Digitale Bildverarbeitung und Mustererkennung

This lecture in one slide

Introduction and motivation for deep learning
Neural network conception

Optimization
Stochastic Gradient Descent
Momentum methods
Adaptive methods
Vanishing and Exploding Gradients
Weight Initialization

Regularization

Neural Network Optimization – Stochastic gradient descent

Stochastic (gradient of a batch) as opposed to deterministic (gradient of the whole dataset)

Unbiased estimate of the gradient.
Computational effort

Neural Network Optimization – Stochastic gradient descent

Stochastic

Randomly selected set of *m* training samples for a batch achieves an **unbiased estimate of the gradient**.

Computational effort

Neural Network Optimization – Stochastic gradient descent

Stochastic

Standard error of the mean.

Limiting number of *m* samples per batch, sets an upper bound to the **computational effort** during the update (growing datasets, growing sample size)

SGD

```
Biases: [[ 3.99840403]]
Prediction [[ 13.96173477]]

Gradient [  7.84316492   7.84316492  23.60477257]
Weights: [ 1.99761569  1.99761569  1.99283969]
Biases: [[ 3.997612]]
Prediction [[ 13.9427824]]

Gradient [  7.7917676    7.7917676   23.47492409]
Weights: [ 1.99683142  1.99683142  1.99047923]
Biases: [[ 3.99682403]]
Prediction [[ 13.9239502]]
```

Observations

Gradients:
- Optimization slows down with smaller gradients

Weights and Biases:
- Symmetry
- Different initialization would lead to different outcome

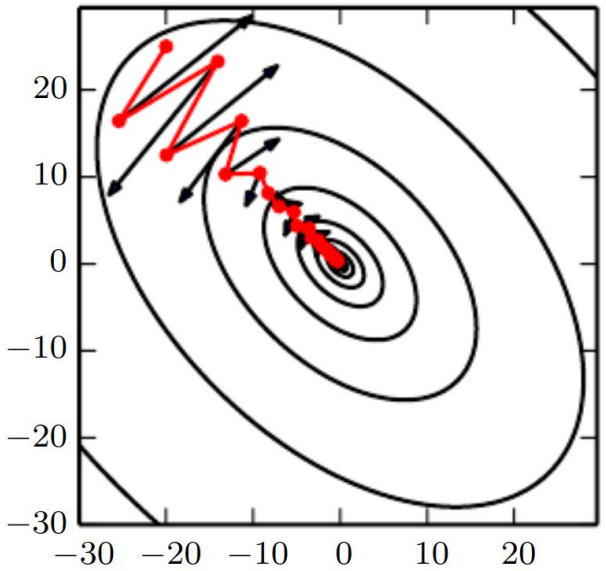
Neural Network Optimization – Momentum methods

Average gradients of past iterations as velocity **v**

Consider recent gradients stronger by accounting for friction α in $[0,1)$

$$\mathbf{v} = \alpha\mathbf{v} - \epsilon\mathbf{g},$$

$$\theta' = \theta + \mathbf{v}.$$



Red velocity, black current gradient
[1]

Neural Network Optimization – Adaptive methods

Adapting the learning rate throughout the optimization process

AdaGrad (Adaptive Gradient)

Individually adapts the learning rates of each model parameter, inversely proportional to the historical values of the (squared) gradients. This helps features which are "rarely" updated.

RMSProp

modifies AdaGrad by approaching the accumulation of historical gradient values as a exponentially weighted moving average. Influence of very old historical values is reduced.

Adam

combination of exponential weight decay together with first- and second-order moments (mean and variance).

Neural Network Optimization – Adaptive methods

Adapting the learning rate throughout the optimization process

AdaGrad (Adaptive Gradient)

Individually adapts the learning rates of each model parameter, inversely proportional to the historical values of the (squared) gradients. This helps features which are "rarely" updated.

RMSProp

modifies AdaGrad by approaching the accumulation of historical gradient values as a exponentially weighted moving average. Influence of very old historical values is reduced.

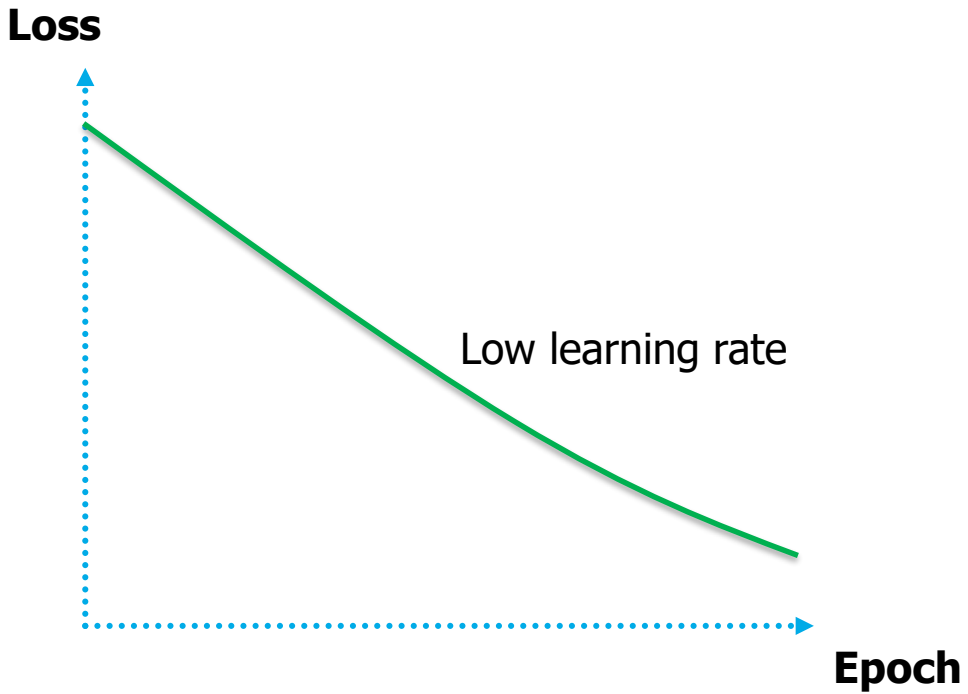
Adam

combination of exponential weight decay together with first- and second-order moments (mean and variance).

Note:
There is no single best optimization algorithm. Adam is generally and, hence, a reasonable choice for a start.

Neural Network Optimization – Learning Rate

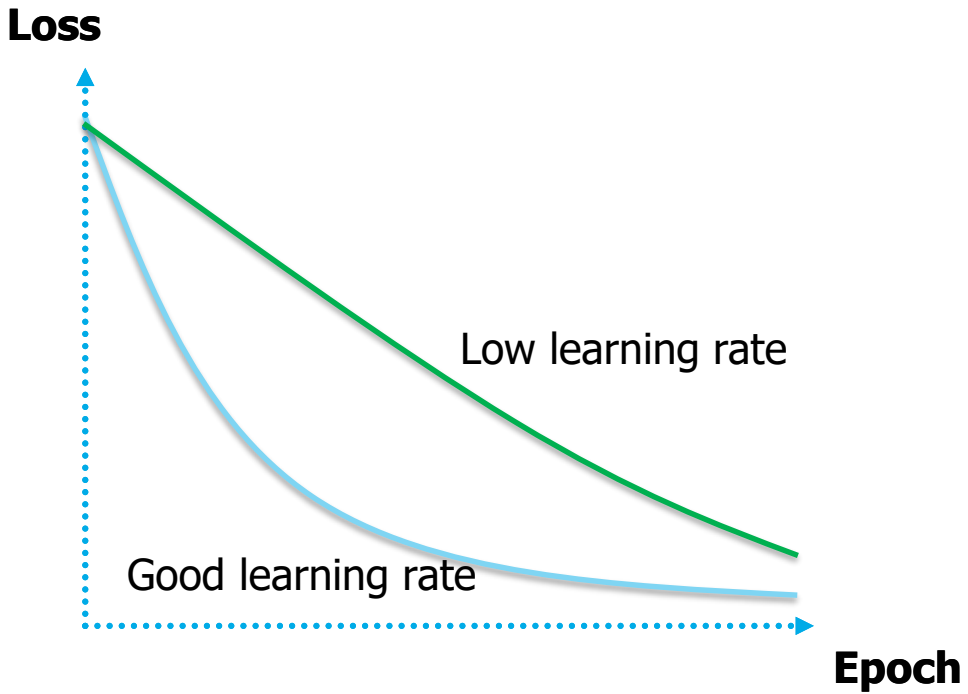
Low learning rates
Loss decay will be linear, and result in high training times.



Neural Network Optimization – Learning Rate

Low learning rates
Loss decay will be linear, and result in high training times.

Higher learning rates
Loss decay will start to decline exponentially.

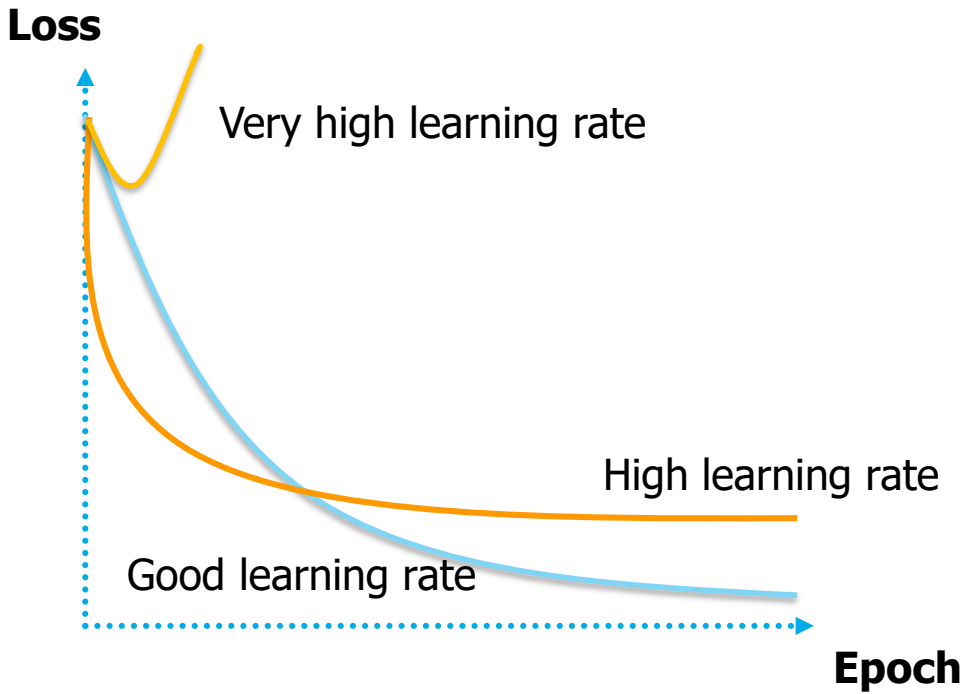


Neural Network Optimization – Learning Rate

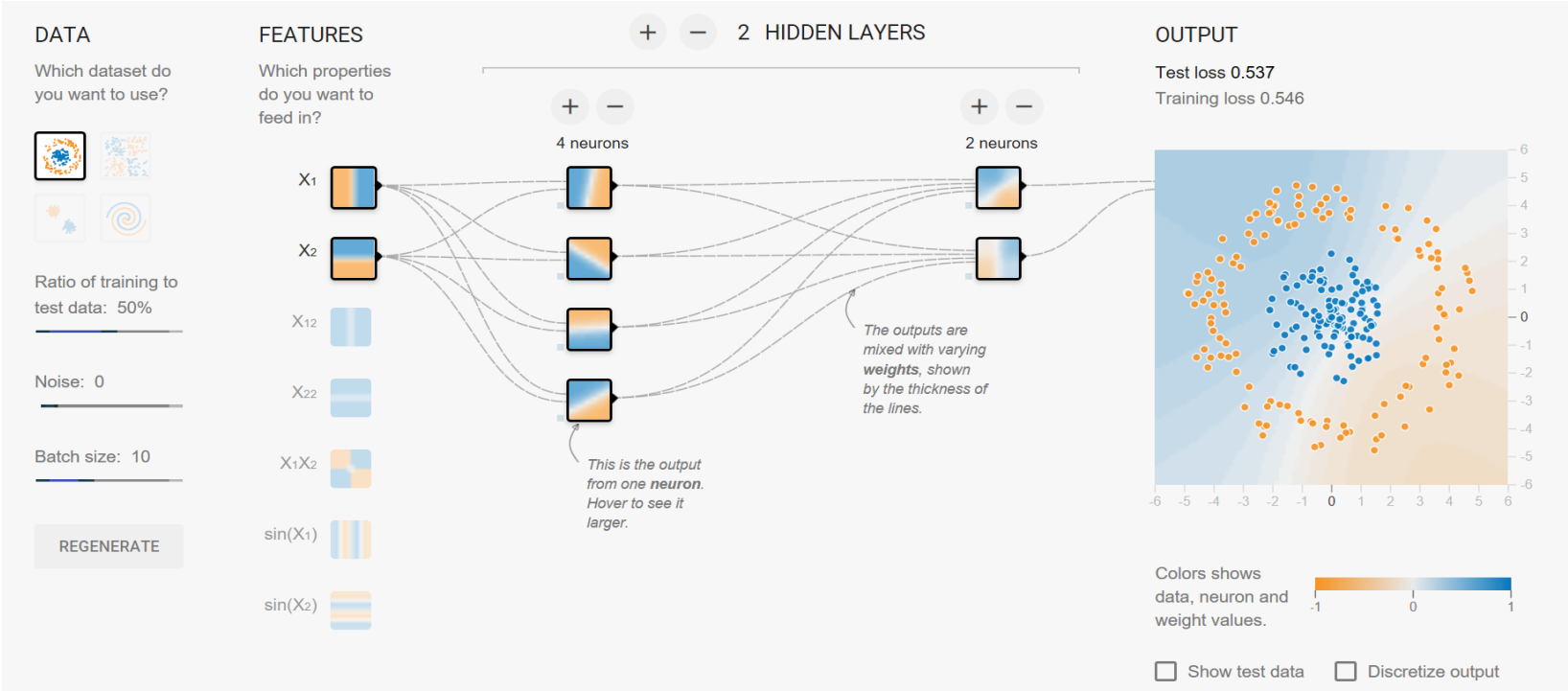
Higher learning rates

Loss decay will start to decline exponentially.

At some point the parameters will start to bounce around an optimal point, not being able to settle.



Neural Network Playground - Tinker with a Neural Network in your browser



http://playground.tensorflow.org

References

[12] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[13] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[14] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951.

[15] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[16] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[17] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations*, 2014.

[18] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

This lecture in one slide

Introduction and motivation for deep learning
Neural network conception
Optimization

Regularization
Parameter constraints
Batch methods
Dropout
Augmentation
Early stopping
Hyperparameter search

Neural Network Regularization

Optimization minimizes the error of a model on observed samples.

Machine Learning
Regularization

Neural Network Regularization

Optimization

Machine Learning prioritizes the model performance on unobserved data assuming *i.i.d* (independent and identically distributed). Hence, we are targeting generalization over the data distribution.

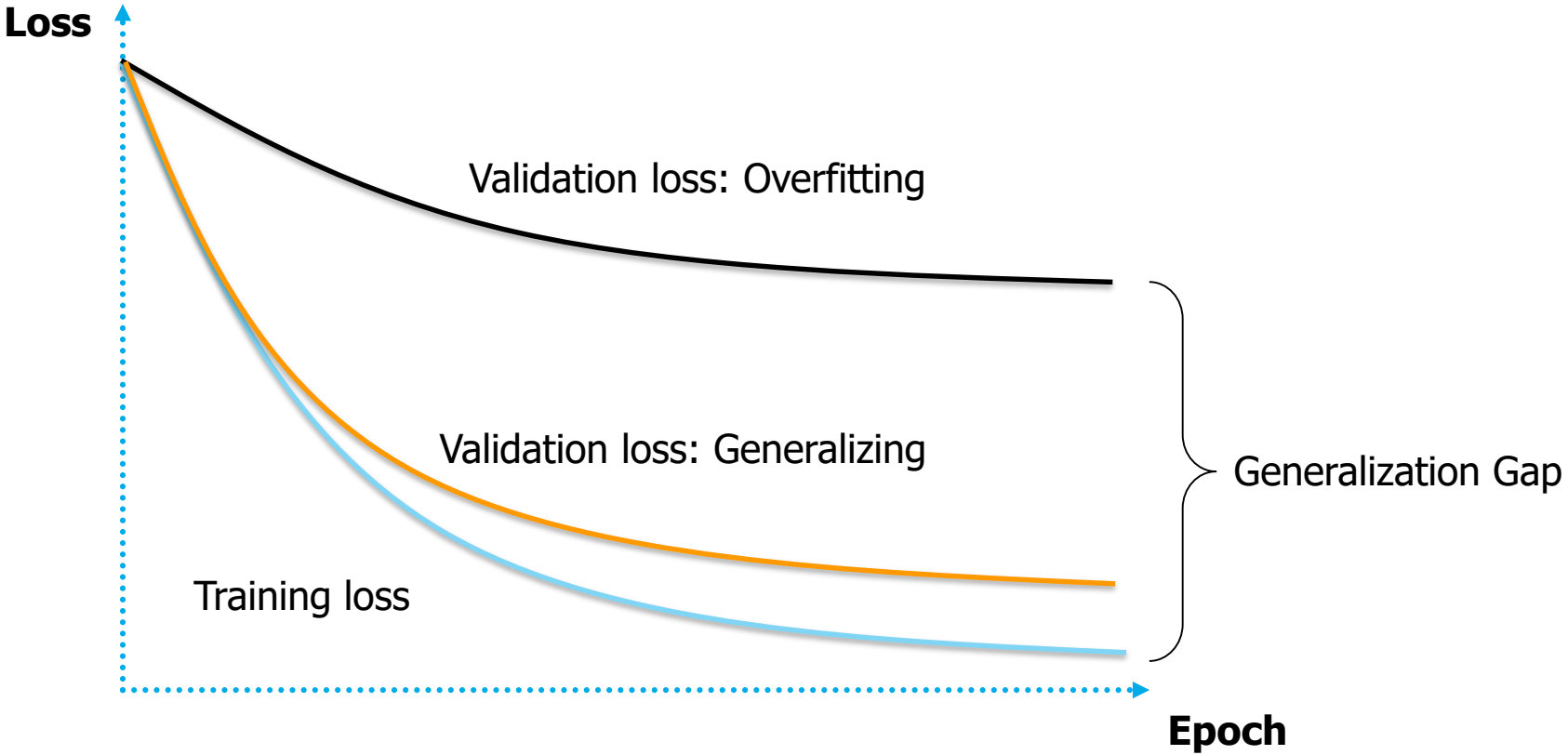
Regularization

Neural Network Regularization

Optimization
Machine Learning

Regularization techniques are used for bridging the generalization gap between the performance on observed (training data) and unobserved samples (validation and test data).

Neural Network Regularization – Bridging the generalization gap

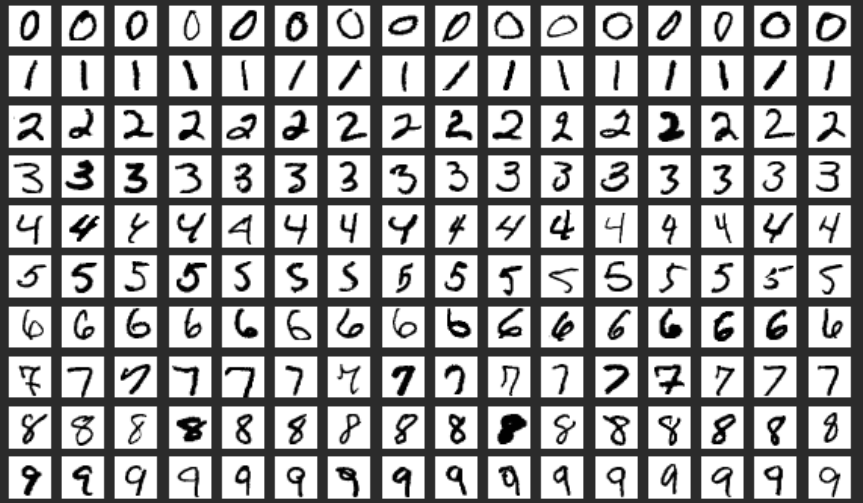


MNIST Dataset

The MNIST database, the ‘hello world!’ of machine learning.

Large database of handwritten digits. Grayscale images with dimension of 28x28 pixels.

60k training samples
10k testing images



https://en.wikipedia.org/wiki/MNIST_database#/media/File:MnistExamples.png

Parameter norm penalties

Adding a cost depending on the parameter values:

$$\hat{L}(\theta; \mathbf{X}, \mathbf{y}) = L(\theta; \mathbf{X}, \mathbf{y}) + \alpha\Omega(\theta).$$

$$\Omega(\theta) = \frac{1}{2}||\mathbf{w}||_2^2,$$

The most common is the L2 norm penalty, shifting the parameter values to be small (also known as weight decay).

Idea: Small changes in the input have small influence on the predicted output.

Parameter sharing

Parameter norm penalties

Parameter sharing

Force tying parameter values, due to prior knowledge:

$$\mathbf{w}^A \text{ to equal } \mathbf{w}^B.$$

- Translation invariance in images (Convolution Filters)
- Recurring similar inputs (Recurrent Neural Networks)

Neural Network Regularization – Batch methods

Why minibatches?

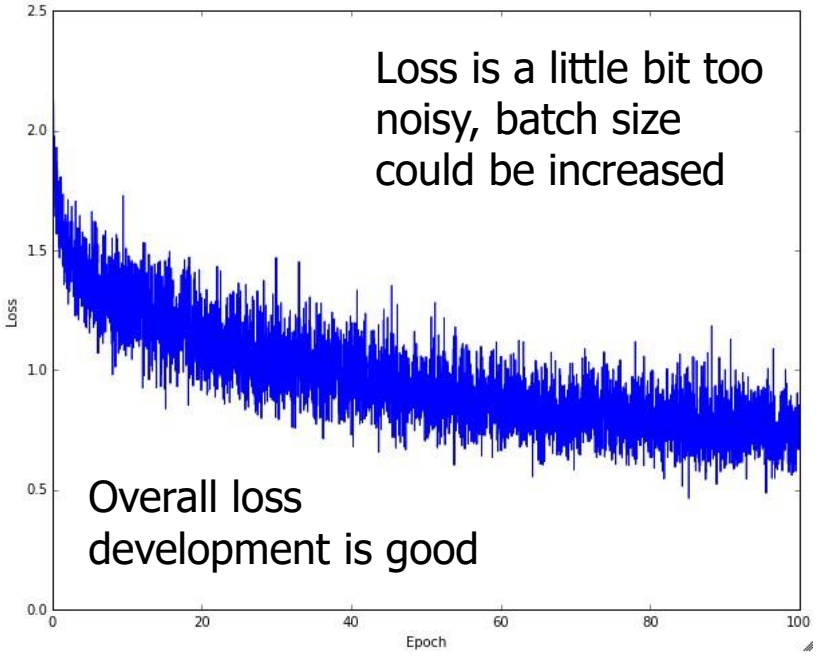
- Unbiased estimate of the gradient
- Computational effort

- Noise induced regularization for small batch sizes

Neural Network Regularization – Batch methods

Which batch size should I go for?

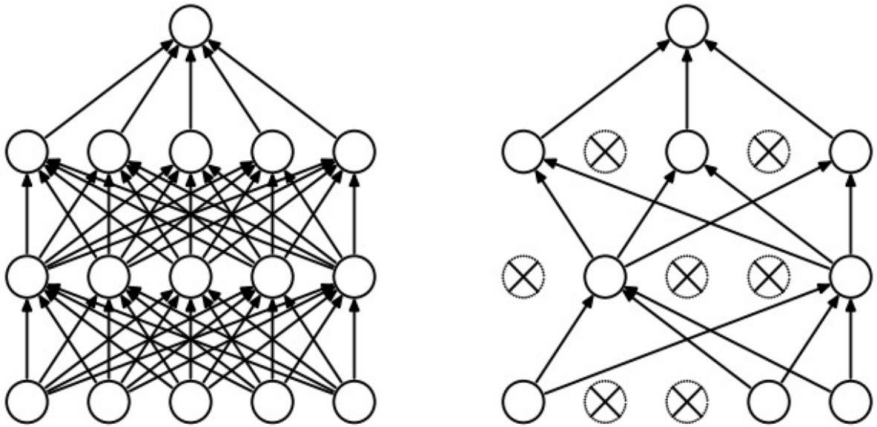
- Hardware restrictions set upper limit
- Power-of-two batch sizes match physical processor and improve runtime
- Loss band should be smooth, implying even gradient estimates.



http://cs231n.github.io/neural-networks-3/#baby

Neural Network Regularization – Dropout

Dropout keeps a neuron active with some probability (keep rate) during training, or setting it zero otherwise.



(a) Standard Neural Net (b) After applying dropout.

http://cs231n.github.io/neural-networks-2/#reg

Neural Network Regularization – Dropout

Dropout keeps a neuron active with some probability (keep rate) during training, or setting it zero otherwise.

For each weight update a different **sub neural network** is sampled from the standard neural network.

This implicitly trains an ensemble of networks, while inducing a regularization pressure, because **each parameter needs to function in all the ensembles**.

Neural Network Regularization – Dropout

Dropout keeps a neuron active with some probability (keep rate) during training, or setting it zero otherwise.

For each weight update a different sub neural network is sampled from the standard neural network.

This implicitly trains an ensemble of networks, while inducing a regularization pressure, because each parameter needs to function in all the ensembles.

During inference there is no dropout applied.

Neural Network Regularization – Augmentation

Generalization improves with an **increased dataset size**.
The number of iterations an individual samples is used for training

Neural Network Regularization – Augmentation

Generalization improves with an increased dataset size.
The number of iterations an individual samples is used for training

Increasing number of samples demands a great effort:

- Collecting data
- Preparing data
- Annotate data

Neural Network Regularization – Augmentation

Generalization improves with an increased dataset size.
The number of iterations an individual samples is used for training

Increasing number of samples demands a great effort

Data augmentation presents a useful solution
By transforming the existing training samples, while keeping the affiliated ground truth samples.

Neural Network Regularization – Augmentation

Generalization improves with an increased dataset size.
The number of iterations an individual samples is used for training

Increasing number of samples demands a great effort

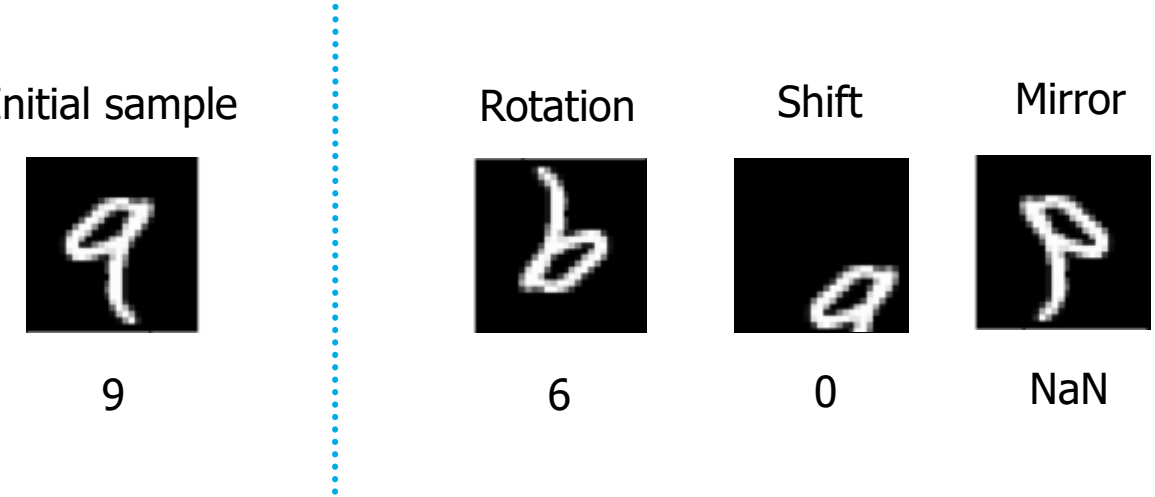
Data augmentation presents a useful solution

Examples of augmentation operations

- Rotation, Zoom, Cropping, Distortion and Translation
- Brightness and Saturation

Think before you augment:

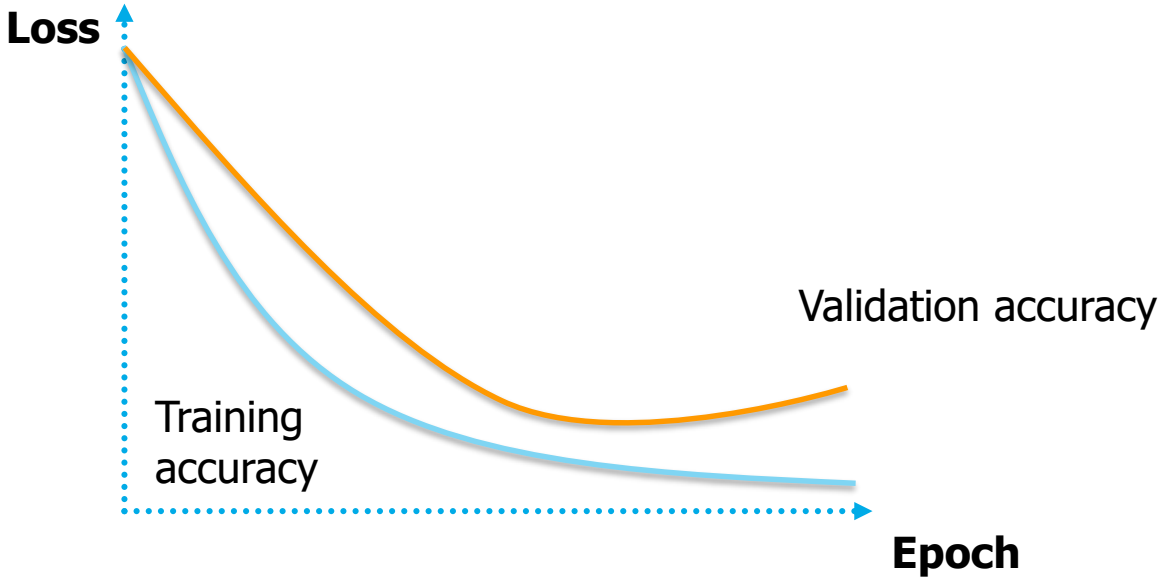
Prevent class switches and class breaks, know your data and your problem statement.



Note: Make sure to motivate the boundary conditions of your augmentation operations.

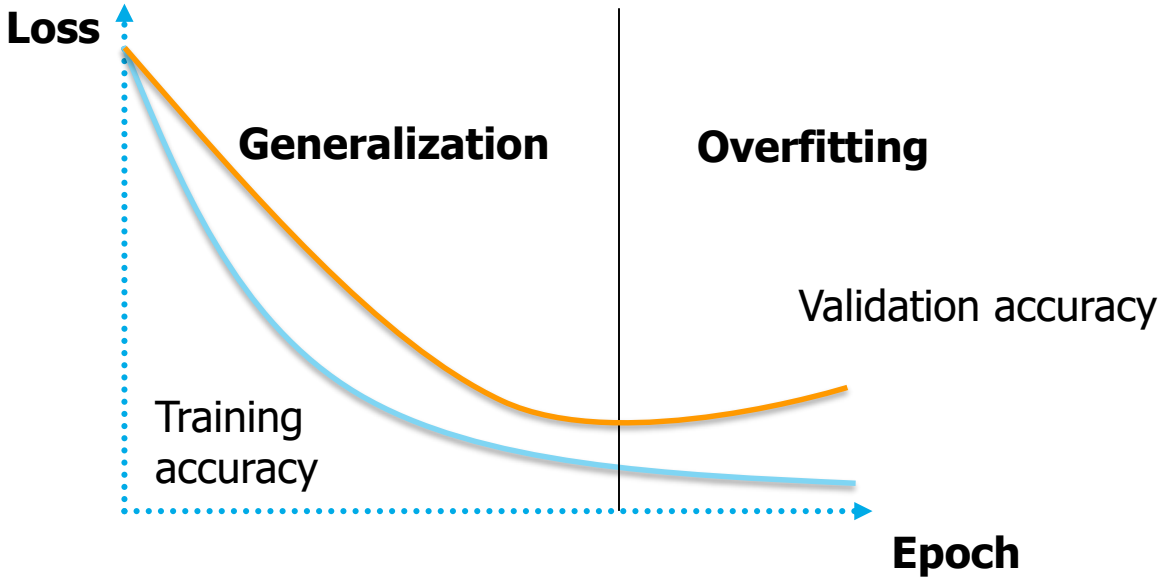
Neural Network Regularization – Early Stopping

When **training a model with large capacity** (large number of parameters), the training error steadily decreases.



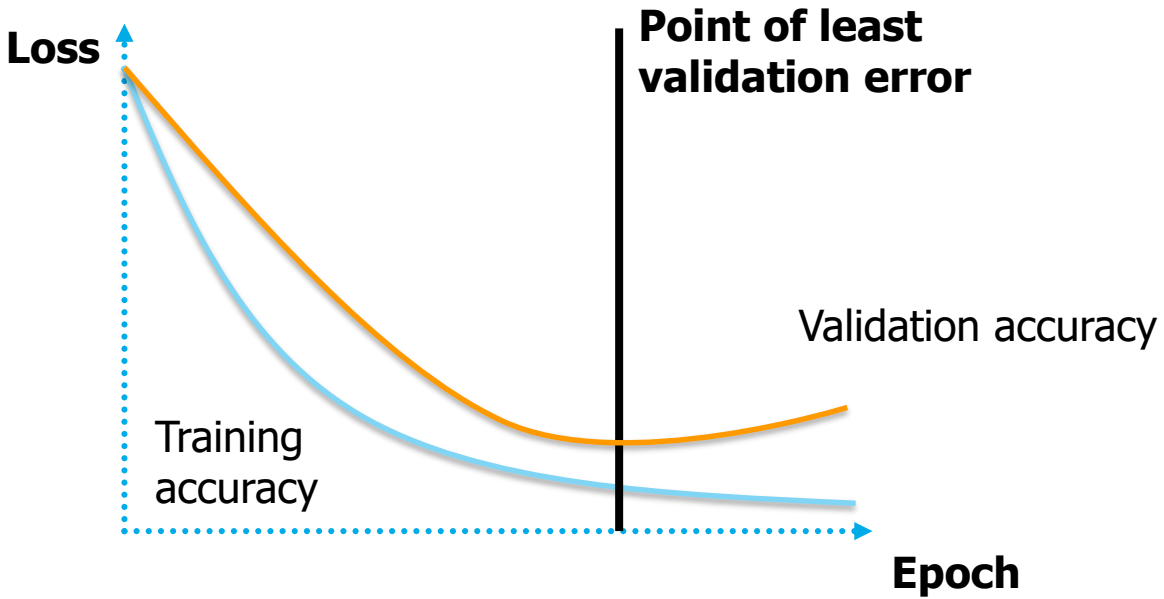
Neural Network Regularization – Early Stopping

At some point the model overfits on the training samples, leading to an **increased validation loss**.



Neural Network Regularization – Early Stopping

Early stopping is the process of finding the point of least validation error by monitoring the validation accuracy and then exiting the training process.



References

[19] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[20] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

[21] John L. Hennessy and David A. Patterson. *Computer Organization and Design (2Nd Ed.): The Hardware/Software Interface*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[22] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[23] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621, 2017.

[24] David Kriesel. A brief introduction on neural networks, 2015.

[25] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

Course overview

Deep Learning Foundations

Classification & Object Detection and Transfer Learning

Segmentation Networks

Deep Reinforcement Learning

Generative Adversarial Networks

Recurrent Neural Networks

This lecture in one slide

Classification and Object Detection with neural networks

Problems & Datasets

Convolutional Neural Networks

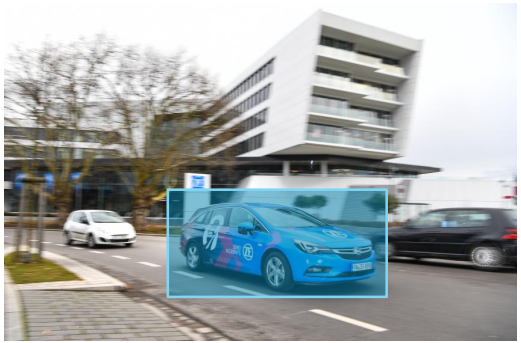
Application to Object Detection

Transfer Learning with neural networks

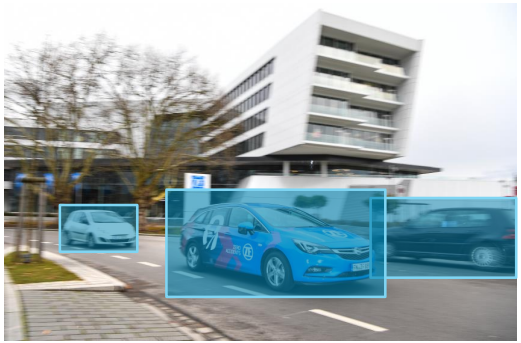
Introduction – Classification & Object Detection a problem statement

Classification

Classification
+ Localization



Object Detection



Neural Network Object Detection - Datasets

Common Objects in Context

COCO-Detection has 200k images with bounding boxes or pixel-wise labels

Annotations

80 object categories (person, elephant, etc.), as well as captions.

Number of samples

200000 bounding box level annotations

https://github.com/nightrome/cocostuff

Neural Network Object Detection - Datasets

PASCAL Visual Object Classes

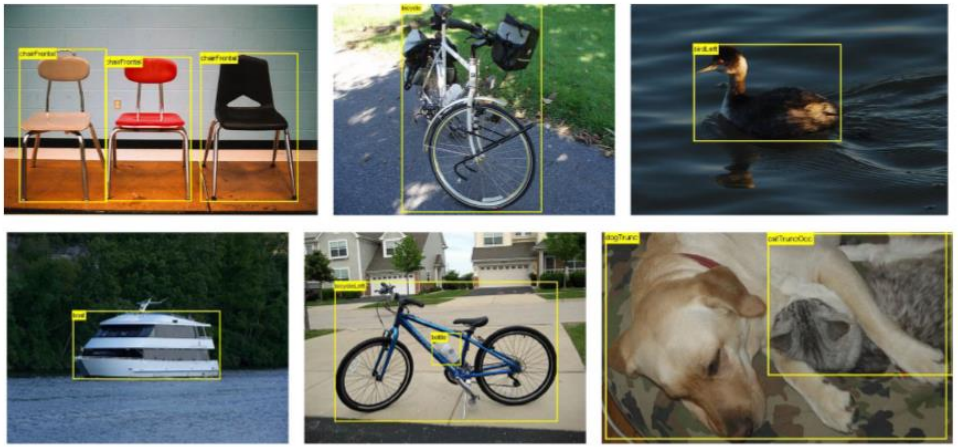
For each of twenty object classes predict the presence/absence of at least one object of that class in a test image.

Annotations

20 object classes (Person, Bicycle, etc.)

Number of samples

11540 bounding box level annotations



http://host.robots.ox.ac.uk/pascal/VOC/pubs/everingham15.pdf

Neural Network Object Detection - Datasets

KITTI

We take advantage of our autonomous driving platform Annieway to develop novel challenging real-world computer vision benchmarks.

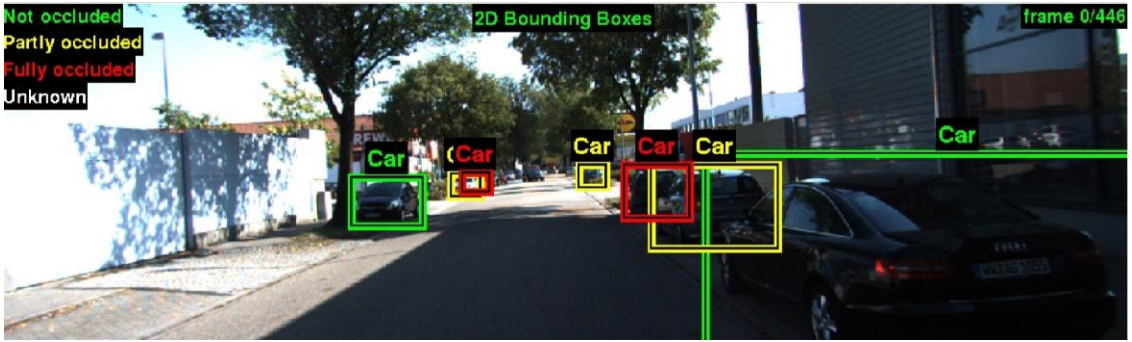
Annotations

2D bounding box annotations with classes

Number of samples

7481 training images and

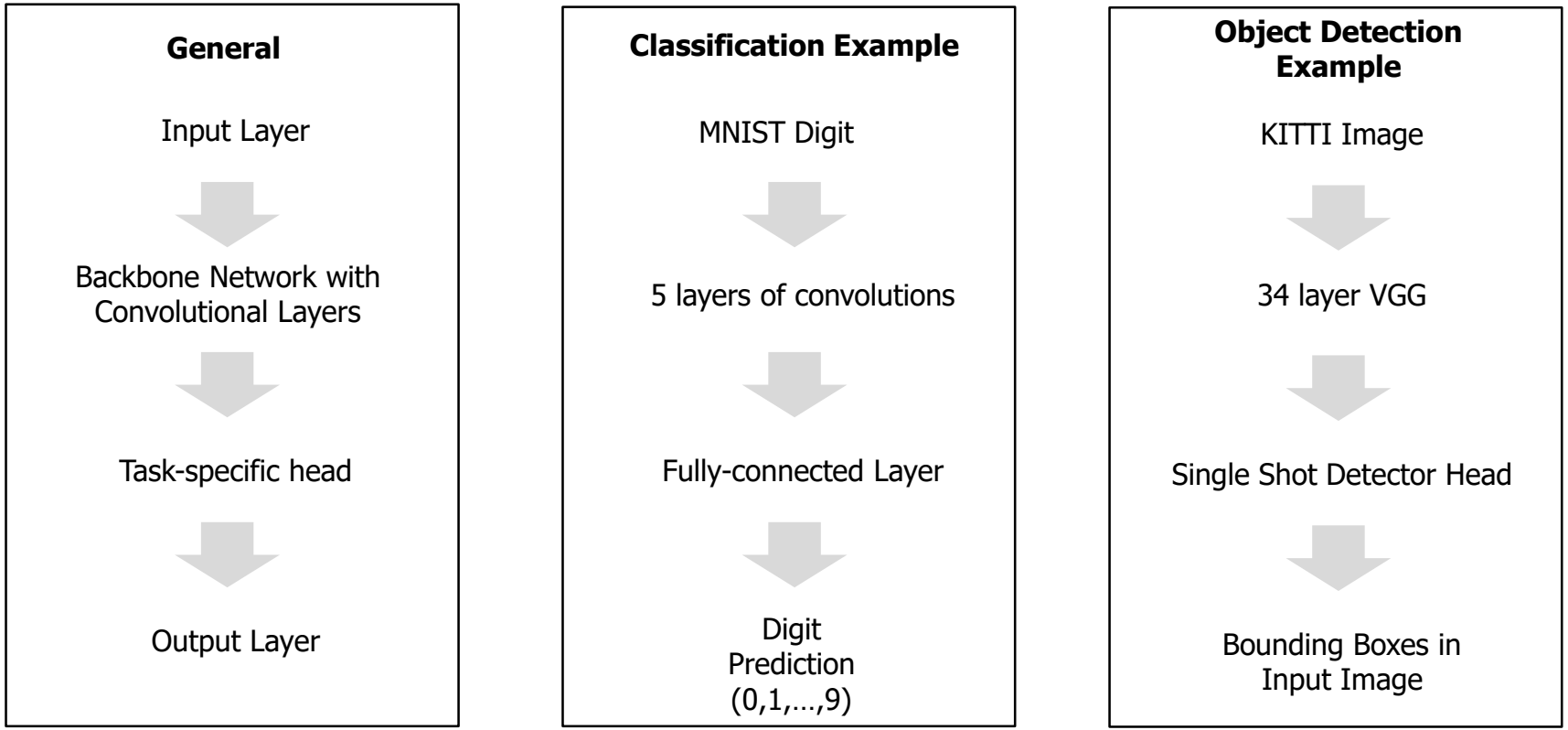
7518 test images



http://www.cvlibs.net/publications/Geiger2013IJRR.pdf

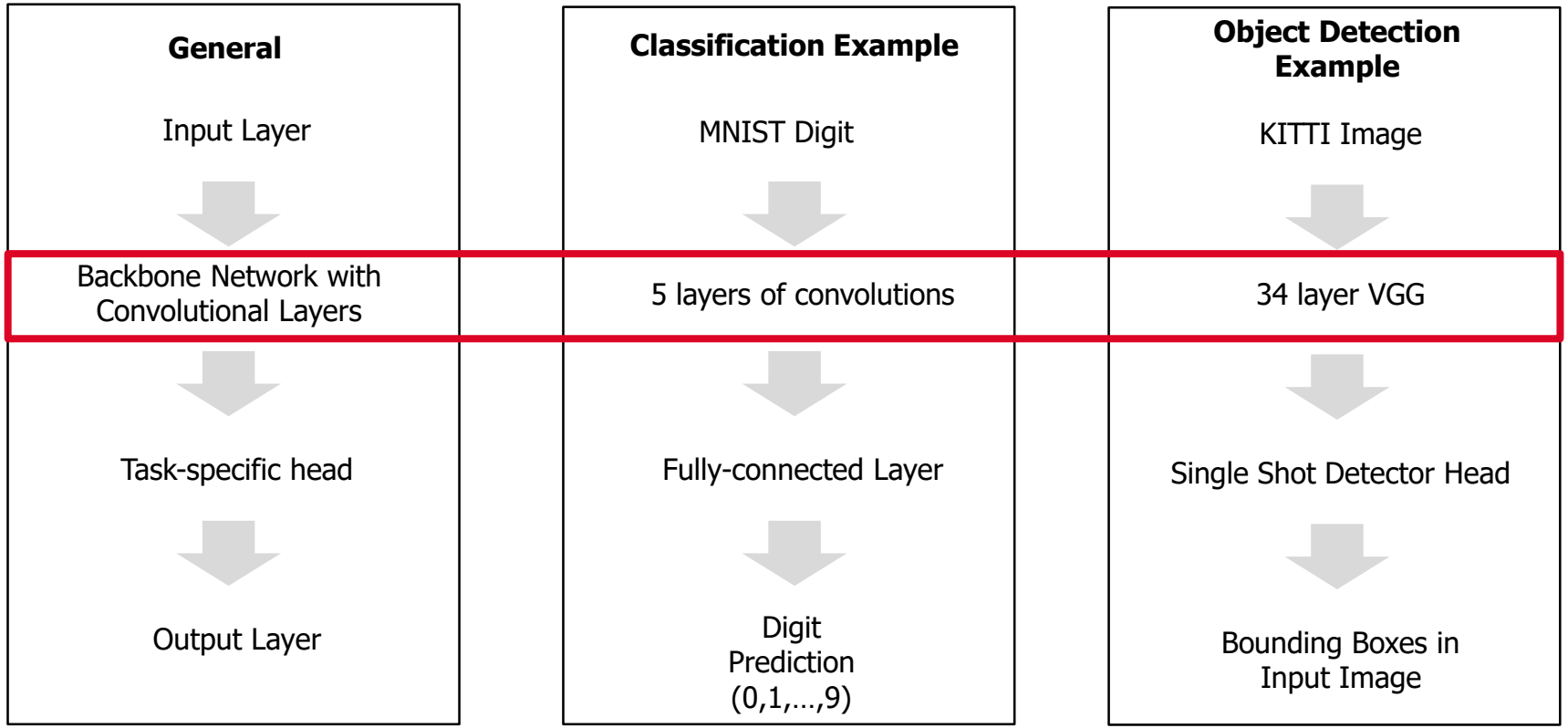
Convolutional Neural Networks

Typical structure of a neural network in Computer Vision:



Convolutional Neural Networks

Typical structure of a neural network in Computer Vision:



Convolutional Neural Networks – The convolution

In a convolutional layer several **kernels** with a predefined **filter size** are applied to a feature map:

Forward Pass

Convolutional Kernel

Kernel Size
Height Width Depth
2 x 2 x 1

$$\begin{matrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{matrix}$$

Amount of Conv. Kernels
1

Depth of Feature Map
1

$$\begin{matrix} x_{11} & x_{21} & x_{31} & x_{41} \\ x_{12} & x_{22} & x_{32} & x_{42} \\ x_{13} & x_{23} & x_{33} & x_{43} \\ x_{14} & x_{24} & x_{34} & x_{44} \end{matrix}$$

Feature Map Size
Height Width Depth
4 x 4 x 1

Feature Map Size
Height Width Depth
3 x 3 x 1

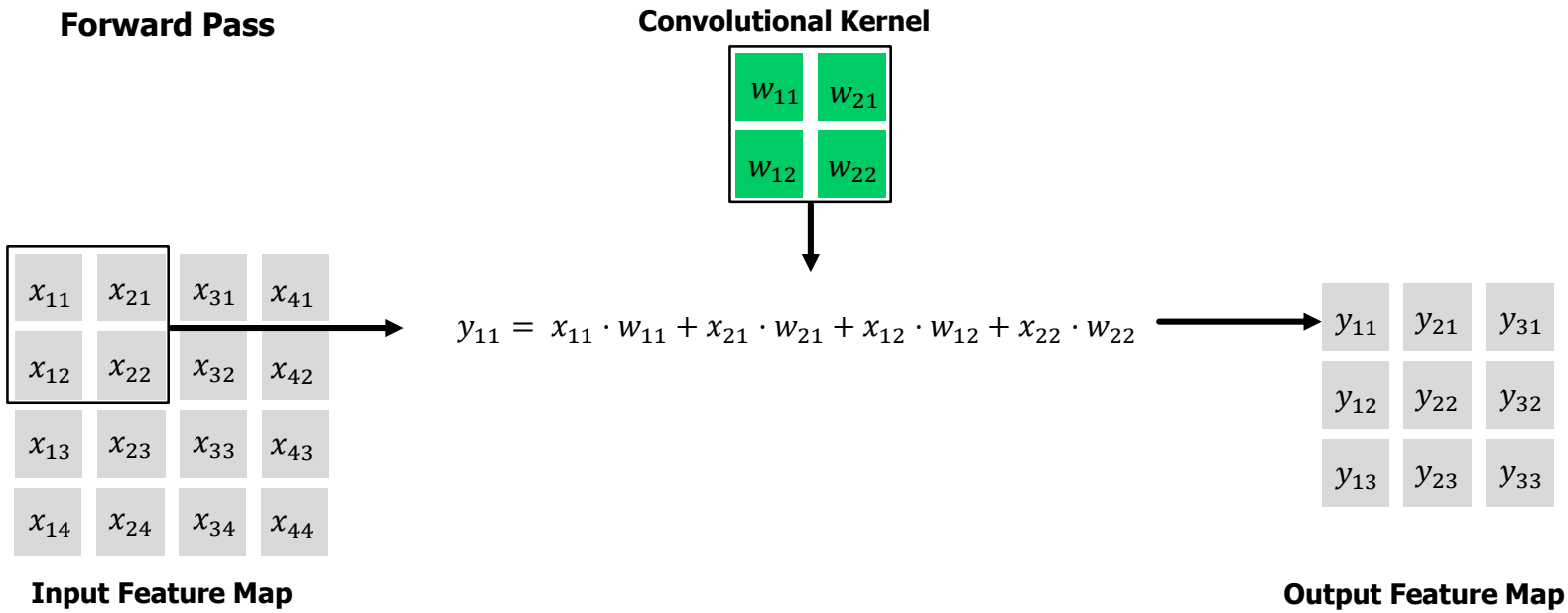
$$\begin{matrix} y_{11} & y_{21} & y_{31} \\ y_{12} & y_{22} & y_{32} \\ y_{13} & y_{23} & y_{33} \end{matrix}$$

Input Feature Map

Output Feature Map

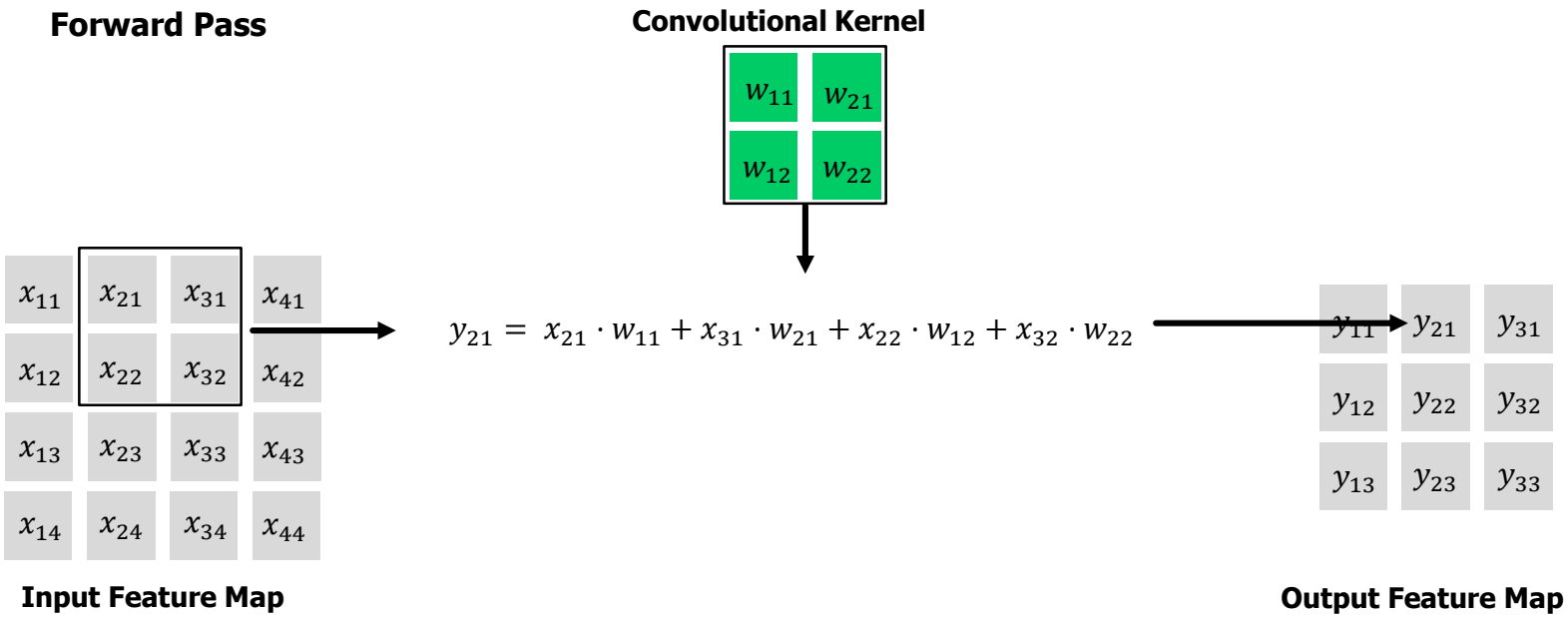
Convolutional Neural Networks – The convolution

In a convolutional layer several **kernels** with a predefined **filter size** are applied to a feature map:



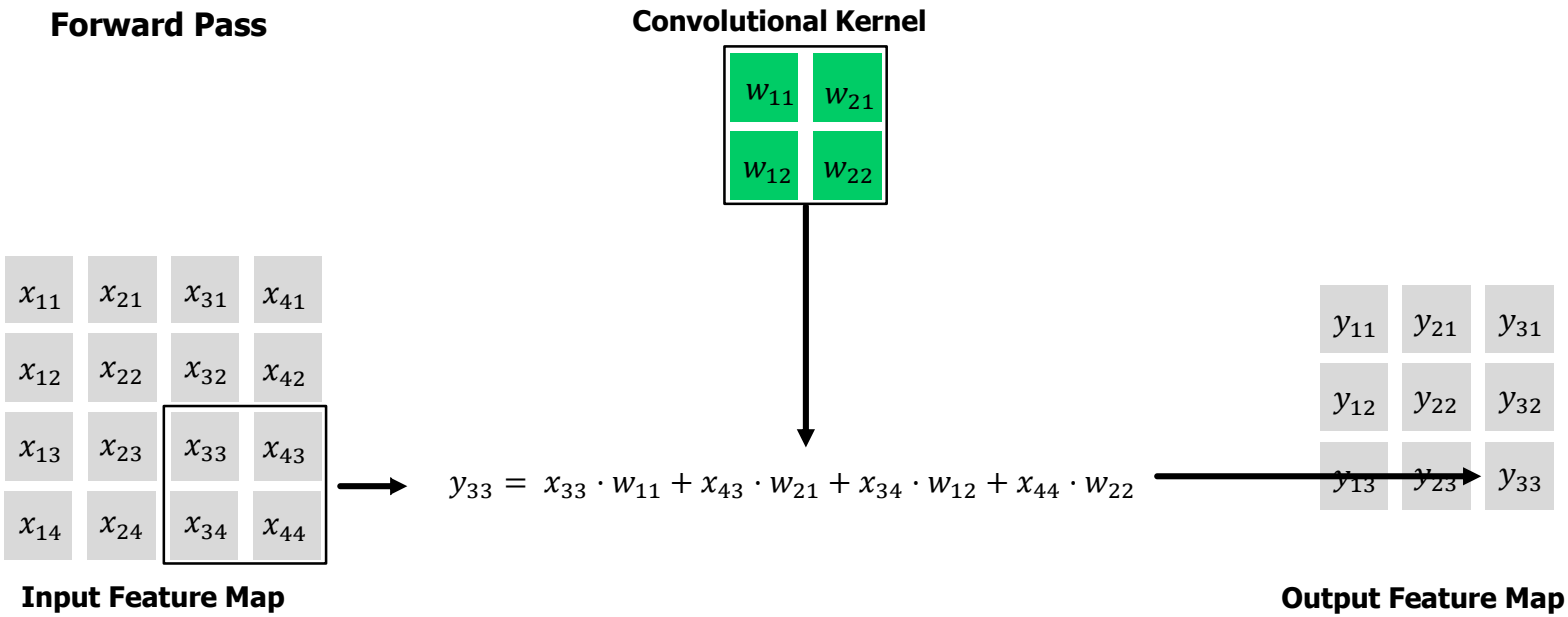
Convolutional Neural Networks – The convolution

In a convolutional layer several **kernels** with a predefined **filter size** are applied to a feature map:



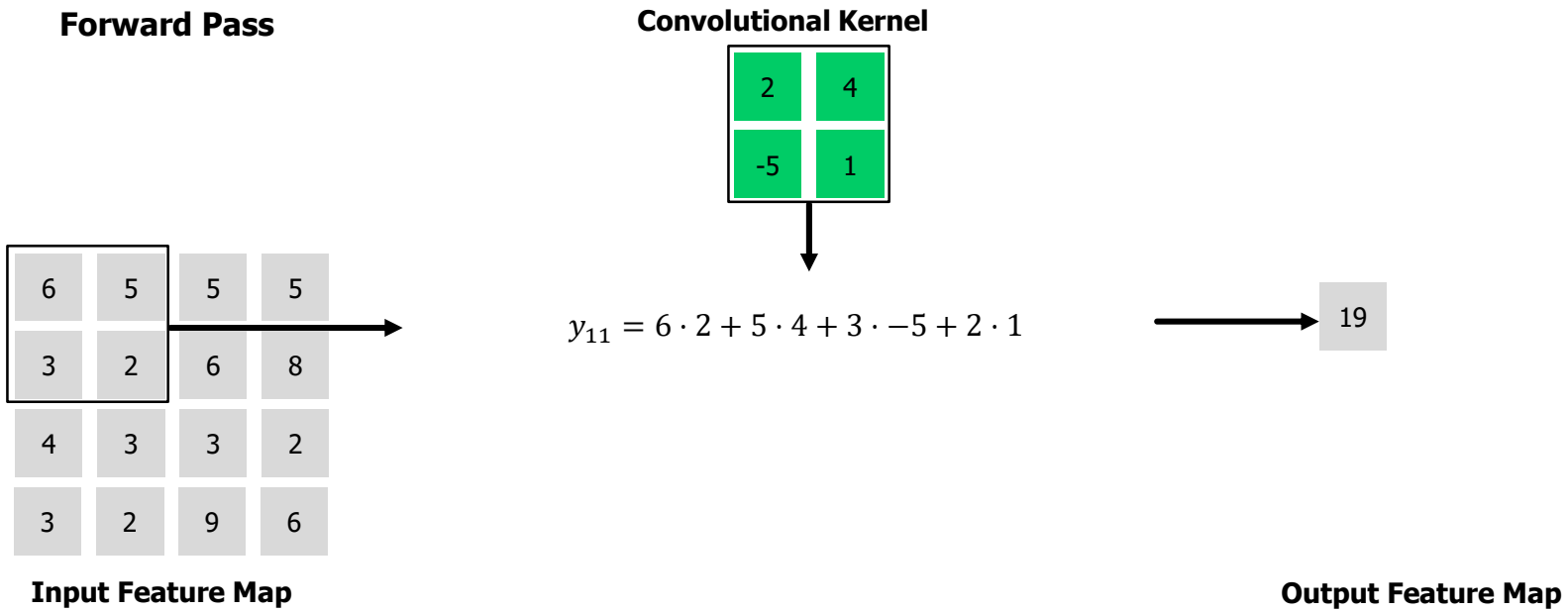
Convolutional Neural Networks – The convolution

In a convolutional layer several **kernels** with a predefined **filter size** are applied to a feature map:



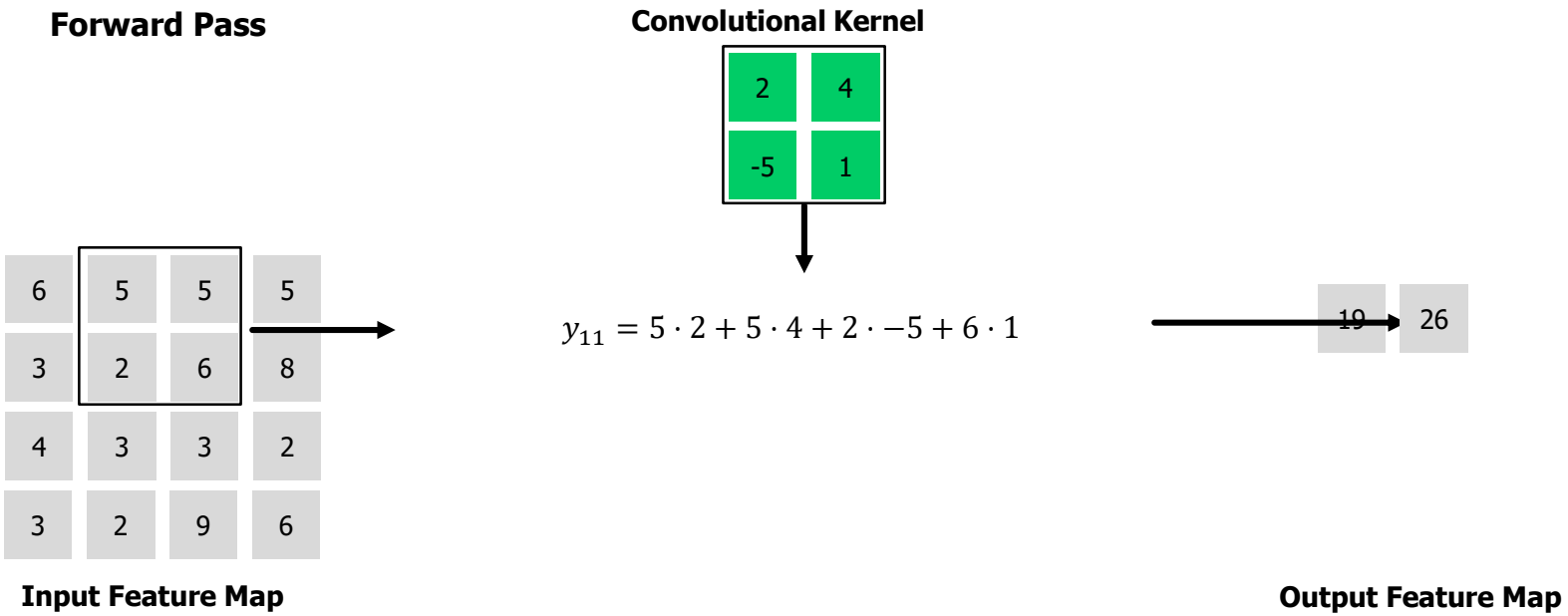
Convolutional Neural Networks – The convolution

In a convolutional layer several **kernels** with a predefined **filter size** are applied to a feature map:



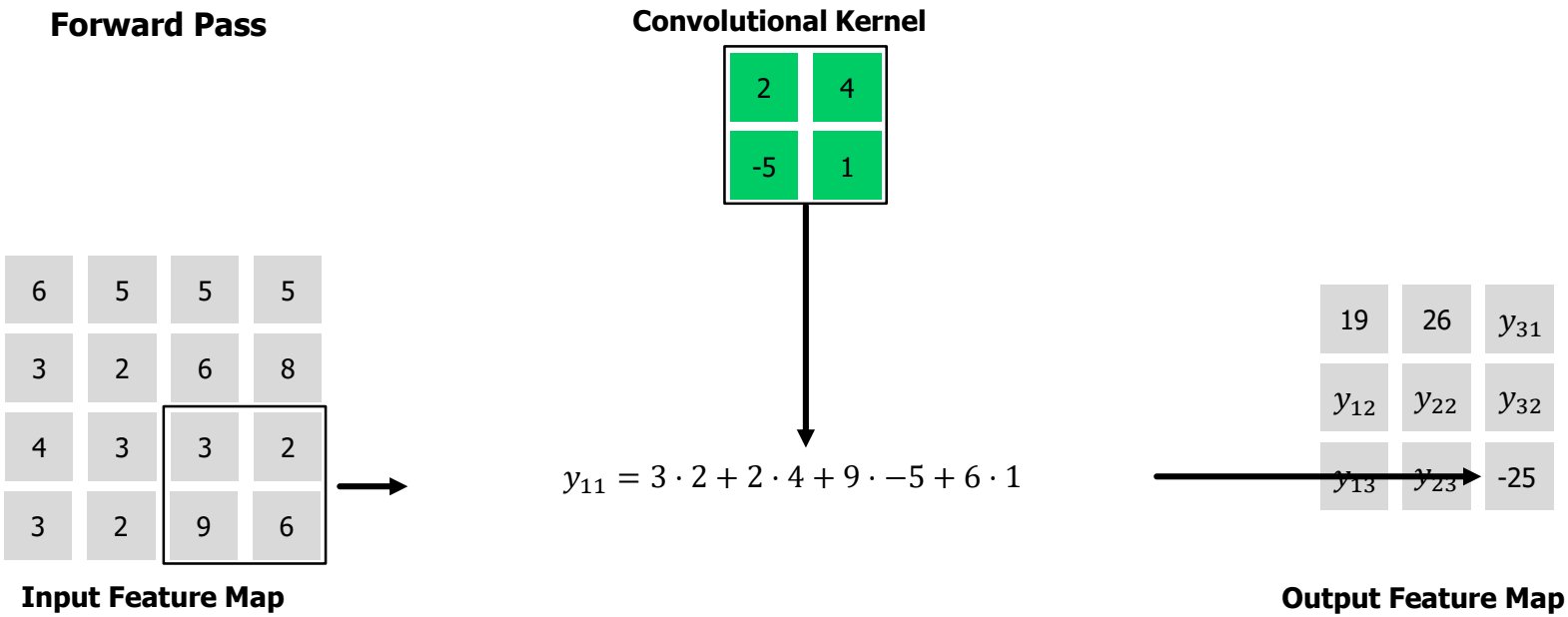
Convolutional Neural Networks – The convolution

In a convolutional layer several **kernels** with a predefined **filter size** are applied to a feature map:



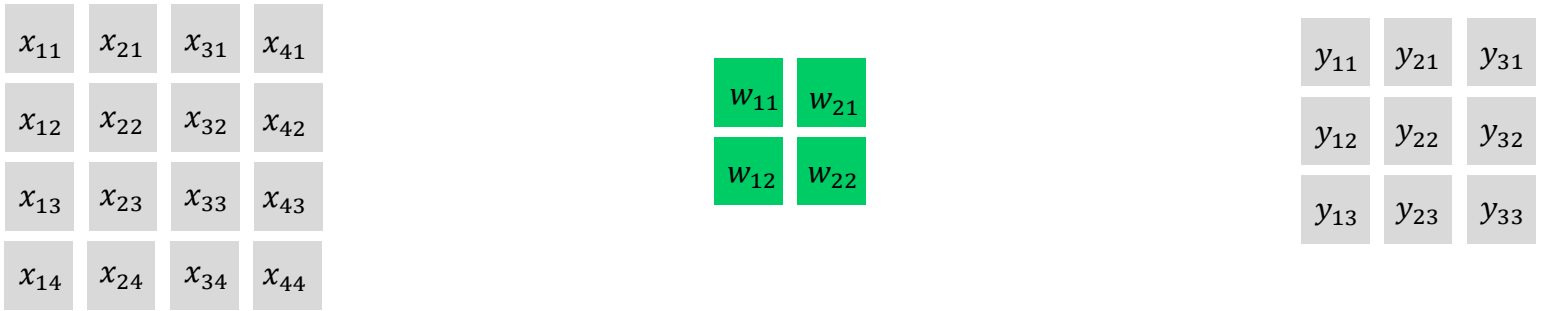
Convolutional Neural Networks – The convolution

In a convolutional layer several **kernels** with a predefined **filter size** are applied to a feature map:



Convolutional Neural Networks – Convolutions

In a convolutional layer several **kernels** with a predefined **filter size** are applied to a feature map:



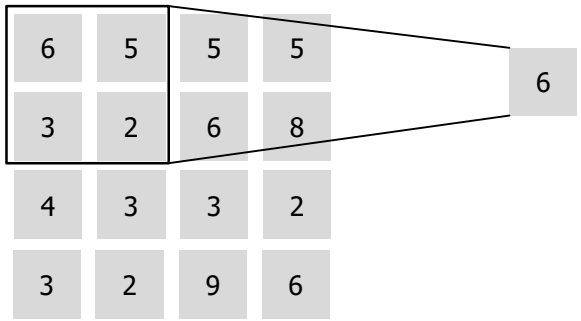
Convolutional layers introduce an **inductive bias** to our neural network:

- In images (or similar measurements) there is **locality in features**. Pixels close to each other are related.
- Those local features are same no matter where the feature is in the image (**weight sharing**).

Convolutional Neural Networks – Pooling

Pooling Layers are used to reduce the size of a feature map.

Max Pooling (filter 2x2, stride of 2)



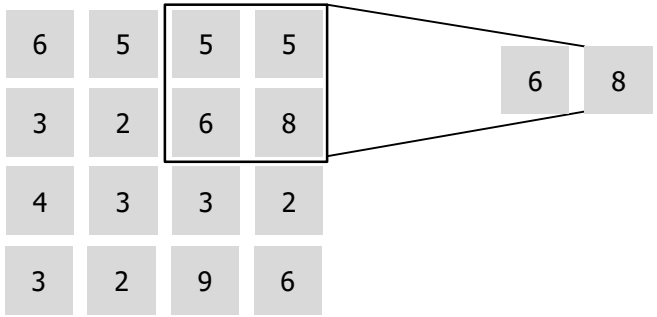
No trainable weights

Find max value in current window

Convolutional Neural Networks – Pooling

Pooling Layers are used to reduce the size of a feature map.

Max Pooling (filter 2x2, stride of 2)



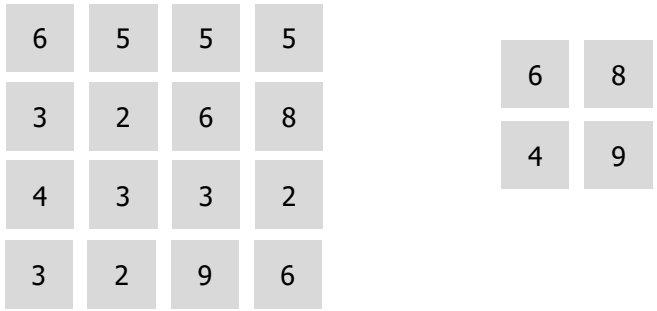
No trainable weights

Find max value in current window

Convolutional Neural Networks – Pooling

Pooling Layers are used to reduce the size of a feature map.

Max Pooling (filter 2x2, stride of 2)



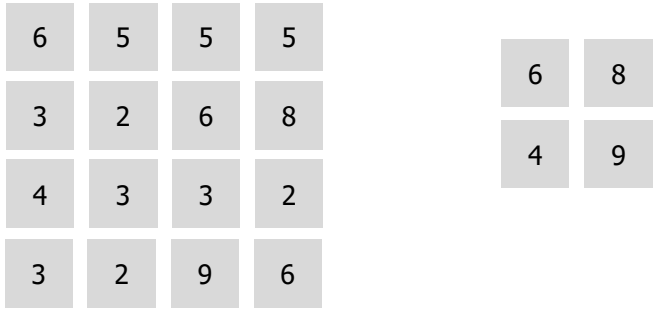
No trainable weights

Find max value in current window

Convolutional Neural Networks – Pooling

Pooling Layers are used to reduce the size of a feature map.

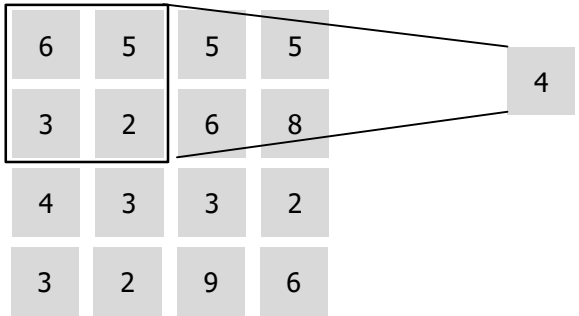
Max Pooling (filter 2x2, stride of 2)



No trainable weights

Find max value in current window

Average Pooling (filter 2x2, stride of 2)

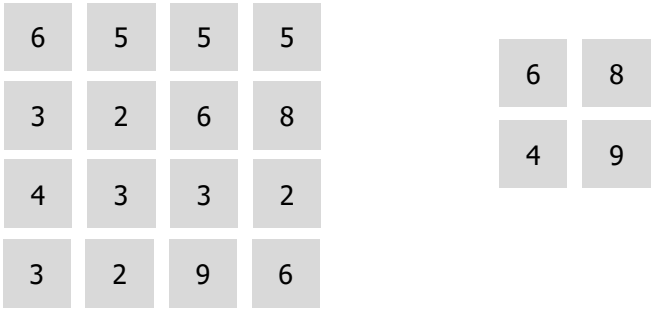


Find average value of current window

Convolutional Neural Networks – Pooling

Pooling Layers are used to reduce the size of a feature map.

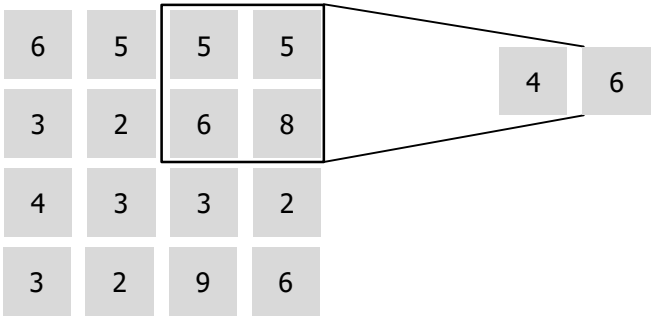
Max Pooling (filter 2x2, stride of 2)



No trainable weights

Find max value in current window

Average Pooling (filter 2x2, stride of 2)

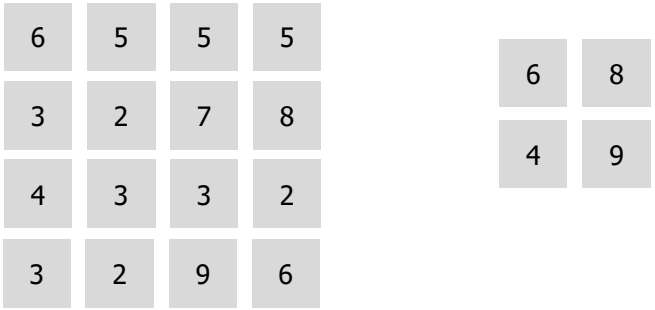


Find average value of current window

Convolutional Neural Networks – Pooling

Pooling Layers are used to reduce the size of a feature map.

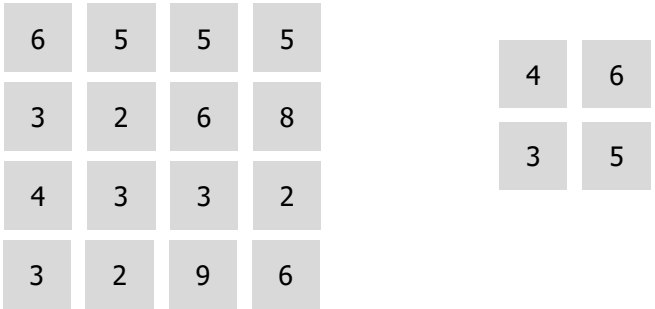
Max Pooling (filter 2x2, stride of 2)



No trainable weights

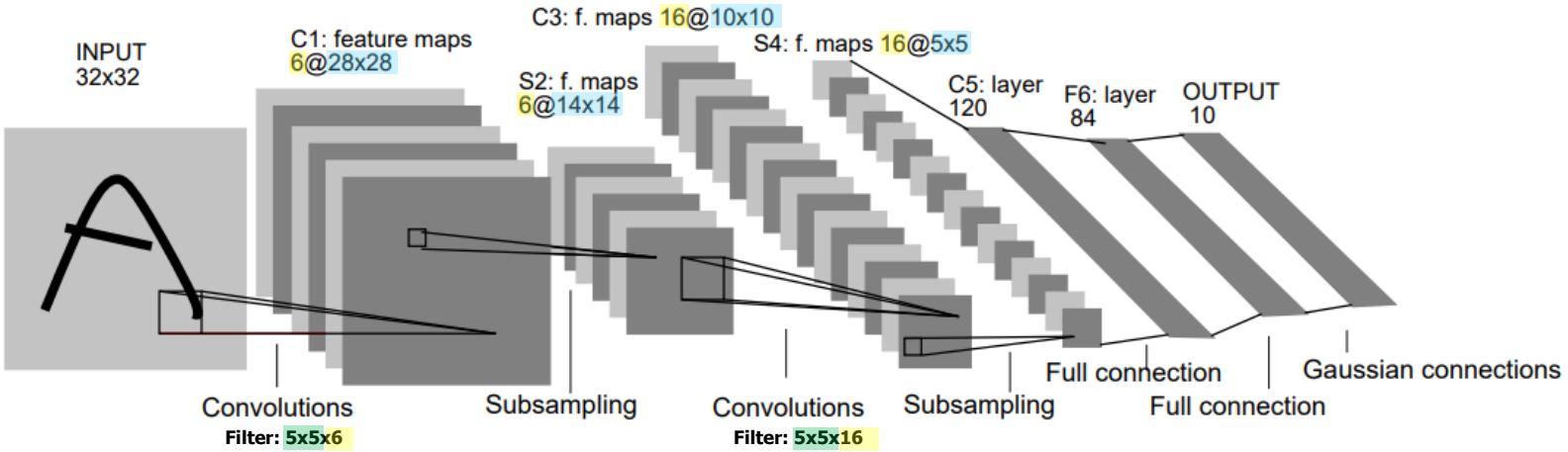
Find max value in current window

Average Pooling (filter 2x2, stride of 2)



Find average value of current window

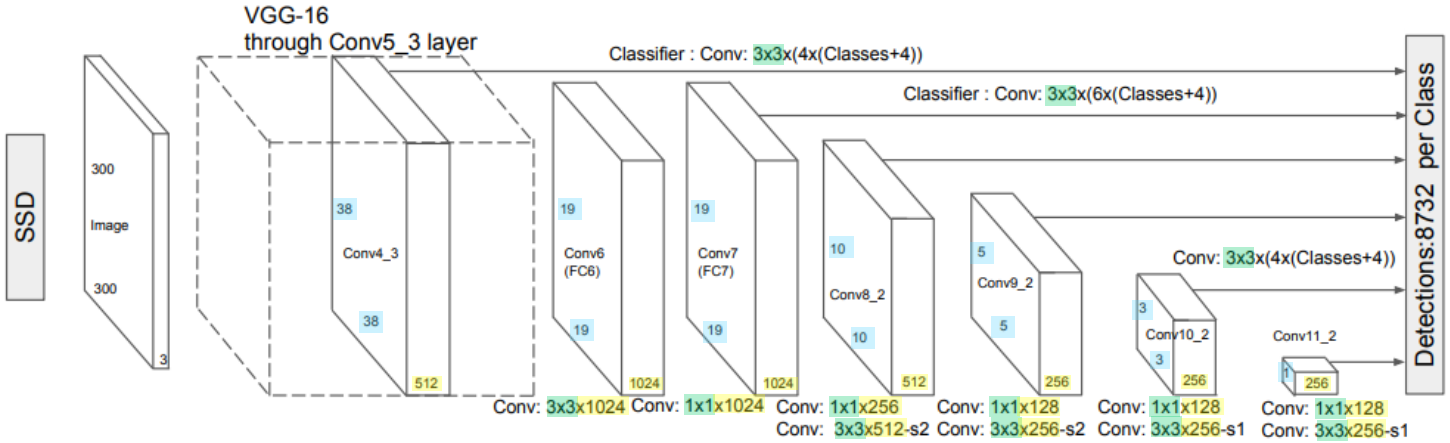
Neural Network Object Detection – Digit Classification with LeNet-5



Kernel Size
Amount of Kernels in Layer = Depth of subsequent Feature Map
Width & Height of Feature Map

LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.(http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf)

Neural Network Object Detection – Single Shot Detector with VGG-16 Backbone



Kernel Size
Amount of Kernels in Layer = Depth of subsequent Feature Map
Width & Height of Feature Map

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). SSD: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham. (*https://arxiv.org/pdf/1512.02325.pdf*)

Neural Network Object Detection - Basic structure

SSD: Single Shot MultiBox Detector

Discretization of the input image into a SxS grid of different sizes.

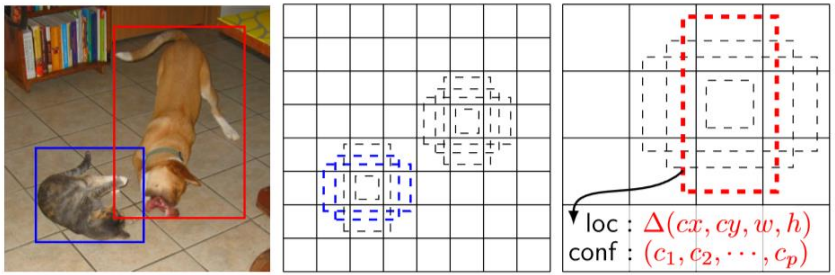
Fully Convolutional Network predicts class scores and box offsets for given default bounding boxes per size.

Final detections are chosen with non-maximum suppression.

SSD: Single Shot MultiBox Detector

Wei Liu[1], Dragomir Anguelov[2], Dumitru Erhan[3], Christian Szegedy[3],
Scott Reed[4], Cheng-Yang Fu[1], Alexander C. Berg[1]

[1]UNC Chapel Hill [2]Zoox Inc. [3]Google Inc. [4]University of Michigan, Ann-Arbor
wliu@cs.unc.edu, [2]drago@zoox.com, [3]{dumitru,szegedy}@google.com,
[4]reedscot@umich.edu, [1]{cyfu,aberg}@cs.unc.edu



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

https://arxiv.org/abs/1512.02325

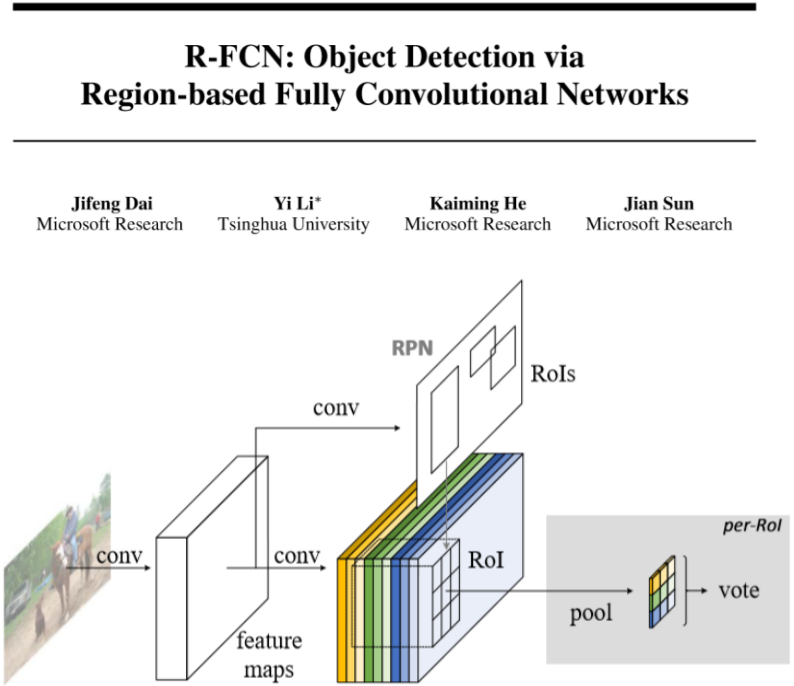
Neural Network Object Detection - Basic structure

RFCN: Region-based Fully Convolutional Network

Two stage object detection.

First stage is Fully Convolutional Network, such as ResNet-101 for region proposals

Second stage does classification on the max-pooled proposed regions



https://arxiv.org/abs/1605.06409

TensorFlow Model Zoo

Collection of detection models pre-trained on different object detection datasets.

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes
faster_rcnn_nas_lowproposals_coco	540		Boxes

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

References

[2] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[3] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *14th european conference on computer vision*, pages 21–37, 2016.

[4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016.

[5] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[6] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun 2010.

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM.

[9] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun 2010.

References

[1] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

This lecture in one slide

Transfer Learning with neural networks

Fine Tuning
Freezing Weights

Transfer Learning with Neural Networks

The intuition behind transfer learning is that if a model trained on a large and general enough dataset, this model will effectively serve as a **generic model of the visual world**.

Transfer Learning with Neural Networks

The intuition behind transfer learning is that if a model trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world.

You can then take advantage of these learned features without having to start from scratch training a large model on a large dataset.

Transfer Learning with Neural Networks - Fine Tuning

A **Pre-trained model** is a model that is trained on the source domain for a source task.

Pre-trained model

Car

Transfer Learning with Neural Networks - Fine Tuning

A **Pre-trained model** is a model that is trained on source domain for a source task.

Think **Object Detection Model Zoo**

Pre-trained model

Car

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

Transfer Learning with Neural Networks - Fine Tuning

A **Pre-trained model** is a model that is trained on the source domain for a source task.

The source domain is usually a **large dataset**.

Pre-trained model

Car

Transfer Learning with Neural Networks - Fine Tuning

A **Pre-trained model** is a model that is trained on the source domain for a source task.

The source domain is usually a **large dataset**.

Think **COCO**, **ImageNet**, **PASCAL VOC**

Pre-trained model

Car

Transfer Learning with Neural Networks - Fine Tuning

You either use a **pre-trained model as it is**.

Pre-trained model

Car

Transfer Learning with Neural Networks - Fine Tuning

You either use a pre-trained model as it is.

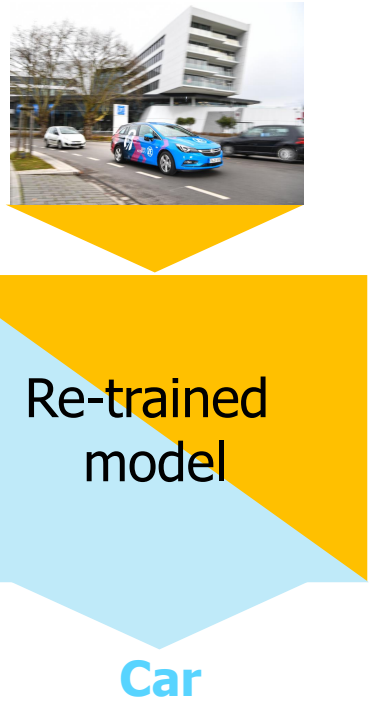
- No additional training
- Usually **performance low**

Pre-trained model

Car

Transfer Learning with Neural Networks - Fine Tuning

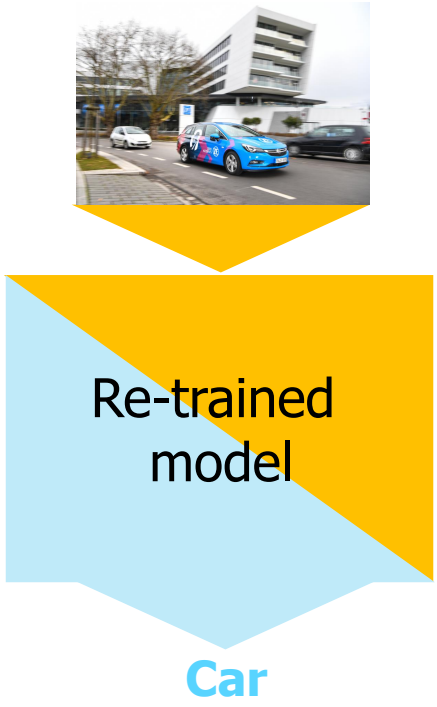
You can **fine-tune** a **pre-trained model** by retraining on additional data.



Transfer Learning with Neural Networks - Fine Tuning

You can fine-tune a pre-trained model by retraining on additional data.

- Additional work to set up retraining pipeline
- Usually performance drops in the source domain
- **Performance improvement** in the target domain



Transfer Learning with Neural Networks - Feature Extraction

You can **repurpose** the **pre-trained layers** as feature extraction layers for low level features.

And adding layers that learn the required high level features on the new data.

Pre-trained layers

Newly-trained layers

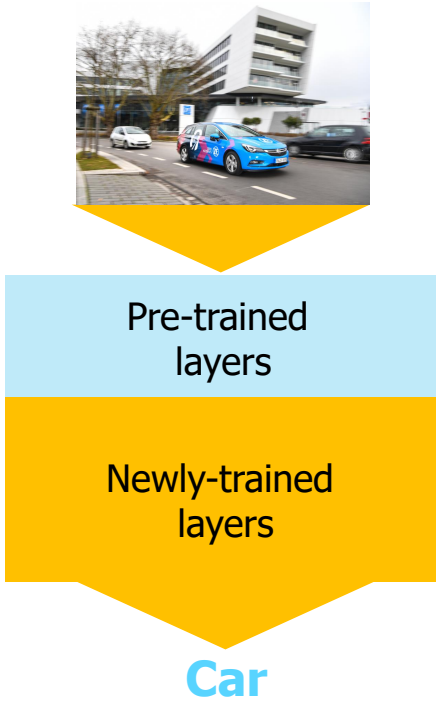
Car

Transfer Learning with Neural Networks - Feature Extraction

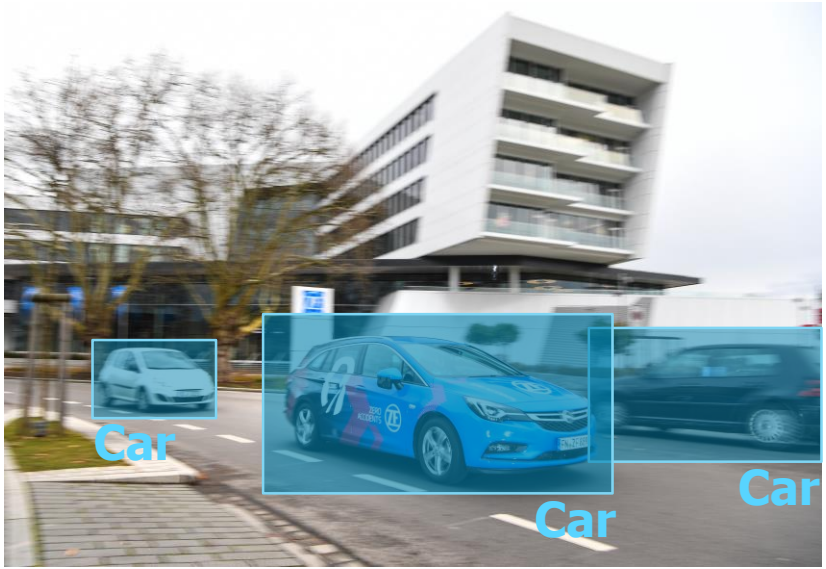
You can repurpose the pre-trained layers as feature extraction layers for low level features.

And adding layers that learn the required high level features on the new data.

- Additional work to assemble layer structure
- **Regularization** effects
- **Performance improvement** in the target domain
- **Output** layer becomes **adjustable** (such as for adding a class)

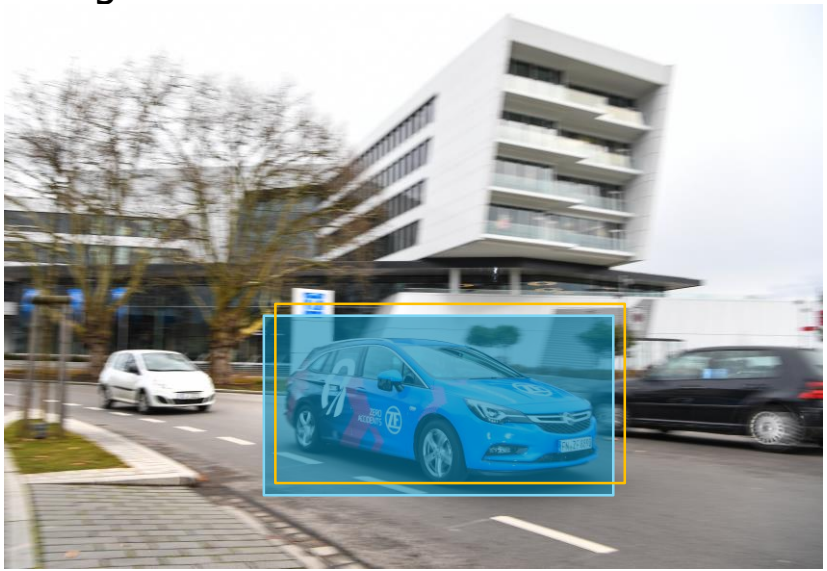


How do we evaluate our Object Detectors?



So how do we evaluate our Object Detectors?

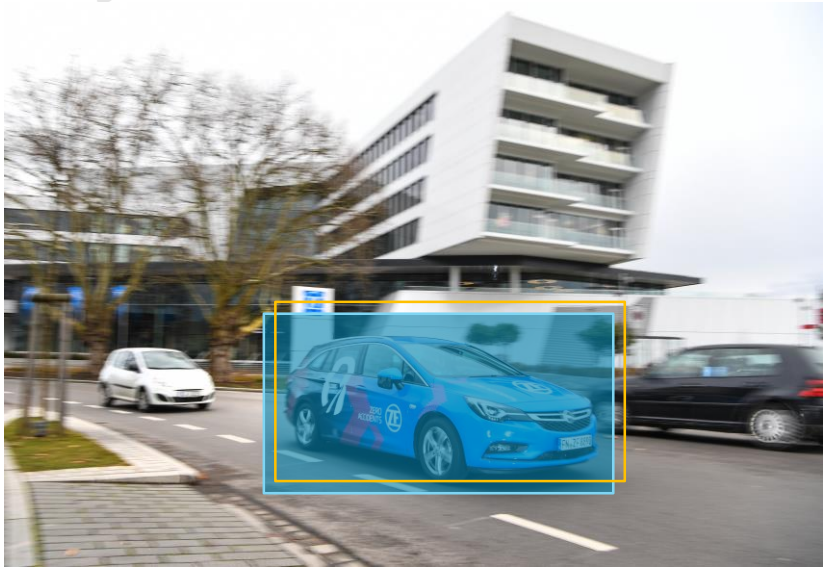
Define when a detection is correct, a true positive and when it is not.
A prediction will usually not overlap perfectly with the ground truth.



So how do we evaluate our Object Detectors?

Define when a detection is correct, a true positive and when it is not.
A prediction will usually not overlap perfectly with the ground truth.

Filter predictions based on
the object detectors **confidence**



So how do we evaluate our Object Detectors?

Define when a detection is correct, a true positive and when it is not.
A prediction will usually not overlap perfectly with the ground truth.

Assign true positives based on **Intersection over Union** with the ground truth

$$IoU(A,B) = \frac{A \cap B}{A \cup B}$$

$$IoU \geq p, p \in (0,1)$$

So how do we evaluate our Object Detectors?

Define when a detection is correct, a true positive and when it is not.
A prediction will usually not overlap perfectly with the ground truth.

Assign true positives based on
Intersection over Union

```
prediction_prob = np.asarray([0.23, 0.37, 0.59, 0.84, 0.10, 0.92, 0.57, 0.49, 0.67, 0.51])
# Threshold for assignment
p = 0.5

# Define ground truth and true positives (1) and false negatives (0) of the prediction
groundtruth = np.asarray([1, 1, 1, 1, 1, 1, 1, 1, 0, 0])
prediction = (prediction_prob > p)*1
print(prediction)
# [0 0 1 1 0 1 1 0 1 1]
```

Precision

Detections over all Predictions
True Positives over True Positives and False Positives

$$mAP = \frac{1}{|c|} \sum_{c} \frac{TP(c)}{TP(c) + FP(c)}$$

```
# ------------------------------------------------------------------------------------------------------------------------
# 1. Precision
P = sum(prediction*groundtruth) / sum(prediction)
print('Precision: ', P)
# Precision: 0.66

# ------------------------------------------------------------------------------------------------------------------------
```

Recall

Detections over the number of Object Instances.
True Positives over True Positives and False Negatives.

$$mAR = \frac{1}{|c|} \sum_{c} \frac{TP(c)}{TP(c) + FN(c)}$$

```
# ----------------------------------------------------------------------------------------------------
# 2. Recall
R = sum(prediction*groundtruth) / sum(groundtruth)
print('Recall: ', R)
# Recall: 0.5

# ----------------------------------------------------------------------------------------------------
```

F1-Score

Harmonic mean between recall and precision
Harmonic mean lays more emphasis on the weaker metric, than the arithmetic would.

$$F1 = 2\,\frac{mAP\;mAR}{mAP + mAR}$$

```
# ------------------------------------------------------------------------------------------------------------------
# 3. F1-Score
# Harmonic mean
F = 2*R*P / (R+P)
print('F1-Score: ', F)
# F1-Score:  0.57

meanPR = (R+P) / 2
print('Arithmetic mean: ', meanPR)
# Arithmetic mean:  0.58

# ------------------------------------------------------------------------------------------------------------------
```

Confusion Matrix

The comparison of the model's predictions and the correct values in the form of a contingency table

```
# ------------------------------------------------------------------------------------------------------------------
# 4. Confusion matrix
CM = tf.confusion_matrix(labels=groundtruth,
                         predictions=prediction)

with tf.Session() as sess:
    print('Confusion Matrix: ', sess.run(CM))
# Confusion Matrix:
# [ [0 2]
#   [4 4] ]
#
# [ [true_negatives  | false_positives]
#   [false_negatives | true_positives ] ]

# ------------------------------------------------------------------------------------------------------------------
```

Confusion Matrix

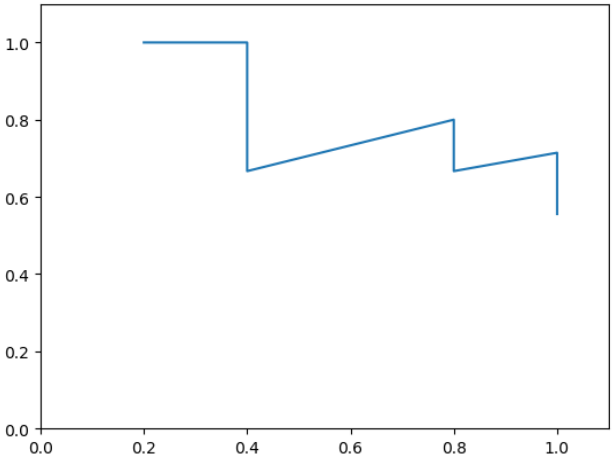
Which number in the matrix does never apply, when dealing with Object Detectors?

```
# ------------------------------------------------------------------------------------------------------------------
# 4. Confusion matrix
CM = tf.confusion_matrix(labels=groundtruth,
                         predictions=prediction)

with tf.Session() as sess:
    print('Confusion Matrix: ', sess.run(CM))
# Confusion Matrix:
# [ [0 2]
#   [4 4] ]
#
# [ [true_negatives  | false_positives]
#   [false_negatives | true_positives ] ]

# ------------------------------------------------------------------------------------------------------------------
```

Receiver Operating Characteristic (ROC)

The ROC curve displays the trade off between true-positive-rate and true-negative-rate.

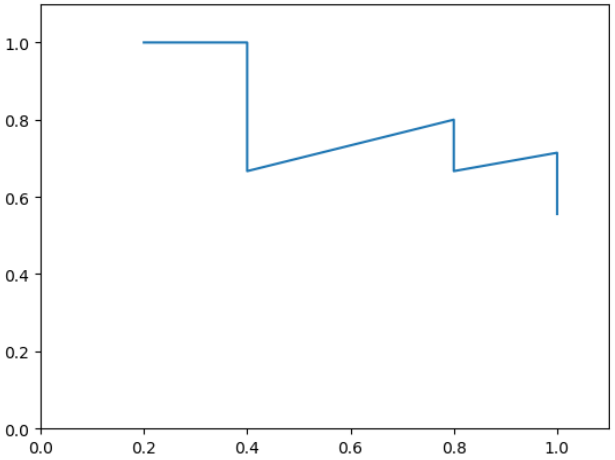
It can be seen as a matter of optimizing a threshold value with respect to a functional requirement.



Precision Recall Curve (PRC)

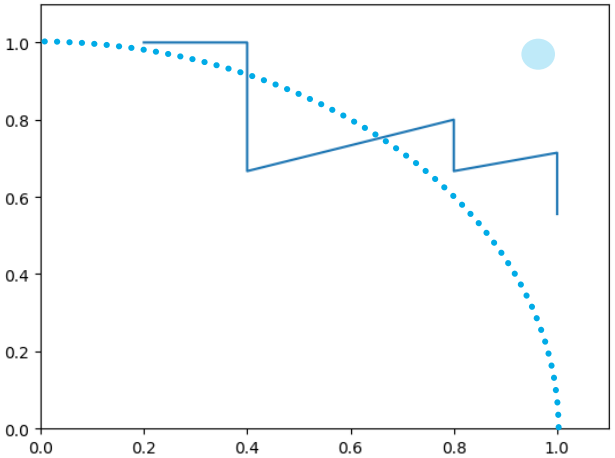
The PR curve displays the trade off between Precision and Recall.

It can be seen as a matter of optimizing a threshold value with respect to a functional requirement.



Receiver Operating Characteristic (ROC)

The ROC curve displays the trade off between Precision and Recall.



It can be seen as a matter of optimizing a **threshold** value with respect to a functional requirement.

- Intersection over Union
- Confidence

Mean Average Metric AP@[0.5:0.05:0.95]

COCO mAP. Mean Average Precision over a set of IoU levels. Can be understood as interpolated average precision (see COCO paper)