy Video ConvNet
- STOP. Do you really?
- Okay fine: do you want to model:
 - local motion? (use 3D CONV), or
 - global motion? (use LSTM).
- Try out using Optical Flow in a second stream (can work better sometimes)
- Try out GRU-RCN! (imo best model)

Unsupervised Learning

Unsupervised Learning Overview

- Definitions
- 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 $x \rightarrow y$

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 $x \rightarrow y$

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc

Unsupervised Learning

Data: x

Just data, no labels!

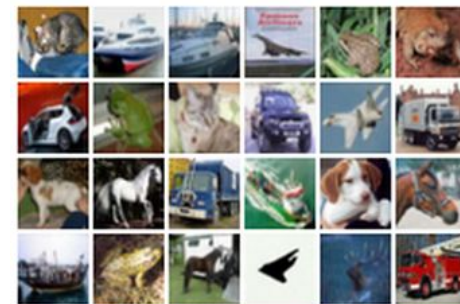
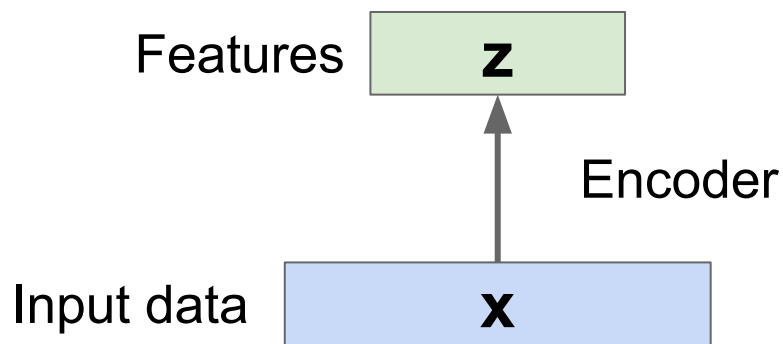
Goal: Learn some *structure* of the data

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

Unsupervised Learning

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

Autoencoders

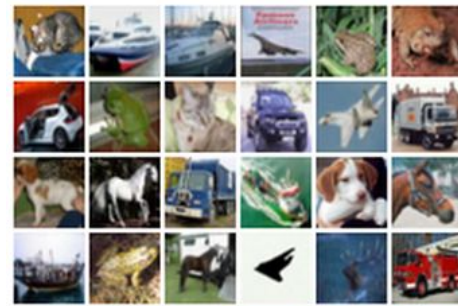
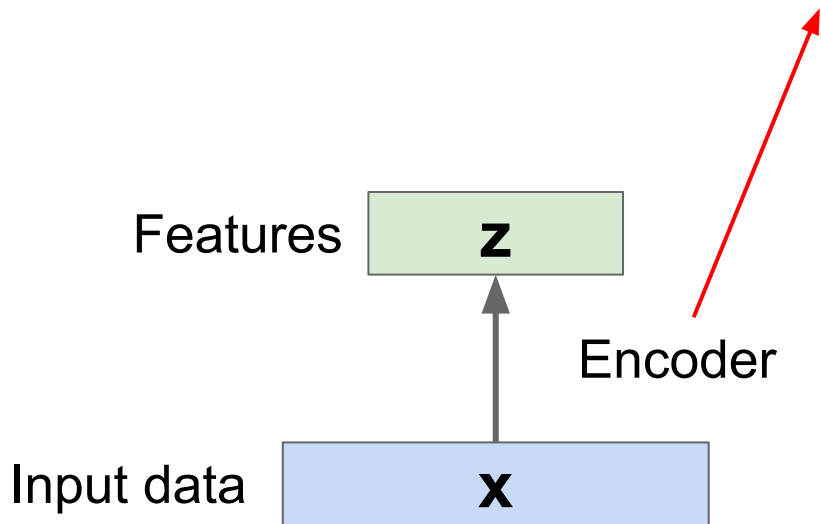


Autoencoders

Originally: Linear + nonlinearity (sigmoid)

Later: Deep, fully-connected

Later: ReLU CNN



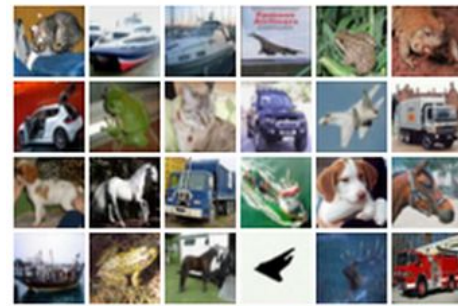
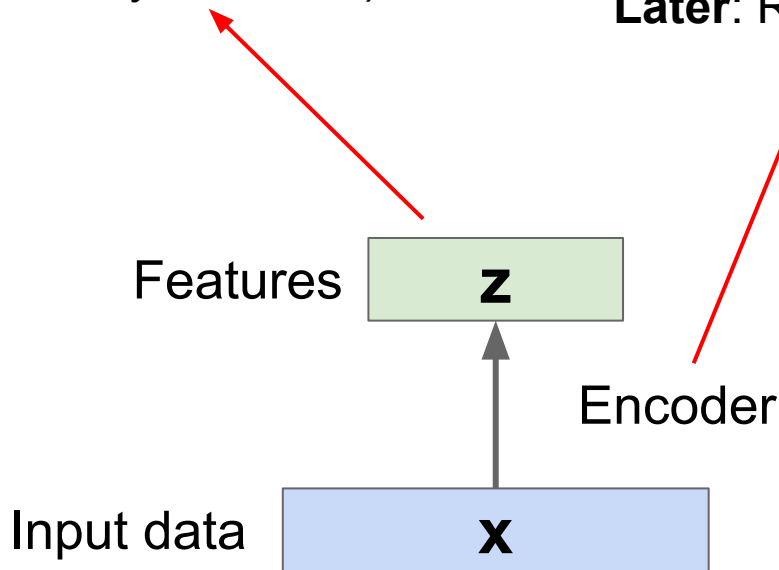
Autoencoders

z usually smaller than x
(dimensionality reduction)

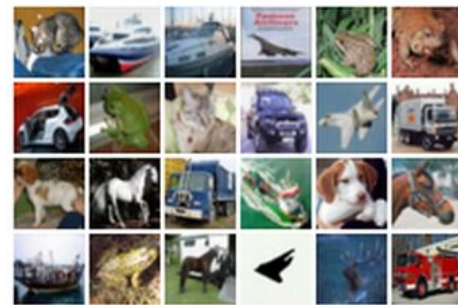
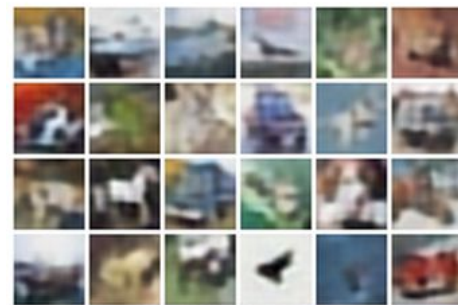
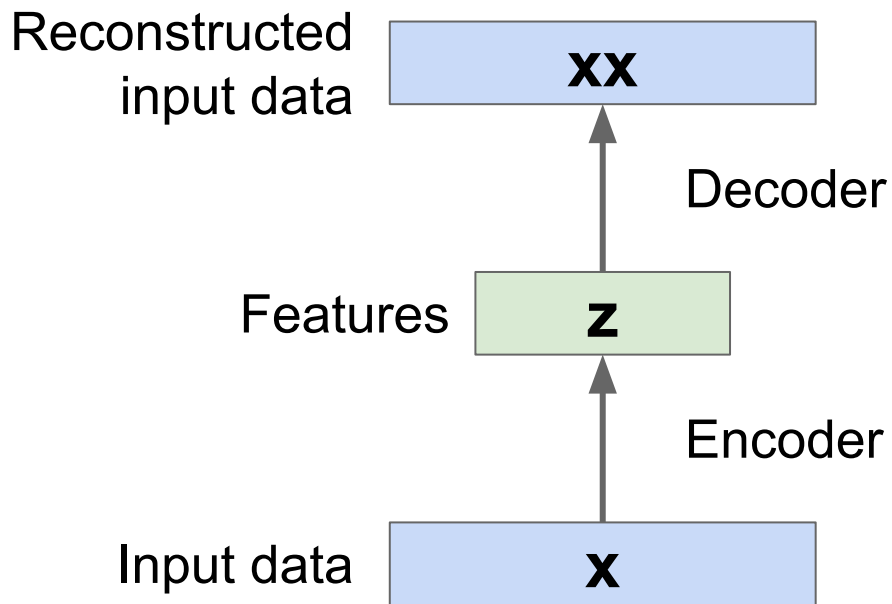
Originally: Linear + nonlinearity (sigmoid)

Later: Deep, fully-connected

Later: ReLU CNN



Autoencoders

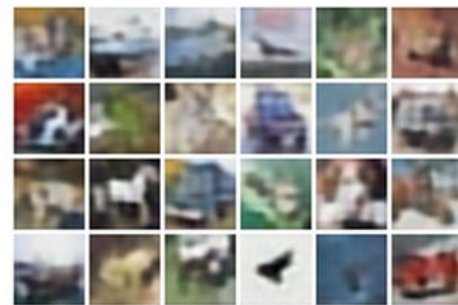
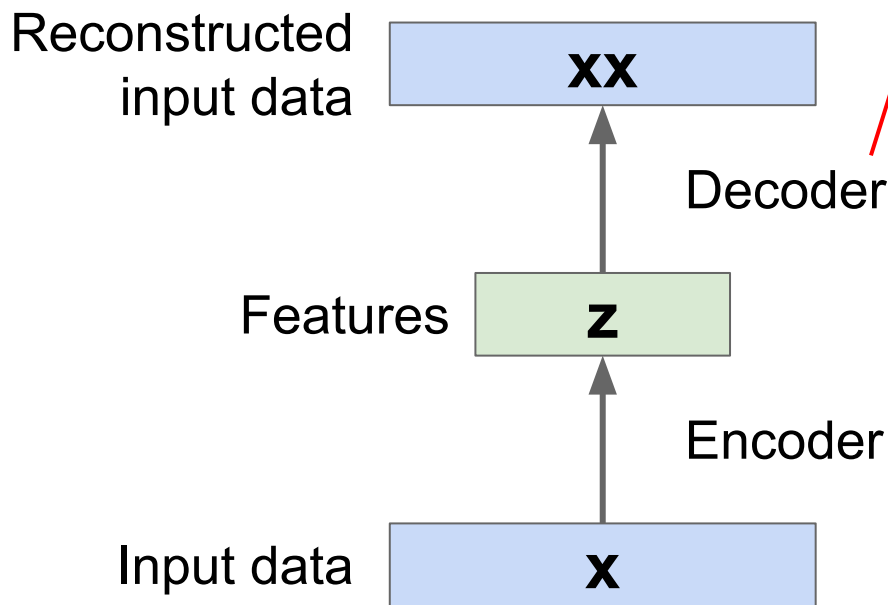


Autoencoders

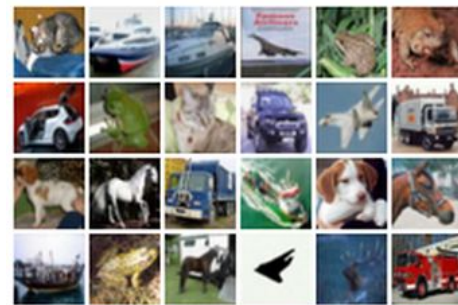
Originally: Linear +
nonlinearity (sigmoid)

Later: Deep, fully-connected

Later: ReLU CNN (upconv)



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

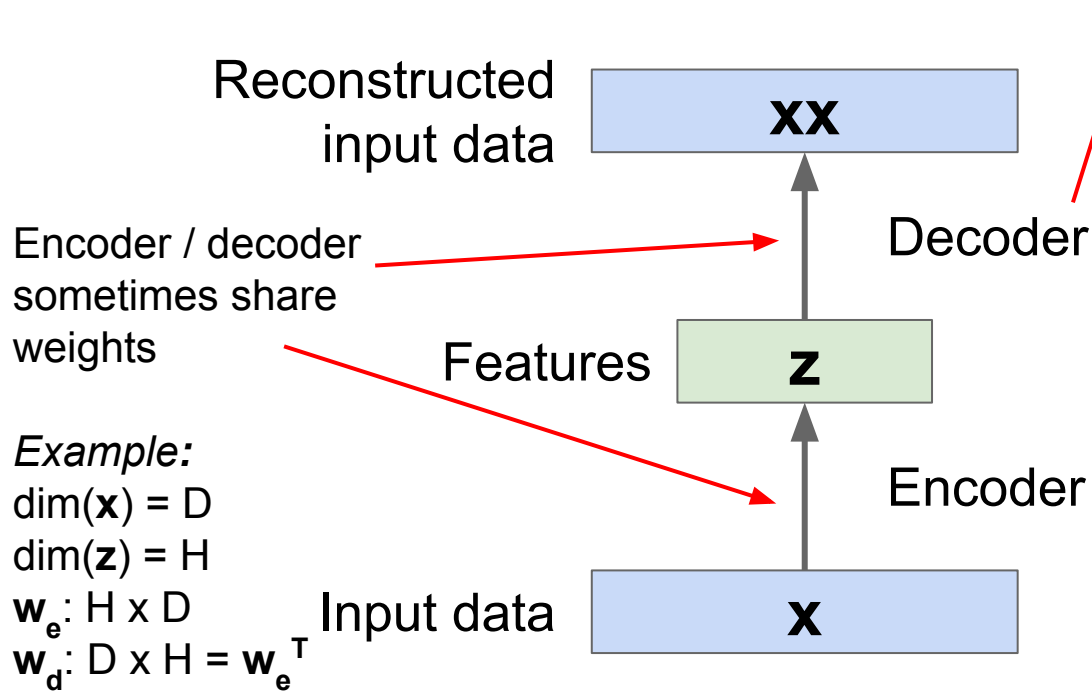


Autoencoders

Originally: Linear +
nonlinearity (sigmoid)

Later: Deep, fully-connected

Later: ReLU CNN (upconv)



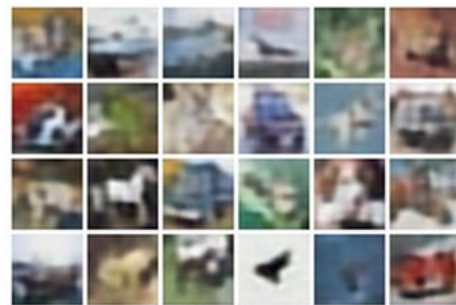
Example:

$\dim(\mathbf{x}) = D$

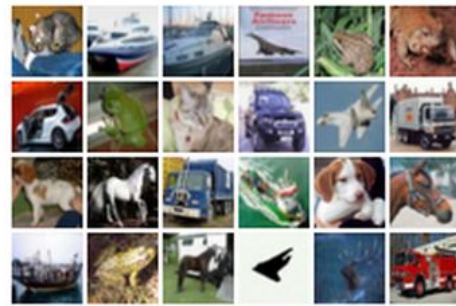
$\dim(\mathbf{z}) = H$

$\mathbf{w}_e: H \times D$

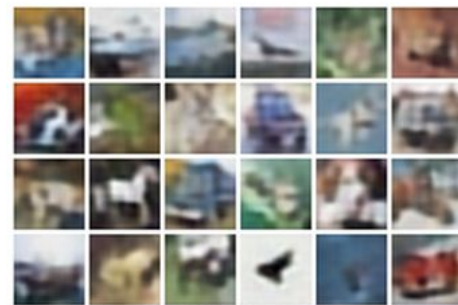
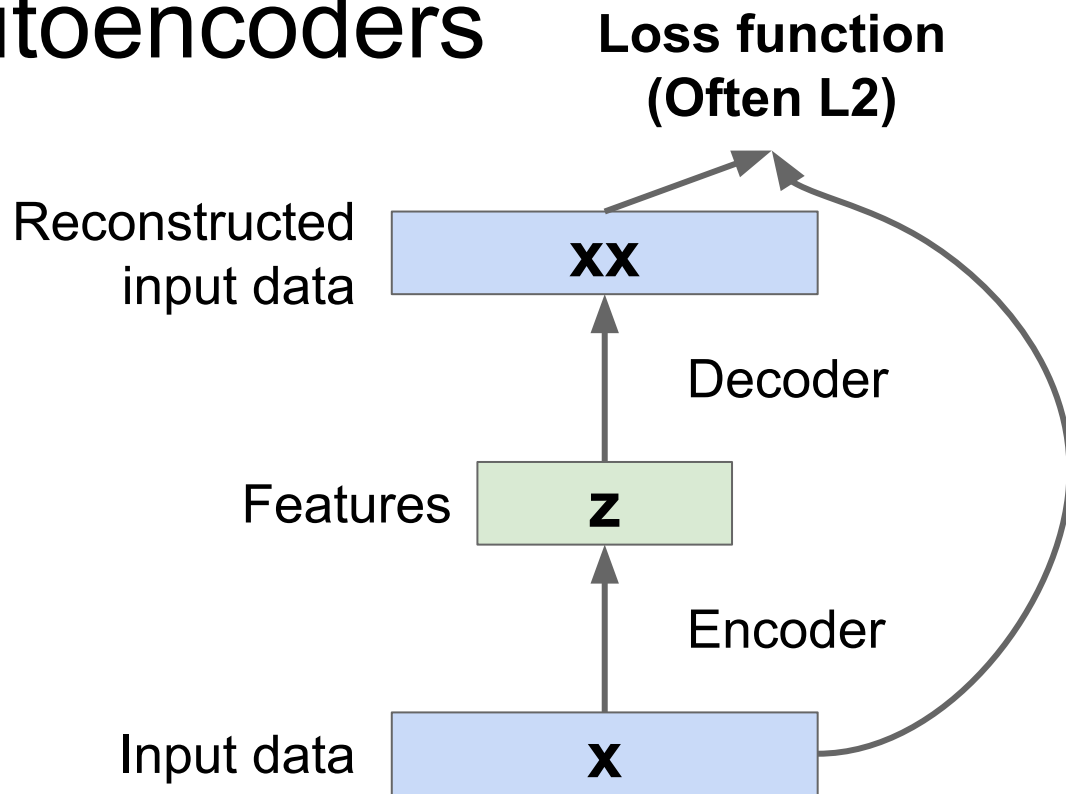
$\mathbf{w}_d: D \times H = \mathbf{w}_e^T$



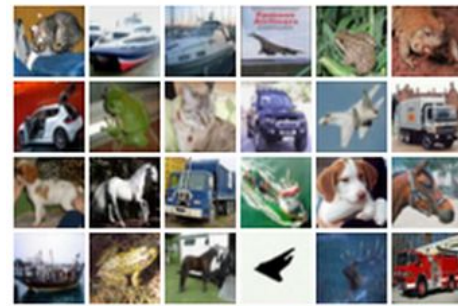
Train for
reconstruction
with no labels!



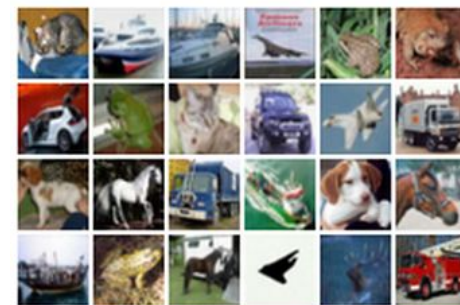
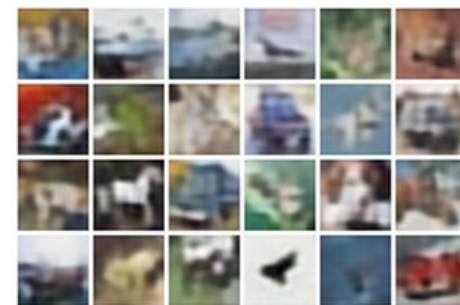
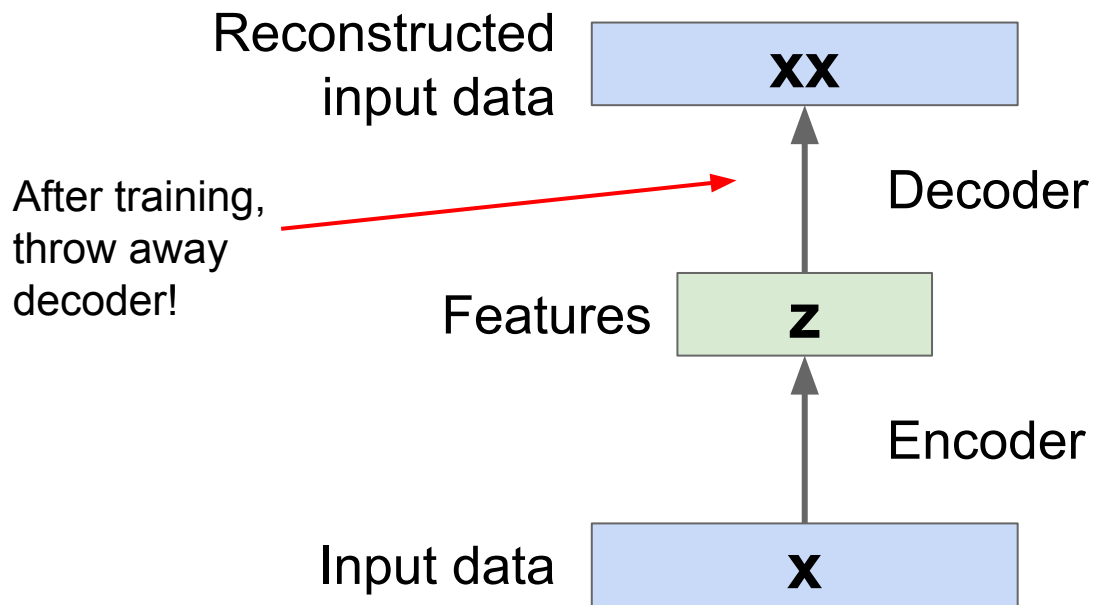
Autoencoders



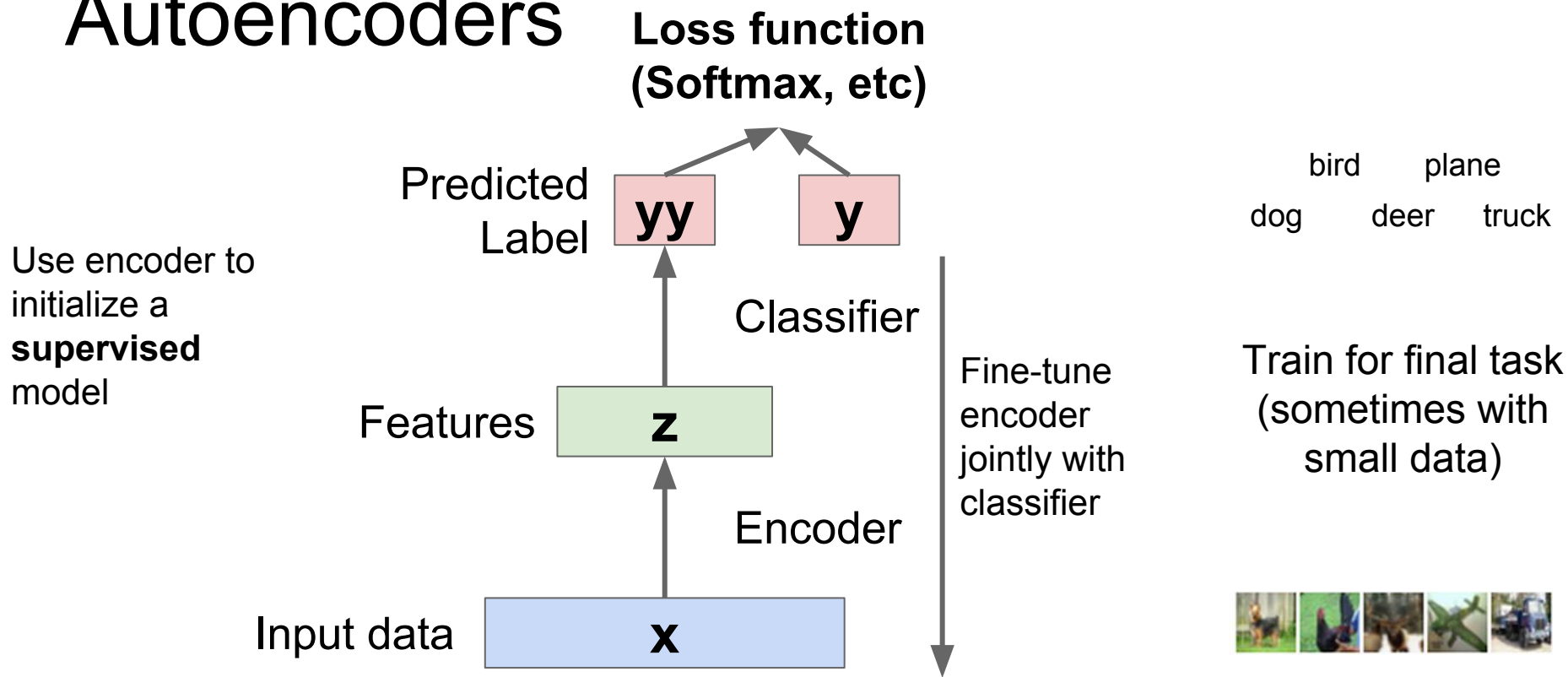
Train for
reconstruction
with no labels!



Autoencoders



Autoencoders

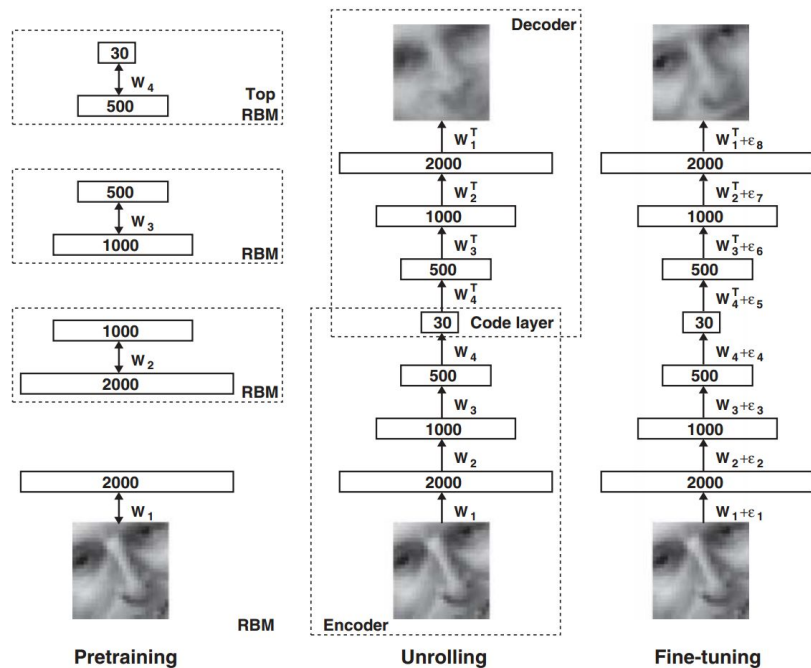


Autoencoders: Greedy Training

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



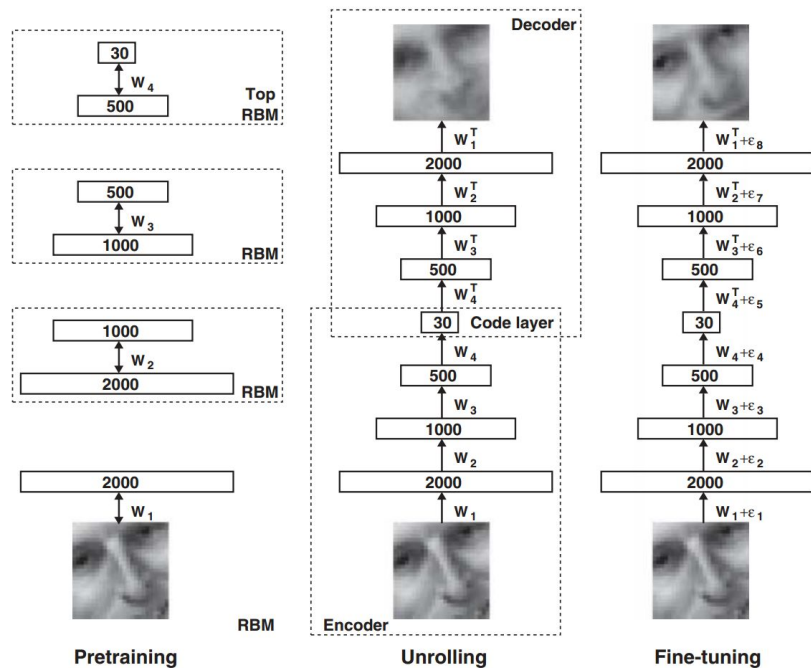
Hinton and Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks", Science 2006

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

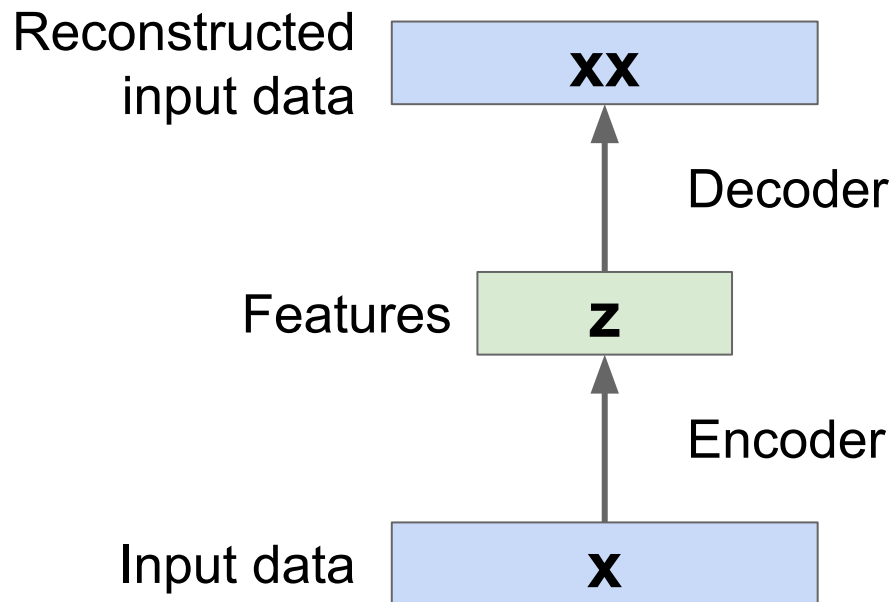


Not common anymore

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

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

Autoencoders



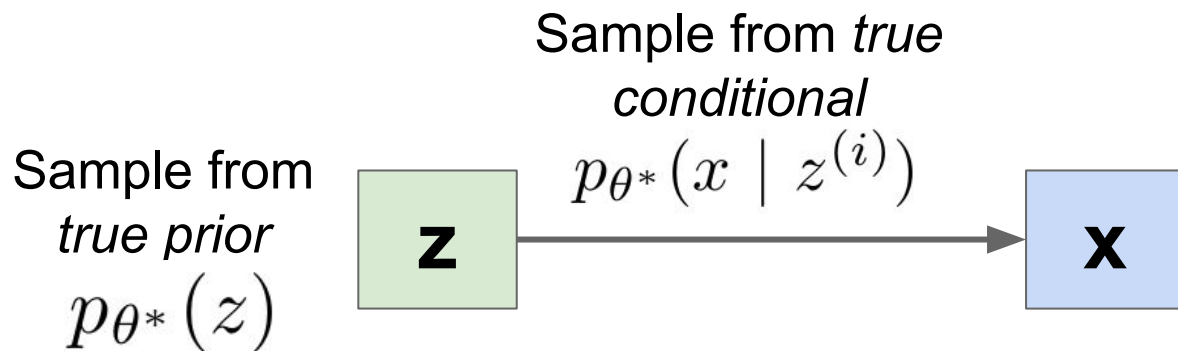
Autoencoders can reconstruct data, and can learn features to initialize a supervised model

Can we generate images from an autoencoder?

Variational Autoencoder

A Bayesian spin on an autoencoder - lets us generate data!

Assume our data $\{x^{(i)}\}_{i=1}^N$ is generated like this:



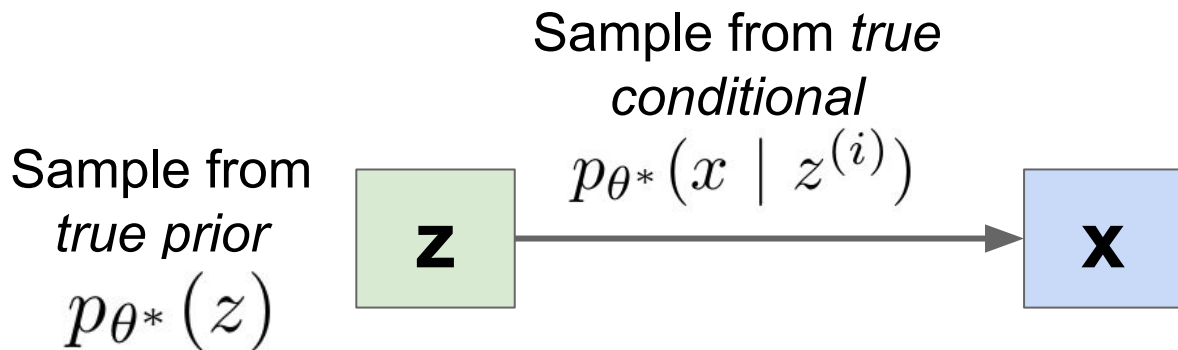
Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoder

A Bayesian spin on an autoencoder!

Assume our data $\{x^{(i)}\}_{i=1}^N$ is generated like this:

Intuition: \mathbf{x} is an image, \mathbf{z} gives class, orientation, attributes, etc

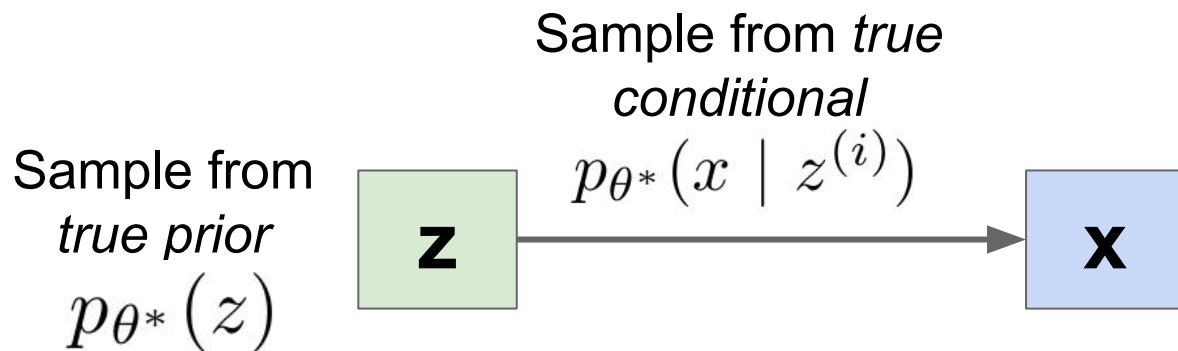


Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoder

A Bayesian spin on an autoencoder!

Assume our data $\{x^{(i)}\}_{i=1}^N$ is generated like this:



Intuition: \mathbf{x} is an image, \mathbf{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

Prior: Assume $p_{\theta}(z)$
is a unit Gaussian

Variational Autoencoder

Prior: Assume $p_{\theta}(z)$
is a unit Gaussian

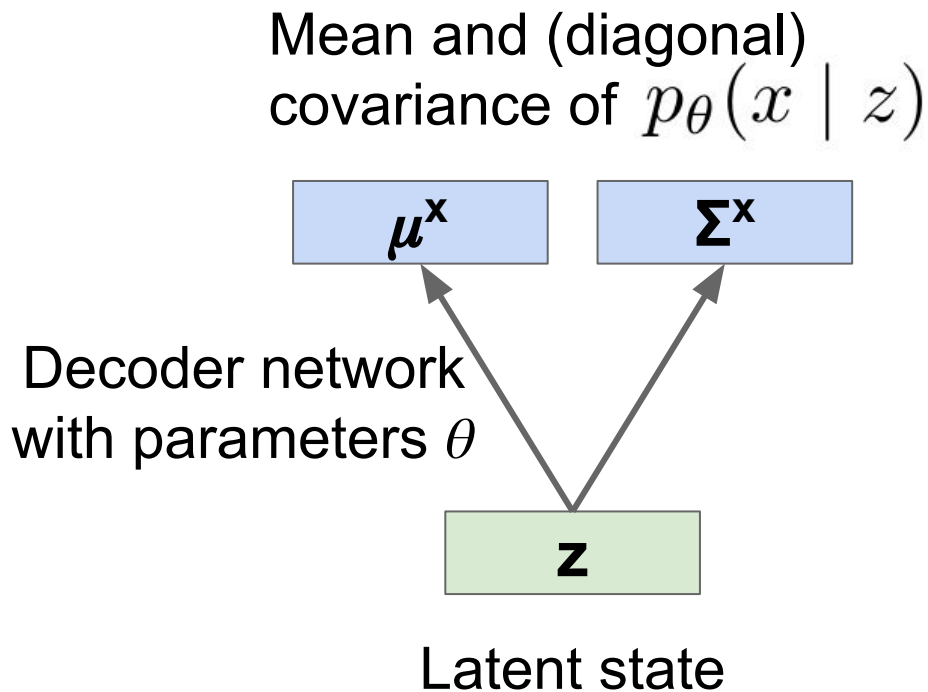
Conditional: Assume
 $p_{\theta}(x | z)$ is a
diagonal Gaussian,
predict mean and
variance with neural
net

Kingma and Welling, ICLR 2014

Variational Autoencoder

Prior: Assume $p_{\theta}(z)$ is a unit Gaussian

Conditional: Assume $p_{\theta}(x | z)$ is a diagonal Gaussian, predict mean and variance with neural net

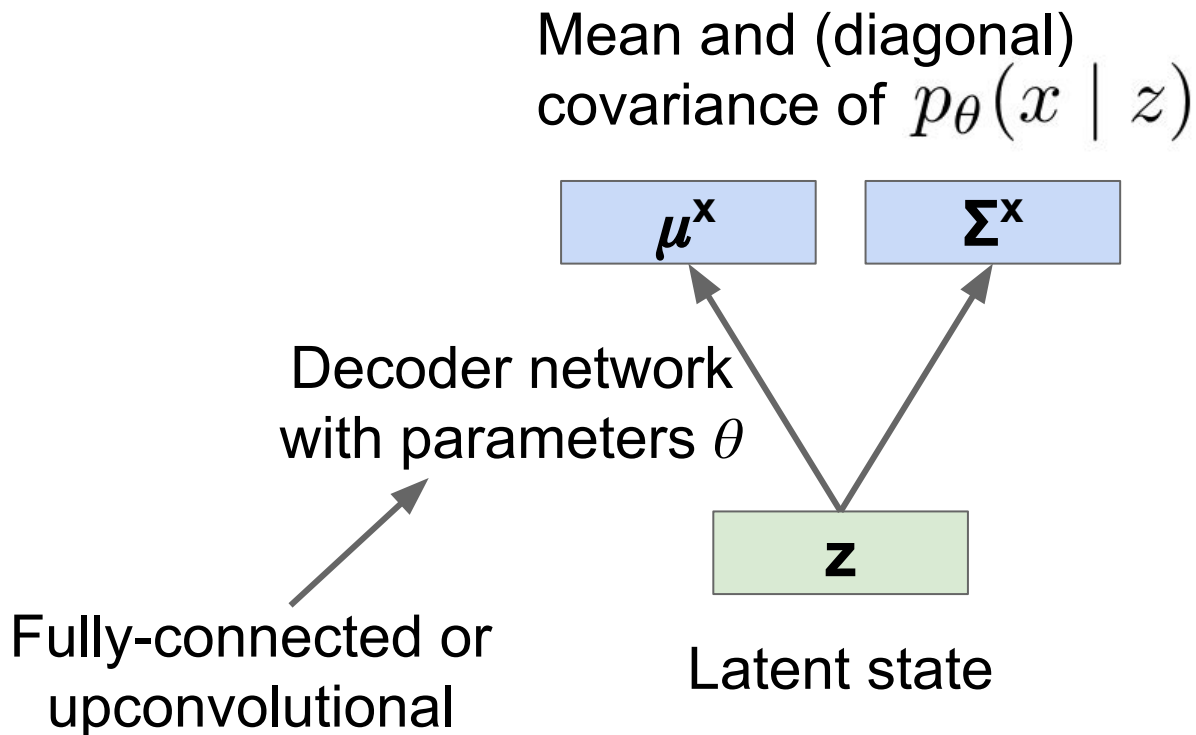


Kingma and Welling, ICLR 2014

Variational Autoencoder

Prior: Assume $p_{\theta}(z)$ is a unit Gaussian

Conditional: Assume $p_{\theta}(x | z)$ is a diagonal Gaussian, predict mean and variance with neural net



Kingma and Welling, ICLR 2014

Variational Autoencoder: Encoder

By Bayes Rule the posterior is:

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

Kingma and Welling,
ICLR 2014

Variational Autoencoder: Encoder

By Bayes Rule the posterior is:

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

Use decoder network =)

Gaussian =)

Intractable integral =(

Kingma and Welling,
ICLR 2014

Variational Autoencoder: Encoder

By Bayes Rule the posterior is:

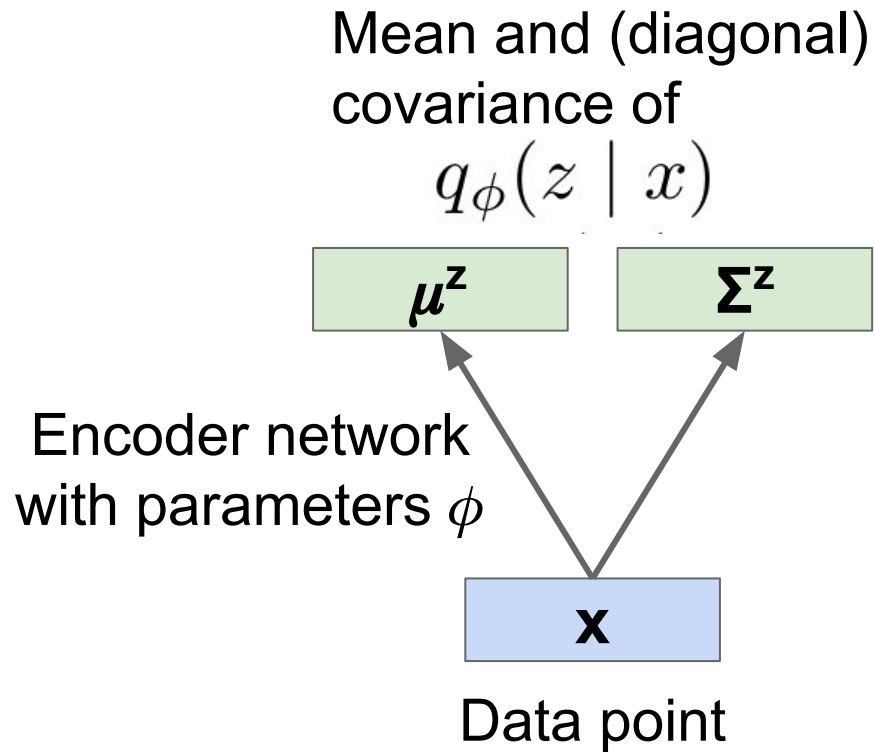
$$p_{\theta}(z | x) = \frac{p_{\theta}(x | z) p_{\theta}(z)}{p_{\theta}(x)}$$

Use decoder network =)

Gaussian =)

Intractable integral =)

Kingma and Welling,
ICLR 2014



Variational Autoencoder: Encoder

By Bayes Rule the posterior is:

$$p_{\theta}(z | x) = \frac{p_{\theta}(x | z) p_{\theta}(z)}{p_{\theta}(x)}$$

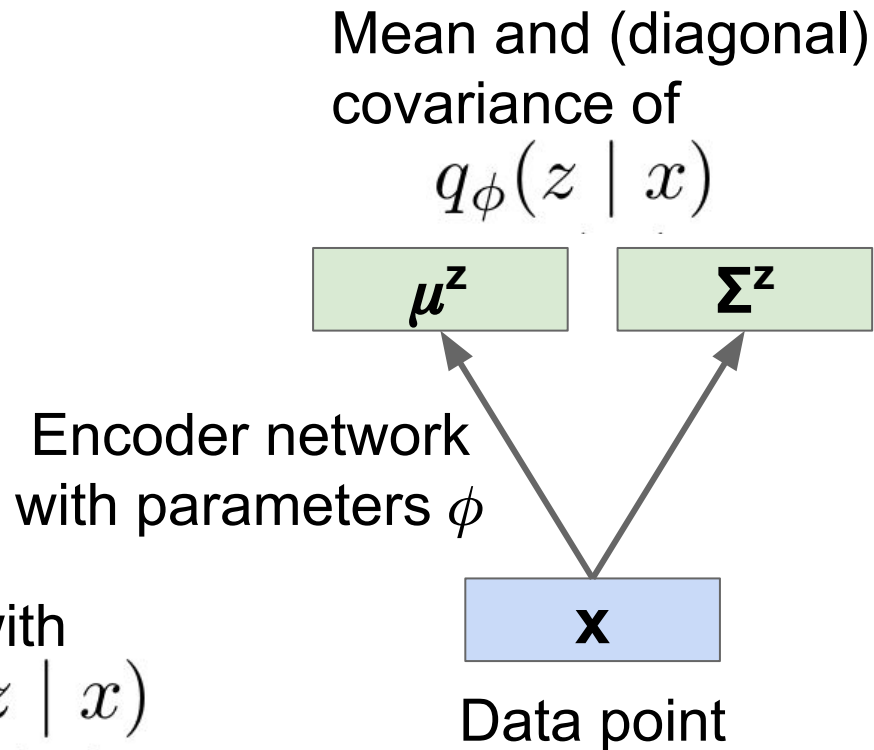
Use decoder network =)

Gaussian =)

Intractible integral =)

Approximate posterior with
encoder network $q_{\phi}(z | x)$

Kingma and Welling,
ICLR 2014



Variational Autoencoder: Encoder

By Bayes Rule the posterior is:

$$p_{\theta}(z | x) = \frac{p_{\theta}(x | z) p_{\theta}(z)}{p_{\theta}(x)}$$

Use decoder network =)

Gaussian =)

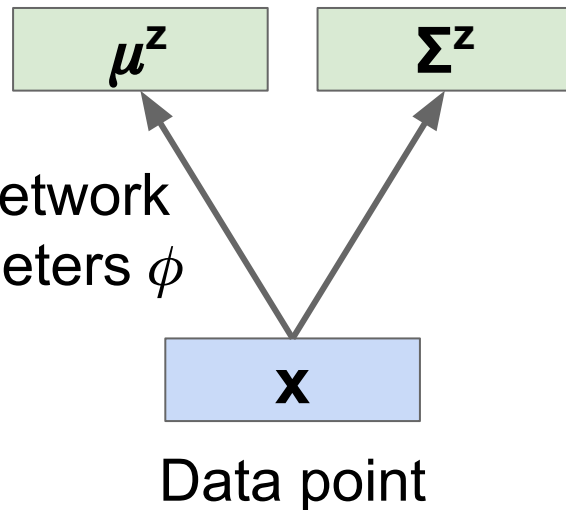
Intractible integral =)

Approximate posterior with
encoder network $q_{\phi}(z | x)$

Fully-connected
or convolutional

Encoder network
with parameters ϕ

Mean and (diagonal)
covariance of
 $q_{\phi}(z | x)$



Variational Autoencoder

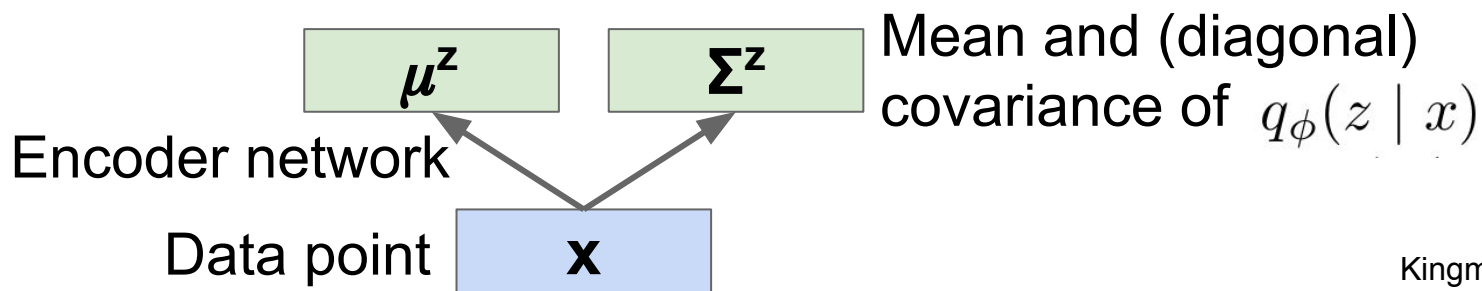
Data point



x

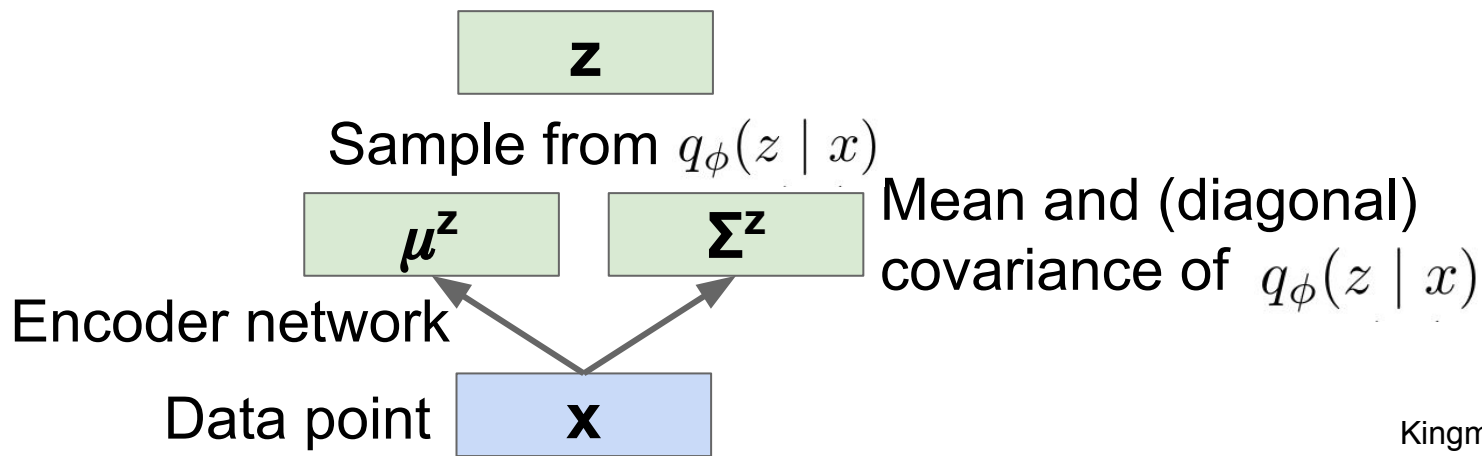
Kingma and Welling, ICLR 2014

Variational Autoencoder



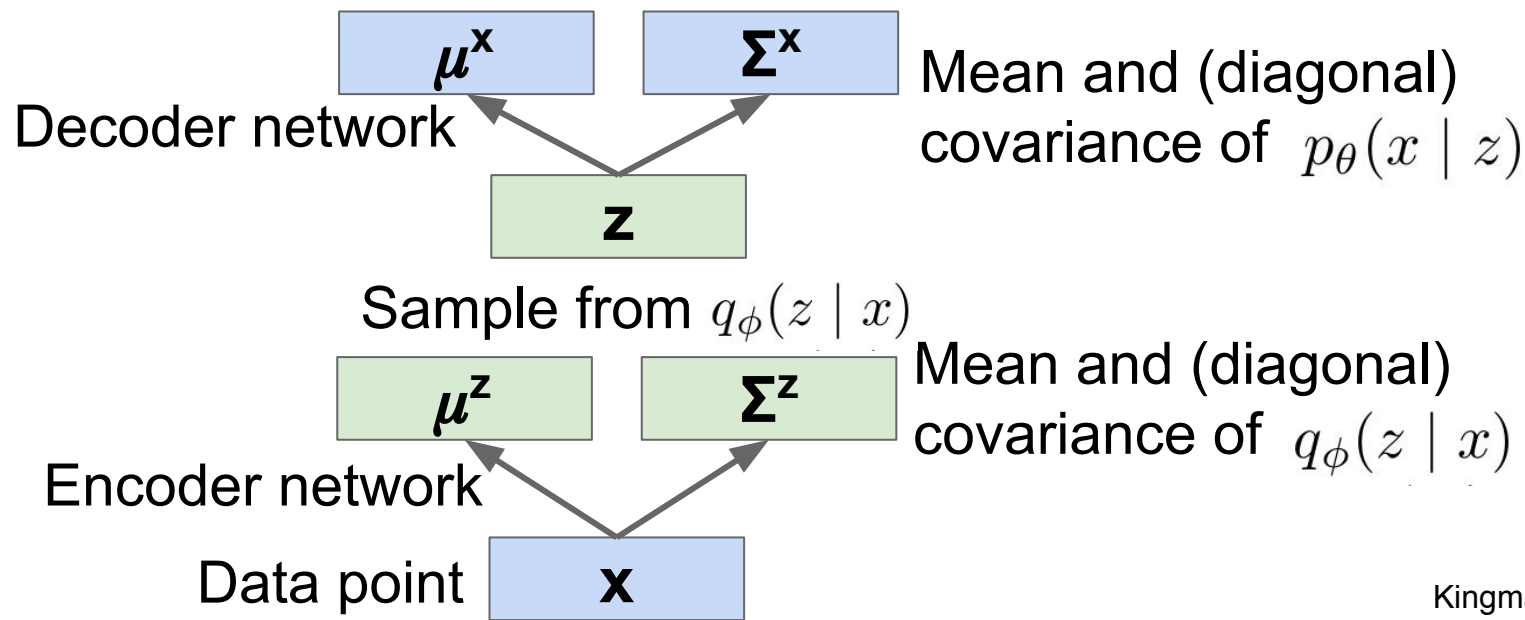
Kingma and Welling, ICLR 2014

Variational Autoencoder



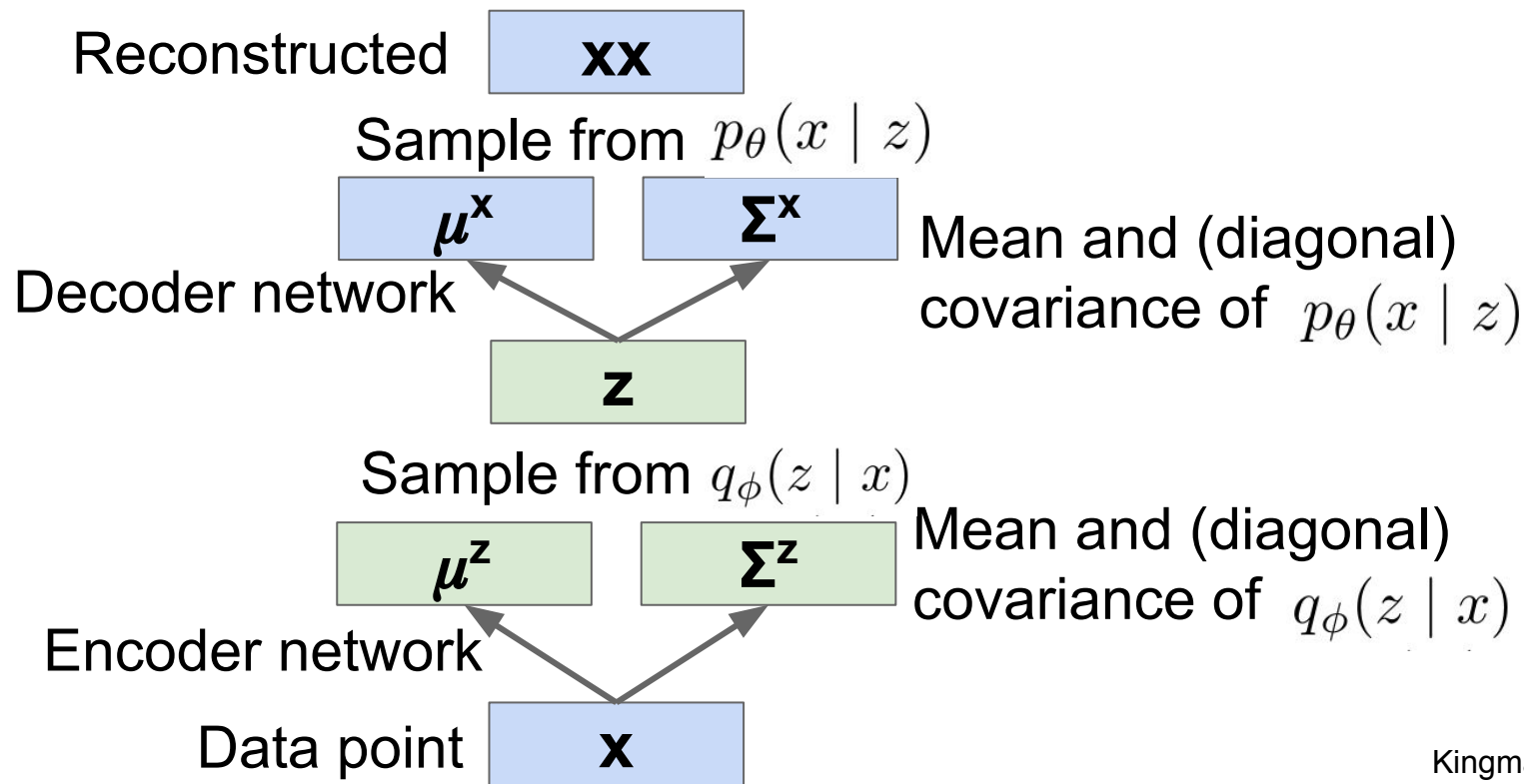
Kingma and Welling, ICLR 2014

Variational Autoencoder



Kingma and Welling, ICLR 2014

Variational Autoencoder



Kingma and Welling, ICLR 2014

Variational Autoencoder

Reconstructed **xx**

Sample from $p_{\theta}(x | z)$

μ^x

Σ^x

Decoder network

z

Sample from $q_{\phi}(z | x)$

μ^z

Σ^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 $p_{\theta}(x | z)$

(should be close to data x)

Mean and (diagonal)
covariance of $q_{\phi}(z | x)$

**(should be close
to prior $p_{\theta}(z)$)**

Kingma and Welling, ICLR 2014

Variational Autoencoder: Generate Data!

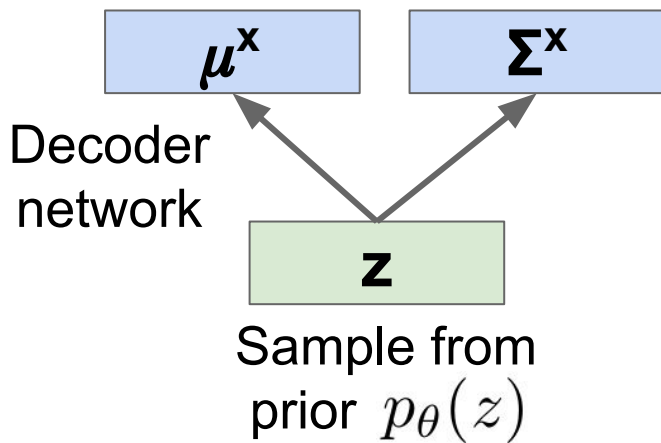
After network is trained:

z

Sample from
prior $p_{\theta}(z)$

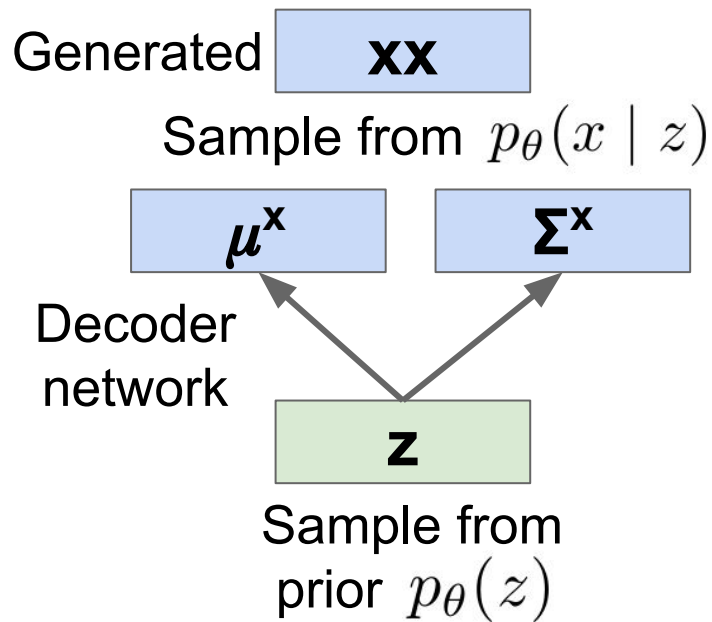
Variational Autoencoder: Generate Data!

After network is trained:



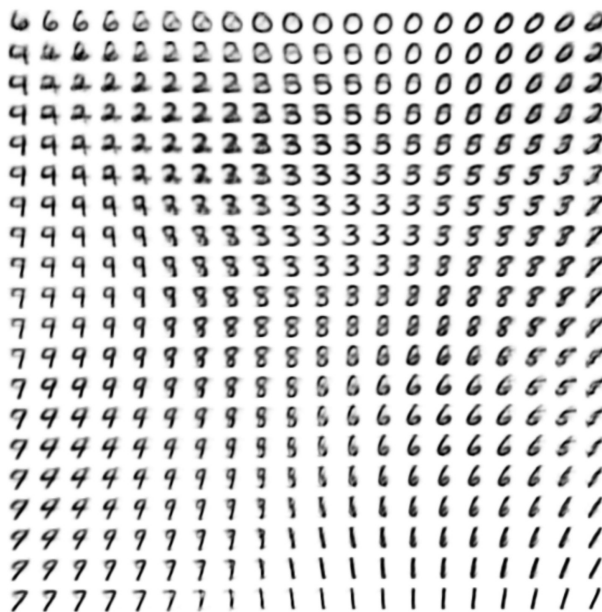
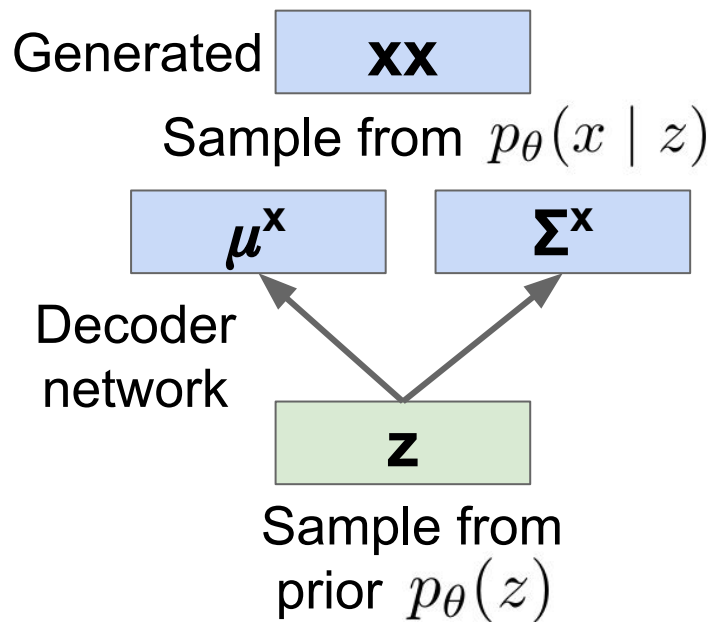
Variational Autoencoder: Generate Data!

After network is trained:



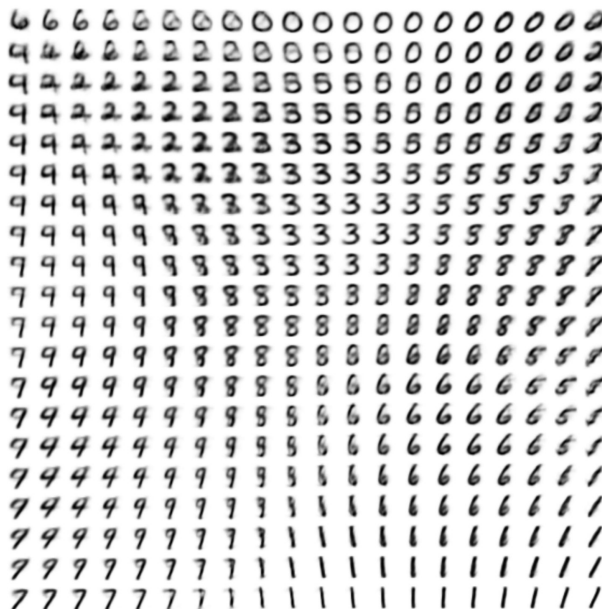
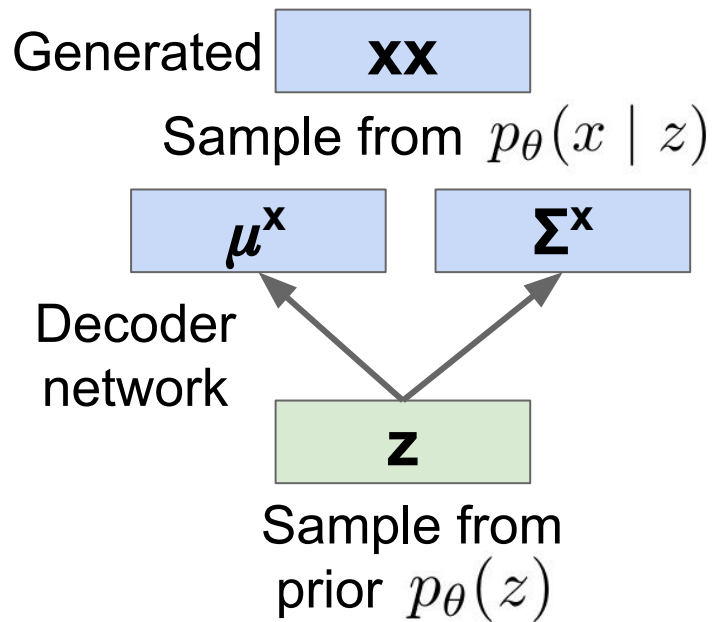
Variational Autoencoder: Generate Data!

After network is trained:



Variational Autoencoder: Generate Data!

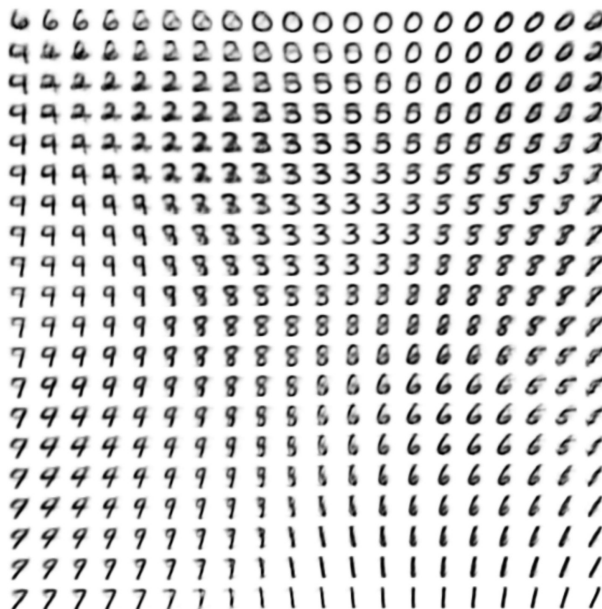
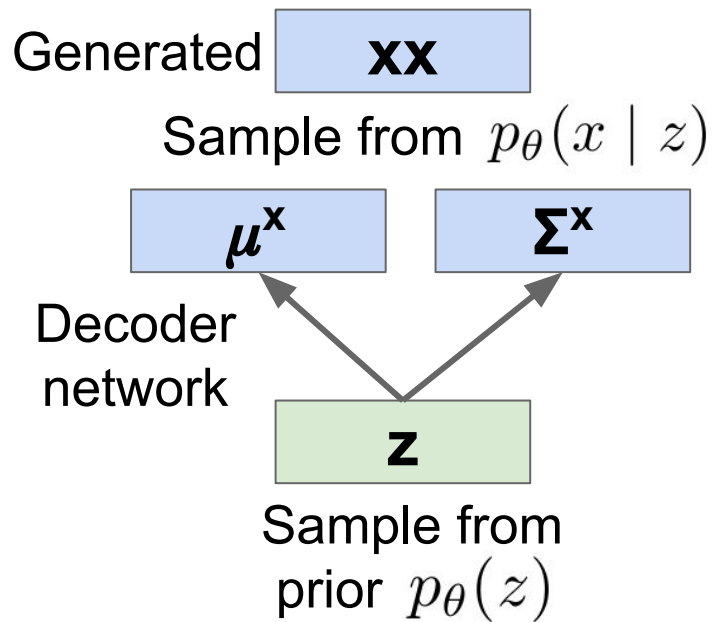
After network is trained:



Variational Autoencoder: Generate Data!

After network is trained:

Diagonal prior on $\mathbf{z} \Rightarrow$
independent latent variables



Variational Autoencoder: Math

Maximum Likelihood?

$$\theta^* = \arg \max_{\theta} \prod_{i=1}^N p_{\theta}(x^{(i)}) \quad \text{Maximize likelihood of dataset } \{x^{(i)}\}_{i=1}^N$$

Kingma and Welling, ICLR 2014

Variational Autoencoder: Math

Maximum Likelihood?

$$\theta^* = \arg \max_{\theta} \prod_{i=1}^N p_{\theta}(x^{(i)}) \quad \text{Maximize likelihood of dataset } \{x^{(i)}\}_{i=1}^N$$

$$= \arg \max_{\theta} \sum_{i=1}^N \log p_{\theta}(x^{(i)}) \quad \text{Maximize log-likelihood instead because sums are nicer}$$

Kingma and Welling, ICLR 2014

Variational Autoencoder: Math

Maximum Likelihood?

$$\theta^* = \arg \max_{\theta} \prod_{i=1}^N p_{\theta}(x^{(i)}) \quad \text{Maximize likelihood of dataset } \{x^{(i)}\}_{i=1}^N$$

$$= \arg \max_{\theta} \sum_{i=1}^N \log p_{\theta}(x^{(i)}) \quad \text{Maximize log-likelihood instead because sums are nicer}$$

$$p_{\theta}(x^{(i)}) = \int p_{\theta}(x^{(i)}, z) dz \quad \text{Marginalize joint distribution}$$

Kingma and Welling, ICLR 2014

Variational Autoencoder: Math

Maximum Likelihood?

$$\theta^* = \arg \max_{\theta} \prod_{i=1}^N p_{\theta}(x^{(i)}) \quad \text{Maximize likelihood of dataset } \{x^{(i)}\}_{i=1}^N$$

$$= \arg \max_{\theta} \sum_{i=1}^N \log p_{\theta}(x^{(i)}) \quad \text{Maximize log-likelihood instead because sums are nicer}$$

$$p_{\theta}(x^{(i)}) = \int p_{\theta}(x^{(i)}, z) dz = \int p_{\theta}(x^{(i)} | z) p_{\theta}(z) dz \quad \text{Intractible integral} =$$

Variational Autoencoder: Math

$$\log p_{\theta}(x^{(i)})$$

Variational Autoencoder: Math

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} [\log p_{\theta}(x^{(i)})] \quad (p_{\theta}(x^{(i)})) \text{ Does not depend on } z)$$

Variational Autoencoder: Math

$$\begin{aligned}\log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)})) \text{ Does not depend on } z \\ &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z)p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule})\end{aligned}$$

Variational Autoencoder: Math

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)})) \text{ Does not depend on } z$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] \quad (\text{Multiply by constant})$$

Variational Autoencoder: Math

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)})) \text{ Does not depend on } z$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Logarithms})$$

Variational Autoencoder: Math

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)})) \text{ Does not depend on } z$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Logarithms})$$

$$= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z)) + D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))$$

Variational Autoencoder: Math

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)})) \text{ Does not depend on } z$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Logarithms})$$

$$= \underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z)) + D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))}_{\mathcal{L}(x^{(i)}, \theta, \phi) \text{ “Elbow”}}$$

Variational Autoencoder: Math

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)})) \text{ Does not depend on } z$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Logarithms})$$

$$= \underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi) \text{ “Elbow”}} + \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))}_{\geq 0}$$

Variational Autoencoder: Math

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)})) \text{ Does not depend on } z$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Logarithms})$$

$$= \underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi) \text{ “Elbow”}} + \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))}_{\geq 0}$$

$$\log p_{\theta}(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound (elbow)

Variational Autoencoder: Math

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)})) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Logarithms})$$

$$= \underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi) \text{ “Elbow”}} + \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))}_{\geq 0}$$

$$\log p_{\theta}(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

$$\theta^*, \phi^* = \arg \max_{\theta, \phi} \sum_{i=1}^N \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound (elbow)

Training: Maximize lower bound

Variational Autoencoder: Math

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} [\log p_{\theta}(x^{(i)})] \quad (p_{\theta}(x^{(i)})) \text{ Does not depend on } z)$$

**Reconstruct
the input
data**

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Logarithms})$$

$$= \underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right]}_{\mathcal{L}(x^{(i)}, \theta, \phi) \text{ "Elbow"}} - \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z)) + D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))}_{\geq 0}$$

$$\log p_{\theta}(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

$$\theta^*, \phi^* = \arg \max_{\theta, \phi} \sum_{i=1}^N \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound (elbow)

Training: Maximize lower bound

Variational Autoencoder: Math

Latent states
should follow
the prior

Reconstruct the input data

$$\begin{aligned}
 \log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)})) \text{ Does not depend on } z \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant}) \\
 &= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] && (\text{Logarithms}) \\
 &= \underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right]}_{\mathcal{L}(x^{(i)}, \theta, \phi) \text{ "Elbow"}} - \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))}_{\geq 0} + \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))}_{\geq 0}
 \end{aligned}$$

$$\log p_{\theta}(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound (elbow)

$$\theta^*, \phi^* = \arg \max_{\theta, \phi} \sum_{i=1}^N \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

Variational Autoencoder: Math

Latent states
should follow
the prior

Reconstruct the input data

Sampling with reparam. trick (see paper)

$$\begin{aligned}
 \log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)})) \text{ Does not depend on } z \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant}) \\
 &= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] && (\text{Logarithms}) \\
 &= \underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right]}_{\mathcal{L}(x^{(i)}, \theta, \phi) \text{ "Elbow"}} - \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))}_{\geq 0} + \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))}_{\geq 0}
 \end{aligned}$$

$$\log p_{\theta}(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound (elbow)

$$\theta^*, \phi^* = \arg \max_{\theta, \phi} \sum_{i=1}^N \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

Variational Autoencoder: Math

Reconstruct the input data
Sampling with reparam. trick (see paper)

$$\begin{aligned}
 \log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)})) \text{ Does not depend on } z \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant}) \\
 &= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] && (\text{Logarithms}) \\
 &= \underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right]}_{\mathcal{L}(x^{(i)}, \theta, \phi) \text{ "Elbow"}} - \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))}_{\geq 0} + \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))}_{\geq 0}
 \end{aligned}$$

Latent states should follow the prior
Everything is Gaussian, closed form solution!

$$\log p_{\theta}(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound (elbow)

$$\theta^*, \phi^* = \arg \max_{\theta, \phi} \sum_{i=1}^N \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

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 Nets

Can we generate images with less math?

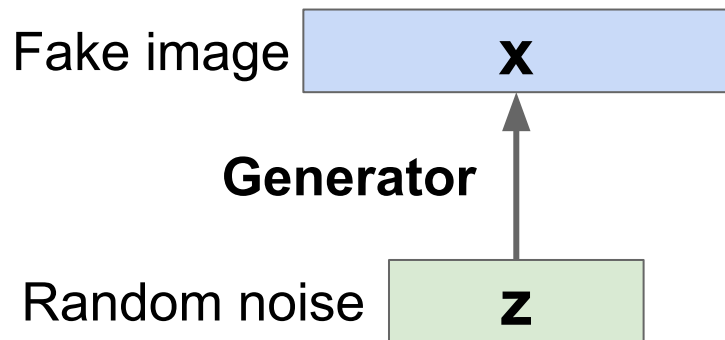
Random noise



z

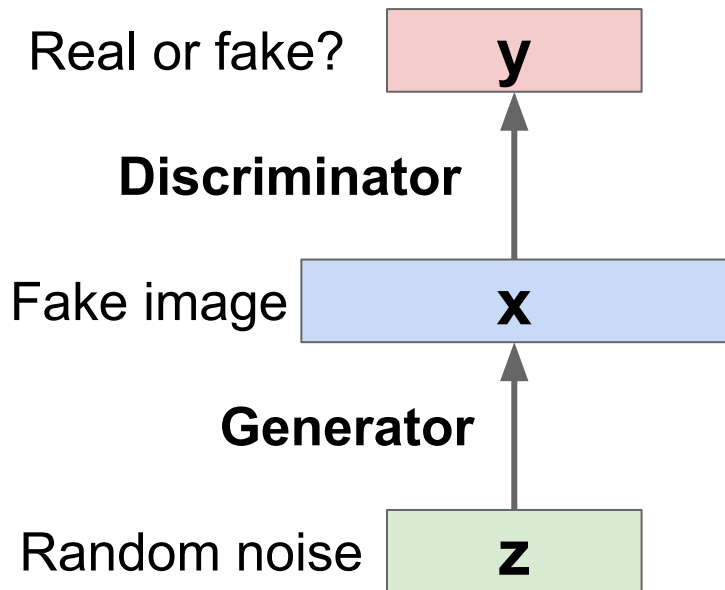
Generative Adversarial Nets

Can we generate images with less math?



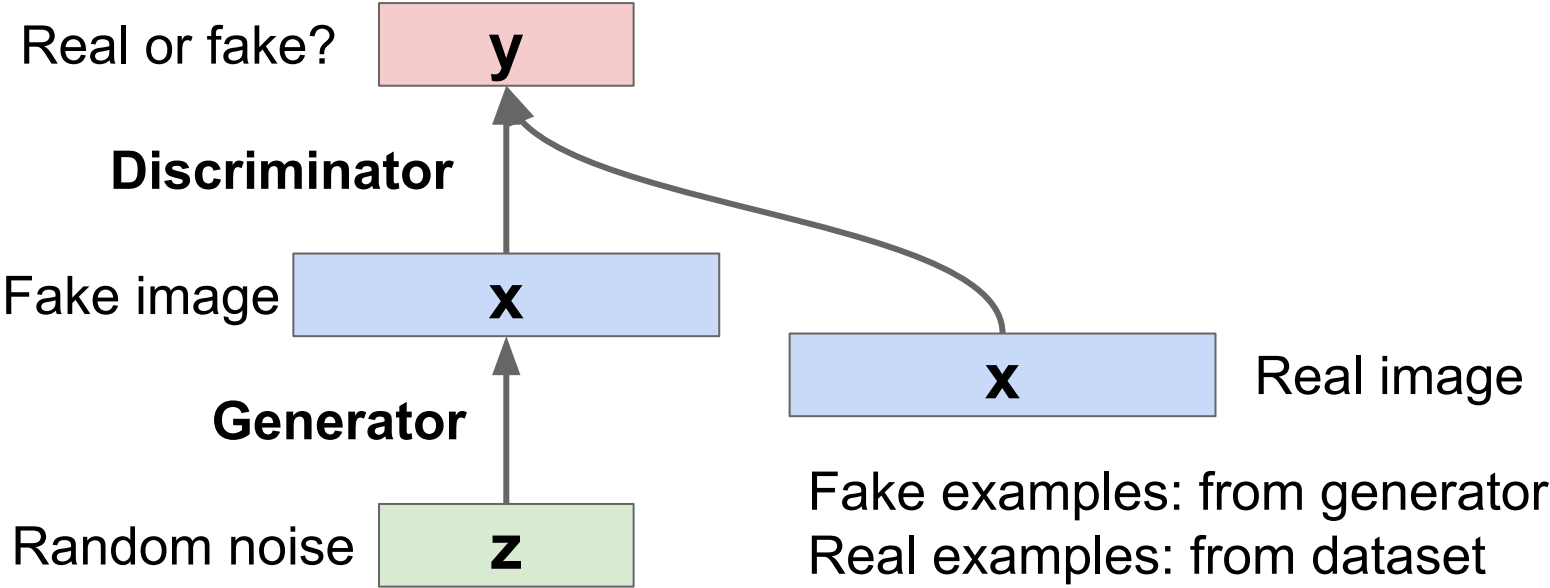
Generative Adversarial Nets

Can we generate images with less math?



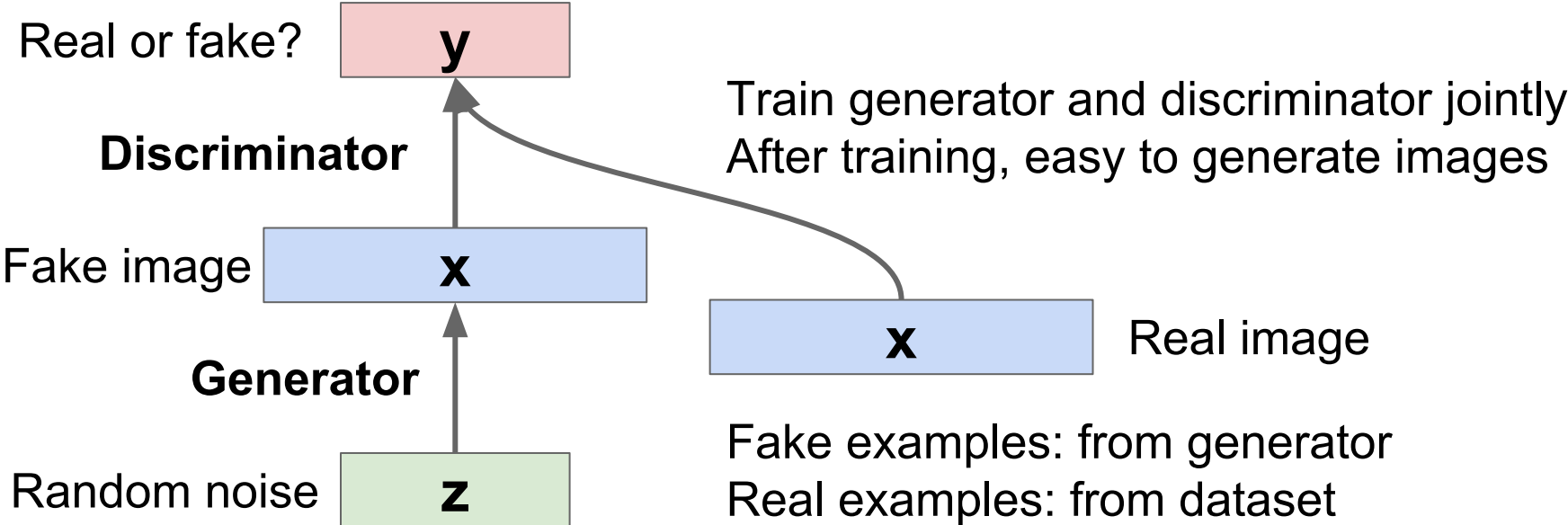
Generative Adversarial Nets

Can we generate images with less math?



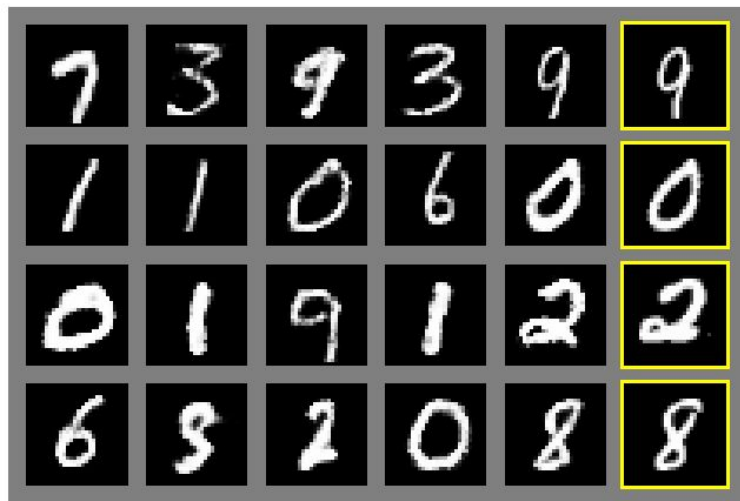
Generative Adversarial Nets

Can we generate images with less math?



Generative Adversarial Nets

Generated samples



Nearest neighbor from training set

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

Generative Adversarial Nets

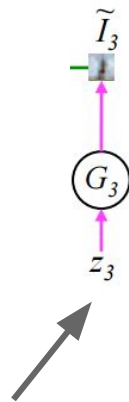
Generated samples (CIFAR-10)



Nearest neighbor from training set

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

Generative Adversarial Nets: Multiscale

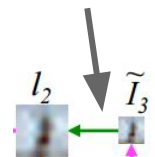


Generate
low-res

Denton et al, "Deep generative image models using a Laplacian pyramid of adversarial networks", NIPS 2015

Generative Adversarial Nets: Multiscale

Upsample



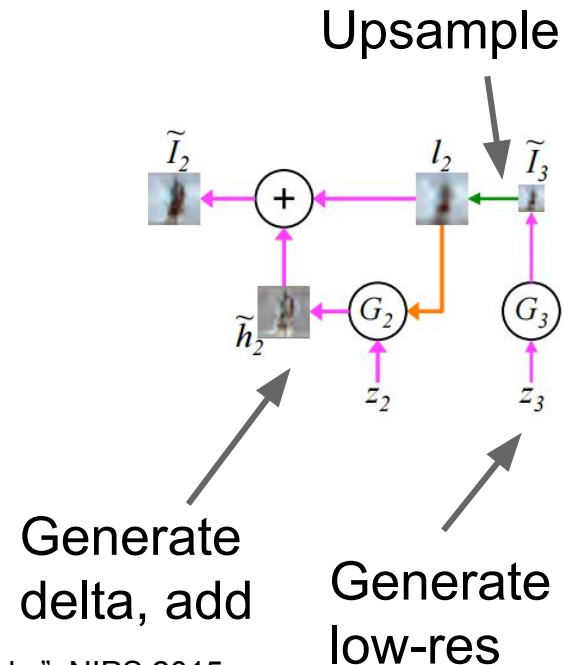
G_3

z_3

Generate
low-res

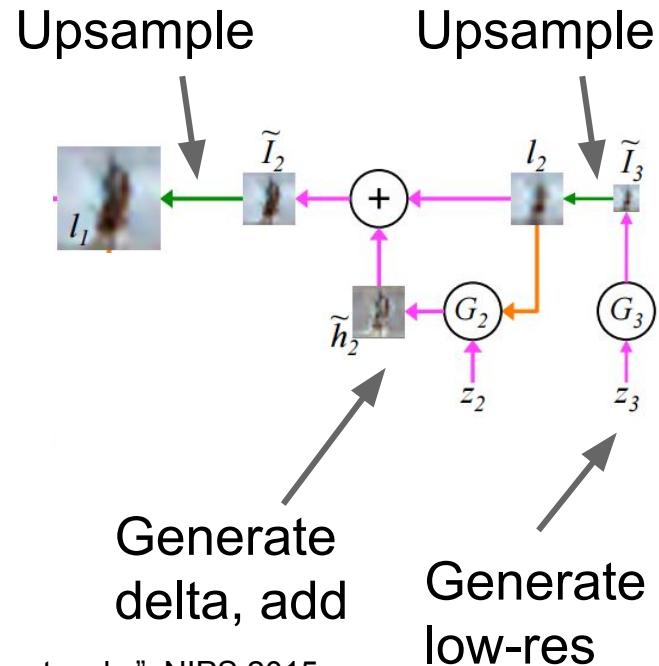
Denton et al, "Deep generative image models using a Laplacian pyramid of adversarial networks", NIPS 2015

Generative Adversarial Nets: Multiscale



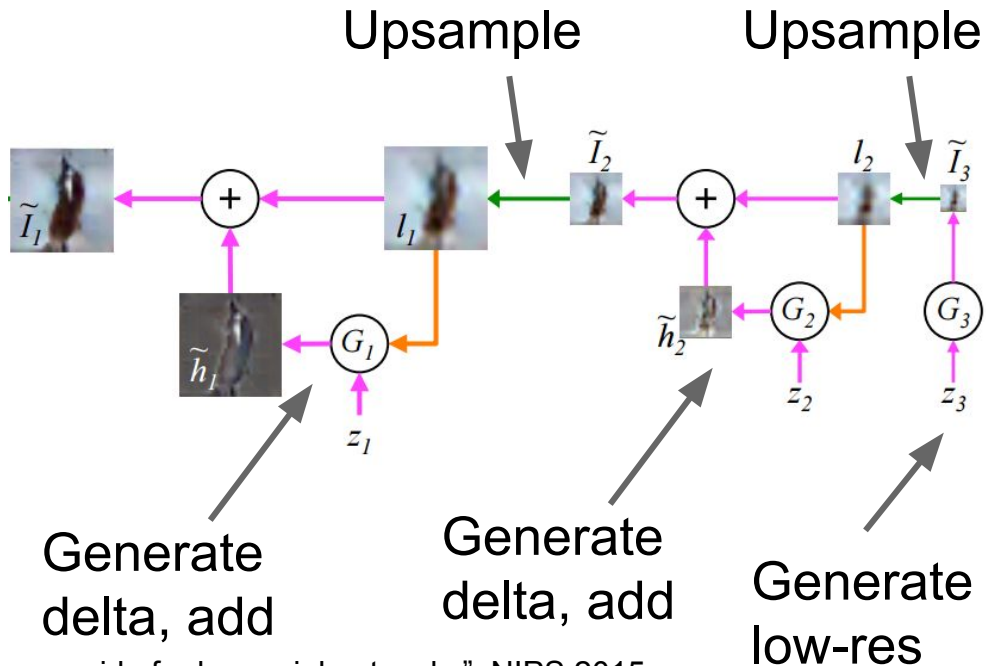
Denton et al, "Deep generative image models using a Laplacian pyramid of adversarial networks", NIPS 2015

Generative Adversarial Nets: Multiscale



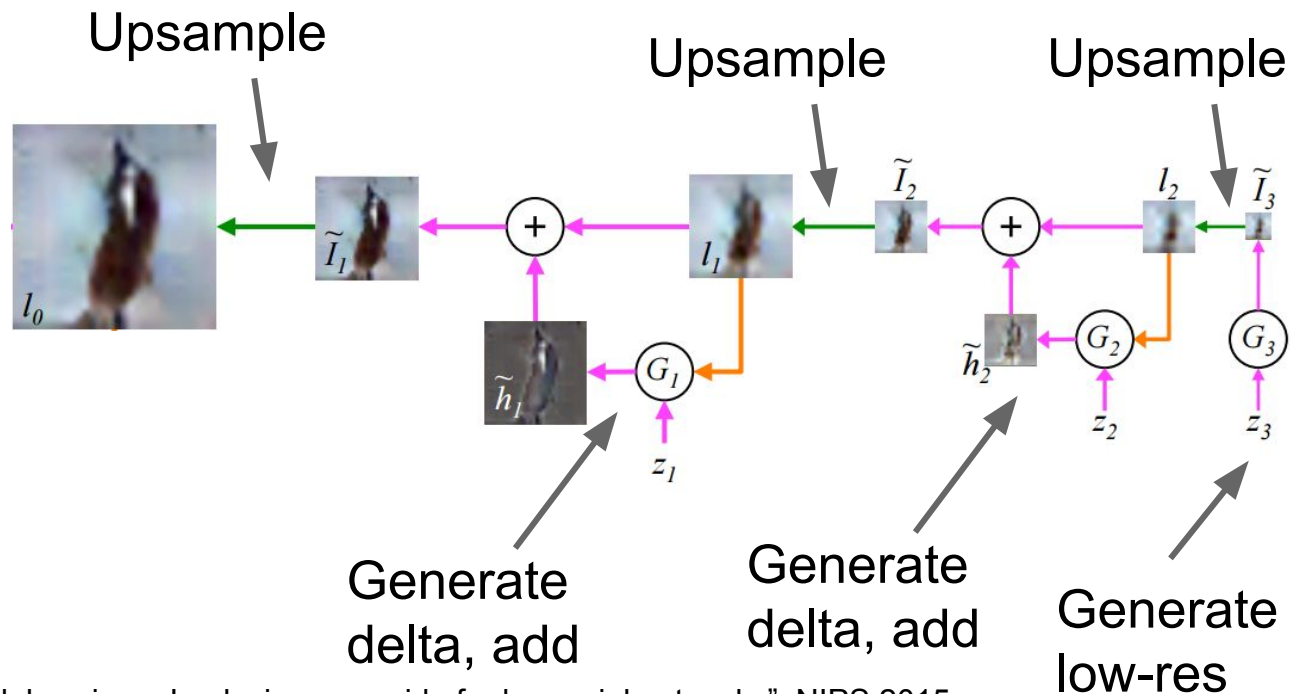
Denton et al, "Deep generative image models using a Laplacian pyramid of adversarial networks", NIPS 2015

Generative Adversarial Nets: Multiscale



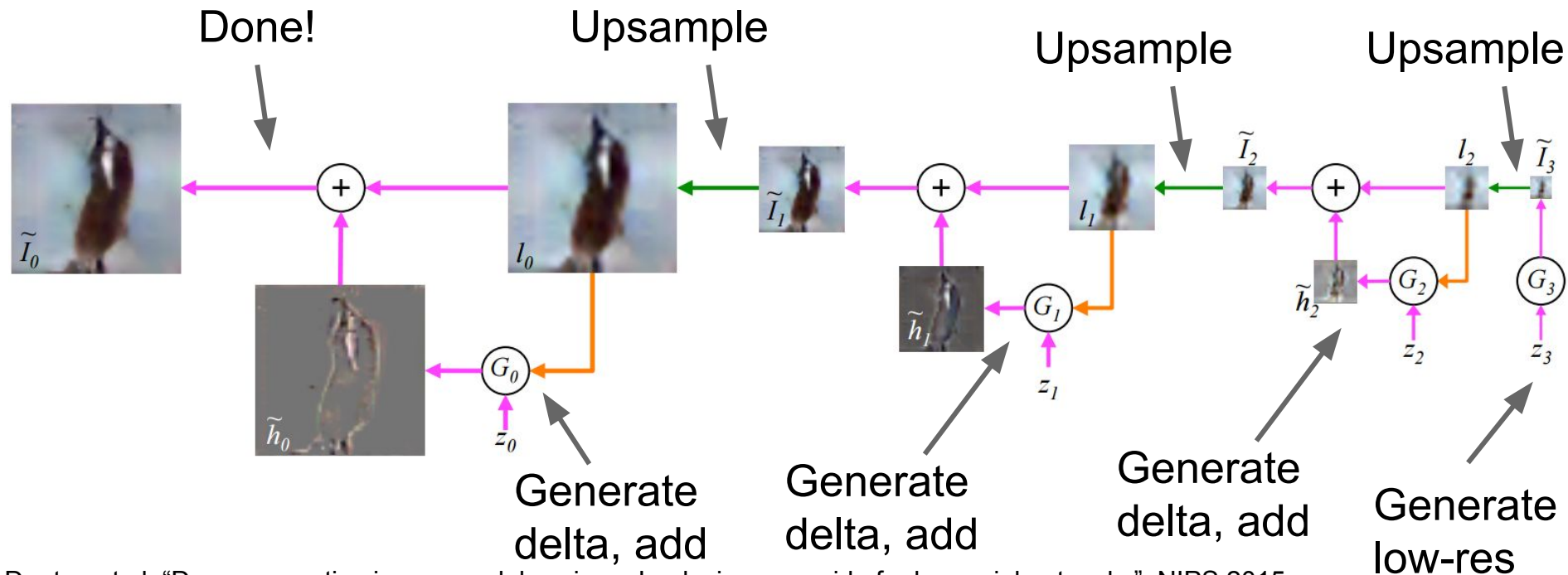
Denton et al, "Deep generative image models using a Laplacian pyramid of adversarial networks", NIPS 2015

Generative Adversarial Nets: Multiscale



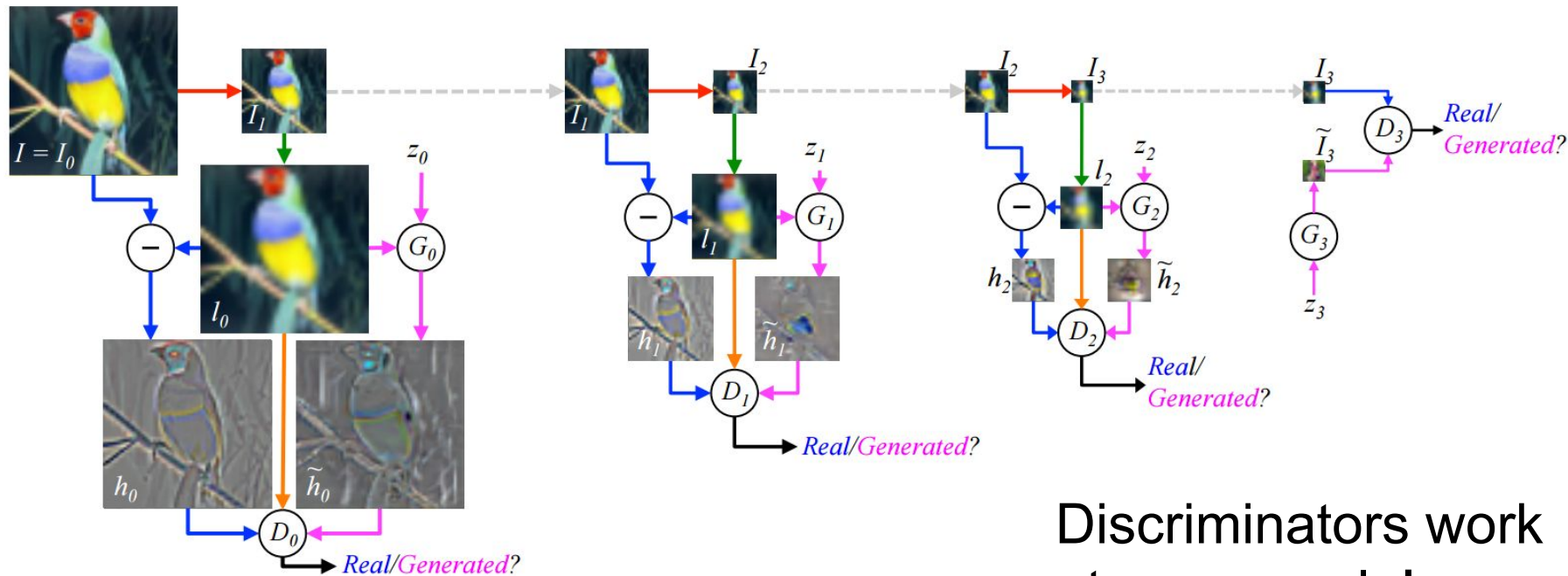
Denton et al, "Deep generative image models using a Laplacian pyramid of adversarial networks", NIPS 2015

Generative Adversarial Nets: Multiscale



Denton et al, "Deep generative image models using a Laplacian pyramid of adversarial networks", NIPS 2015

Generative Adversarial Nets: Multiscale



Discriminators work
at every scale!

Denton et al, NIPS 2015

Generative Adversarial Nets: Multiscale



Train separate model per-class on CIFAR-10

Denton et al, NIPS 2015

Generative Adversarial Nets: Simplifying

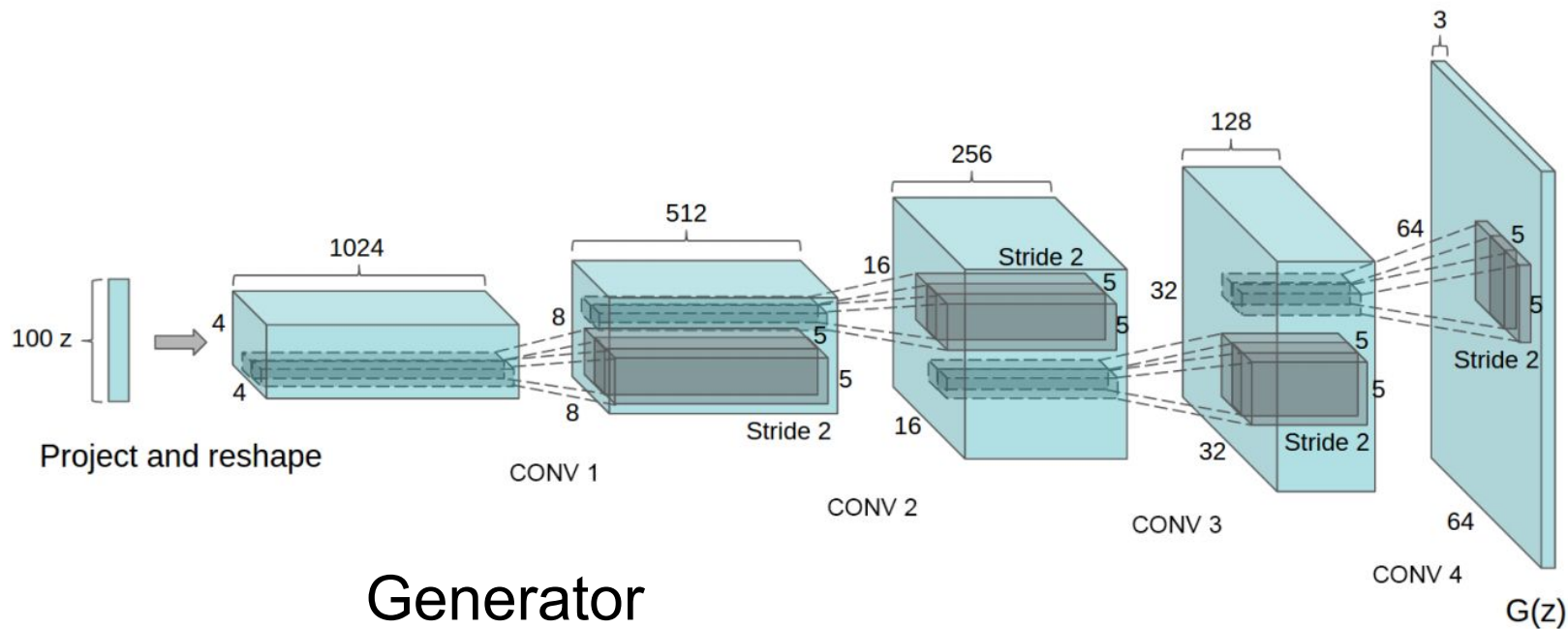
Generator is an upsampling network with fractionally-strided convolutions
Discriminator is a convolutional network

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

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

Generative Adversarial Nets: Simplifying

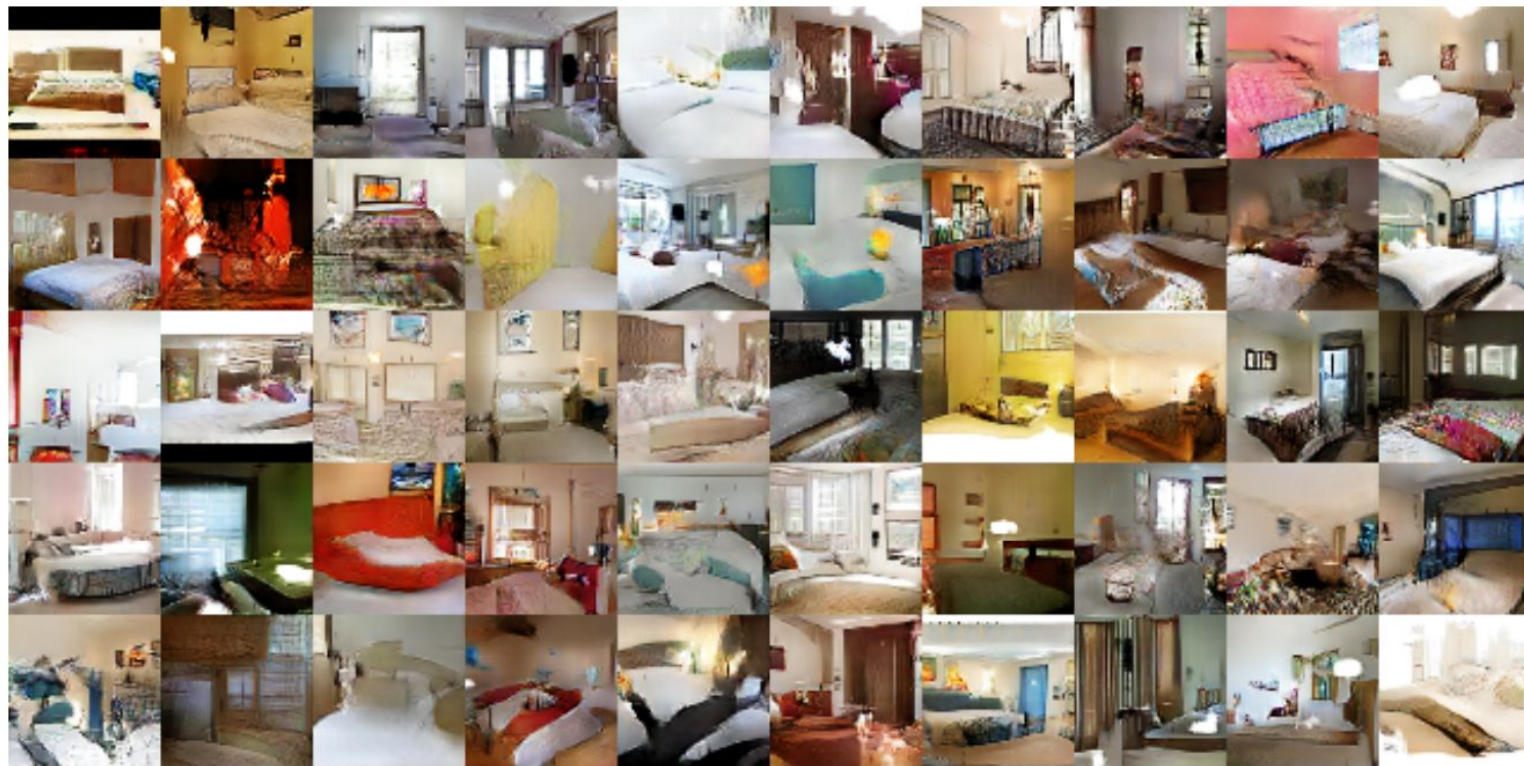


Generator

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

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

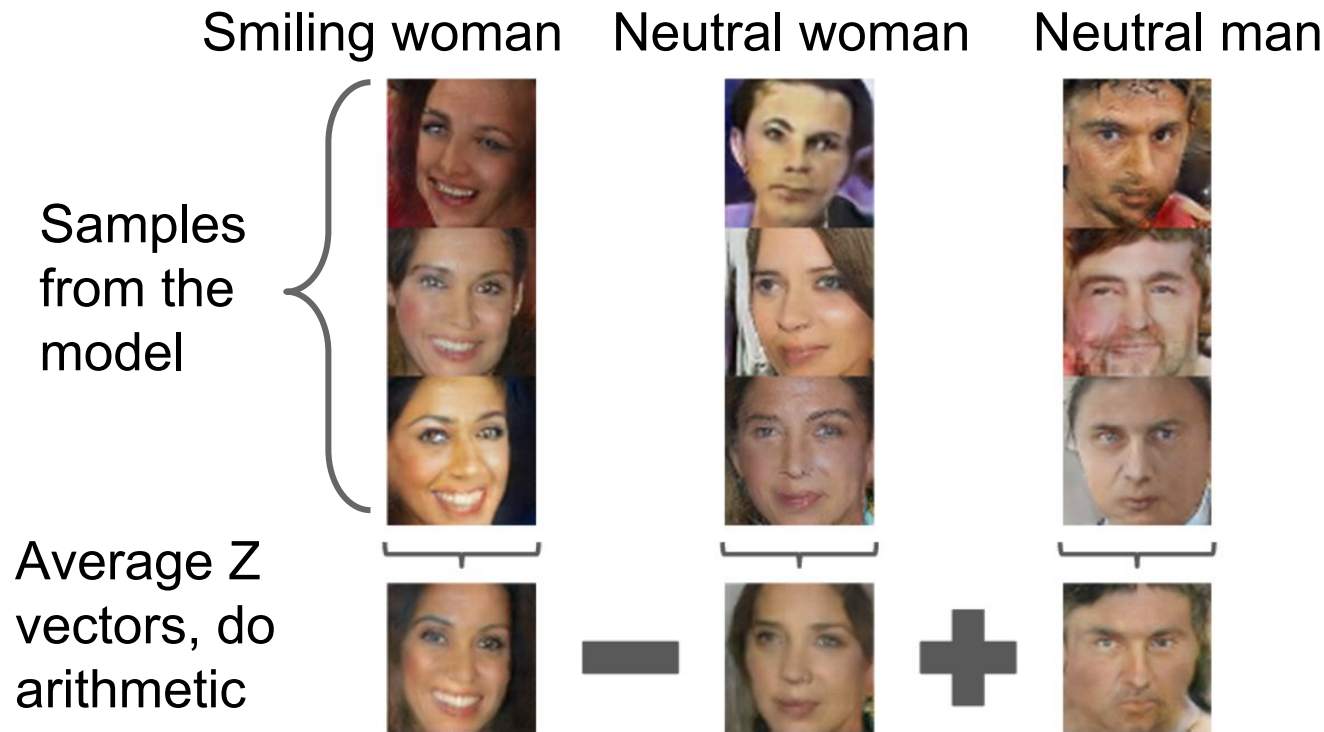
Neutral man

Samples
from the
model



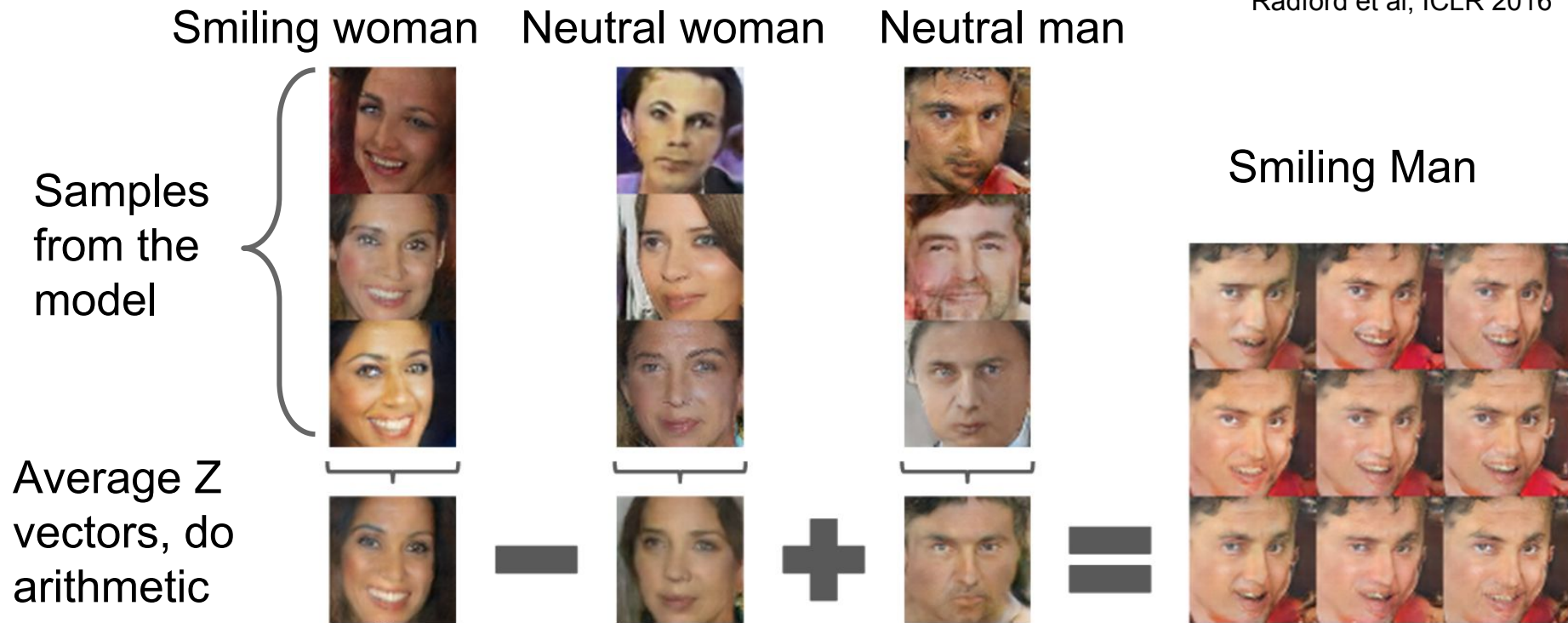
Generative Adversarial Nets: Vector Math

Radford et al, ICLR 2016



Generative Adversarial Nets: Vector Math

Radford et al, ICLR 2016



Generative Adversarial Nets: Vector Math

Glasses man

No glasses man

No glasses woman



Radford et al,
ICLR 2016

Generative Adversarial Nets: Vector Math

Glasses man

No glasses man

No glasses woman



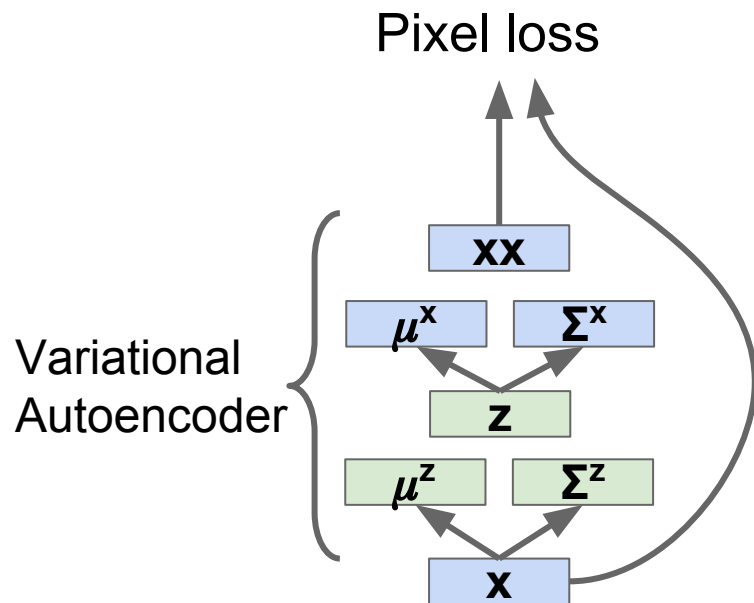
Woman with glasses



Radford et al,
ICLR 2016

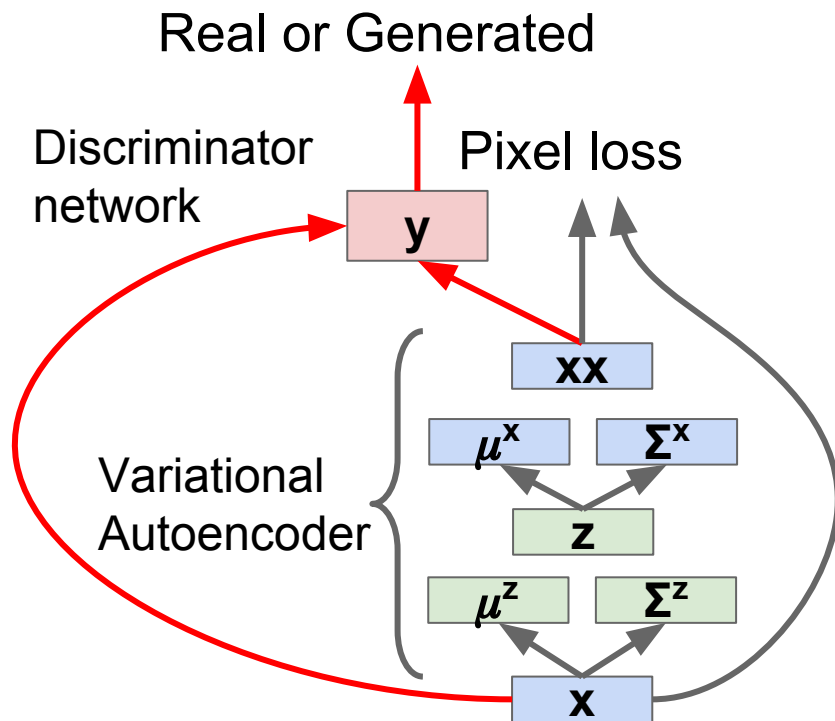
Putting everything together

Dosovitskiy and Brox, “Generating Images with Perceptual Similarity Metrics based on Deep Networks”, arXiv 2016



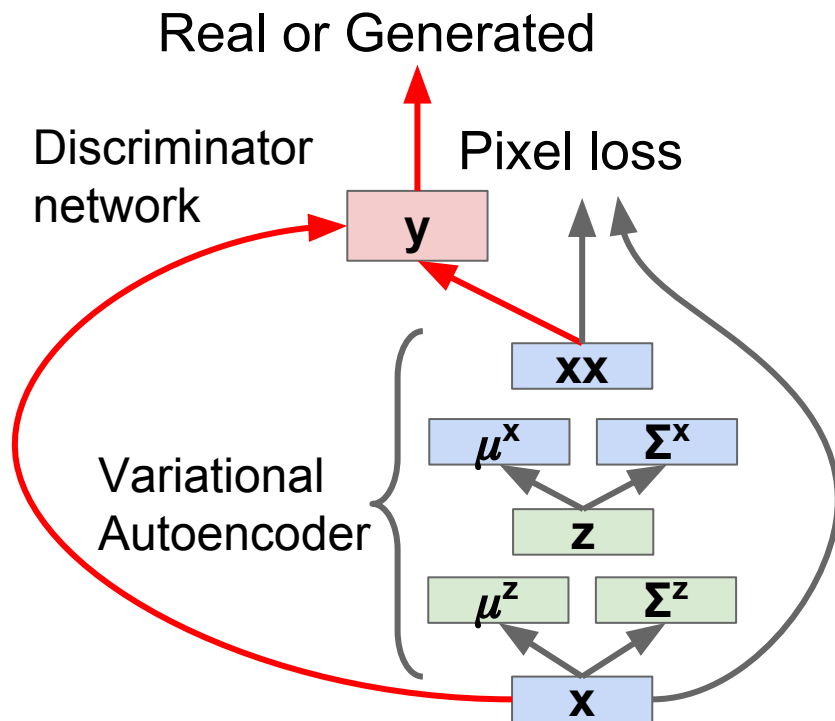
Putting everything together

Dosovitskiy and Brox, “Generating Images with Perceptual Similarity Metrics based on Deep Networks”, arXiv 2016

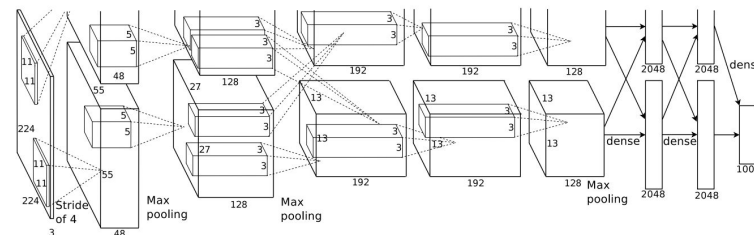


Putting everything together

Dosovitskiy and Brox, “Generating Images with Perceptual Similarity Metrics based on Deep Networks”, arXiv 2016

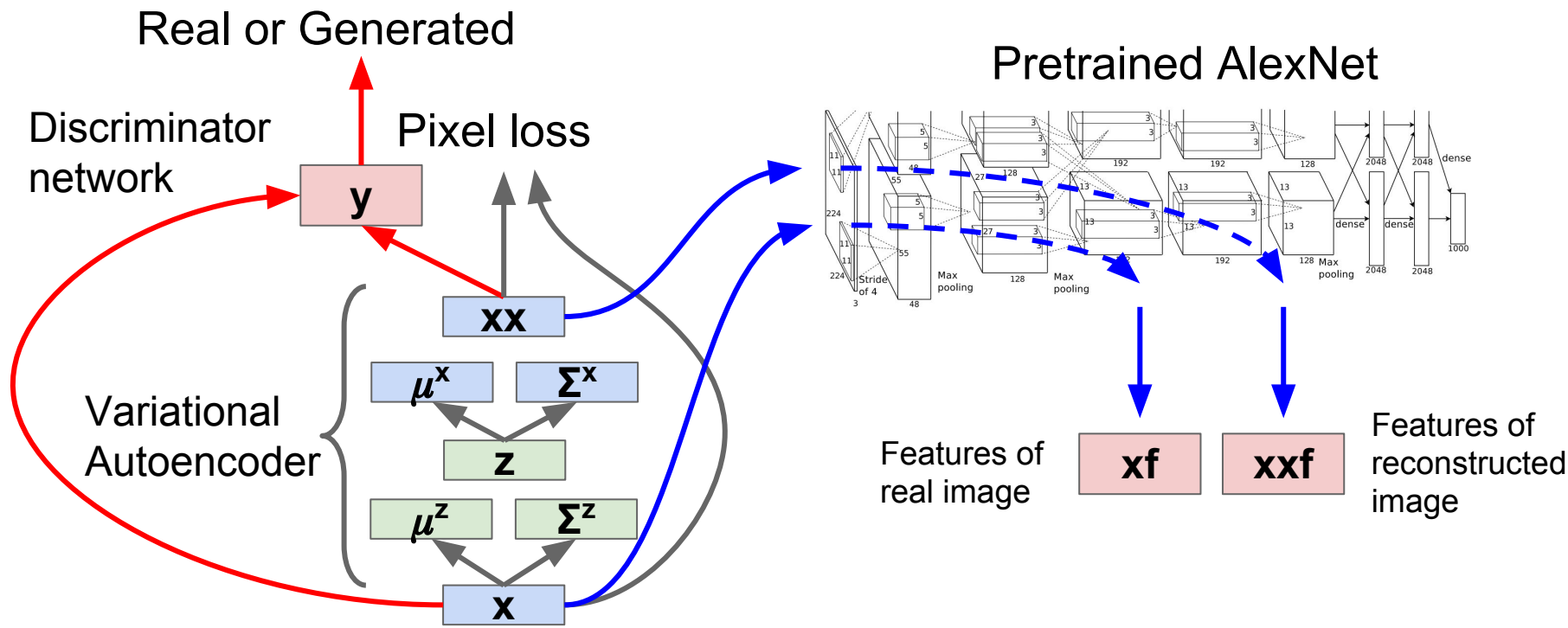


Pretrained AlexNet



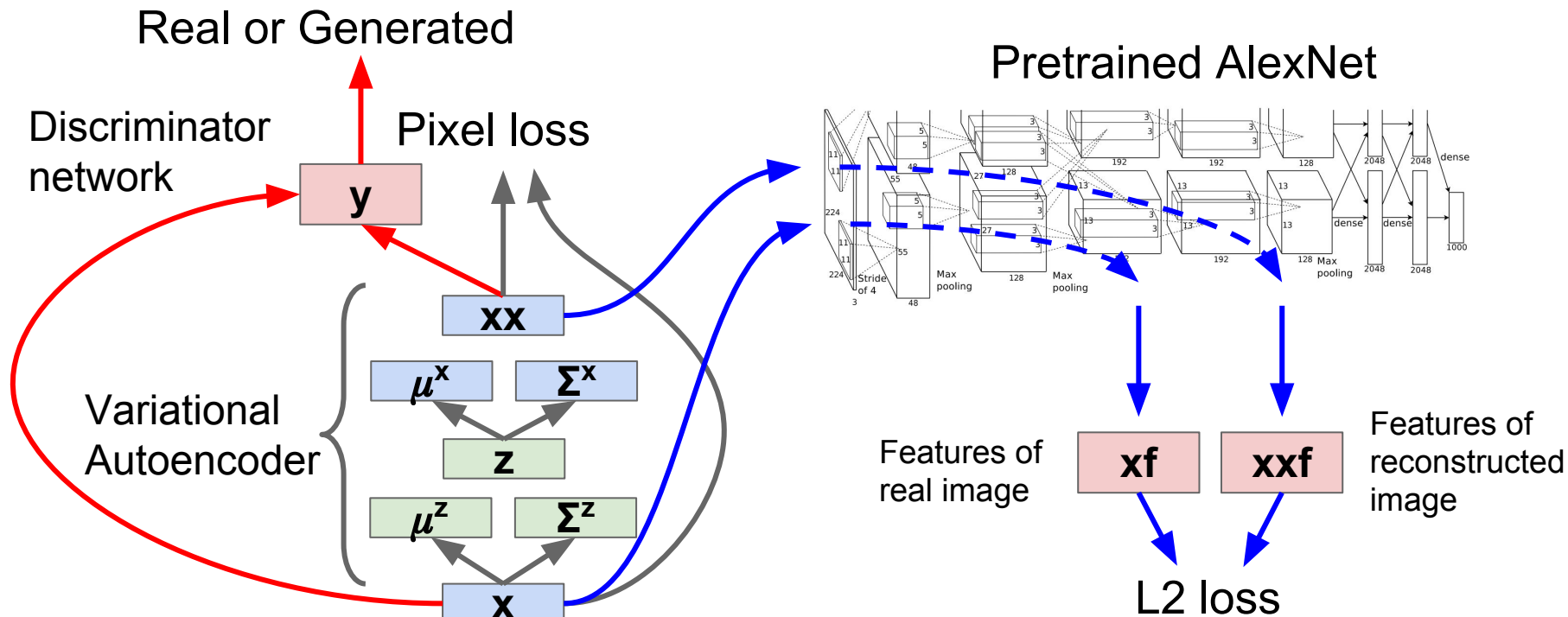
Putting everything together

Dosovitskiy and Brox, “Generating Images with Perceptual Similarity Metrics based on Deep Networks”, arXiv 2016



Putting everything together

Dosovitskiy and Brox, “Generating Images with Perceptual Similarity Metrics based on Deep Networks”, arXiv 2016



Putting everything together

Dosovitskiy and Brox, “Generating Images with Perceptual Similarity Metrics based on Deep Networks”, arXiv 2016

Samples
from the
model, trained
on ImageNet



Recap

- Videos
- Unsupervised learning
 - Autoencoders: Traditional / variational
 - Generative Adversarial Networks
- Next time: Guest lecture from Jeff Dean