CNN ARCHITECTURES

Paul Dragan

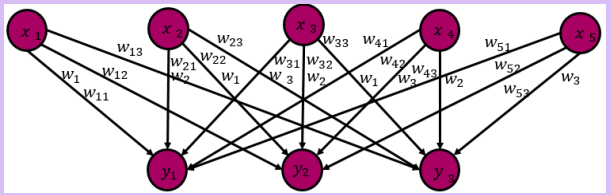
BOSCH

RECAP

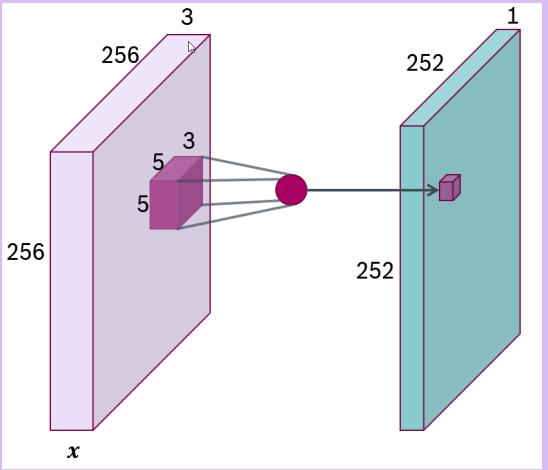
BOSCH

Recap
Basic layer types

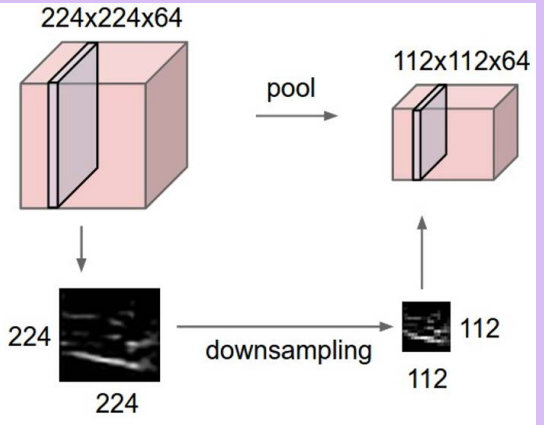
- Fully connected layer



- Convolution layer

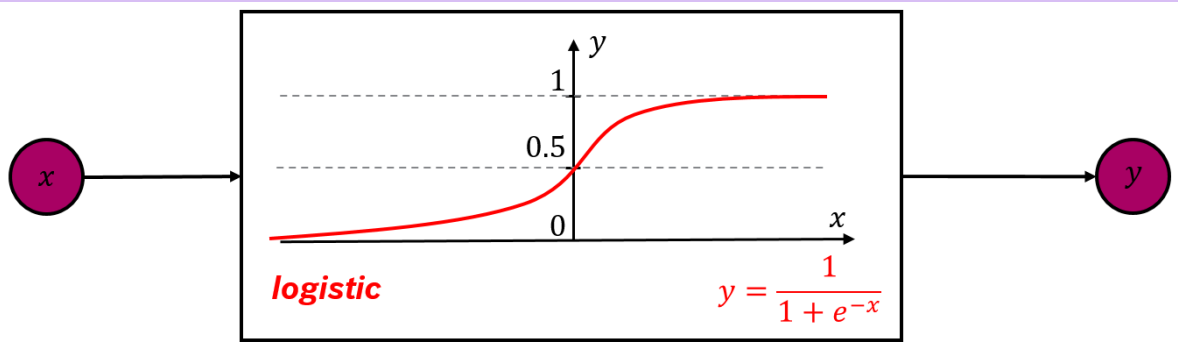


- Max-pooling layer

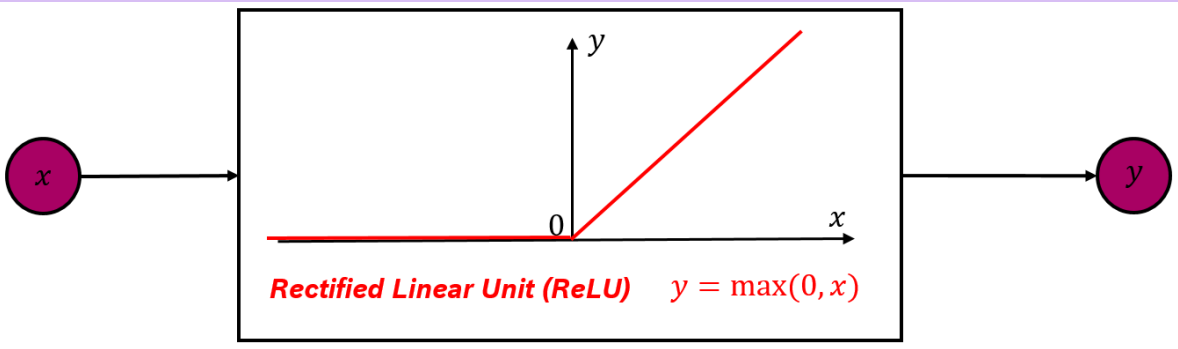


Recap
Non-linearities

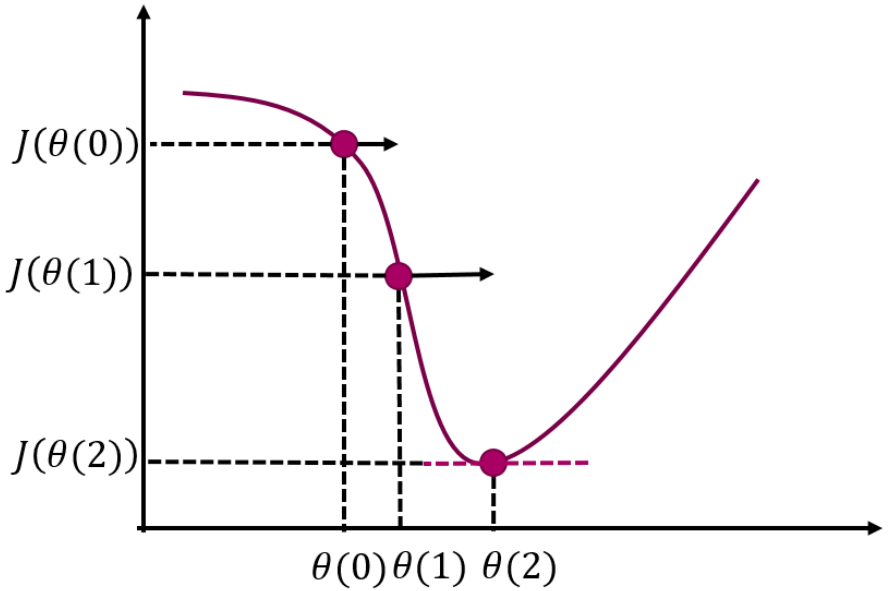
- Logistic (Sigmoid)

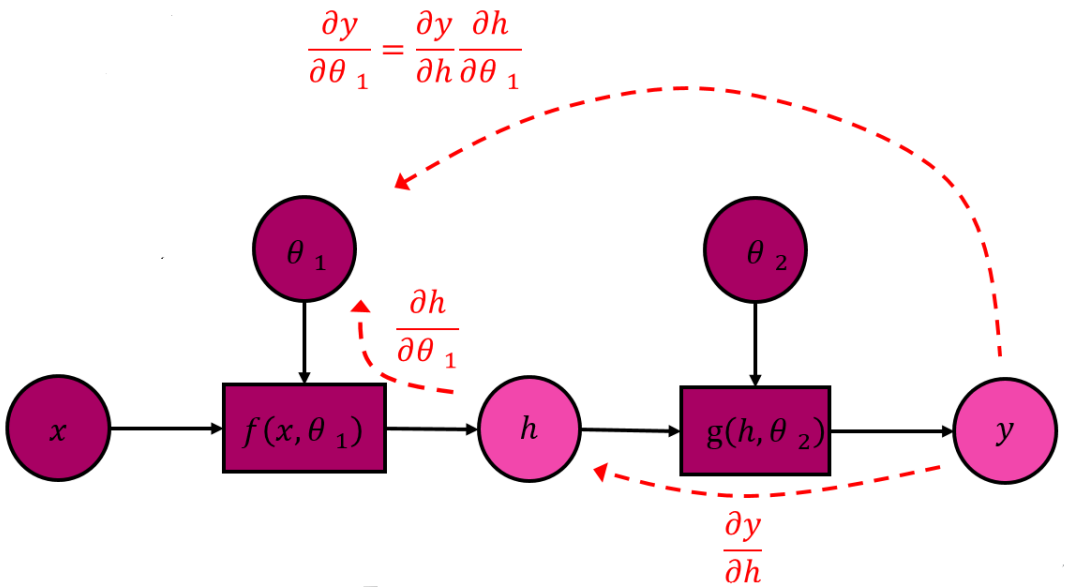


- ReLU (Rectified Linear Unit



Recap
Optimization, gradient descent, and backpropagation



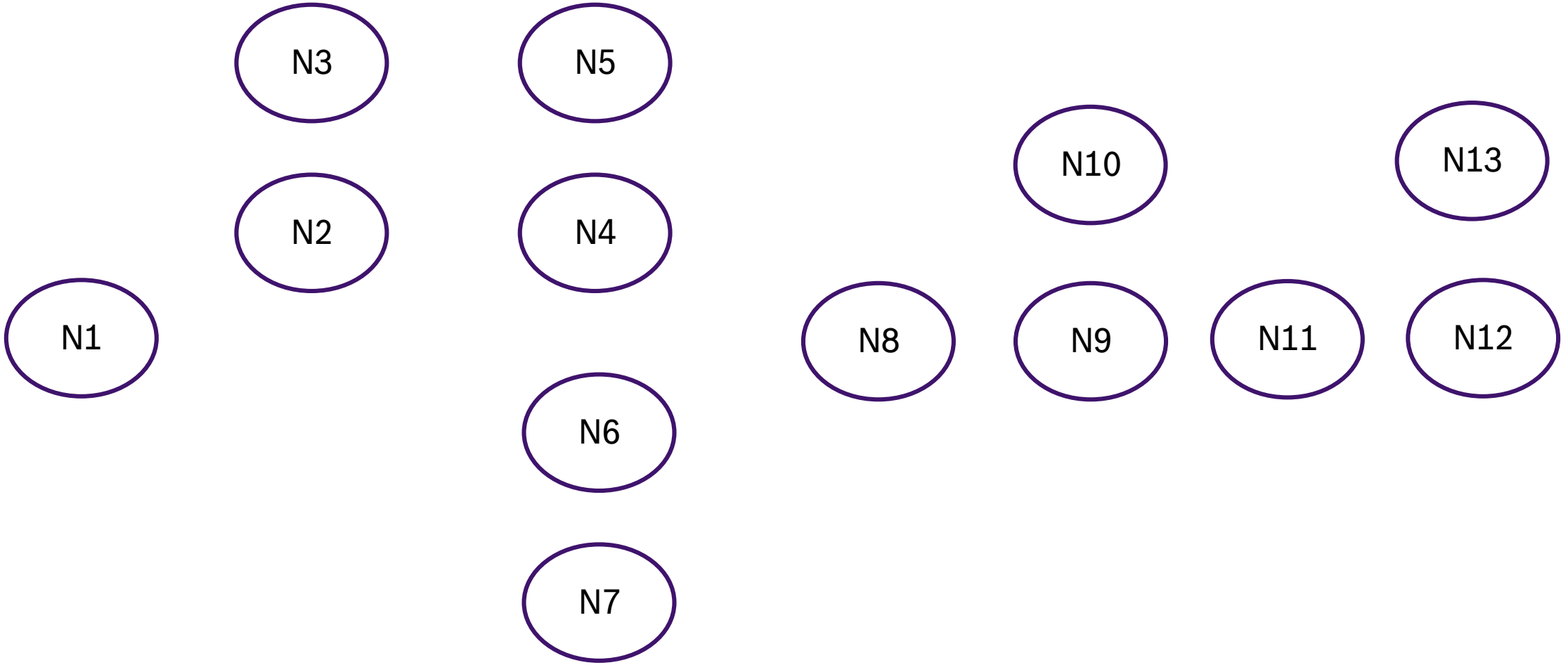


$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$J(\boldsymbol{\theta}) \cong J(\boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \nabla J(\boldsymbol{\theta}_0)$$

Recap

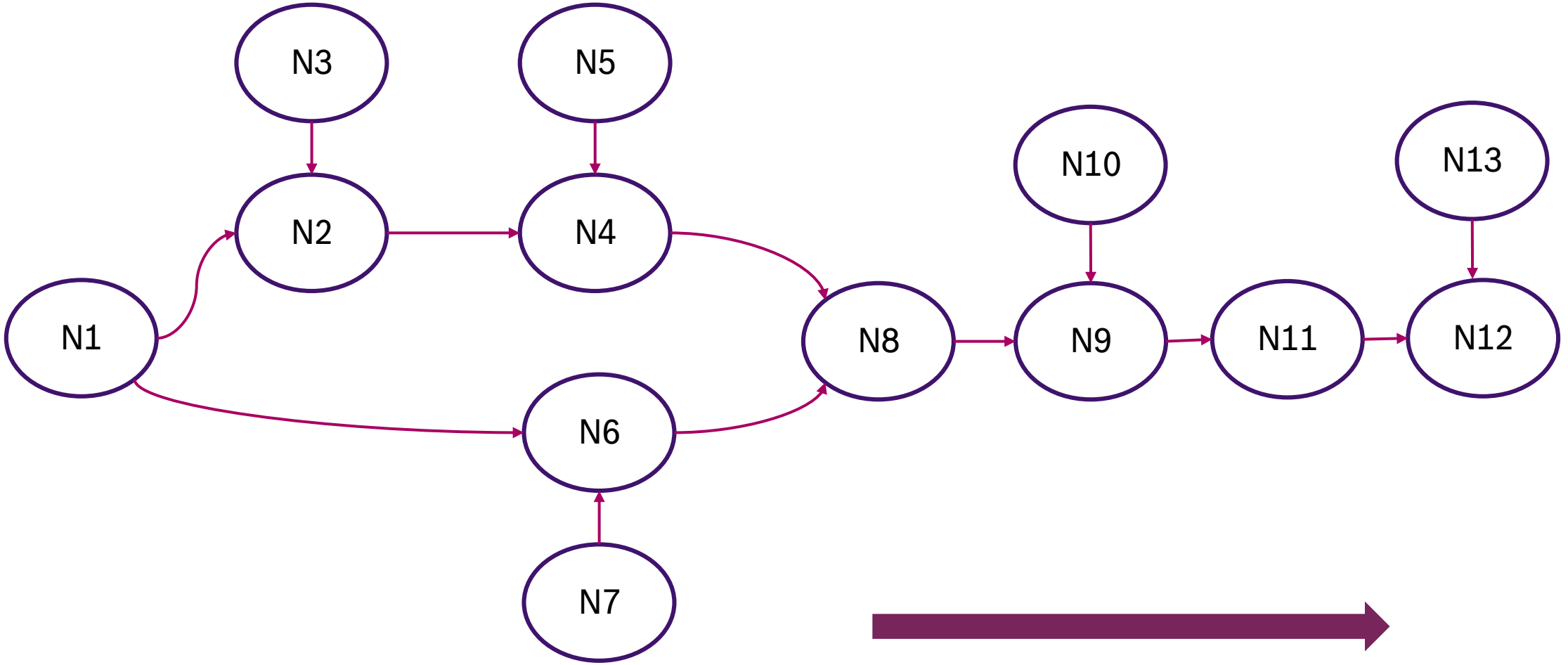
Deep learning as differentiable computational graphs



BOSCH

Recap

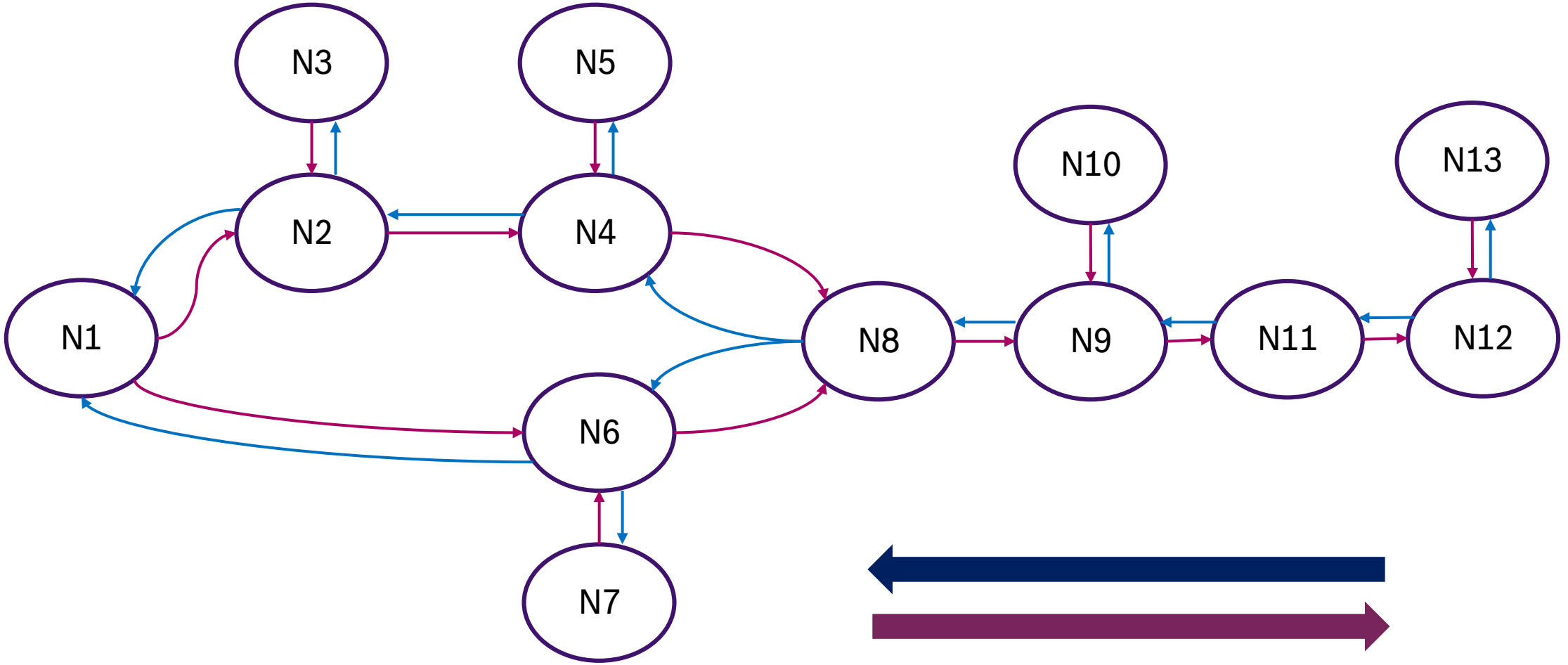
Deep learning as differentiable computational graphs



BOSCH

Recap

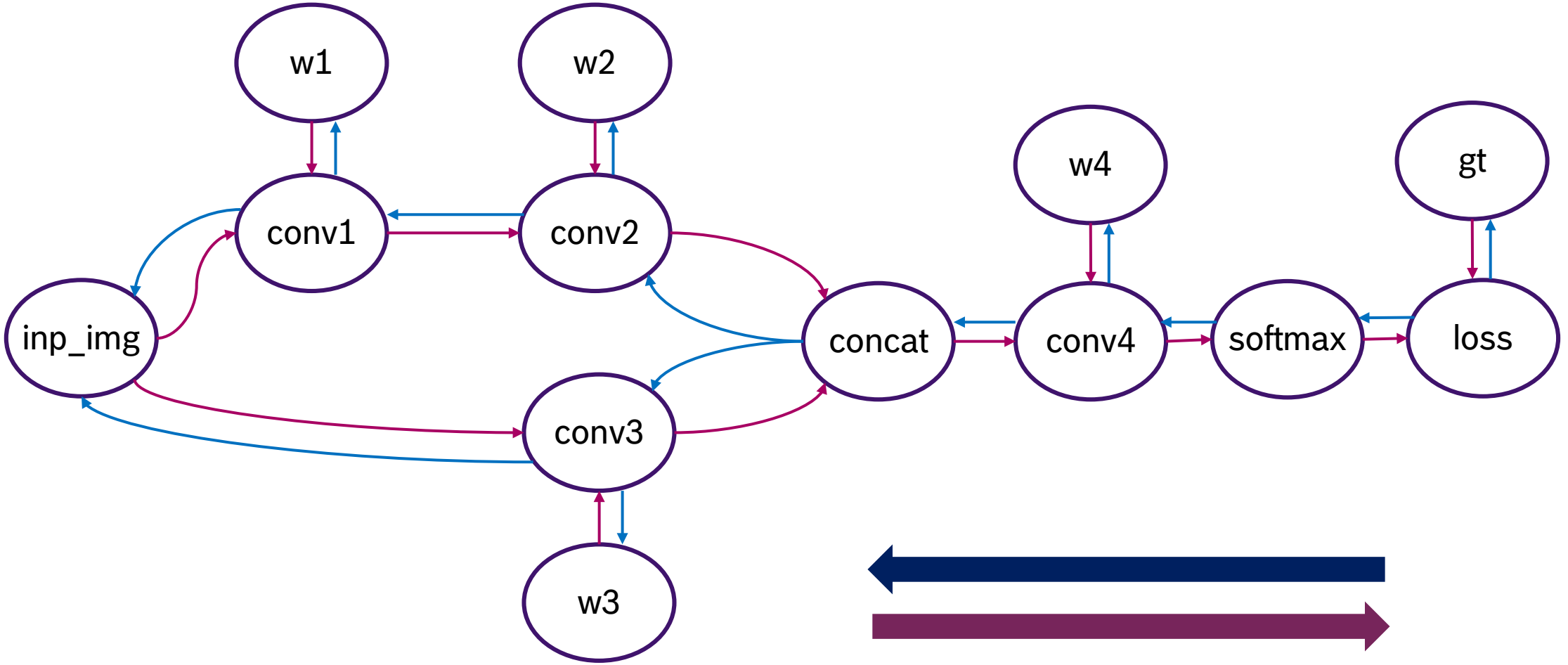
Deep learning as differentiable computational graphs



BOSCH

Recap

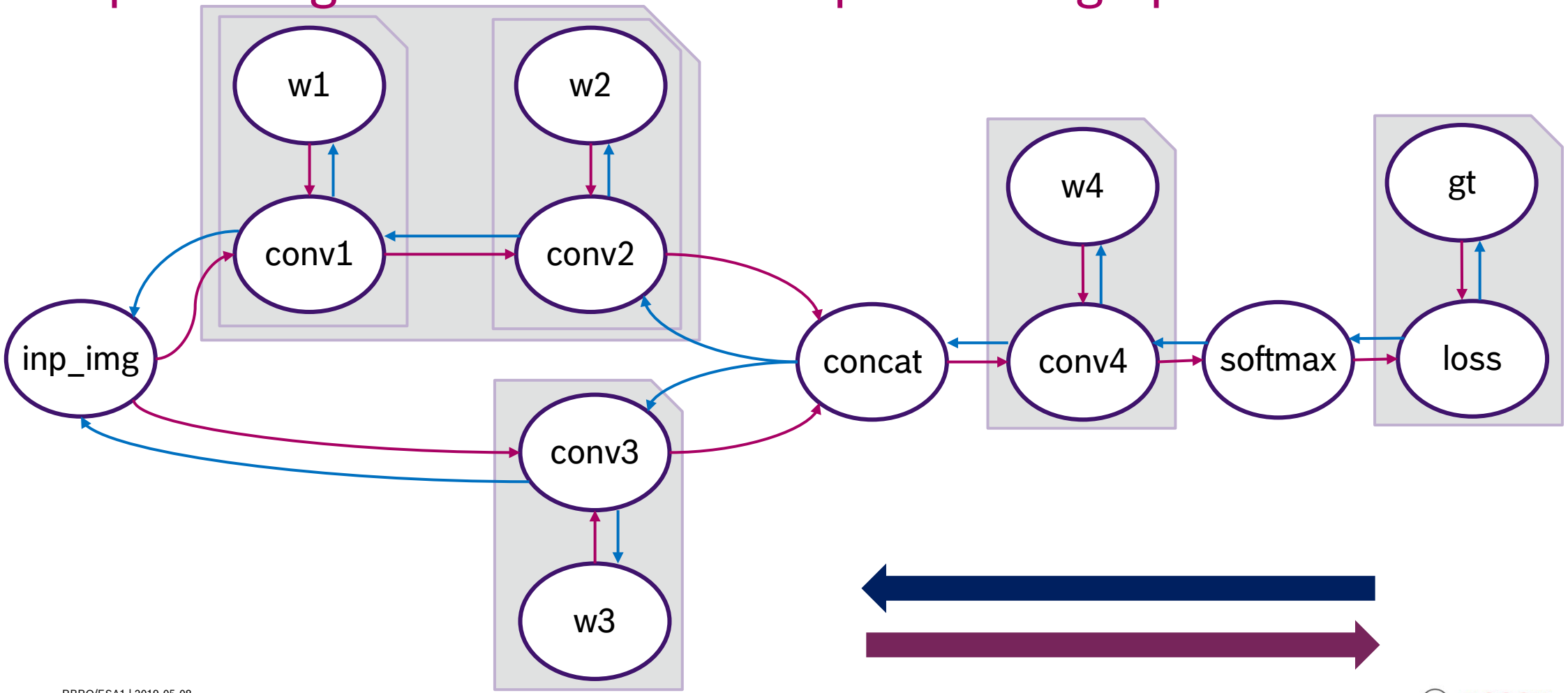
Deep learning as differentiable computational graphs



BOSCH

Recap

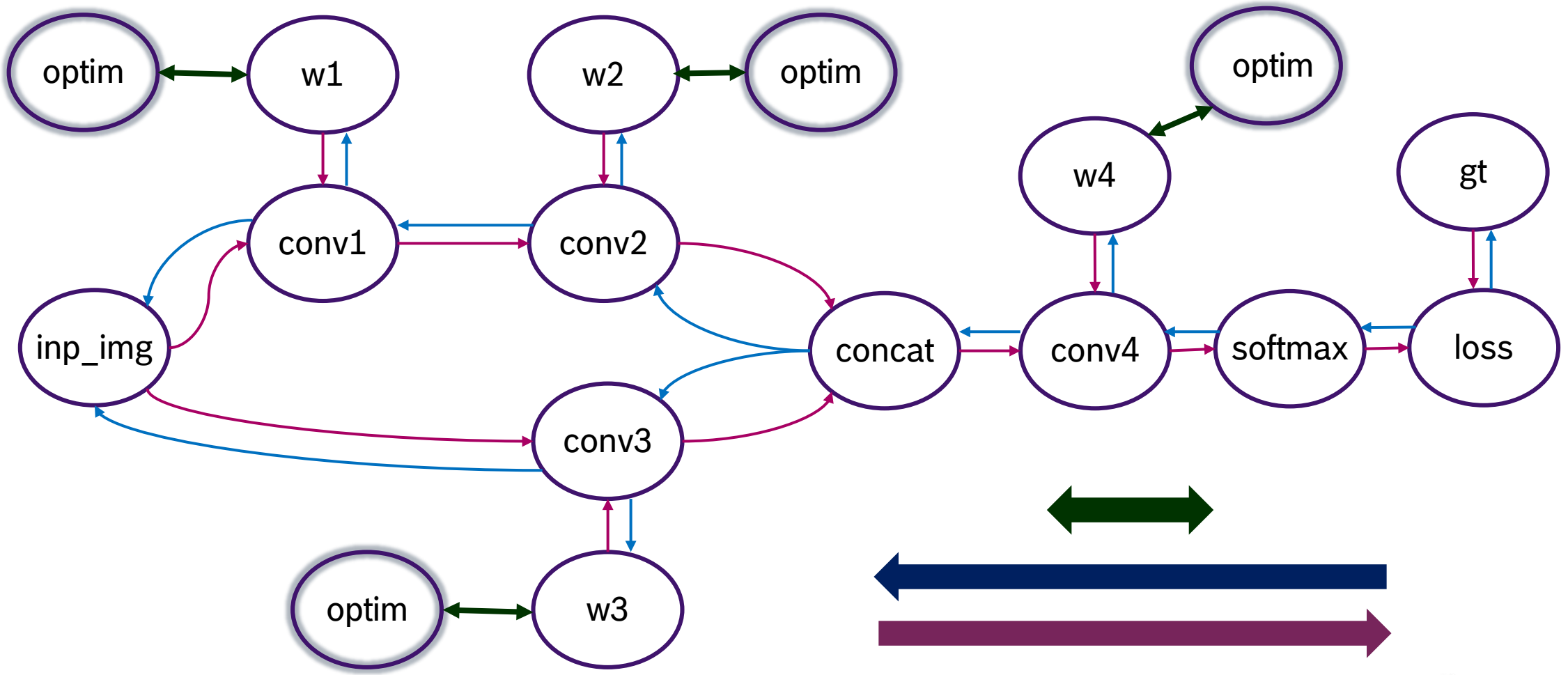
Deep learning as differentiable computational graphs



BOSCH

Recap

Deep learning as differentiable computational graphs

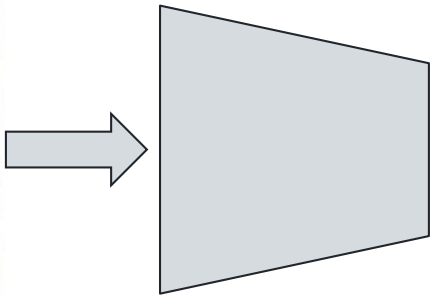


BOSCH

TASKS AND ARCHITECTURES

IMAGE CLASSIFICATION

Image classification
Task definition



CNN

cat	0.8
dog	0.1
house	0.01
car	0.05
plane	0.04

Most probable class (cat)

$$L(y) = -\log(p(y|\theta))$$

We can minimize:

- NLL
- Cross-entropy
- KL divergence

Image classification
Typical architectures – AlexNet & VGG

About

- AlexNet and VGG represent the *prototypical* image classification architectures
- They consist from a **feature extractor** and some **decision layers** on top of that
- The feature extractor gradually reduces the initial height / width dimensions, while increasing the level of abstraction

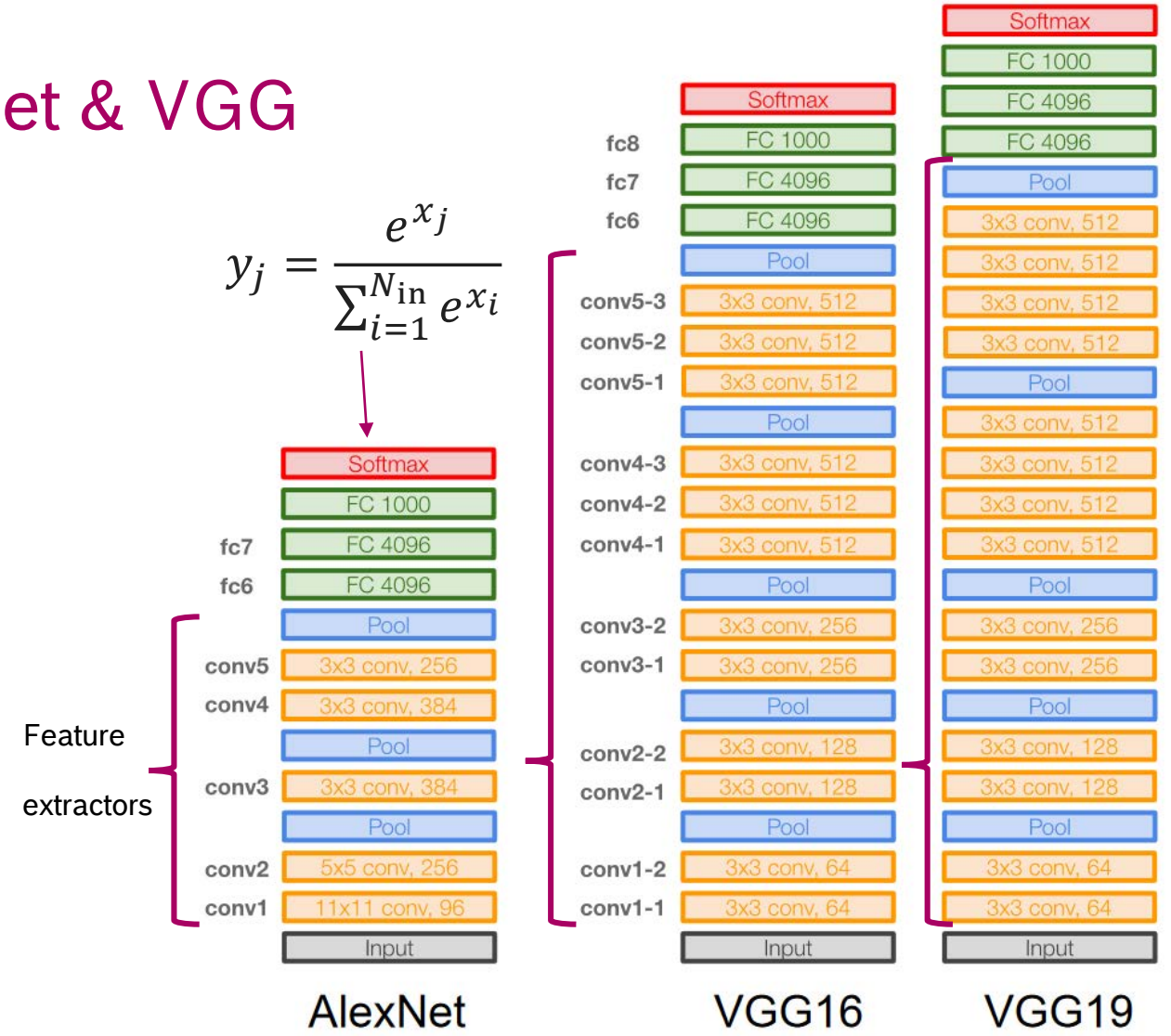


Image classification
Case study – AlexNet

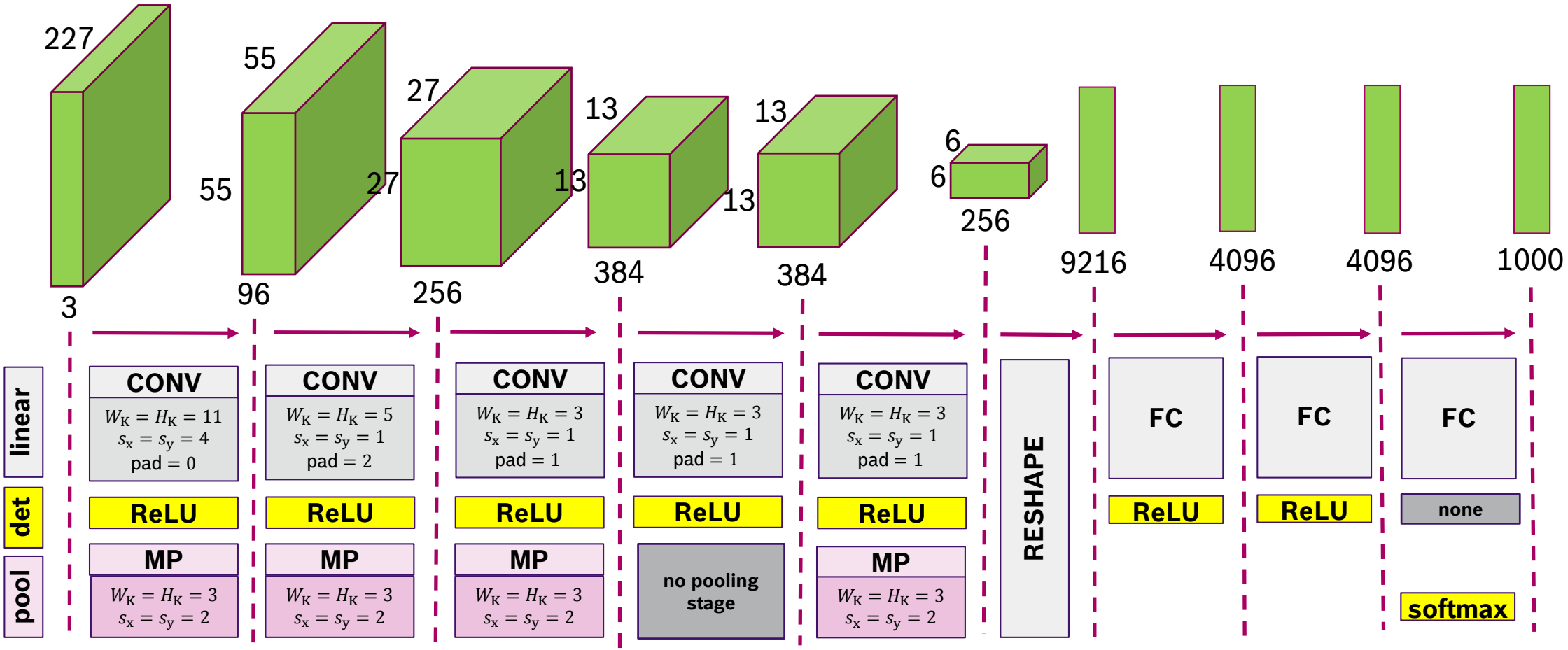


Image classification
Typical architectures – AlexNet & VGG

Issues

- Vanishing gradients -> lower layers are difficult to train
- Unfeasible thus to have too many levels

Possible solution

- Step-by-step training -> add a new level only after you trained the previous

$$y_j = \frac{e^{x_j}}{\sum_{i=1}^{N_{\text{in}}} e^{x_i}}$$

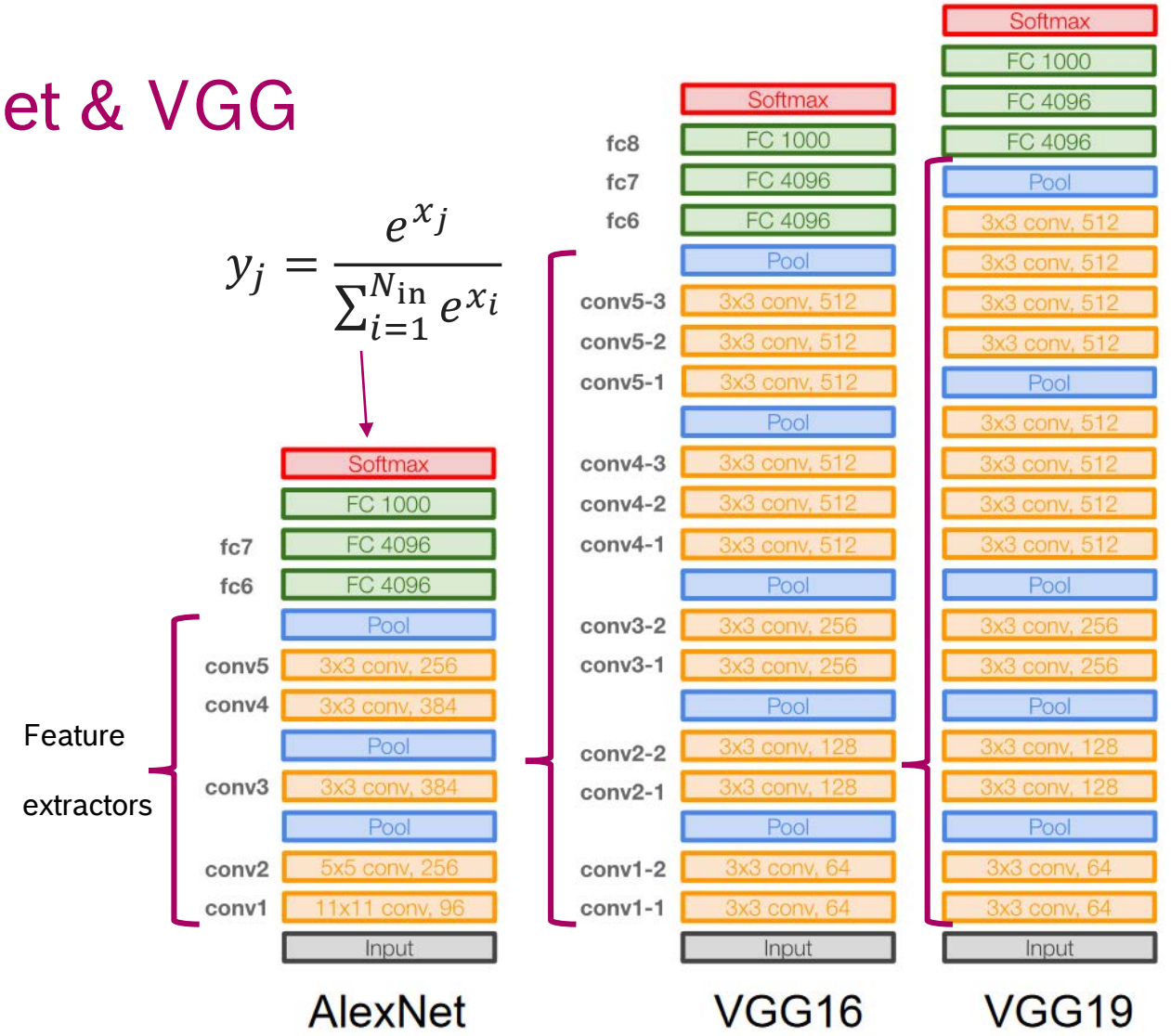
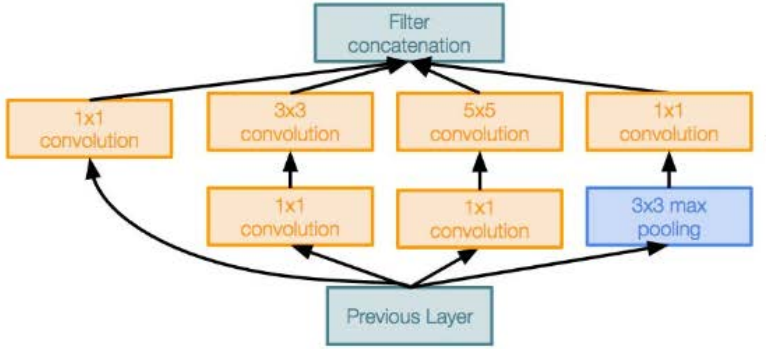


Image classification
Block based architectures – GoogLeNet

About

- GoogLeNet is a good example of a modularized CNN architecture
- Each module computes multiple convolution types
- The loss is computed at **multiple levels** for stronger gradients
- With careful output dimension management for each module, we can get quite deep models



Inception module

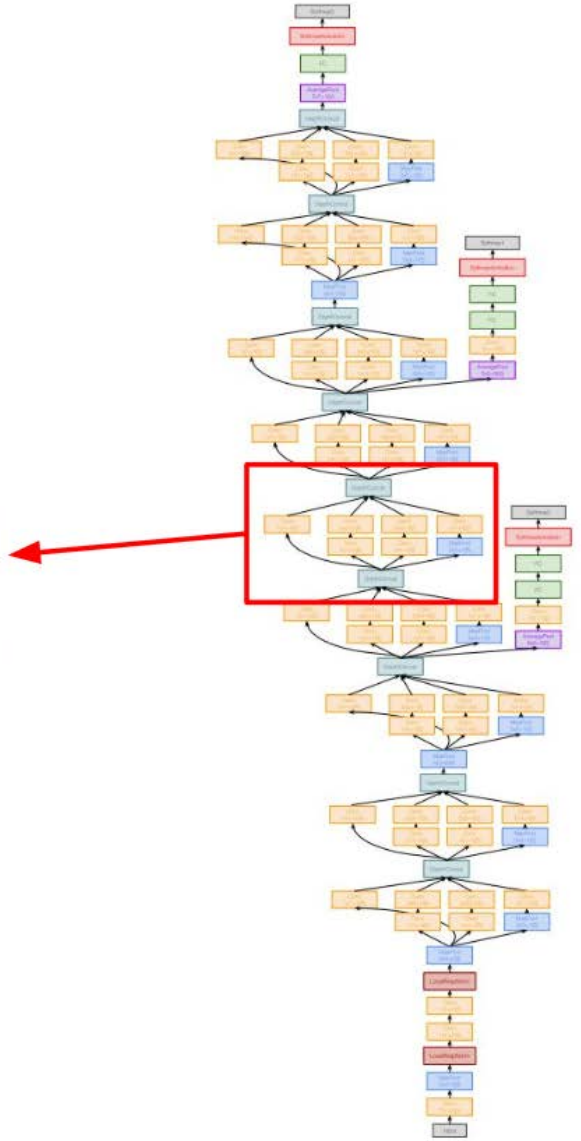
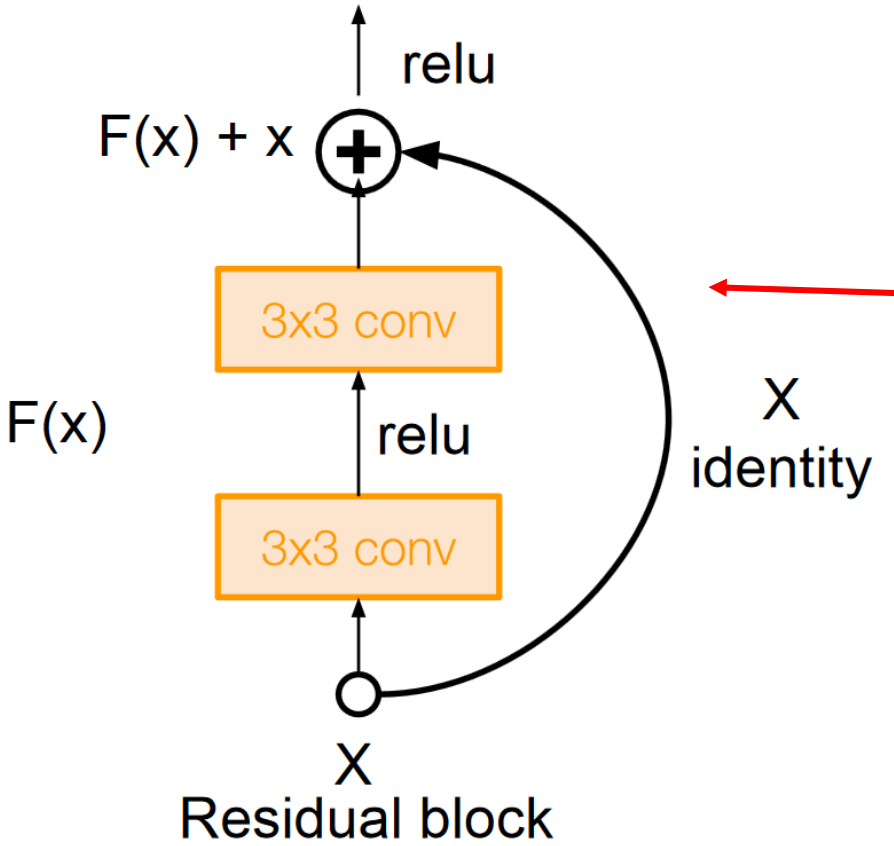


Image classification
Block based architectures – ResNet

About

- ResNet tackles vanishing gradients by adding **skip connections** between modules
- The resulting **residual blocks** are chained together, the output of each block is an **additive transformation** of the input



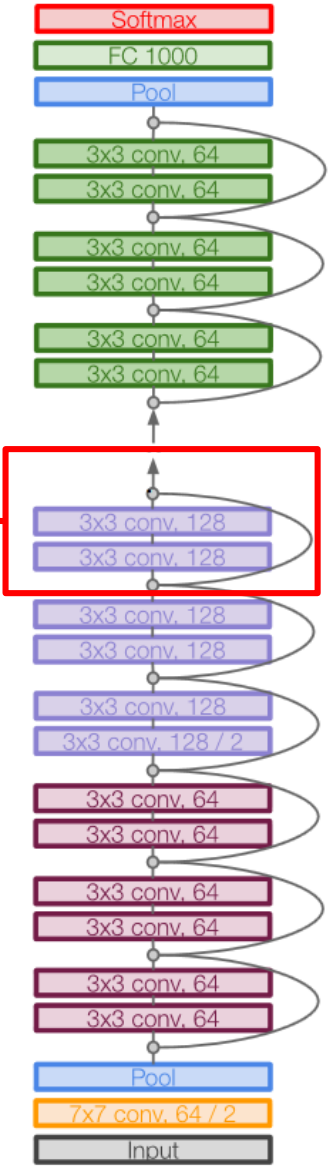
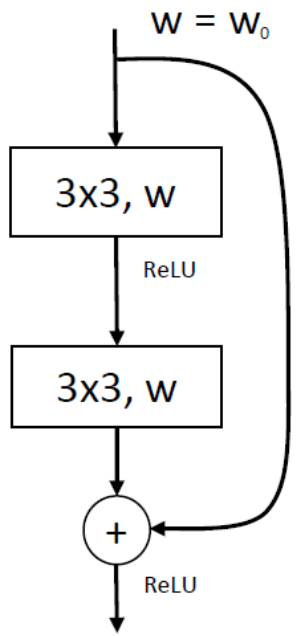


Image classification
Block based architectures – ResNet bottleneck

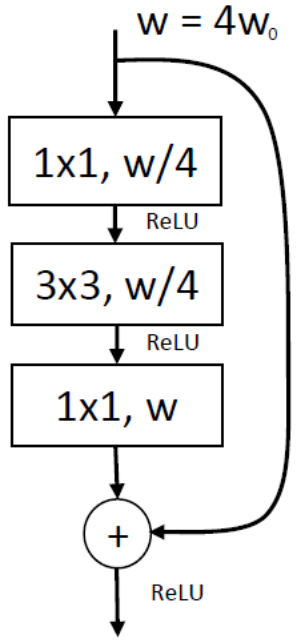
About

- The residual block with **bottleneck** reduces the number parameters -> we can get even deeper models
- Based around the **information compression** assumption / theory



Normal residual block

- better behaved gradients
- faster training



Residual block with **bottleneck**

- reduces computational burden

Image classification
Training problem

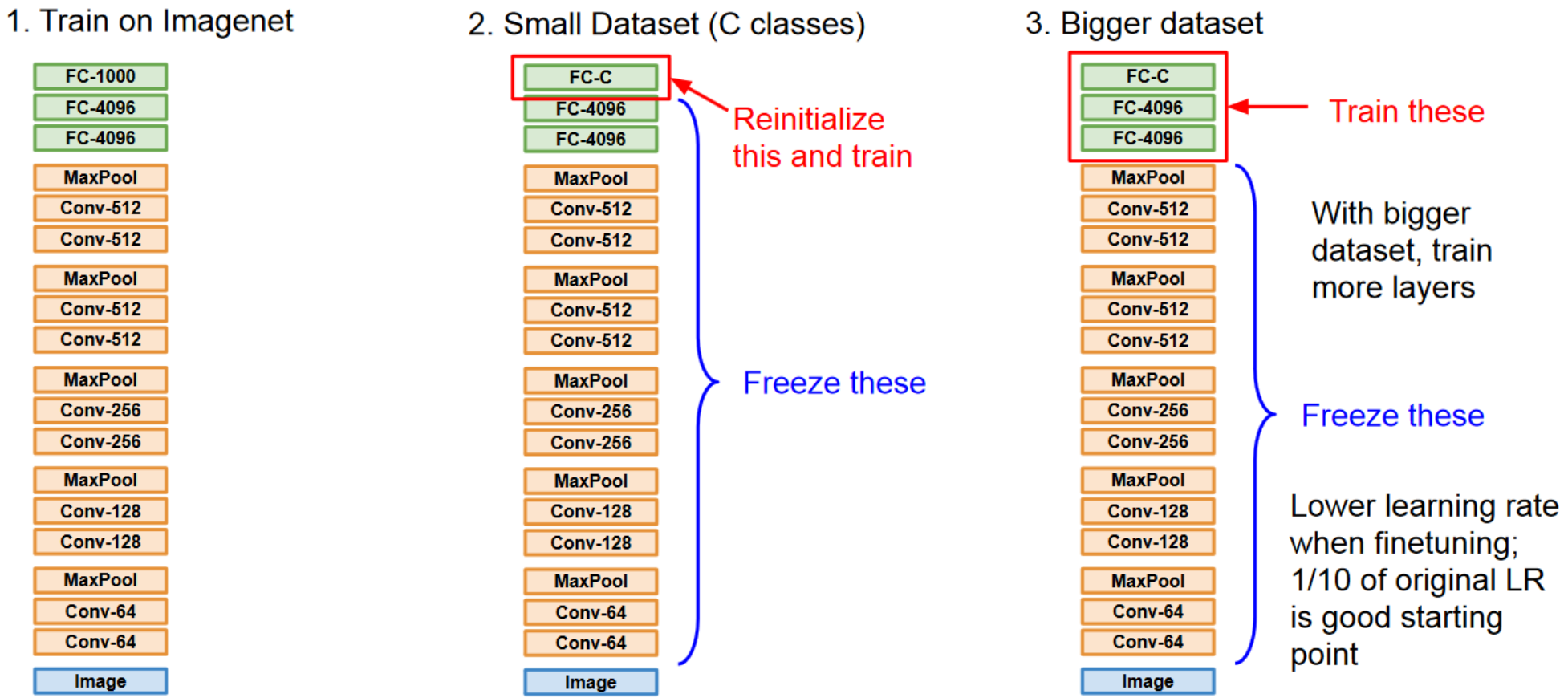
Issue

- No matter how good our gradients are, training huge convolutional neural networks with a **simple first order optimization method** (gradient descent) still **requires large amounts of training data**
- For most tasks, there not much labeled data available
- **Manual labeling is an expensive process!**

Possible solution:

- Feature extractor **pre-training**, a.k.a. **transfer learning**

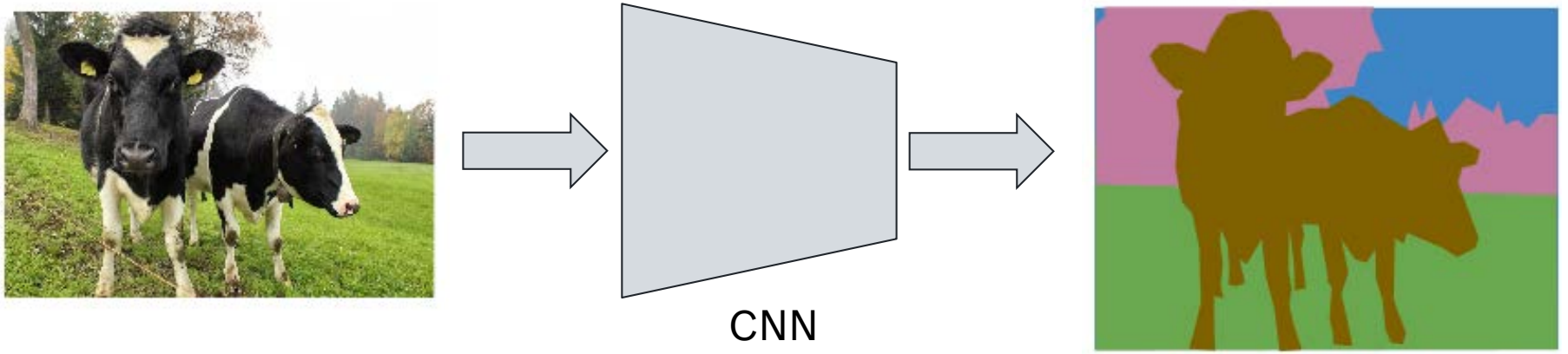
Image classification
Transfer learning



SEMANTIC SEGMENTATION

BOSCH

Semantic segmentation
Task definition



Semantic segmentation -> Per-pixel classification -> **we can use similar loss functions**

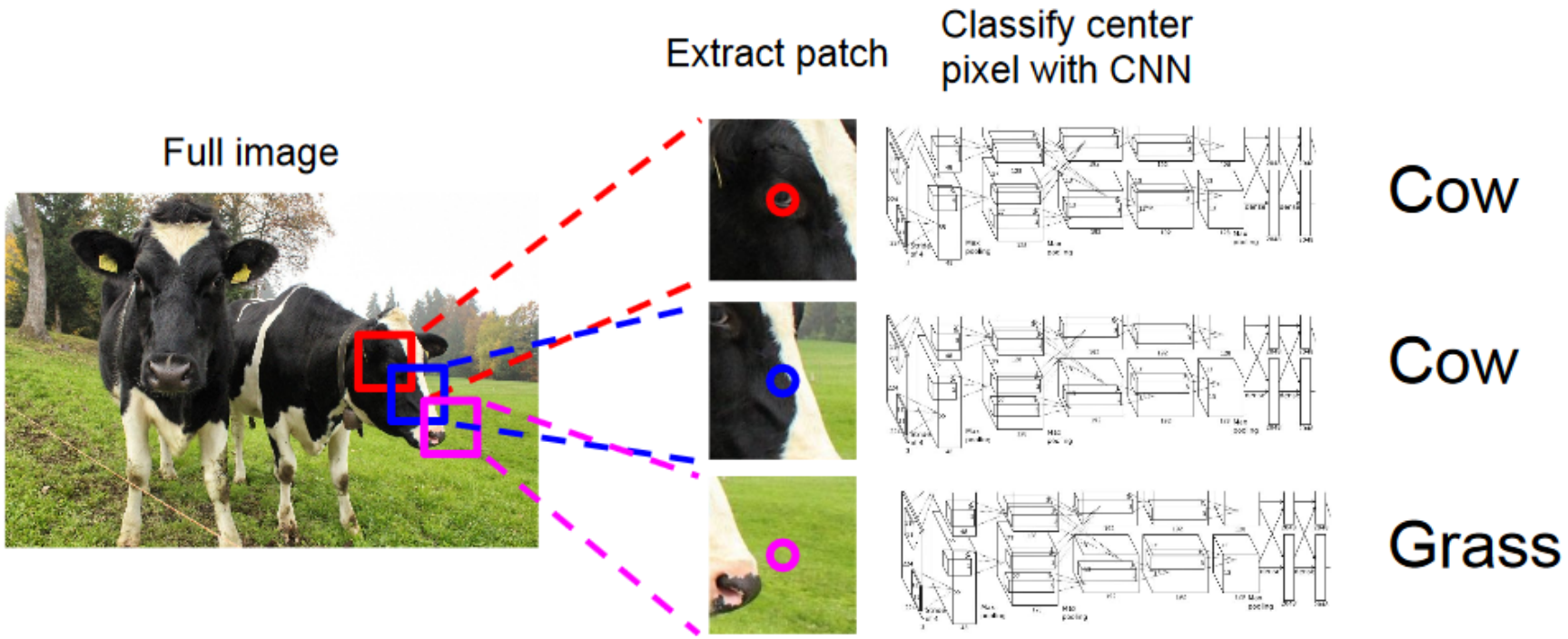
$$L(y) = -\log(p(y|\theta))$$

BOSCH

Semantic segmentation
Example

BOSCH

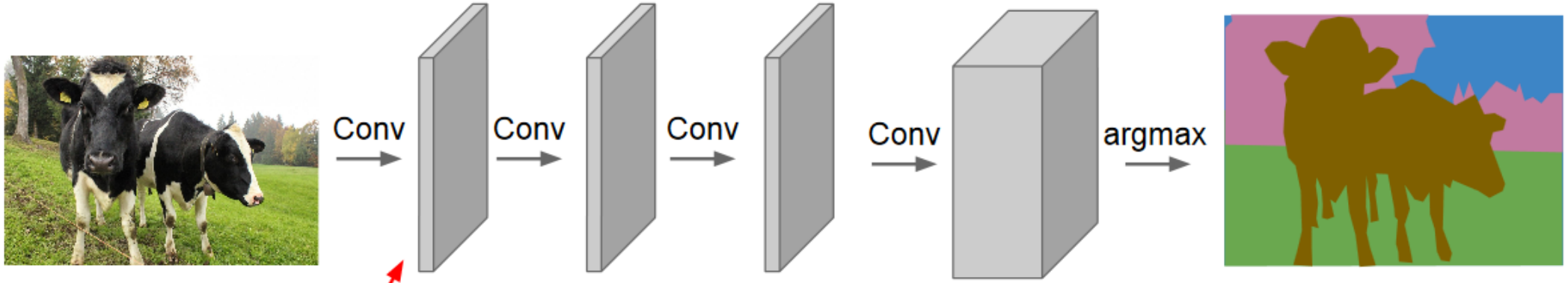
Semantic segmentation
Naive approach



Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

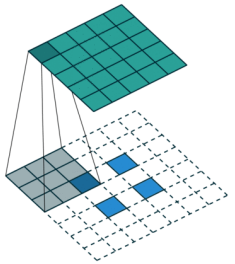
Semantic segmentation
Better approach – but impractical

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Problem: convolutions at original image resolution will be very expensive ...

Semantic segmentation
Better approach

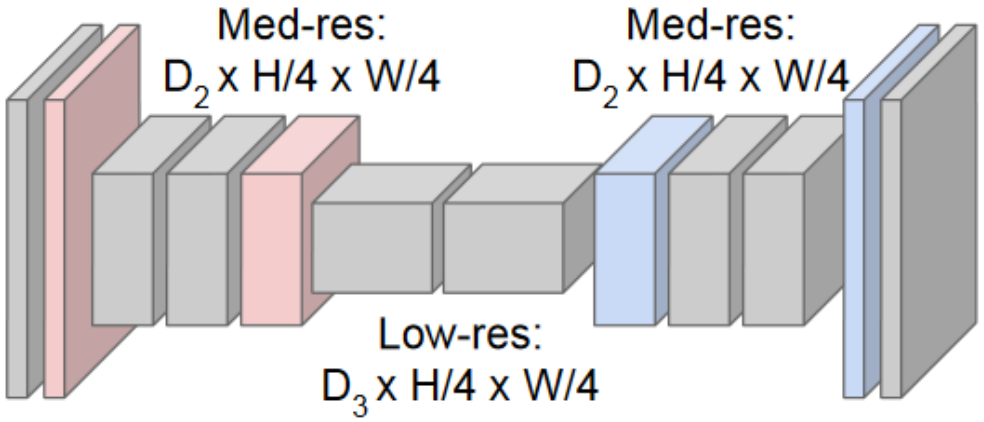


Transposed convolution

(a.k.a. “deconvolution”) for upsampling

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



High-res:
D1 x H/2 x W/2

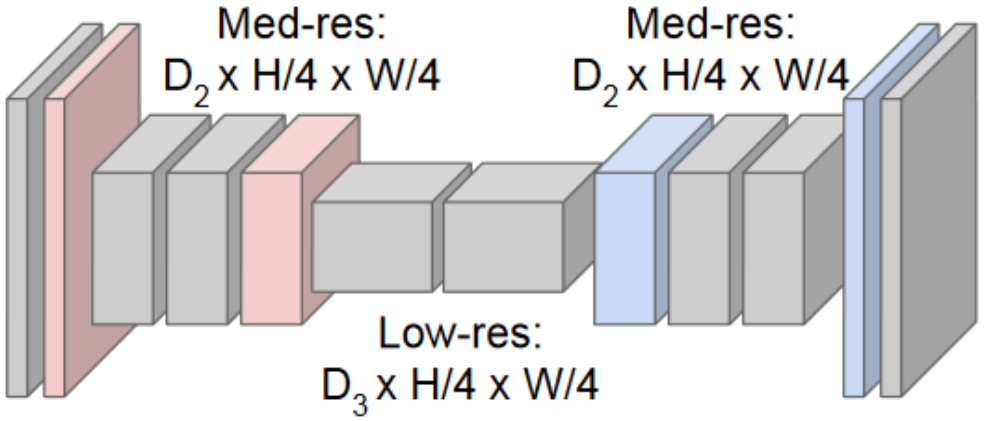
High-res:
D1 x H/2 x W/2

Predictions:
H x W

Semantic segmentation
Better approach

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

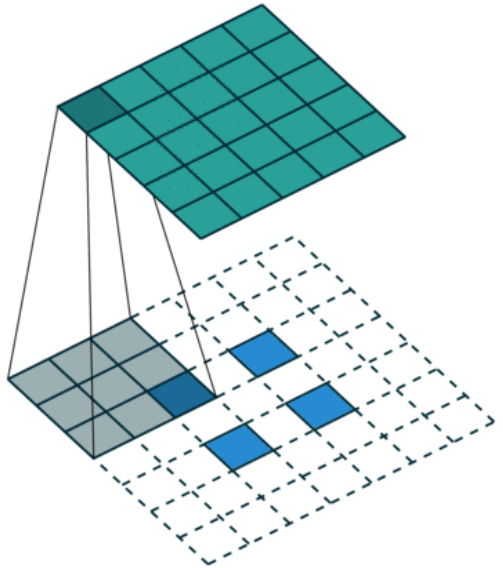
Predictions:
H x W

Feature extractor -> can be pre-trained on image classification

Semantic segmentation
Transposed convolution

About

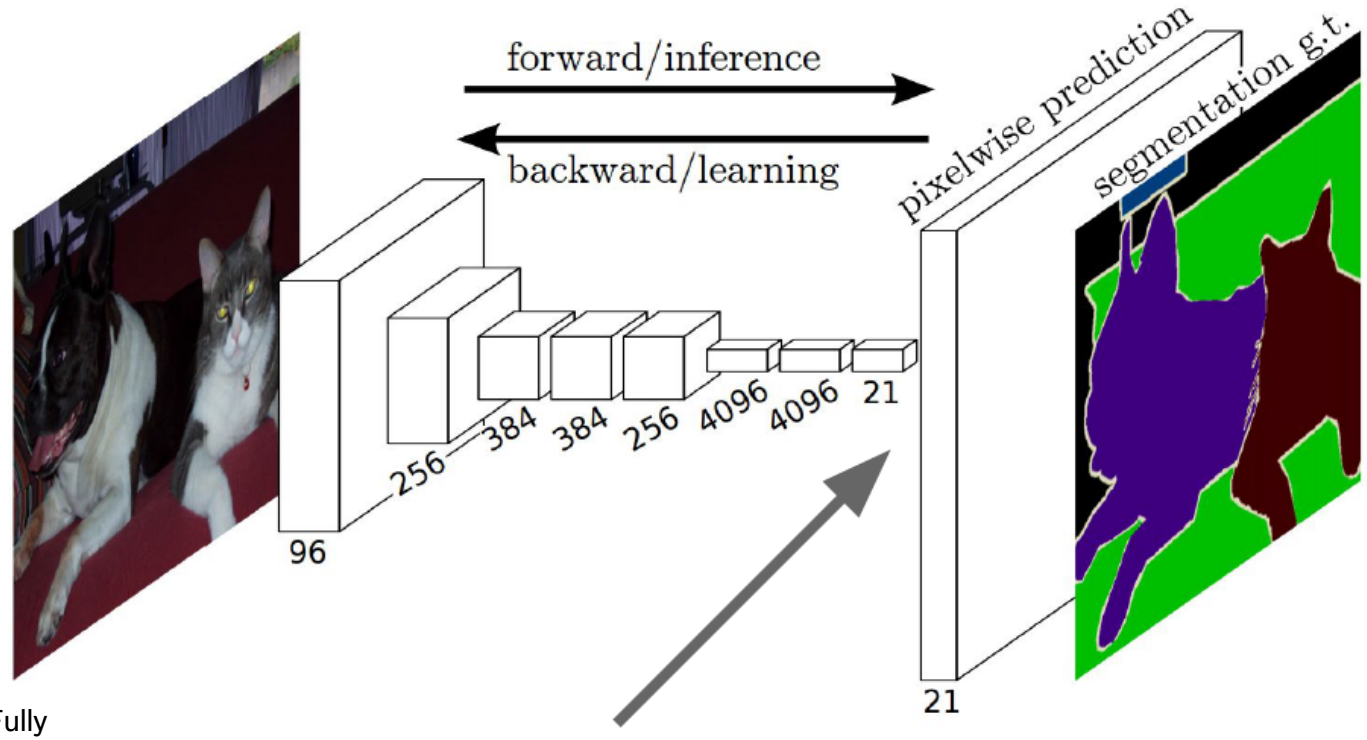
- Upsamples its input
- It is parameterized, therefore it can be trained
- Should obtain better results than standard interpolation methods



Transposed convolution

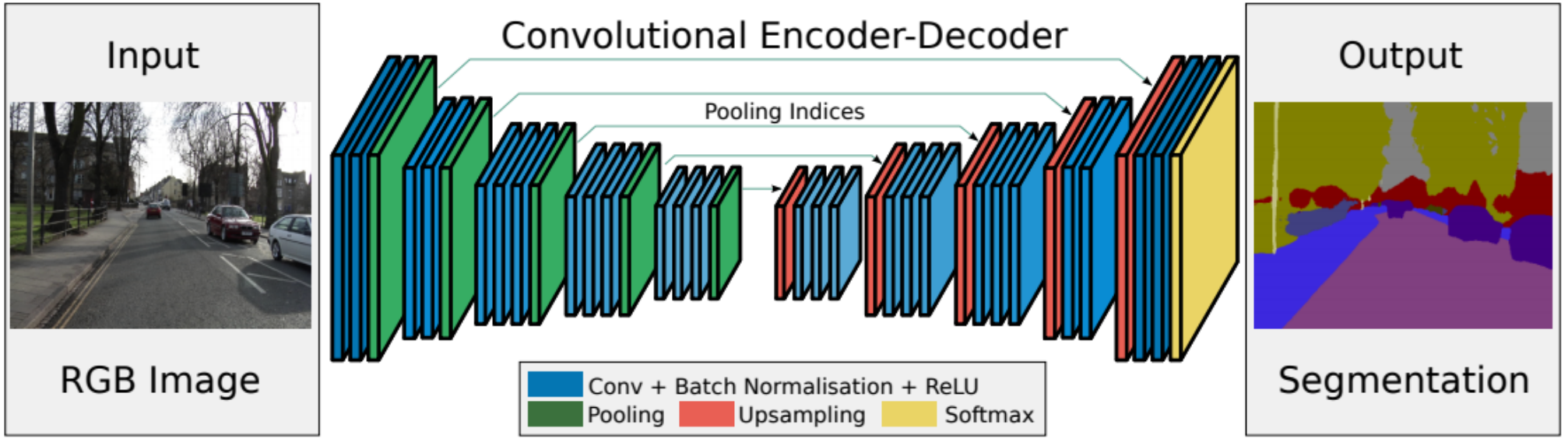
(a.k.a. “deconvolution”)

Semantic segmentation
Simplest architecture



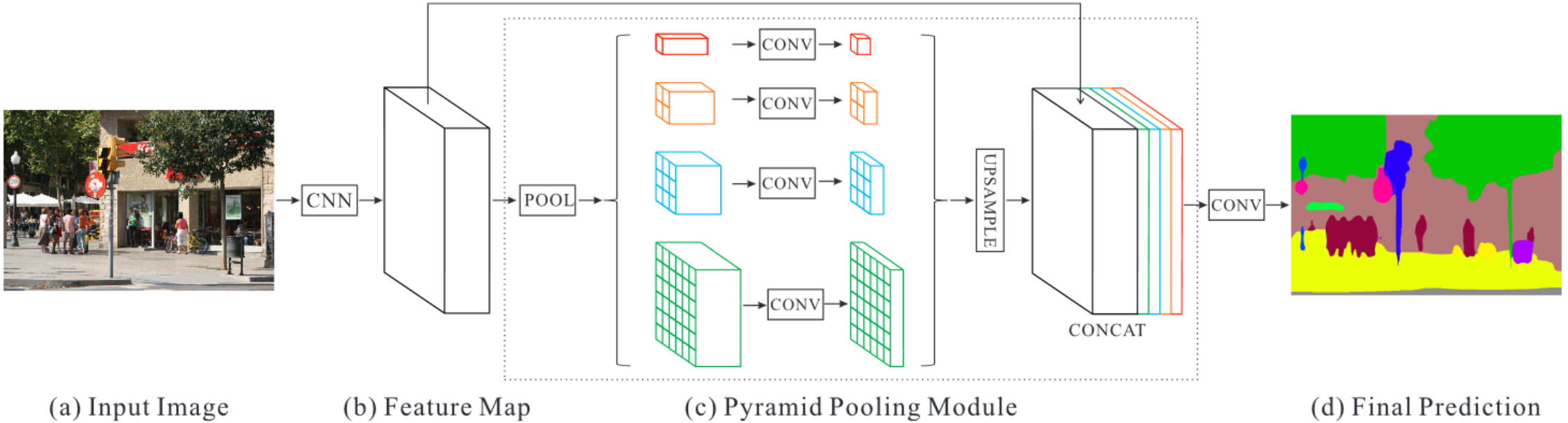
Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

Semantic segmentation
Better architectures



Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. "**Segnet**: A deep convolutional encoder-decoder architecture for image segmentation." *arXiv preprint arXiv:1511.00561*(2015).

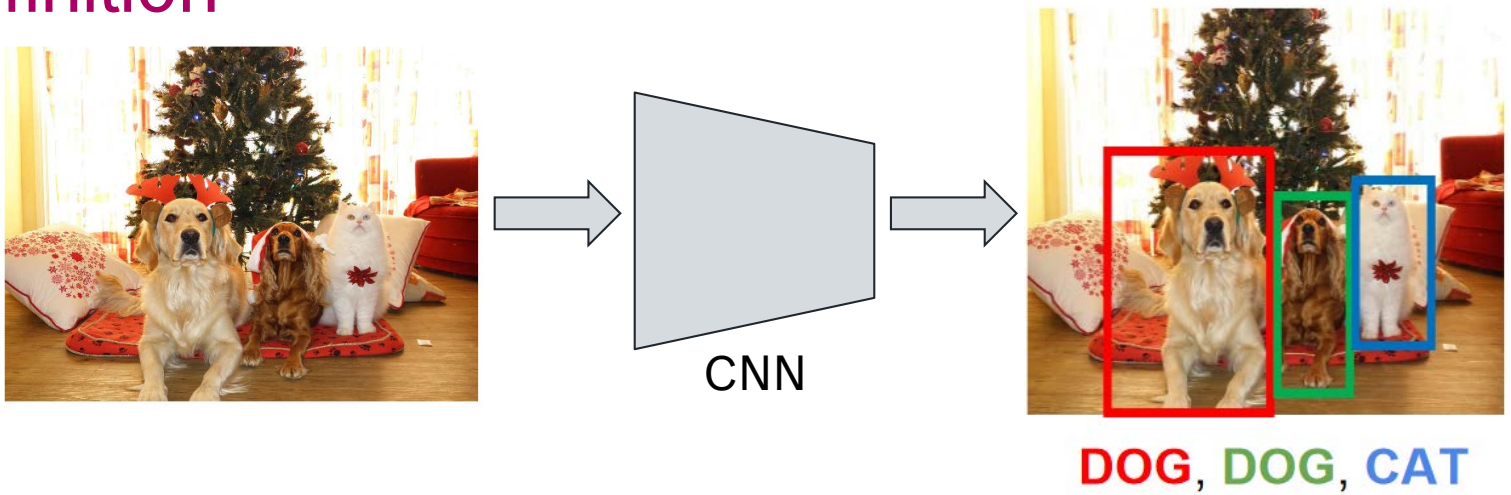
Semantic segmentation
Better architectures



Zhao, Hengshuang, et al. "Pyramid scene parsing network." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

OBJECT DETECTION

Object detection
Task definition



Object detection -> predict **bounding boxes** and the **associated classes**

-> Compound loss function

$$L(y_c, y_{coord}) = \alpha\, L_c(y_c) + \beta\, L_{coord}(y_{coord})$$

$$L_c = NLL(y_c) = -\log(p(y_c))$$

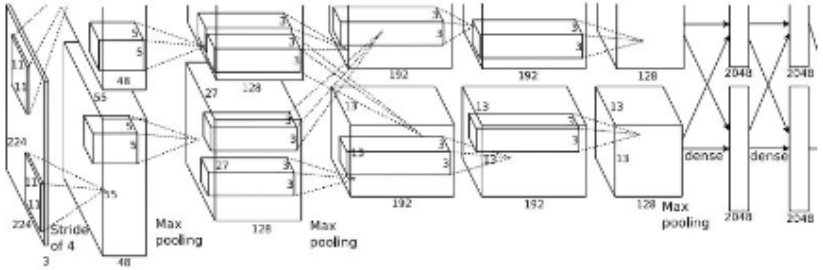
$$L_{coord}(y_{coord}) = MSE(y_{coord}) = (y_{coord} - y_{truecoord})^2$$

Object detection
Example

Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767* (2018).

Object detection
Naive approach

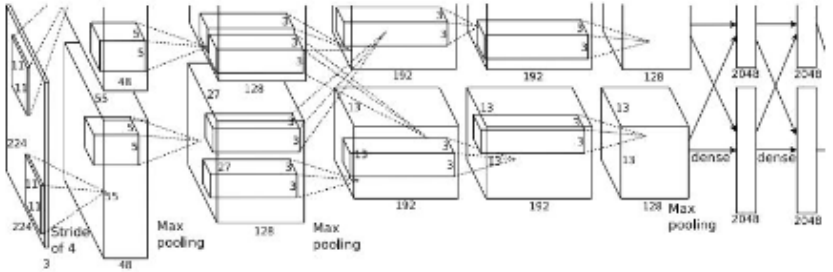
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Object detection
Naive approach

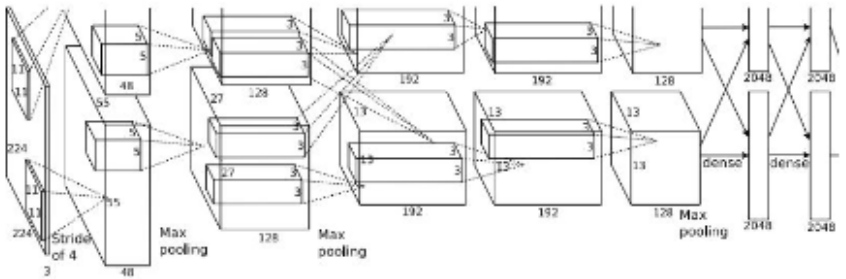
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object detection
Naive approach

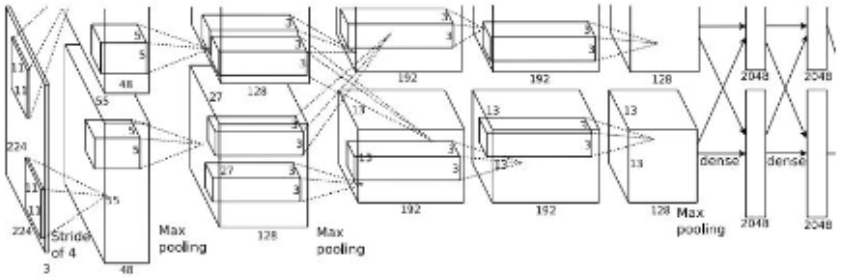
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

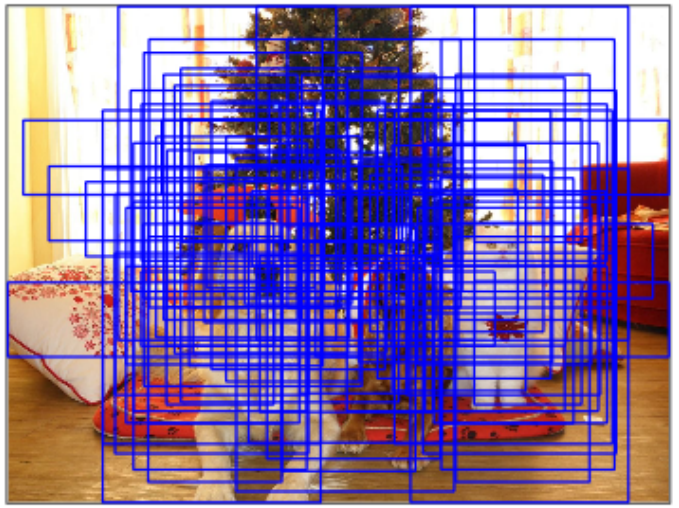
Object detection
Naive approach

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

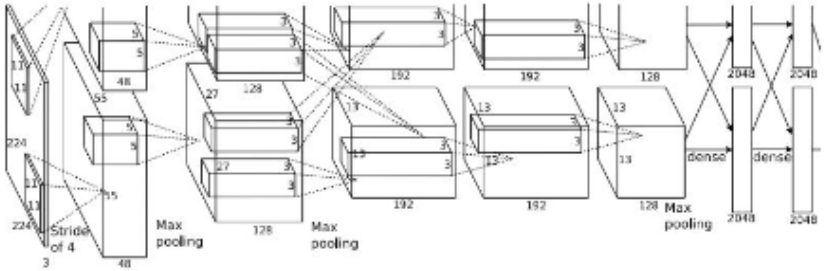


Dog? NO
Cat? YES
Background? NO

Object detection
Naive approach



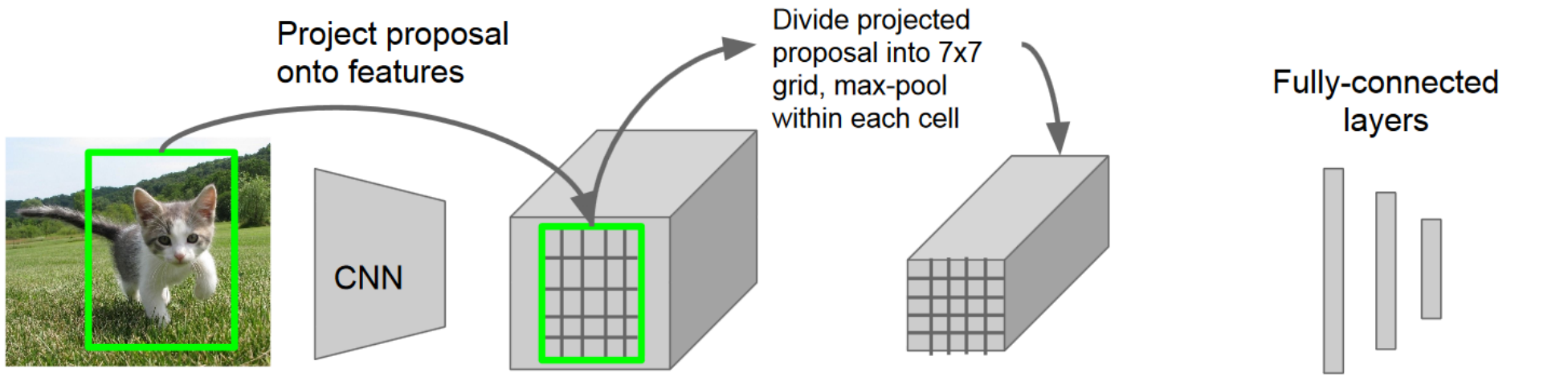
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Object detection
Better approach – only classify some proposals



Girshick, "Fast R-CNN", ICCV 2015.

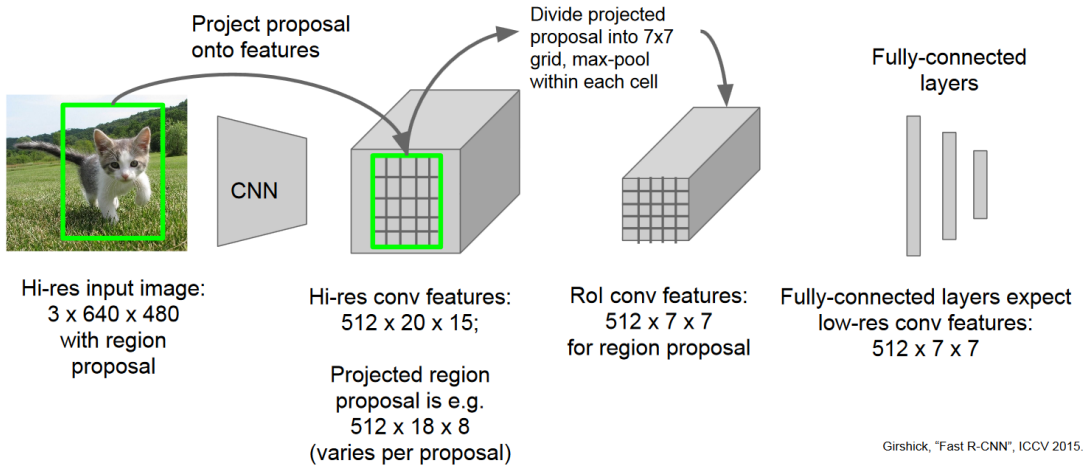
Object detection
Better approach – only classify some proposals

Issues

- Independently processing each cropped feature block is very costly

Solution

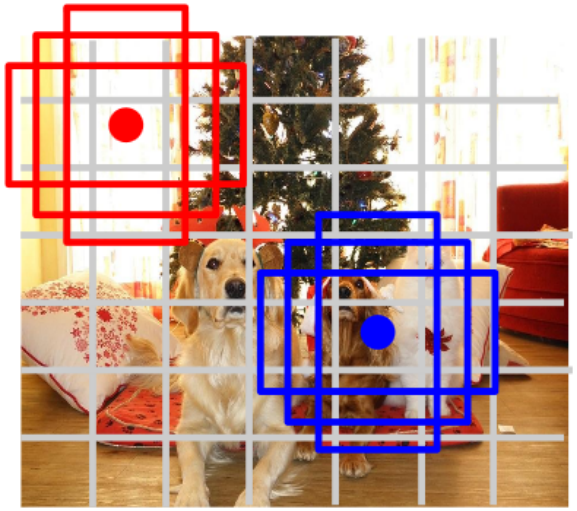
- Ideally, we would like to obtain both the bounding boxes and the classses in a single pass



Girshick, "Fast R-CNN", ICCV 2015.

Object detection
One pass detectors – YOLO / SSD

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

Output:
7 x 7 x (5 * B + C)

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

Object detection
Bounding box parameterization in YOLO

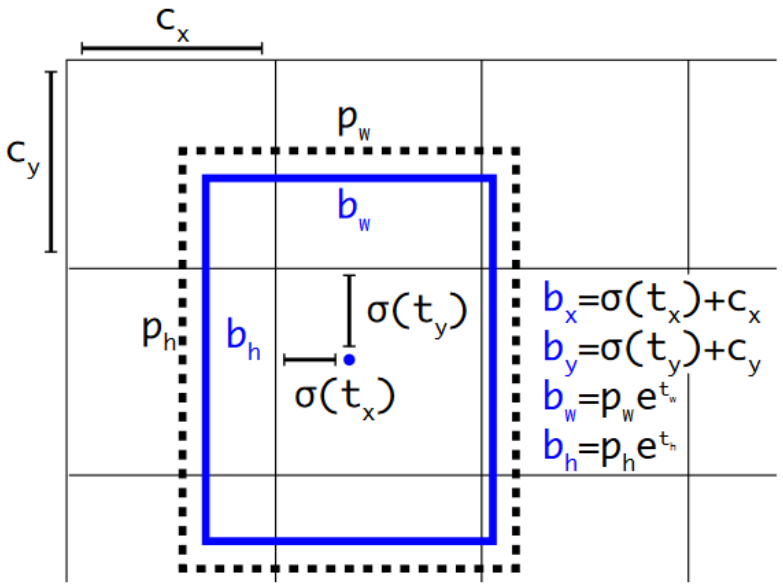


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

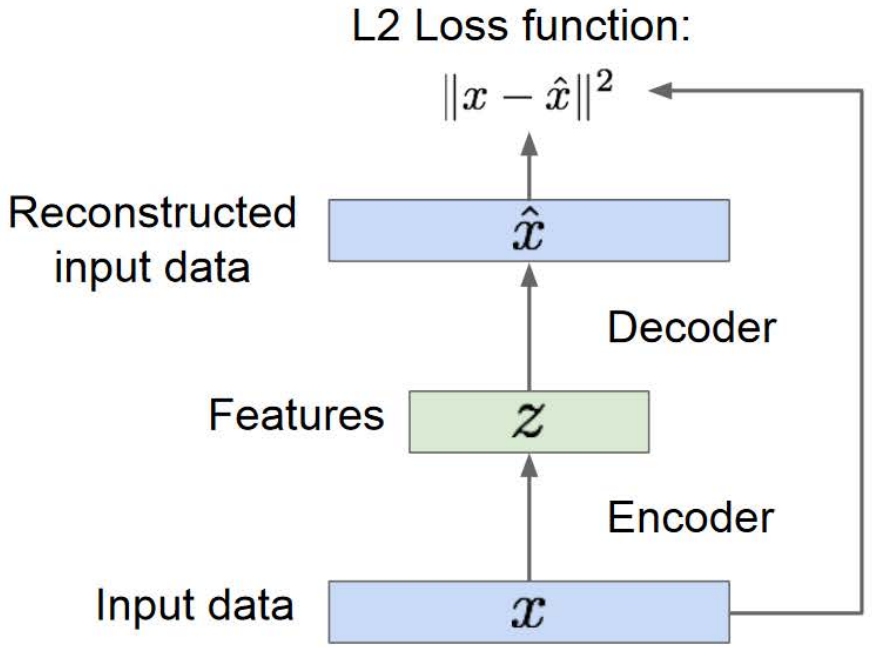
- $[c_x, c_y]$ — Coordinates of current cell (integers)
- $[\sigma(t_x), \sigma(t_y)]$ — Predicted bounding box center coordinates (relative to the current grid cell)
- $[b_w, b_h]$ — Predicted bounding box width and height (relative) to the anchor / base bounding box
- $[p_w, p_h]$ — Width and height of the anchor / base bounding box (pixels)

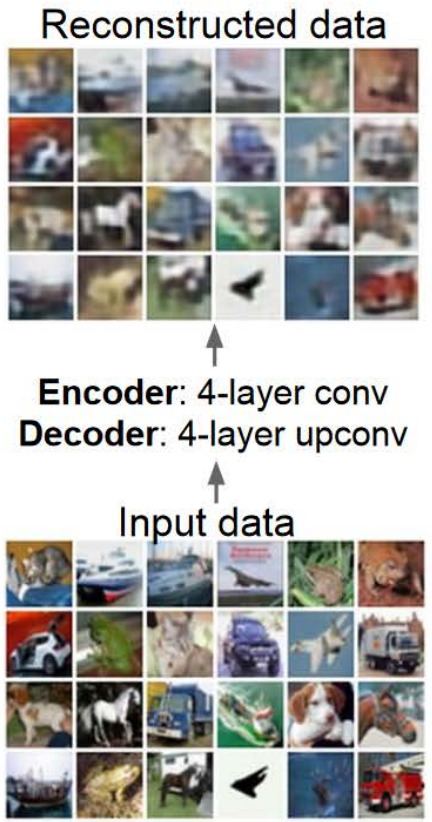
AUTOENCODERS

Autoencoders
Overview

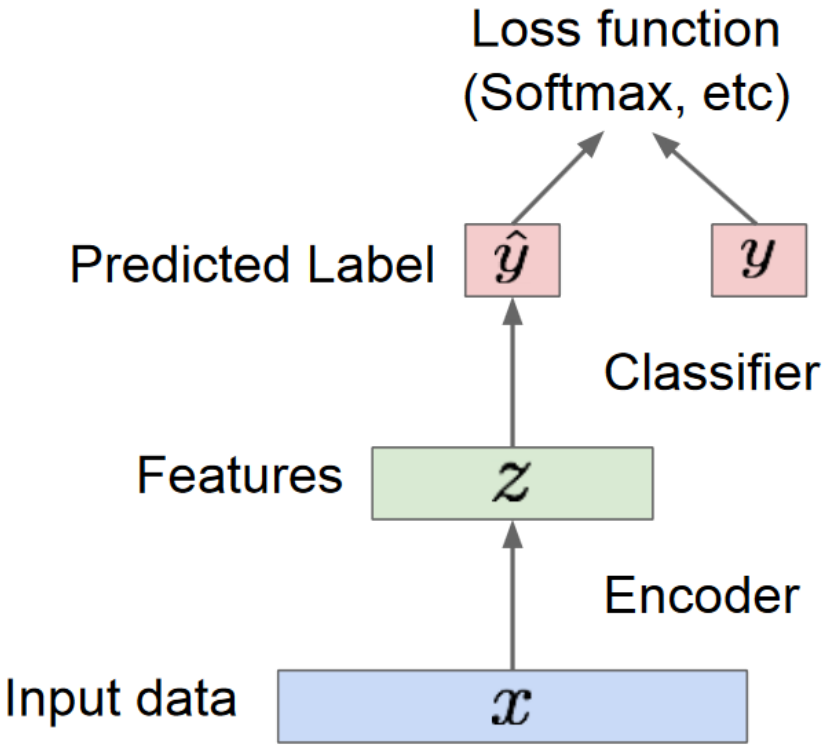
About

- They do not require labels (**unsupervised learning**)
- Based on information bottleneck (compression)
- Can be used for compression, denoising etc.
- Also great for **pre-training** segmentation, classification, or object detection networks





Autoencoders
Pre-training supervised models

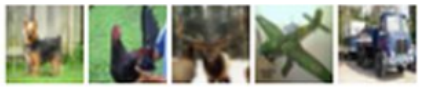


Encoder can be used to initialize a **supervised** model

Fine-tune encoder jointly with classifier

bird plane
dog deer truck

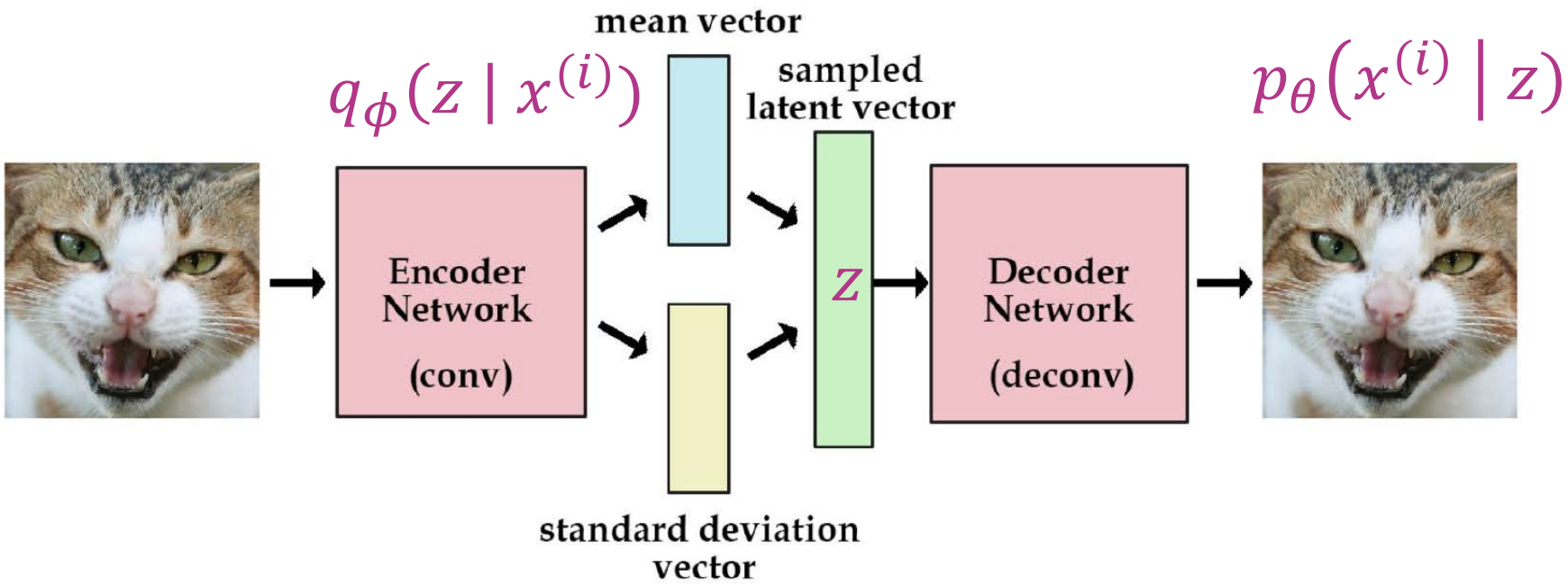
Train for final task (sometimes with small data)



GENERATIVE

BOSCH

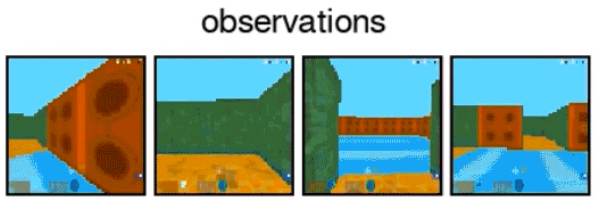
Generative architectures
Variational Autoencoder (VAE)

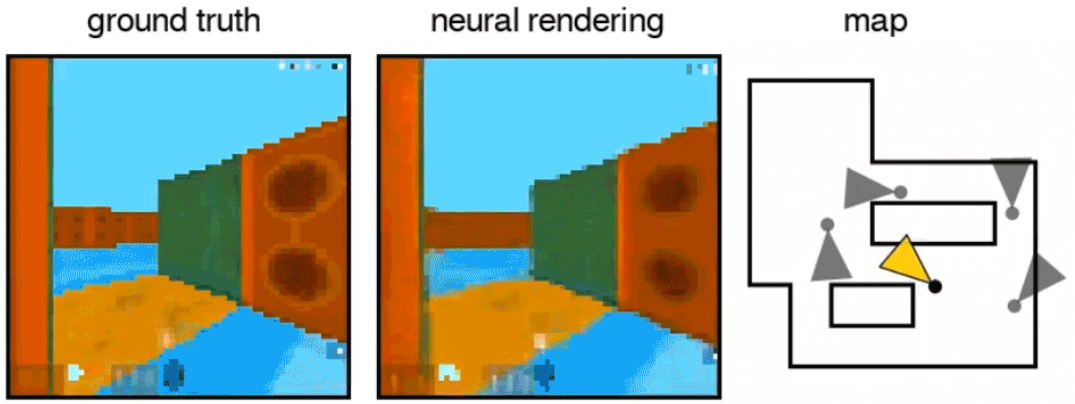


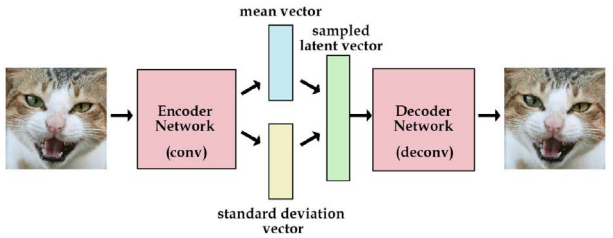
Maximize: $\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))$

http://kvfrans.com/variational-autoencoders-explained/

Generative architectures
Variational Autoencoder (VAE)

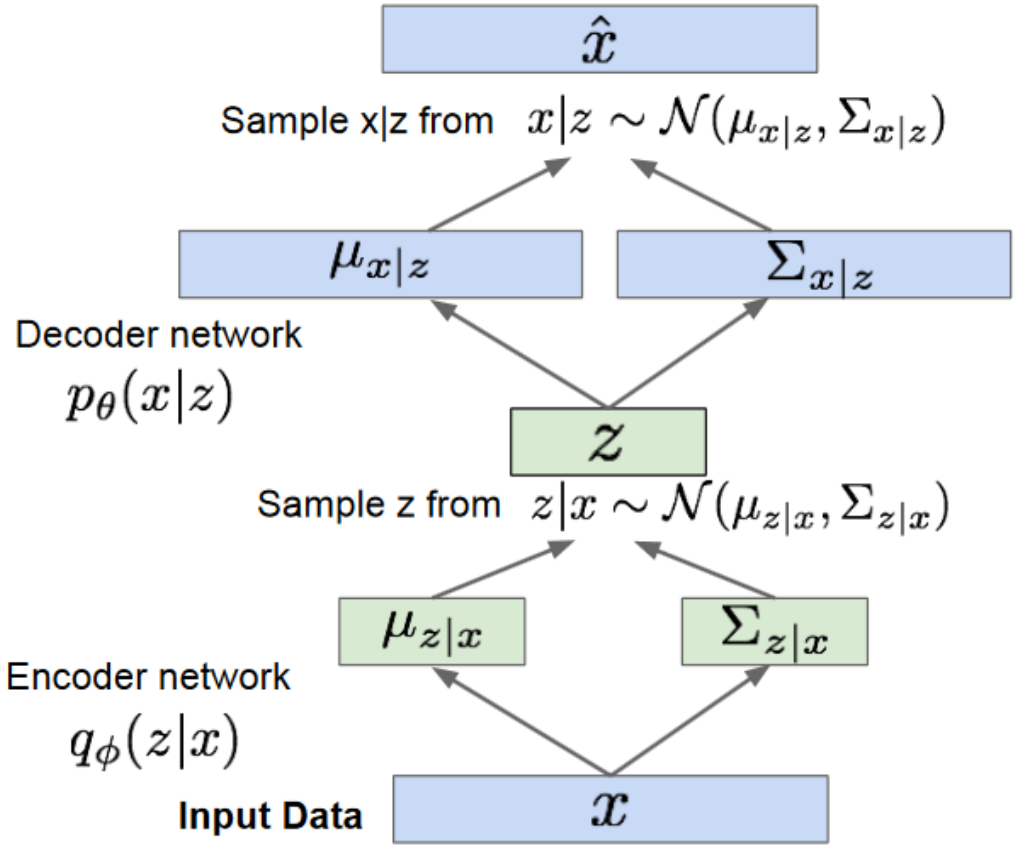




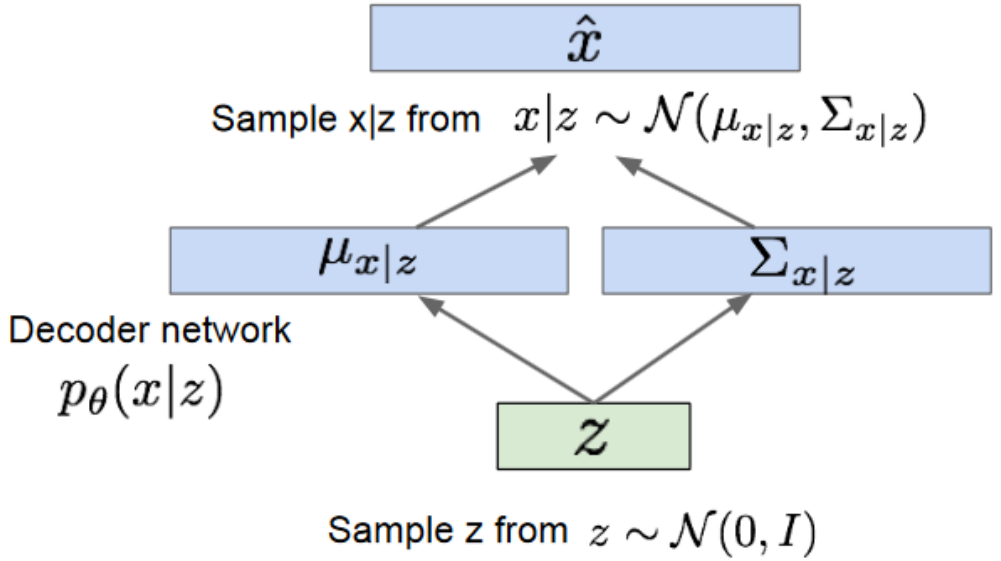


https://deepmind.com/blog/neural-scene-representation-and-rendering/

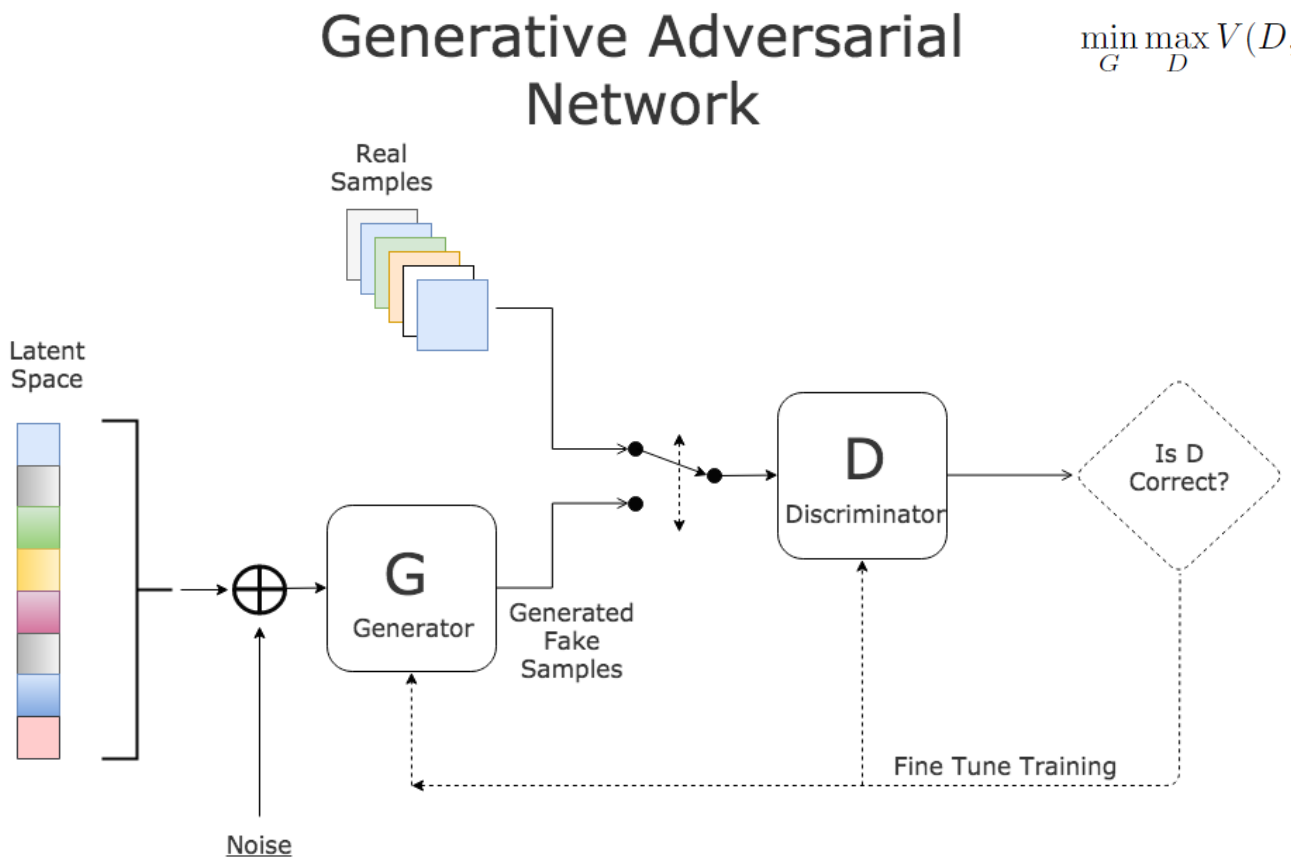
Generative architectures
Variational Autoencoder (VAE)



To generate new images use:



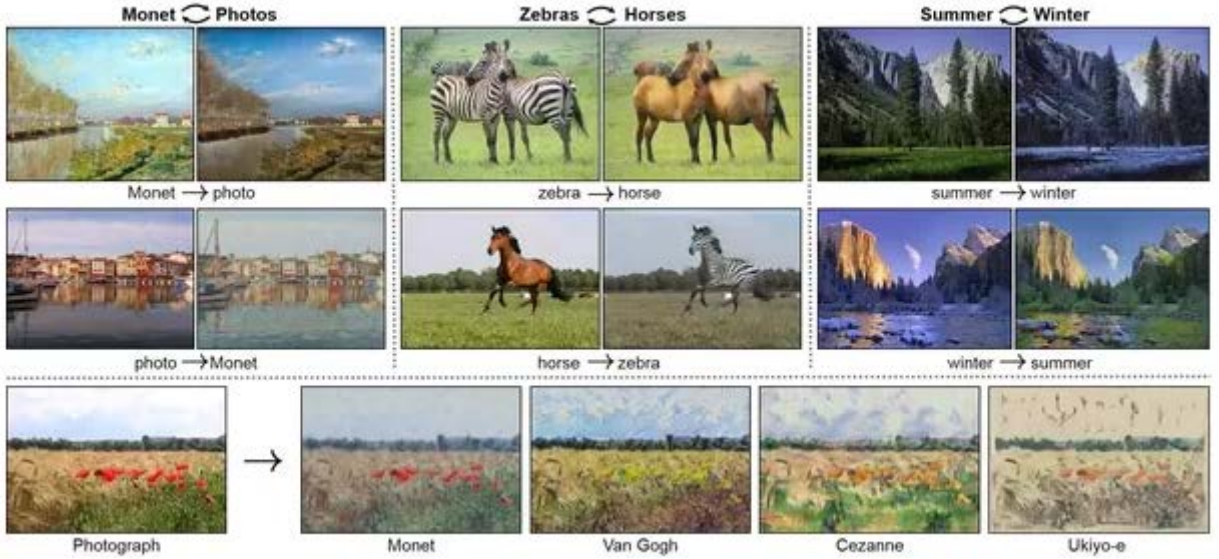
Generative architectures
Generative Adversarial Network (GAN)



$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems*. 2014.

Generative architectures
Generative Adversarial Network (GAN)

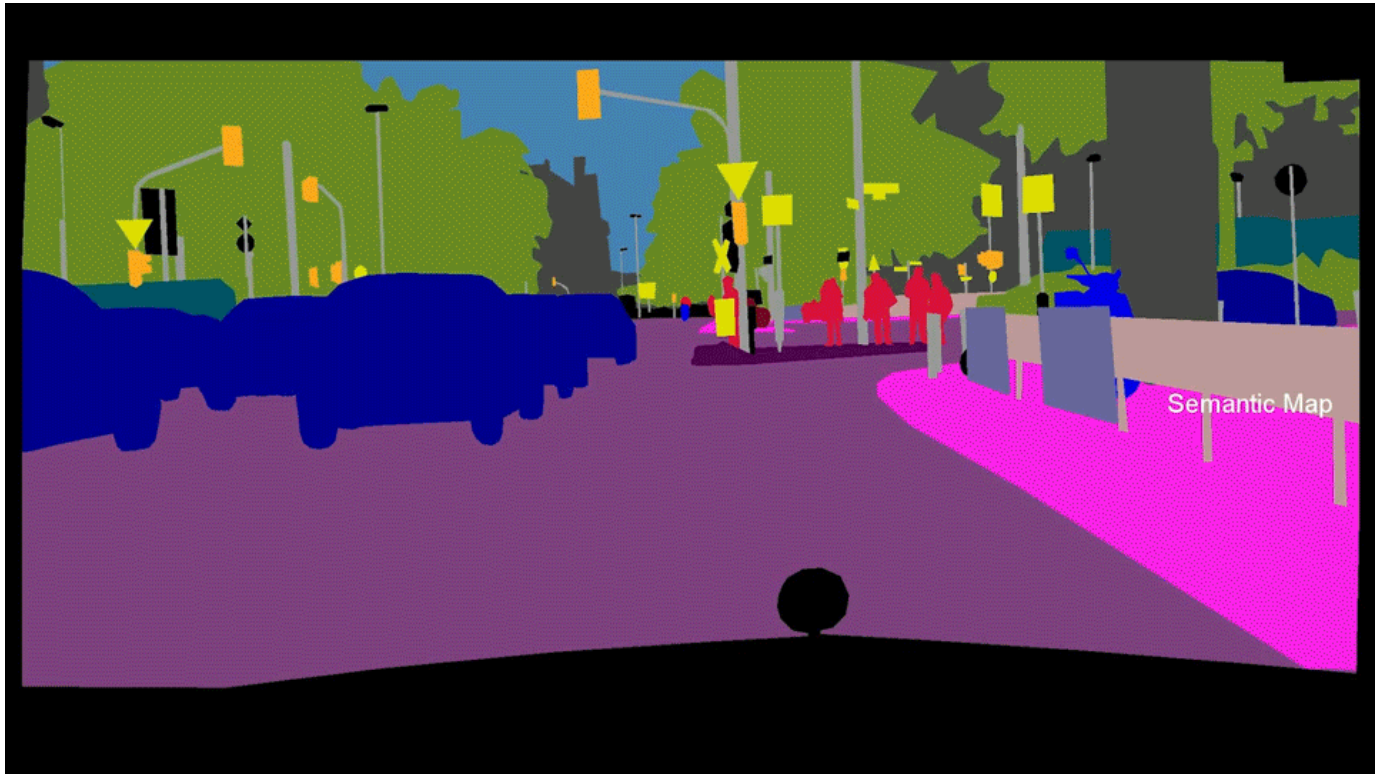


Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." *arXiv preprint*(2017).

Liu, Ming-Yu, Thomas Breuel, and Jan Kautz. "Unsupervised image-to-image translation networks." *Advances in Neural Information Processing Systems*. 2017.

https://artix41.github.io/static/domain-adaptation-in-2017/index.html

Generative architectures
Generative Adversarial Network (GAN)



https://tcwang0509.github.io/pix2pixHD/

OPTIMIZING FOR EMBEDDED

Optimizing for embedded
Some methods

- Pruning: remove “unimportant” weights or neurons
- Weight quantization: restrict weights to a discrete set of possible values; ternarization / binarization
- Filter design optimization: reduce the number of necessary operations (use 1x1 filters or dephwise convolution filters) (MobileNet, ShuffleNet)
- Knowledge distillation – Teacher/Student methods; train a smaller network using what the teacher has learned

https://medium.com/@nicolas_19145/state-of-the-art-in-compressing-deep-convolutional-neural-networks-cfd8c5404f22

BOSCH

THANK YOU FOR YOUR ATTENTION