# Autoencoders & VAE

# Machine learning

## Unsupervised learning

## Supervised learning

- Labels or target classes
- Goal: learn a mapping from input to label
- Classification, regression

# Machine learning

## Unsupervised learning

## Supervised learning

CAT

DOG

DOG

CAT

CAT

DOG

# Machine learning

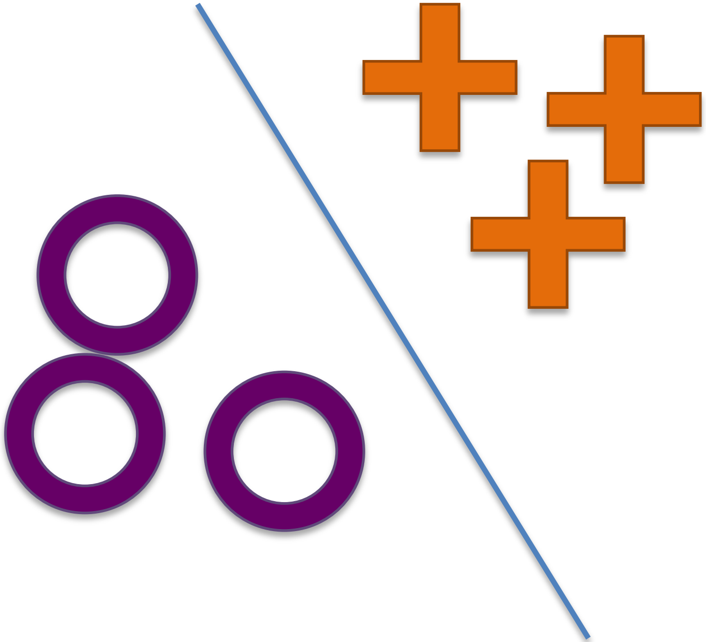
## Unsupervised learning

- No label or target class
- Find out properties of the structure of the data
- Clustering (k-means, PCA)

## Supervised learning

# Machine learning

## Unsupervised learning



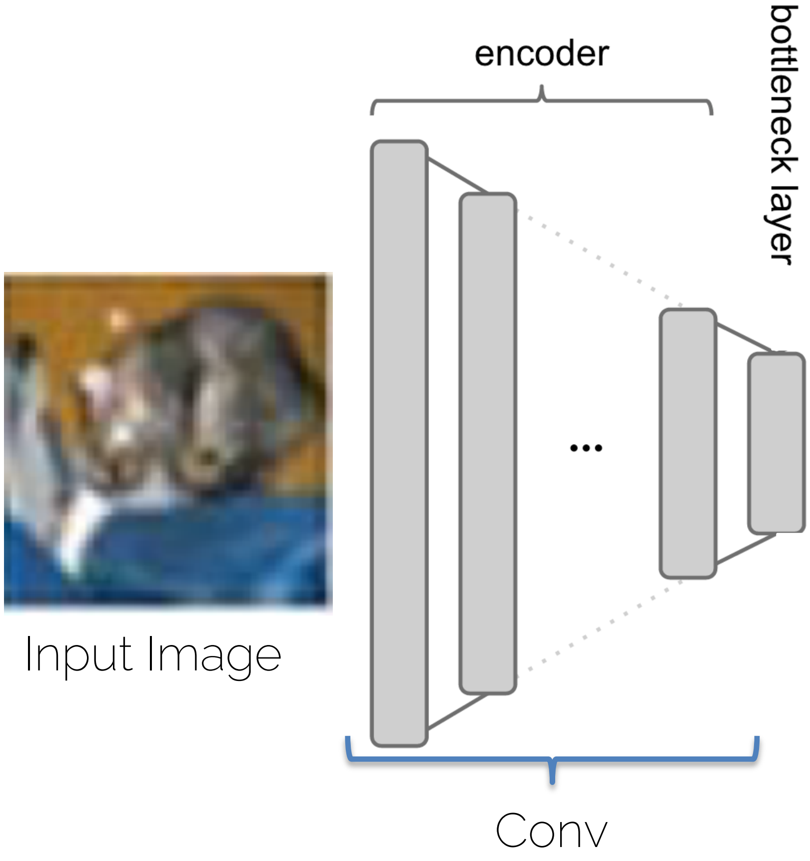
## Supervised learning

# Unsupervised learning with autoencoders

# Autoencoders

- Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

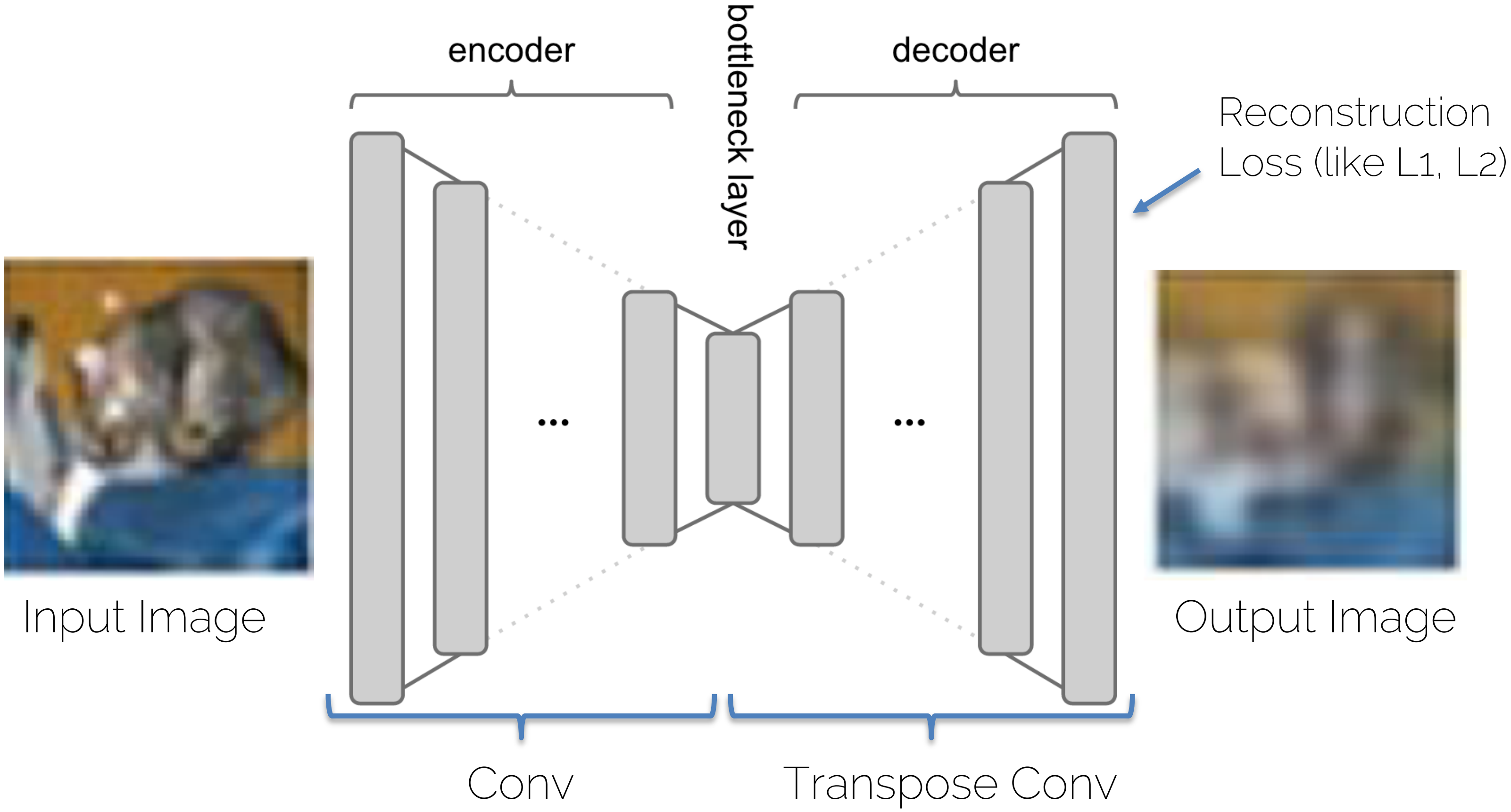
# Autoencoders



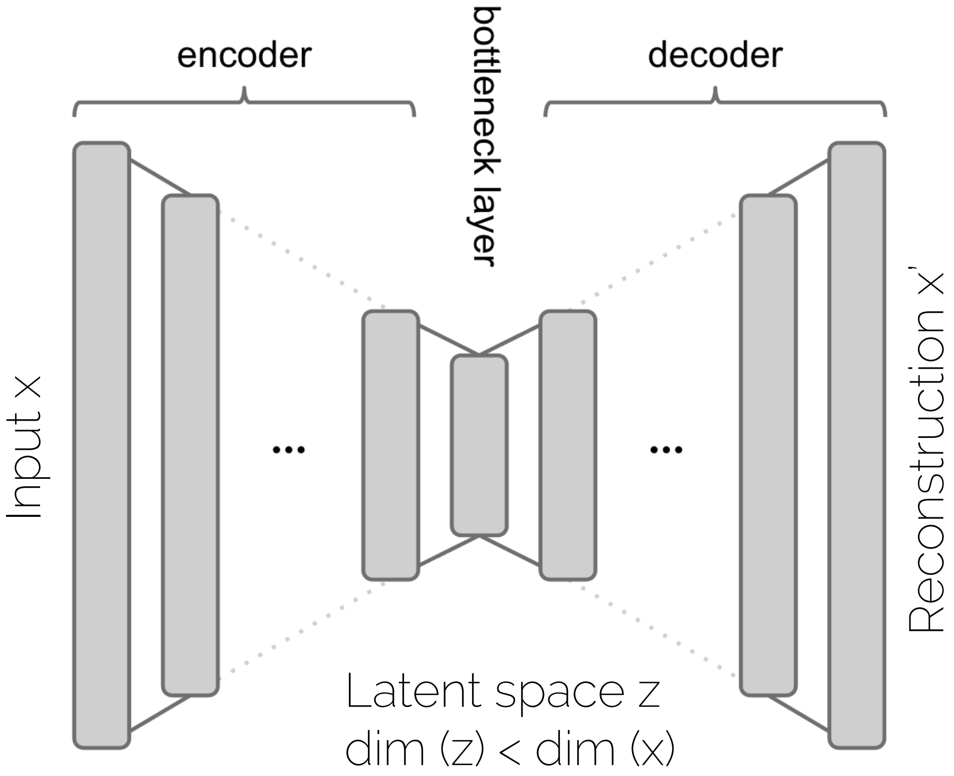

- From an input image to a feature representation (bottleneck layer)

- Encoder: a CNN in our case

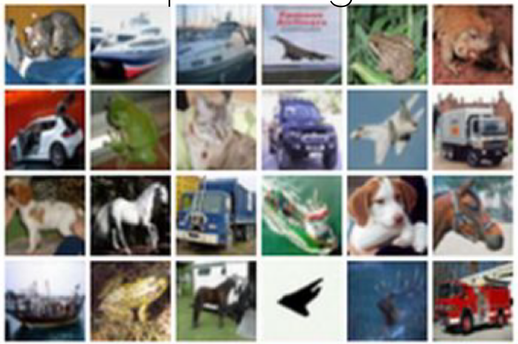
# Autoencoder: training



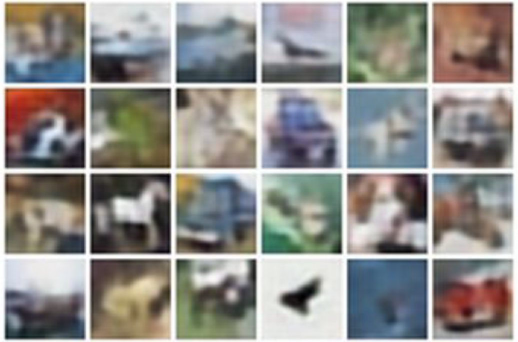
# Autoencoder: training



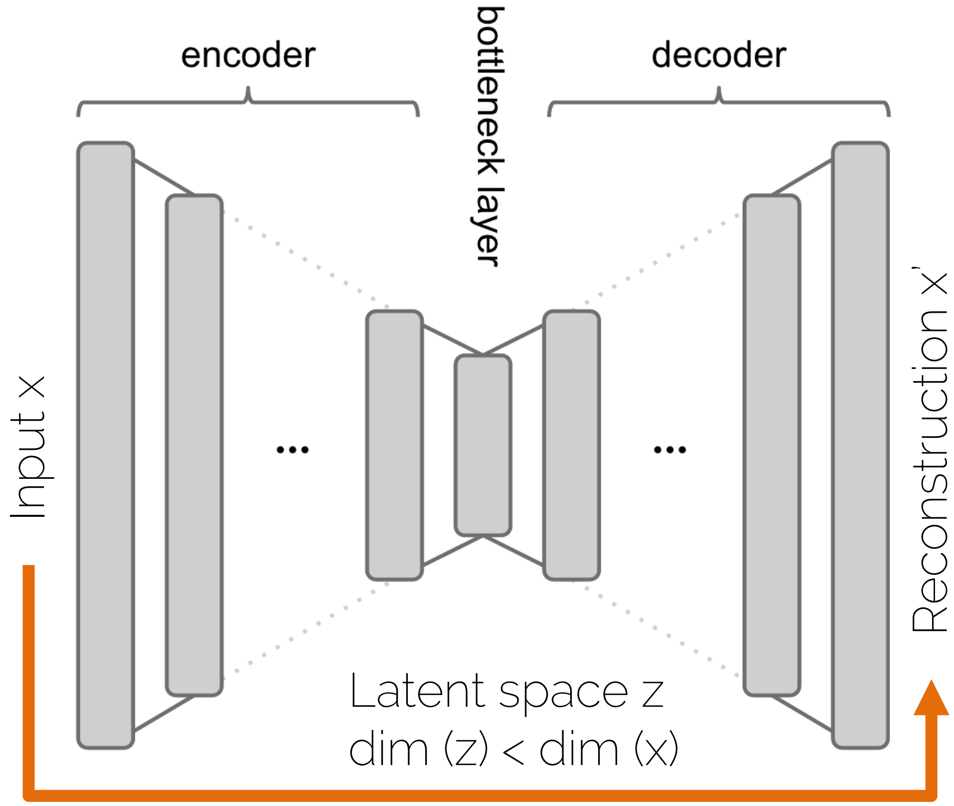

Input images



Reconstructed images



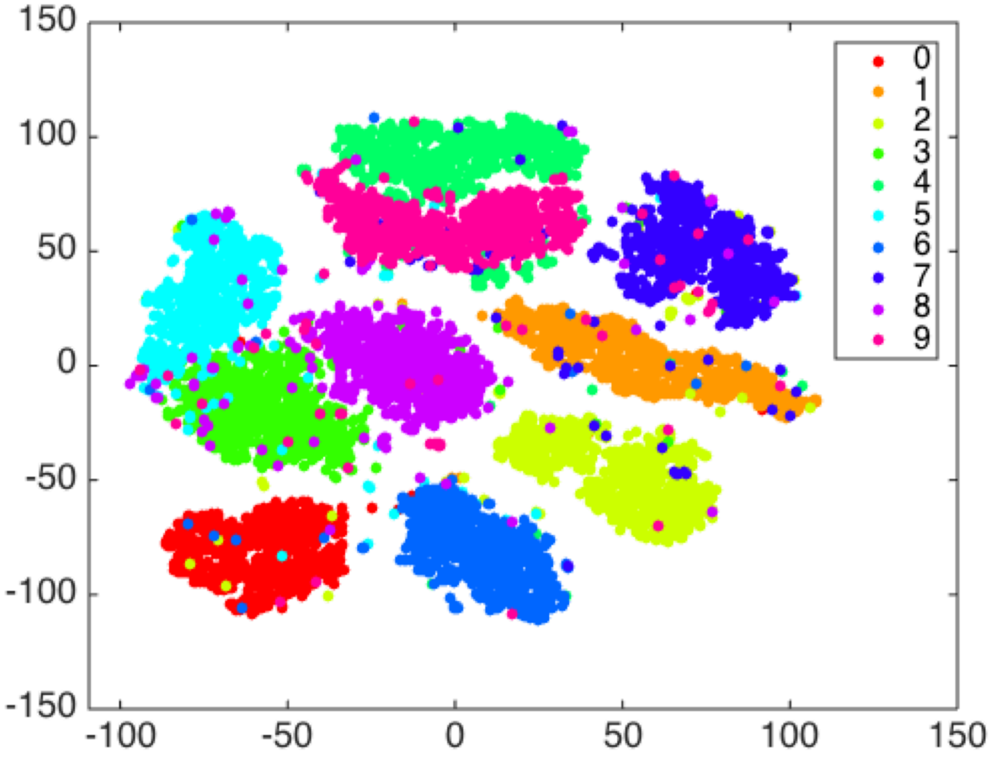
# Autoencoder: training




- No labels required
- We can use unlabeled data to first get its structure

# Autoencoder: Use Cases

Embedding of
MNIST numbers



# Autoencoder for pre-training

- Test case: medical applications based on CT images
  - Large set of *unlabeled* data.
  - Small set of *labeled* data.

- We cannot do: take a network pre-trained on ImageNet. Why?

- The image features are different CT vs natural images

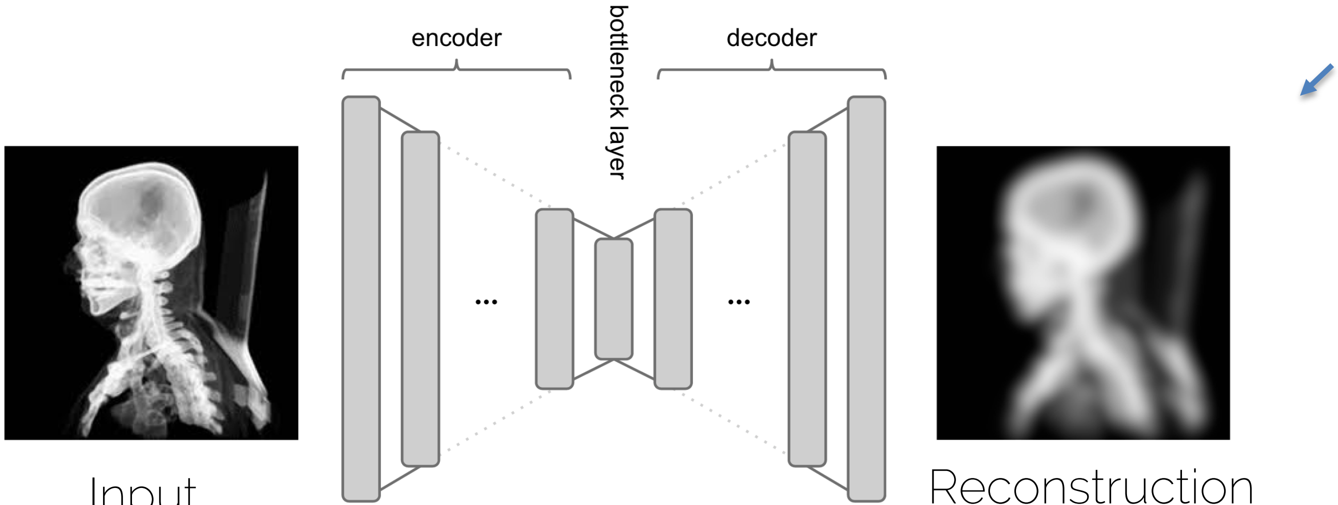
# Autoencoder for pre-training

- Test case: medical applications based on CT images
  - Large set of *unlabeled* data.
  - Small set of *labeled* data.

- We can do: pre-train our network using an autoencoder to “learn” the type of features present in CT images

# Autoencoder for pre-training

- Step 1: Unsupersived training with autoencoders

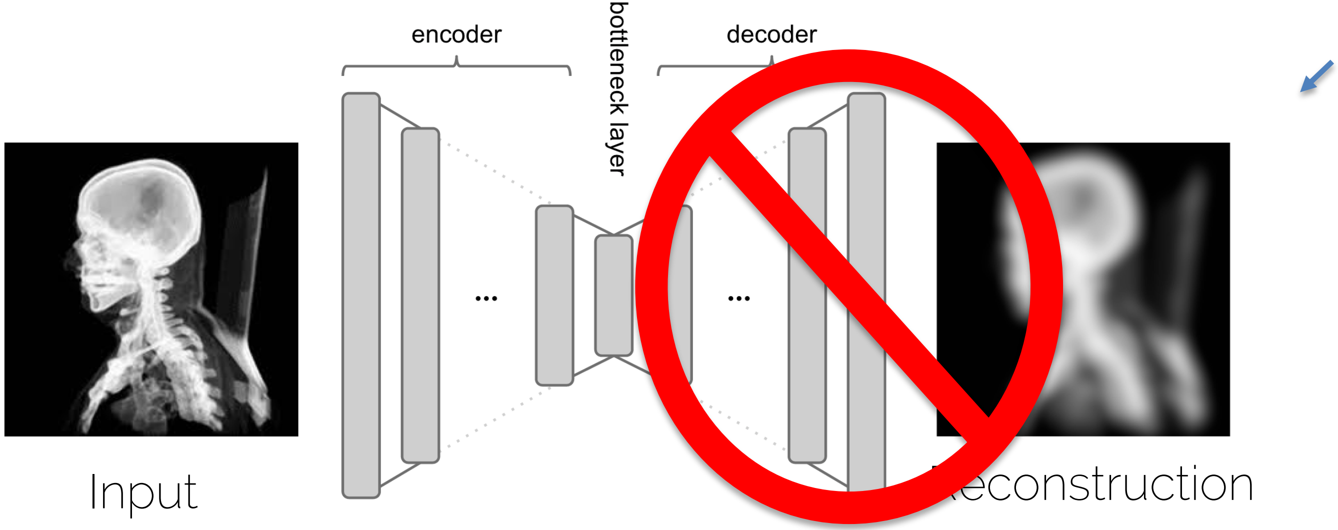



Input

Reconstruction

# Autoencoder for pre-training

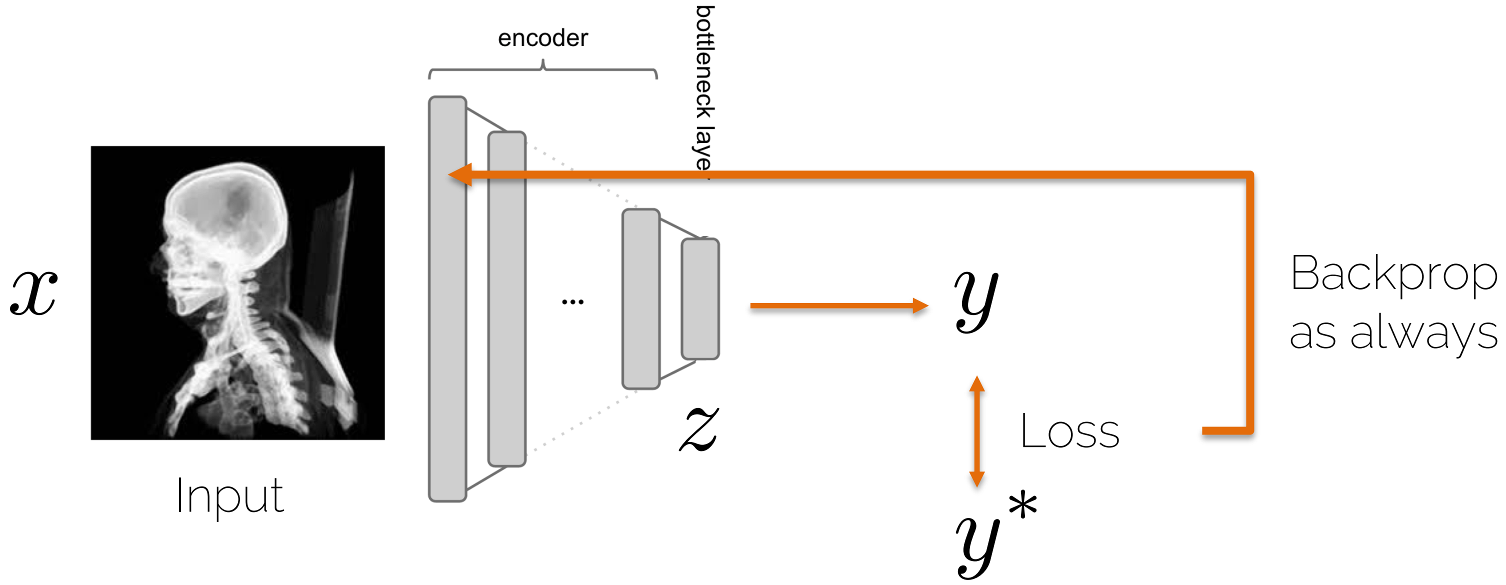
- Step 2: *Supervised* training with the labeled data




Throw away the decoder

# Autoencoder for pre-training

- Step 2: *Supervised* training with the labeled data




Ground truth labels for supervised learning

# Why using autoencoders?

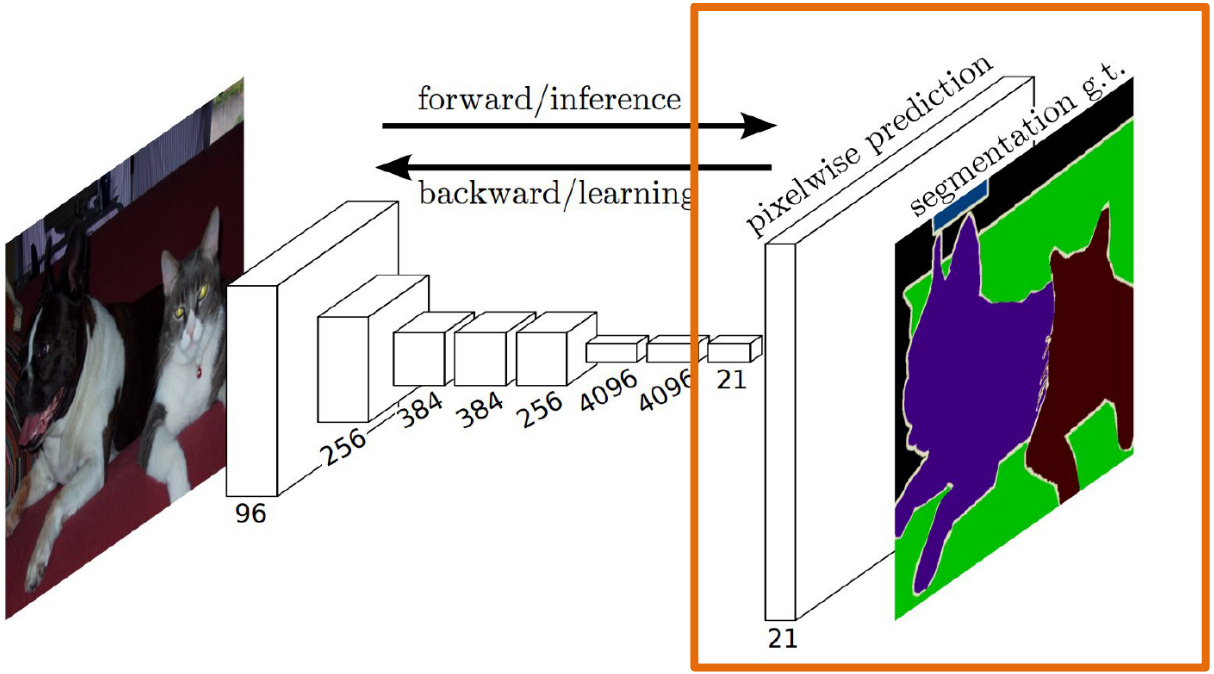
- Use 1: pre-training, as mentioned before
  - Image → same image reconstructed
  - Use the encoder as "feature extractor"

- Use 2: Use them to get pixel-wise predictions
  - Image → semantic segmentation
  - Low-resolution image → High-resolution image
  - Image → Depth map

# Autoencoders for pixel-wise predictions

# Semantic Segmentation (FCN)

- Recall the Fully Convolutional Networks

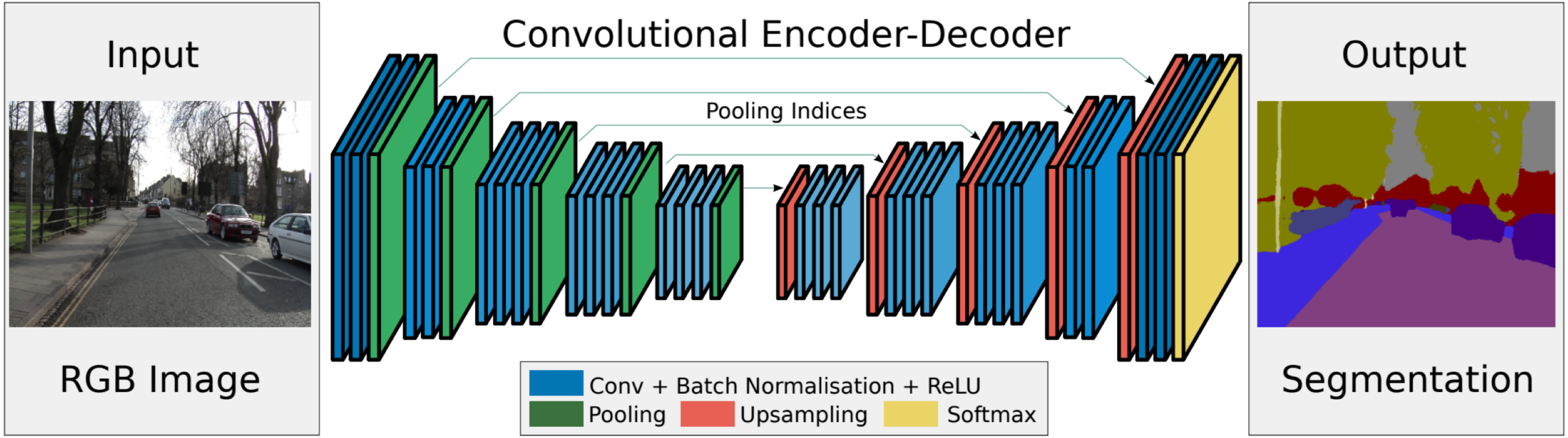



Can we do better?



[Long et al. 15] Fully Convolutional Networks for Semantic Segmetnation (FCN)

# SegNet

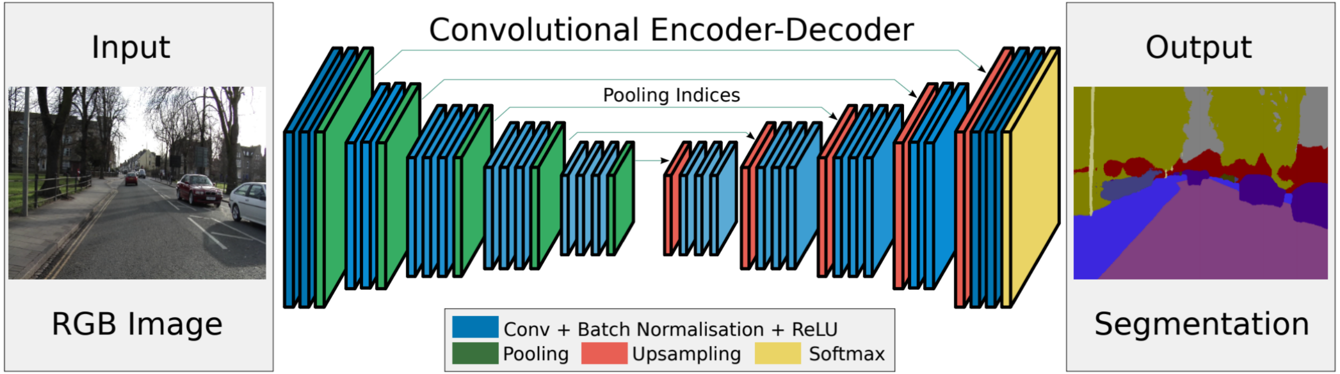



Badrinarayanan et al. „SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". TPAMI 2016

# SegNet

- **Encoder**: normal convolutional filters + pooling

- **Decoder**: Upsampling + convolutional filters

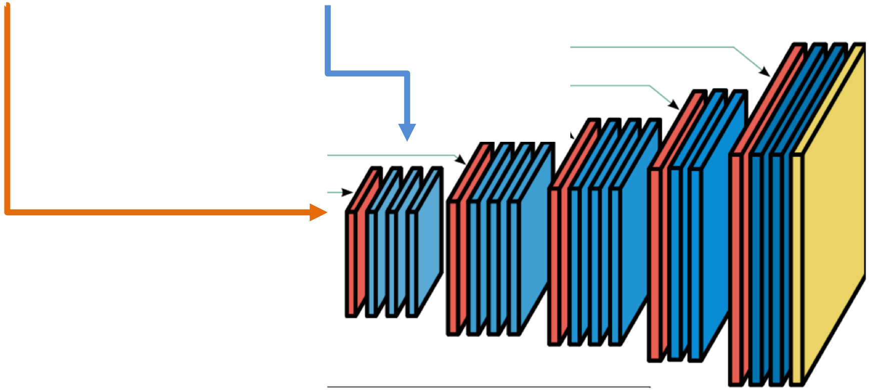



Badrinarayanan et al. „SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation“. TPAMI 2016

# SegNet

- **Encoder**: normal convolutional filters + pooling

- **Decoder**: Upsampling + convolutional filters



Badrinarayanan et al. „SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation“. TPAMI 2016

# SegNet

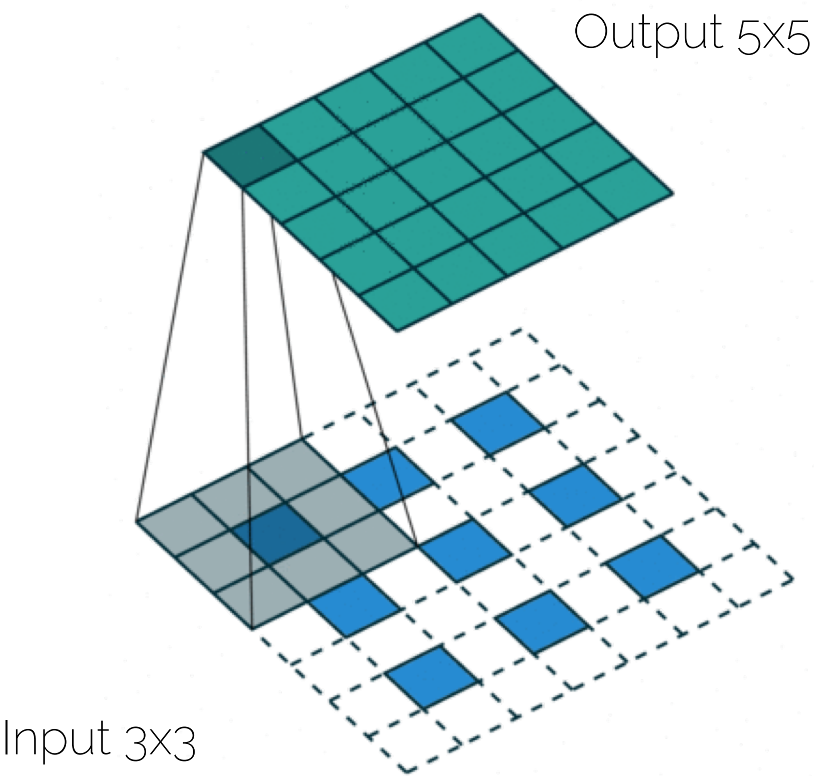
- **Encoder**: normal convolutional filters + pooling
- **Decoder**: Upsampling + convolutional filters
- The convolutional filters in the decoder are learned using backprop and their goal is to refine the upsampling

Badrinarayanan et al. „SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation“. TPAMI 2016

# Recall transposed convolution

- Transposed convolution
  - Unpooling
  - Convolution filter (learned)

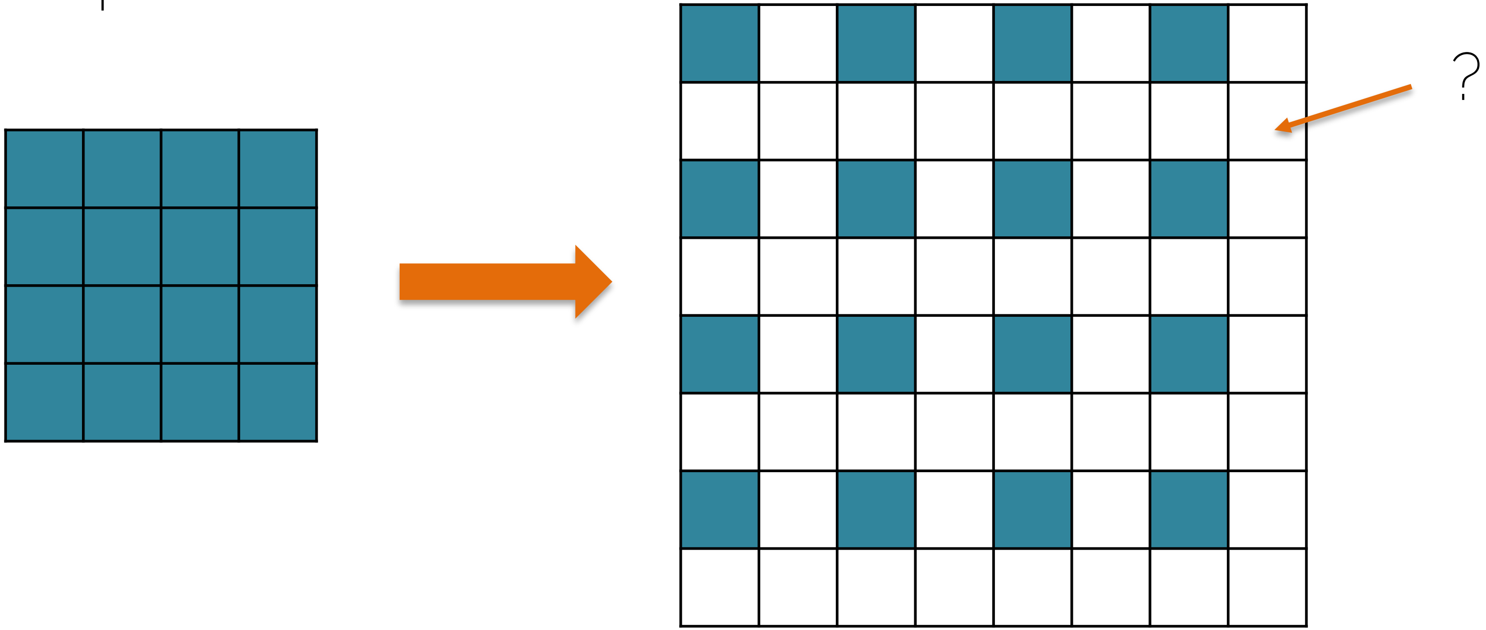
  - Also called up-convolution (never deconvolution)



# Upsampling

# Types of upsamplings

- 1. Interpolation



# Types of upsamplings

- 1. Interpolation

Original image  x 10

Nearest neighbor interpolation

Bilinear interpolation

Bicubic interpolation

Image: Michael Guerzhoy

# Types of upsamplings

- 1. Interpolation

    Few artifacts

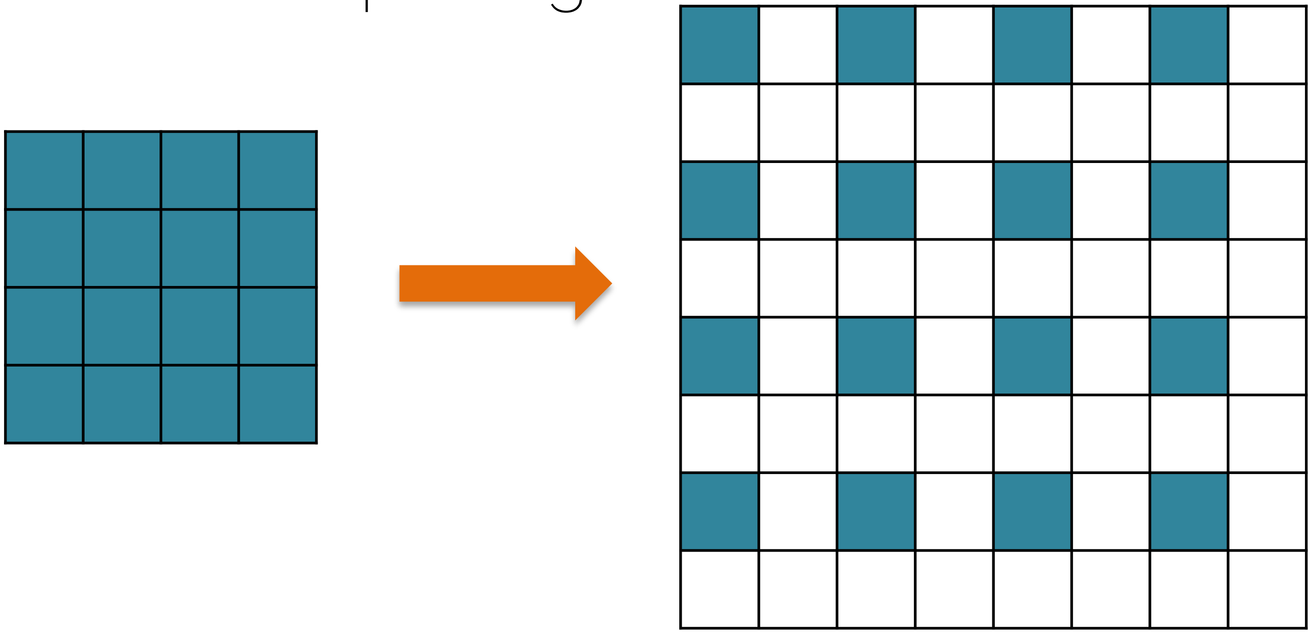


Image: Michael Guerzhoy

# Types of upsamplings

- 2. Fixed unpooling

efficient



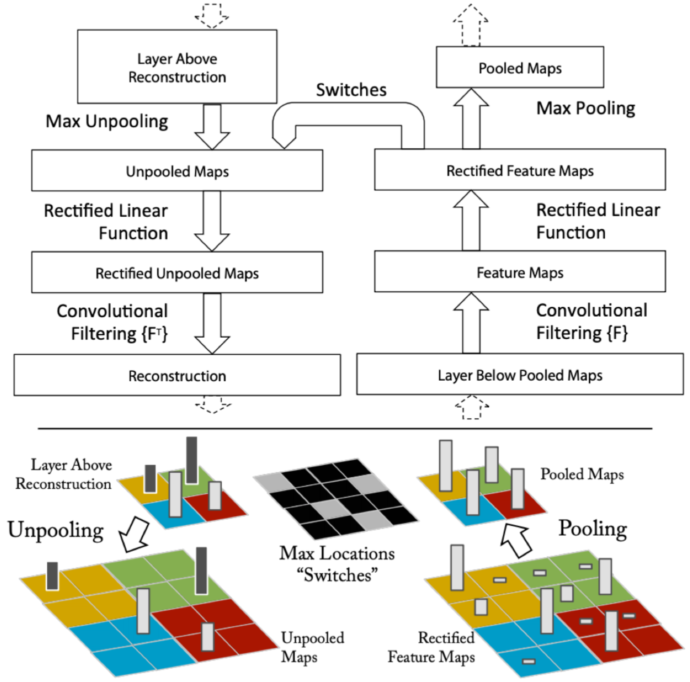
+ CONVS

A. Dosovitskiy, "Learning to Generate Chairs, Tables and Cars with Convolutional Networks". TPAMI 2017

# Types of upsamplings

- 3. Unpooling: “à la DeconvNet”

Keep the locations where the max came from

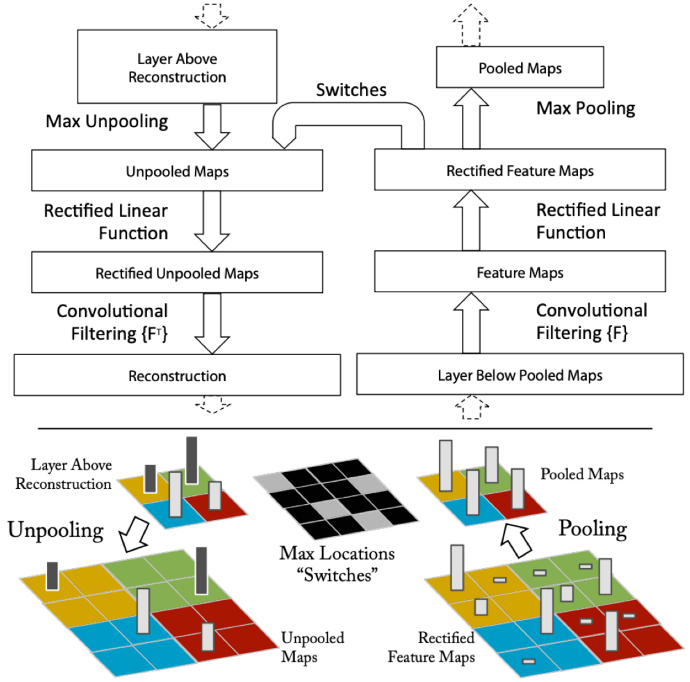


# Types of upsamplings

- 3. Unpooling: “à la DeconvNet”

Now: convolutional filters are LEARNED

In DeConvNet: we convolve with the transpose of the learned filter



# Types of upsamplings

- 3. Unpooling: “à la DeconvNet”

Keep the details of the structures

# U-Net or skip connections in autoencoders

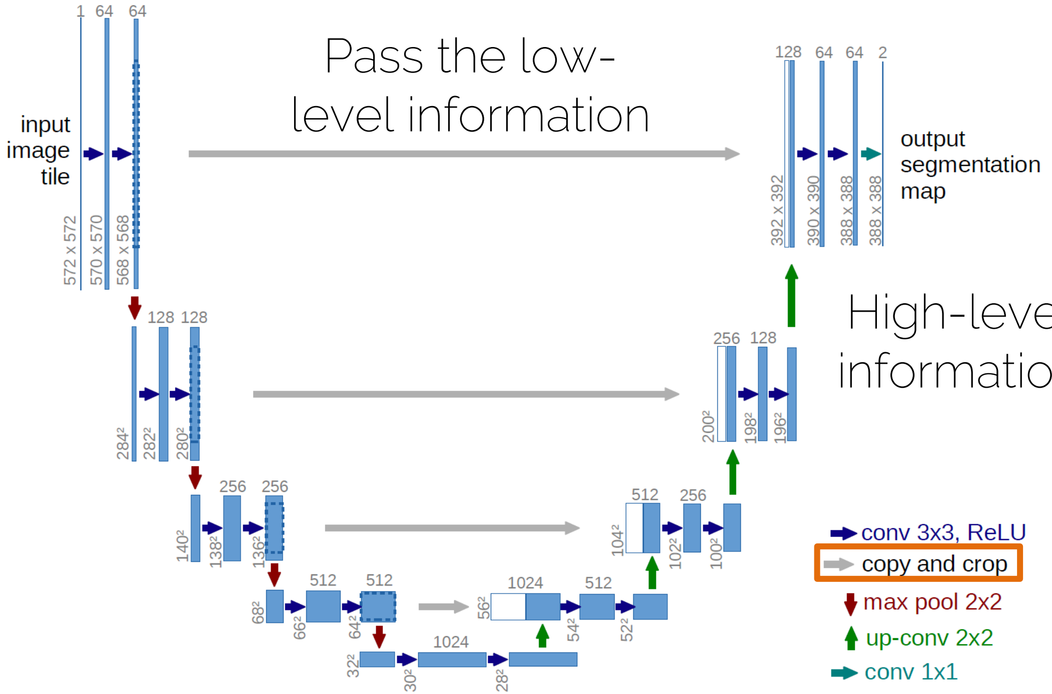
# Skip Connections

- U-Net

Recall ResNet

Pass the low-level information

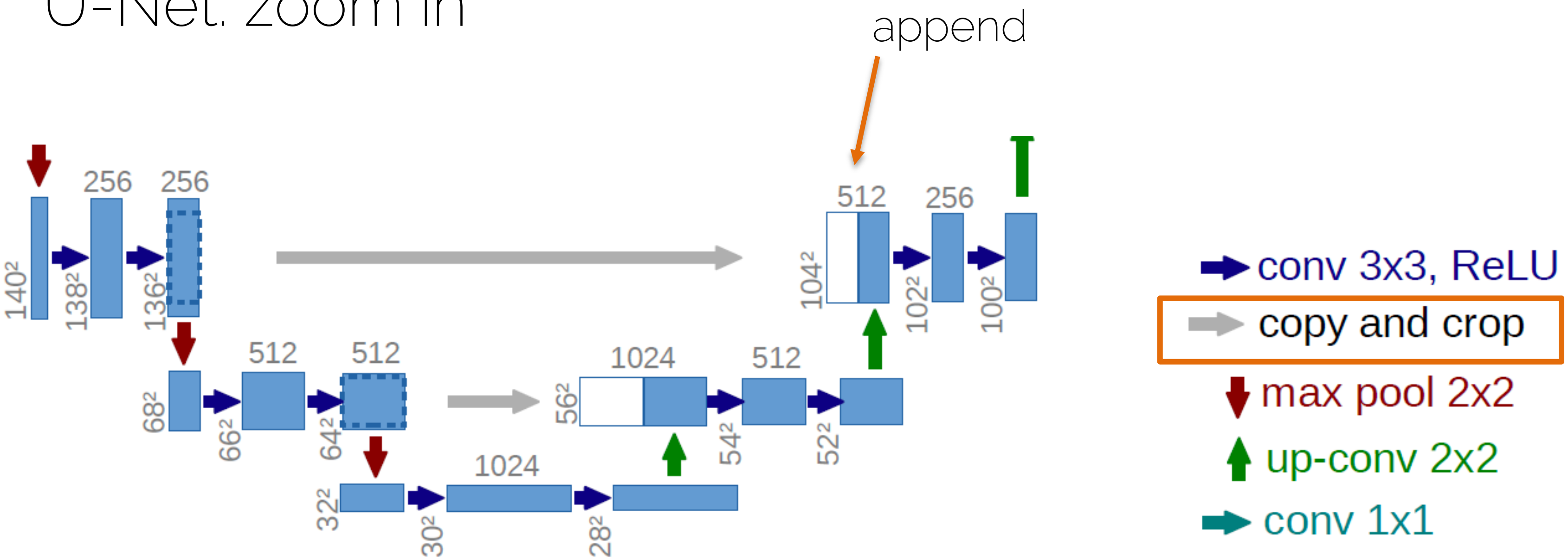
High-level information




O. Ronneberger et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation". MICCAI 2015

# Skip Connections

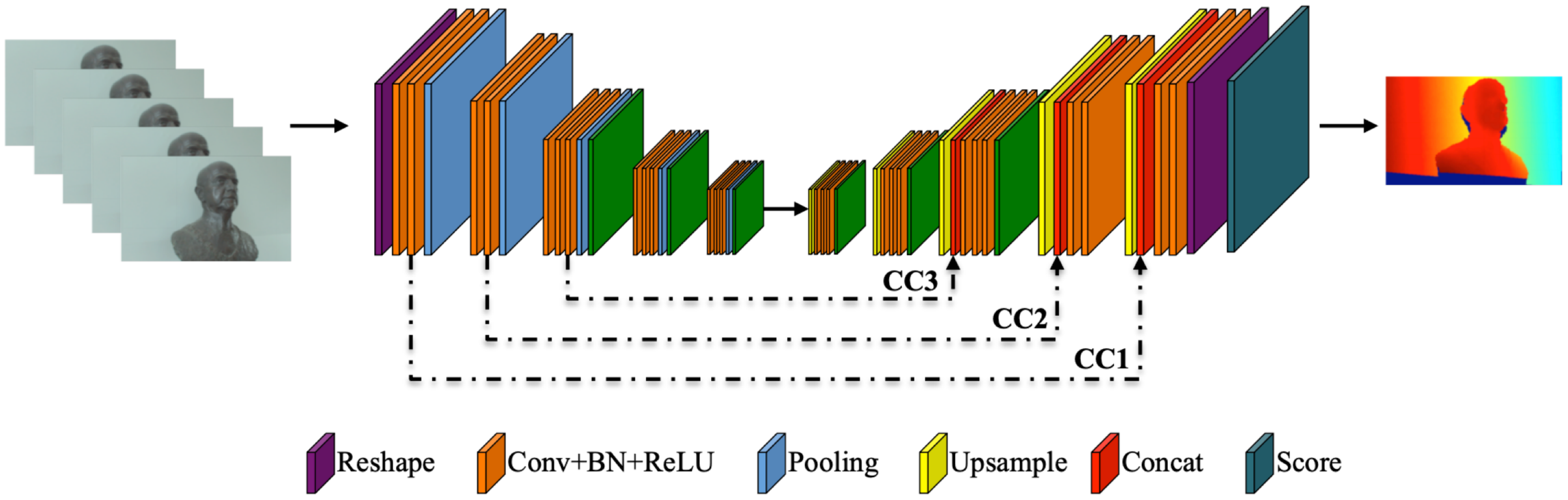
- U-Net: zoom in




O. Ronneberger et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation". MICCAI 2015

# Skip Connections

- Concatenation connections






C. Hazirbas et al. "Deep depth from focus". ACCV 2018

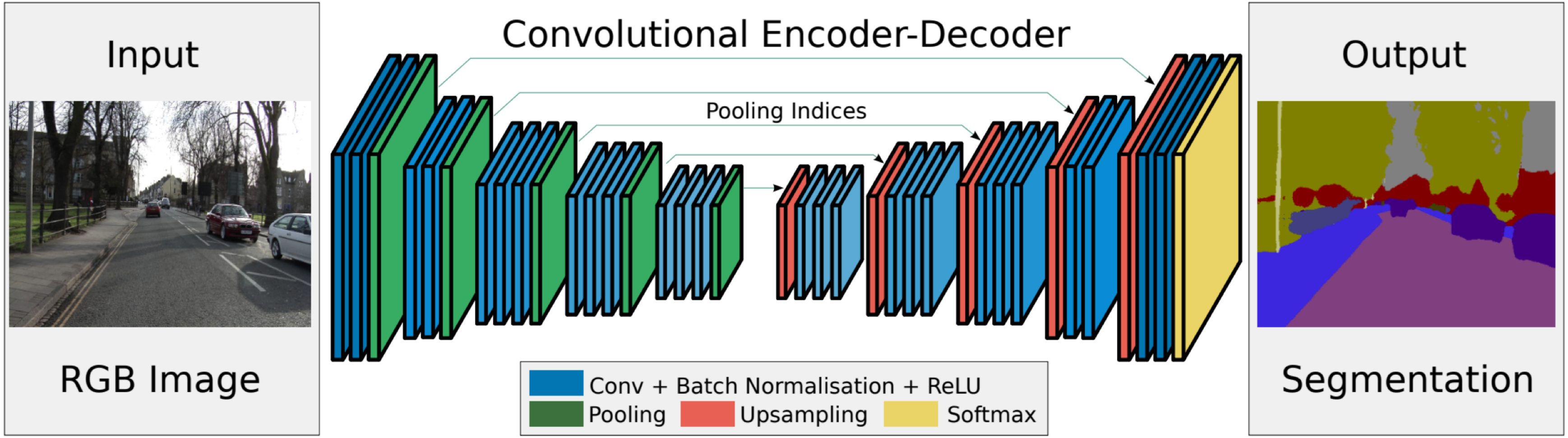
# Skip Connections

- Widely used in Autoencoders

- At what levels the skip connections are needed depends on your problem

# Autoencoders in Vision

# SegNet



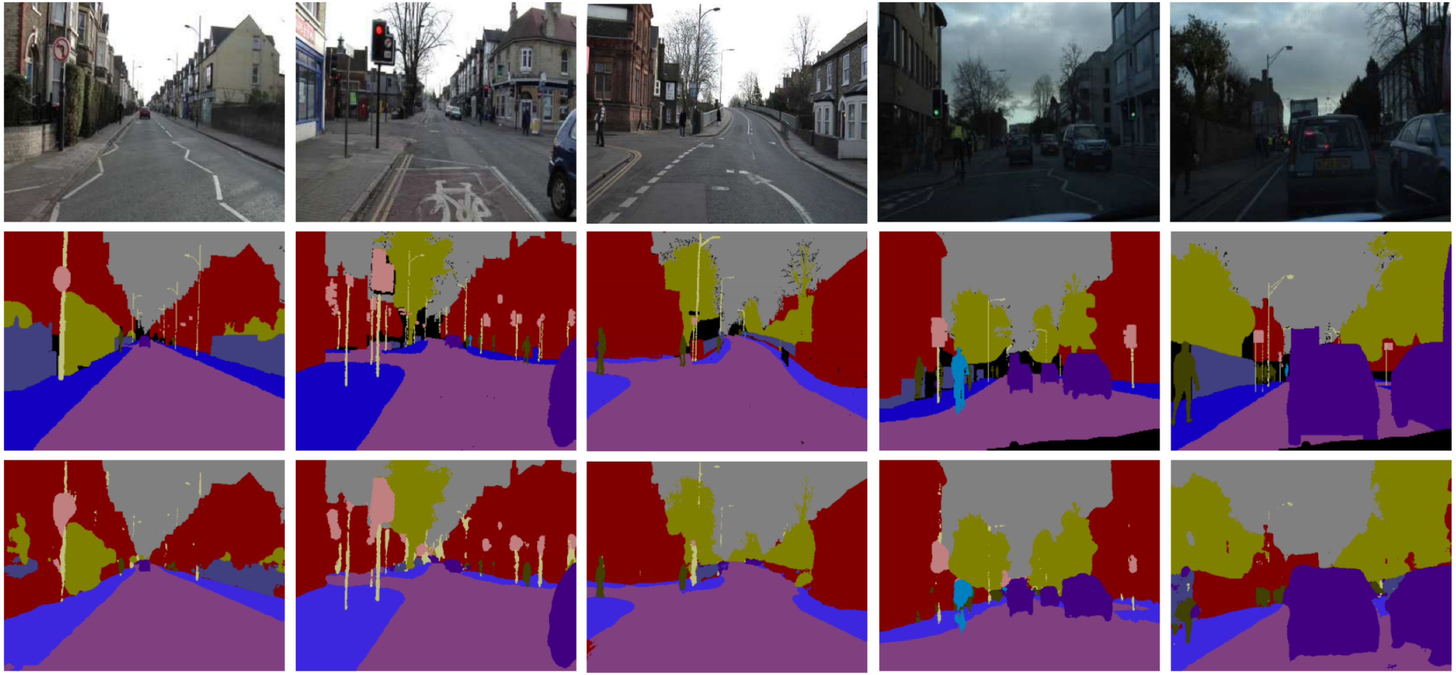

Badrinarayanan et al. „SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". TPAMI 2016

# SegNet

Input

Ground truth

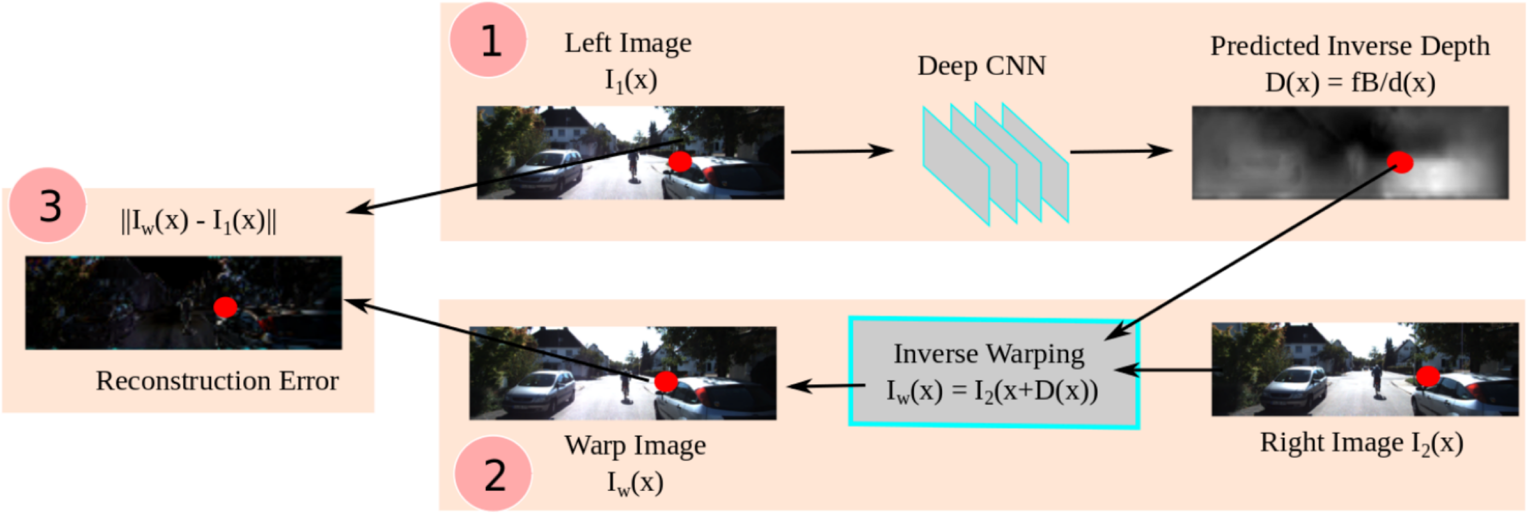
SegNet



Badrinarayanan et al. „SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". TPAMI 2016

# Monocular depth

- Unsupervised monocular depth estimation




R. Garg et al. „Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue" ECCV 2016

# Image super resolution

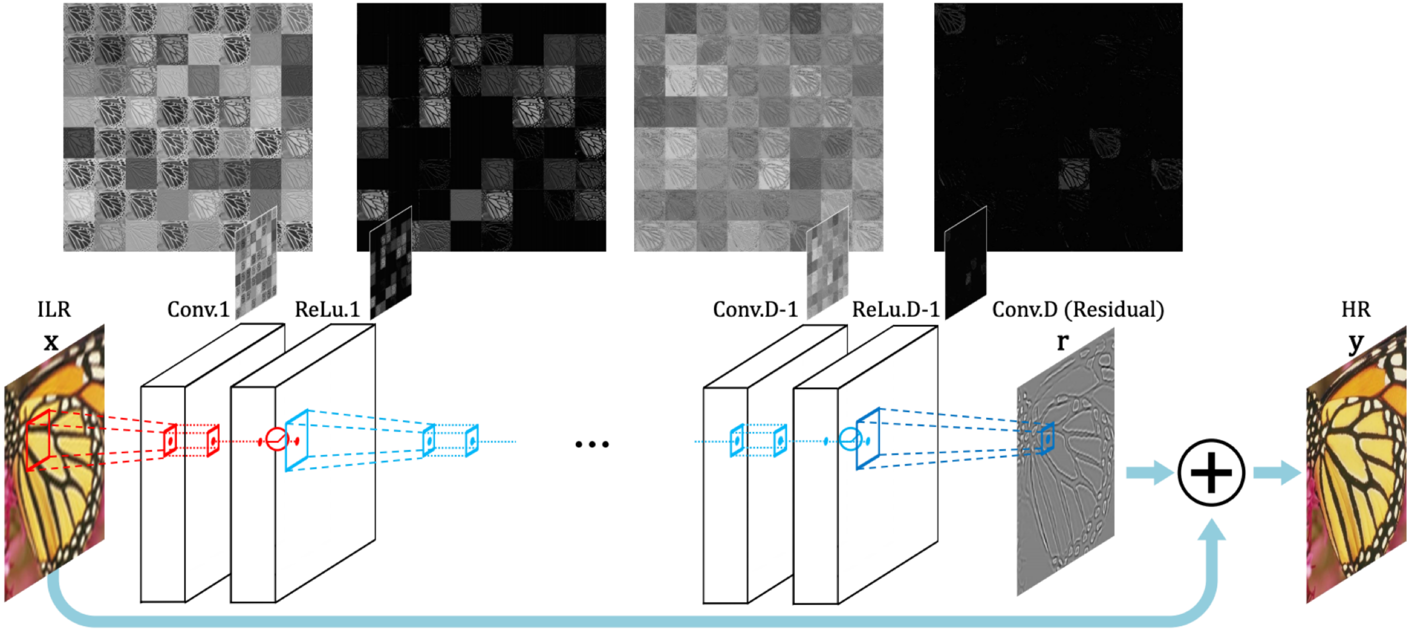
- Image in low resolution → Image in high resolution
- Problems:
  - The content of the image needs to pass through the network (skip connections [2] or other strategies [1]).

[1] C. Dong et al. „Image Super-Resolution Using Deep Convolutional Networks“. TPAMI 2015

[2] XJ. Mao et al. “Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections“. NIPS 2016

# Image super resolution

- Why not learning the residual only? → Much easier!



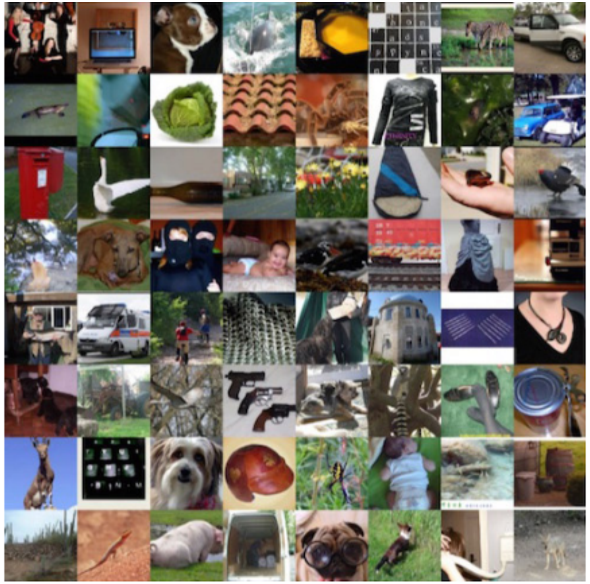

J. Kim et al. „Accurate Image Super-Resolution Using Very Deep Convolutional Networks". CVPR 2016

# Generative models

# Generative models

- Given training data, I want to generate new samples from the same distribution

Real Images



Generated Images

Source: https://openai.com/blog/generative-models/

# Taxonomy of generative models

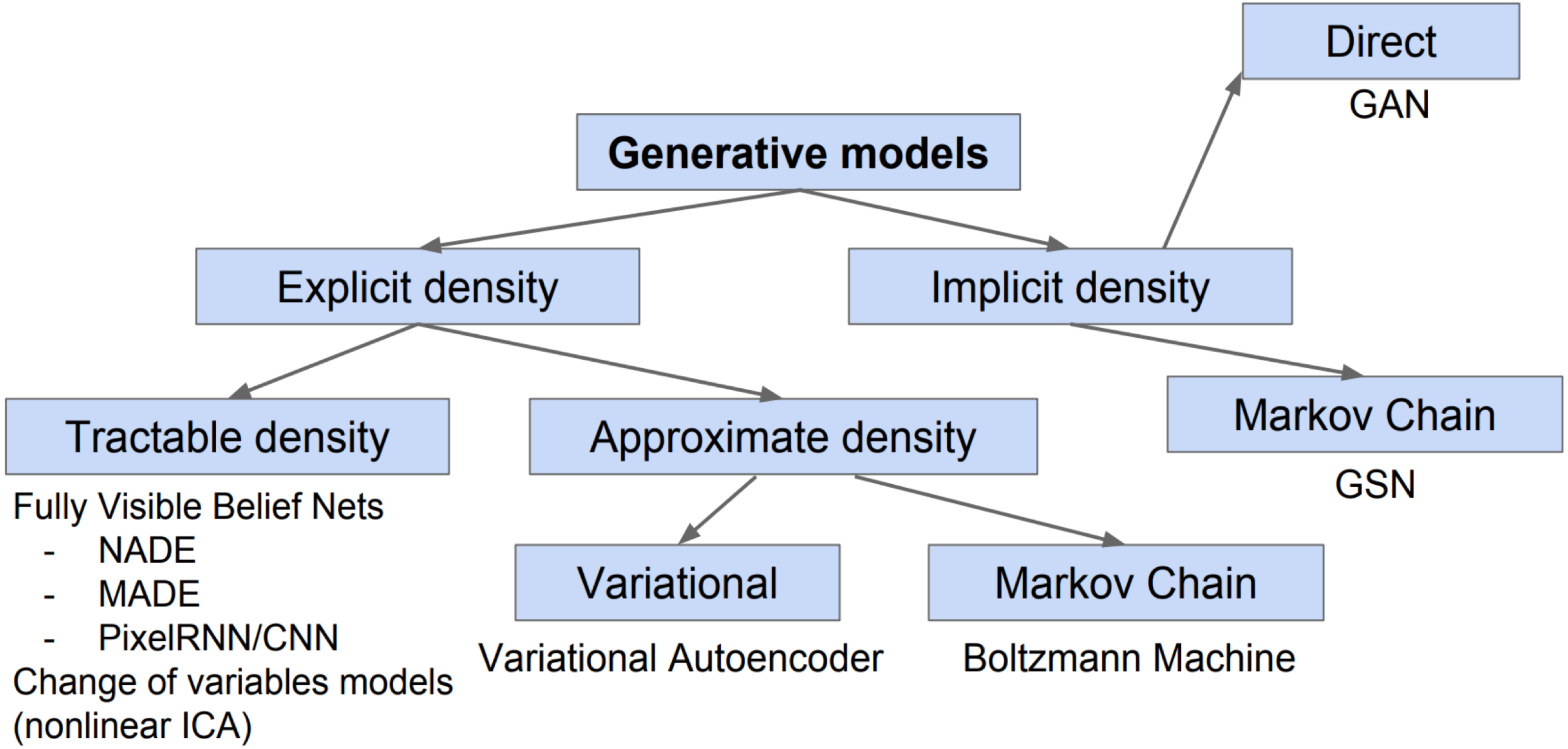



Figure from Ian Goodfellow, Tutorial on Generative Adversarial /networks, 2017

# Taxonomy of generative models

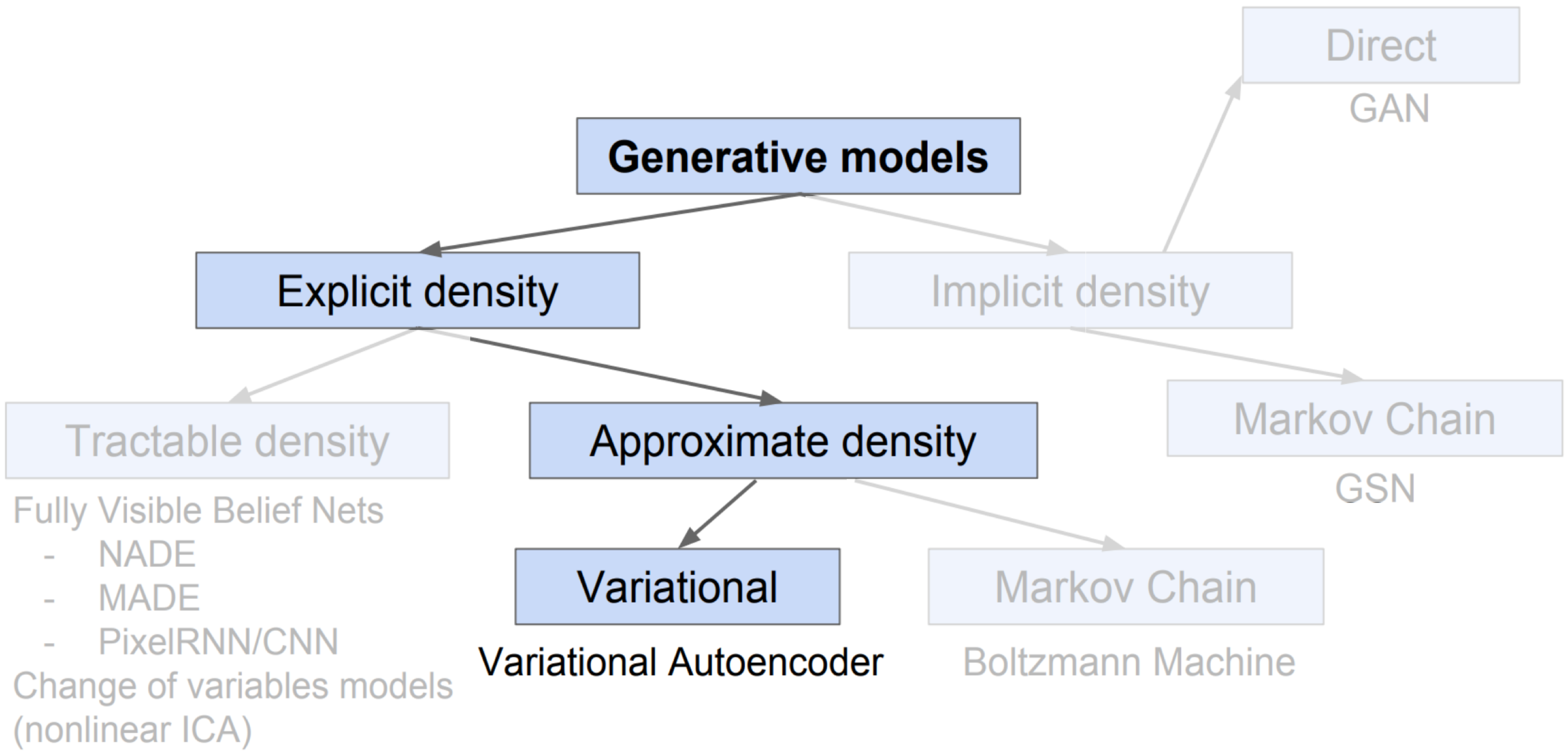
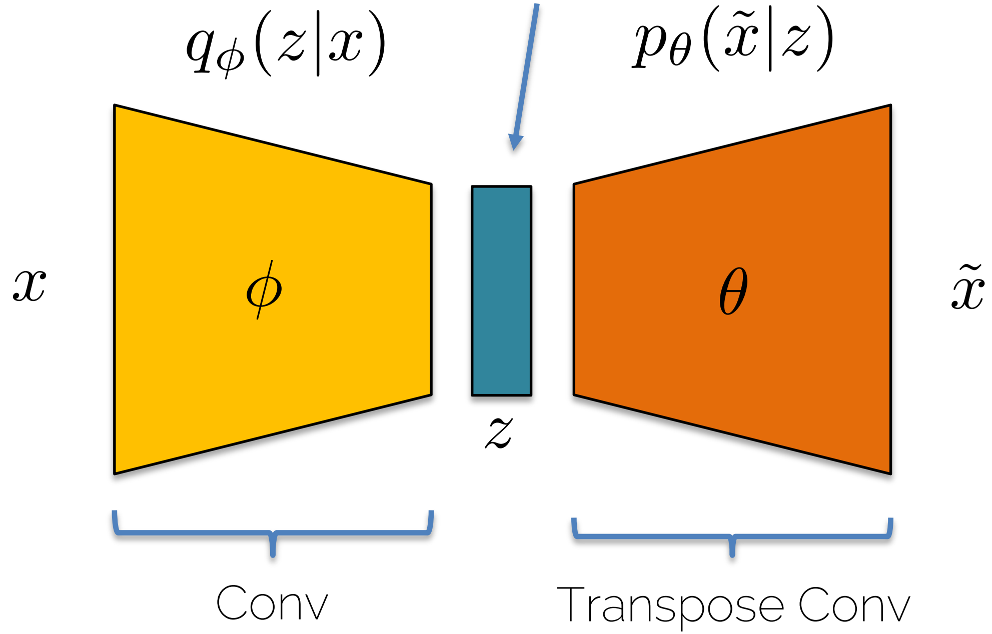



Figure from Ian Goodfellow, Tutorial on Generative Adversarial /networks, 2017

# Variational Autoencoders

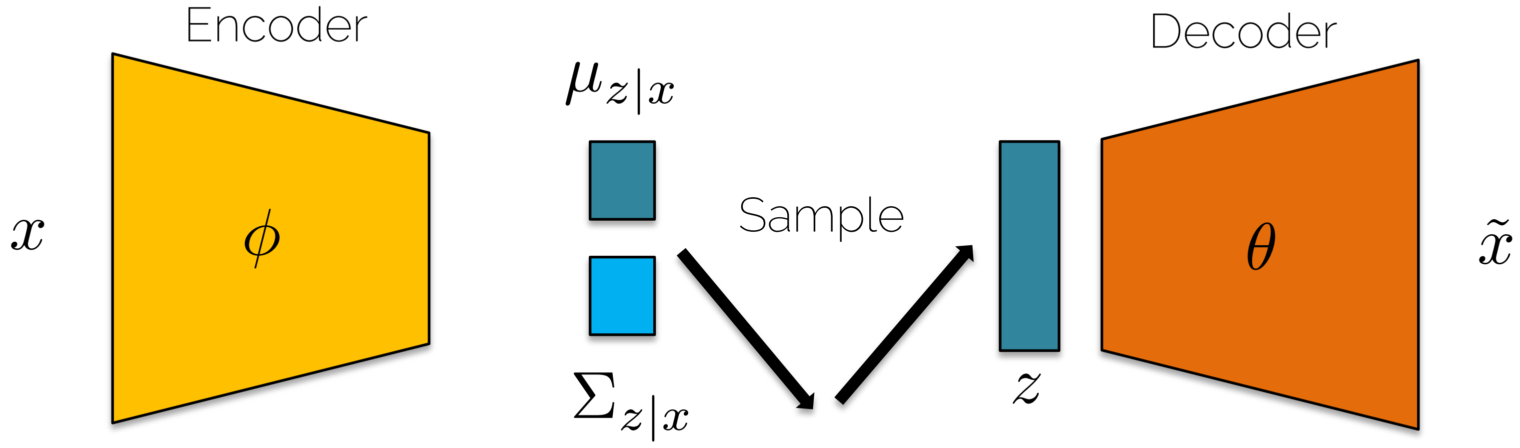
# Variational Autoencoder

Goal: Sample from the latent distribution to generate new outputs!



# Variational Autoencoder

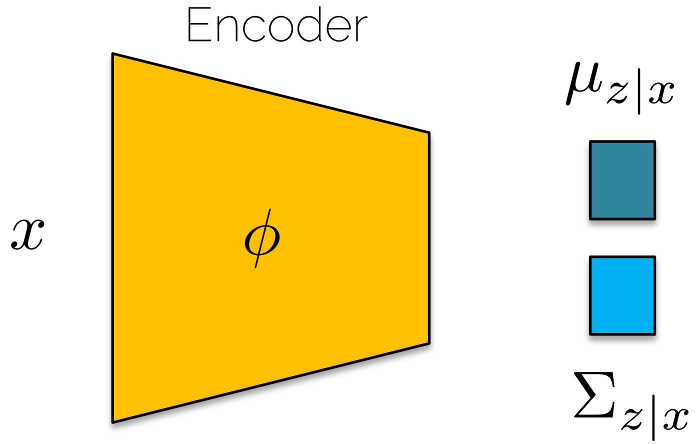
- Latent space is now a distribution
- Specifically it is a Gaussian




$$z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$$

# Variational Autoencoder

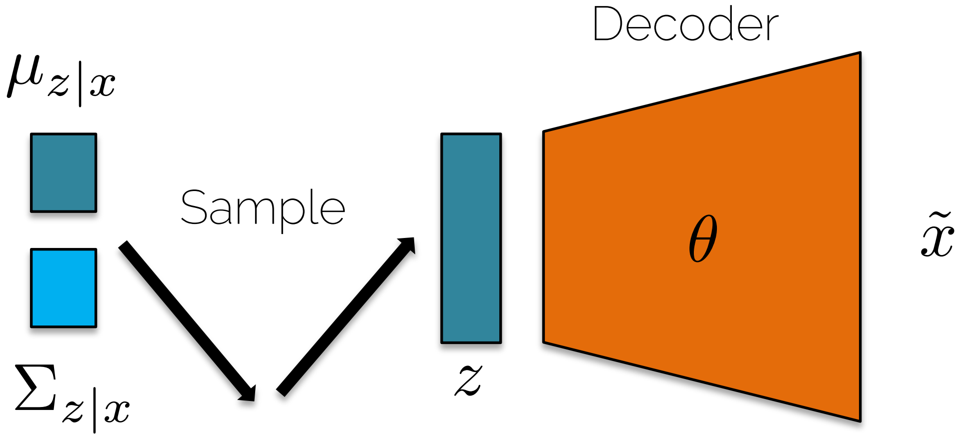
- Latent space is now a distribution
- Specifically it is a Gaussian




$$z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$$

# VAE: testing

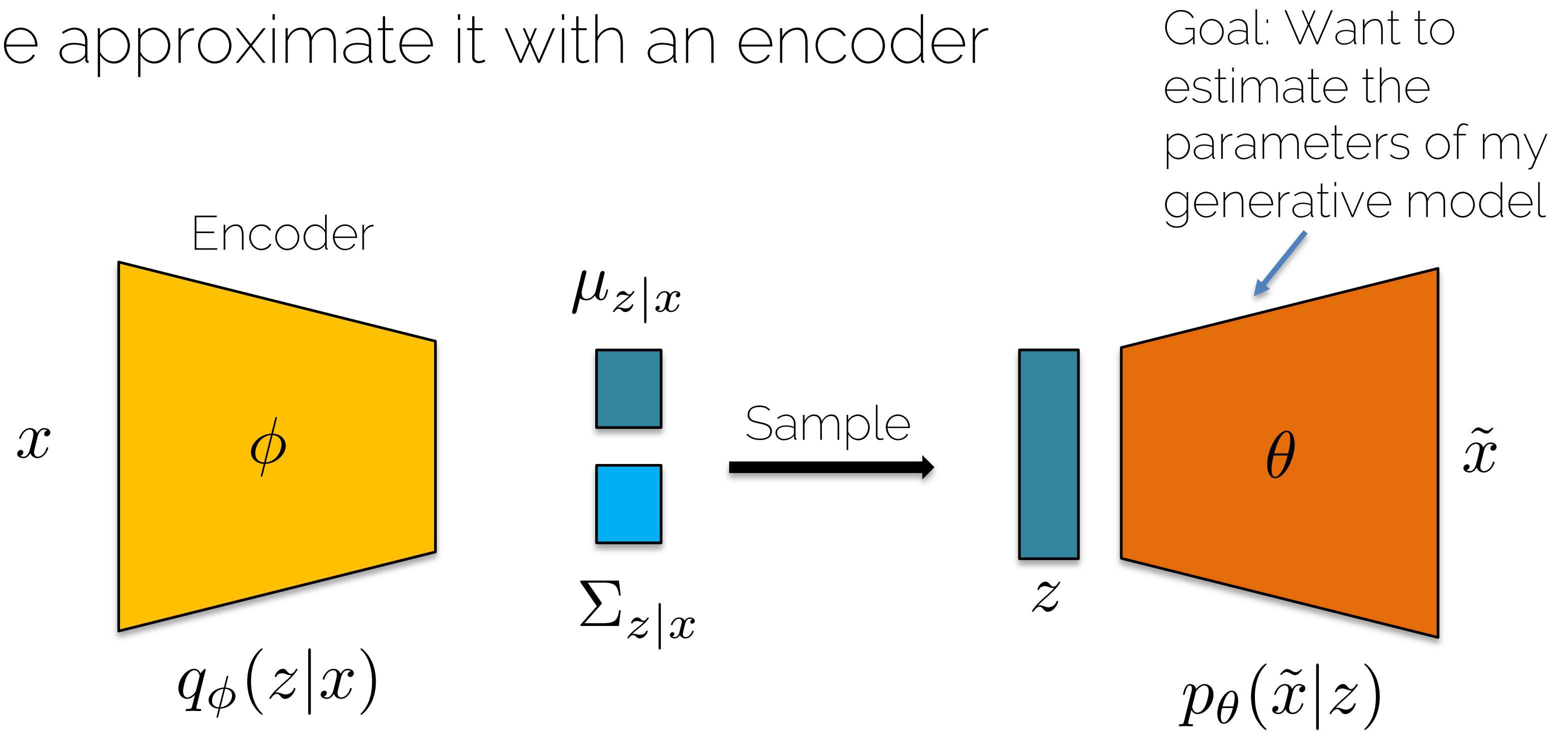
- Test: sampling from the latent space




$$z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$$

# VAE: training

- We approximate it with an encoder



# VAE: loss function

- Loss function for a data point $x_i$

$$\log(p_\theta(x_i)) = \boldsymbol{E}_{z \sim q_\phi(z|x_i)}[\log(p_\theta(x_i))]$$

I draw samples of the latent variable z from my encoder

decoder

# VAE: loss function

- Loss function for a data point $x_i$

$$\log(p_\theta(x_i)) = \boldsymbol{E}_{z\sim q_\phi(z|x_i)}[\log(p_\theta(x_i))]$$

$$= \boldsymbol{E}_{z\sim q_\phi(z|x_i)}\left[\log \frac{p_\theta(x_i|z)p_\theta(z)}{p_\theta(z|x_i)}\right]$$

Bayes Rule

Recall:

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

Using the latent variable, which will become useful to simplify the expressions later according to our AE formulation

# VAE: loss function

- Loss function for a data point $x_i$

$$\log(p_\theta(x_i)) = \boldsymbol{E}_{z\sim q_\phi(z|x_i)}[\log(p_\theta(x_i))]$$

$$= \boldsymbol{E}_{z\sim q_\phi(z|x_i)}\left[\log \frac{p_\theta(x_i|z)p_\theta(z)}{p_\theta(z|x_i)}\right]$$

$$= \boldsymbol{E}_z\left[\log \frac{p_\theta(x_i|z)p_\theta(z)}{p_\theta(z|x_i)} \boxed{\frac{q_\phi(z|x_i)}{q_\phi(z|x_i)}}\right]$$

Just a constant

# VAE: loss function

- Loss function for a data point $x_i$

$$\log(p_\theta(x_i)) = \boldsymbol{E}_z \left[\log \frac{\boxed{p_\theta(x_i|z)}\boxed{p_\theta(z)}}{\boxed{p_\theta(z|x_i)}} \frac{\boxed{q_\phi(z|x_i)}}{\boxed{q_\phi(z|x_i)}}\right]$$

$$= \boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{\boldsymbol{E}_z\left[\log \frac{q_\phi(z|x_i)}{p_\theta(z)}\right]} + \boxed{\boldsymbol{E}_z\left[\log \frac{q_\phi(z|x_i)}{p_\theta(z|x_i)}\right]}$$

Apply the logarithm and group as needed

# VAE: loss function

- Loss function for a data point $x_i$

$$= \boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{\boldsymbol{E}_z\left[\log \frac{q_\phi(z|x_i)}{p_\theta(z)}\right]} + \boxed{\boldsymbol{E}_z\left[\log \frac{q_\phi(z|x_i)}{p_\theta(z|x_i)}\right]}$$

Kullback-Leibler Divergences to measure how similar two distributions are

# VAE: loss function

- Loss function for a data point $x_i$

$$= \boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{\boldsymbol{E}_z\left[\log \frac{q_\phi(z|x_i)}{p_\theta(z)}\right]} + \boxed{\boldsymbol{E}_z\left[\log \frac{q_\phi(z|x_i)}{p_\theta(z|x_i)}\right]}$$

$$= \boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$

Kullback-Leibler Divergences

# VAE: loss function

- Loss function for a data point $x_i$

$$= \boxed{\boldsymbol{E}_z \left[ \log p_\theta(x_i|z) \right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$

Reconstruction loss (how well does my decoder reconstruct a data point given the latent vector z). We need to sample from z.

Measures how good my latent distribution is with respect to my Gaussian prior

I still cannot express the shape of the distribution. But I know

$\geq 0$

# VAE: loss function

- Loss function for a data point $x_i$

$$\underbrace{\boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))}}_{\substack{\text{Loss function (lower bound)} \\ \mathcal{L}(x_i,\phi,\theta)}} + \underbrace{\boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}}_{\geq 0}$$

$$\log(p(x_i)) \geq \mathcal{L}(x_i,\phi,\theta)$$

# VAE: loss function

- Loss function for a data point $x_i$

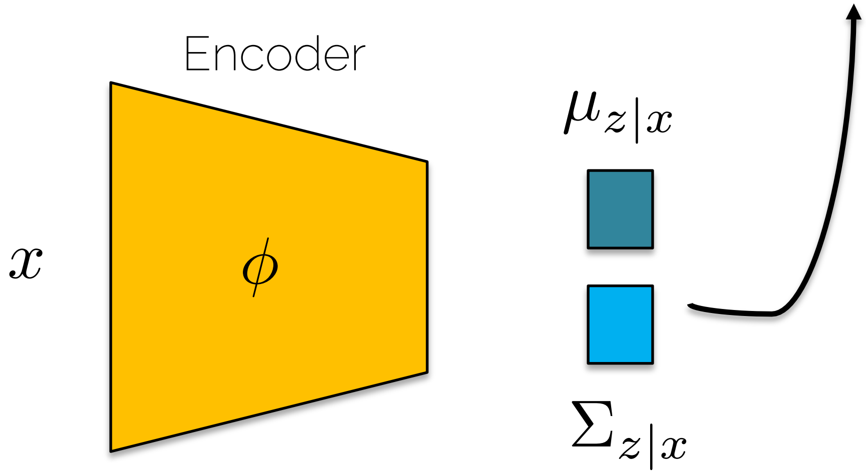
$$\underbrace{\boxed{\boldsymbol{E}_z \left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))}}_{\substack{\text{Loss function (lower bound)} \\ \mathcal{L}(x_i, \phi, \theta)}} + \underbrace{\boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}}_{\geq 0}$$

- Optimize $\phi^*, \theta^* = \arg\max \sum_{i=1}^{N} \mathcal{L}(x_i, \phi, \theta)$

# Variational Autoencoder

- Training

$$\boxed{\boldsymbol{E}_z \left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$

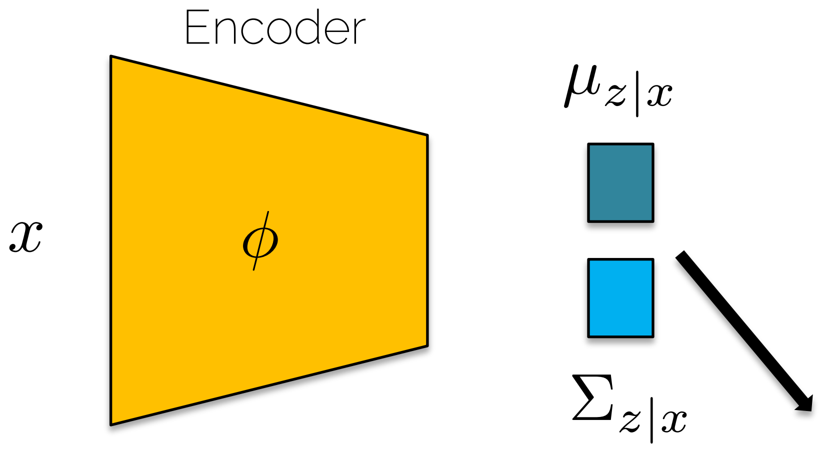



Make posterior distribution close to prior (close to unit Gaussian distribution)

# Variational Autoencoder

- Training

$$\boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$

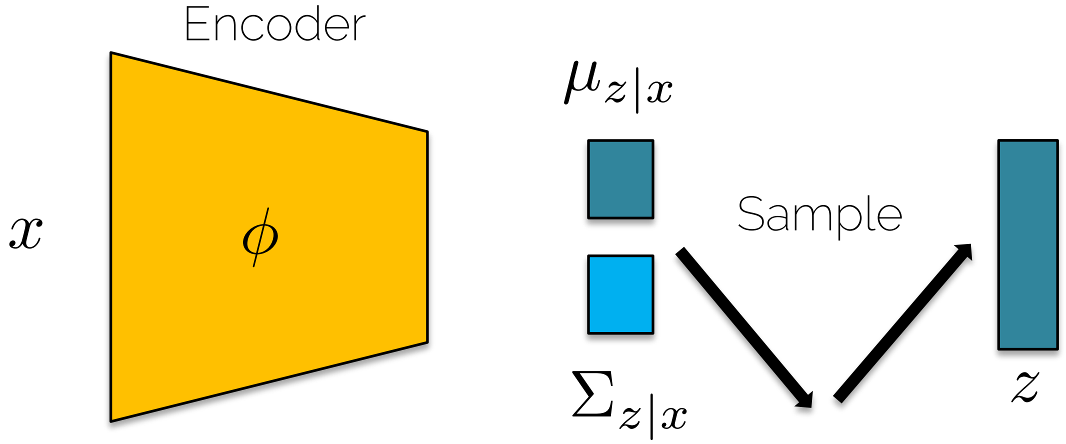



$$z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$$

# Variational Autoencoder

- Training

$$\boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$

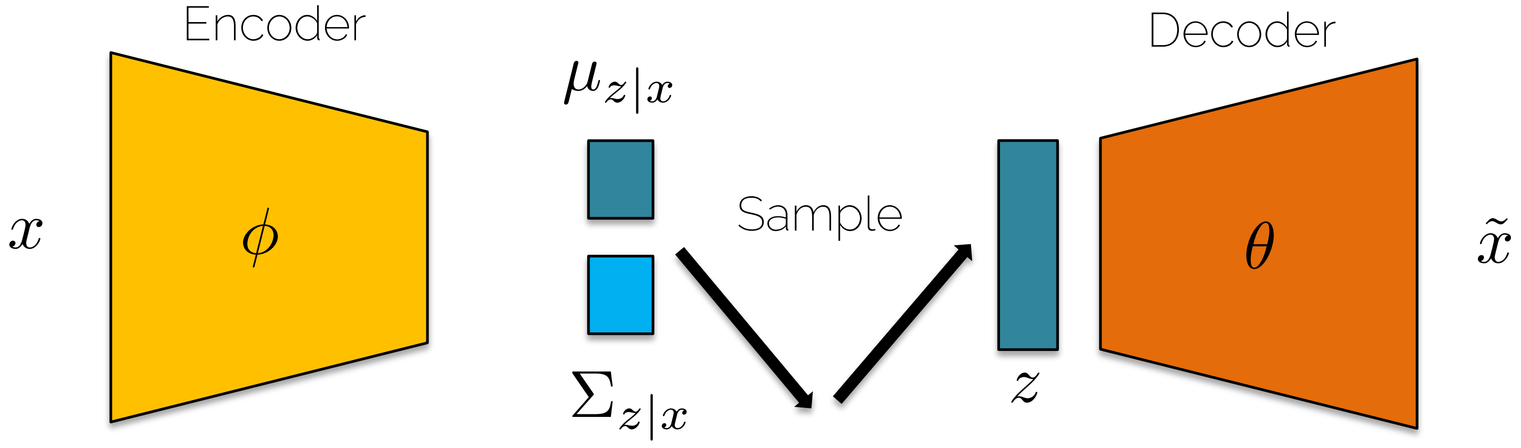



$$z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$$

# Variational Autoencoder

- Training

$$\boxed{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$




$$z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$$

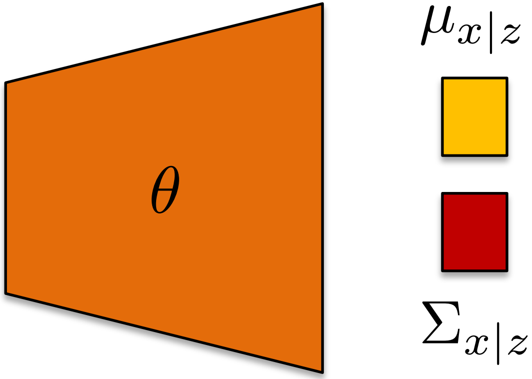
# Variational Autoencoder

- Training

$$\underbrace{\boldsymbol{E}_z\left[\log p_\theta(x_i|z)\right]}_{} - KL(q_\phi(z|x_i)||p_\theta(z)) + KL(q_\phi(z|x_i)||p_\theta(z|x_i))$$

Decoder

$\mu_{x|z}$




$\Sigma_{x|z}$

$$x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z}) \xrightarrow{\text{Sample}} \tilde{x}$$

Output is also parameterized

# Variational Autoencoder

- Training

$$\boxed{\boldsymbol{E}_z \left[\log p_\theta(x_i|z)\right]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$

$\tilde{x}$

Maximize the likelihood of reconstructing the input

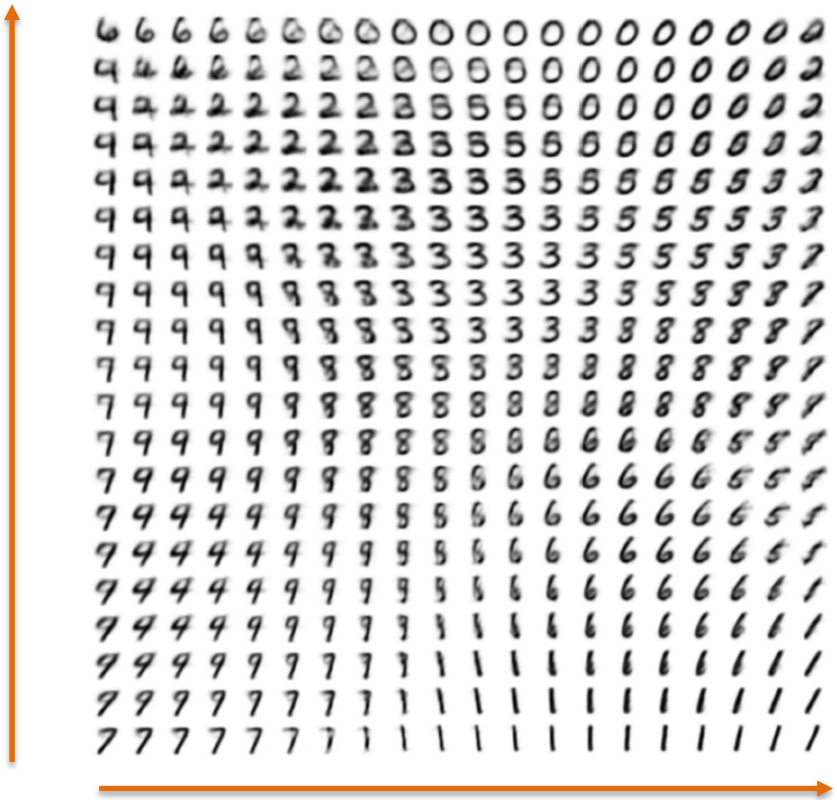
# Variational Autoencoder

- For more details and mathematical derivation

- Reparameterization trick (expressing variables as Gaussians) that allows us to perform backpropagation

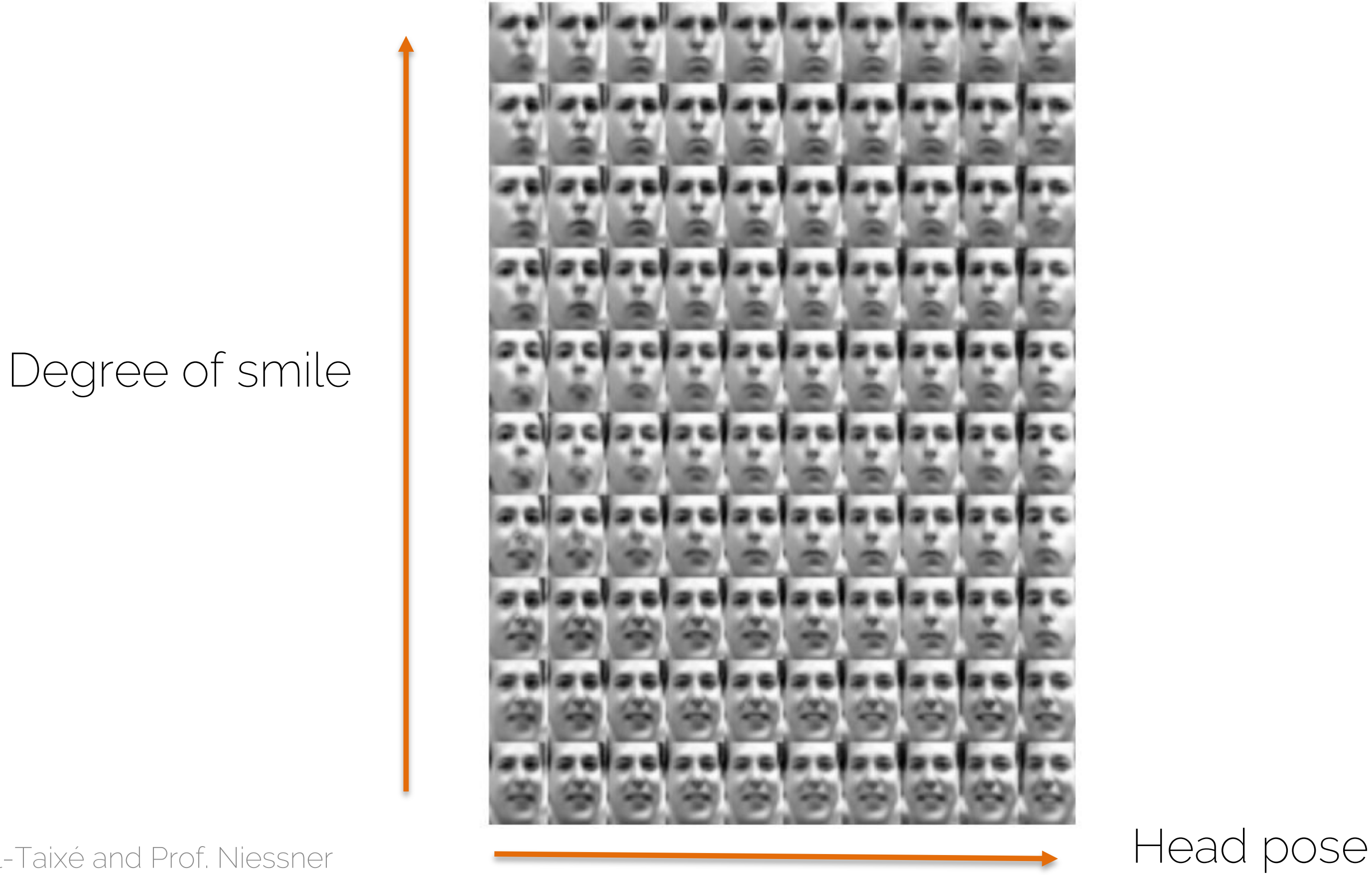
- Kingman and Welling. "Auto-Encoding Variational Bayes". ICLR 2014

# Generating data



Each element of z encodes a different feature

# Generating data



# Autoencoder vs VAE

Autoencoder

Variational Autoencoder

Ground Truth



https://github.com/kvfrans/variational-autoencoder

# Autoencoder Overview

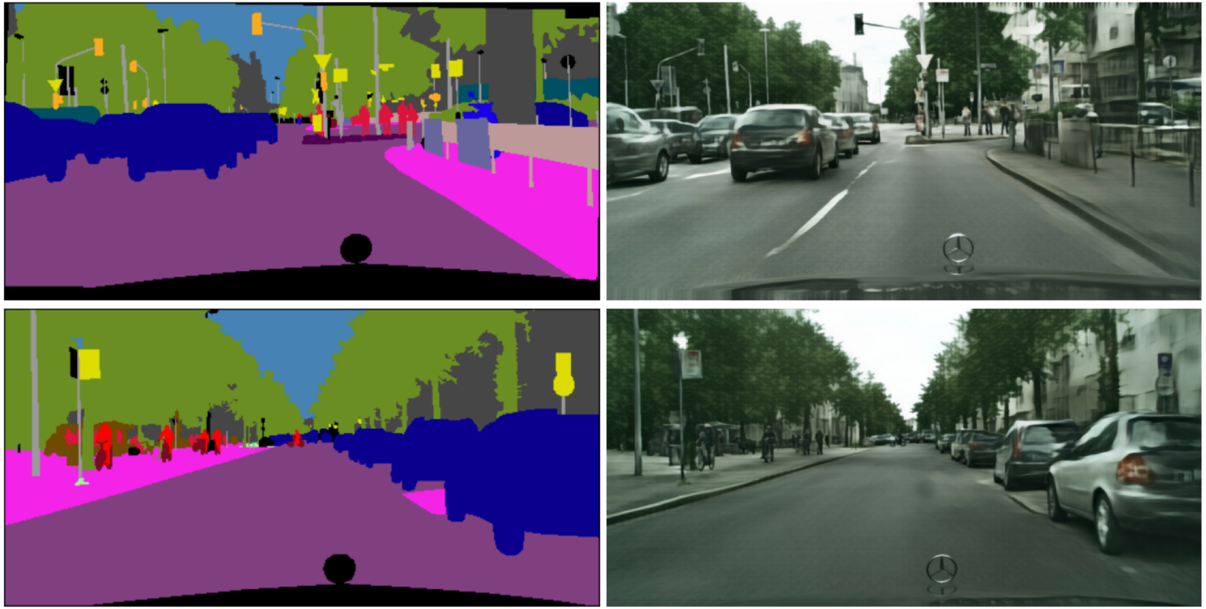
- Autoencoders (AE)
  - Reconstruct input
  - Unsupervised learning
  - Latent space features are useful

- Variational Autoencoders (VAE)
  - Probability distribution in latent space (e.g., Gaussian)
  - Interpretable latent space (head pose, smile)
  - Sample from model to generate output

# Image synthesis (without GANs?)

# Image synthesis

- Semantic segmentation image → Real image



(a) Input semantic layouts (b) Synthesized images

Q. Chen and V. Koltun „Photographic Image Synthesis with Cascaded Refinement Networks“. ICCV 2017

# Image synthesis

- Semantic segmentation image → Real image
- No GANs?

Q. Chen and V. Koltun „Photographic Image Synthesis with Cascaded Refinement Networks“. ICCV 2017

# Image synthesis

- Several works show that one can use a *perceptual loss* to achieve high quality results
- Cannot use the L2 loss as this could penalize realistic results (black car vs white car)
- Perceptual loss measures the „content of the image"

A. Dosovitskiy and T. Brox. „Generating Images with Perceptual Similarity Metrics based on Deep Networks". NIPS 2016

Q. Chen and V. Koltun „Photographic Image Synthesis with Cascaded Refinement Networks". ICCV 2017

# Perceptual loss and style transfer

# Content loss

- Content loss (or perceptual loss or feature reconstruction loss).
- Use a network to compute the loss

$$\ell_{feat}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

Gatys et al „A neural algorithm of artistic style". arXiv preprint arXiv:1508.06576 (2015)

# Content loss

- 1. Take a VGG network trained for image classification
- 2. Pass the generated image and the ground truth through the network
- 3. Compare the feature maps

Feature maps of the generated image at layer j

$$\ell_{feat}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

Feature map size (channels, height, width)

Feature maps of the ground truth image at layer j

# Content loss

- Intuition: if there was a car in the original image, we want to have "similar" features triggered for the generated image

- This means we want to "roughly see a car" in the generated image too (but, e.g., color does not matter)

# Style Transfer

- The content loss was originally introduced for style transfer [1]


Image: J. Johnson

[1] Gatys et al „A neural algorithm of artistic style". arXiv preprint arXiv:1508.06576 (2015)


A
B
E
F

# Style Transfer

- Content loss: feature representation similarity

- Style loss:

Gram matrix of the features of layer j

$$\ell_{style}^{\phi,j}(\hat{y}, y) = \|G_j^{\phi}(\hat{y}) - G_j^{\phi}(y)\|_F^2$$

- Comparing Gram matrices

J. Johnson at al. „Perceptual losses for real-time style transfer and super-resolution“ ECCV 2016

Gatys et al „A neural algorithm of artistic style“. arXiv preprint arXiv:1508.06576 (2015)

# Style loss

- 1. Take a VGG network trained for image classification
- 2. Pass the generated image and the ground truth through the network
- 3. Compute the Gram matrices at a certain layer

$$G_j^{\phi}(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'}$$

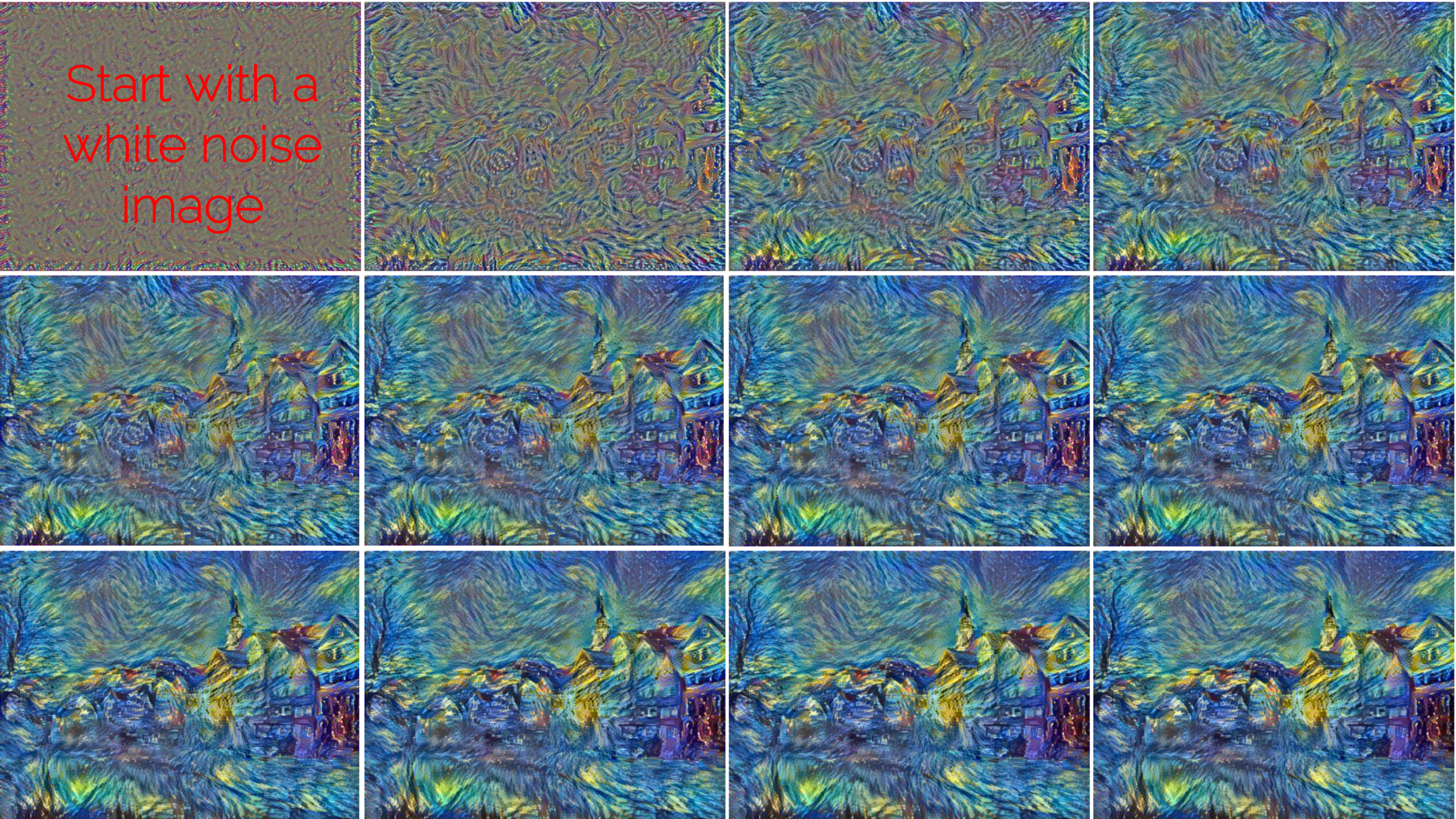
- Comparing channels c and c'

# Style loss

- Intuition: it captures information about which features tend to activate *together*.

$$G_j^{\phi}(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'}$$

- This loss preserves the stylistic features but not the content


Start with a white noise image

# Style Transfer

More weight to the content loss

More weight to the style loss

Image: Johnson/Fei-Fei/ Yeung

# Style Transfer

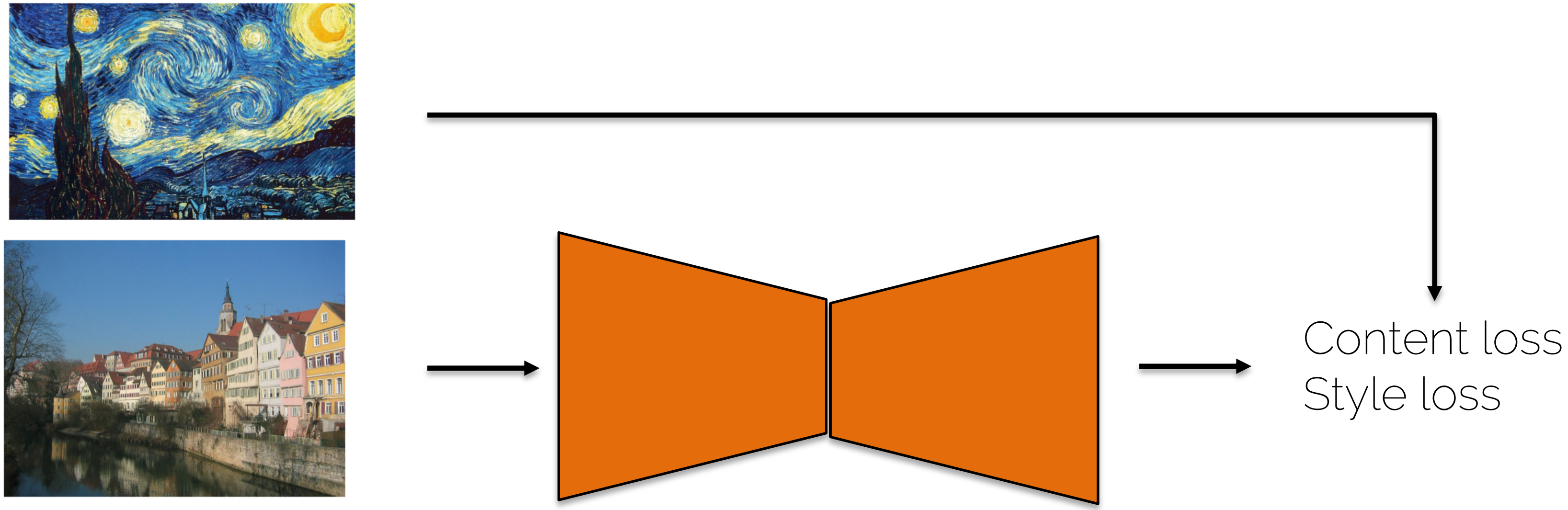
- The aforementioned method is slow, requires many forward/backward passes through VGG.

- Fast Neural style transfer → Train a Neural network to do the transfer (one network per style)

J. Johnson at al. „Perceptual losses for real-time style transfer and super-resolution“ ECCV 2016

# Fast style transfer

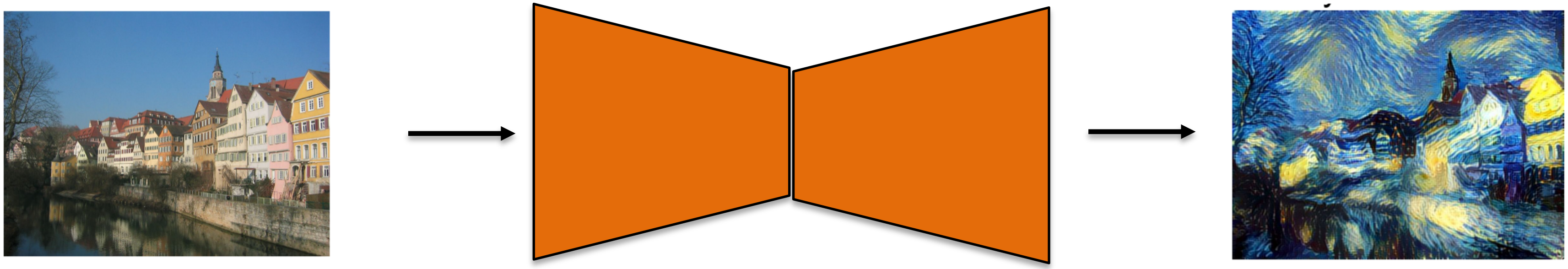
- Training: use multiple content images, use the style image to compute the loss



# Fast style transfer

- Training: use multiple content images, use the style image to compute the loss

- Test: one forward pass is enough!



# Autoencoders
# &
# VAE

# Other uses of autoencoders

- Anomaly detection. For example: C. Baur et al. „Deep Autoencoding Models for Unsupervised Anomaly Segmentation in Brain MR Images" MICCAI 2018

- Deep multimodal autoencoders → to mix the representation of several sources (audio and video)

# Other references

- Conditional Variational Autoencoders:
  - Sohn, Kihyuk, Honglak Lee, and Xinchen Yan. "Learning Structured Output Representation using Deep Conditional Generative Models." Advances in Neural Information Processing Systems. 2015.
  - Xinchen Yan, Jimei Yang, Kihyuk Sohn, Honglak Lee, Attribute2Image: Conditional Image Generation from Visual Attributes, ECCV, 2016 –

# Other references

- Interesting read:
  - Jacob Walker, Carl Doersch, Abhinav Gupta, Martial Hebert, An Uncertain Future: Forecasting from Static Images using Variational Autoencoders, ECCV, 2016
  - Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, Ole Winther, Autoencoding beyond pixels using a learned similarity metric, ICML, 2016
  - Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, David Forsyth, Learning Diverse Image Colorization, arXiv, 2016
  - Raymond Yeh, Ziwei Liu, Dan B Goldman, Aseem Agarwala, Semantic Facial Expression Editing using Autoencoded Flow, arXiv, 2016
  - Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling, Semi-Supervised Learning with Deep Generative Models, NIPS, 2014