Hochschule Düsseldorf
University of Applied Sciences

Fachbereich Medien
Faculty of Media

Neural Nets 3

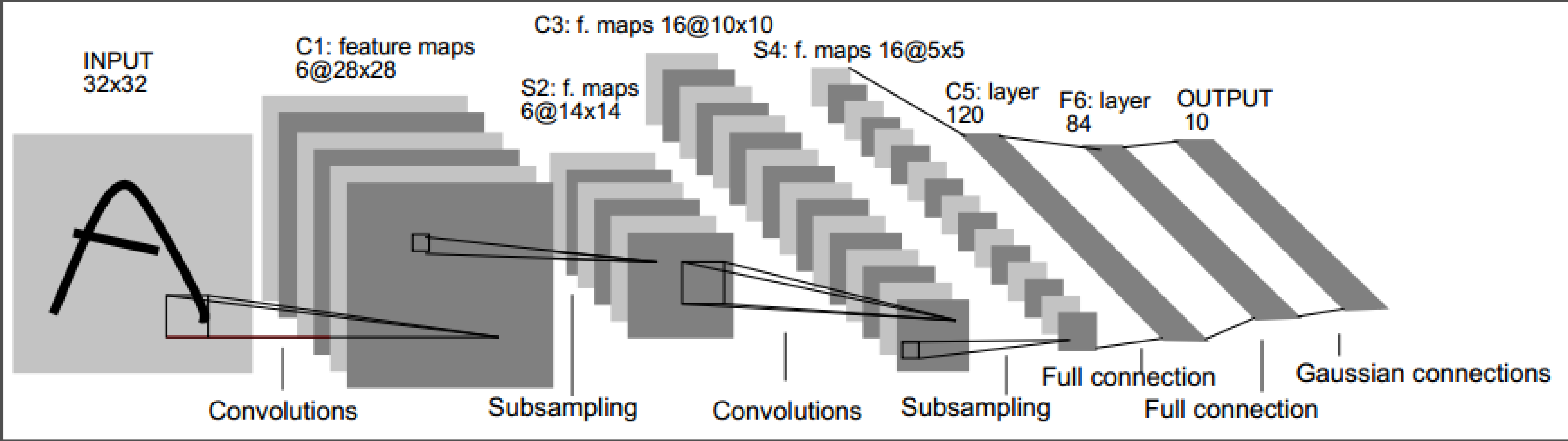
Performance & Optimization

Hochschule Düsseldorf
University of Applied Sciences

Fachbereich Medien
Faculty of Media

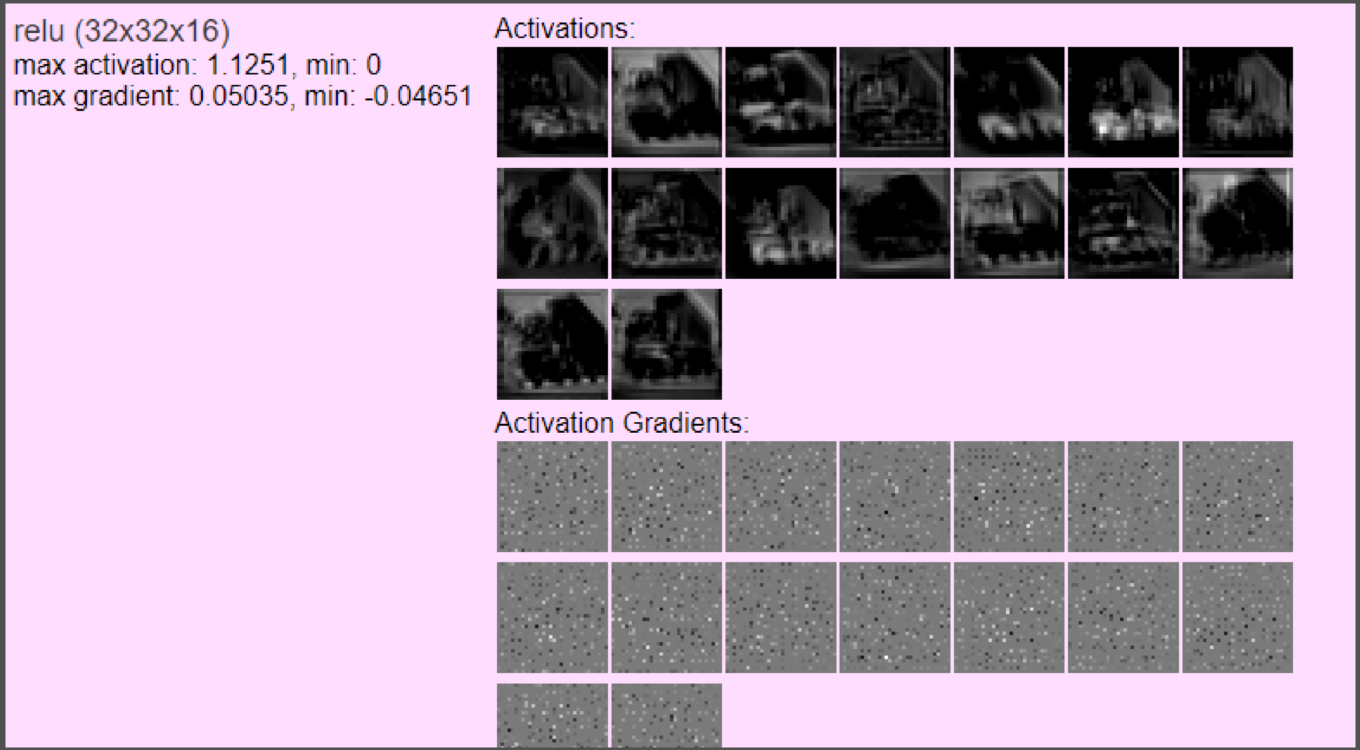
Overview:

- CNN Visualization
- Performance Metrics
- Batch, Mini-Batch and Stochastic Learning
- K-Fold-Cross Validation
- Momentum
- Dropout
- Plot while Training in tensorflow


C3: f. maps 16@10x10
C1: feature maps
6@28x28
INPUT
32x32
S4: f. maps 16@5x5
S2: f. maps
6@14x14
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

Visualizing the process within a CNN:

https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

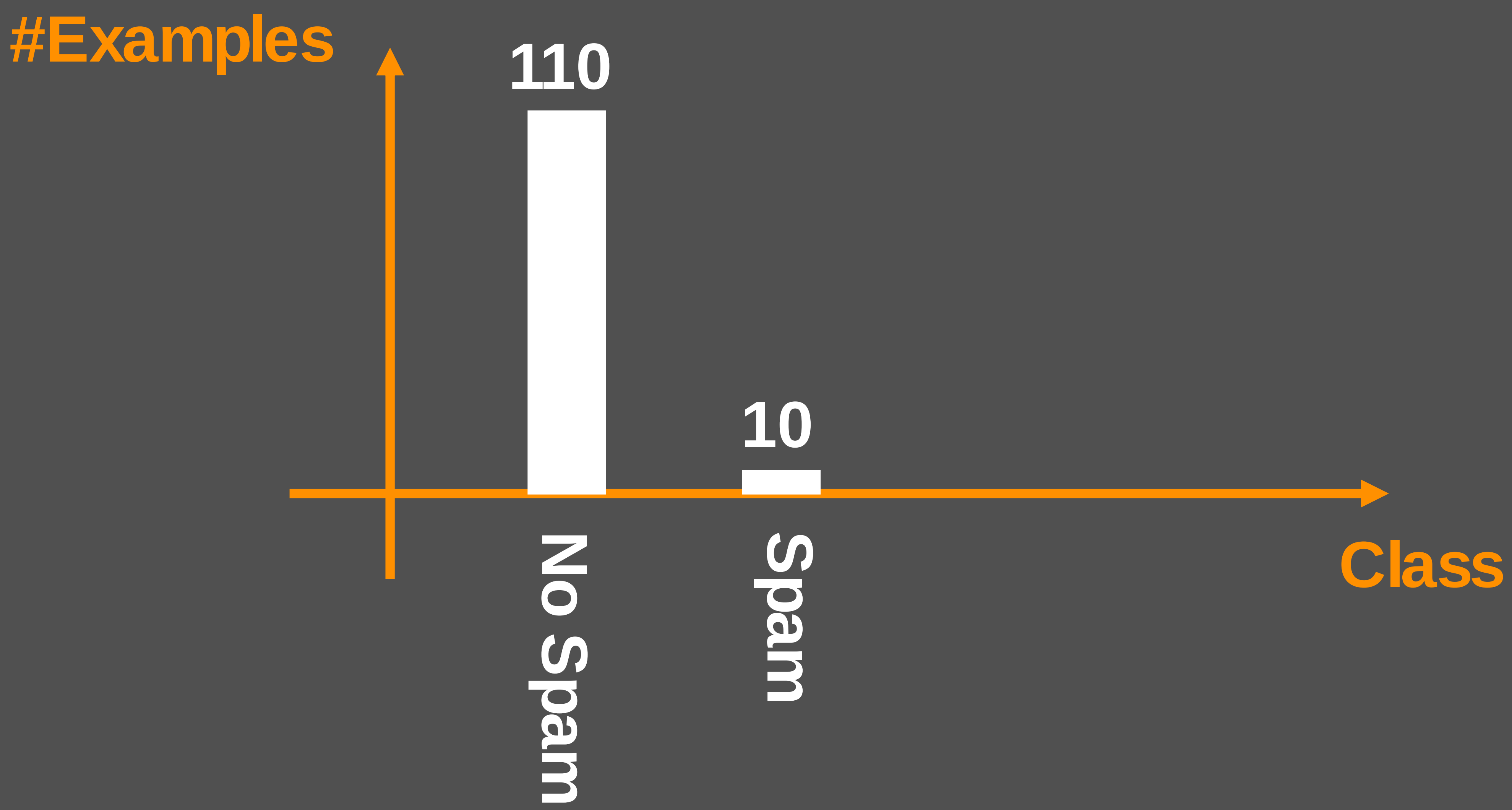


Improve generalization of CNN through augmentation of dataset:

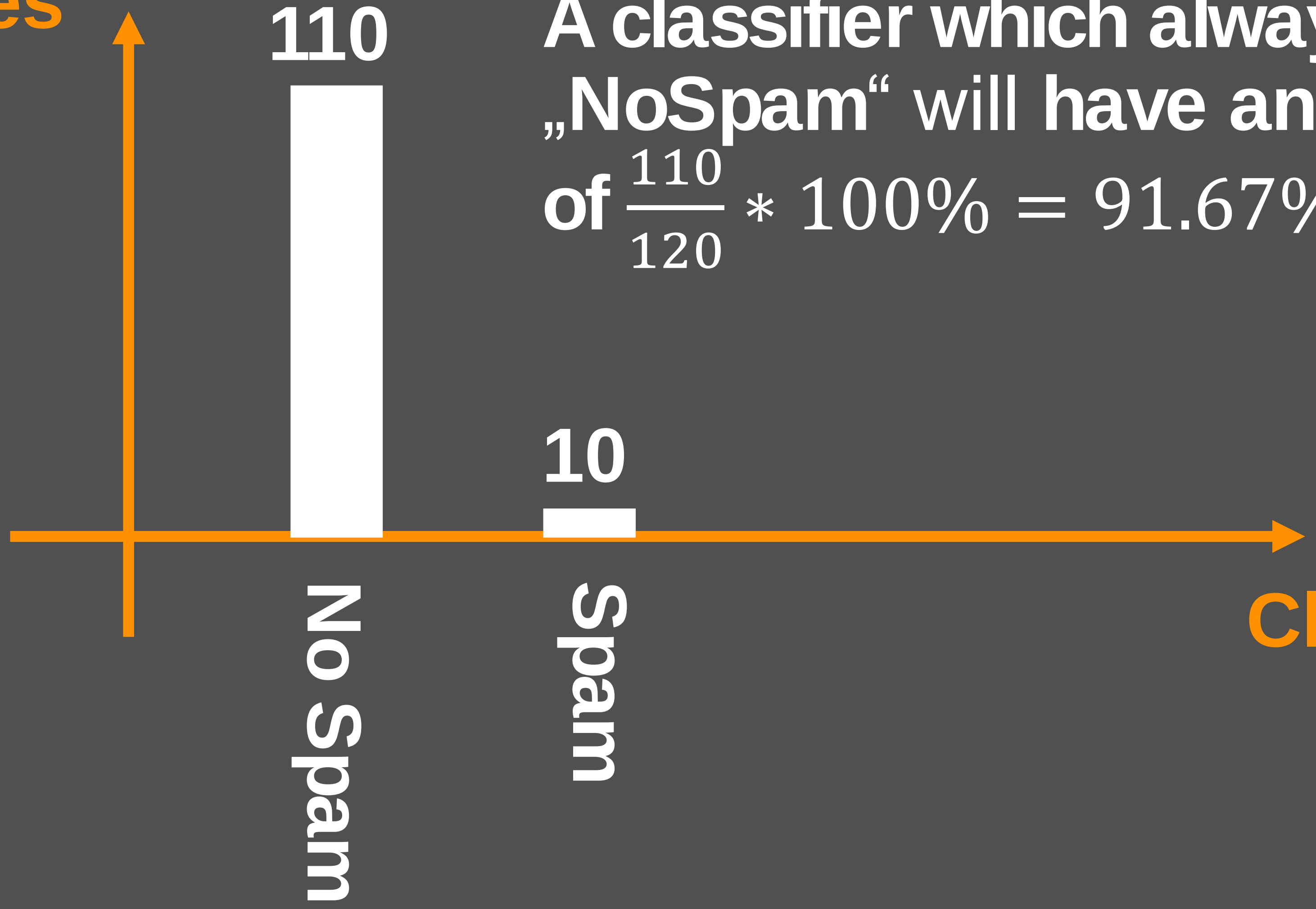
- Rotate
- Add Noise
- Mirror
- …

In which situations is the accuracy of a classifier not an appropriate metric?

Test-Set



Test-Set



A classifier which always says „NoSpam“ will **have an accuracy of** $\frac{110}{120} * 100\% = 91.67\%$

More Metrics

Binary Classification

True Class	Predicted Class	Type
0	0	True Positive
0	1	False Positive
1	0	False Negative
1	1	True Negative

Confusion Matrix

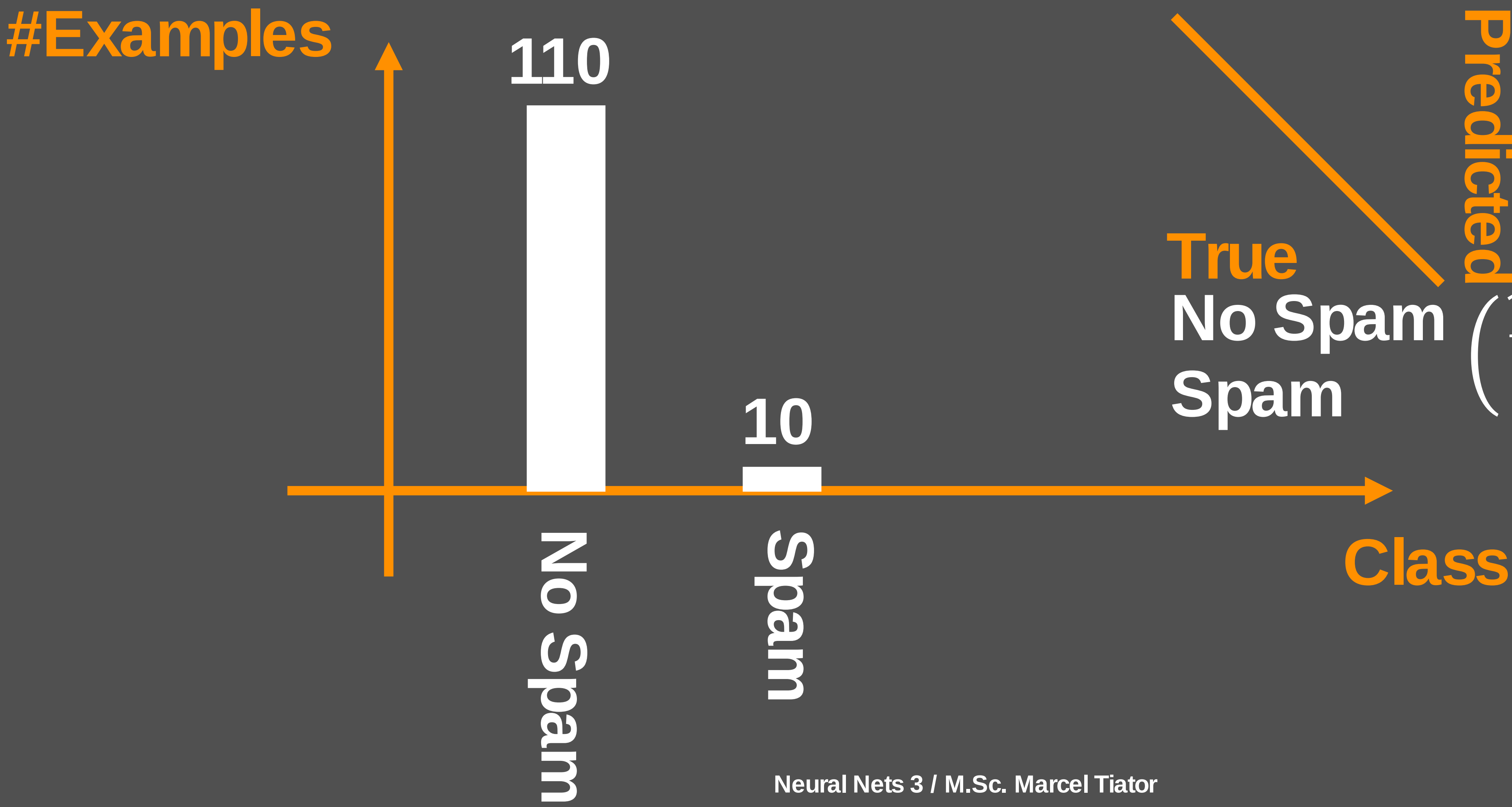
$$\begin{array}{ccc} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \multicolumn{2}{c}{\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}} \\ \mathbf{1} & & \end{array}$$

True Class	Predicted Class	Type
0	0	True Positive
0	1	False Positive
1	0	False Negative
1	1	True Negative

$$\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}$$

$$Accuracy = \frac{\#True\ Positive + \#True\ Negative}{\#Examples}$$

$$\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}$$



$$\begin{array}{ll} & \begin{array}{cc} \textbf{Predicted} & \\ \textbf{No Spam} & \textbf{Spam} \end{array} \\ \begin{array}{l} \textbf{True} \\ \textbf{No Spam} \\ \textbf{Spam} \end{array} & \begin{array}{c} \\ \begin{pmatrix} 110 & 0 \\ 10 & 0 \end{pmatrix} \end{array} \end{array}$$

$$\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}$$

$$\begin{array}{l|l} \text{True} \backslash \text{Predicted} & \begin{array}{cc} \text{No Spam} & \text{Spam} \end{array} \\ \begin{array}{l} \text{No Spam} \\ \text{Spam} \end{array} & \begin{pmatrix} 110 & 0 \\ 10 & 0 \end{pmatrix} \end{array}$$

$$Accuracy = \frac{\#True\ Positive + \#True\ Negative}{\#Examples}$$

$$Accuracy * 100\% = \frac{110 + 0}{120} * 100\% = 91.67\%$$

$$\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}$$

$$Precision = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Positive}$$

$$\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}$$

$$\begin{array}{ll} & \textbf{Predicted} \\ \textbf{True} & \begin{array}{cc} \textbf{No Spam} & \textbf{Spam} \end{array} \\ \textbf{No Spam} & \multirow{2}{*}{$\begin{pmatrix} 110 & 0 \\ 10 & 0 \end{pmatrix}$} \\ \textbf{Spam} & \end{array}$$

$$Precision = \frac{\#True\ Positive}{\#True\ Positive + False\ Positive}$$

$$Precision * 100\% = \frac{110}{110 + 10} * 100\% = 91.67\%$$

$$\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}$$

$$Recall = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Negative}$$

$$\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}$$

Predicted / **True**

$$\begin{array}{c|c} & \begin{array}{cc} \text{No Spam} & \text{Spam} \end{array} \\ \begin{array}{c} \text{No Spam} \\ \text{Spam} \end{array} & \begin{pmatrix} 110 & 0 \\ 10 & 0 \end{pmatrix} \end{array}$$

$$Recall = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Negative}$$

$$Precision * 100\% = \frac{110}{110 + 0} * 100\% = 100\%$$

$$Precision = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Positive}$$

$$Recall = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Negative}$$

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

$$Precision = 1$$

$$Recall = 1$$

$$F1Score = \frac{2 * 0.9167 * 1}{1 + 0.9167} \sim 0.957$$

$$\begin{array}{lc} \text{True} \; \backslash \; \text{Predicted} & \begin{array}{cc} \text{No Spam} & \text{Spam} \end{array} \\ \begin{array}{l} \text{No Spam} \\ \text{Spam} \end{array} & \begin{pmatrix} 110 & 0 \\ 10 & 0 \end{pmatrix} \end{array}$$

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

		predicted condition			
	total population	prediction positive	prediction negative	Prevalence $= \frac{\Sigma \text{ condition positive}}{\Sigma \text{ total population}}$	
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)	True Positive Rate (TPR), Sensitivity, Recall, Probability of Detection $= \frac{\Sigma \text{ TP}}{\Sigma \text{ condition positive}}$	False Negative Rate (FNR), Miss Rate $= \frac{\Sigma \text{ FN}}{\Sigma \text{ condition positive}}$
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**	False Positive Rate (FPR), Fall-out, Probability of False Alarm $= \frac{\Sigma \text{ FP}}{\Sigma \text{ condition negative}}$	True Negative Rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ TN}}{\Sigma \text{ condition negative}}$
	Accuracy $= \frac{\Sigma \text{ TP} + \Sigma \text{ TN}}{\Sigma \text{ total population}}$	Positive Predictive Value (PPV), Precision $= \frac{\Sigma \text{ TP}}{\Sigma \text{ prediction positive}}$	False Omission Rate (FOR) $= \frac{\Sigma \text{ FN}}{\Sigma \text{ prediction negative}}$	Positive Likelihood Ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic Odds Ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$
		False Discovery Rate (FDR) $= \frac{\Sigma \text{ FP}}{\Sigma \text{ prediction positive}}$	Negative Predictive Value (NPV) $= \frac{\Sigma \text{ TN}}{\Sigma \text{ prediction negative}}$	Negative Likelihood Ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$	

$$\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}$$

$$\begin{array}{ll} & \text{Predicted} \\ \text{True} & \begin{array}{cc} \text{No Spam} & \text{Spam} \end{array} \\ \begin{array}{l} \text{No Spam} \\ \text{Spam} \end{array} & \begin{pmatrix} 3 & 3 \\ 5 & 16 \end{pmatrix} \end{array}$$

$$\#Examples$$
$$= \#True\ Positive + \#False\ Negative + \#False\ Positive + \#True\ Negative$$

$$\#Examples = 3 + 3 + 5 + 16 = 27$$

$$\#Examples = 27$$

$$Accuracy = \frac{\#True\ Positive + \#True\ Negative}{\#Examples}$$

$$Accuracy = \frac{3+16}{27} \sim 0.7$$

$$\begin{array}{c|cc} \text{True} \backslash \text{Predicted} & \text{No Spam} & \text{Spam} \\ \hline \text{No Spam} & 3 & 3 \\ \text{Spam} & 5 & 16 \end{array}$$

$$\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}$$

$$Precision = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Positive}$$

$$Precision = \frac{3}{3+5} \sim 0.375$$

Predicted

True

$$\begin{array}{c c} & \begin{array}{cc} \text{No Spam} & \text{Spam} \end{array} \\ \begin{array}{c} \text{No Spam} \\ \text{Spam} \end{array} & \begin{pmatrix} 3 & 3 \\ 5 & 16 \end{pmatrix} \end{array}$$

$$\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}$$

$$Recall = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Negative}$$

$$Recall = \frac{3}{3+3} = 0.5$$

$$\begin{array}{ccc} & \multicolumn{2}{c}{\text{Predicted}} \\ \text{True} & \text{No Spam} & \text{Spam} \\ \text{No Spam} & \multicolumn{2}{c}{\multirow{2}{*}{$\begin{pmatrix} 3 & 3 \\ 5 & 16 \end{pmatrix}$}} \\ \text{Spam} & & \end{array}$$

$$\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}$$

Hochschule Düsseldorf
University of Applied Sciences

Fachbereich Medien
Faculty of Media

MIREVI

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

$$Precision = 0.375$$

$$Recall = 0.5$$

$$F1Score = \frac{2 * 0.375 * 0.5}{0.375 + 0.5} \sim 0.429$$

$$\begin{array}{r c}
 & \text{Predicted} \quad \text{No Spam} \quad \text{Spam} \\
\text{True} & \\
\text{No Spam} & \multirow{2}{*}{$\begin{pmatrix} 3 & 3 \\ 5 & 16 \end{pmatrix}$} \\
\text{Spam} &
\end{array}$$

$$\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}$$

$$Accuracy = 0.7$$

$$Precision = 0.375$$

$$Recall = 0.5$$

$$F1Score = 0.429$$

Predicted

True

$$\begin{array}{l c} & \begin{array}{cc} \text{No Spam} & \text{Spam} \end{array} \\ \begin{array}{l} \text{No Spam} \\ \text{Spam} \end{array} & \begin{pmatrix} 3 & 3 \\ 5 & 16 \end{pmatrix} \end{array}$$

$$\begin{pmatrix} \#True\ Positive & \#False\ Negative \\ \#False\ Positive & \#True\ Negative \end{pmatrix}$$

Hochschule Düsseldorf
University of Applied Sciences

Fachbereich Medien
Faculty of Media

Given the following Confusion Matrix – calculate:

- Accuracy
- Average Precision
- Average Recall
- F1Score of averaged Precision and Recall

True \ Predicted	Dog	Cat	Mouse
Dog	1	0	1
Cat	0	3	2
Mouse	2	1	2

$$Accuracy = \frac{\#True\ Positive + \#True\ Negative}{\#Examples}$$

$$Precision = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Positive}$$

$$Recall = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Negative}$$

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

True \ Predicted	Dog	Cat	Mouse
Dog	1	0	1
Cat	0	3	2
Mouse	2	1	2

- $\#Example = 12$

True \ Predicted	Dog	Cat	Mouse
Dog	1	0	1
Cat	0	3	2
Mouse	2	1	2

- $\#Example = 12$
- $Accuracy = 0.5$

True \ Predicted	Dog	Cat	Mouse
Dog	1	0	1
Cat	0	3	2
Mouse	2	1	2

- $\#Example = 12$
- $Accuracy = 0.5$
- $Precision(Dog) \sim 0.3$
- $Precision(Cat) = 0.75$
- $Precision(Mouse) = 0.4$
- $\overline{Precision} = 0.48$

True \ Predicted	Dog	Cat	Mouse
Dog	1	0	1
Cat	0	3	2
Mouse	2	1	2

- $\#Example = 12$
- $Accuracy = 0.5$
- $Precision(Dog) \sim 0.3$
- $Precision(Cat) = 0.75$
- $Precision(Mouse) = 0.4$
- $\overline{Precision} = 0.48$
- $Recall(Dog) = 0.5$
- $Recall(Cat) = 0.6$
- $Recall(Mouse) = 0.4$
- $\overline{Recall} = 0.5$

True \ Predicted	Dog	Cat	Mouse
Dog	1	0	1
Cat	0	3	2
Mouse	2	1	2

- $\#Example = 12$
- $Accuracy = 0.5$
- $Precision(Dog) \sim 0.3$
- $Precision(Cat) = 0.75$
- $Precision(Mouse) = 0.4$
- $\overline{Precision} = 0.48$
- $Recall(Dog) = 0.5$
- $Recall(Cat) = 0.6$
- $Recall(Mouse) = 0.4$
- $\overline{Recall} = 0.5$
- $F1Score \sim 0.49$

True \ Predicted	Dog	Cat	Mouse
Dog	1	0	1
Cat	0	3	2
Mouse	2	1	2

Full-Batch Learning

Full Dataset Length

$$E = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$

repeat until convergence{

$$\theta_1 := \theta_1 - \alpha * \frac{\partial E(\theta_1)}{\partial \theta_1}$$

}

Stochastic Gradient Descent

One Example

$$E = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 = \frac{1}{2}(\hat{y}_i - y_i)^2$$

repeat until convergence{

$$\theta_1 := \theta_1 - \alpha * \frac{\partial E(\theta_1)}{\partial \theta_1}$$

}

Minibatch

Length of Minibatch

$$E = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$

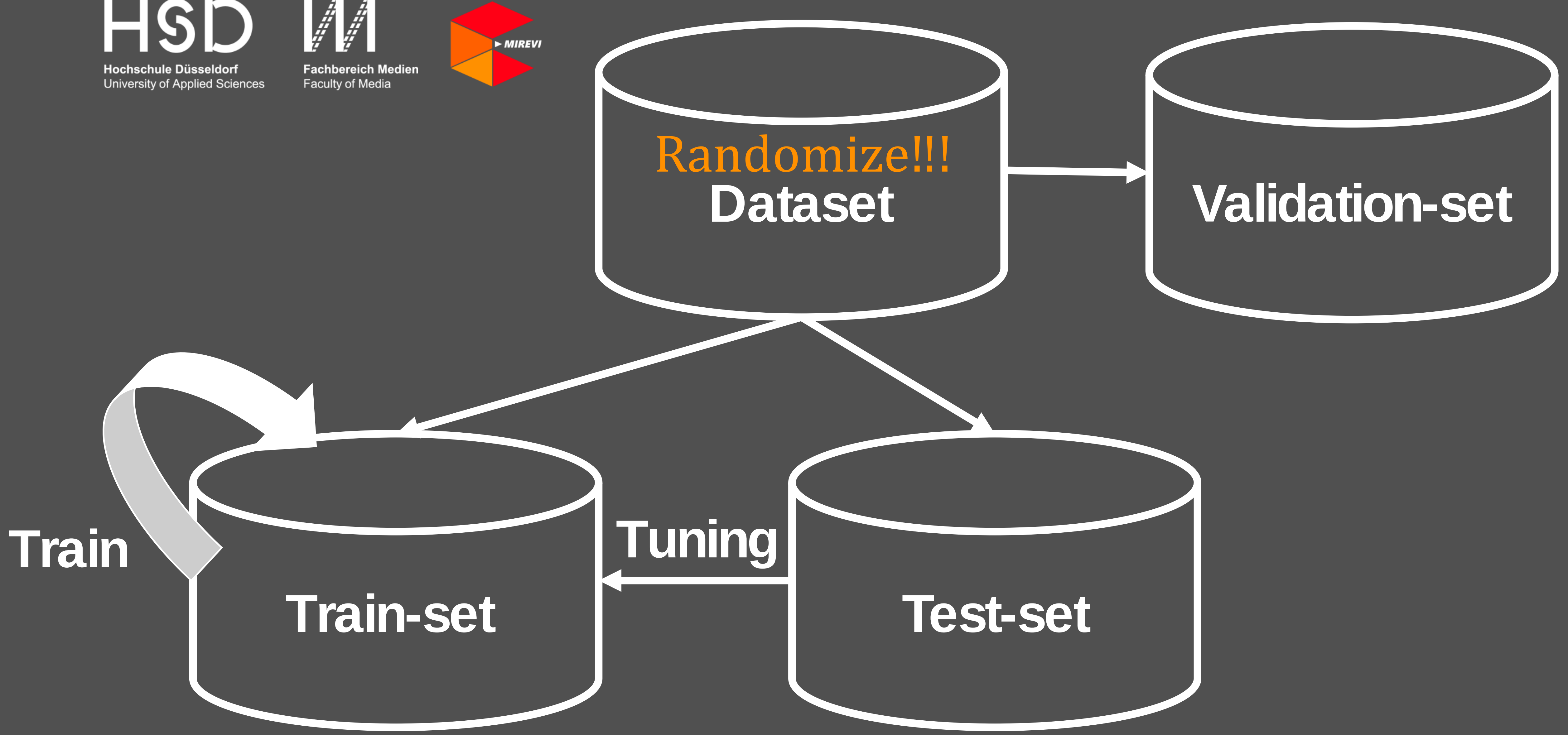
repeat until convergence{

$$\theta_1 := \theta_1 - \alpha * \frac{\partial E(\theta_1)}{\partial \theta_1}$$

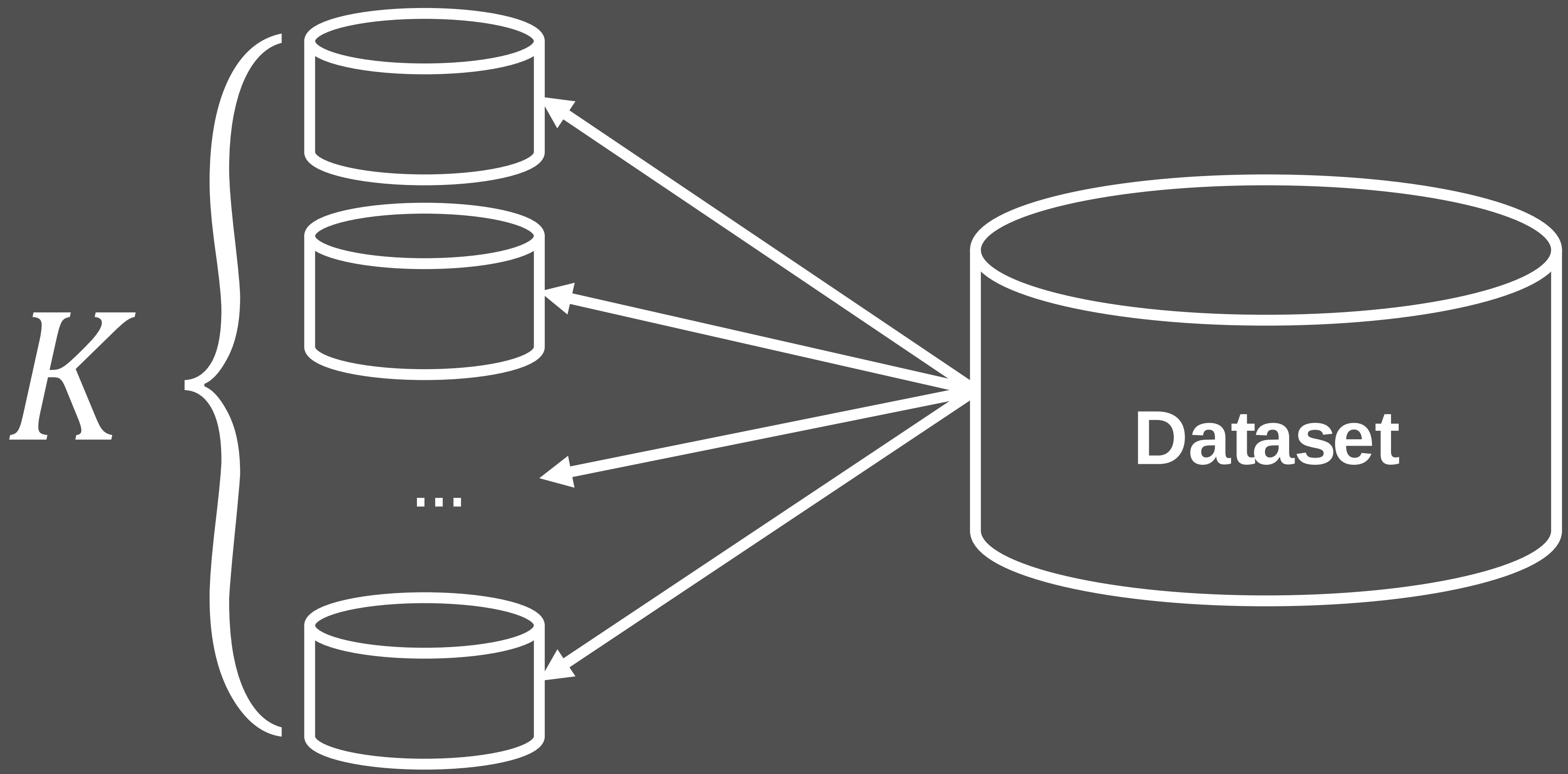
}

	Stochastic GD	Full-Batch	Mini-Batch
Online-Learning	Yes	No	Yes
Variance of Gradients	High	Low	Middle
Fit in Memory	Yes	Depends on dataset length	Depends on batch-size
Speed of Convergence	High	Slow	Medium

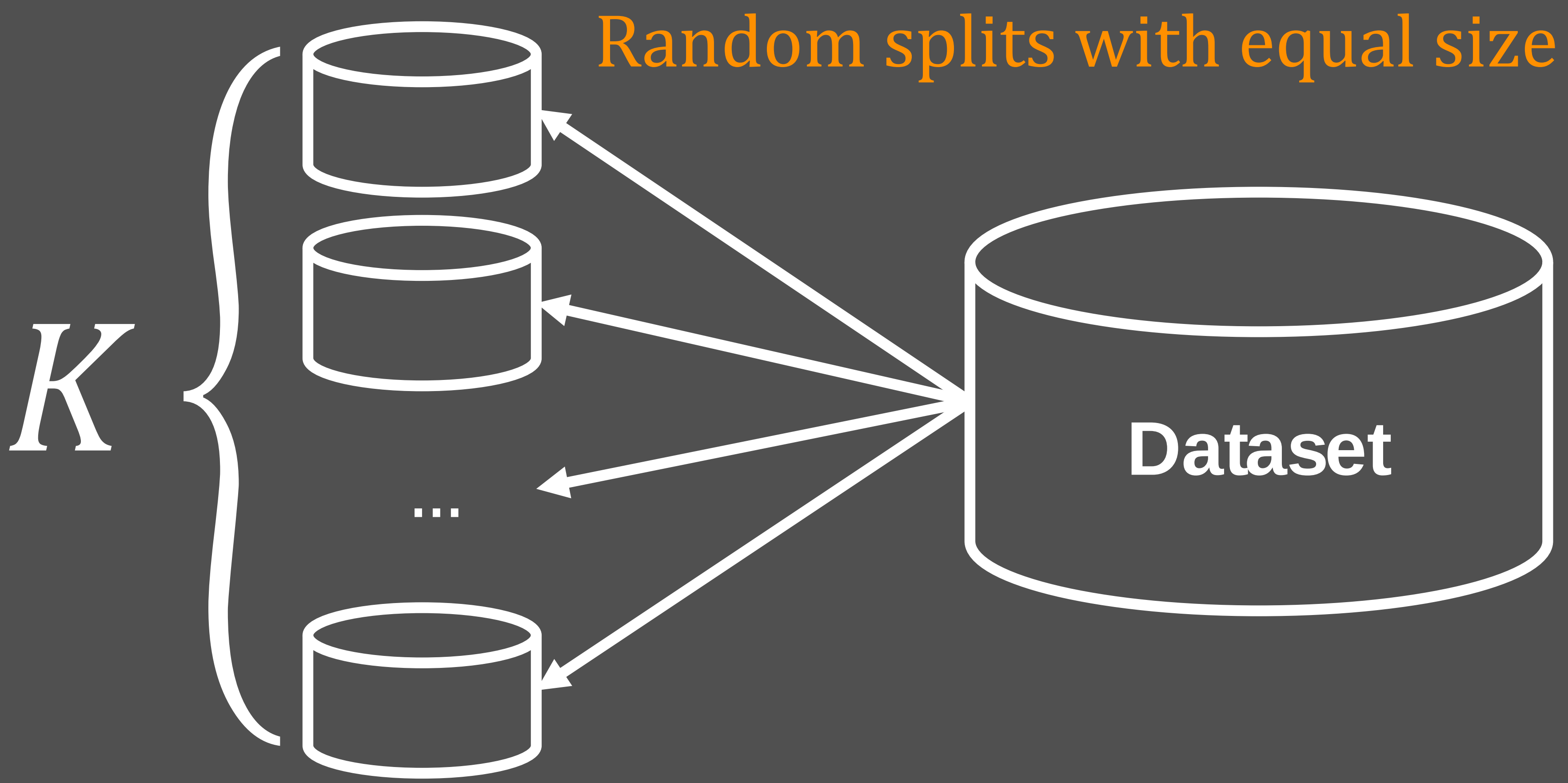
Validation


Randomize!!!
Dataset
Validation-set
Train
Train-set
Tuning
Test-set

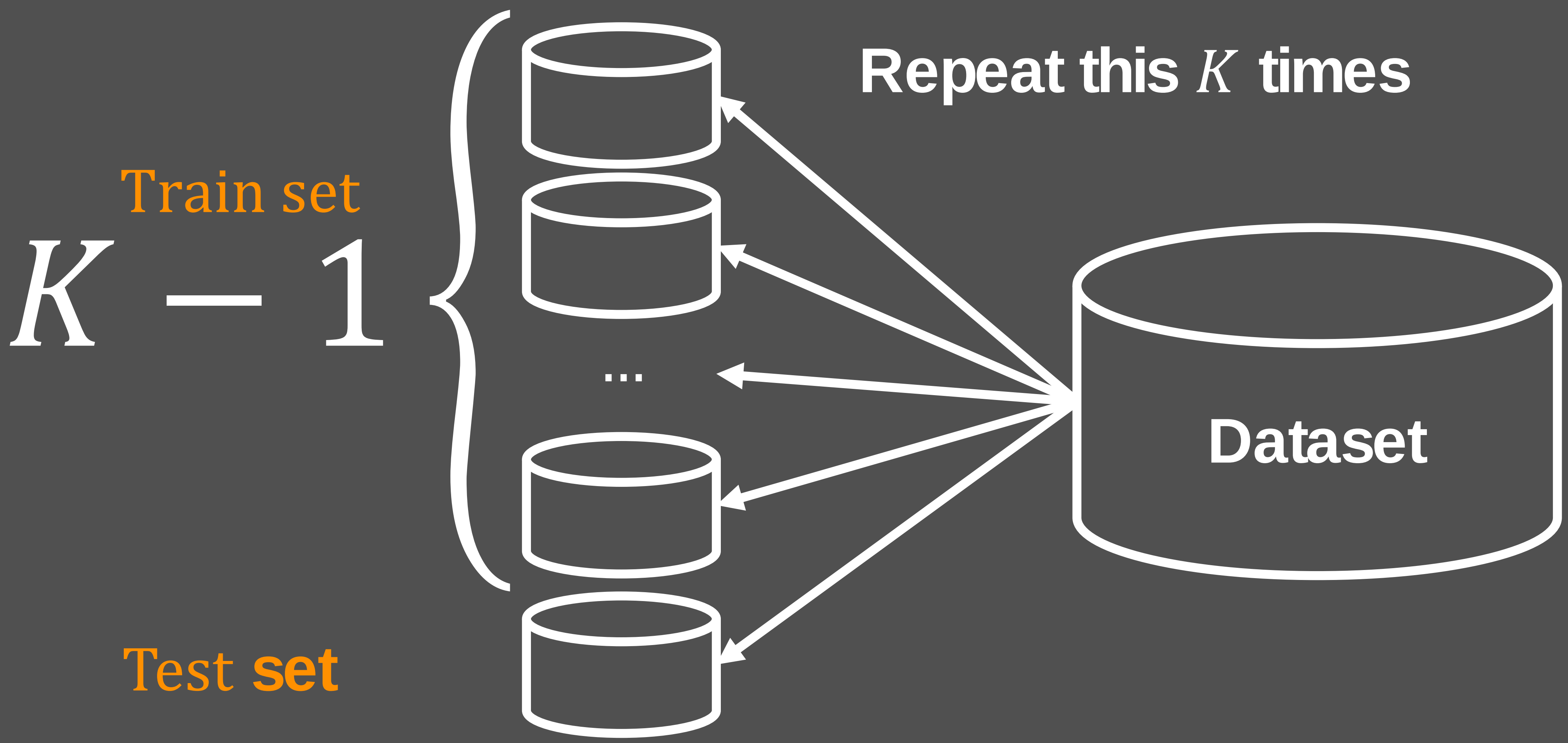
K-Fold Cross Validation



K-Fold Cross Validation



K-Fold Cross Validation



K-Fold Cross Validation

Randomize dataset
Split whole dataset into K datasets (folds)
sum_performance = 0
for $i = 0$ in range(K){
    testset = fold(i)
    trainset=fold($i \neq 0$)
    train(trainset)
    perfomance = test(testset)
    sum_performance+=performance
}
sum_performance /= K

Randomize dataset

K-Fold Cross Validation

Randomize dataset

Split whole dataset into K datasets (folds)

K-Fold Cross Validation

Randomize dataset

Split whole dataset into K datasets (folds)

sum_performance = 0

for $i = 0$ in range(K){

}

K-Fold Cross Validation

Randomize dataset

Split whole dataset into K datasets (folds)

sum_performance = 0

for $i = 0$ in range(K){

testset = fold(i)

}

K-Fold Cross Validation

Randomize dataset

Split whole dataset into K datasets (folds)

sum_performance = 0

for $i = 0$ in range(K){

testset = fold(i)

trainset=fold($i \neq 0$)

}

K-Fold Cross Validation

Randomize dataset

Split whole dataset into K datasets (folds)

sum_performance = 0

for $i = 0$ in range(K){

testset = fold(i)

trainset=fold($i \neq 0$)

train(trainset)

perfomance = test(testset)

}

K-Fold Cross Validation

Randomize dataset
Split whole dataset into K datasets (folds)
sum_performance = 0
for $i = 0$ in range(K){
    testset = fold(i)
    trainset=fold($i \neq 0$)
    train(trainset)
    perfomance = test(testset)
    sum_performance+=performance
}

K-Fold Cross Validation

Randomize dataset
Split whole dataset into K datasets (folds)
sum_performance = 0
for $i = 0$ in range(K){
testset = fold(i)
trainset=fold($i \neq 0$)
train(trainset)
perfomance = test(testset)
sum_performance+=performance
}
sum_performance /= K

K-Fold Cross Validation

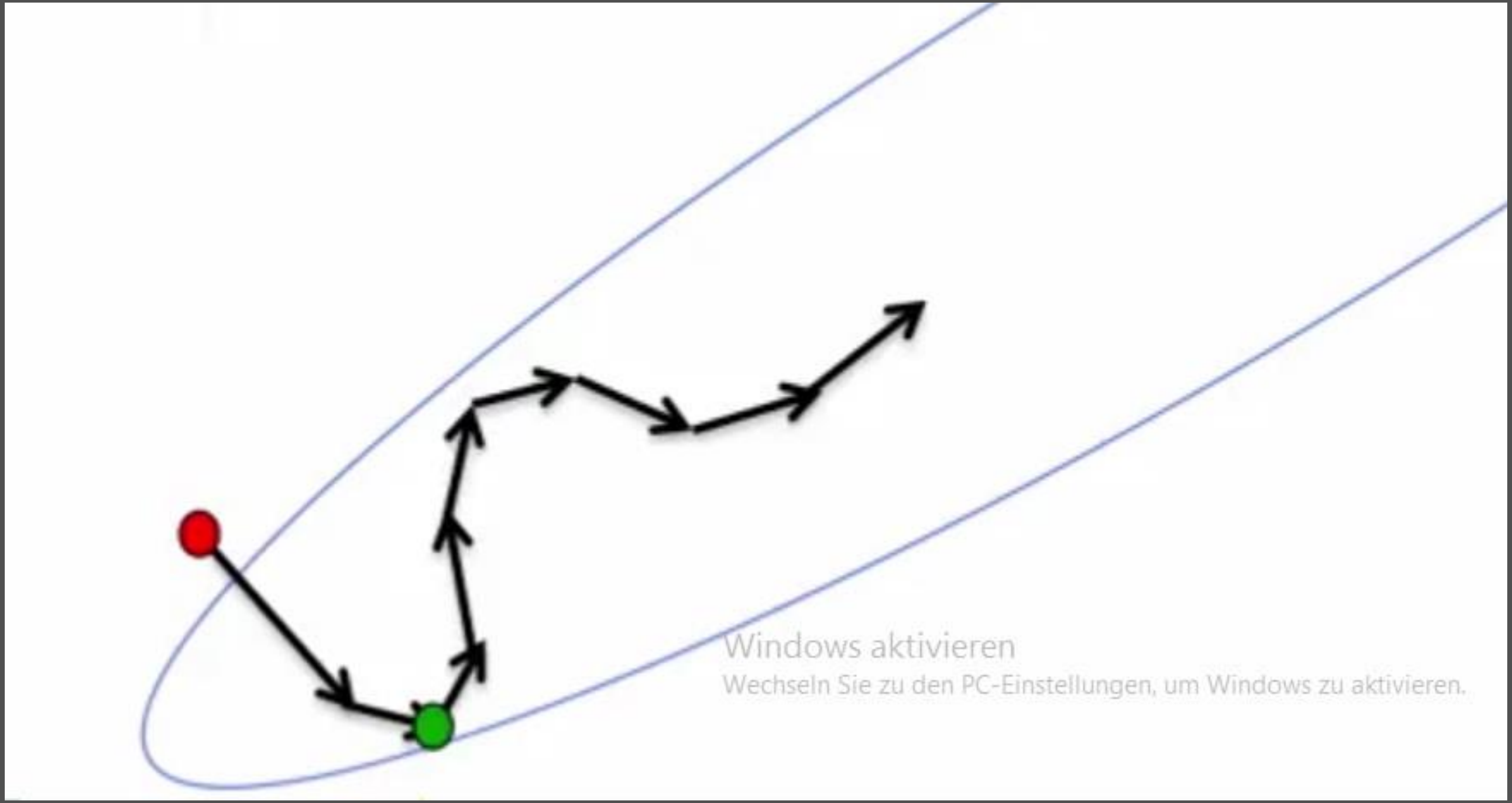
- Recommendation:

  K=10 (Gold Standard)

- Special Case $K = \#$Examples:

  Leave One Out Cross Validation (LOOCV)

Momentum


Windows aktivieren
Wechseln Sie zu den PC-Einstellungen, um Windows zu aktivieren.

Momentum

Usual Weight Update:

$$\theta := \theta - \alpha \frac{\partial E(\theta)}{\partial \theta}, \Delta\theta = -\alpha \frac{\partial E(\theta)}{\partial \theta}$$

$$\theta := \theta + \Delta\theta$$

Usual Weight Update:

$$\theta := \theta - \alpha \frac{\partial E(\theta)}{\partial \theta}, \Delta\theta = -\alpha \frac{\partial E(\theta)}{\partial \theta}$$

$$\theta := \theta + \Delta\theta$$

Momentum:

$$\Delta\theta_t := -\alpha \frac{\partial E(\theta)}{\partial \theta} + \varepsilon * \Delta\theta_{t-1}, 0 < \varepsilon \leq 1$$

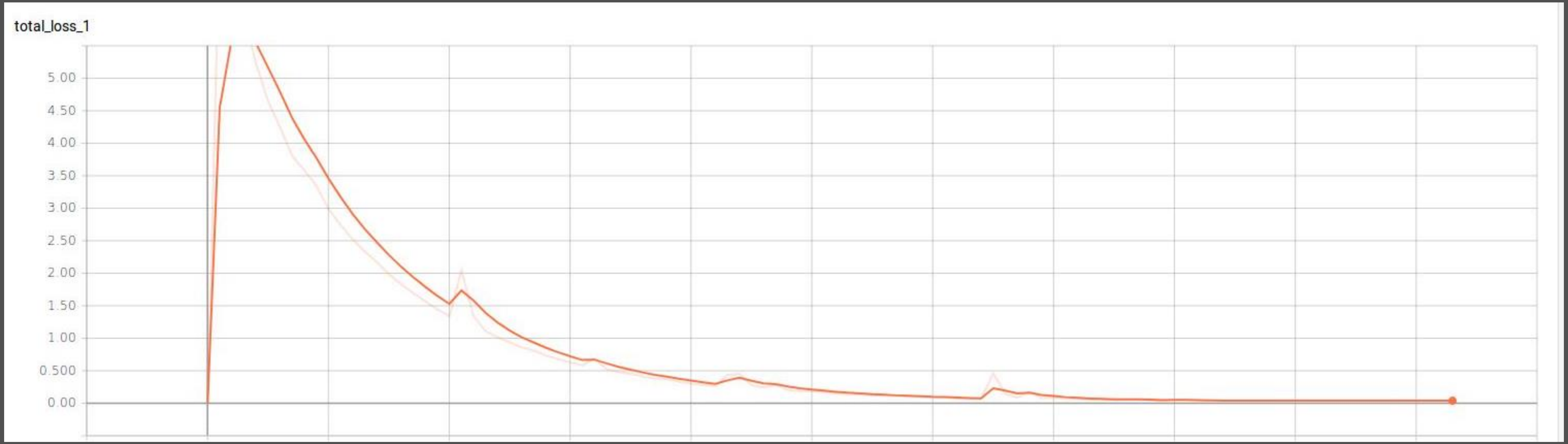
$$\theta_{t+1} = \theta_t + \Delta\theta_t$$

Momentum:

- Faster convergence
- Smooth out variance of gradients
- Use Decay for Momentum ε
- Find good value and decay through cross validation

Rate performance of neural net

- High bias / Underfitting
- High variance / Overfitting



Dropout

- Strategy to avoid overfitting
- Randomly exclude some neurons from training process
  - Exclude from forward pass
  - Excude from backward pass

tensorboard

- Create a tf.summary
  - Store important variables (Scalars, Histograms, Images, …)
- Write tf.summary to file
- Open tensorboard with events-file as target

tensorboard examples:

- Tensorflow documentation
- Internet
- Star Recognition (Repository)
- DQN (Repository)


100%

CNN example

- Start tensorboard to debug the training process
- Implement Momentum
- Use a data augmentation technique to make generalization more robust
- Implement train, test, validation split
- Download weights and image of one of the five stars to test prediction

Start tensorboard to debug the training process

- Start the training process
- Start tensorboard in directory with event-file

Hints: Implement Momentum

- **Implement tf.train.MomentumOptimizer**
- **Search for „optimizer"** in cnn.py

Use a data augmentation technique to make generalization more robust

- Change .tfrecord writing to save RGB-Images
- Implement RGB to Grayscale in training process
- Normalize data before the training process!!!
- Use tf.image.random… operations to augment the input data

Implement train, test, validation split

- Create a new python script
- Count the number of examples
- Randomize the examples
- Create variables for the train, test and validation proportion
- Write .tfrecord files
    - Have a look at write_tfrecord.py

Download weights and image of one of the five stars to test prediction

- Works only if RGB to Grayscale conversion is implemented
- Solution will be pushed at the end of the lesson

- **Data (if not yet downloaded):** https://nextcloud.mirevi.medien.hs-duesseldorf.de/index.php/s/kPXwJiac7vTQVeu
- **Pretrained Weights:** https://nextcloud.mirevi.medien.hs-duesseldorf.de/index.php/s/L6Y6tnD3PpANKmr
- **Repository:** https://github.com/mati3230/modalg181
- **Read:** https://www.tensorflow.org/tutorials/deep_cnn