Autoencoders
Machine learning

Unsupervised learning

Supervised learning

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

Prof. Leal-Taixé and Prof. Niessner
Machine learning

Unsupervised learning

Supervised learning

CAT

DOG

CAT

CAT

DOG

Prof. Leal-Taixé and Prof. Niessner
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

CAT

DOG

CAT

DOG

CAT

DOG

Prof. Leal-Taixé and Prof. Niessner
Machine learning

Unsupervised learning

Supervised learning

CAT

DOG

CAT

DOG

CAT

DOG

CAT

DOG

Prof. Leal-Taixe and Prof. Niessner
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
Autoencoders

• Why do we need this dimensionality reduction?

• To capture the patterns, the most meaningful factors of variation in our data

• Other dimensionality reduction methods?
Autoencoder: training

Input Image

Conv

Transpose Conv

Output Image

Reconstruction Loss (like L1, L2)
Autoencoder: training

Input $x$ → encoder → bottleneck layer → decoder → Reconstruction $x'$

Latent space $z$: $\text{dim} (z) < \text{dim} (x)$

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: Unsupervised training with autoencoders
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

![Diagram of an autoencoder](image)

- Input $x$
- Encoder
- Bottleneck layer $z$
- Output $y$
- Ground truth labels $y^*$
- Loss
- Backprop as always
Why using autoencoders?

• Use 1: pre-training, as mentioned before
  – Image $\rightarrow$ same image reconstructed
  – Use the encoder as "feature extractor"

• Use 2: Use them to get pixel-wise predictions
  – Image $\rightarrow$ semantic segmentation
  – Low-resolution image $\rightarrow$ High-resolution image
  – Image $\rightarrow$ 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 Segmentation (FCN)
SegNet

SegNet

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

- **Decoder**: Upsampling + convolutional filters

SegNet

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

- **Decoder**: Upsampling + convolutional filters

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

Recall transposed convolution

- Transposed convolution
  - Unpooling
  - Convolution filter (learned)
  - Also called up-convolution (never deconvolution)
SegNet

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

• **Decoder**: Upsampling + convolutional filters

• **Softmax** layer: The output of the soft-max classifier is a K channel image of probabilities where K is the number of classes.

Upsampling
Types of upsampling

• 1. Interpolation
Types of upsampleings

1. Interpolation

- Nearest neighbor interpolation
- Bilinear interpolation
- Bicubic interpolation
Types of upsamplings

- 1. Interpolation
  
  Few artifacts
Types of upsamplings

• 2. Fixed unpooling

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

Skip Connections

- U-Net: zoom in

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

Prof. Leal-Taixé and Prof. Niessner
SegNet

Monocular depth

- Unsupervised monocular depth estimation

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]).

Image super resolution

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

Image synthesis

• Semantic segmentation image ➔ Real image

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

• Semantic segmentation image $\rightarrow$ 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

Prof. Leal-Taixé and Prof. Niessner
Perceptual loss and style transfer
Content loss

• Content loss (or perceptual loss or feature reconstruction loss).

• Use a network to compute the loss

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

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

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

Feature maps of the generated image at layer j
Feature maps of the ground truth image at layer j
Feature map size (channels, height, width)
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]

Style Transfer

• Content loss: feature representation similarity

• Style loss:

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

• Comparing Gram matrices

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

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

\[ G^\phi_j(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'} \]

- Comparing channels \( c \) and \( c' \)
Style loss

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

\[ G_{j}^{(x)}_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'} \]

• In practice: vectorize the feature maps to the size of \( C \times (HW) \)

• 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
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!
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)
Next lecture

- Next lecture on Monday 3rd
- Make sure you are working on your projects!
- Group #2 presenting this Friday!