Slides will be found here from now on:

https://github.com/schutera/DeepLearningLecture_Schutera/tree/master/LectureNotes/DHBW22

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

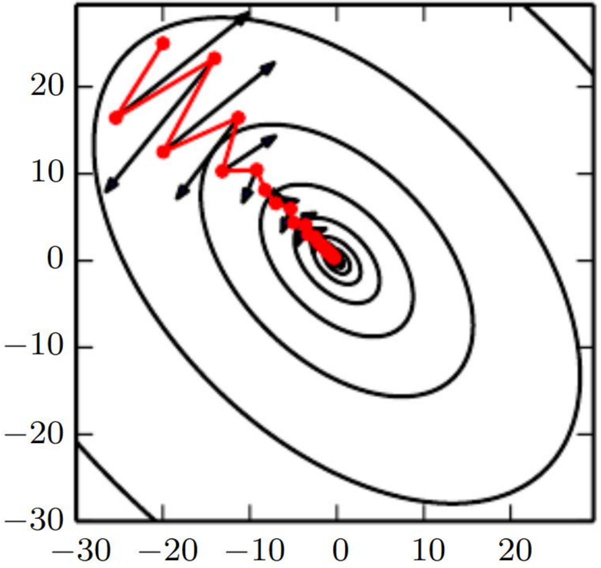
Weights and Biases:
- Symmetry
- Different initialization would lead to different outcome

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.

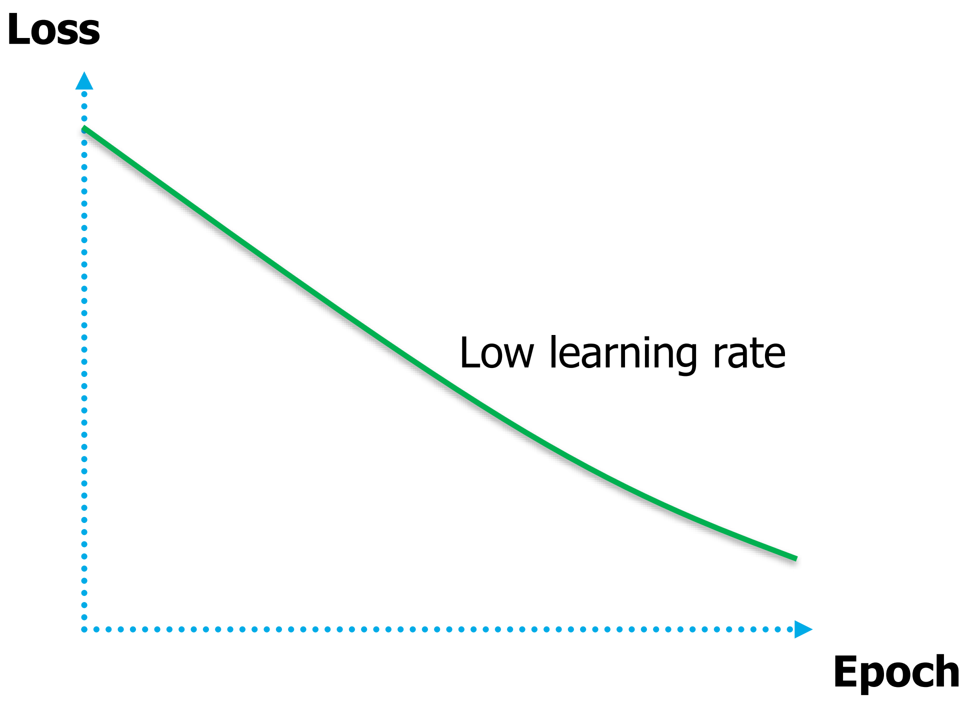
Adam

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.

Low learning rates

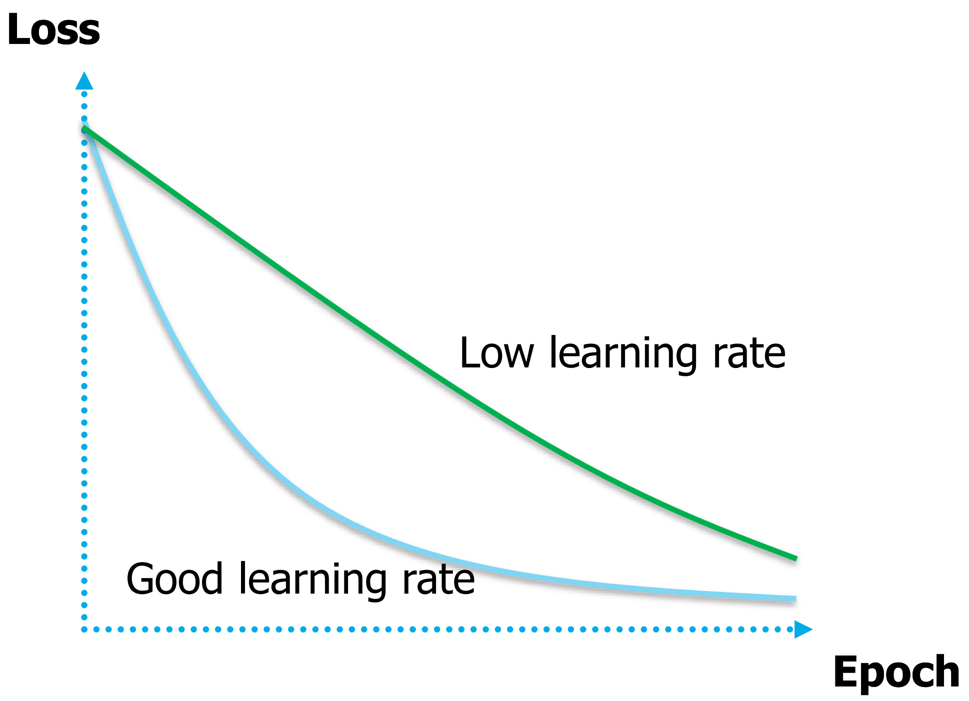
Loss decay will be linear, and result in high training times.



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.

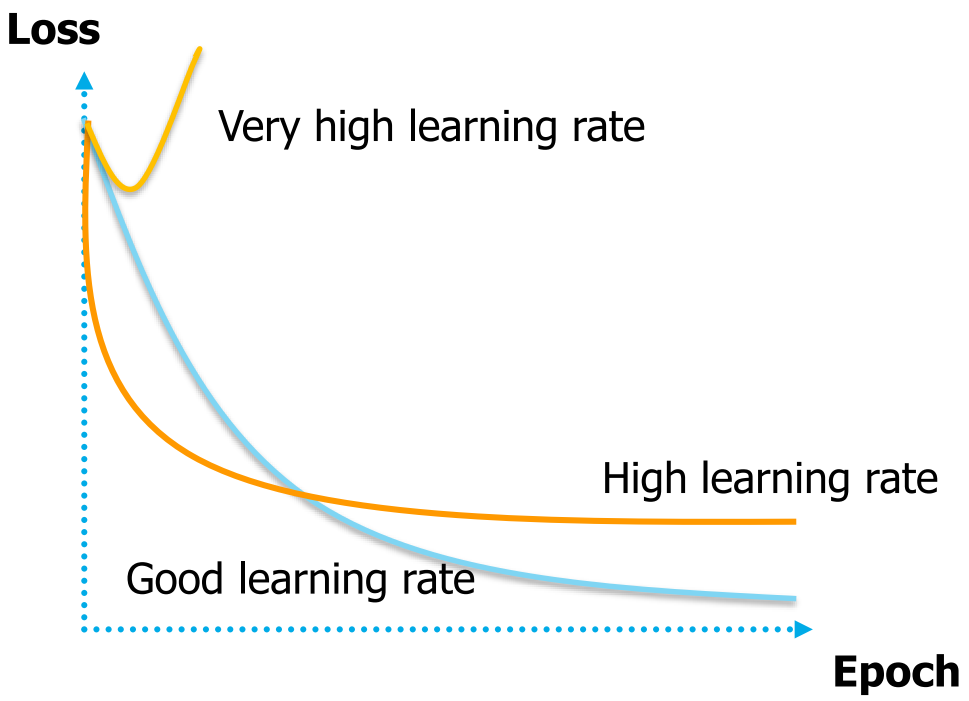


Neural Network Optimization – Learning Rate

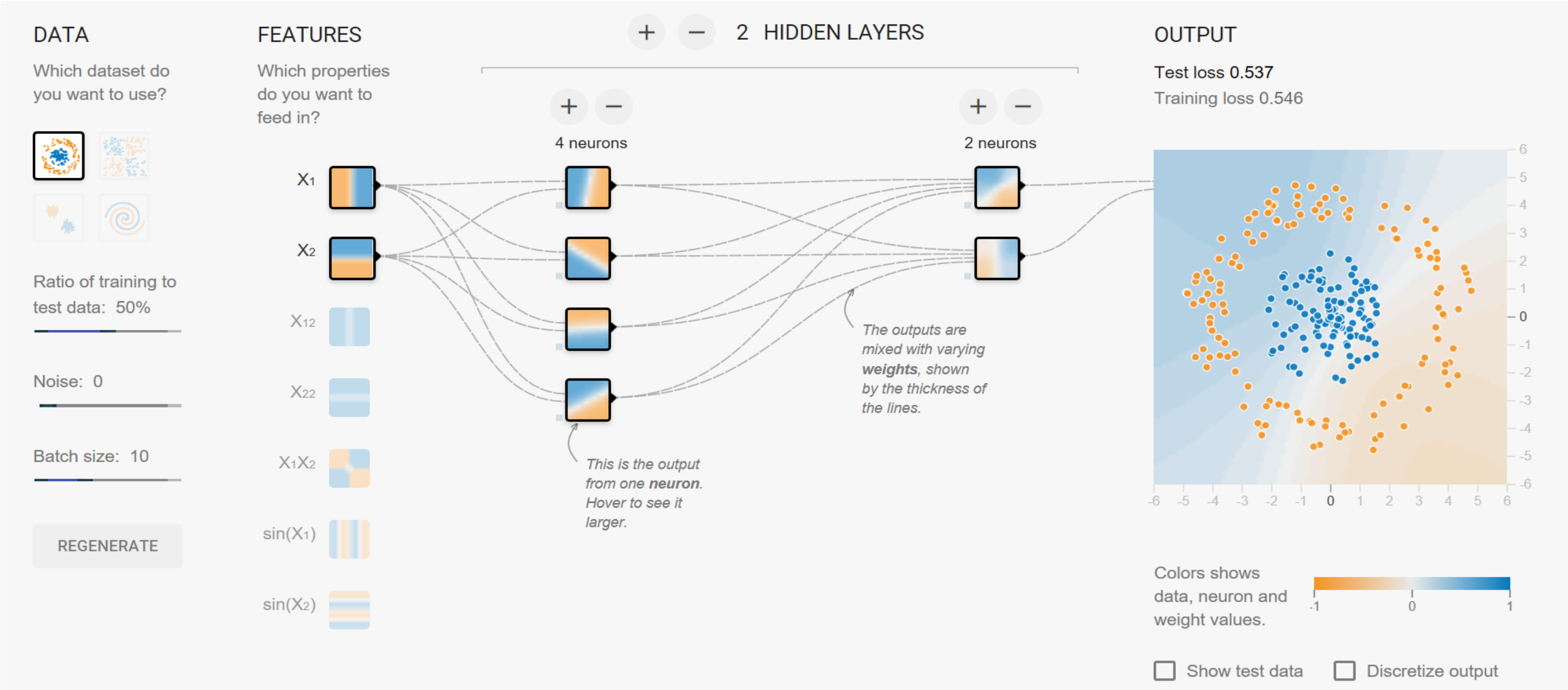
Higher learning rates

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

Optimization minimizes the error of a model on observed samples.

Machine Learning
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.

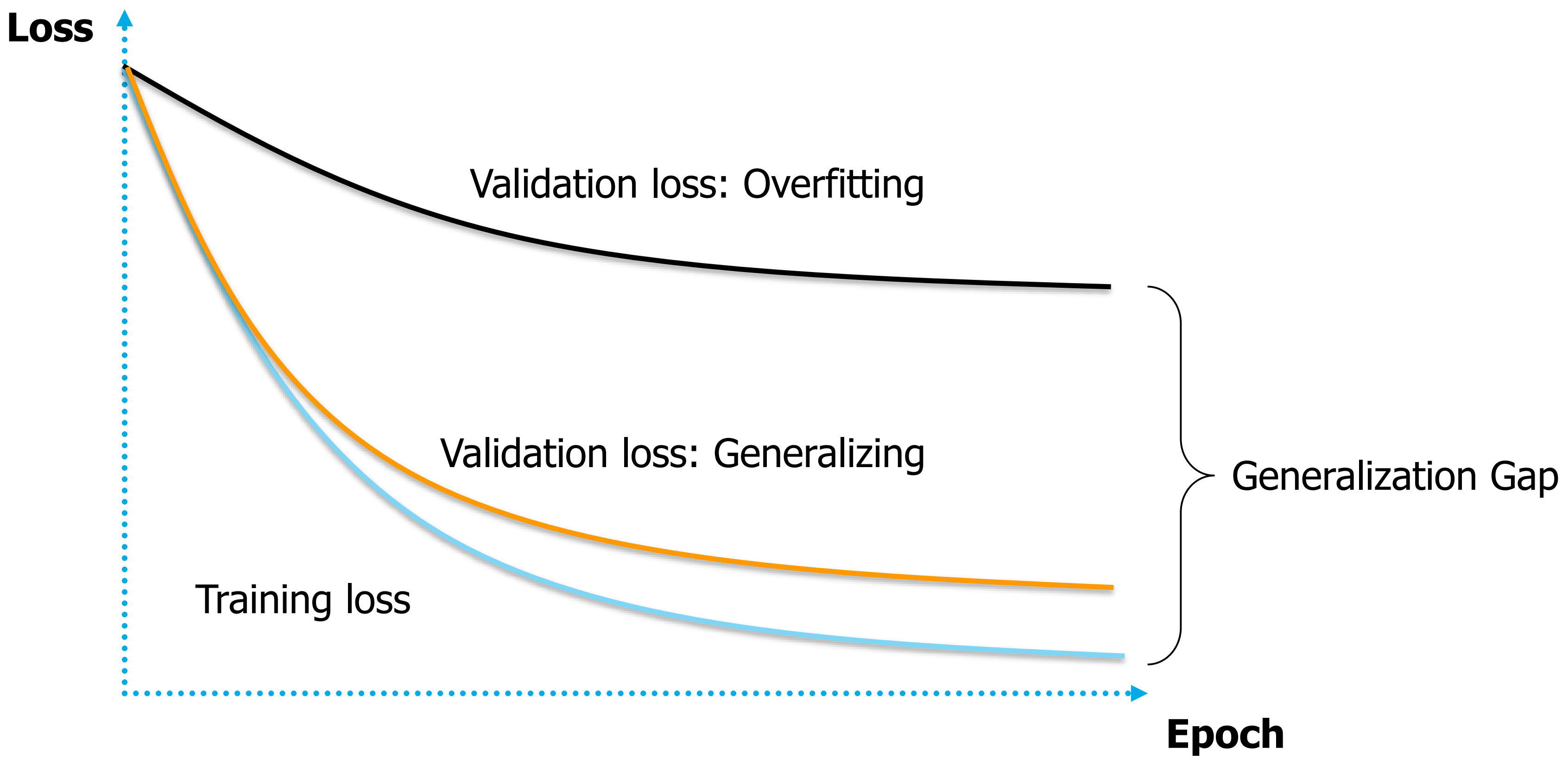
Regularization

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

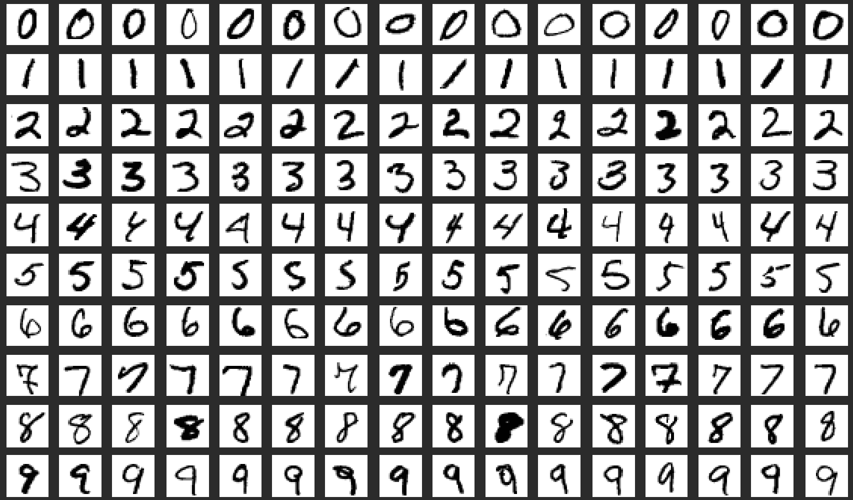


MNIST Dataset

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:

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

$$\Omega(w) = \sum_{i}^{n} w_i^2$$

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

Neural Network Regularization – Parameter Constraints

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)

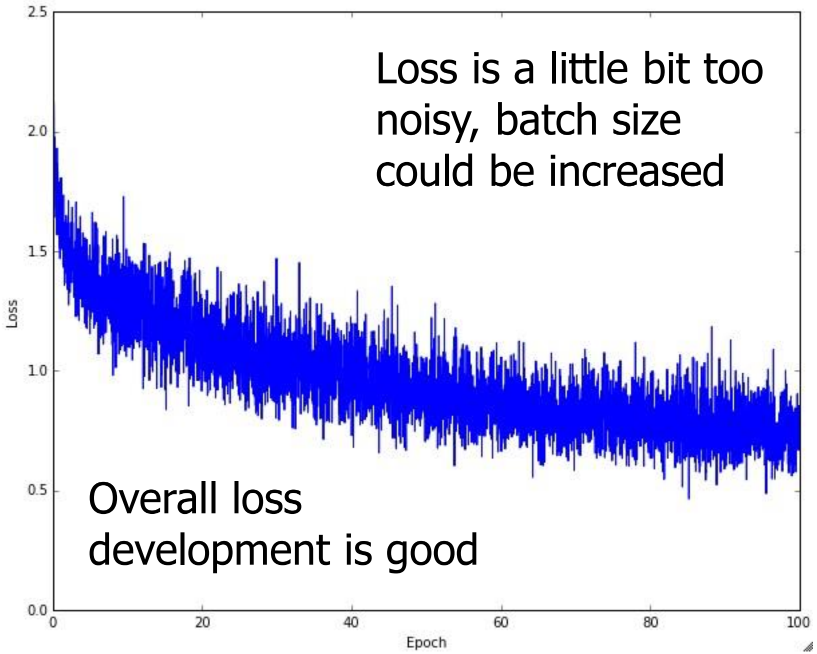
Why minibatches?

- Unbiased estimate of the gradient
- Computational effort

- Noise induced regularization for small batch sizes

Which batch size should I go for?

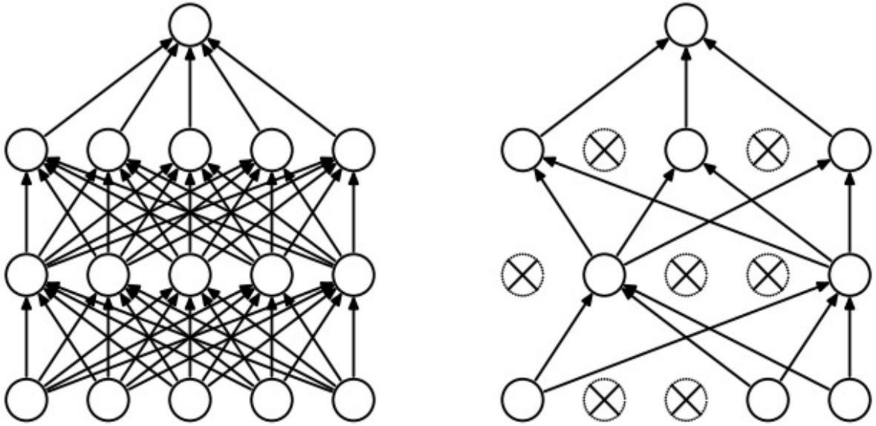
- Hardware restrictions set upper limit
- Power-of-two batch sizes match physical processor and improve runtime
- Loss band should be smooth, implying even gradient estimates.



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

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

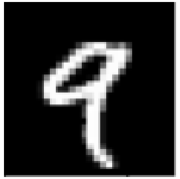
Examples of augmentation operations

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

Think before you augment:

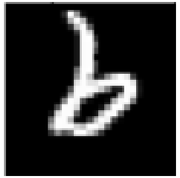
Prevent class switches and class breaks, know your data and your problem statement.

Initial sample



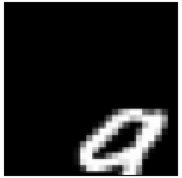
9

Rotation



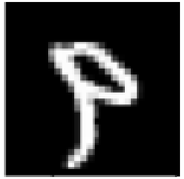
6

Shift



0

Mirror

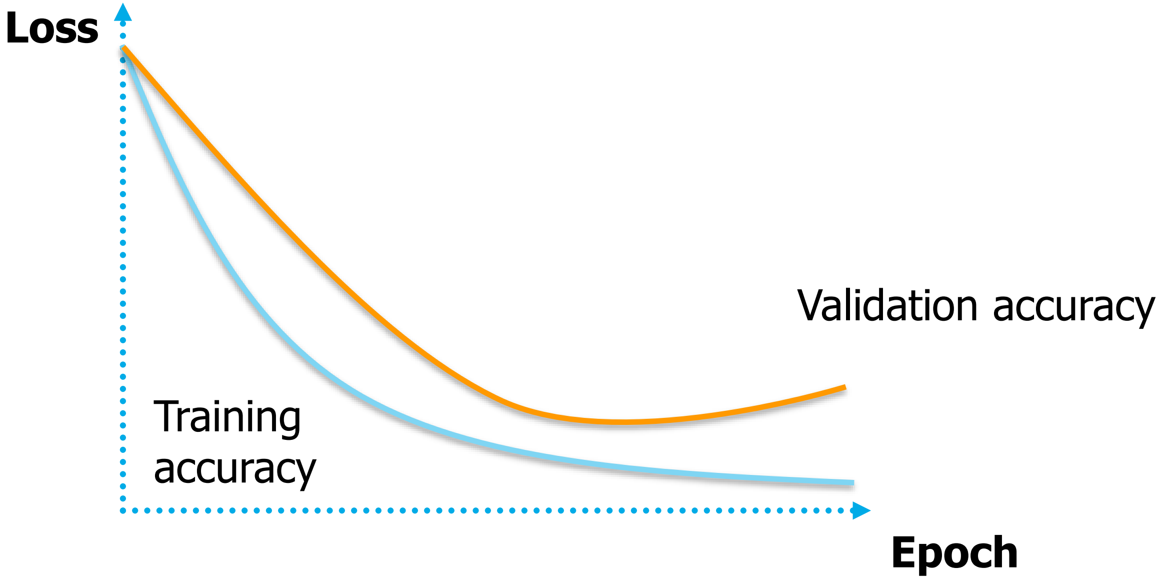


NaN

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

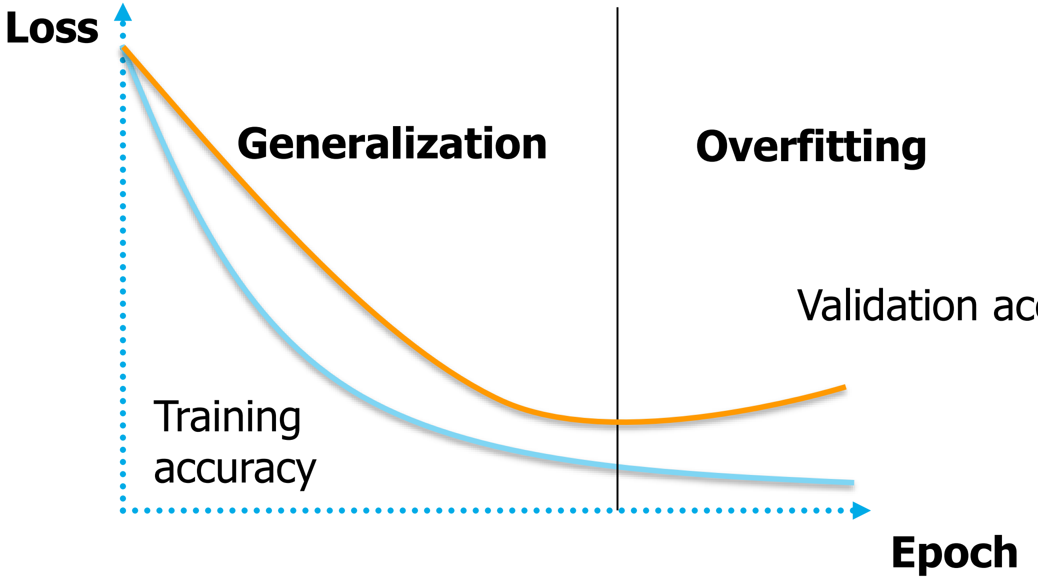
Neural Network Regularization – Early Stopping

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



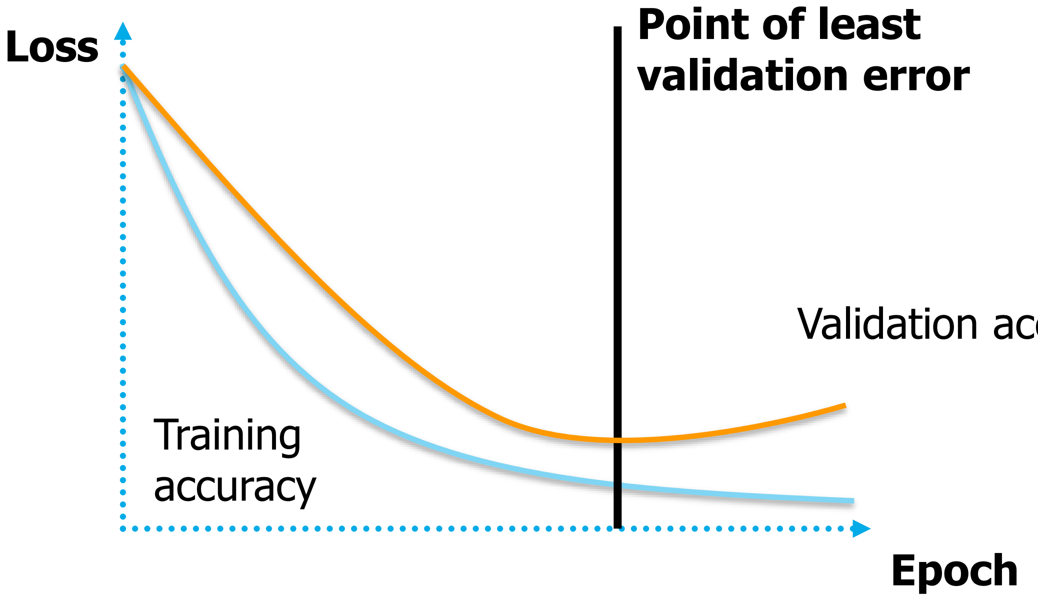
Neural Network Regularization – Early Stopping

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

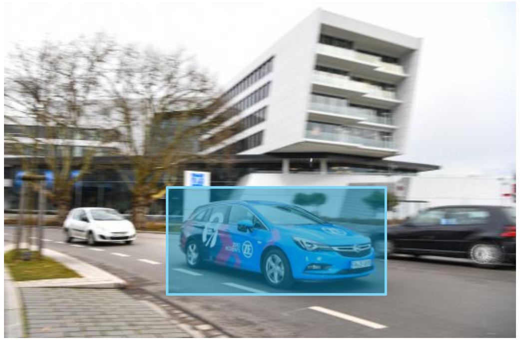
Application to Object Detection

Transfer Learning with neural networks

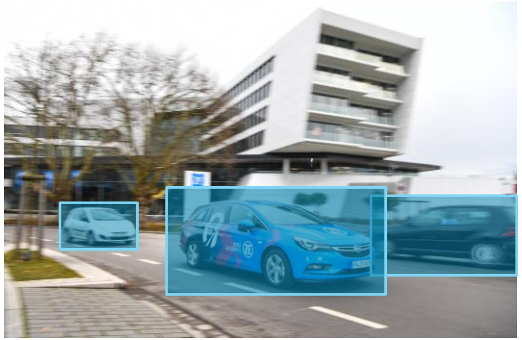
Introduction – Classification & Object Detection a problem statement

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

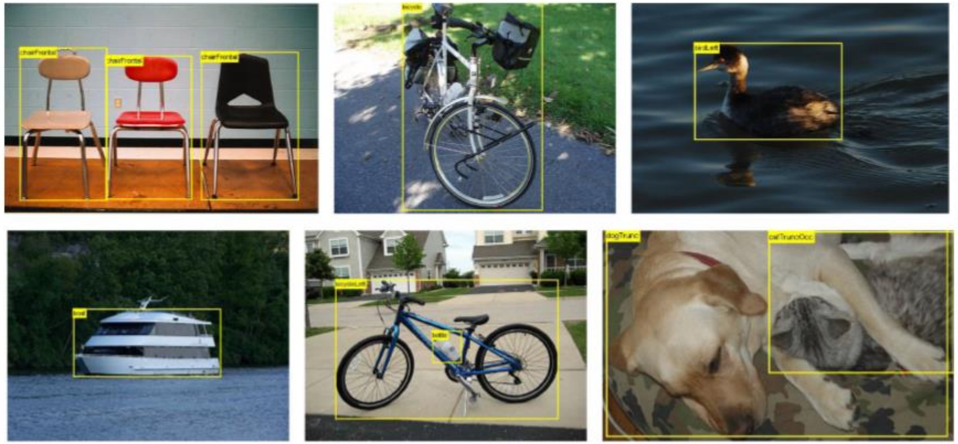
For each of twenty object classes predict the presence/absence of at least one object of that class in a test image.

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

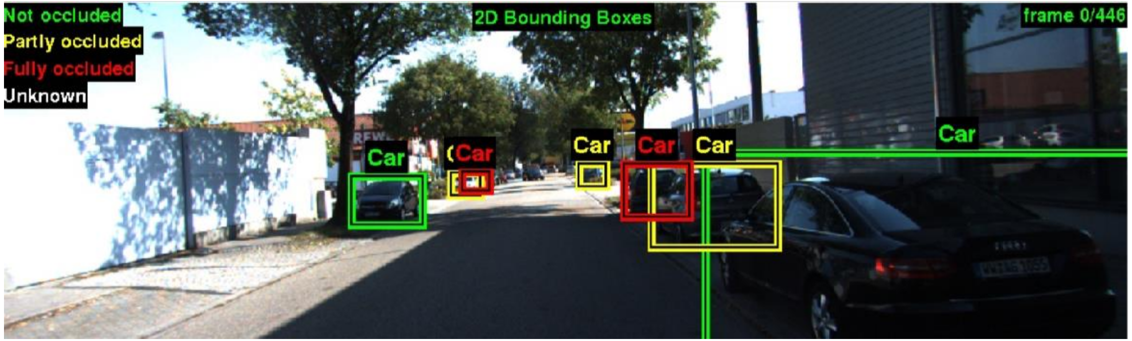
We take advantage of our autonomous driving platform Annieway to develop novel challenging real-world computer vision benchmarks.

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:

General

Input Layer

Backbone Network with Convolutional Layers

Task-specific head

Output Layer

Classification Example

MNIST Digit

5 layers of convolutions

Fully-connected Layer

Digit Prediction (0,1,…,9)

Object Detection Example

KITTI Image

34 layer VGG

Single Shot Detector Head

Bounding Boxes in Input Image

Convolutional Neural Networks

Typical structure of a neural network in Computer Vision:

General	Classification Example	Object Detection Example
Input Layer	MNIST Digit	KITTI Image
Backbone Network with Convolutional Layers	5 layers of convolutions	34 layer VGG
Task-specific head	Fully-connected Layer	Single Shot Detector Head
Output Layer	Digit Prediction (0,1,…,9)	Bounding Boxes in Input Image

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

w_{11}	w_{21}
w_{12}	w_{22}

Amount of Conv. Kernels
1

Depth of Feature Map
1

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}

Input Feature Map

Feature Map Size
Height Width Depth
4 x 4 x 1

Feature Map Size
Height Width Depth
3 x 3 x 1

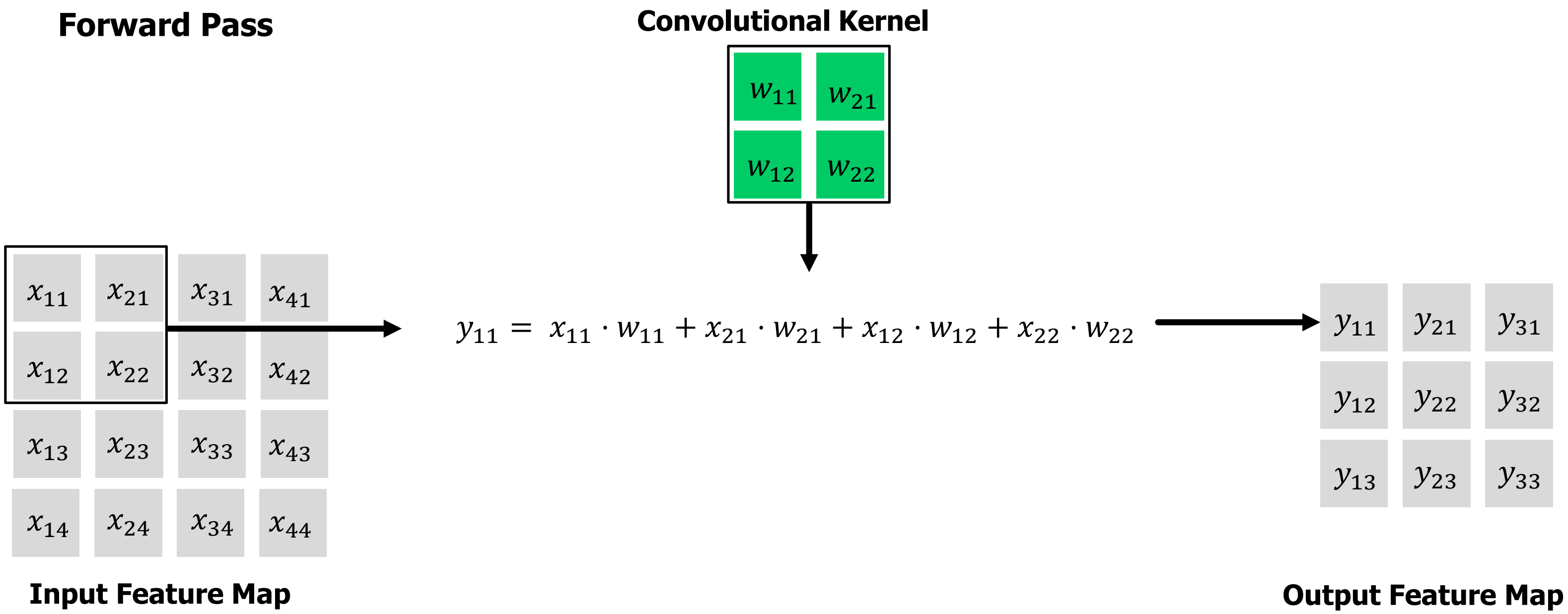
y_{11}	y_{21}	y_{31}
y_{12}	y_{22}	y_{32}
y_{13}	y_{23}	y_{33}

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:

Forward Pass



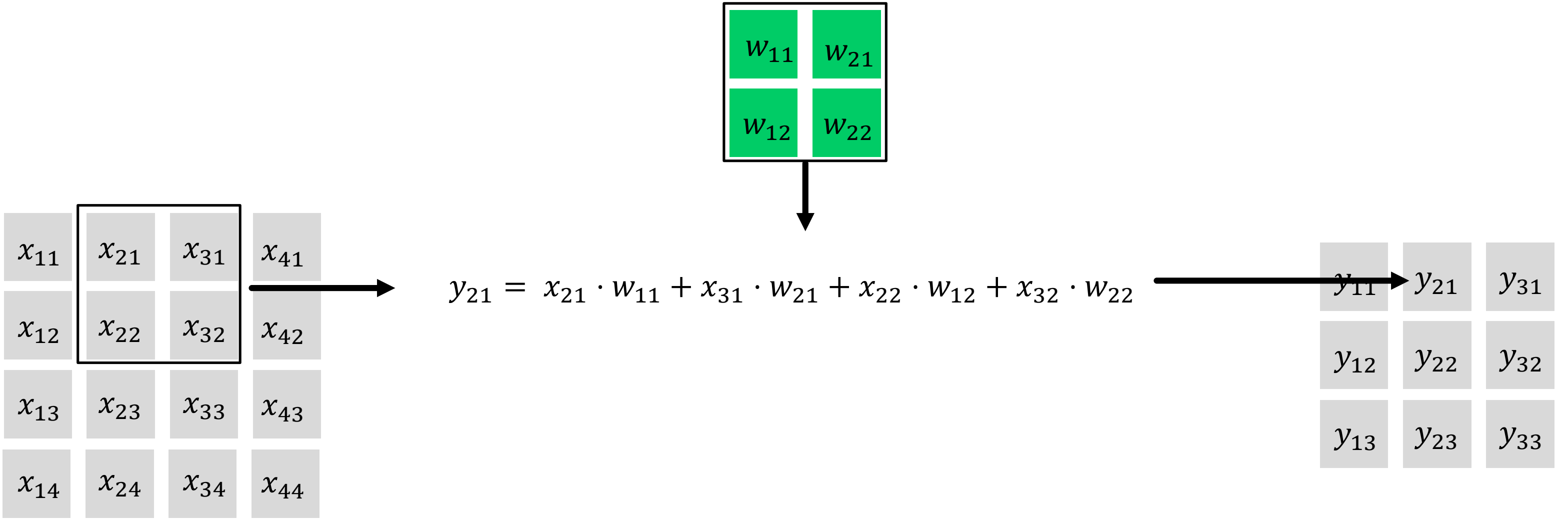
$$y_{11} = x_{11} \cdot w_{11} + x_{21} \cdot w_{21} + x_{12} \cdot w_{12} + x_{22} \cdot w_{22}$$

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



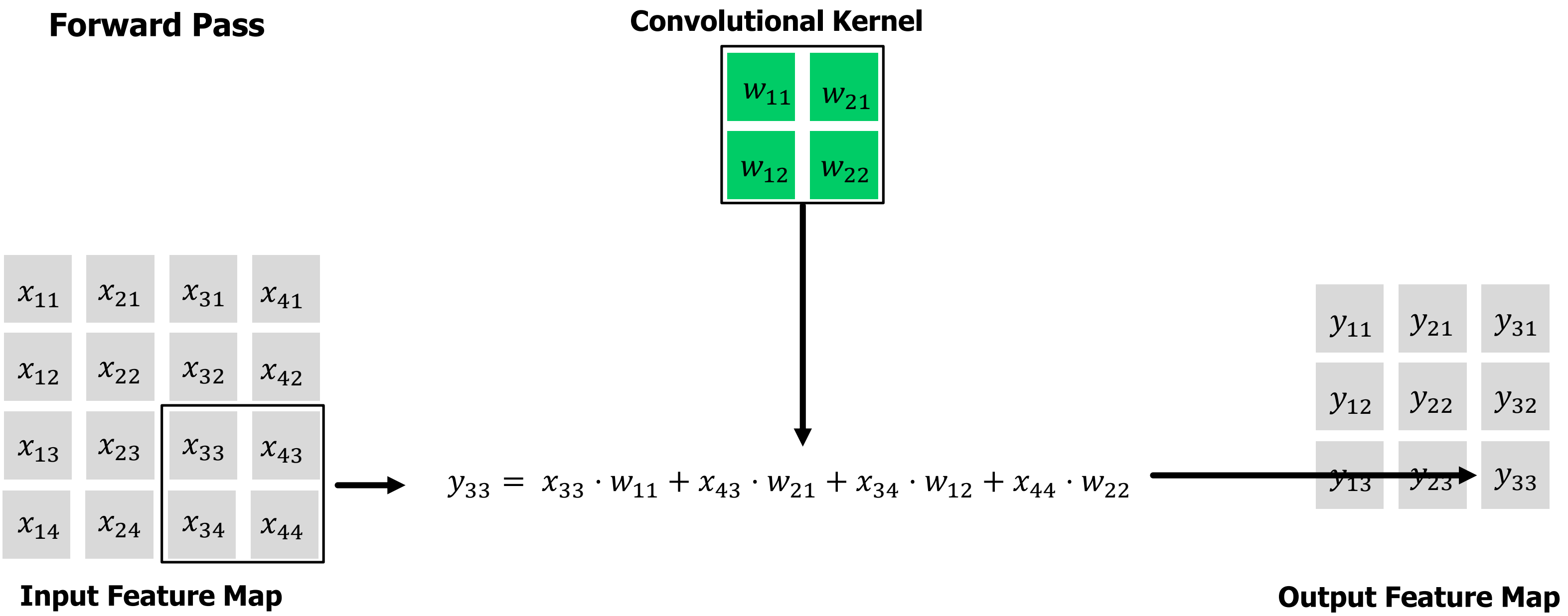
$$y_{21} = x_{21} \cdot w_{11} + x_{31} \cdot w_{21} + x_{22} \cdot w_{12} + x_{32} \cdot w_{22}$$

Input Feature Map

Output Feature Map

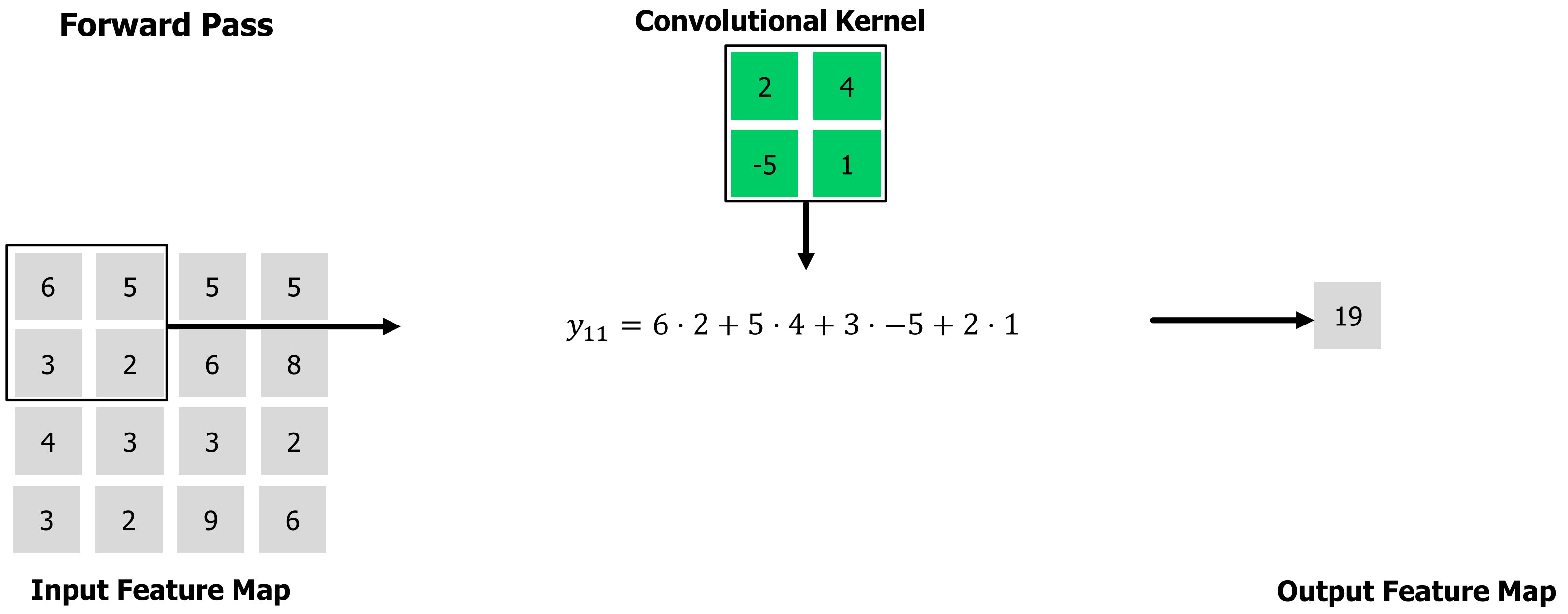
Convolutional Neural Networks – The convolution

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



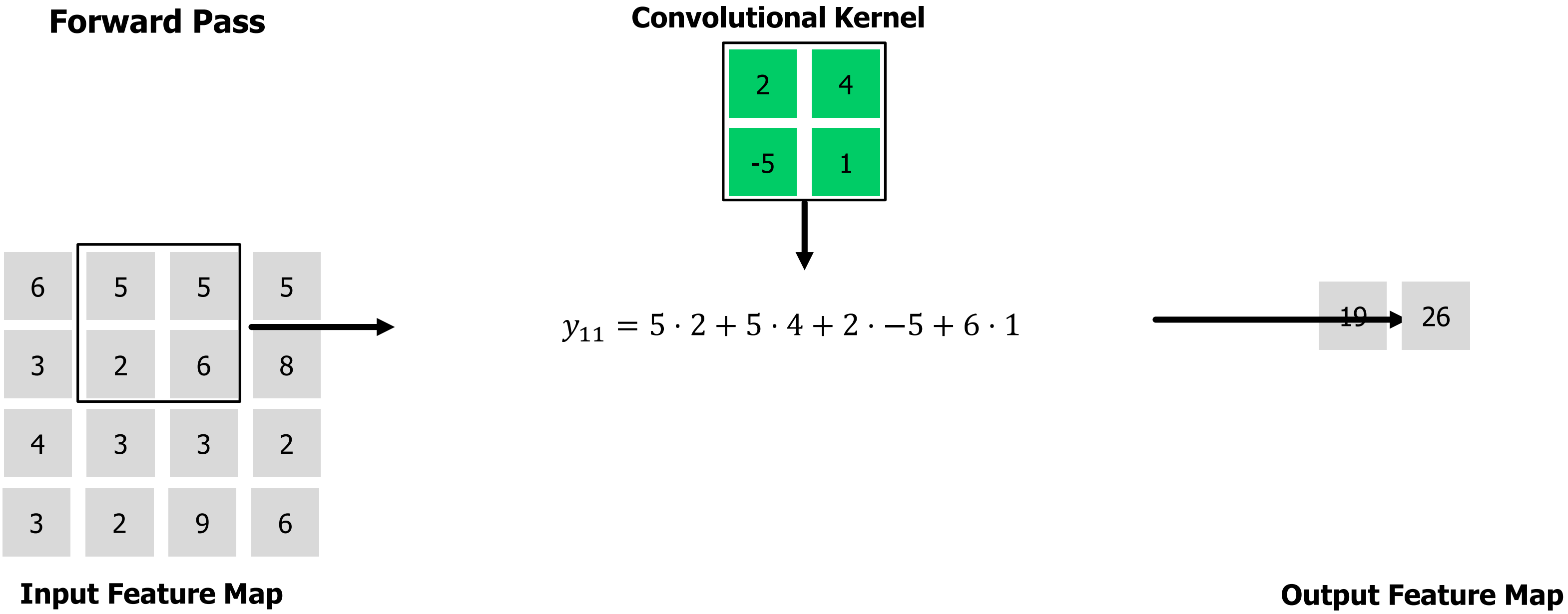
Convolutional Neural Networks – The convolution

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



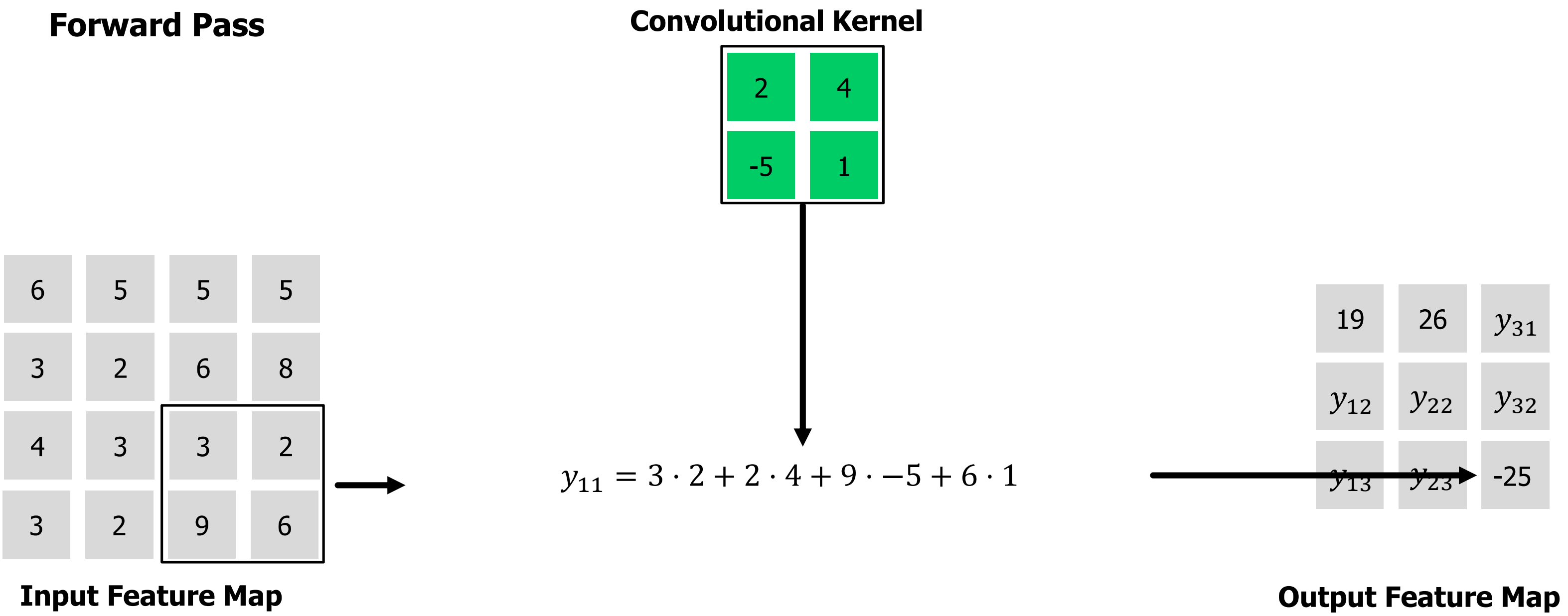
Convolutional Neural Networks – The convolution

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



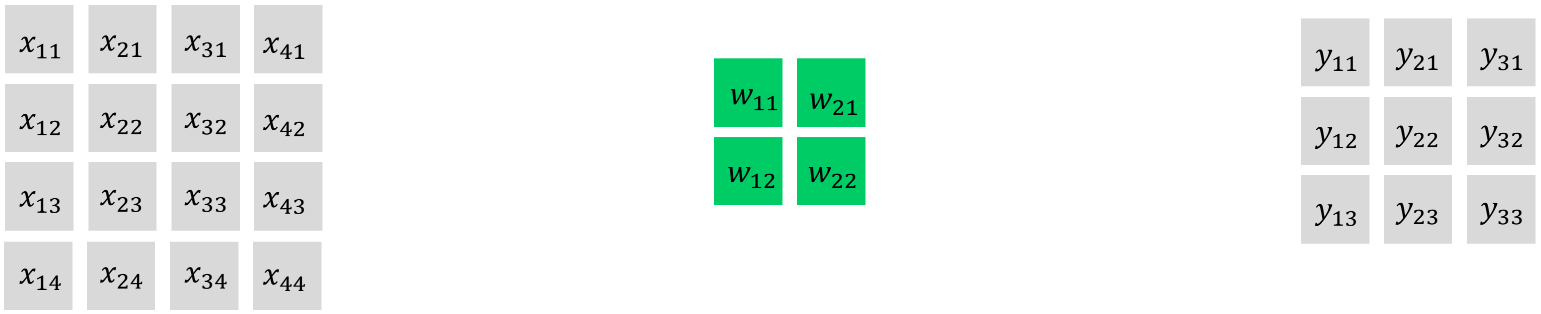
Convolutional Neural Networks – The convolution

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



Convolutional Neural Networks – Convolutions

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



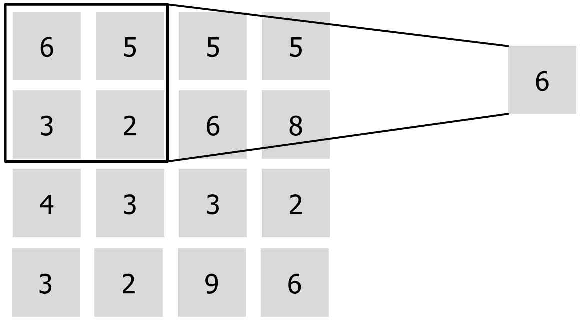
Convolutional layers introduce an **inductive bias** to our neural network:

- 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)



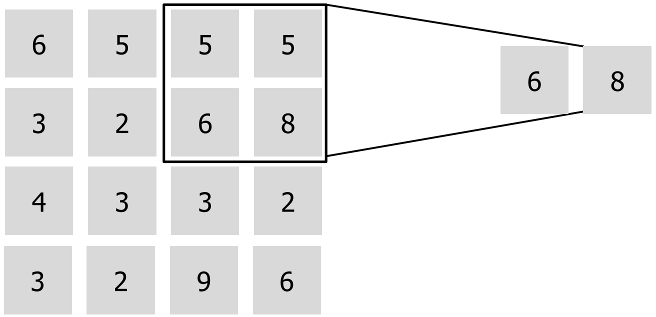
No trainable weights

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)

6	5	5	5
3	2	6	8
4	3	3	2
3	2	9	6

6	8
4	9

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.

No trainable weights

Max Pooling (filter 2x2, stride of 2)

6	5	5	5
3	2	6	8
4	3	3	2
3	2	9	6

6	8
4	9

Find max value in current window

Average Pooling (filter 2x2, stride of 2)

6	5	5	5
3	2	6	8
4	3	3	2
3	2	9	6

4

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)

6	5	5	5
3	2	6	8
4	3	3	2
3	2	9	6

6	8
4	9

No trainable weights

Find max value in current window

Average Pooling (filter 2x2, stride of 2)

6	5	5	5
3	2	6	8
4	3	3	2
3	2	9	6

4	6

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)

6	5	5	5
3	2	7	8
4	3	3	2
3	2	9	6

6	8
4	9

No trainable weights

Find max value in current window

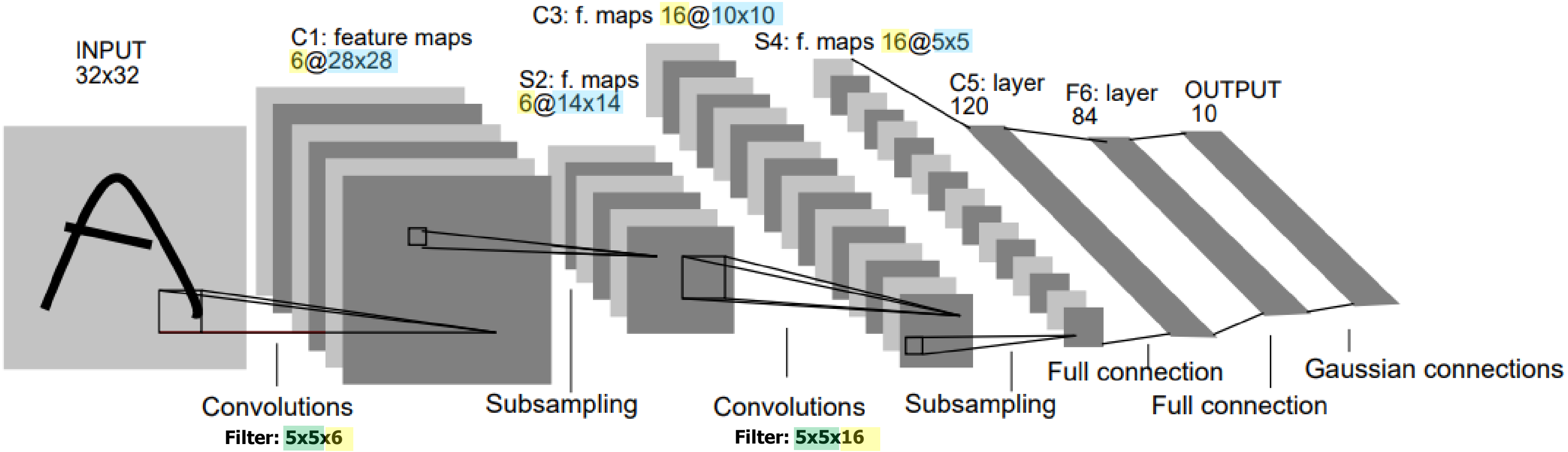
Average Pooling (filter 2x2, stride of 2)

6	5	5	5
3	2	6	8
4	3	3	2
3	2	9	6

4	6
3	5

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)