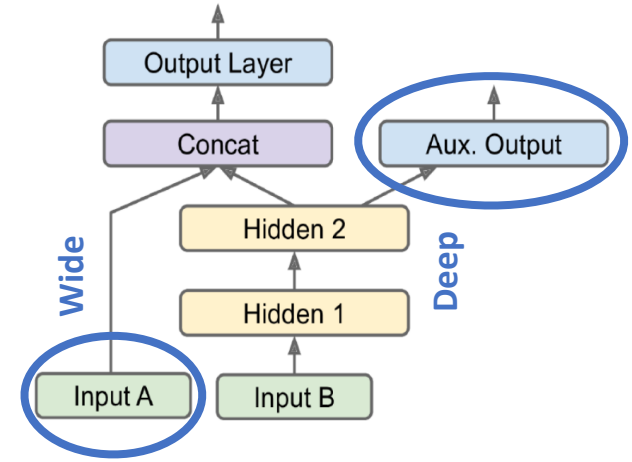
Network Architectures 2, Representation Learning and GANs

Using Machine Learning Tools

Reading: Chapter 14 & 17

Last Time …



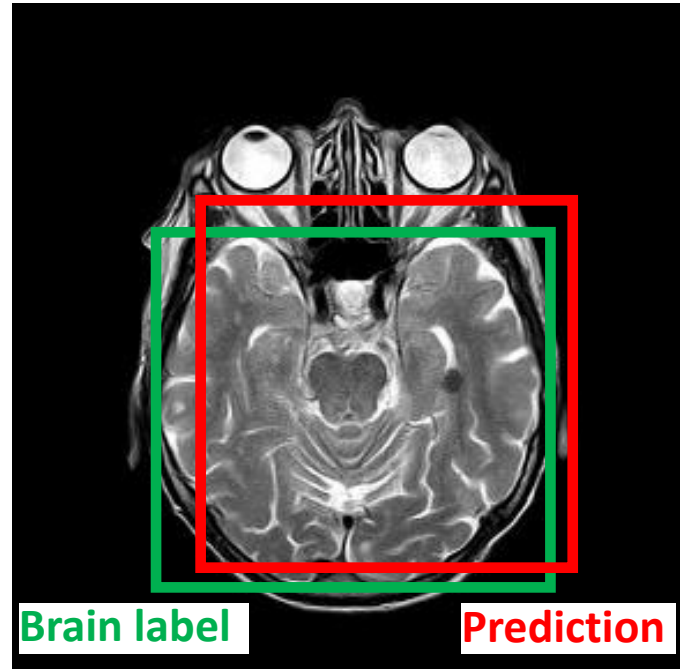
Network Architectures:

- Branching, joining, skipping over layers
- Key design elements:
  - Step sequence: feature extraction, downsampling, upsampling, merging
  - Depth: how many steps, how many layers per step, how many filters
  - Connectivity: which nodes are connected to which other nodes
  - Complexity: number of parameters, convergence, deployment size
- Lower network layers: more distributed, general, lower level features
  → more transferable to similar task
- Transfer learning: Reuse lower layers in new network as a good initialisation
- Many more sophisticated architectures exist, adapted to specific tasks

Image (modified): Géron, Hands On ML 2019

Localisation of Object

- Localisation task = find bounding box around object
- Bounding box (BB) = centre coordinates, width, height
- Add a branch at top end of NN above global average pooling with dense layer and 4 outputs



Training:

- Need location labels (bounding boxes)
- Normalise width and height of image to [0,1]
- Use sqrt of width and height to make large bounding boxes less sensitive to errors
- Cost function: Intersection over Union (IoU)

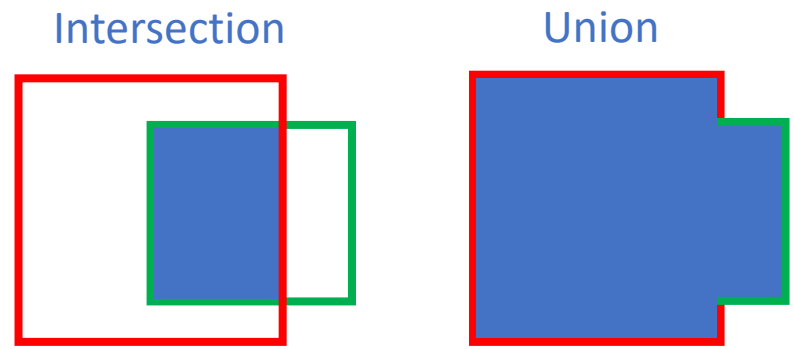


Image at top: Pixbay open license.

Detection of Objects

- Detection = Localise *and classify* multiple objects
- *Sliding window* approach:
  - Scan the image with varying window sizes
- Objectness: probability that in bounding box is an object (any class)
  - Sigmoid activation, binary cross entropy loss
- Output: bounding box, objectness, class probabilities
- Post-processing: non-maximum suppression
  - Exclude all BBs with objectness below threshold
  - Iteratively take BB with highest objectness and remove other BBs with large (>60%) overlap

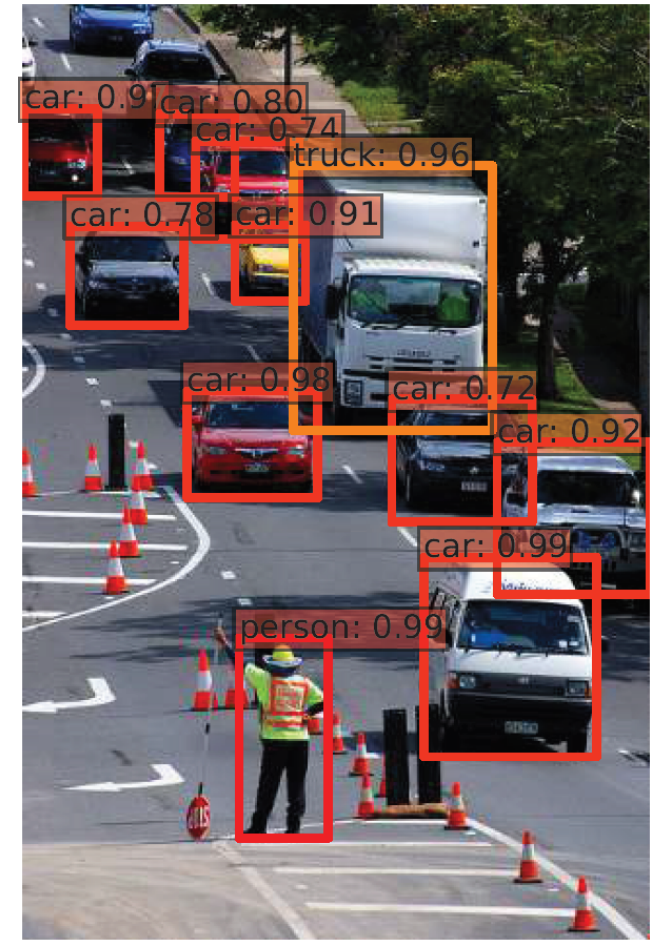
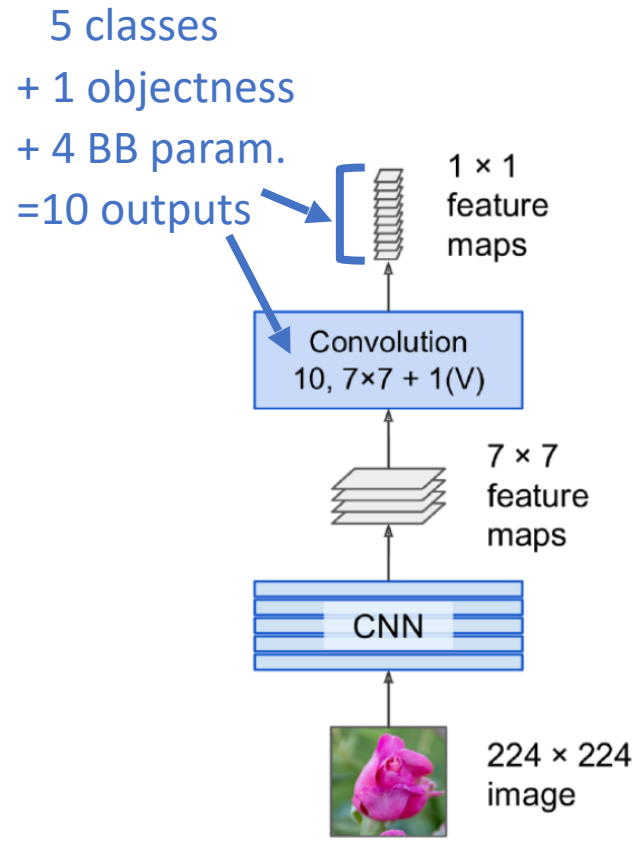


Image (modified): Liu et al. 2016, arXiv:1512.02325v5

Fully Convolutional Networks

- Idea: Use a sliding window to do processing in one shot via convolution
- Replace top dense layers of CNN with convolutional layers (no. units in dense layer = no. filters in conv layer)
- Output N feature maps, where N = class count + objectness + 4 BB parameters
- Use padding “valid” to get 1x1 output for image with size = desired footprint

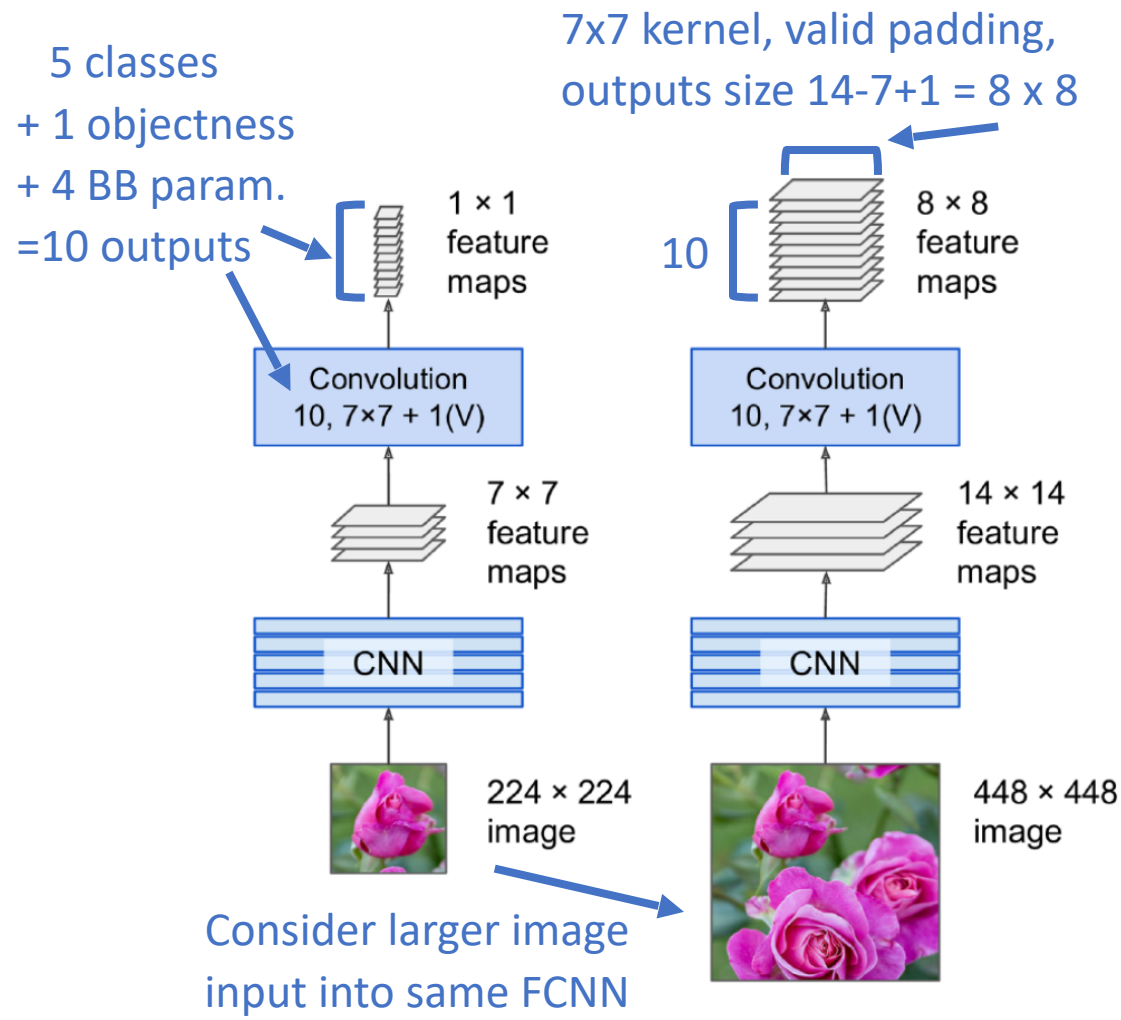


Long et al. 2015 Fully Convolutional Networks for Semantic Segmentation,” Proc. CVPR, p 3431– 3440.

Images (modified): Géron, Hands On ML 2019

Fully Convolutional Networks

- Idea: Use a sliding window to do processing in one shot via convolution
- Replace top dense layers of CNN with convolutional layers (no. units in dense layer = no. filters in conv layer)
- Output N feature maps, where N = class count + objectness + 4 BB parameters
- Use padding “valid” to get 1x1 output for image with size = desired footprint
- Output size scales up as larger input images are used
- Non-maximum suppression at end

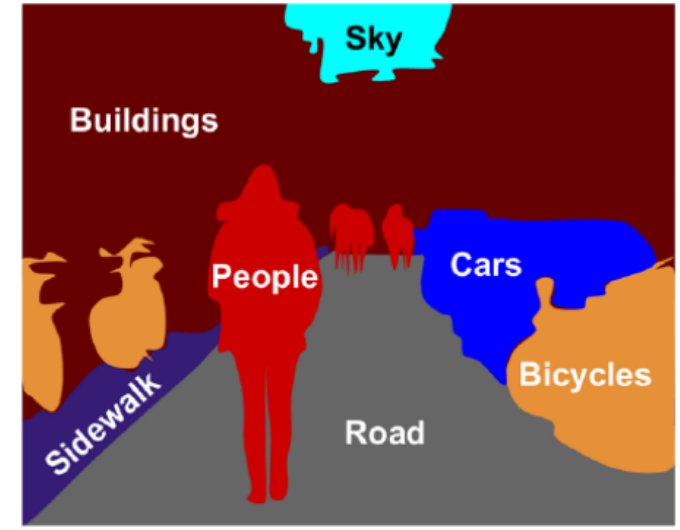


Long et al. 2015 Fully Convolutional Networks for Semantic Segmentation,” Proc. CVPR, p 3431– 3440.

Images (modified): Géron, Hands On ML 2019

Semantic Segmentation

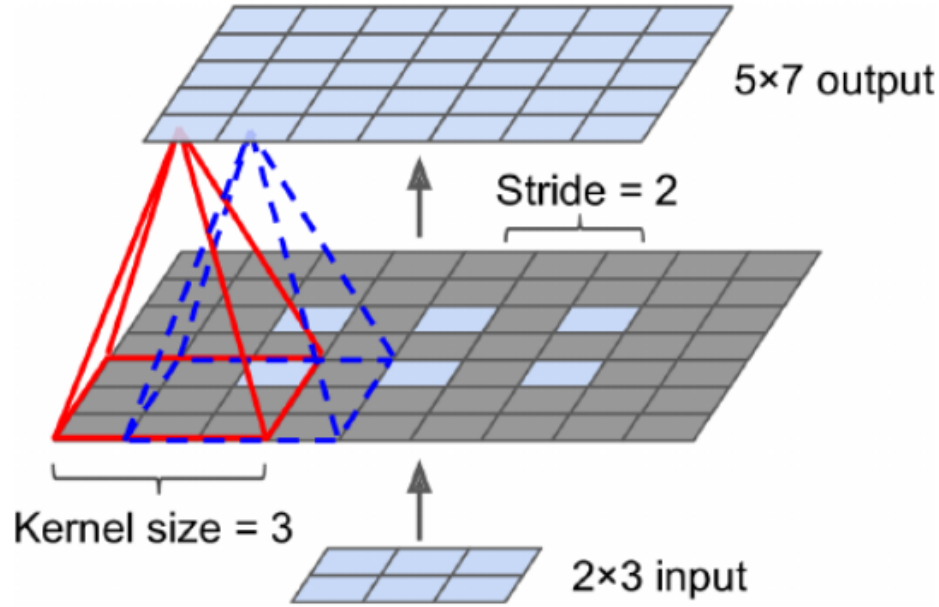
- Task: Classify *each pixel* into one object class
- Boundaries between multiple objects of same class not differentiated
- Multiscale approach in fully convolutional network looses fine detail of larger objects due to downsampling
- Approach:
  - *Upsample* output of smaller layers
  - *Combine* (sum or concatenate) with preceding higher resolution output
  - Do this stepwise/cascaded
- Many different architectures exist

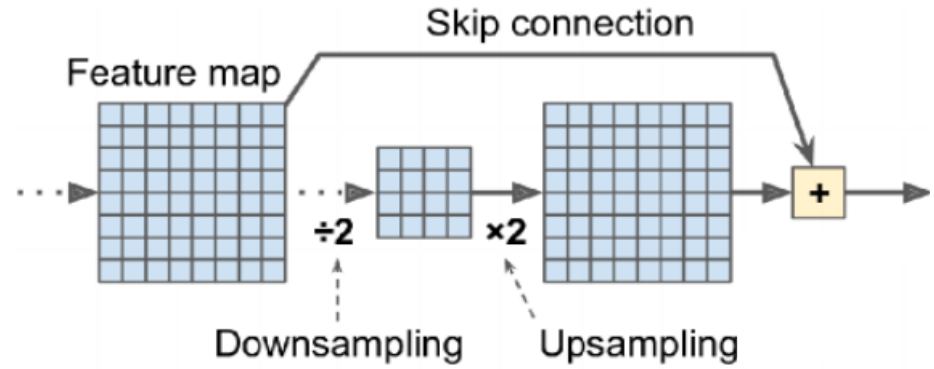


Images (modified): Géron, Hands On ML 2019

Upsampling

- Can use interpolation
  - Fixed algorithm, with no parameters
  - Is very rigid and introduces smoothness
- Transposed convolution layer
  - Pad with zeros between input values
  - Train kernel (with usual, adjustable weights) to perform upsampling
  - Keras.layers.Conv2DTranspose
- Add/concatenate high resolution feature map (of same size) from a previous layer to recover spatial resolution





Images (modified): Géron, Hands On ML 2019

Segmentation: U-Net

- "U-Net" because of "U"-shaped architecture

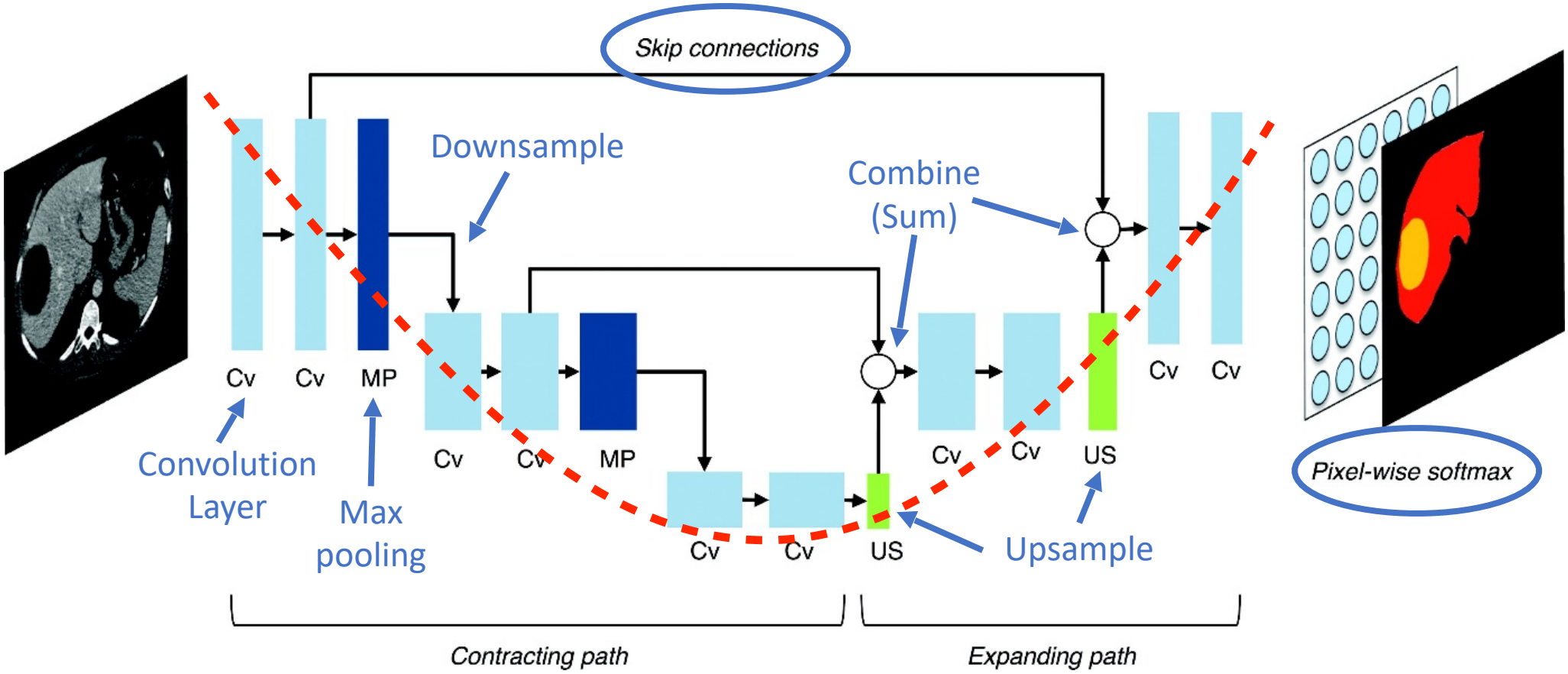


Image (modified): Chartrand et al. RadioGraphics 2017

Segmentation: U-Net

- "U-Net" because of "U"-shaped architecture

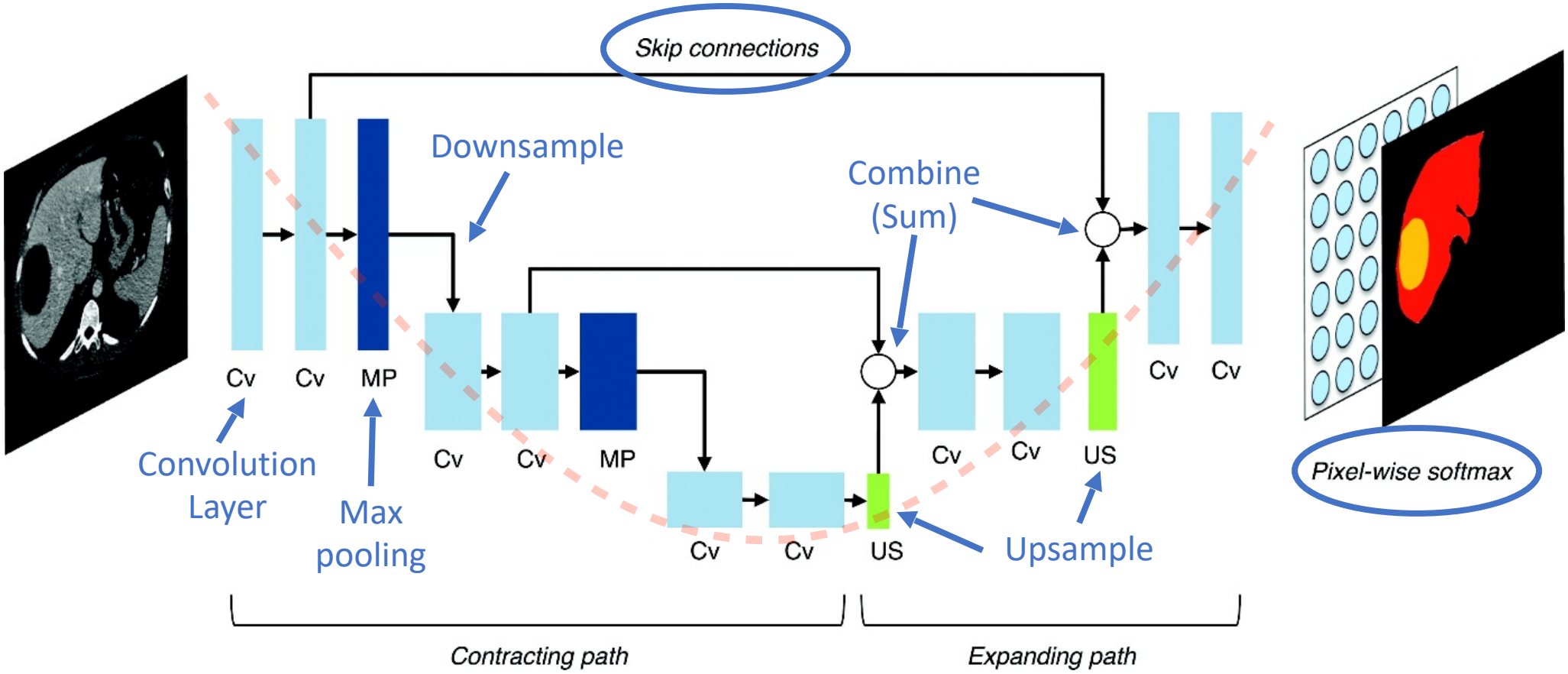


Image (modified): Chartrand et al. RadioGraphics 2017

Classification, Detection, Segmentation

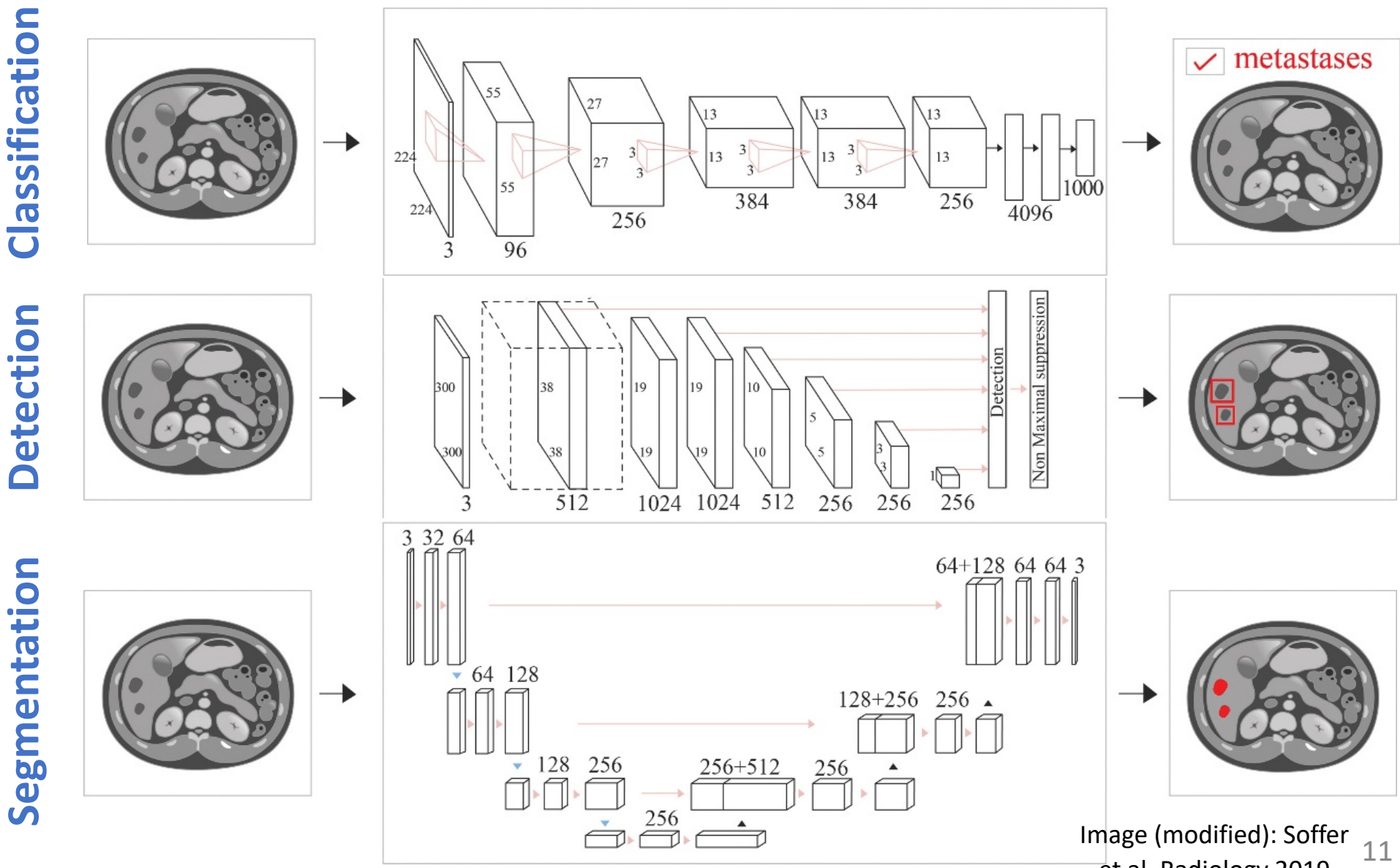


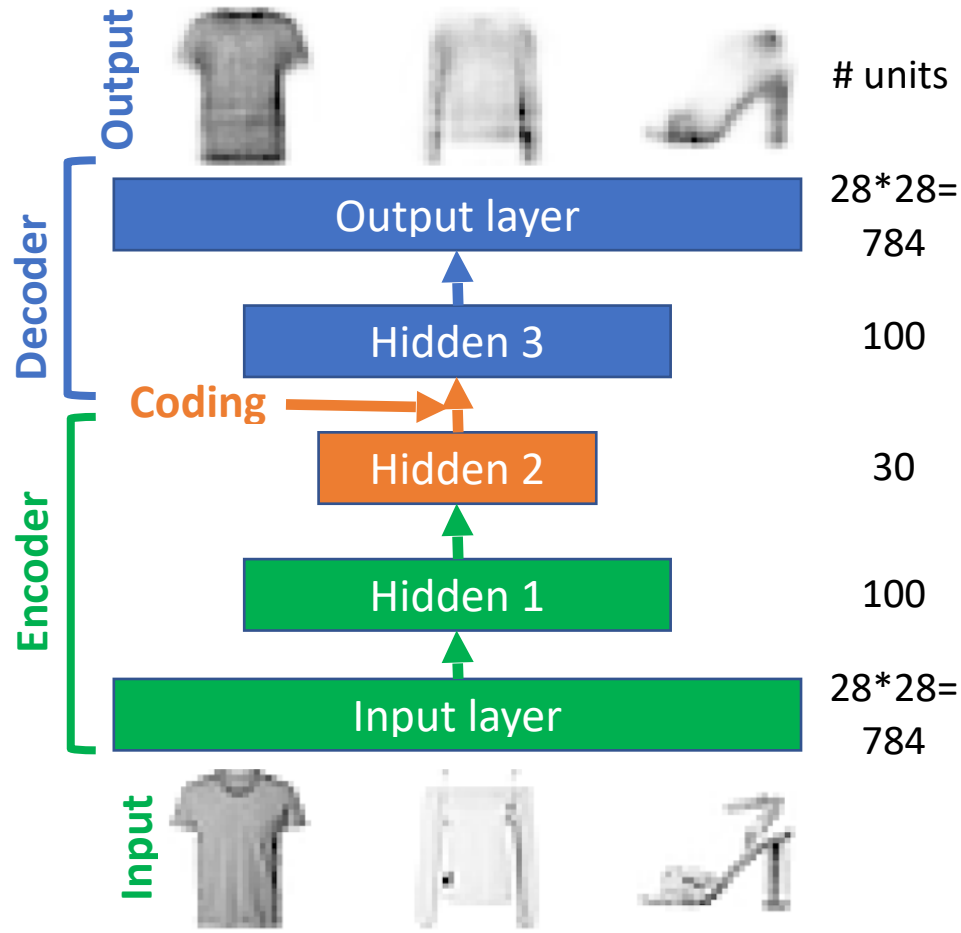
Image (modified): Soffer et al. Radiology 2019

Part 2

Representation Learning and Generative Learning

Autoencoder

- NN that learns small/dense representation of the data without supervision
- Dense representation = “coding” = “latent representation”
- Create a model that can replicate the input data (output = input) but via a bottleneck/ constraint (far fewer parameters to prevent overfitting / simplistic replication)
- Purposes:
  - Dimensionality reduction
  - Feature detection
  - Unsupervised pre-training of networks
  - Denoising
  - Generation/simulation of new data that is similar to the original data



Dimensionality Reduction with Autoencoder

- Create separate encoder and decoder:

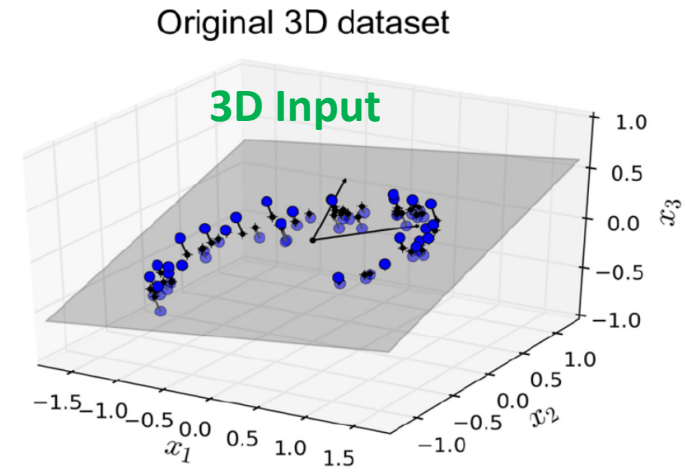
```
encoder = keras.models.Sequential([
    keras.layers.Dense(2, input_shape=[3])])
decoder = keras.models.Sequential([
    keras.layers.Dense(3, input_shape=[2])])
```

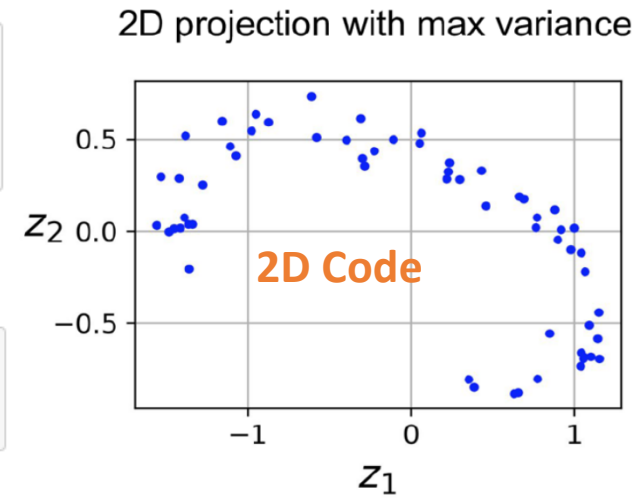
- Combine both:

```
autoencoder = keras.models.Sequential([encoder, decoder])
autoencoder.compile(loss="mse",
                    optimizer=keras.optimizers.Adam())
```

- Train with input=output, extract coding

```
history = autoencoder.fit(X_train, X_train, epochs=100)
codings = encoder.predict(X_train)
```





Data images (modified):
Géron, Hands On ML 2019

Convolutional Autoencoders

```
encoder = keras.models.Sequential([
    keras.layers.Reshape([28, 28, 1], input_shape=[28, 28]),
    keras.layers.Conv2D(16, kernel_size=3,
                        padding="same", activation="selu"),
    keras.layers.MaxPool2D(pool_size=2, padding="valid"),
    keras.layers.Conv2D(32, kernel_size=3,
                        padding="same", activation="selu"),
    keras.layers.MaxPool2D(pool_size=2, padding="valid"),
    keras.layers.Conv2D(64, kernel_size=3,
                        padding="same", activation="selu"),
    keras.layers.MaxPool2D(pool_size=2, padding="valid") ])

decoder = keras.models.Sequential([
    keras.layers.Conv2DTranspose(32, kernel_size=3,
            strides=2, padding="valid", activation="selu",
            input_shape=[3, 3, 64]),
    keras.layers.Conv2DTranspose(16, kernel_size=3,
            strides=2, padding="same", activation="selu"),
    keras.layers.Conv2DTranspose(1, kernel_size=3,
            strides=2, padding="same", activation="sigmoid"),
    keras.layers.Reshape([28, 28]) ])

autoencoder = keras.models.Sequential([encoder, decoder])
```

Input layer

Conv1: 28x28, 16 feature maps

Conv2: 14x14, 32 feature maps

Conv3: 7x7, 64 feature maps

Coding: 3x3, 64 feature maps

ConvT1: 7x7, 32 feature maps

ConvT2: 14x14, 16 feature maps

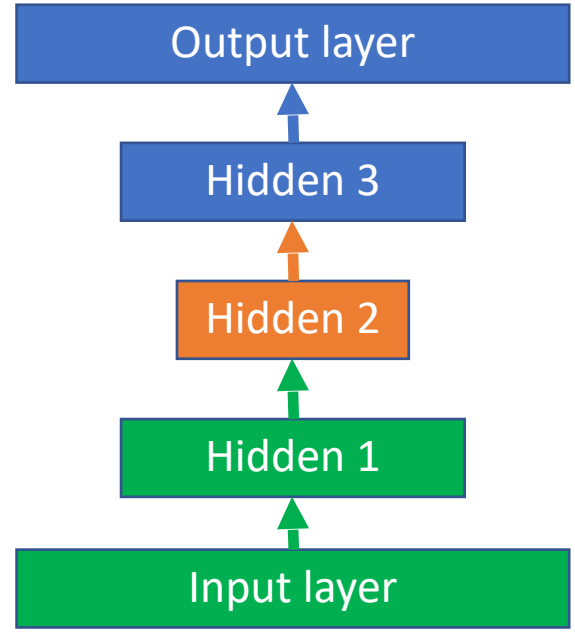
ConvT3: 28x28, 1 feature map

Output layer

Unsupervised Pre-Training

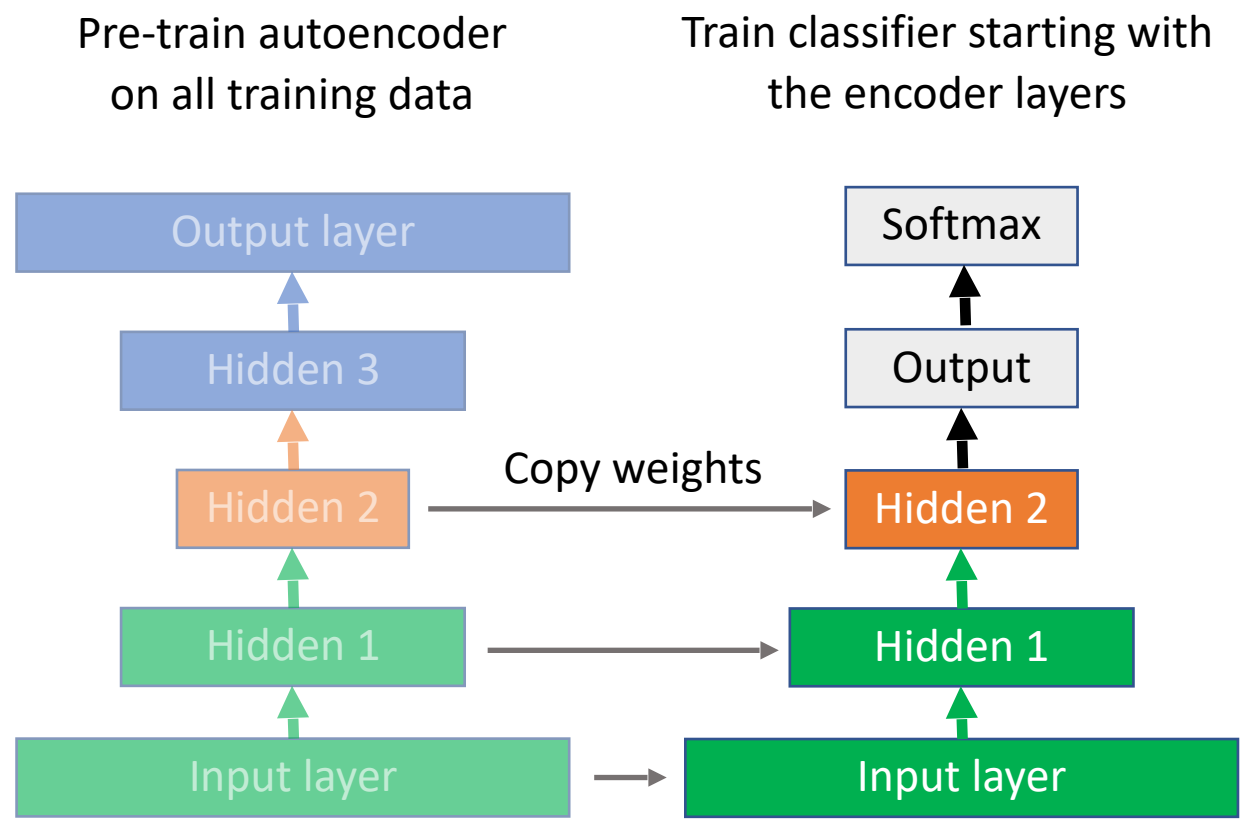
- Useful if data set is small or only a few labels are available
- Initialise classifier with layers that encode important/primary features of data

Pre-train autoencoder
on all training data



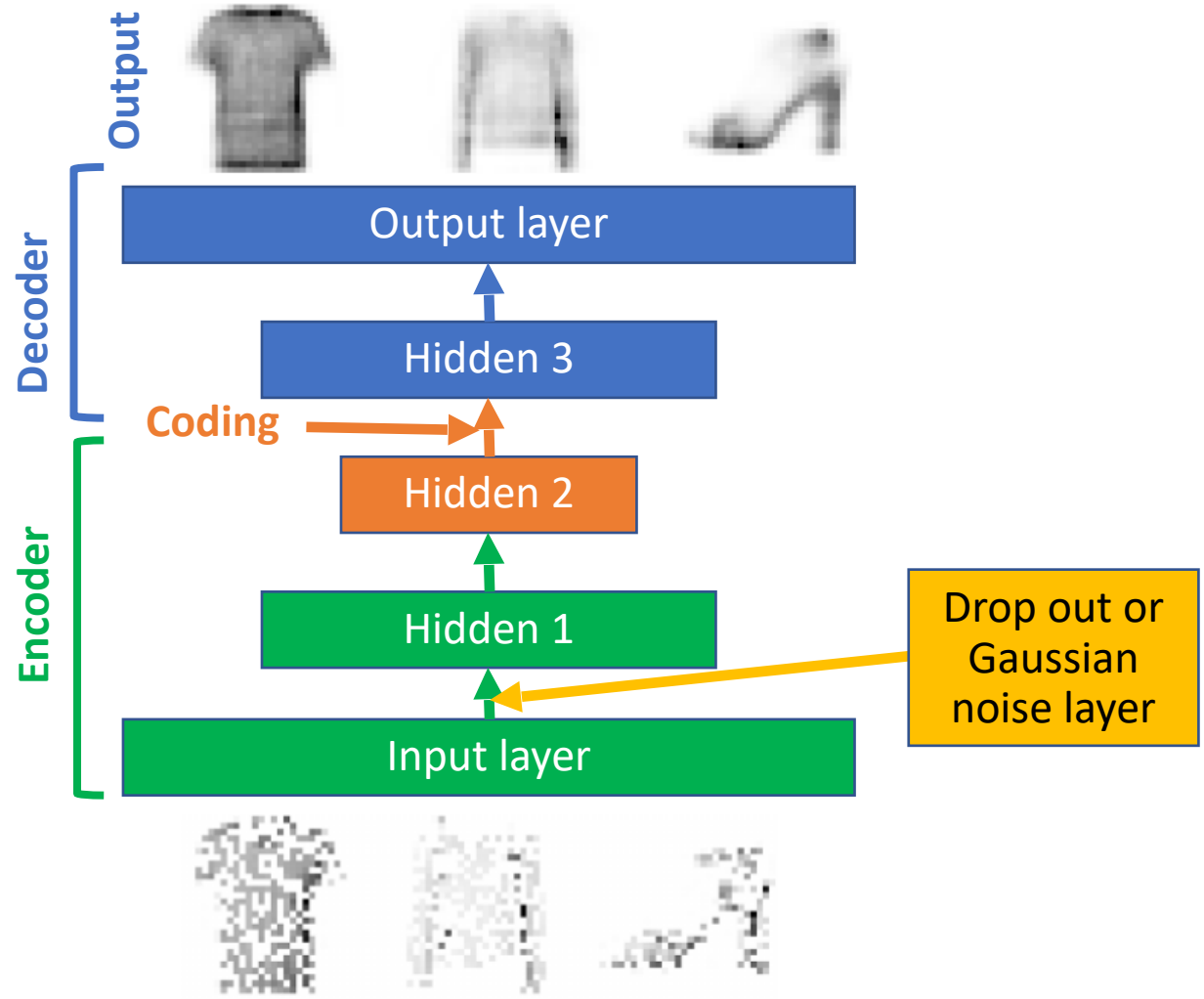
Unsupervised Pre-Training

- Useful if data set is small or only a few labels are available
- Initialise classifier with layers that encode important/primary features of data
- Classifier converges faster



Denoising Autoencoders

- Application: Reconstruct image from noisy input
- Noise can be random or even missing pixels
- Uses learned features to fill in the gaps/counteract noise
- Add noise to inputs during training



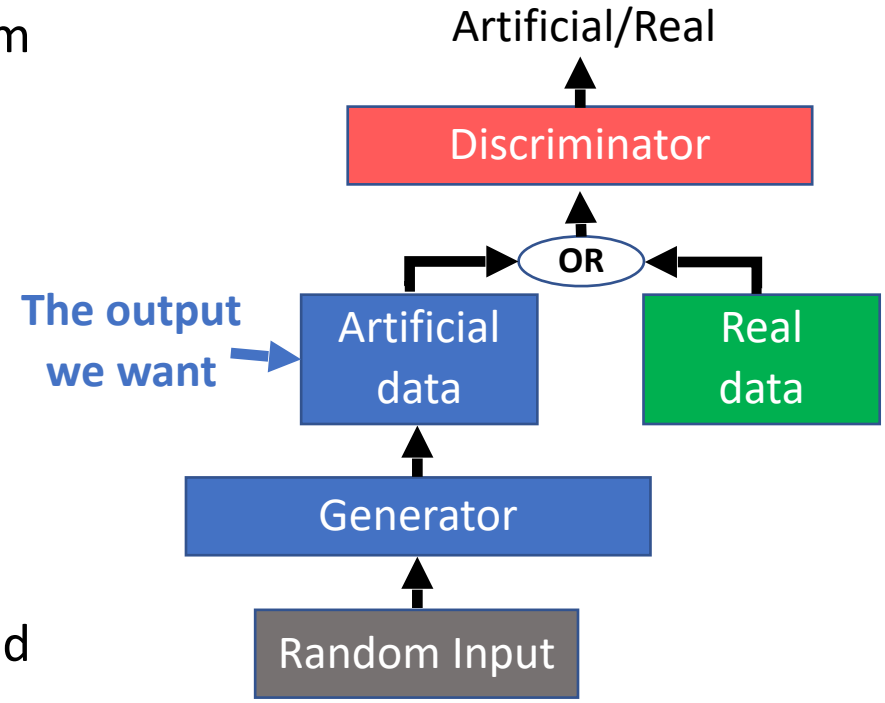
Data images (modified):
Géron, Hands On ML 2019

Generative Adversarial Network (GAN)

- GAN = generator + discriminator
- Generator network trained to generate data that is similar to the training data
- Discriminator trained to separate real data from artificial data

Purposes:

- Generate new outputs similar to data
- Increase resolution of image
- Image editing
- Augmentation of data sets
- Identify weaknesses of other models
- Gets around some difficulties in specifying good loss functions (e.g. to quantify how "realistic" an image is)



GAN Training

Training is more difficult that usual DNN
One strategy is to have in ***each iteration***:

Phase 1 (1 or more times)

- Take batch of 50% real and 50% artificial
- Labels 0=artificial, 1=real
- *Train discriminator only* using binary cross-entropy loss, with *frozen generator*

Phase 2

- Take batch of 100% artificial data,
- Set all labels to 1
- *Train generator only* using *frozen discriminator*

→ If discriminator can tell that an input is not real (0), then the loss function goes up

Problem: Both networks in a closed-loop causing oscillations and instability

Mode collapse:

- Generator becomes good at one class, forgets other classes
- Discriminator learns that class well, forgets other classes
- Generator becomes good at another class → repeat

GANs - Super Resolution

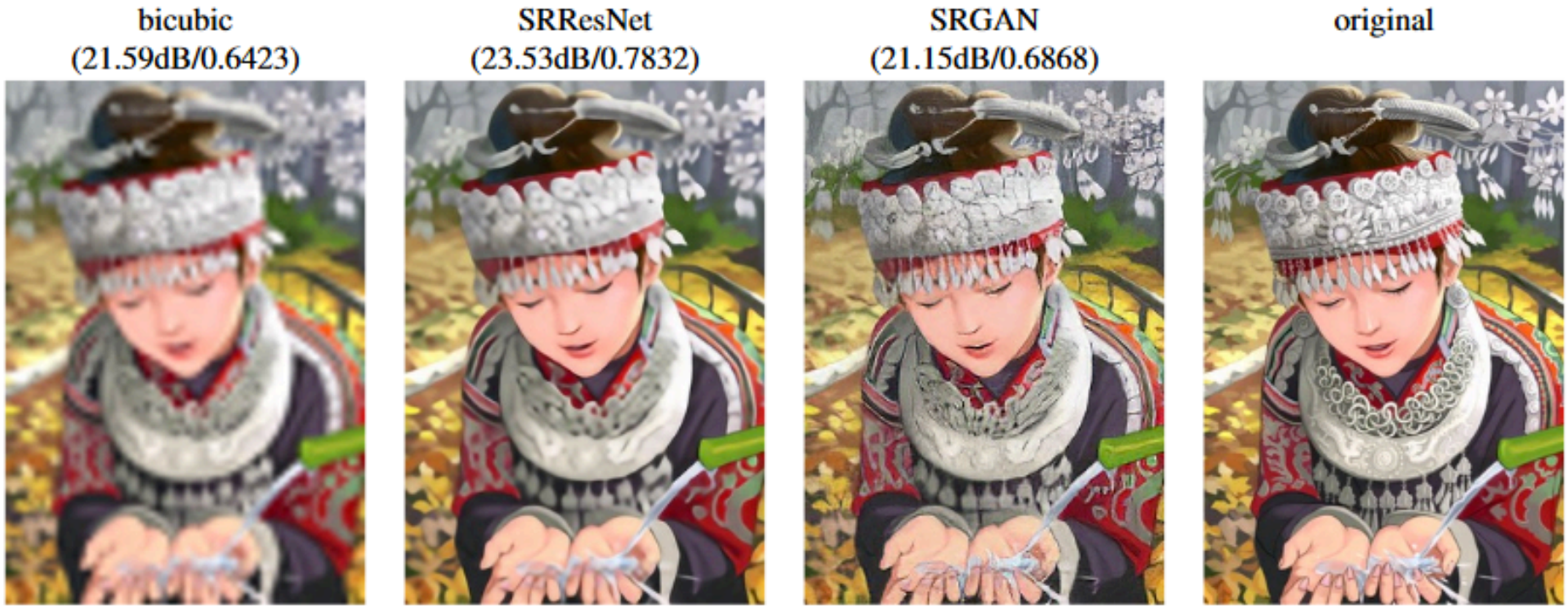


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Ledig et al., 2017 https://arxiv.org/pdf/1609.04802.pdf

GANs - Super Resolution

SRResNet

SRGAN

Original

Ledig et al., 2017 https://arxiv.org/pdf/1609.04802.pdf

Unsupervised Segmentation with GANs

- Can create artificial examples of healthy cases and use these to assist in anomaly detection

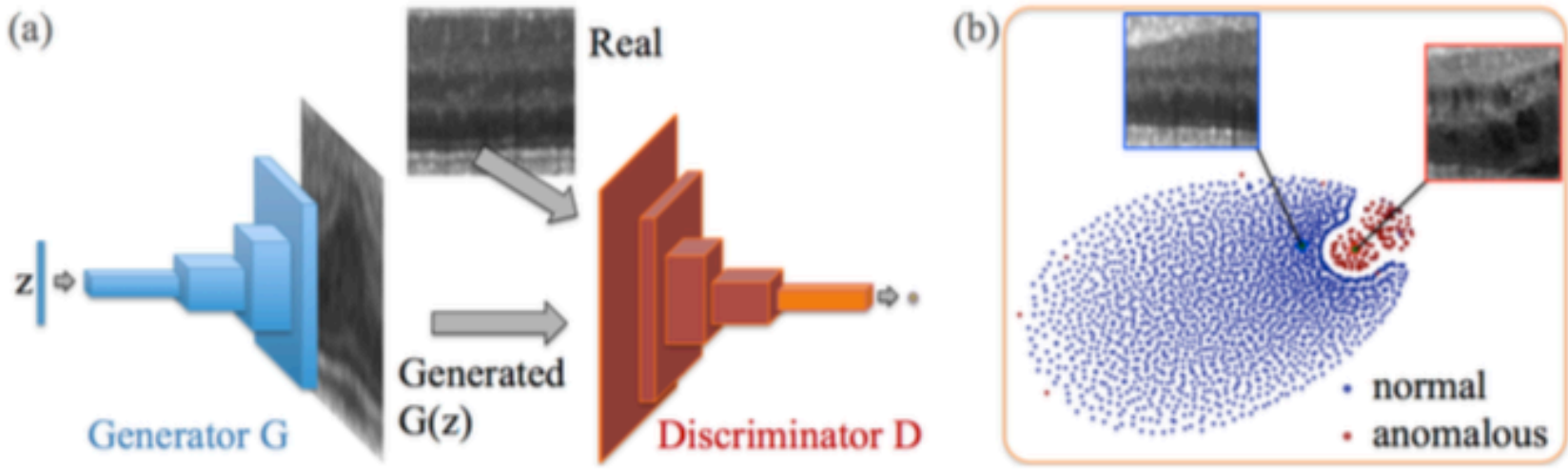


Fig. 2. (a) Deep convolutional generative adversarial network. (b) t-SNE embedding of normal (blue) and anomalous (red) images on the feature representation of the last convolution layer (orange in (a)) of the discriminator.

Schlegl et al., IPMI 2017

Summary

- Application-specific network architectures
  - Localisation: Bounding box around object → CNN
  - Detection: Localise & classify multiple objects → Multiscale FCNN
  - Segmentation: Assign class to every pixel → U-Net
- Autoencoder: Learn dense representation = “coding” of data
  - Unsupervised pre-training of networks
  - Coding for dim reduction, denoising, etc.
- GAN: Train generator network to produce artificial data that are “indistinguishable” from the real data
  - Useful for “unsupervised” methods / anomaly detection
- More architectures & variants exist and are being developed actively

Course Overview

Week	Topic
1	Introduction to machine learning
2	Machine learning workflow and terminology
3	Machine learning workflow, examples
4	Classification and Decision Trees
5	Training models, regularisation
6	Training models 2 and Support Vector Machines
7	Unsupervised learning, clustering, visualisation and dimensionality reduction
8	Deep neural networks
9	Convolutional neural networks
10	Training deep neural networks
11	Training deep neural networks 2 and network architectures
12	Network architectures 2, Autoencoders and GANs

Workflow & practical skills across tools

Traditional ML tools, typically for smaller inputs without ordered dimensions (table of feature values), model adds structure

Neural network tools, typically for inputs with ordered dimensions (time series, images, videos), model very flexible

Next Steps

- Practice, practice, practice!
  - Your Masters research/industry project
  - Chapters 12, 13 & 19 go into more practical detail
- Learn further (application specific) network architectures
  - Chapter 15 Recurrent NN and sequential data processing
  - Chapter 16 Recurrent NN and natural language processing
  - Chapter 18 Reinforcement Learning
- Connect directly with papers and current innovation:
  - Read landmark papers referenced in slides and in footnotes of book
  - Read conference proceedings of CVPR, ICCV, ECCV, NeurIPS, MICCAI
- Don't forget the basics
  - Look at your input and output data (not just summary performance values)
  - Look at your learning curves (and know what they mean and what action to take)
  - Make sure you are careful when creating training/validation/test datasets
  - Know how to interpret your outputs, use them for desired purposes, and show them correctly