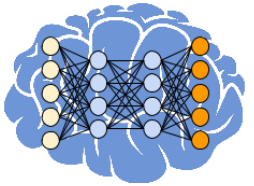
https://bit.ly/3pyXRii ← Tutorial web page



RWTH AACHEN UNIVERSITY

Introduction to Deep Learning

Deep Learning Week, Uppsala, 21.3.2022

- Basic Methods & Techniques
- Deep Learning Frameworks
- Physics Examples and further Applications

Jonas Glombitza

RWTH Aachen

Outline

Monday

I. Deep Learning Basics – short hands-on
II. Convolutional Neural Networks – short hands-on

BREAK

III. Introduction to Generative Adversarial Networks (GANs)
IV. Tutorial: Implementation of GANs

Wednesday

V. Latest developments & advanced techniques
- Wasserstein GANs

VI. Application in physics research
- Simulation acceleration, domain adaption

BREAK

VII. Tutorial: Implementation of Wasserstein GANs

Feel free to ask questions during the seminar!
Just “raise” your hand...

Hands On: Neural Networks

III. Physikalisches Institut A | RWTH AACHEN UNIVERSITY

Machine Learning Basics

- interactive neural network training

ML frameworks and the design of simple neural networks

- implementation of a vanilla neural net

Convolutional neural networks (CNN)

- implementation of a convolutional neural network

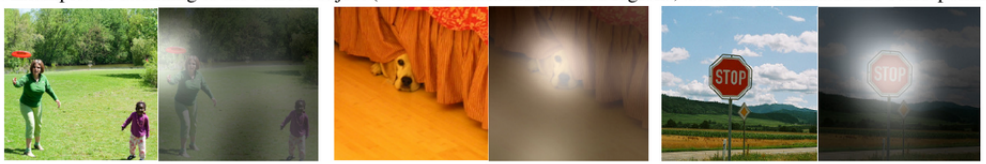
Set up & Requirements: → https://bit.ly/3pyXRii

- we will use **Jupyter Notebooks** and **Keras** / TensorFlow
- we will use **Google Colab** → Google Account required

Deep Learning

- Machine Learning Basics
- Neural Networks
  - Objective function / loss
  - Layout
  - Optimization

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)



A woman is throwing a frisbee in a park. A dog is standing on a hardwood floor. A stop sign is on a road with a mountain in the background.

ArXiv: 1502:03044

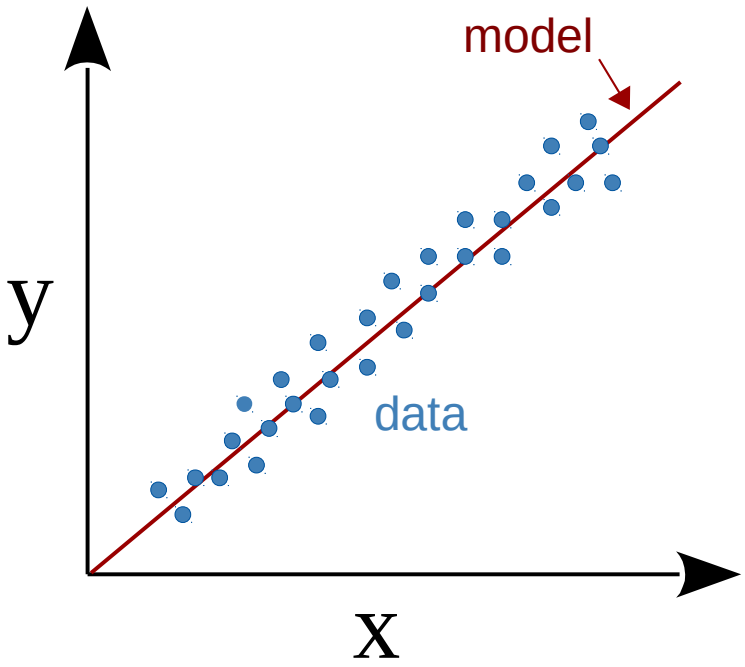
KÜNSTLICHE INTELLIGENZ

Schlau in zwei Stunden

VON ALEXANDER ARMBRUSTER - AKTUALISIERT AM 27.09.2017 - 11:41

Artificial Intelligence - "The effort to automate intellectual tasks normally performed by humans"

Machine Learning – Regression



- Data: $\{x_i, y_i\},\ i = 1, ..., N$
- Define model:
  $y_m(x, \theta) = Wx + b$ with free parameters $\theta = (W, b)$
- Define **objective function** (loss/cost)

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} [y_m(x_i, \theta) - y_i]^2$$

- Train model (minimize objective) $\hat{\theta} = argmin[J(\theta)]$
  - Optimize set of free parameters $\theta = (W, b)$
    e.g., use *gradient descent*

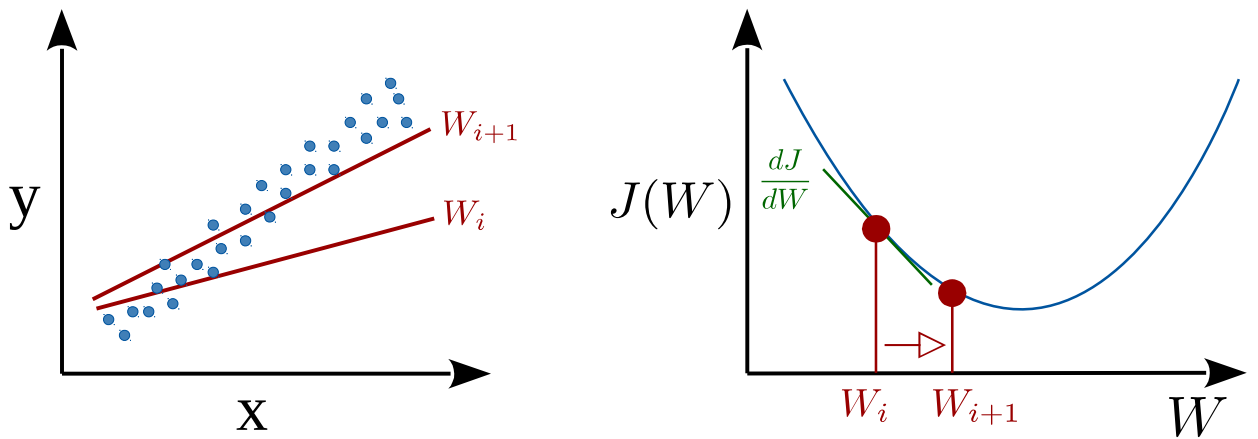
Gradient Descent

- Minimize objective function $J(\theta)$ by updating θ in **opposite** direction of gradient iteratively

gradient:	$dJ/d\theta$
stepsize = **learning rate**:	α

$$\tilde{\theta} \rightarrow \theta - \alpha \frac{dJ}{d\theta}$$

- Example: linear regression with mean squared error



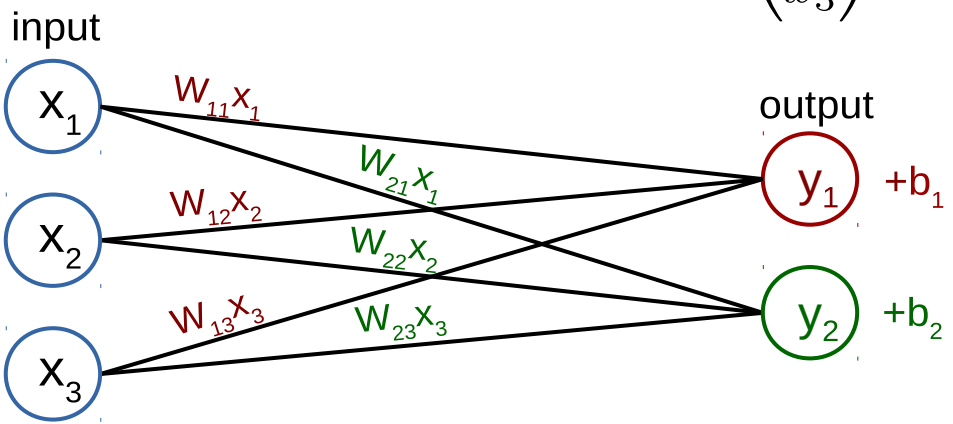
Multidimensional Linear Models

- Predict multiple outputs $\mathbf{y} = (y_1, ..., y_n)$ from multiple inputs $\mathbf{x} = (x_1, ..., x_n)$ using linear function $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$

Note: We define linear = affine in this course

- Example: $x \in \mathbb{R}^3,\ y \in \mathbb{R}^2$

$$\begin{pmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$



Non-Linear Network Models

$\mathbf{Wx} + \mathbf{b}$ only describes linear models

- Use network with several linear layers:

$$h' = W^{(1)}x + b^{(1)}$$

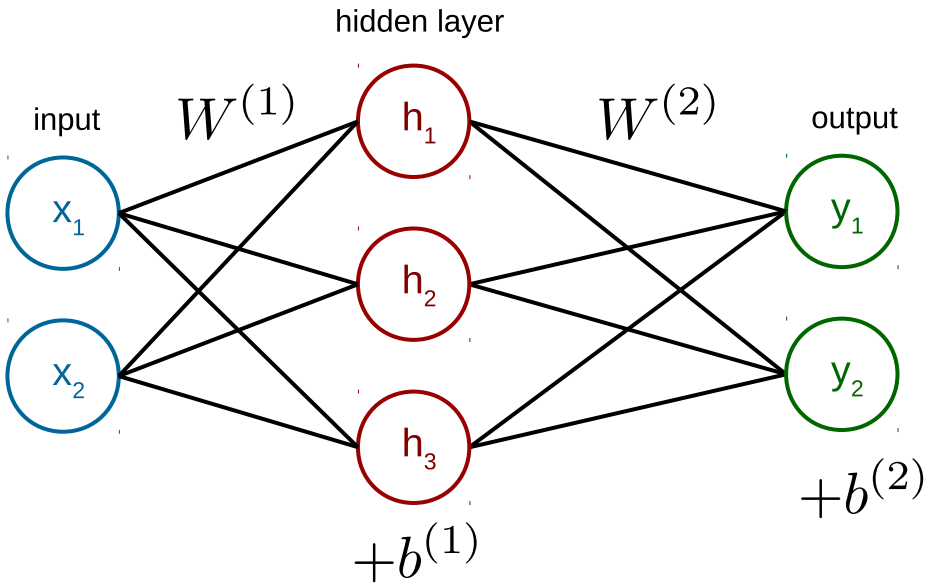
$$y = W^{(2)}h' + b^{(2)}$$

- Model is still linear!

$$y = W^{(2)}\left(W^{(1)}x + b^{(1)}\right) + b^{(2)}$$

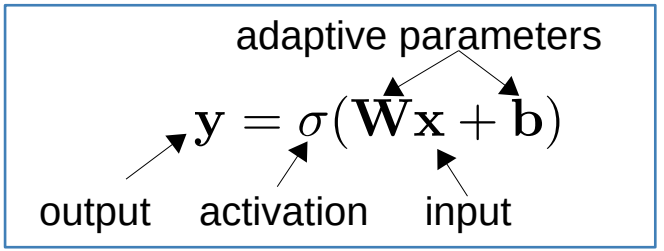
$$y = \underbrace{W^{(2)}W^{(1)}}_{W}x + \underbrace{W^{(2)}b^{(1)} + b^{(2)}}_{b}$$

- Solution: Apply non-linear activation σ to each element $\longrightarrow h = \sigma(h') = \sigma(Wx + b)$



Activation Functions

- Using an activation function the layer becomes a non linear mapping
  - Allows for stacking several layers



Examples

- **Rectified Linear Unit**

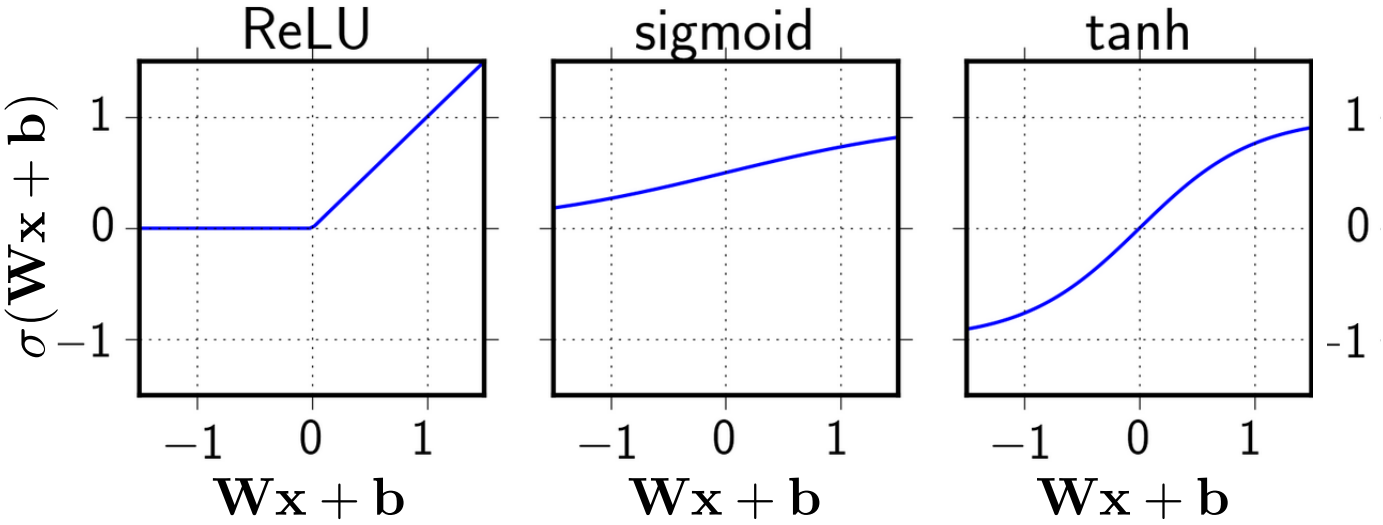
  $\sigma(x) = \max(0, x)$

- **Sigmoid**

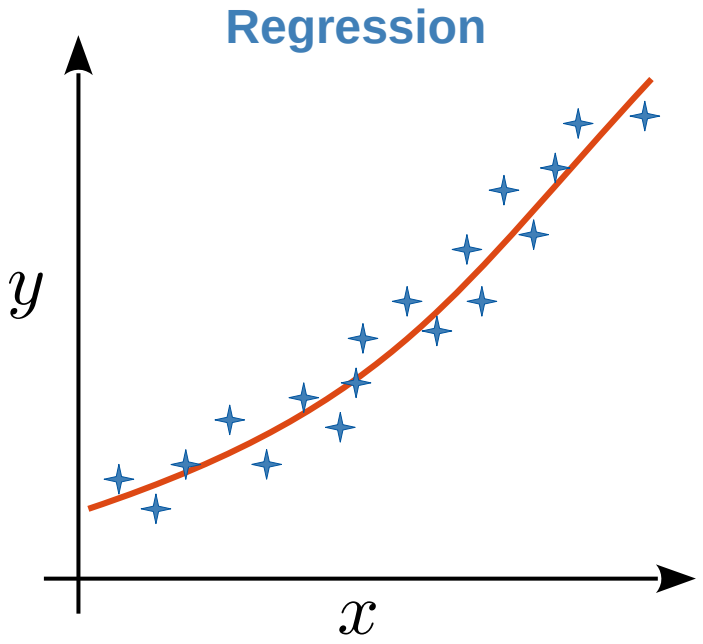
  $\sigma(x) = \frac{1}{1 + e^{-x}}$

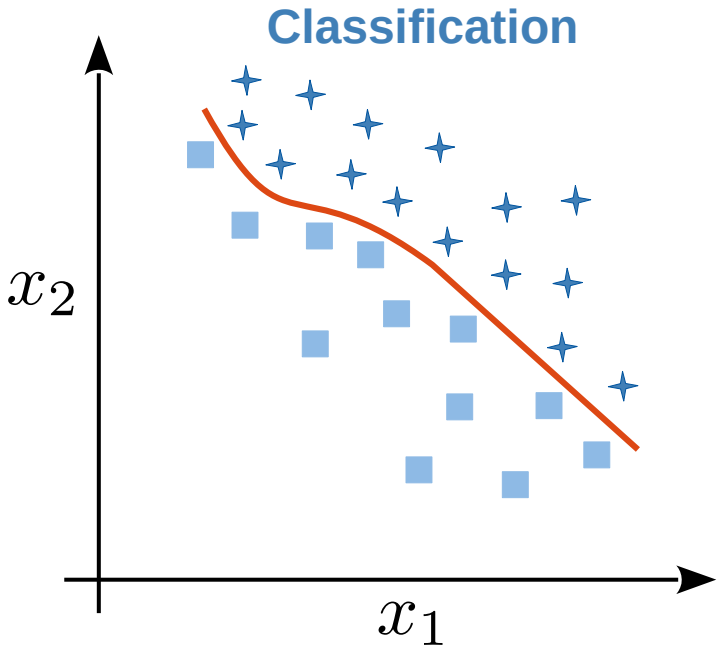
- **Hyperbolic tangent**

  $\sigma(x) = \frac{e^{+2x} - 1}{e^{-2x} + 1}$



Machine Learning Tasks

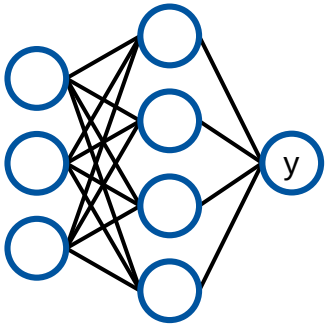




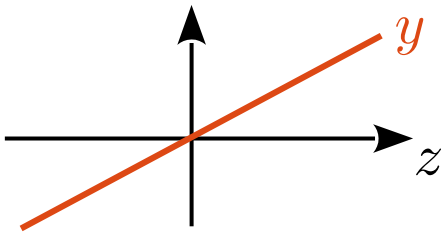
- Regression: Predict continuous label y
- Classification: Separate into different classes (cats, dogs, airplanes, …)
- Can sometimes convert to the other

Classification vs. Regression

Regression



Linear



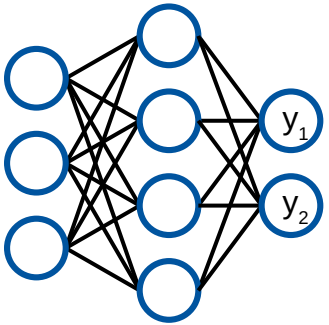
no activation function

Minimize mean-squared-error

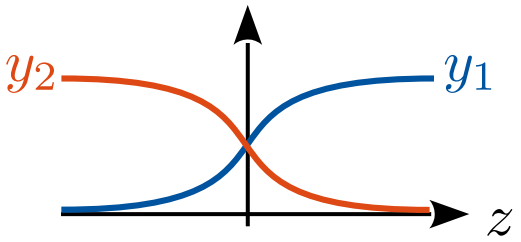
$$J(\theta) = \frac{1}{n}\sum_{i}^{n}[y_i - y_m(x_i)]^2$$

average over n samples =: batch

Classification



Softmax



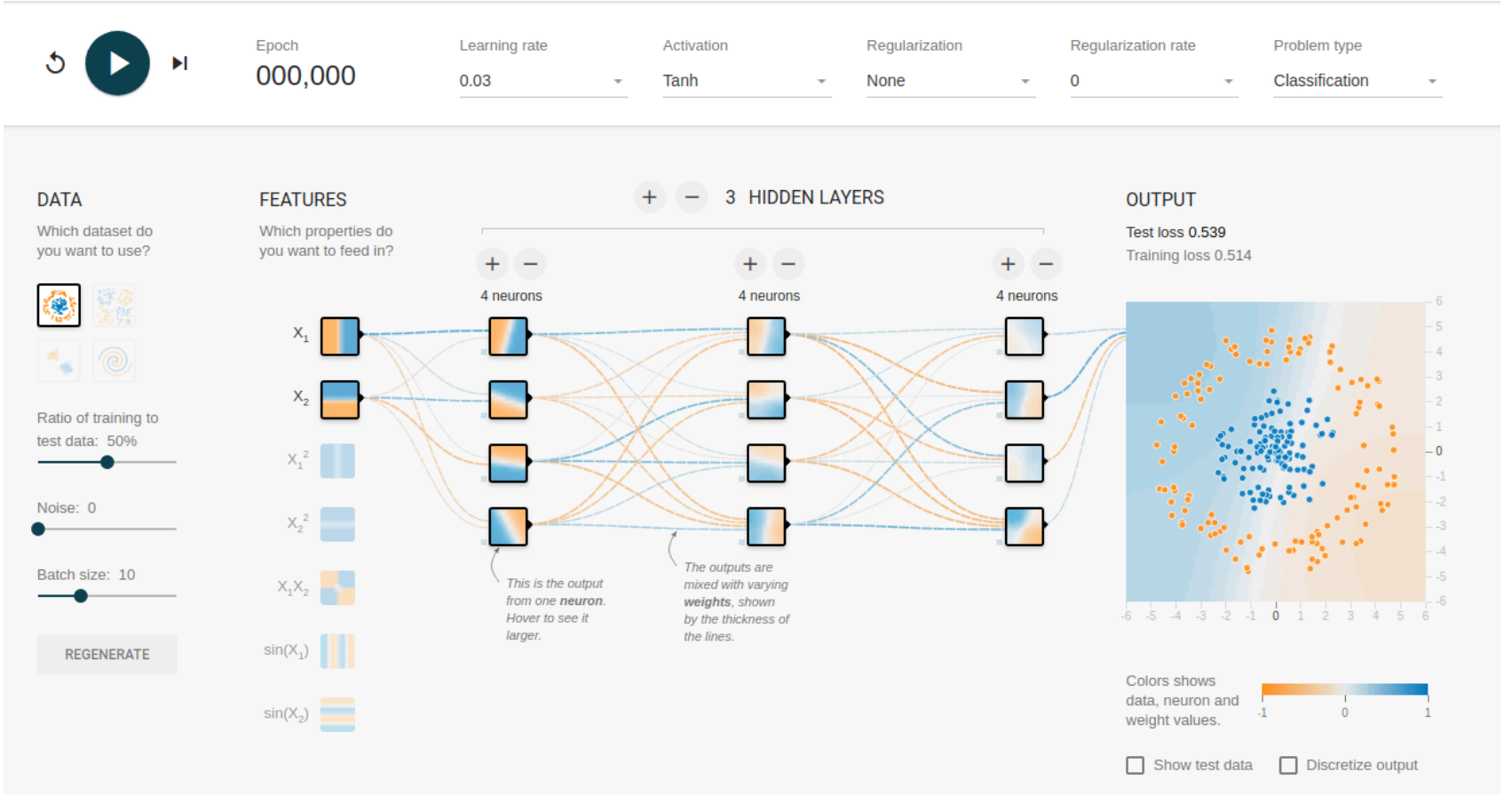
$$y_j(z) = \frac{e^{z_j}}{\sum_i e^{z_i}}$$

Minimize cross entropy

$$J(\theta) = -\sum_{i}^{n} y_i \log[y_m(x_i)]$$

average over n samples =: batch

Example Training



Initialization

- **Weights need different (random) initial values → symmetry breaking**
- Scale of weights very important
  - Too large → exploding signals & gradients
  - Too small → vanishing signals & gradients

} **No learning!**

- For forward pass in each layer:
  $$Var[x_l] = 1$$
- For Backward pass in each layer:
  $$Var[\Delta x_l] = 1$$

➢ Depends from activation function and number of in and outgoing nodes

$$Var[W] = \frac{2}{n_{\text{in}} + n_{\text{out}}}$$ → For tanh Glorot, Bengio

$$Var[W] = \frac{2}{n_{\text{in}}}$$ → For ReLU He et al.

- Can be sampled from Gaussian or uniform distribution (Var. scaled by factor of 3)

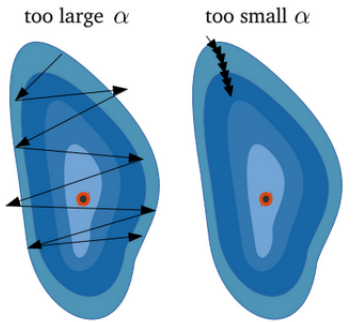
Learning Rate

Learning rate α determines speed of training

- High rate $\rightarrow$ poor or none convergence behavior
- Small rate $\rightarrow$ Very slow training or none at all
- Typical learning rate: $\alpha = 10^{-3}$
- Advanced: reduce learning rate on loss plateau
  - increase sensitivity to smaller scales

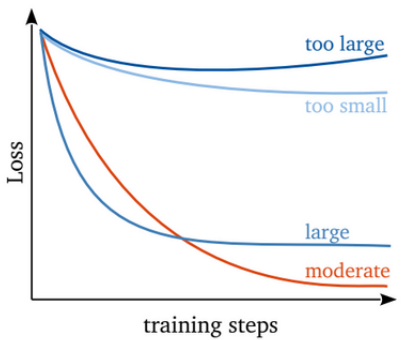
$$\theta \rightarrow \theta - \alpha \frac{dJ}{d\theta}$$

Learning rate



Stochastic Gradient Descent (SGD)

- Use small subset (**batch** ~ 16 - 100 samples) of data set
  - 1 **epoch** =: full pass through training data set
  - Reduces computational effort, improved generalization



TensorFlow Playground - 10 Minutes

III. Physikalisches Institut A | RWTH AACHEN UNIVERSITY

Checkerboard task

- Choose the Checkerboard data set (XOR)
- What do you observe when changing the activation function?
- What do you see when inspecting the features of deeper layers?
- Choose the ReLU activation:
  - What is the minimum number of nodes / layers needed to solve the task?

Bonus:

- Solve the Spiral / Swiss roll task

Open the example at:

https://playground.tensorflow.org/

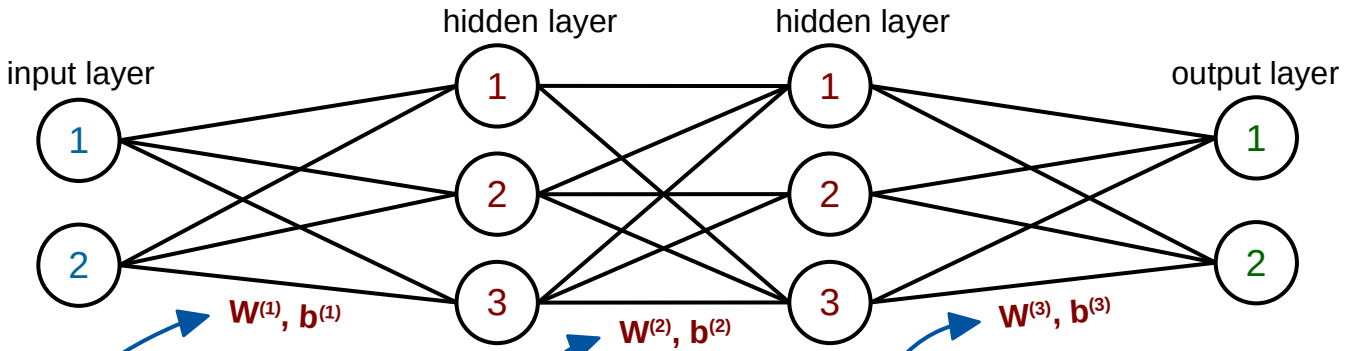
or visit https://bit.ly/3pyXRii

Deep Neural Networks

Feature Hierarchy: each new layer extract more abstract information of the data.

Probabilistic Mapping: learns to combine the extracted features

Train model (to find $\theta = \{W_i, b_i\}$ that minimizes objective) is automatic process.



adaptive parameters

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

output activation input

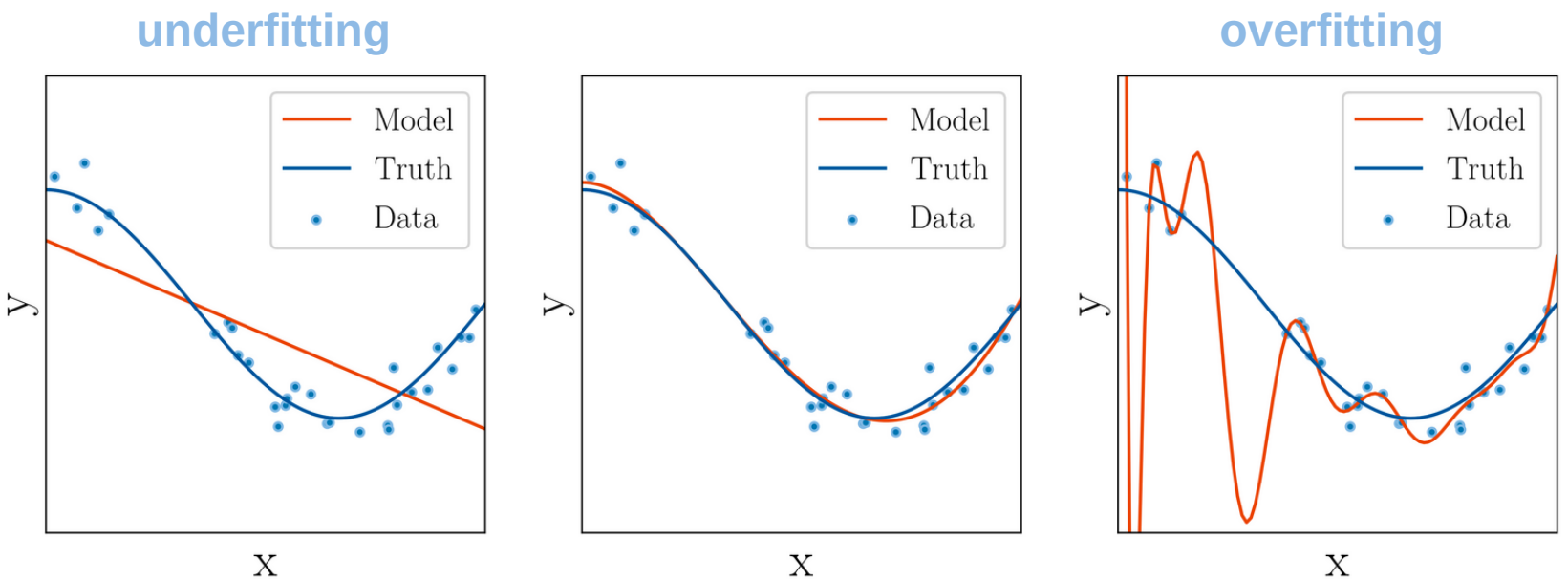
$$\text{objective}: \quad J(\theta) = \sum_i [y_m(x_i, \theta) - y_i]^2$$

$$\text{optimization}: \quad \frac{dJ}{d\theta} \to 0$$

$$\tilde{\theta} \to \theta - \alpha \frac{dJ}{d\theta}$$

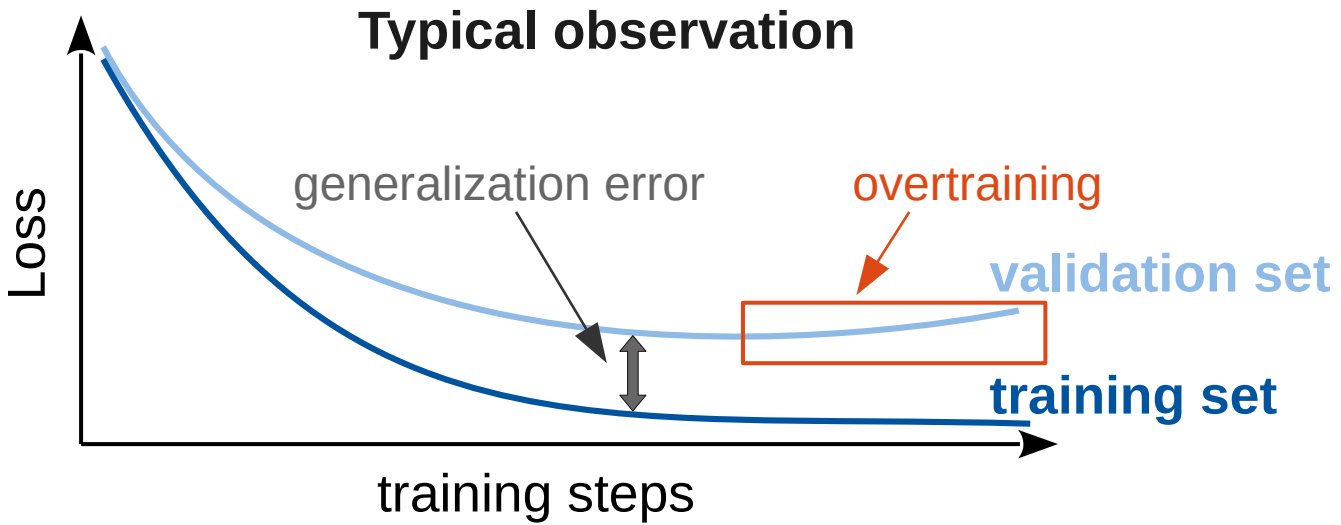
Under- and Overfitting

- Challenging to find a good network design
- Under-complex models show bad performance
- complex models are prone to overfitting
  - Model memorizes training data under loss of generalization performance



Under- and Overtraining

- During training monitor the loss separately for training and validation set

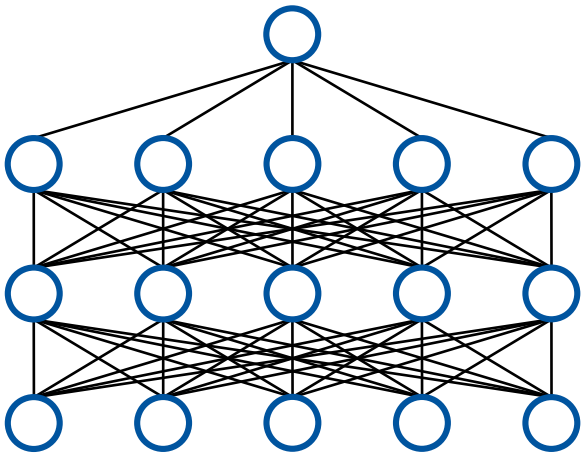


Strategy → Divide you data into three data sets:
Training set: to train the network
Validation set: to monitor and tune the training (training of hyperparameter)
Test set: to estimate final performance. Use only **once!**

Dropout

Randomly turn of fraction p_{drop} of neurons in each training step

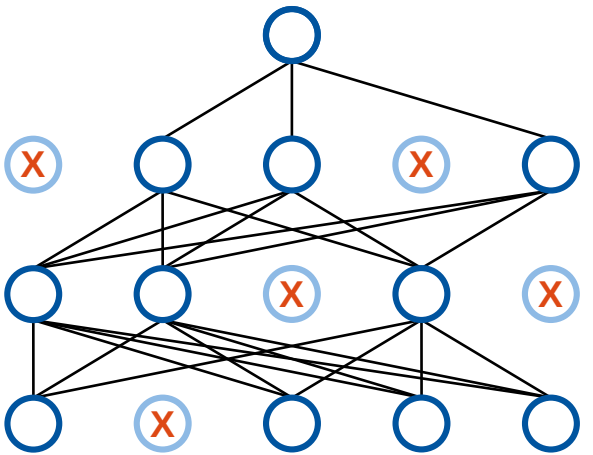
standard network



Typical fraction

$$0.2 < p_{drop} < 0.5$$

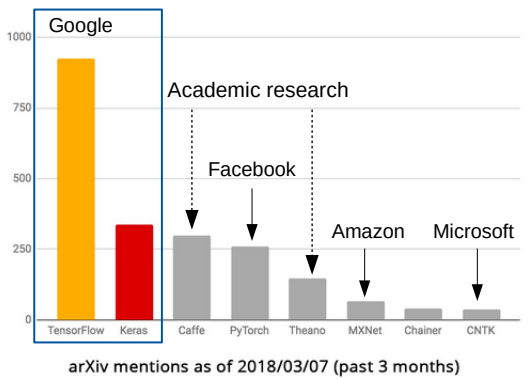
dropout applied

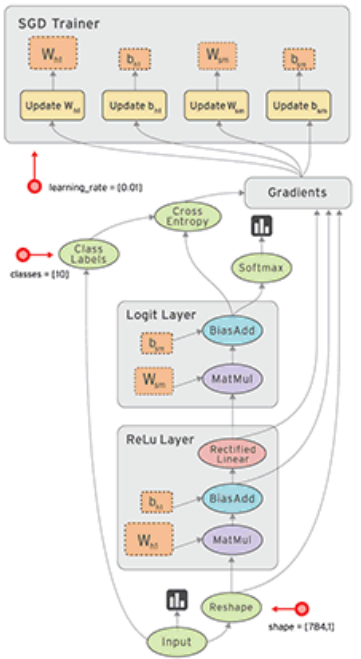


- Adds noise to process of feature extraction
- Force network to train redundant representations
- During training: each activation is scaled with inverse dropout probability
- During validation: no dropout applied → large ensemble of “sub models”

Keras

- Will use keras in this tutorial (TensorFlow backend) - https://keras.io
- High-level neural networks API, written in Python
- Concise syntax with many reasonable default settings
- Useful callbacks for monitoring the training procedure
- Nice Documentation & many examples and tutorials
- Supports: CPUs, TPUs and **GPUs**





How to train your Model?

I. **Define** Model
- Add layers, nodes, regularization, activation functions, ….)

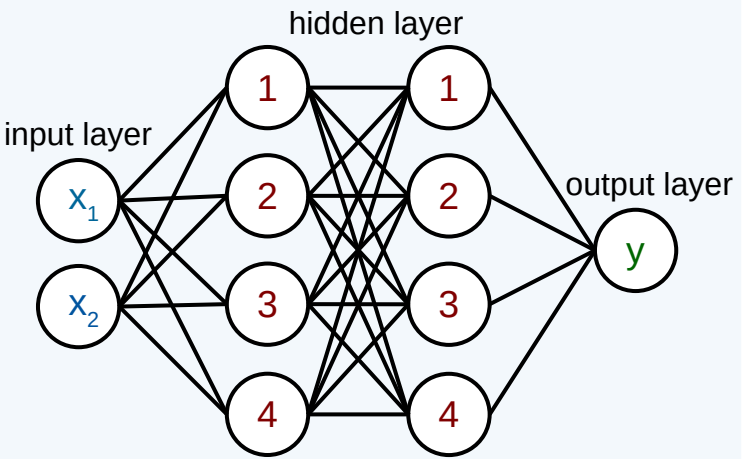
II. **Compile** Model
- Set Loss, optimizer settings and useful metrics

III. **Fit** Model
- Set number of iterations and train model on given data

```
from tensorflow import keras
layers = keras.layers
models = keras.models

# setup and train a 3-layer regression network with Keras
model = models.Sequential()
model.add(layers.Dense(4, activation='relu', input_dim=2))
model.add(layers.Dense(4, activation='relu'))
model.add(layers.Dense(1, activation='tanh'))
model.compile(loss='MSE', optimizer='SGD', metrics=['accuracy'])
model.fit(xdata, ydata, epochs=200)
```

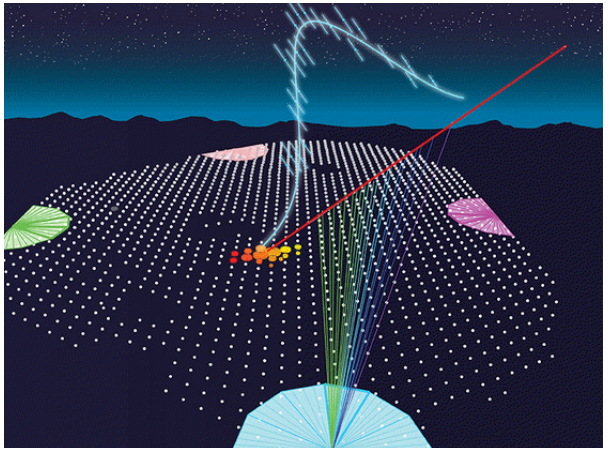
I
II
III



Air Shower Reconstruction

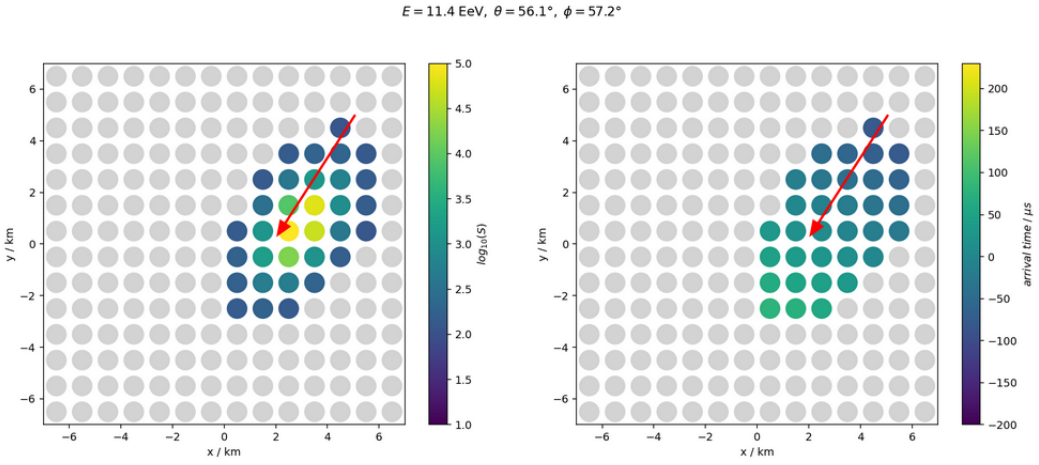
III. Physikalisches Institut A
RWTH AACHEN UNIVERSITY

- Cosmic-ray-induced air showers
  - 14 x 14 particle detectors, arranged in a Cartesian grid at a height of 1400 m
  - stations measure arrival time of the shower and the deposited energy



Task

- Create Google Account
- Reconstruct energy of the shower
  - footprint is 2D image
  - cannot directly be used as input → reshape to a vector with length (14 x 14 x 2 = 392)



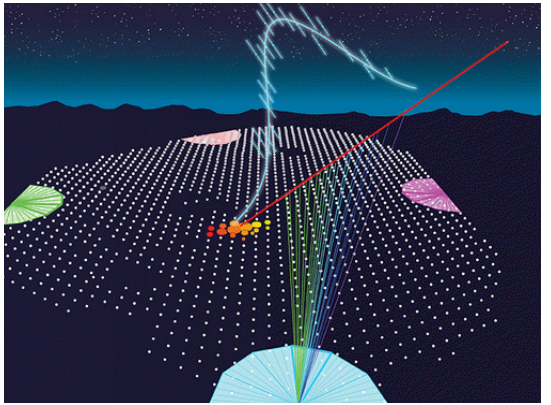
Air Shower Reconstruction - FCN

III. Physikalisches Institut A | RWTH AACHEN UNIVERSITY

Now: OPEN tutorial at:

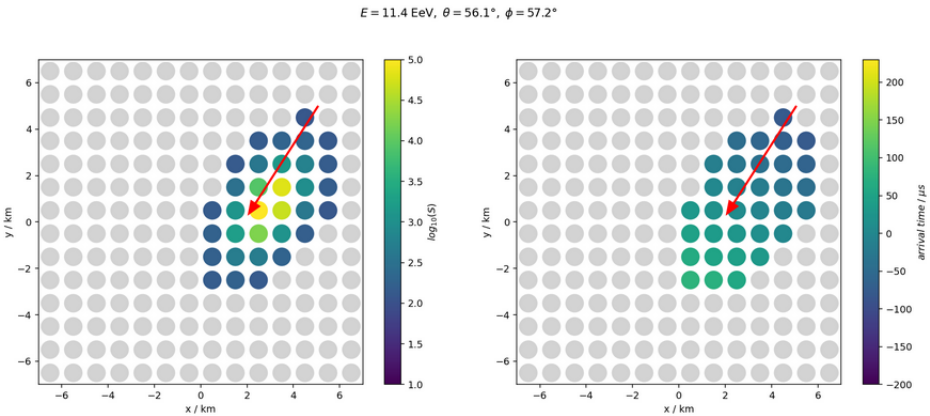
- https://github.com/jglombitza/tutorial_nn_airshowers
- or click and login

Open in Colab



Task

- Reconstruct energy of the shower
  - footprint is 2D image
  - cannot directly be used as input
    → reshape to a vector with length
    (14 x 14 x 2 = 392)
  - **Try to reach a resolution better than 4 EeV**

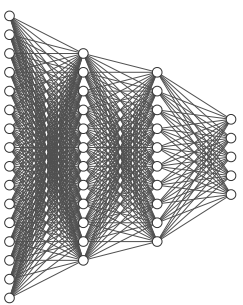


Results I

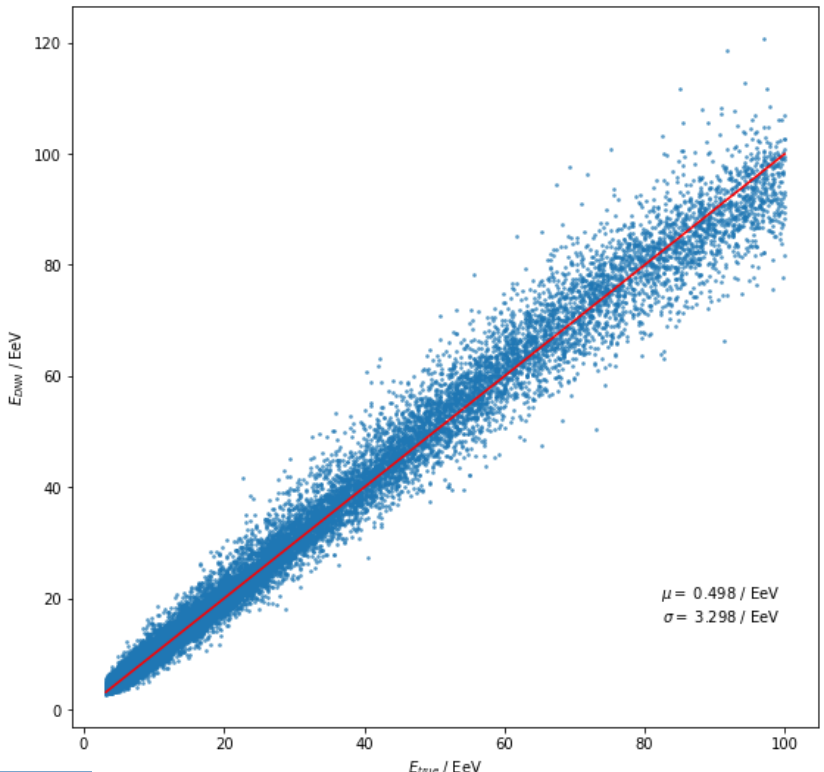
- Train fully-connected network as benchmark

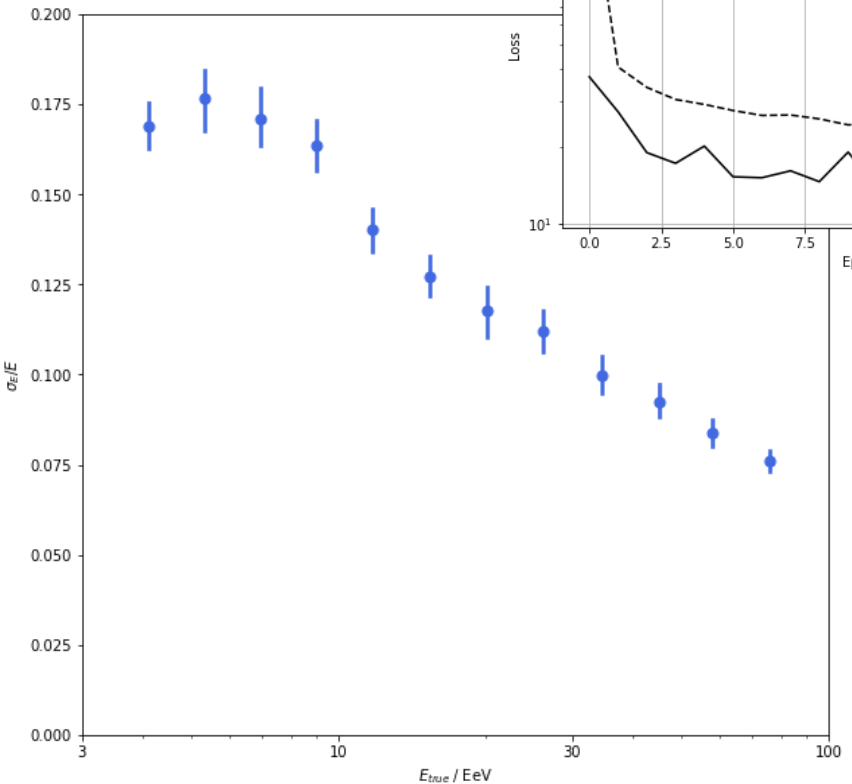
- Model – **add**:
  - additional layers
  - more nodes
  - regularization (Dropout)

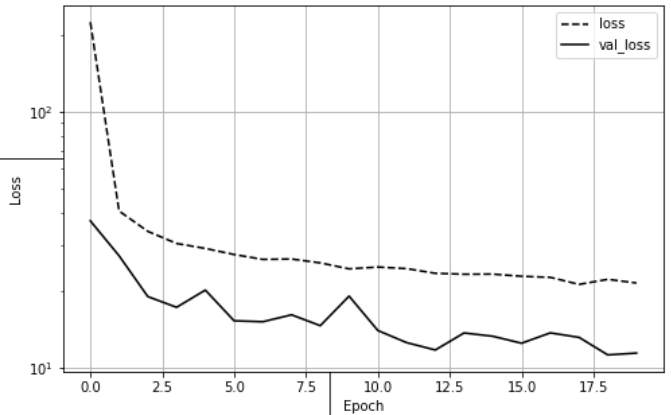
```
model = keras.models.Sequential()
model.add(layers.Flatten(input_shape=X_train.shape[1:]))
model.add(layers.Dense(100, activation="elu"))
model.add(layers.Dense(100, activation="elu"))
model.add(layers.Dense(100, activation="elu"))
model.add(layers.Dense(100, activation="elu"))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(1))
```



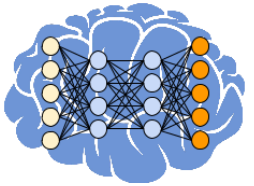
Results II


E_{DNN} / EeV
E_{true} / EeV
μ = 0.498 / EeV
σ = 3.298 / EeV


σ_E/E
E_{true} / EeV


loss
val_loss
Loss
Epoch

https://bit.ly/3pyXRii ← Tutorial web page

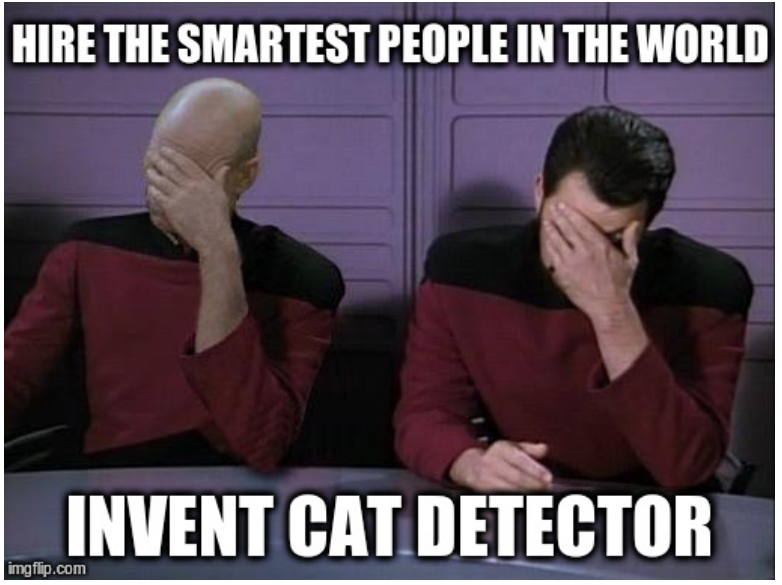


RWTH AACHEN UNIVERSITY

Introduction to Deep Learning

Deep Learning Week, Uppsala, 21.3.2022

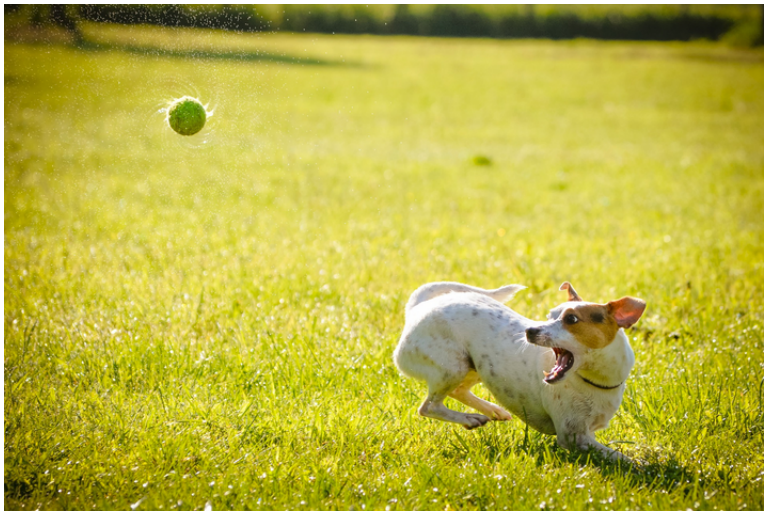
- Convolutional Neural Networks (CNNs)
- Basic Methods & Techniques



Jonas Glombitza

RWTH Aachen

Natural Images

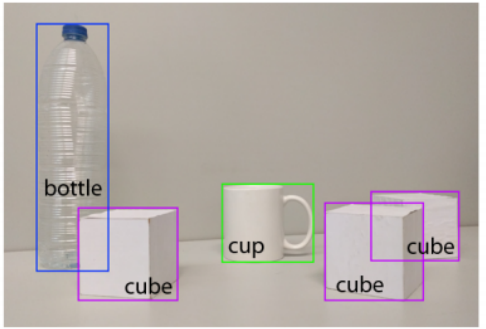


Automate task for humans, very challenging for machine learning models:

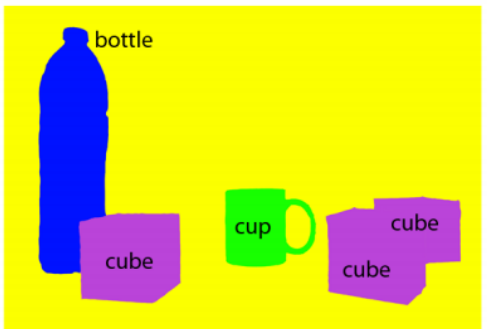
- High dimensional input (up to millions of pixels)
- Many possible classes depending on task
- Multiple variations
  - Viewing angle, light conditions, deformation, object variations, occlusions….

Computer Vision Tasks

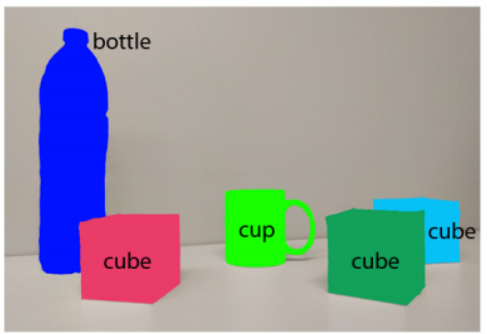
(a) Image classification



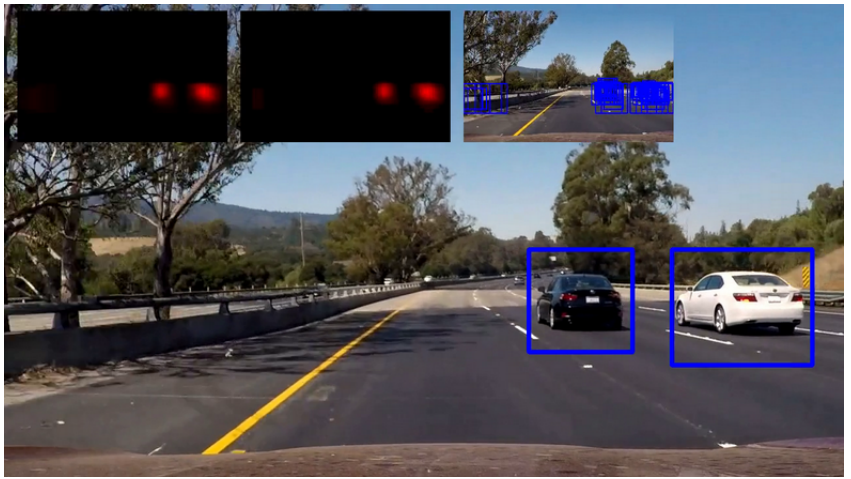
(b) Object localization



(c) Semantic segmentation

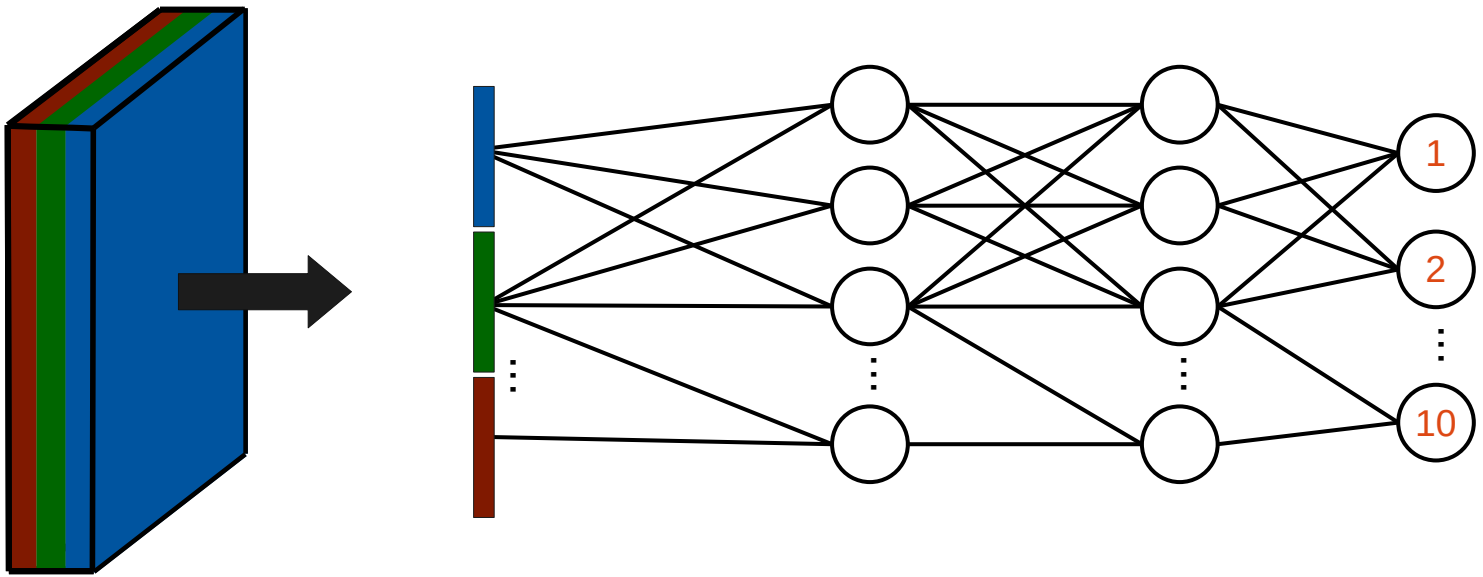


(d) Instance segmentation

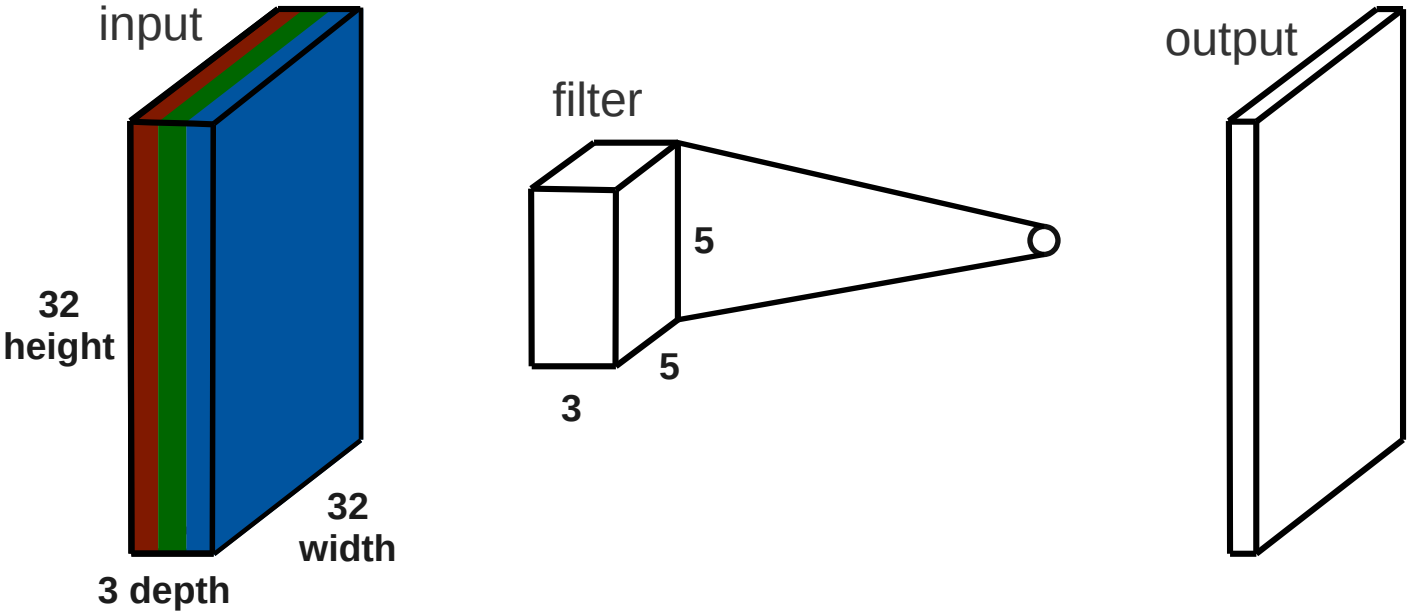


Fully Connected Network

- Input layer: Flatten image to 32 x 32 x 3 = 3072 vector
- Use fully connected network with some hidden layers, ReLU and dropout
- Output layer: 10 layer output with softmax

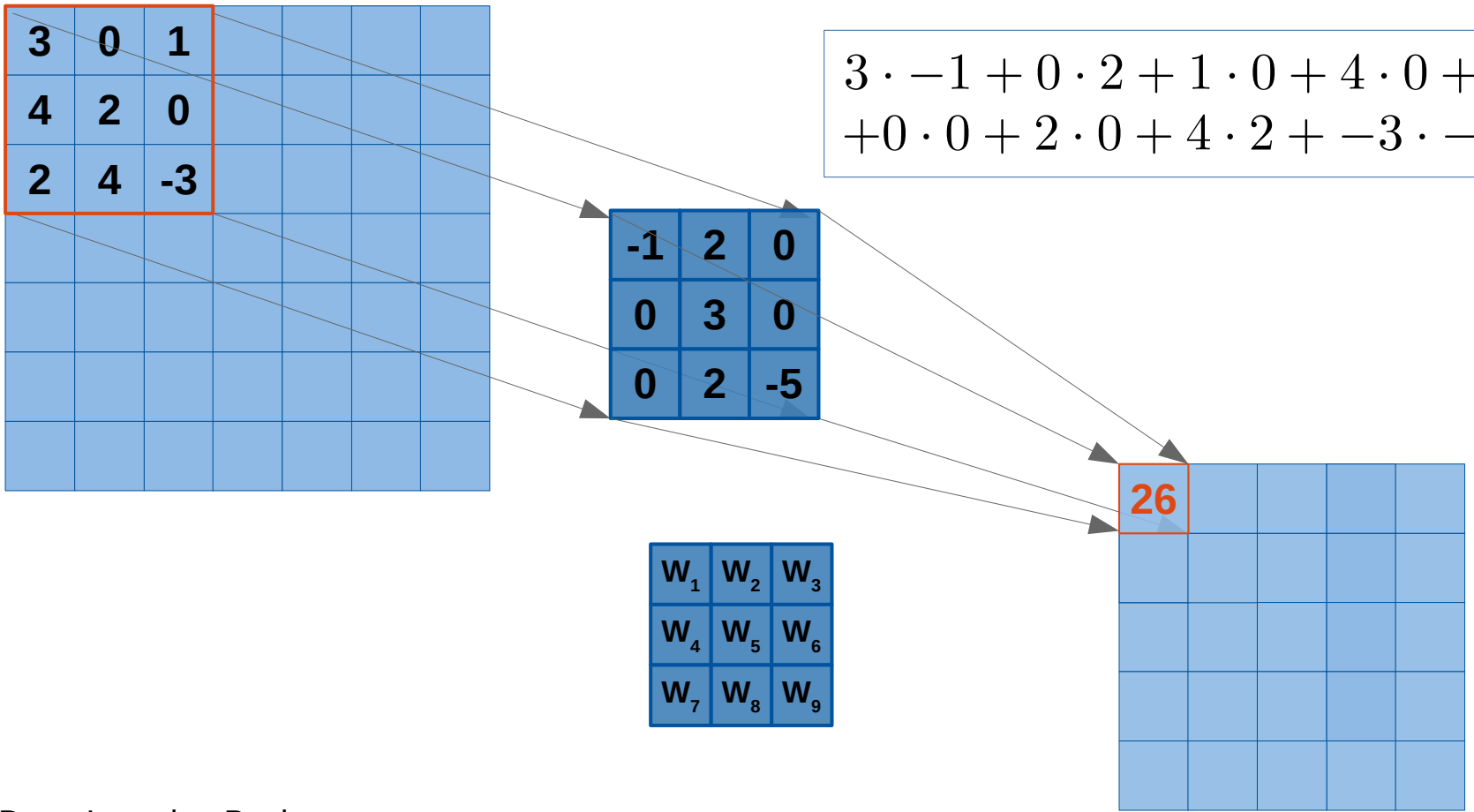


2D Convolutional Neural Networks



- Consider input volume (width x height x depth), e.g., 3 color channels
- Use convolutional filter with smaller width and height but same depth
- Slide filter over the entire volume and calculate linear transformation to get one output value for each position

Convolutional Operation



$$3 \cdot -1 + 0 \cdot 2 + 1 \cdot 0 + 4 \cdot 0 + 2 \cdot 3 + 0 \cdot 0 + 2 \cdot 0 + 4 \cdot 2 + -3 \cdot -5 = 26$$

Convolutional filters

hand-designed filters

Edge

-1	-1	-1
-1	8	-1
-1	-1	-1

Diagonal edge

-1	-1	2
-1	2	-1
2	-1	-1

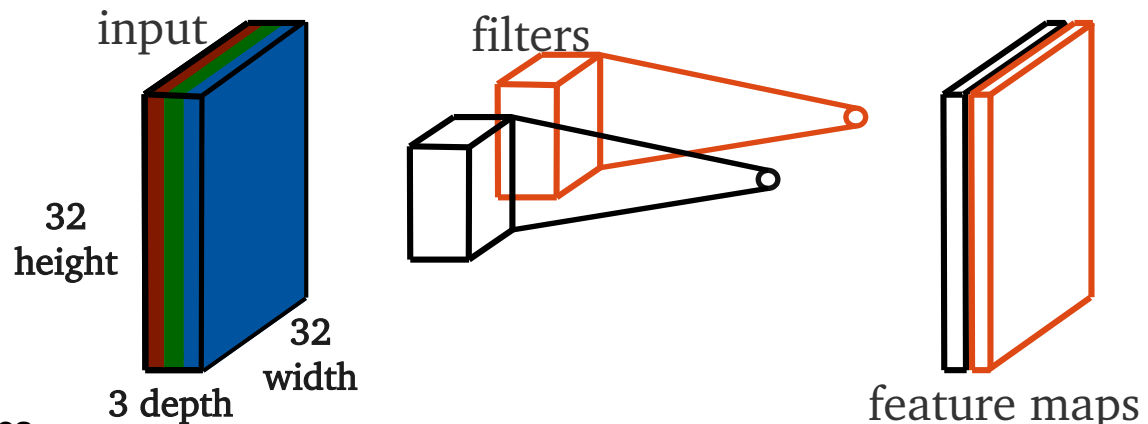
deep learning

Convolutional Networks

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

adaptive parameters

- scan input image for the presence of specific feature using **filters**
- use multiple filters and stack the results as **feature maps** (depth-wise stacking)

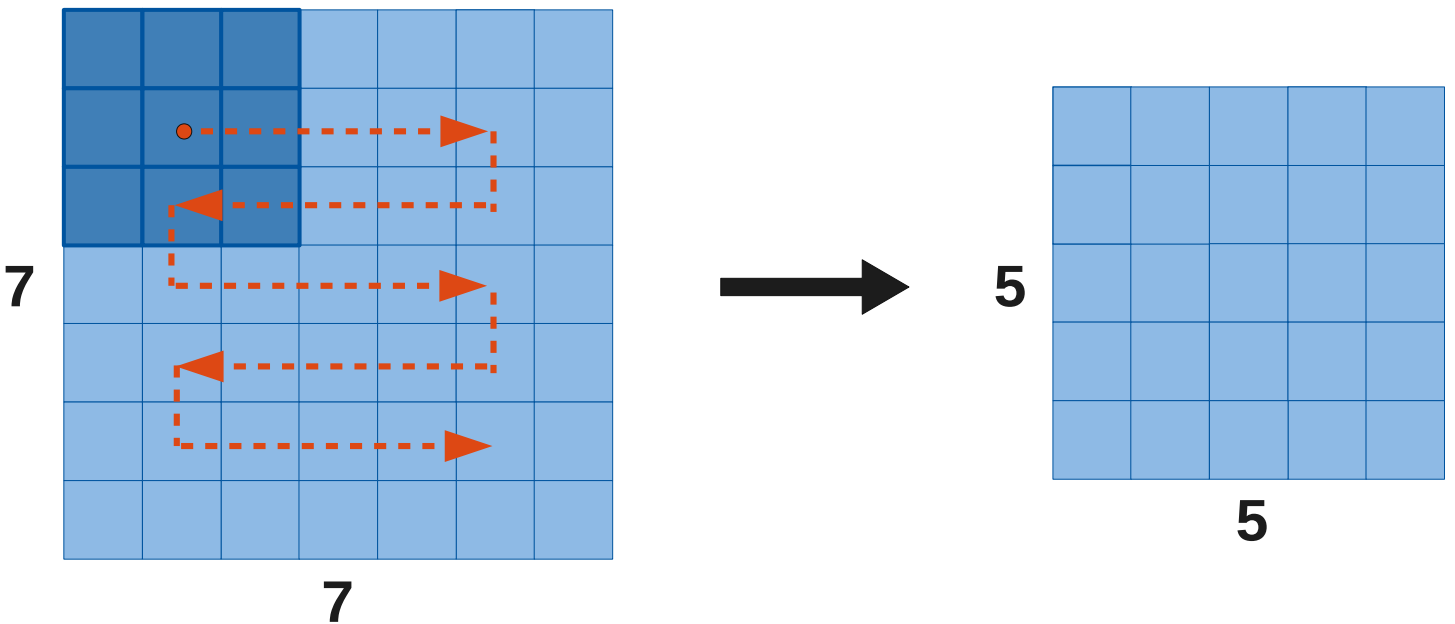


Spatial Output Size

Standard convolution reduces the output size due to extent of the filter

- Sets upper bound to the number of convolutional layers

- **Example:** Convolution with 3 x 3 filter

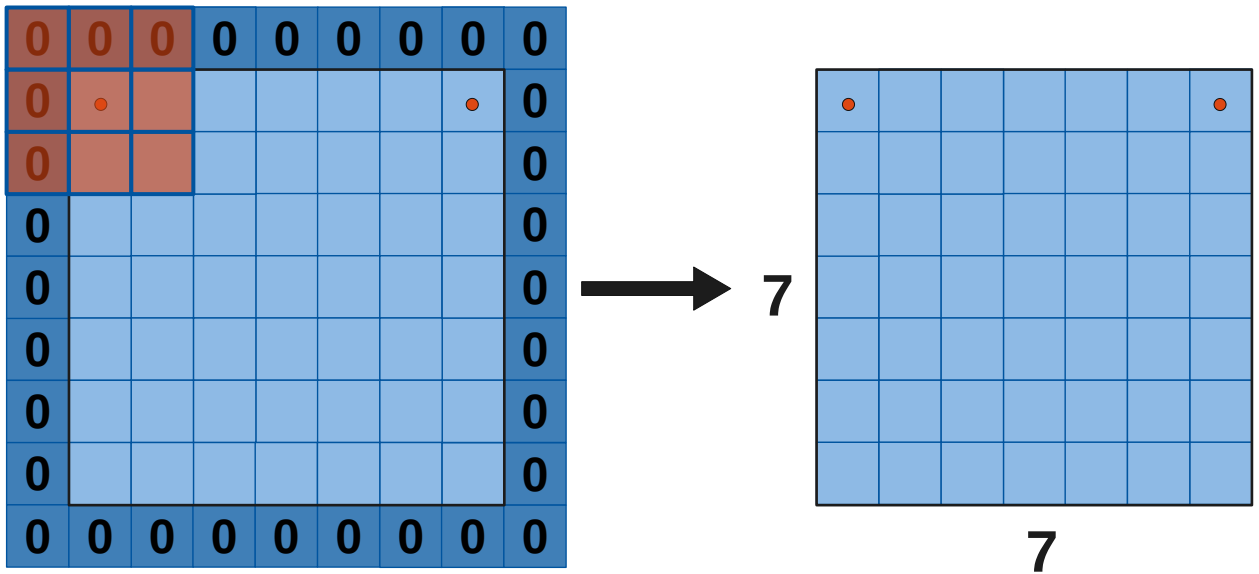


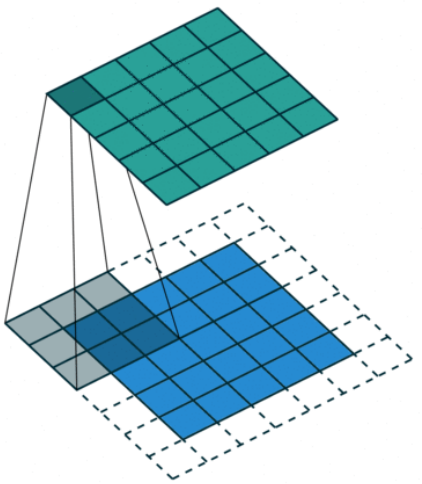
Padding

Add zeros around image borders to conserve the spatial extent of the input

- Prevents fast shrinking of the network input

- **Example:** Convolution with 3 x 3 filter and padding





Paul-Louis Pröve,
Towards Data Science

Pooling

Sub-sample the input to reduce the output size

- Used to merge semantically similar features
- Make network invariant to small translations or perturbations

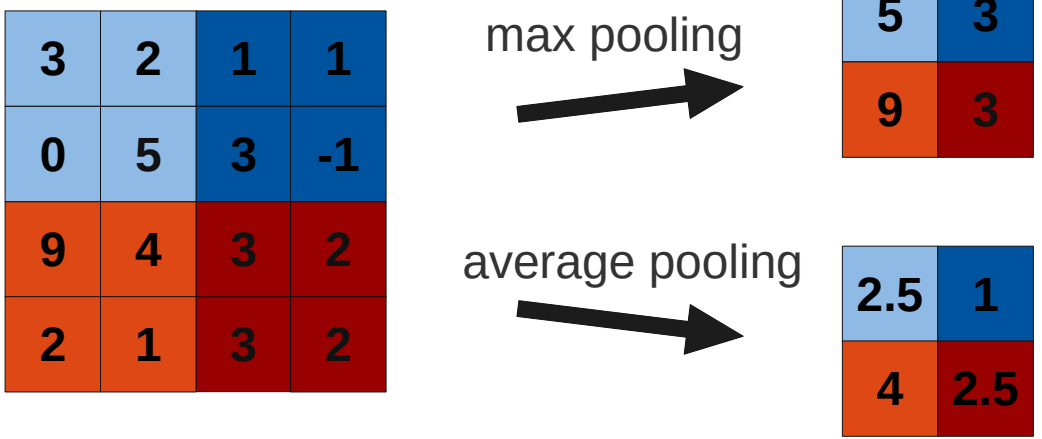
Average pooling: Take the mean of each patch → for some regressions preferable

Max pooling: Take the maximum of each patch

→ in practice often better performance, applies stronger constraint

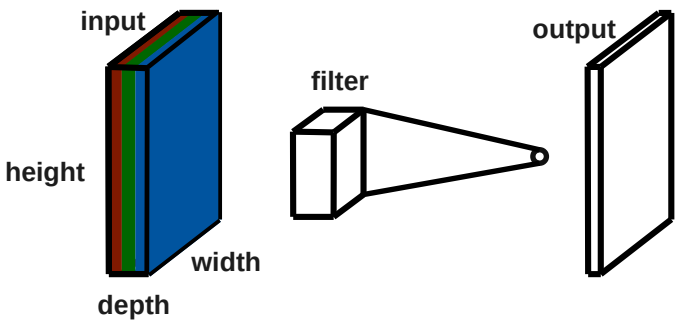
- **Typical Pooling:**
  Pooling using 2 x 2 patches
  and a stride of 2

- **Overlapping Pooling:**
  3 x 3 patches with stride of 2



Summary

- 2D Convolution acts on 3D input (width x height x depth)
- Slide small filter over input and make linear transformation (dot product + bias)
- Hyperparameter:
  - Size of filter, typically (1 x 1), (3 x 3), (5 x 5) or (7 x 7)
  - Number of filters (feature maps)
  - **Padding** (maintain spatial extent)
  - **Striding** or **pooling** (reduce spatial extent)
- Reduction of parameters using symmetry in data:
  - Prior on **local correlations** (use small filters)
  - **Translational invariance** (weight sharing)

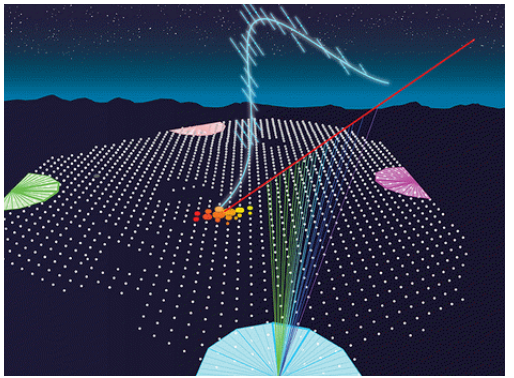


Air Shower Reconstruction - CNN

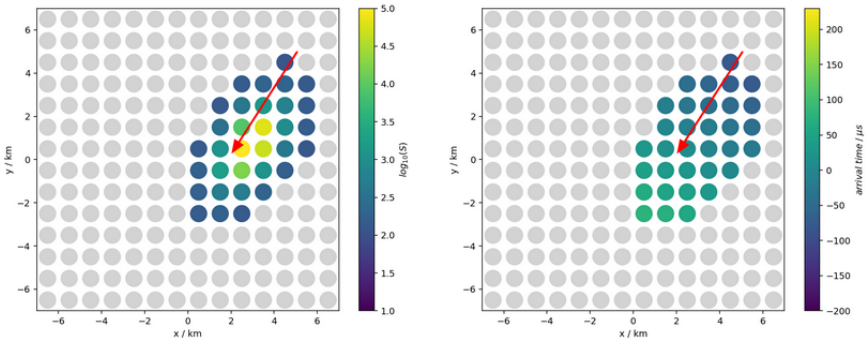
III. Physikalisches Institut A | RWTH AACHEN UNIVERSITY

Now: OPEN tutorial at:

- https://github.com/jglombitza/tutorial_nn_airshowers
- or click



$E = 11.4$ EeV, $\theta = 56.1°$, $\phi = 57.2°$



Task

- Reconstruct energy of the shower
  - Footprint is 2D image
  - **can** directly be used as input
    → input shape: 14 x 14 x 2
  - **Try to reach a resolution better than 2 EeV!**

Results I

Model – add:

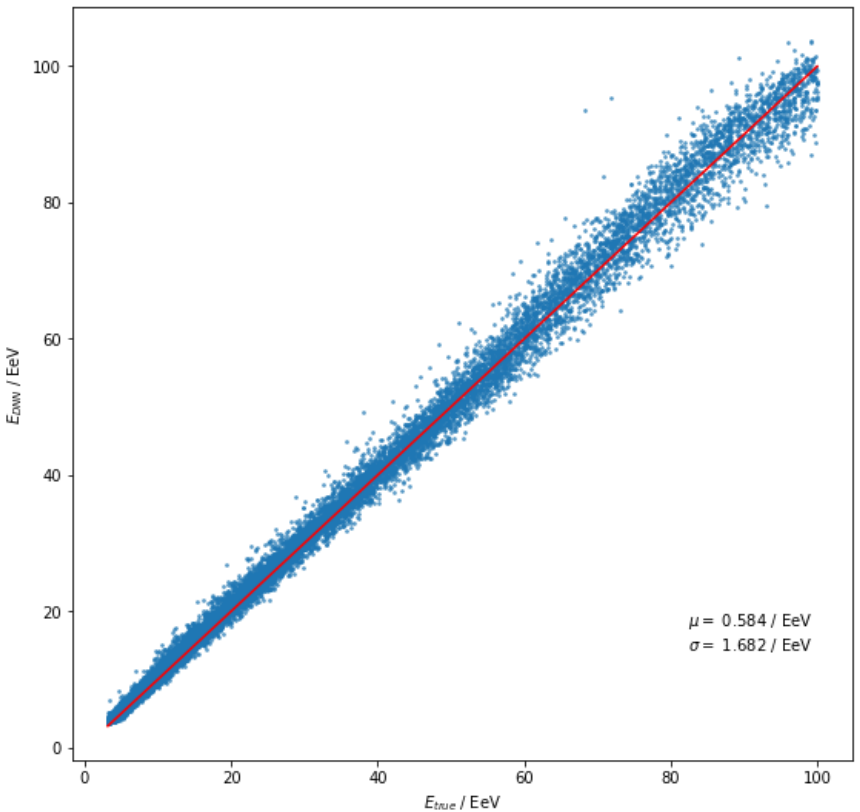
- Conv. layers and filters
- Pooling, Dense (FC) layers
- Regularization (after Flatten)

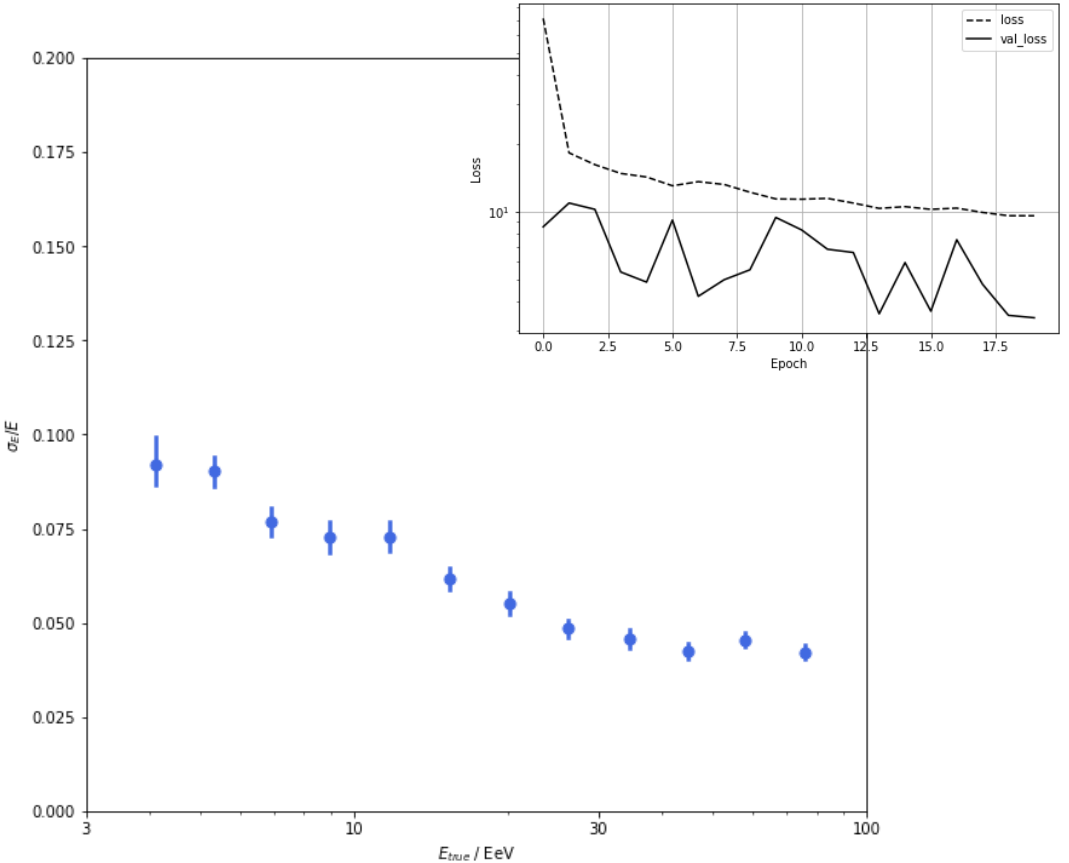
Model – modify:

- Batch size, epochs
- Kernel size, strides
- Optimizer, learning rate

```
kwargs = dict(activation='elu', padding='same',)
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3),
input_shape=X_train.shape[1:], **kwargs))
model.add(layers.Conv2D(32, (3, 3), **kwargs))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), **kwargs))
model.add(layers.Conv2D(64, (3, 3), **kwargs))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), **kwargs))
model.add(layers.Conv2D(128, (3, 3), **kwargs))
model.add(layers.GlobalMaxPooling2D())
model.add(layers.Dropout(0.3))
model.add(layers.Dense(1))
```

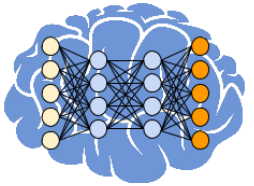
Results II


100
80
60
40
20
0
0
20
40
60
80
100
E_{DNN} / EeV
E_{true} / EeV
$\mu = 0.584$ / EeV
$\sigma = 1.682$ / EeV


0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
3
10
30
100
σ_E/E
E_{true} / EeV
loss
val_loss
Loss
10^1
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch

References & Further Reading

- M. Erdmann, J. Glombitza, G. Kasieczka, U. Klemradt, Deep Learning for Physics Research, World Scientific, 2021, www.deeplearningphysics.org/
- I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, Chapter 7 / 8 / 9, MIT Press, 2016, www.deeplearningbook.org
- Xu et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, arXiv:1502.03044
- Y. LeCun, Y. Bengio, G. Hinton: Deep Learning, Nature 521, pages 436–444
- K. Simonyan, A. Zissermann: Very Deep Convolutional Networks for Large-Scale Image Recognition - ArXiv 1409.1556
- Toy Simulation: M. Erdmann, J. Glombitza, D. Walz, Astroparticle Physics 97, 46-53



RWTH AACHEN UNIVERSITY

Introduction to Deep Learning

- Basic Methods & Techniques
- Deep Learning Frameworks
- Physics Examples and further Applications

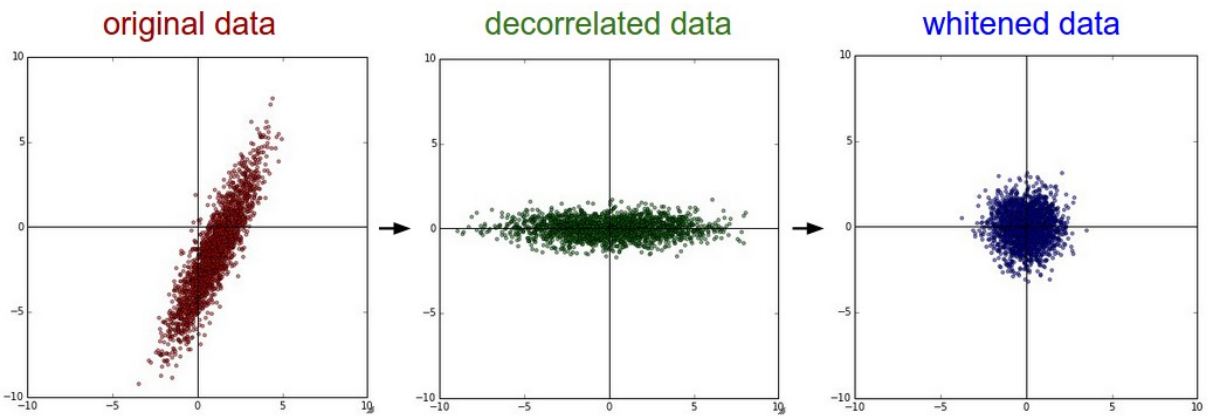
BACKUP

Jonas Glombitza

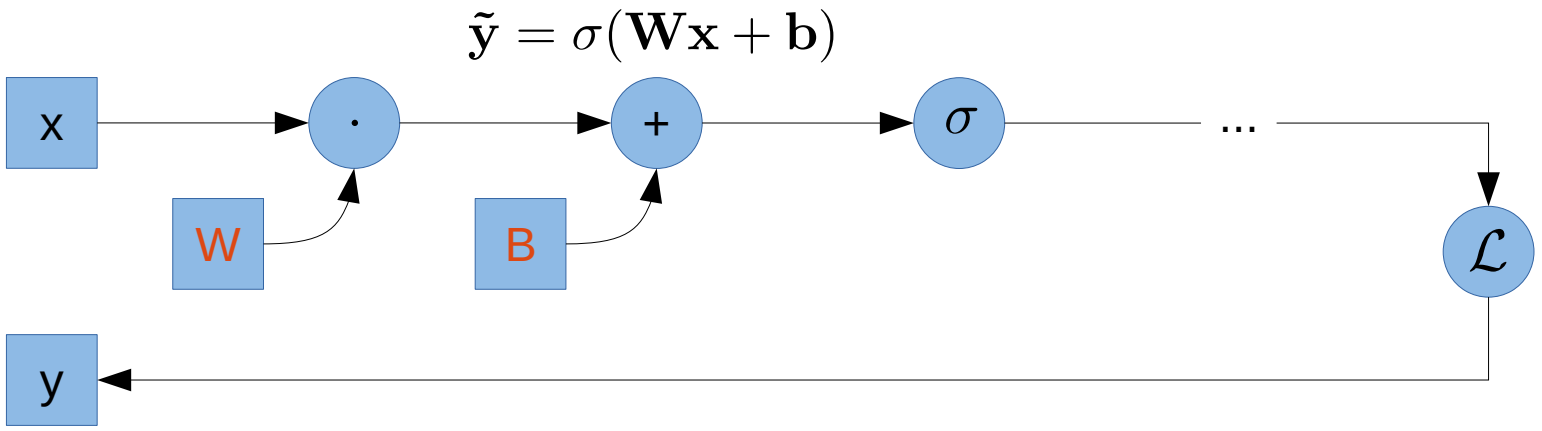
RWTH Aachen

Data Preprocessing

- Input features of dataset should be on same scale
  - Prevent particular sensitivity to few features
- Common normalization strategies
  - Limit range between [0, 1] or [-1,1]
  - Standard normalization: $\mu(x_i) = 0 \;\&\; \sigma(x_i) = 1$
  - Whitening: standard normalization + decorrelation

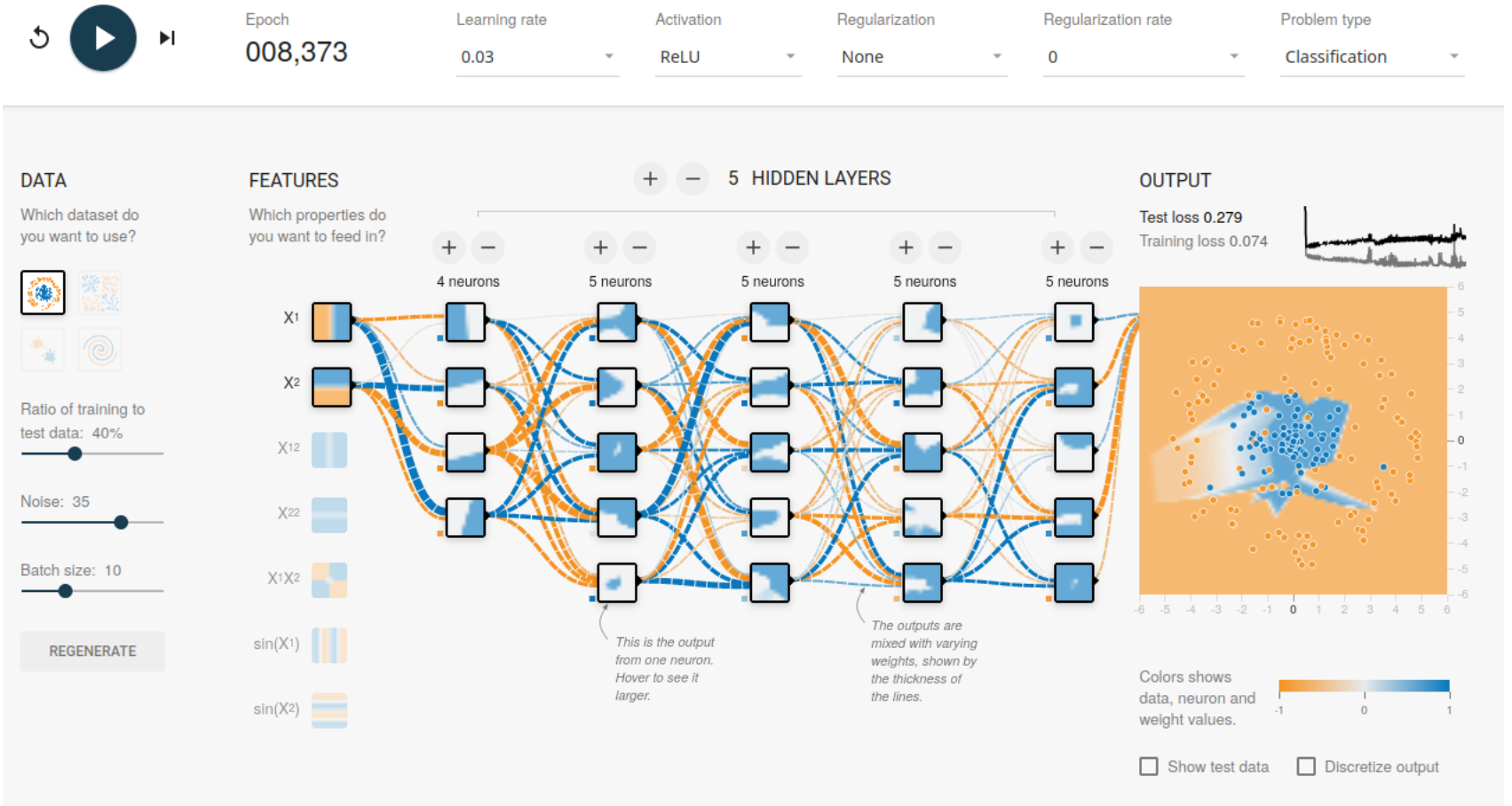


Backpropagation



- Network is series of simple operations (linear mappings/activations/loss ...)
- Each operations knows how to to calculate:
  - Its local output (forward pass)
  - Its derivative (backward pass)
- Use chain rule to evaluate gradient for each parameter
- Fast evaluation of the gradient → **Backpropagation**

Overtraining

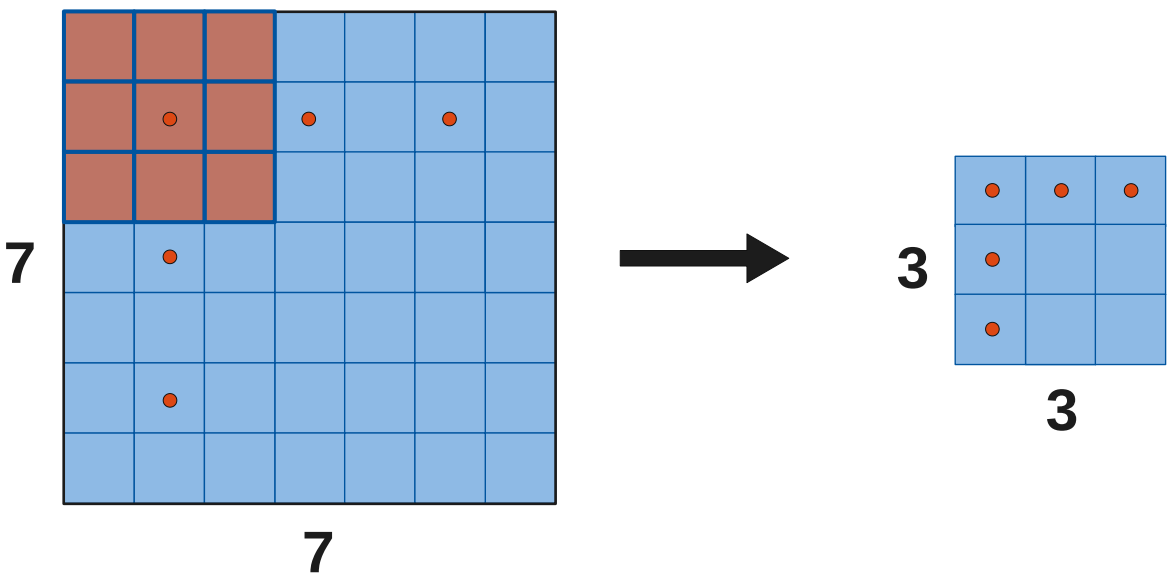


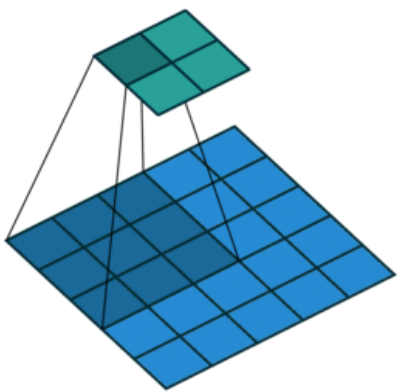
Striding

Using a larger stride when sliding over the input, reduces the output size

- Useful for switching to smaller image sizes / larger scales

- **Example:** Convolution with 3 x 3 filter and stride of 2





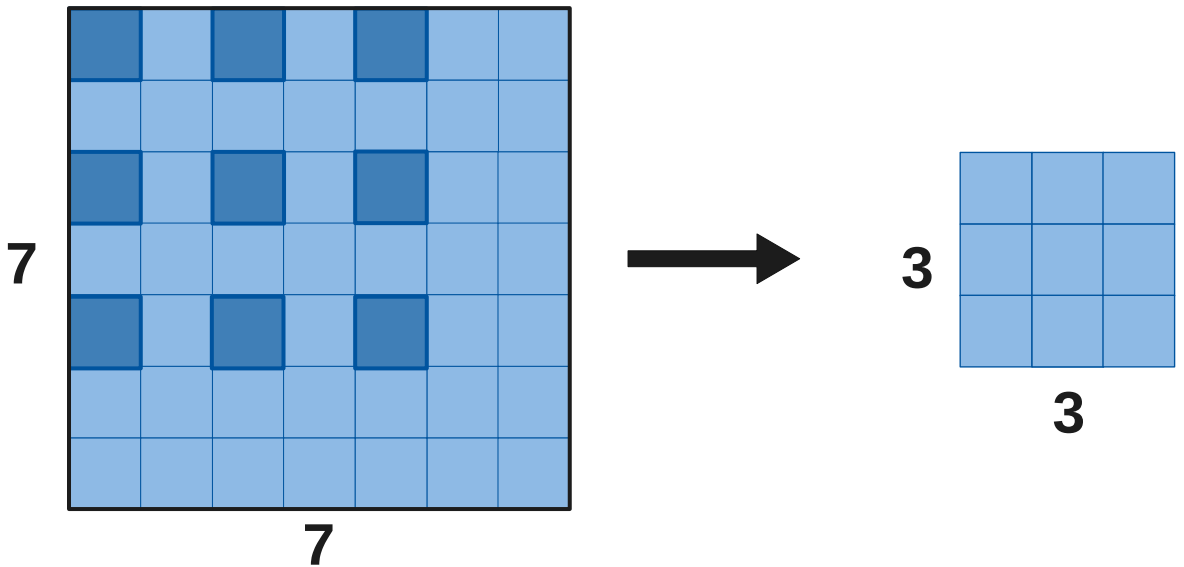
Paul-Louis Pröve,
Towards Data Science

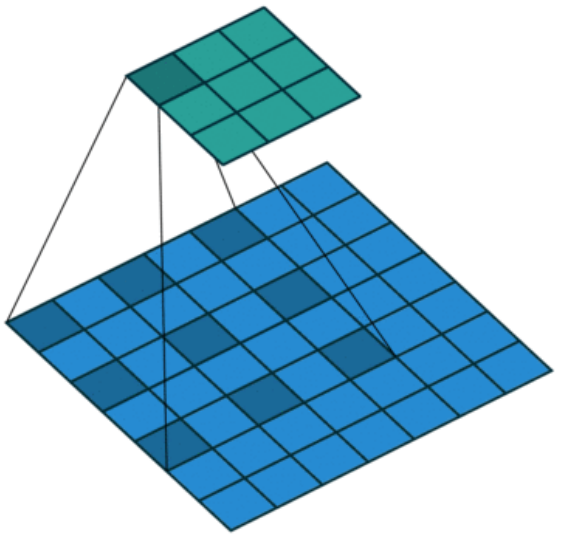
Dilating

Dilation leaves holes in where the filter is applied (also called **atrous convolution**)

- Useful for aggressively merging spatial information in large images
- Allows for a large field of view

- **Example:** Convolution with 3 x 3 filter and dilation 1





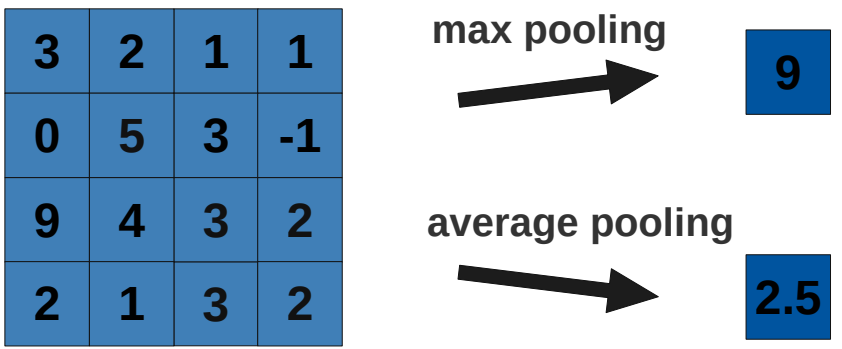
Paul-Louis Pröve,
Towards Data Science

Global Pooling Operation

- Take maximum/average over complete image → usually second last layer
- Replace fully connected layers
  - Saves parameters in later layers of the models → prevent overfitting
- Can be seen as regularizer
  - Fully connected transformation matrix with diagonal shape
- Enforcing correspondences between feature maps and categories
- Allows object detection in the input space

"The pooling operation used in convolutional neural networks is a big mistake, and the fact that it works so well is a disaster"

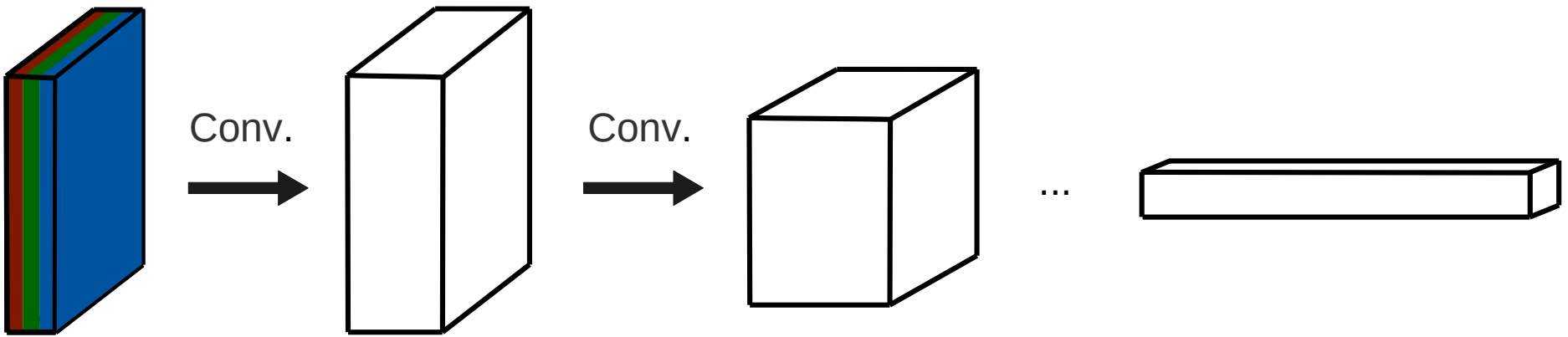
- Geoffrey Hinton



Convolutional Pyramid

ConvNet architectures usually have a pyramidal shape. For deeper layers:

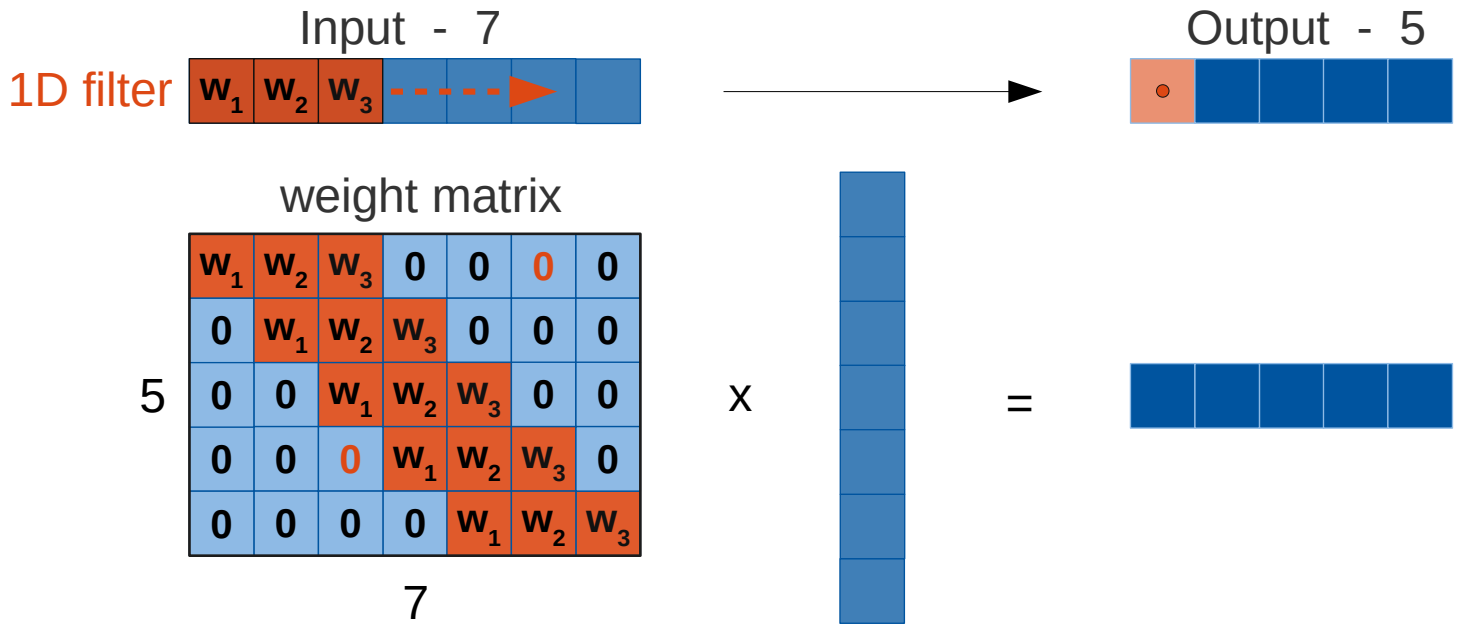
- Increase of feature space
- Decrease of spatial extent



- Spatial information is converted to representational features with increasing hierarchy

Convolutional Operation

- Fully connected layers are special case of convolutional layers



- Parameters greatly reduced due to **sparsity** and **weight sharing**
- Strong prior on **local correlation** and **translational invariance**

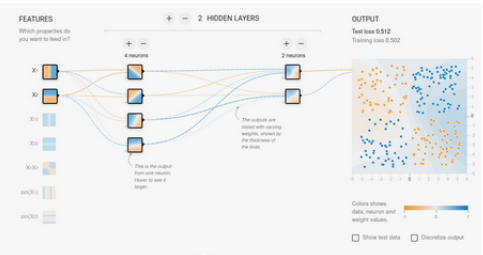
Practice 0: TensorFlow Playground

III. Physikalisches Institut A | RWTH AACHEN UNIVERSITY

https://playground.tensorflow.org

Checkerboard pattern, choose ReLU activation:

- Repeat the training several times? What do you observe? Why?
- What is the minimum number of nodes / layers needed to solve the task?
- Which input simplifies the task at most?
- What do you see when inspecting the features of deeper layers?



Circle pattern

- Increase the noise to 30, build a deep network (>4 layers and >4 nodes each) and train for >300 epochs. What do you observe?
- Use L1 and L2 regularization and try different regularization rates
- What is the difference between L1 and L2? (inspect weights and features)

Bonus

- Solve the Spiral pattern:
- What do you observe when changing the activation function?