Deep_Learning_Presentation

May 17, 2018

```
In [4]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from scipy import signal
        import sklearn
        from sklearn.preprocessing import MinMaxScaler
```

1 Überblick

1.1 Was ist Deep Learning?

1.2 Arten des Deep Learning

- Suppervised Learning
- Unsupervised Learning
- Reinforcement Learning

1.3 Die wichtigsten Frameworks

- Tensorflow (Google)
  - Keras
  - Sonnet
- Pytorch (Facebook)
- DeepLearning4J (Eclipse Foundation)
- Theano (Université de Montréal)
- MXNet (Apache)

1.4 Grundbegriffe

4.6	3.2	1.4	0.2	Iris-setosa
5.3	3.7	1.5	0.2	Iris-setosa
5.0	3.3	1.4	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4.0	1.3	Iris-versicolor

1.4.1 - Sample

1.4.2 - Features

1.4.3 - Label

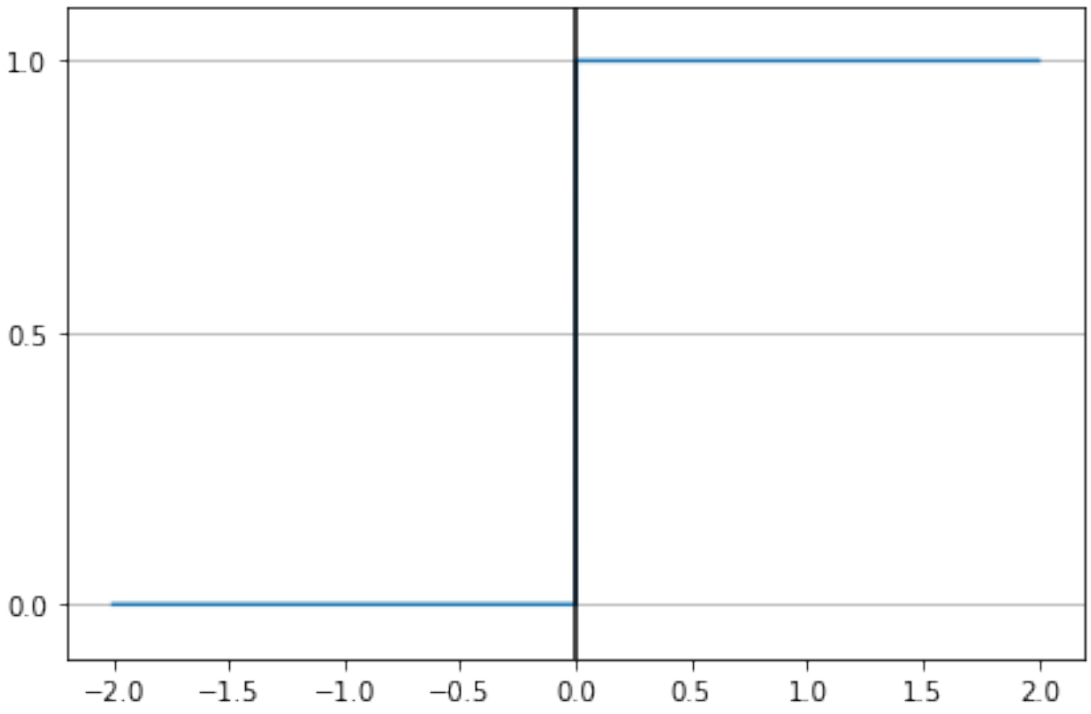
2 Das Neuron

2.1 Activation Function

```
In [5]: def plot_activation(x, y, yrange=(0, 1)):
            plt.plot(x, y)
            plt.axvline(0.0, color='k')
            plt.ylim(yrange[0]-0.1, yrange[1]+0.1)
            plt.yticks(np.arange(yrange[0], yrange[1]+0.1, 0.5))
            ax = plt.gca()
            ax.yaxis.grid(True)
            plt.tight_layout()
            plt.show()
```

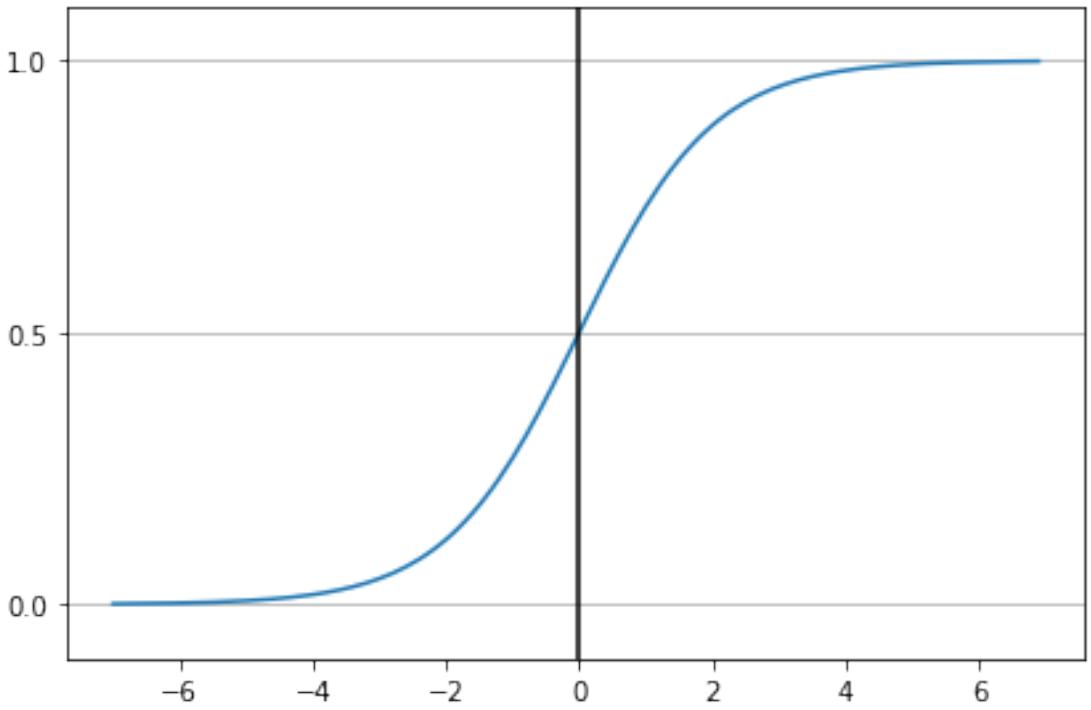
2.1.1 Binary Step

```
In [6]: def threshold(x):
            return np.where(x >= 0.0, 1, 0)

        x = np.arange(-2, 2, 0.001)
        threshold = threshold(x)
        plot_activation(x, threshold)
```



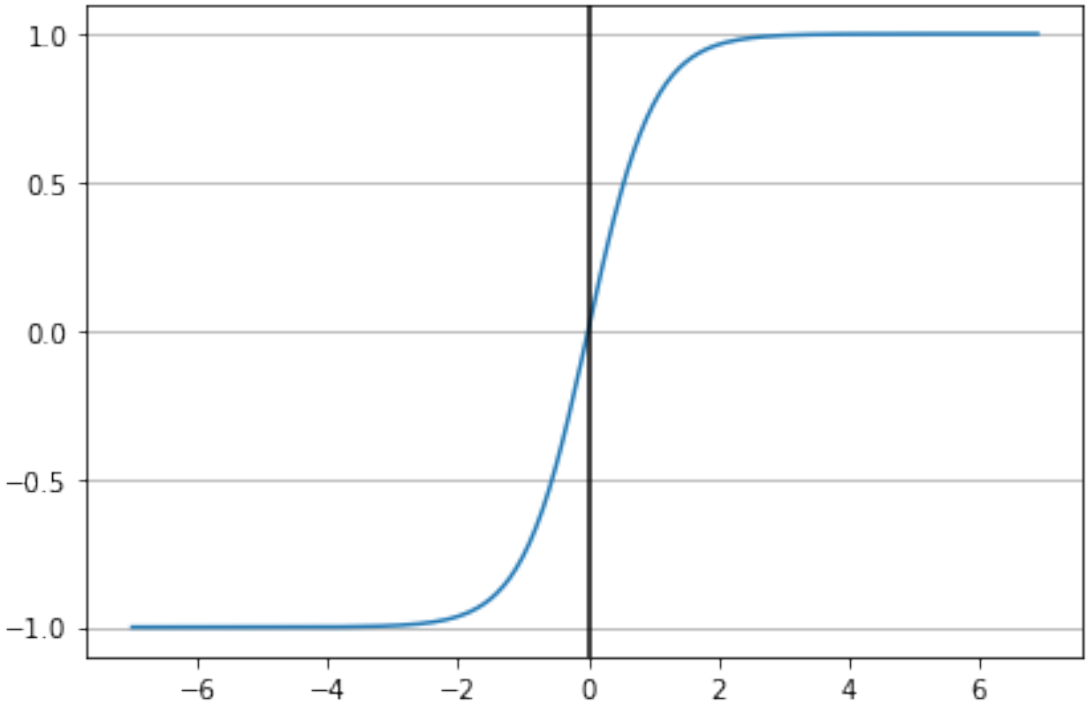
2.1.2 Sigmoid

```
In [7]: def sigmoid(x):
            return 1.0 / (1.0 + np.exp(-x))

        x = np.arange(-7, 7, 0.1)
        sig = sigmoid(x)
        plot_activation(x, sig)
```



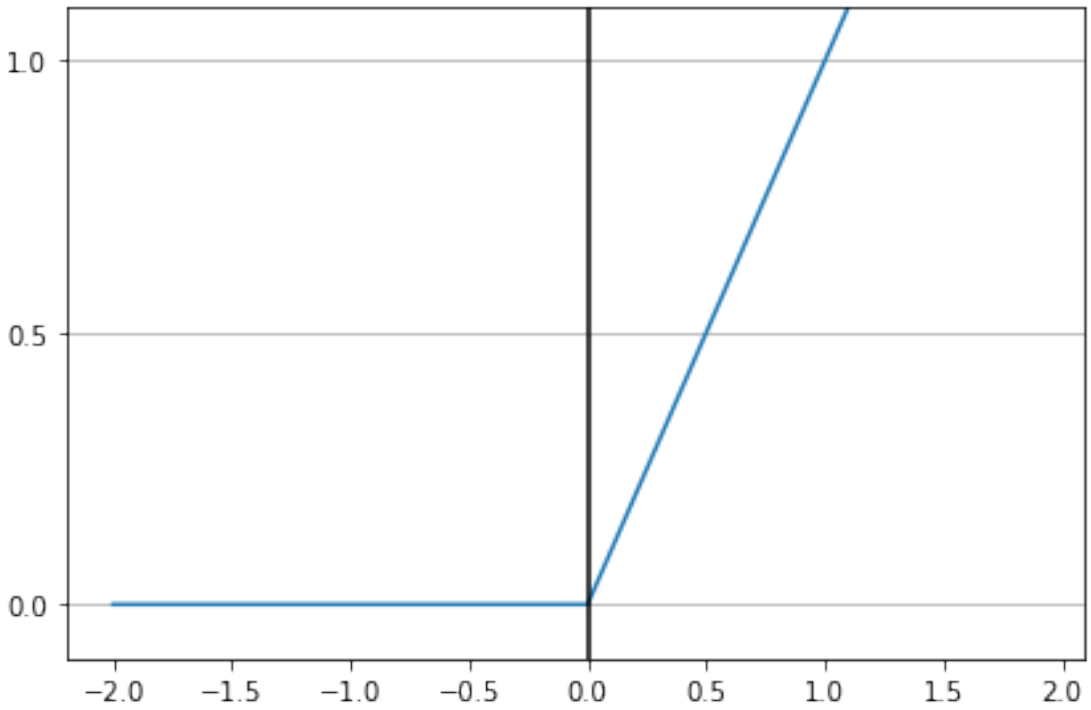
2.1.3 Tanh

```
In [8]: def tanh(x):
            return ((np.exp(x)-np.exp(-x))/(np.exp(x)+np.exp(-x)))

        x = np.arange(-7, 7, 0.1)
        tanh = tanh(x)
        plot_activation(x, tanh, yrange=(-1, 1))
```



2.1.4 Rectified linear unit

```
In [9]: def relu(x):
            return np.maximum(0, x)

        x = np.arange(-2, 2, 0.1)
        relu = relu(x)
        plot_activation(x, relu)
```



2.2 Gradient Descent

```
In [10]: df = pd.read_csv("iris.data", header=None)
         df.head()
```

Out[10]:

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

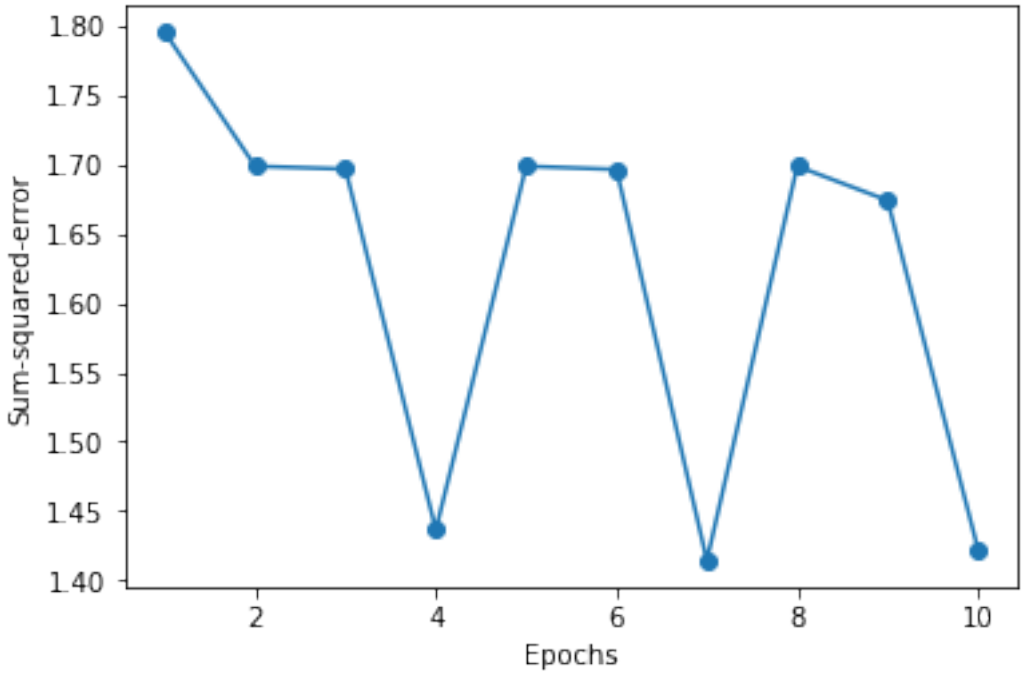
```
In [11]: class AdalineGD(object):

             def __init__(self, eta=0.01, n_iter=50):
                 self.eta = eta
                 self.n_iter = n_iter

             def fit(self, X, y):
                 self.w_ = np.zeros(1 + X.shape[1])
                 self.cost_ = []

                 for i in range(self.n_iter):
                     output = self.activation(X)
```

```
                    print("-------------------------")
                    print("Epoch: ", (i+1))
                    print("Prediction on First Sample: ", output[0])
                    print("-------------------------")
                    errors = (y - output)
                    self.w_[1:] += self.eta * X.T.dot(errors)
                    self.w_[0] += self.eta * errors.sum()
                    cost = (errors**2).sum() / 2.0
                    self.cost_.append(cost)
                return self

            def net_input(self, X):
                return np.dot(X, self.w_[1:]) + self.w_[0]

            def activation(self, X):
                return sigmoid(self.net_input(X))

            def predict(self, X):
                return np.where(self.activation(X) >= 0.0, 1, -1)
```

In [12]:
```
y = df.iloc[0:100, 4].values
y = np.where(y == 'Iris-setosa', -1, 1)
x = df.iloc[0:100, [0, 2]].values

n_iter = 10
eta = 0.01

ada = AdalineGD(n_iter=n_iter, eta=eta).fit(x, y)
plt.plot(range(1, len(ada.cost_) + 1), np.log10(ada.cost_), marker='o')
plt.xlabel('Epochs')
plt.ylabel('Sum-squared-error')
plt.show()
```

```
-------------------------
Epoch:  1
Prediction on First Sample:  0.5
-------------------------
-------------------------
Epoch:  2
Prediction on First Sample:  5.418153852432133e-06
-------------------------
-------------------------
Epoch:  3
Prediction on First Sample:  0.0004107165108106998
-------------------------
-------------------------
Epoch:  4
Prediction on First Sample:  0.02727456901589131
```

```
-------------------------
-------------------------
Epoch:  5
Prediction on First Sample:  2.6354619235982345e-08
-------------------------
-------------------------
Epoch:  6
Prediction on First Sample:  1.9988435518708985e-06
-------------------------
-------------------------
Epoch:  7
Prediction on First Sample:  0.00013720821532945467
-------------------------
-------------------------
Epoch:  8
Prediction on First Sample:  7.63281271557861e-10
-------------------------
-------------------------
Epoch:  9
Prediction on First Sample:  5.787913907136857e-08
-------------------------
-------------------------
Epoch:  10
Prediction on First Sample:  1.3098865107064567e-06
-------------------------
```



2.3 Stochastic und Batch Gradient Descent

3 Das Neuronale Netz

```
In [13]: from keras.models import Sequential
         from keras.layers import Dense, Dropout
         from sklearn.preprocessing import LabelEncoder
         from keras.utils import np_utils
         from keras.optimizers import Adam
         df = pd.read_csv("iris.data", header=None)
         dataset = df.values
         X = dataset[:, 0:4].astype(float)
         y = dataset[:,4]
```

```
/usr/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second a
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

```
In [14]: encoder = LabelEncoder()
         encoder.fit(y)
         encoded_y = encoder.transform(y)
         encoded_y
```

```
Out[14]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
In [15]: dummy_y = np_utils.to_categorical(encoded_y)
         dummy_y
```

```
Out[15]: array([[1., 0., 0.],
                [1., 0., 0.],
                [1., 0., 0.],
                [1., 0., 0.],
                [1., 0., 0.],
                [1., 0., 0.],
                [1., 0., 0.],
                [1., 0., 0.],
                [1., 0., 0.],
                [1., 0., 0.],
                [1., 0., 0.],
                [1., 0., 0.],
                [1., 0., 0.],
```

```
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[1., 0., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
```

```
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 1., 0.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
```

```
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.],
[0., 0., 1.]])
```

In [16]:
```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(-1,1))
X_scaled = scaler.fit_transform(X)
```

In [17]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X_scaled, dummy_y, test_size=0.2,
```

```
In [18]: model = Sequential()
         model.add(Dense(16, input_dim=4, activation='relu'))
         model.add(Dense(32, activation='relu'))
         model.add(Dense(3, activation='softmax'))
         model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
         model.fit(X_train, Y_train, epochs=50, batch_size=1, validation_data=[X_test, Y_test])

Train on 120 samples, validate on 30 samples
Epoch 1/50
120/120 [==============================] - 1s 8ms/step - loss: 0.8131 - acc: 0.7583 - val_loss
Epoch 2/50
120/120 [==============================] - 0s 2ms/step - loss: 0.5110 - acc: 0.8583 - val_loss
Epoch 3/50
120/120 [==============================] - 0s 2ms/step - loss: 0.3286 - acc: 0.8833 - val_loss
Epoch 4/50
120/120 [==============================] - 0s 2ms/step - loss: 0.2453 - acc: 0.9333 - val_loss
Epoch 5/50
120/120 [==============================] - 0s 2ms/step - loss: 0.2008 - acc: 0.9083 - val_loss
Epoch 6/50
120/120 [==============================] - 0s 2ms/step - loss: 0.1751 - acc: 0.9333 - val_loss
Epoch 7/50
120/120 [==============================] - 0s 1ms/step - loss: 0.1516 - acc: 0.9333 - val_loss
Epoch 8/50
120/120 [==============================] - 0s 2ms/step - loss: 0.1423 - acc: 0.9500 - val_loss
Epoch 9/50
120/120 [==============================] - 0s 2ms/step - loss: 0.1259 - acc: 0.9500 - val_loss
Epoch 10/50
120/120 [==============================] - 0s 1ms/step - loss: 0.1171 - acc: 0.9500 - val_loss
Epoch 11/50
120/120 [==============================] - 0s 2ms/step - loss: 0.1092 - acc: 0.9583 - val_loss
Epoch 12/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0994 - acc: 0.9500 - val_loss
Epoch 13/50
120/120 [==============================] - 0s 1ms/step - loss: 0.0910 - acc: 0.9583 - val_loss
Epoch 14/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0877 - acc: 0.9500 - val_loss
Epoch 15/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0851 - acc: 0.9583 - val_loss
Epoch 16/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0824 - acc: 0.9583 - val_loss
Epoch 17/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0811 - acc: 0.9667 - val_loss
Epoch 18/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0770 - acc: 0.9583 - val_loss
Epoch 19/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0763 - acc: 0.9583 - val_loss
Epoch 20/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0752 - acc: 0.9583 - val_loss
```

```
Epoch 21/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0749 - acc: 0.9583 - val_loss
Epoch 22/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0690 - acc: 0.9667 - val_loss
Epoch 23/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0638 - acc: 0.9667 - val_loss
Epoch 24/50
120/120 [==============================] - 0s 1ms/step - loss: 0.0788 - acc: 0.9583 - val_loss
Epoch 25/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0685 - acc: 0.9667 - val_loss
Epoch 26/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0615 - acc: 0.9667 - val_loss
Epoch 27/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0651 - acc: 0.9583 - val_loss
Epoch 28/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0693 - acc: 0.9750 - val_loss
Epoch 29/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0599 - acc: 0.9667 - val_loss
Epoch 30/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0646 - acc: 0.9667 - val_loss
Epoch 31/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0585 - acc: 0.9833 - val_loss
Epoch 32/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0626 - acc: 0.9667 - val_loss
Epoch 33/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0628 - acc: 0.9750 - val_loss
Epoch 34/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0589 - acc: 0.9583 - val_loss
Epoch 35/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0590 - acc: 0.9667 - val_loss
Epoch 36/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0571 - acc: 0.9750 - val_loss
Epoch 37/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0605 - acc: 0.9750 - val_loss
Epoch 38/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0535 - acc: 0.9667 - val_loss
Epoch 39/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0561 - acc: 0.9750 - val_loss
Epoch 40/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0586 - acc: 0.9667 - val_loss
Epoch 41/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0547 - acc: 0.9750 - val_loss
Epoch 42/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0554 - acc: 0.9667 - val_loss
Epoch 43/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0529 - acc: 0.9750 - val_loss
Epoch 44/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0561 - acc: 0.9750 - val_loss
```

```
Epoch 45/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0498 - acc: 0.9750 - val_loss
Epoch 46/50
120/120 [==============================] - 0s 1ms/step - loss: 0.0558 - acc: 0.9833 - val_loss
Epoch 47/50
120/120 [==============================] - 0s 1ms/step - loss: 0.0563 - acc: 0.9667 - val_loss
Epoch 48/50
120/120 [==============================] - 0s 1ms/step - loss: 0.0506 - acc: 0.9833 - val_loss
Epoch 49/50
120/120 [==============================] - 0s 2ms/step - loss: 0.0534 - acc: 0.9750 - val_loss
Epoch 50/50
120/120 [==============================] - 0s 1ms/step - loss: 0.0490 - acc: 0.9750 - val_loss
```

```
Out[18]: <keras.callbacks.History at 0x7f4e0a8c8cc0>
```

4 Wichtige Arten von Neuronalen Netzen

- Densenet
- CNN
- RNN
  - LSTM
  - GRU
- Autoencoder

5 Resourcen

5.1 Video Kurse:

- The Morpheus Tutorials - Pytorch - Youtube (deutsch)
- The Morpheus Tutorials - Machine Learning - Youtube (deutsch)
- Deep Learning for Coders - fast.ai
- Machine Learning Kurs der Stanford University
- Deep Learning Kursreihe von deeplearning.ai
- Deep Learning A-Z - Udemy (kostenpflichtig)
- Neural Networks for Machine Learning - Kurs der University of Toronto

5.2 Blogs:

- Machinelearningmastery
- Fastml

5.3 Bücher:

- Fundamentals of Deep Learning
- Deep Learning
- Python Deep Learning

- Deep Learning with Keras
- Python Machine Learning
- Deep Learning (Bibel)
- Sammlung kostenloser Deep Learning Bücher

5.4 Platformen:

- Kaggle - Machine Learning Competitions
- UCI - Machine Learning Repositroy