

# Deep Learning mit Keras und TensorFlow

- Einführung Deep Learning
- Ein Neuronales Netz trainieren (Keras und TensorFlow)
- Selbststudium ermöglichen



Motivation

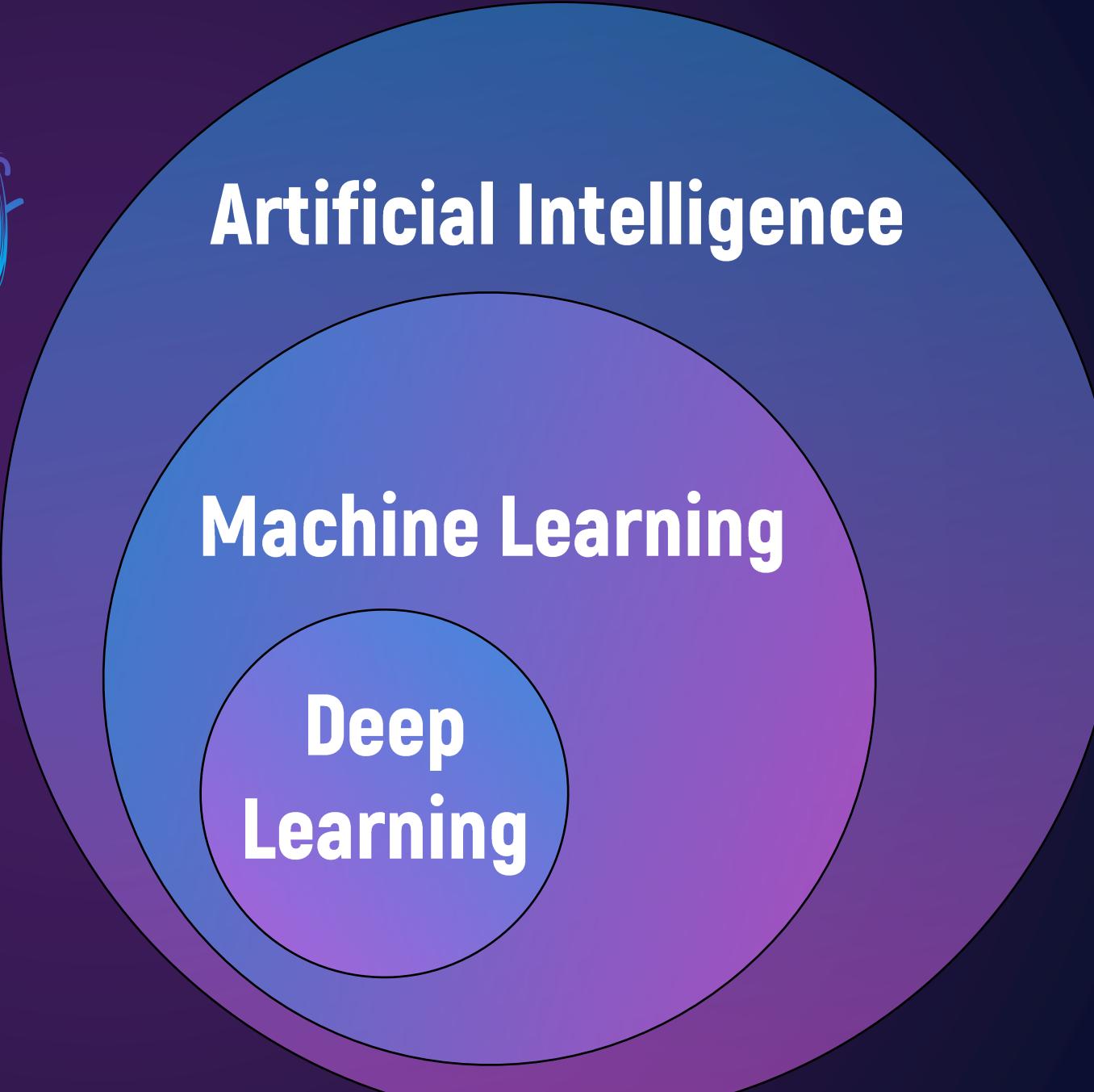
The logo for Anholt University of Applied Sciences features the word "Anholt" in a stylized, blue and purple cursive font. It is overlaid on a series of concentric circles that transition through various colors including blue, purple, pink, red, orange, and yellow.

# Anholt

- Deep Learning
- Neuronales Netze
- Keras und TensorFlow (Python)
- Google Colab (Jupyter Notebook)

# Deep Learning

Was ist  
Deep Learning



Artificial Intelligence

Machine Learning

Deep  
Learning

# Deep Learning

# Warum brauchen wir Deep Learning

<https://hackernoon.com/>



# Deep Learning

## Typische

## Deep Learning

## Probleme

## Classification



<https://praxistipps.focus.de>

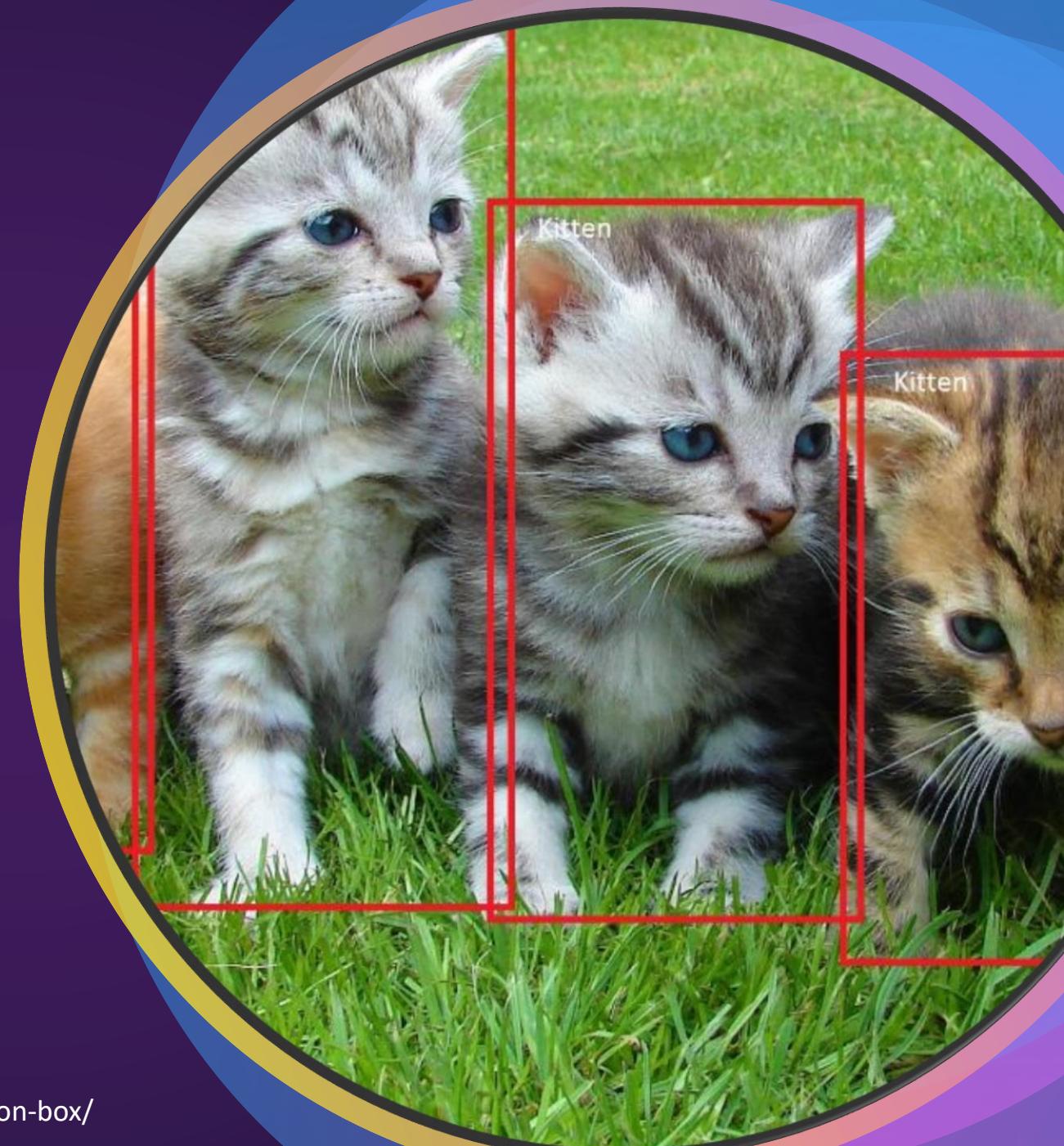


<https://www.zooroyal.de>

# Deep Learning

## Typische Deep Learning Probleme

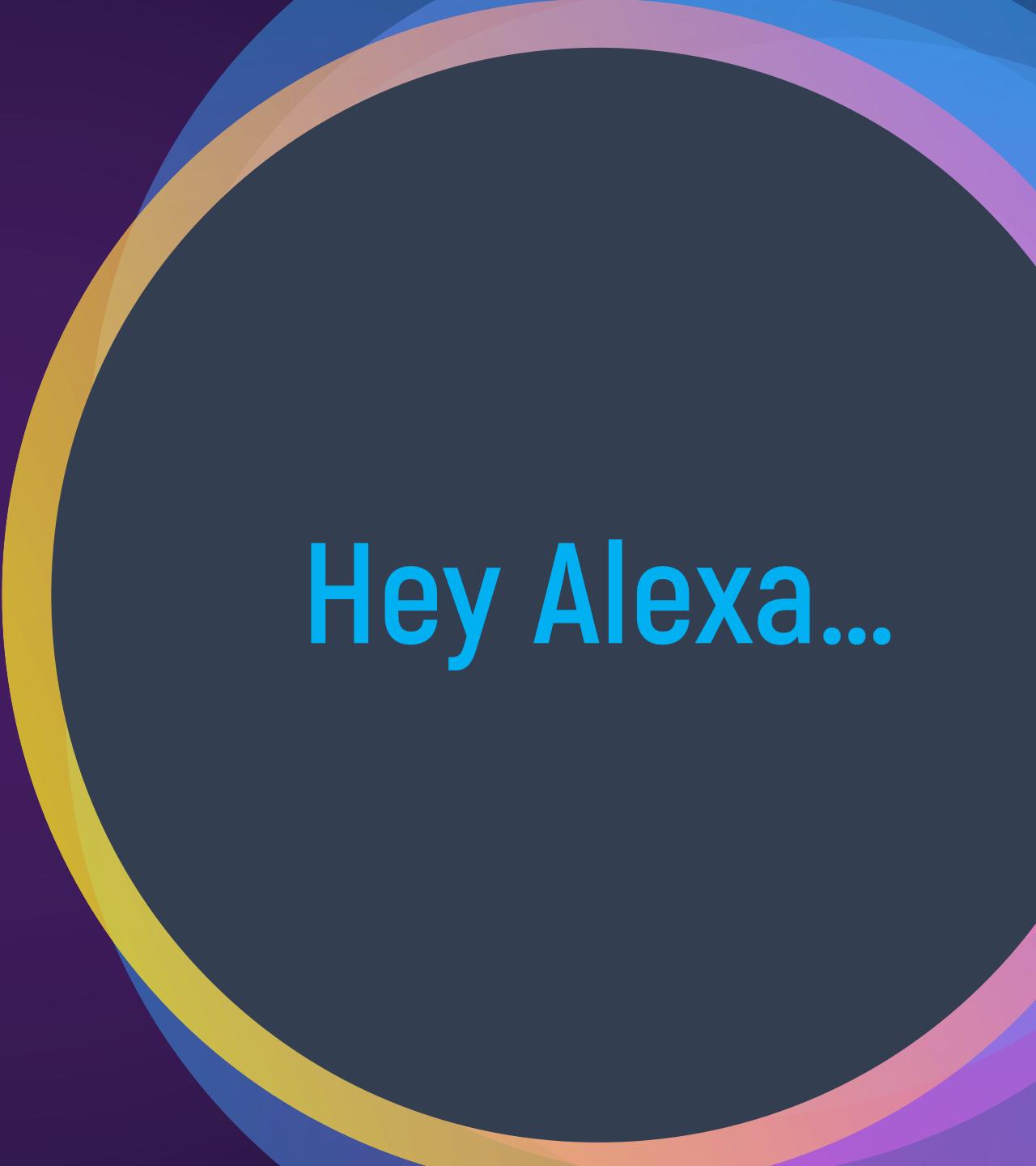
### Object Detection



# Deep Learning

Typische  
Deep Learning  
Probleme

Natural Language  
Processing



Hey Alexa...

# Deep Learning

Typische  
Deep Learning  
Probleme

Regression



# Deep Learning

Typische

Deep Learning

Probleme

Clustering



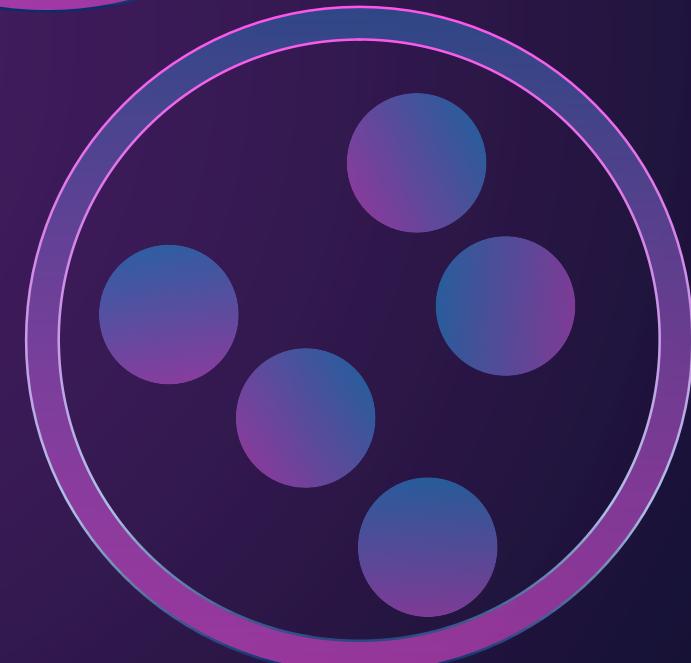
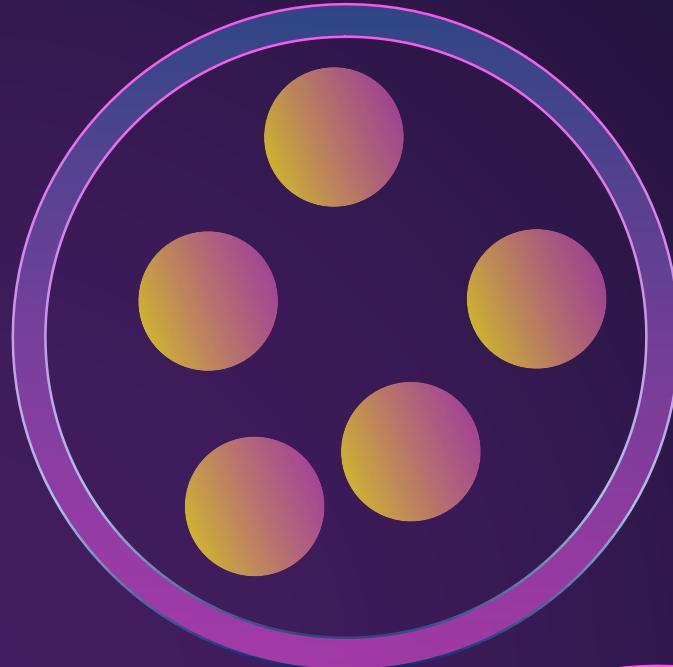
# Deep Learning

## Typische

### Deep Learning

### Probleme

### Clustering



# Deep Learning

Typische

Deep Learning

Probleme

Games



<https://www.fotocommunity.de>



<https://www.heikovaneckert.de>

Deep Learning

Wie funktioniert  
Deep Learning

# Deep Learning

Wie funktioniert  
Deep Learning

Problem



<https://theleadershipnetwork.com/>

# Deep Learning

Wie funktioniert  
Deep Learning

Problem



<https://theleadershipnetwork.com/>

Daten

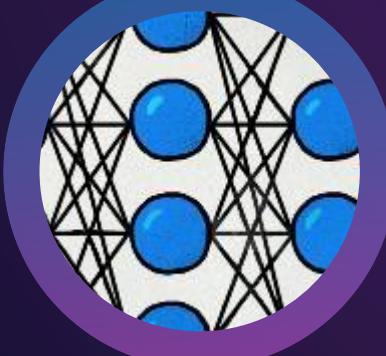


<http://farefwd.com/>

# Deep Learning

Wie funktioniert  
Deep Learning

Model



<https://medium.com/@datamonsters/>

Problem



<https://theleadershipnetwork.com/>

Daten



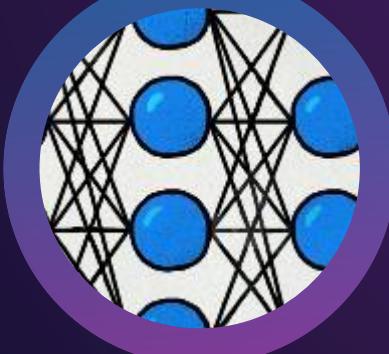
<http://farefwd.com/>

# Deep Learning

Wie funktioniert

# Deep Learning

Model



<https://medium.com/@datamonsters/>

Problem



<https://theleadershipnetwork.com/>

Daten



<http://farefwd.com/>

Algorithmus



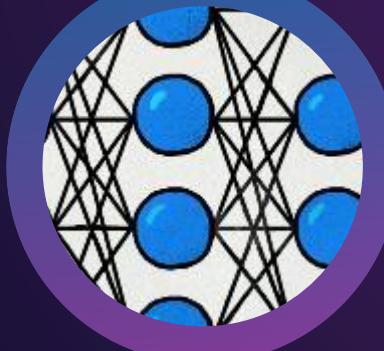
<https://oupeitglobalblog.com/2018/03/27/>

# Deep Learning

Wie funktioniert

# Deep Learning

Model



<https://medium.com/@datamonsters/>

Problem



<https://theleadershipnetwork.com/>

Daten



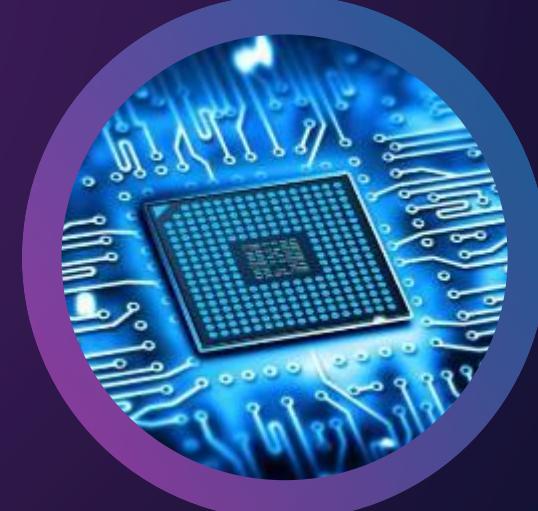
<http://farefwd.com/>

Algorithmus



<https://oupeitglobalblog.com/2018/03/27/>

Rechenzeit und Leistung



<https://www.pc-magazin.de/>

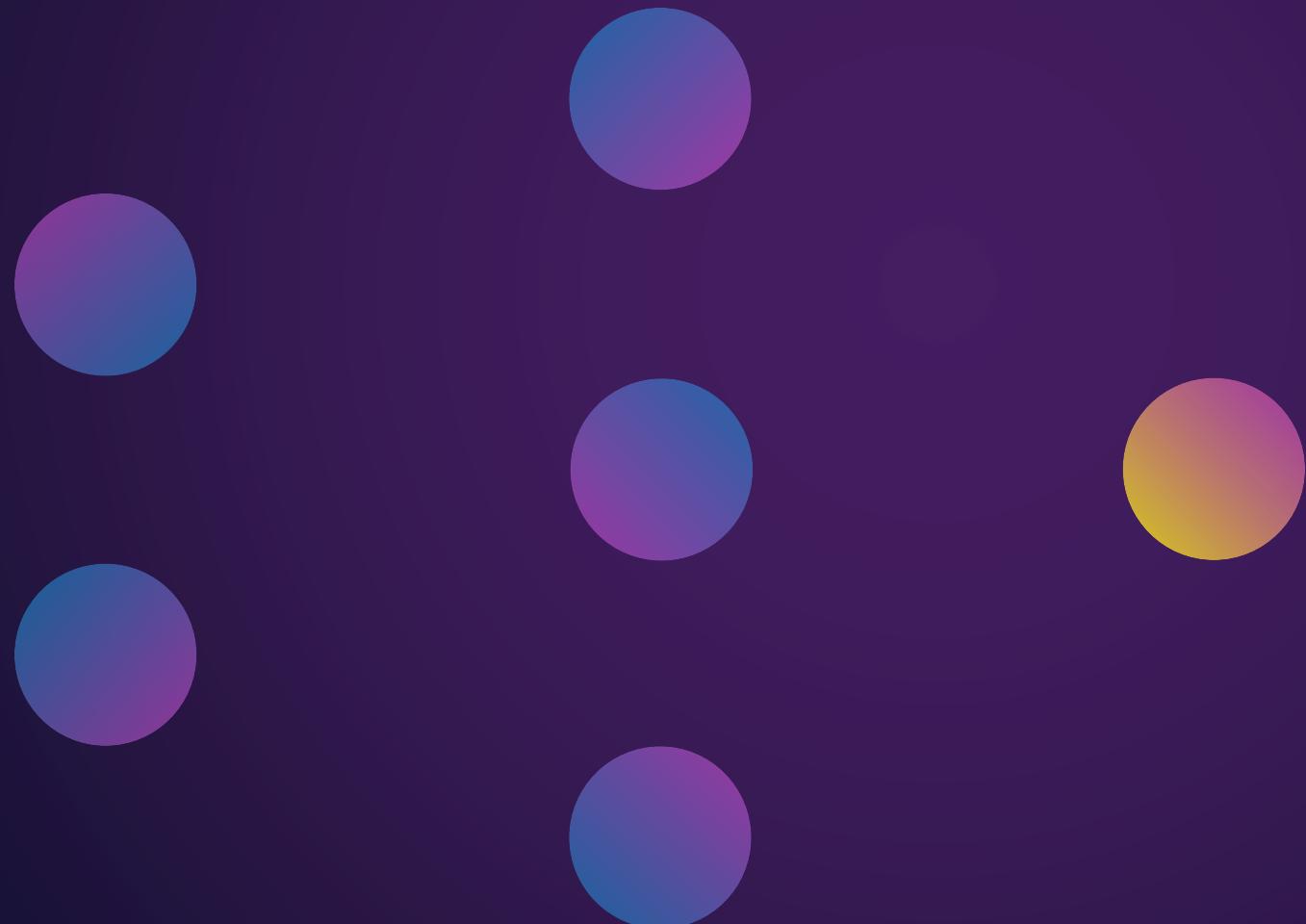


# Neuronale Netze

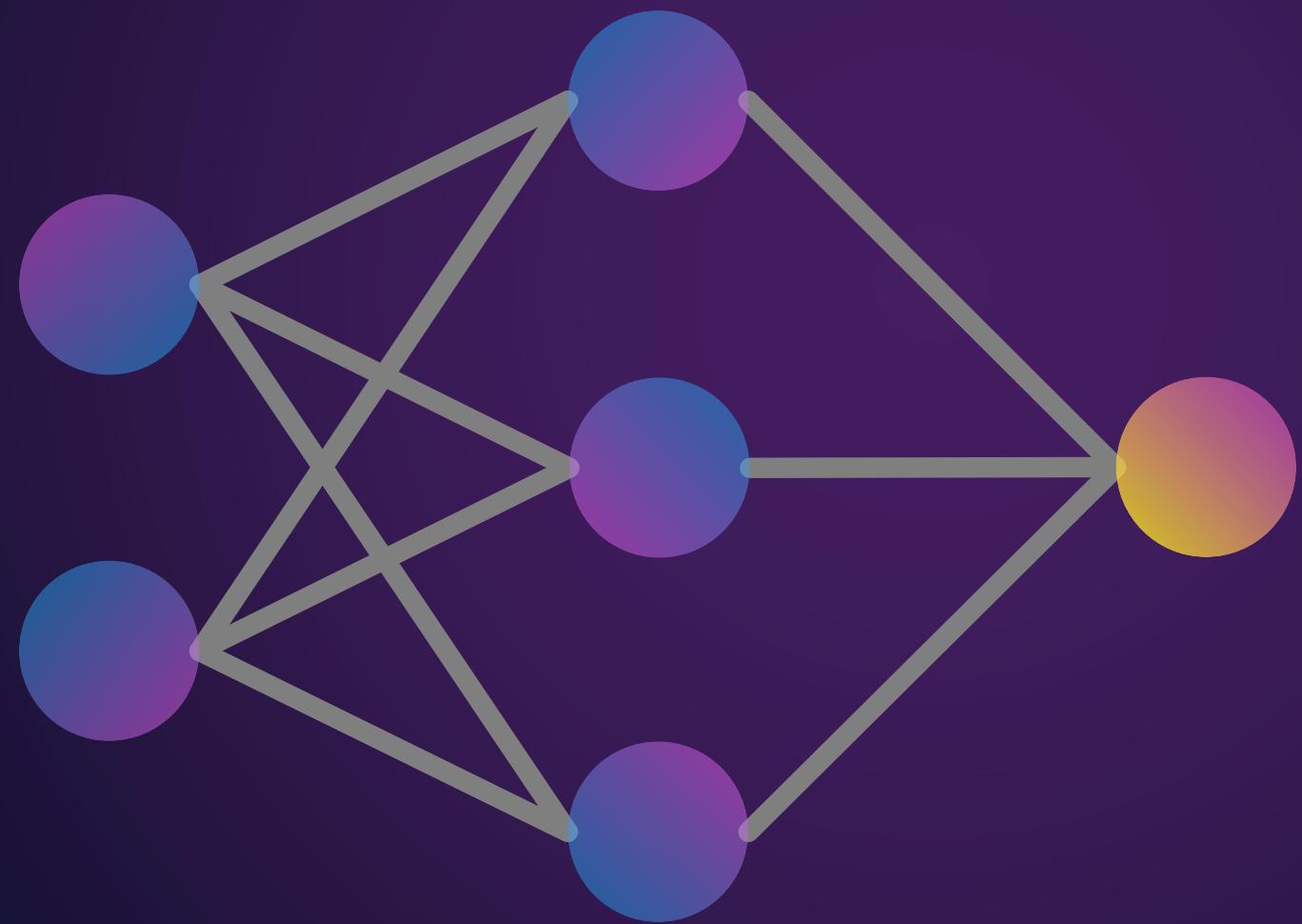
# Und ihre Geschichte

- Approximation beliebiger Funktion
  - 1943: Mathematisches Modell
  - 1986: Stabiler Lernalgorithmus
- 'loosely modeled after the human brain'
  - Vielzahl an Architecturen

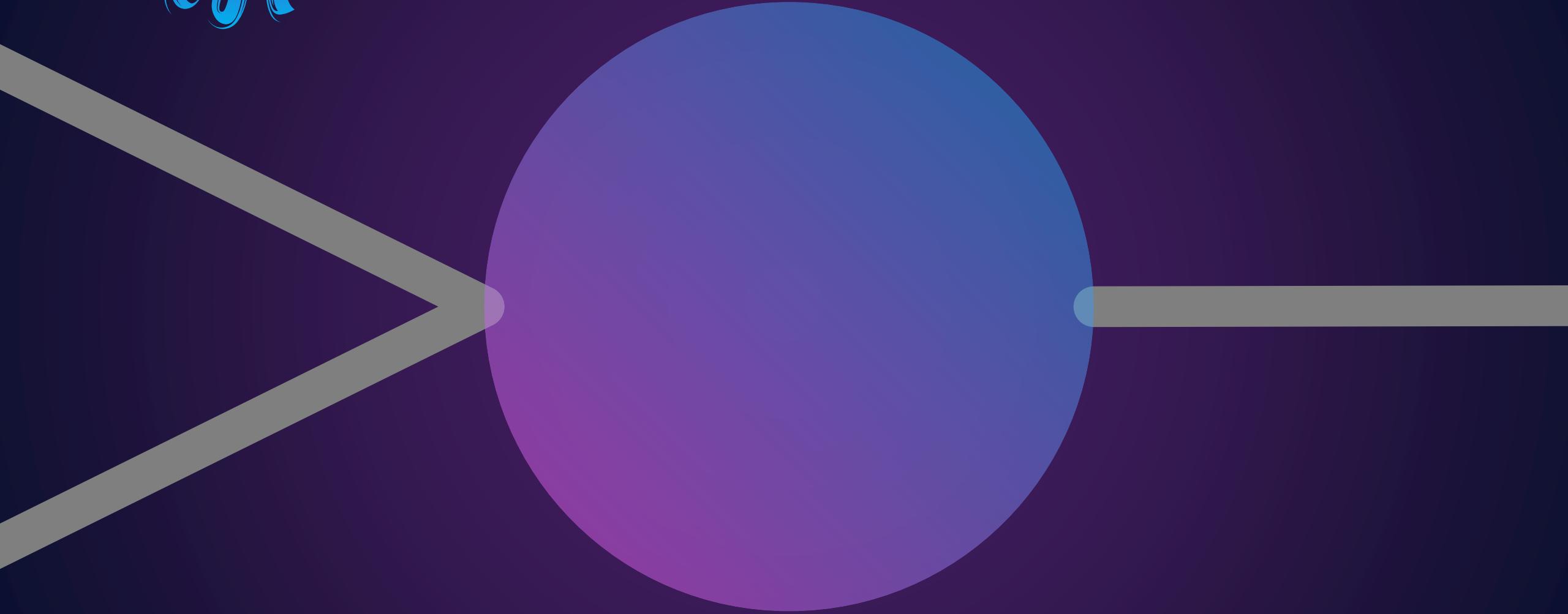
*Neuronale Netze*



# Neuronale Netze



# Newton



# Newton

$w_1$

$w_2$



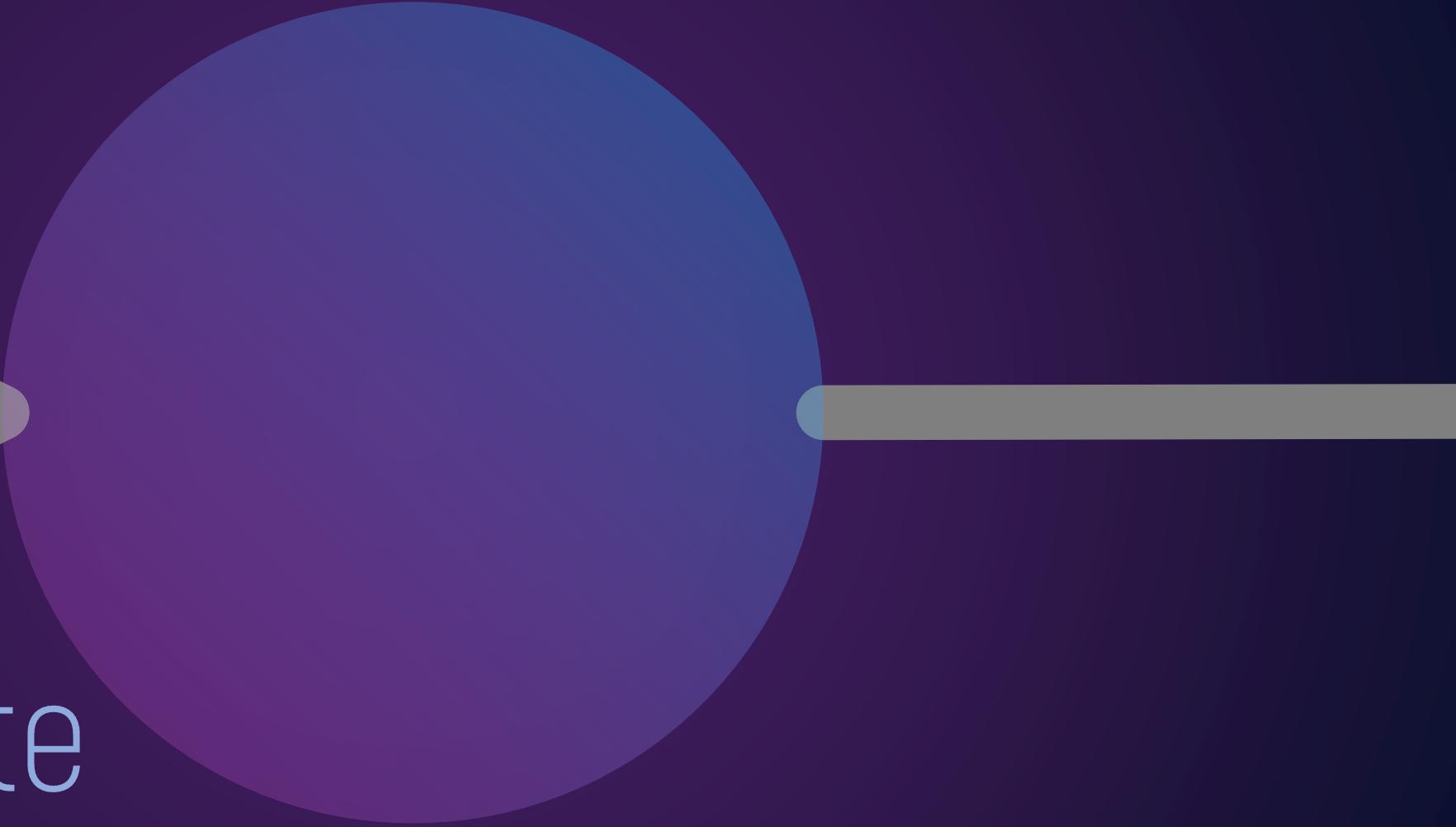
# Neuron

$w_1$

$w_2$

Gewichte

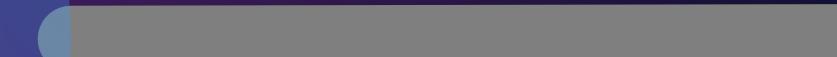
weights



# Newton

$w_1$

$w_2$



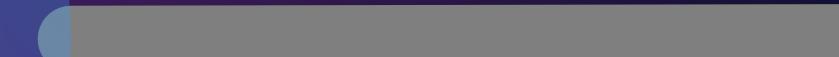
# Newton

$x_1$

$w_1$

$w_2$

$x_2$



# Newton

$$z = \begin{matrix} x_1 w_1 \\ + x_2 w_2 \end{matrix}$$

# Newton

$$z = \sum_i w_i x_i$$

# Newton

$$z = \sum_i w_i x_i$$

# Newton

$$z = \sum_i w_i x_i$$

$$a = f( )$$

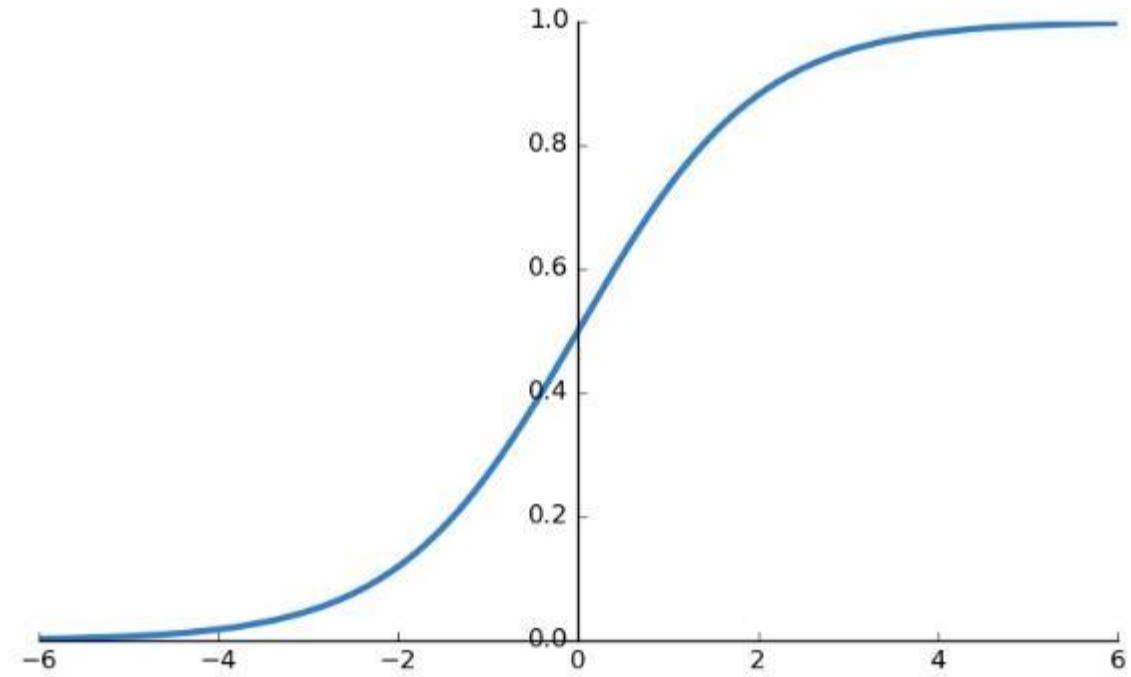
# Newton

$$z = \sum_i w_i x_i$$

$$a = f(z)$$

$$a = f(z)$$

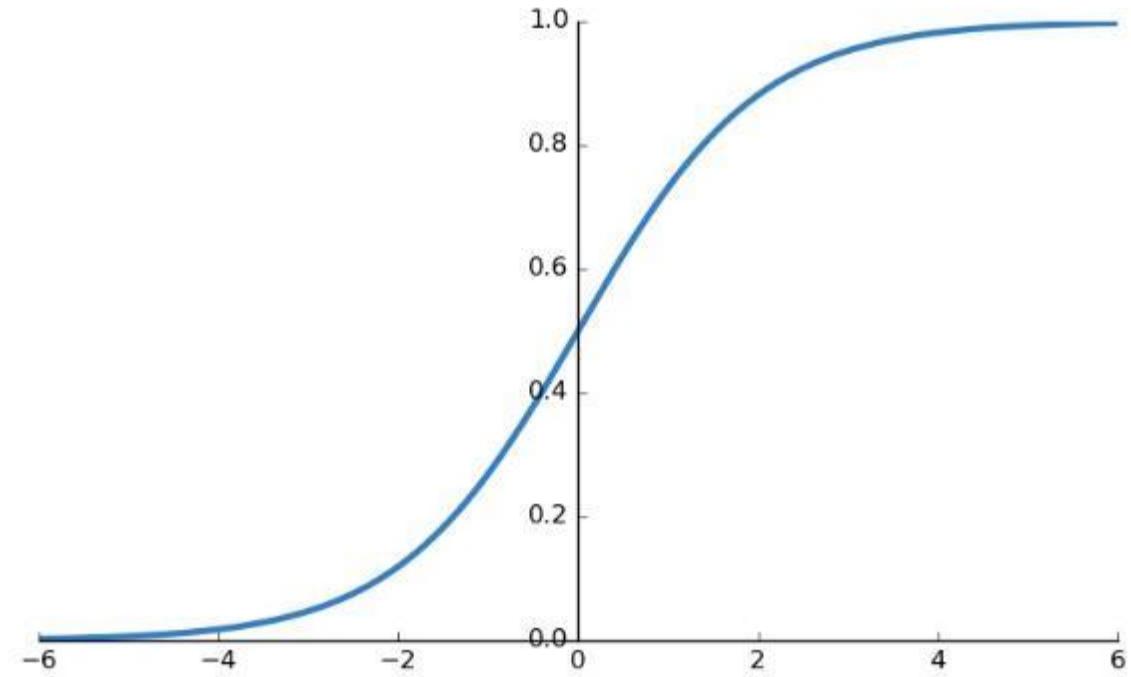
Aktivierungs  
funktion



[http://ronny.rest/blog/post\\_2017\\_08\\_10\\_sigmoid/](http://ronny.rest/blog/post_2017_08_10_sigmoid/)

$$a = f(z)$$

Aktivierungs  
funktion



[http://ronny.rest/blog/post\\_2017\\_08\\_10\\_sigmoid/](http://ronny.rest/blog/post_2017_08_10_sigmoid/)

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

# Newton

$$z = \sum_i w_i x_i$$

$$a = f(z)$$

# Newton

$$z = \sum_i w_i x_i$$

$$a = f(z)$$

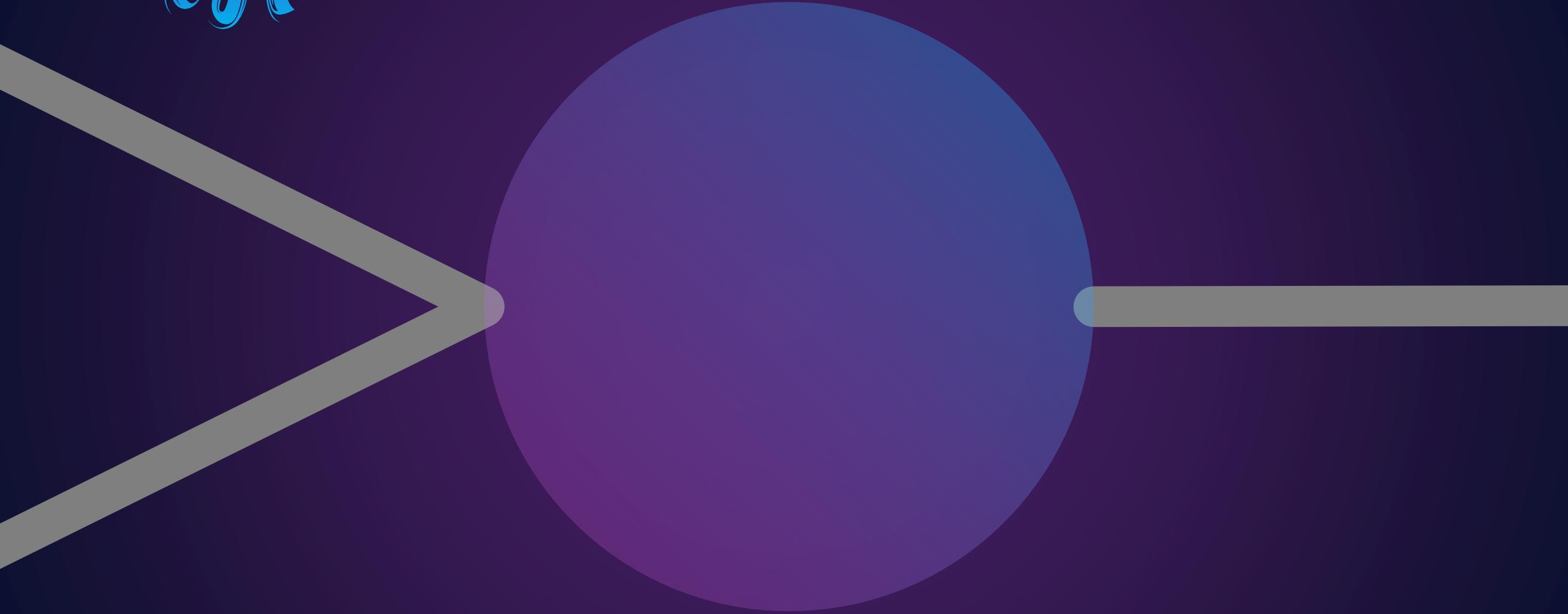
$$a \cdot w_3$$

# Newton

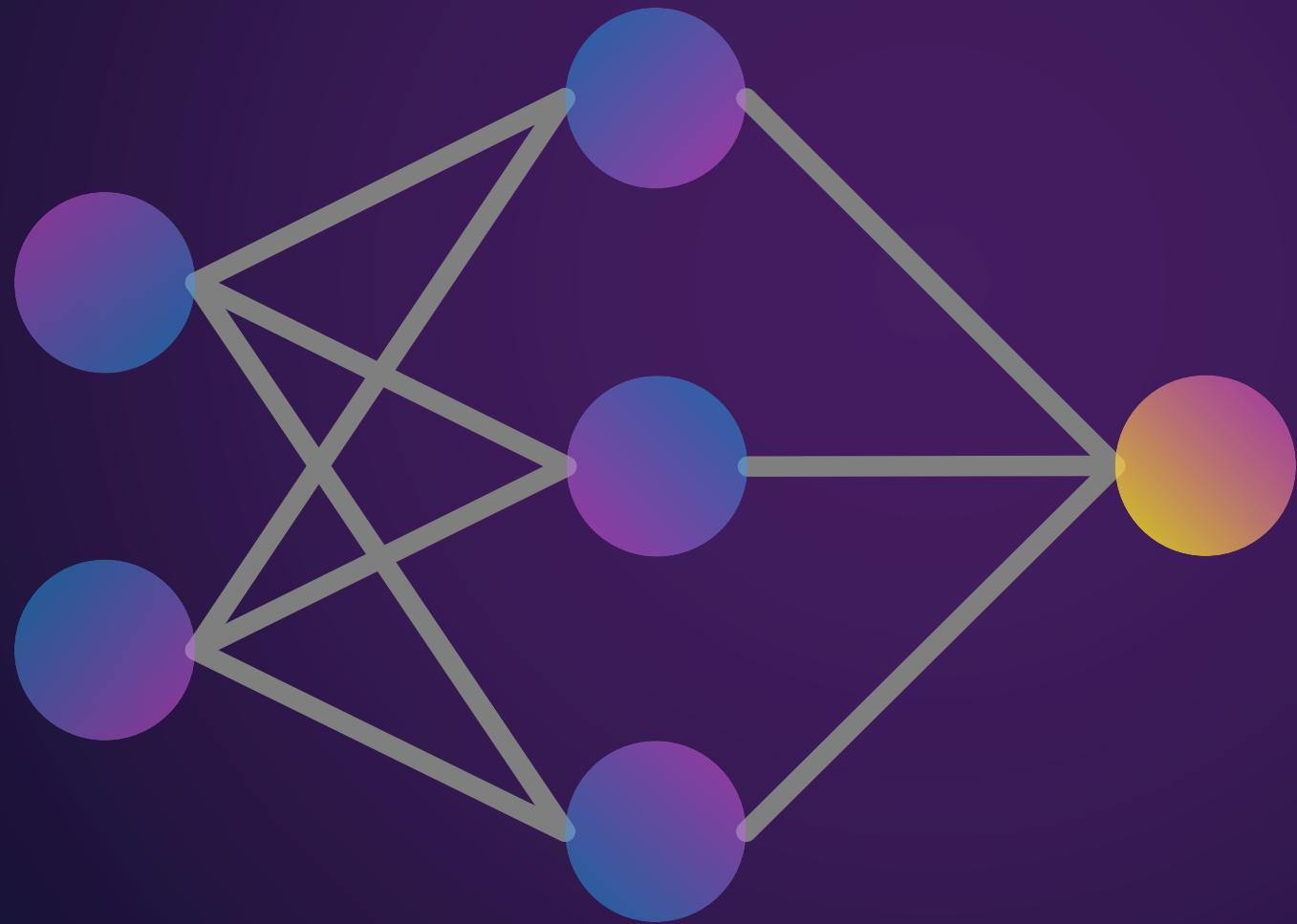
$$z = \sum_i w_i x_i$$

$$a = f(z)$$

# Newton

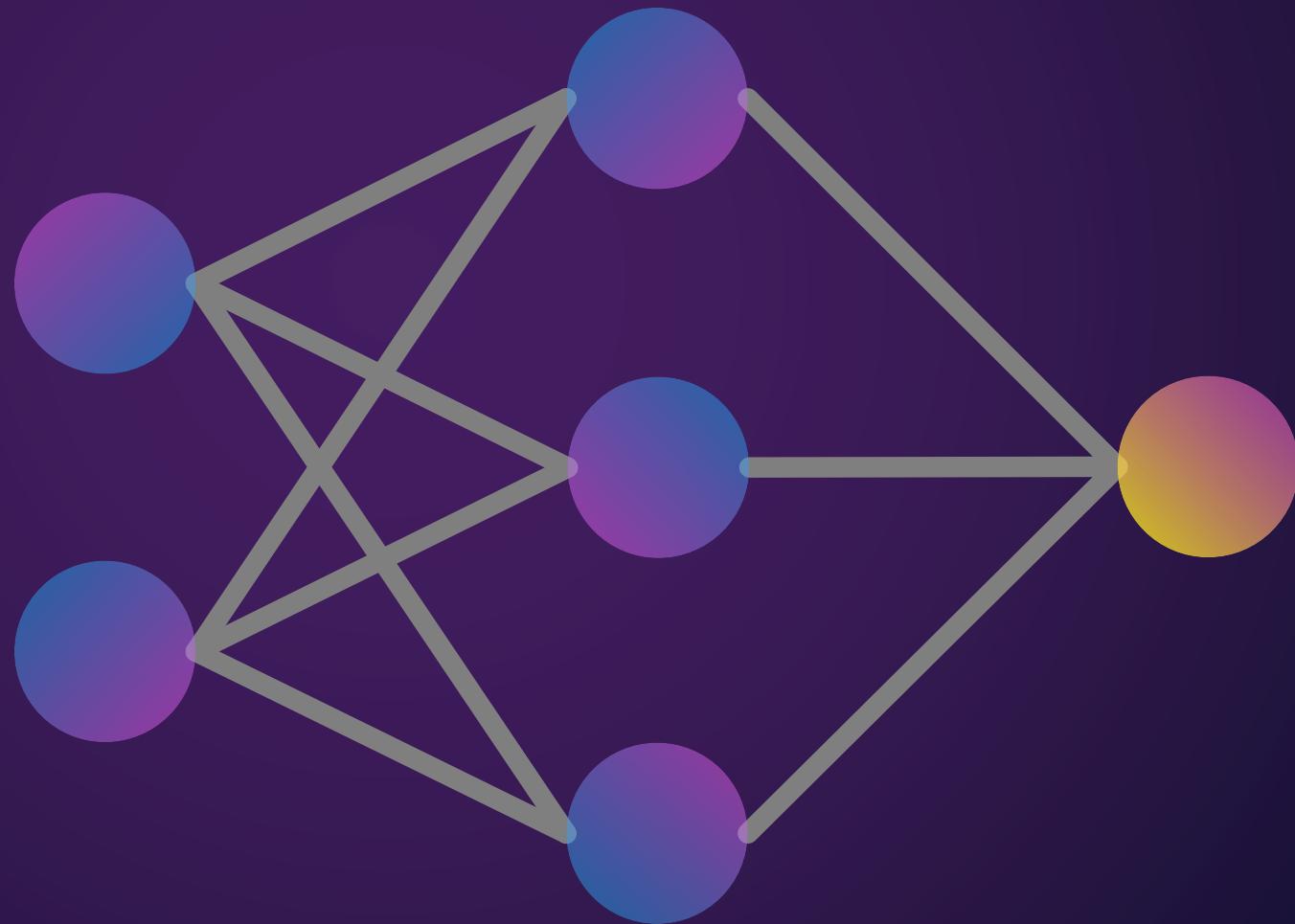


# Neuronale Netze



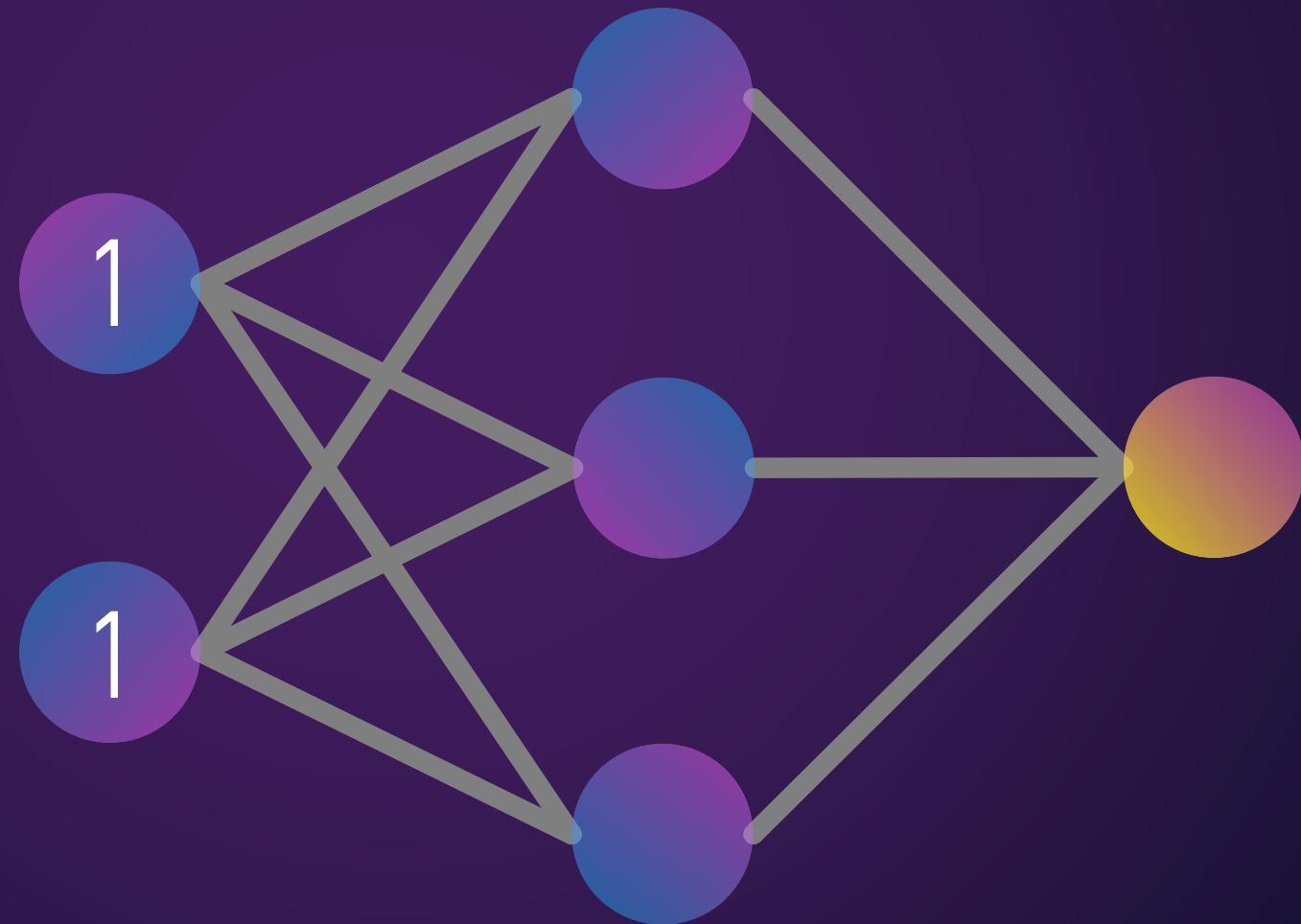
# *Lernalgorithmen*

**Supervised  
Learning**



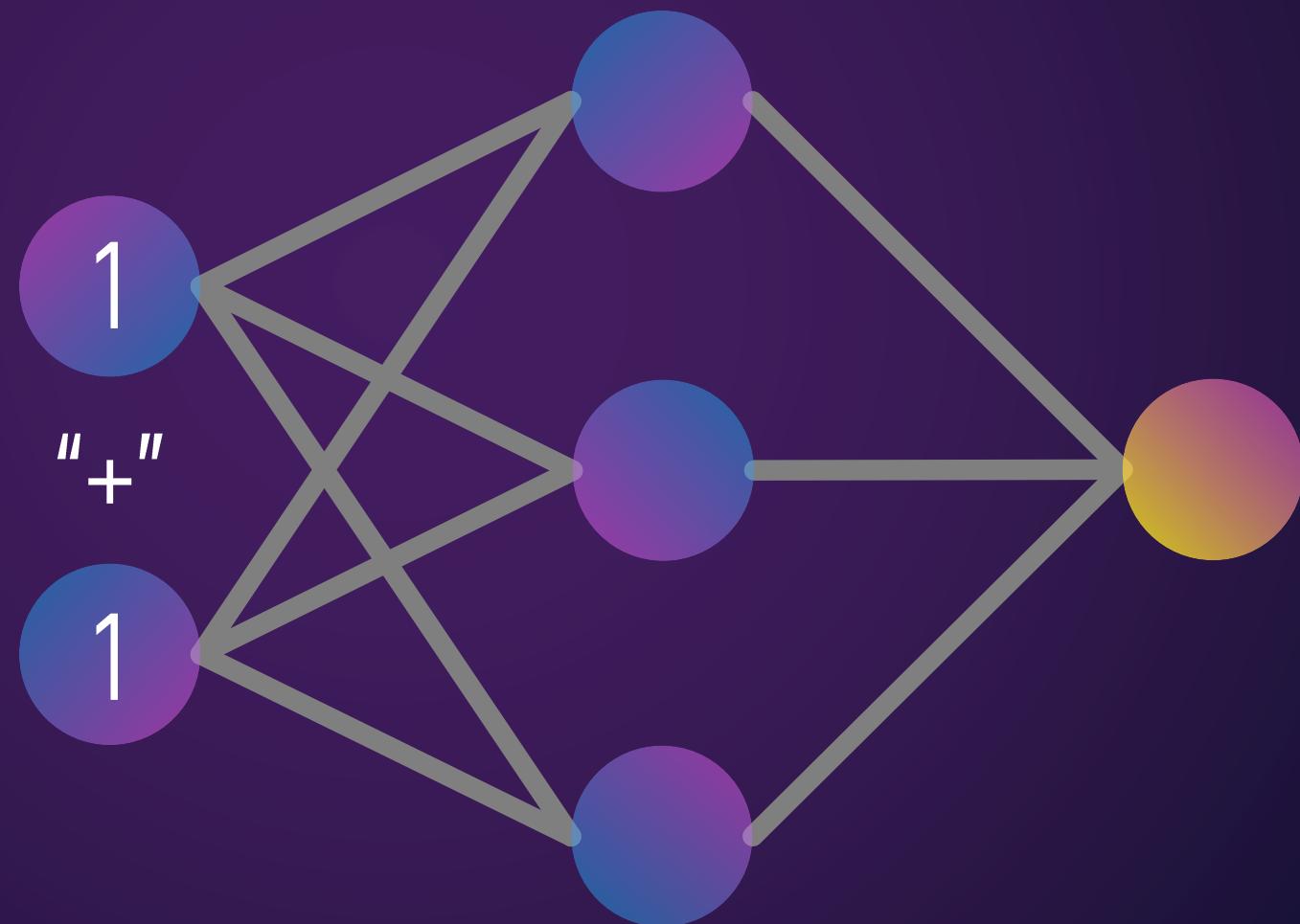
# Lernalgorithmen

Supervised  
Learning



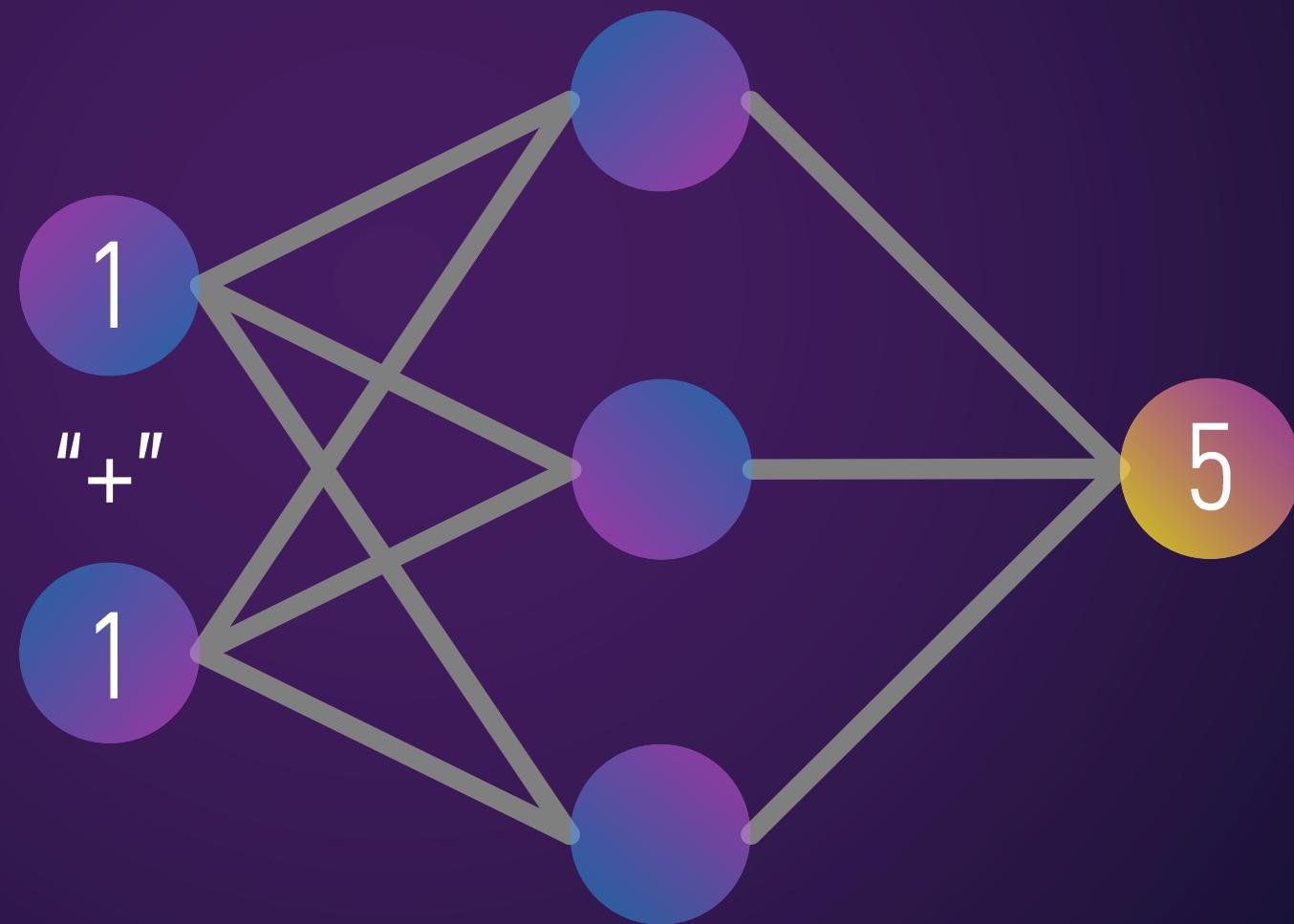
# Lernalgorithmen

Supervised  
Learning



# Lernalgorithmen

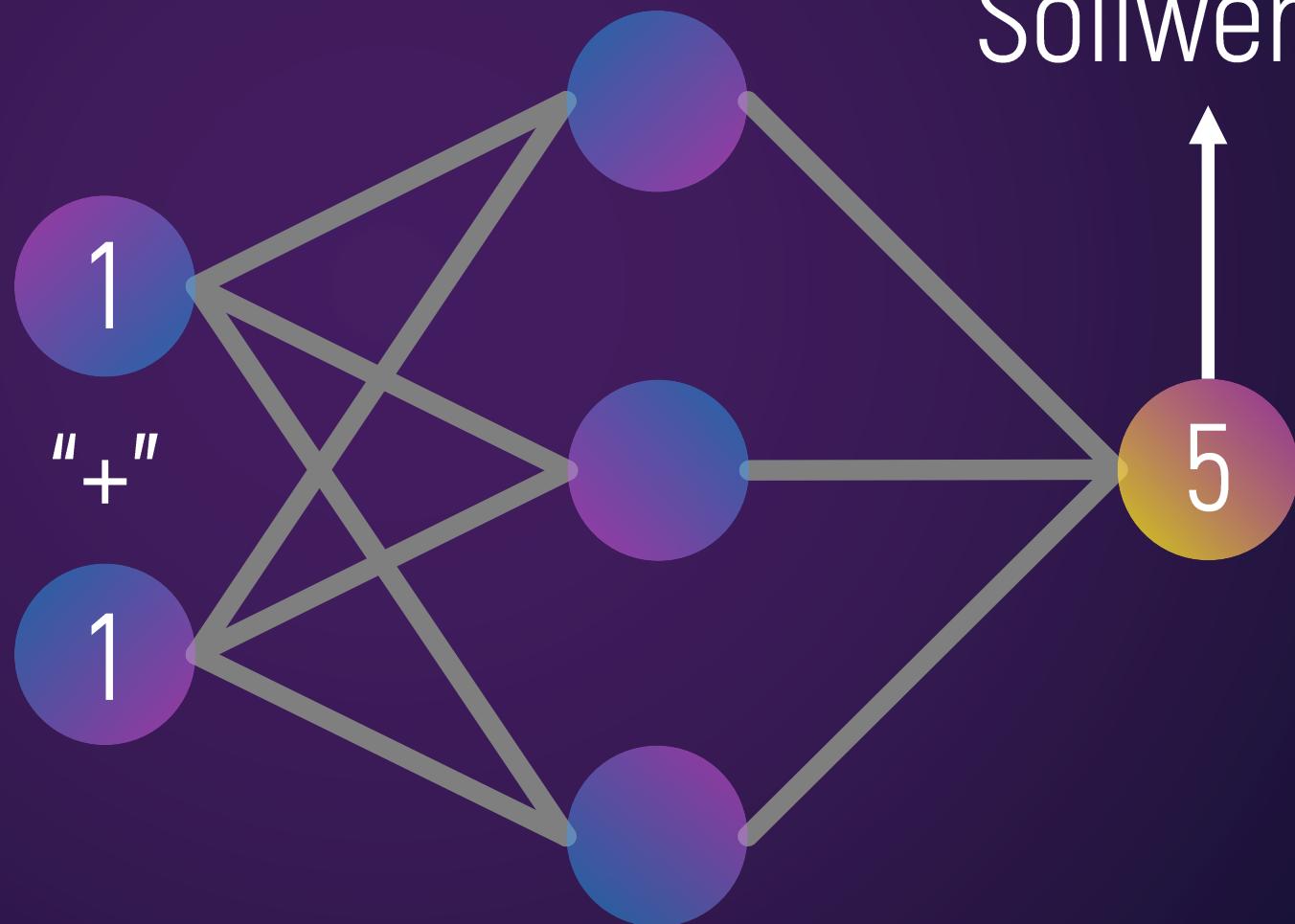
Supervised  
Learning



# Lernalgorithmen

Supervised  
Learning

Istwert: 5  
Sollwert: 2



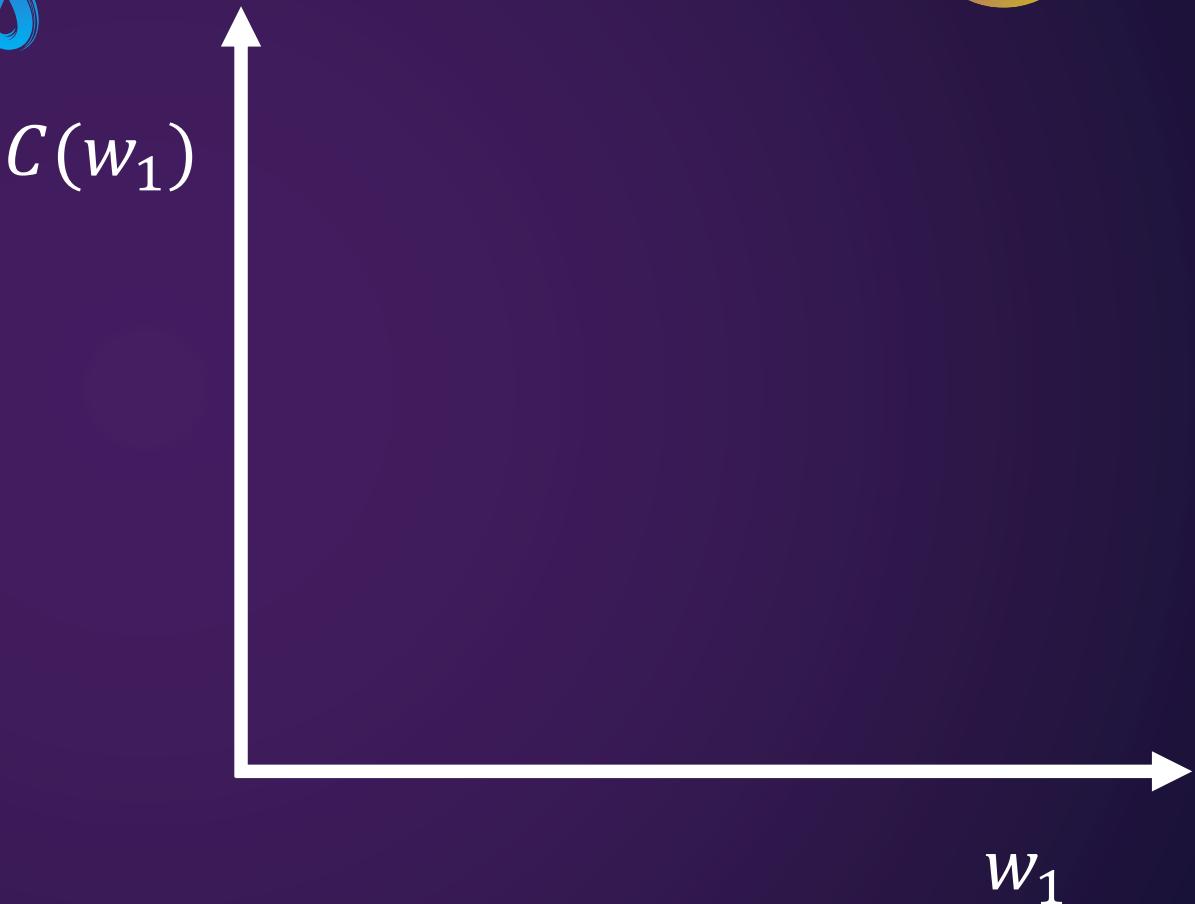
# *Lernalgorithmen*



## Gradient Descent

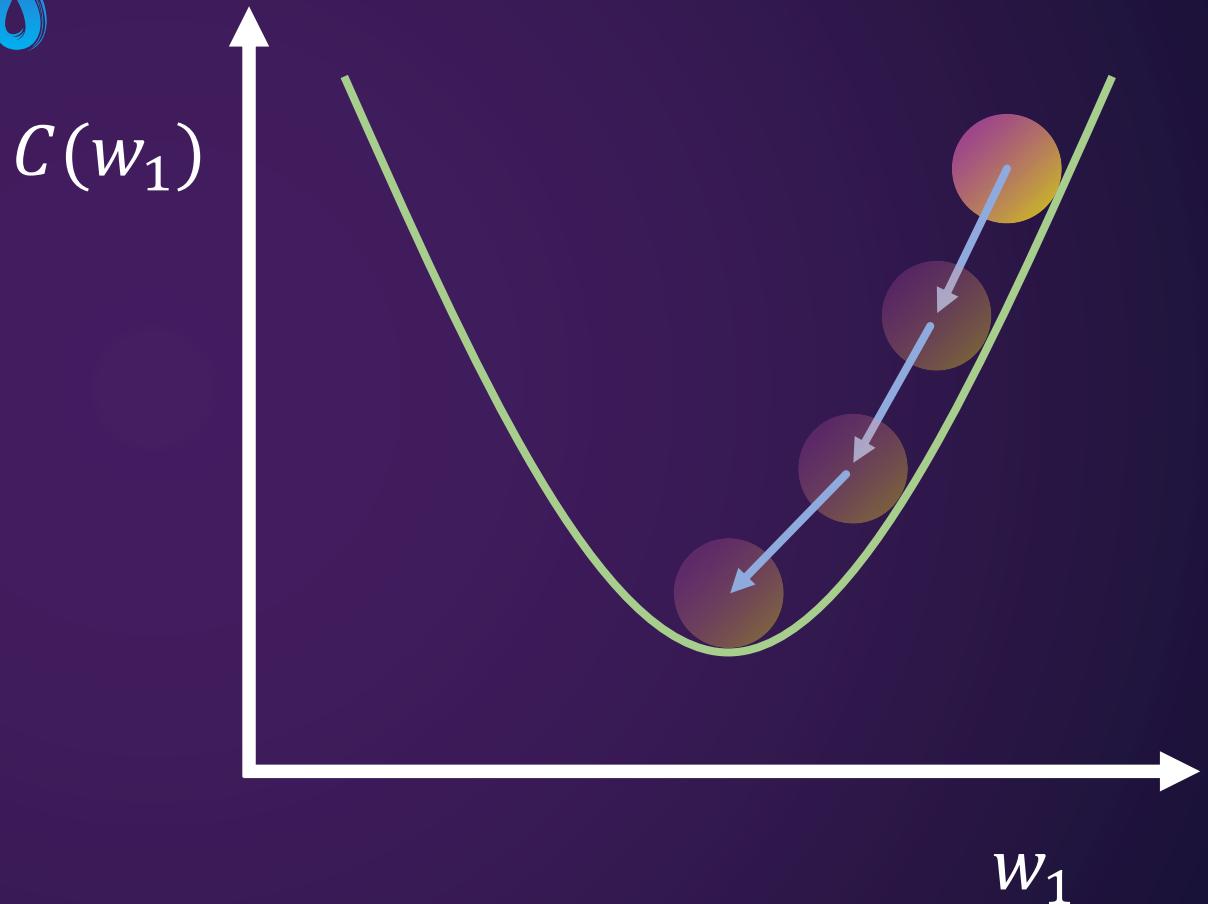
# Lernalgorithmus

Gradient  
Descent



# *Lernalgorithmen*

## Gradient Descent



The background features a central yellow circle surrounded by several concentric rings in shades of blue, purple, and pink. The word "Daten" is written in a large, stylized, cursive font that follows the color gradient of the rings.

Daten



*Training*



*Test*

# *Lernalgorithmen*

Hyper-  
Parameter



# *Lernalgorithmen*

## Hyper- Parameter

# Lernalgorithmus

Hyper-  
Parameter



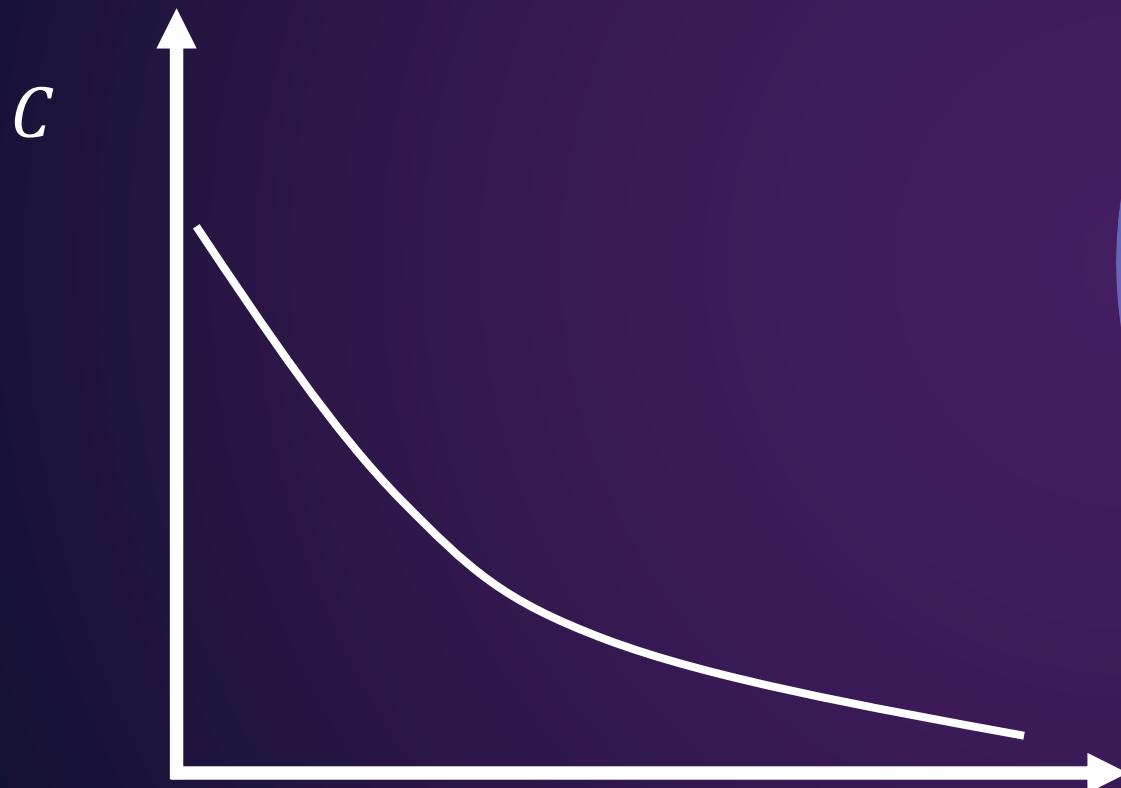
Epochen

# Lernalgorithmus



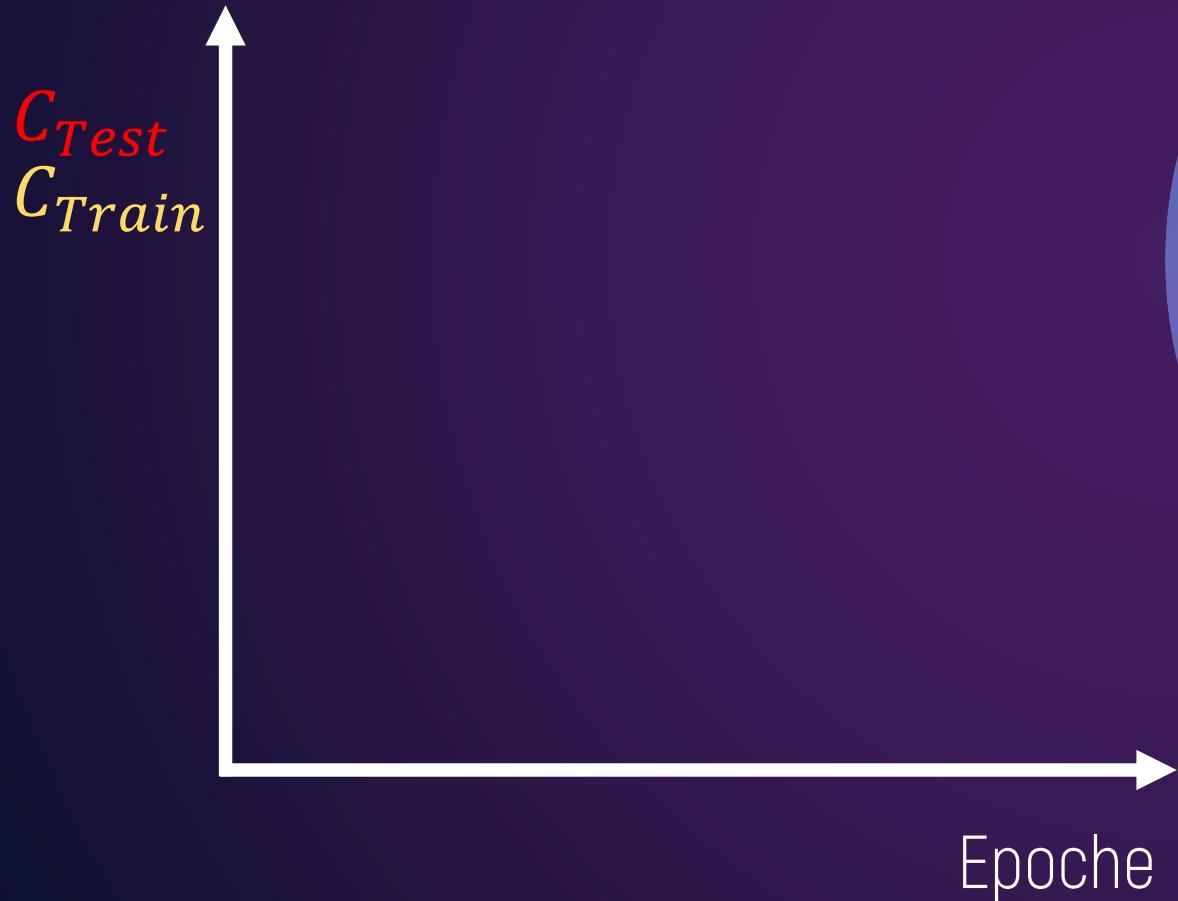
Epochen

# Lernalgorithmus



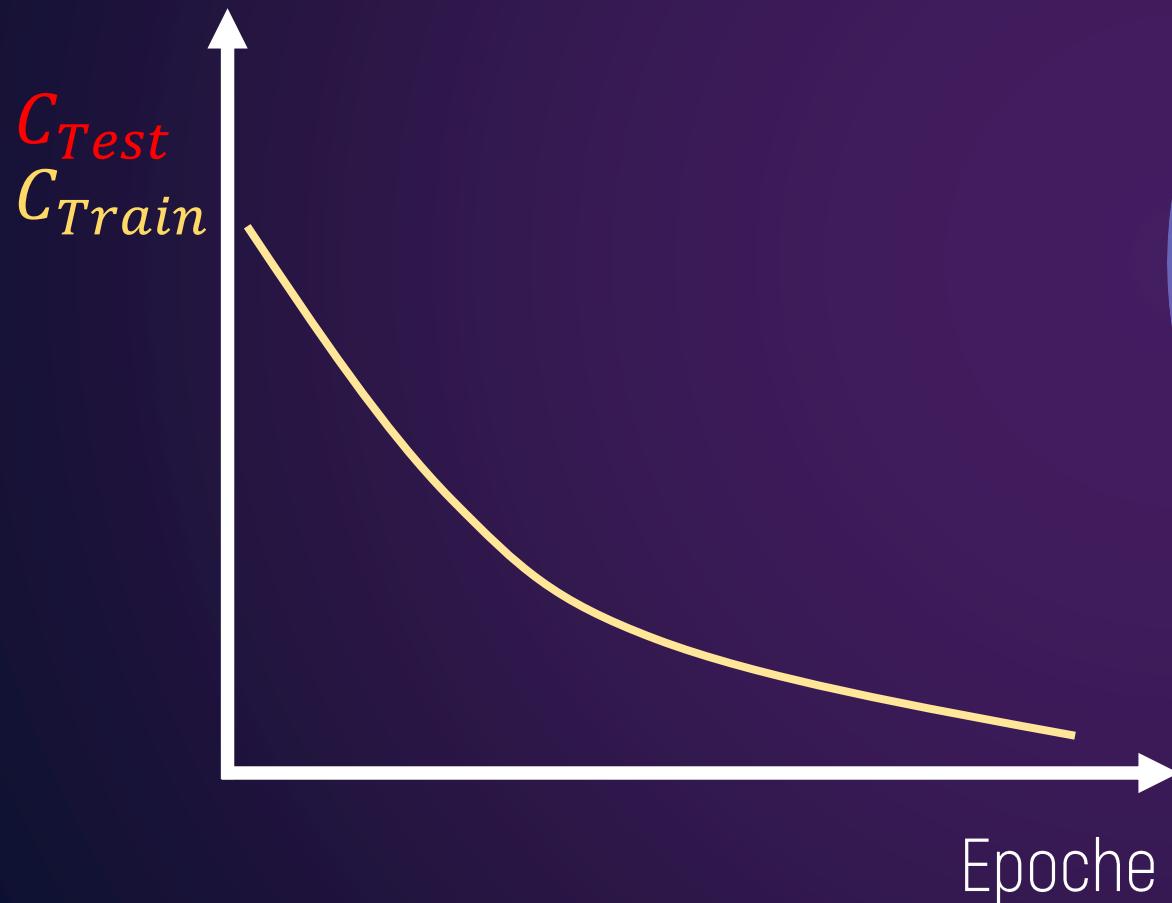
Epochen

# Lernalgorithmus



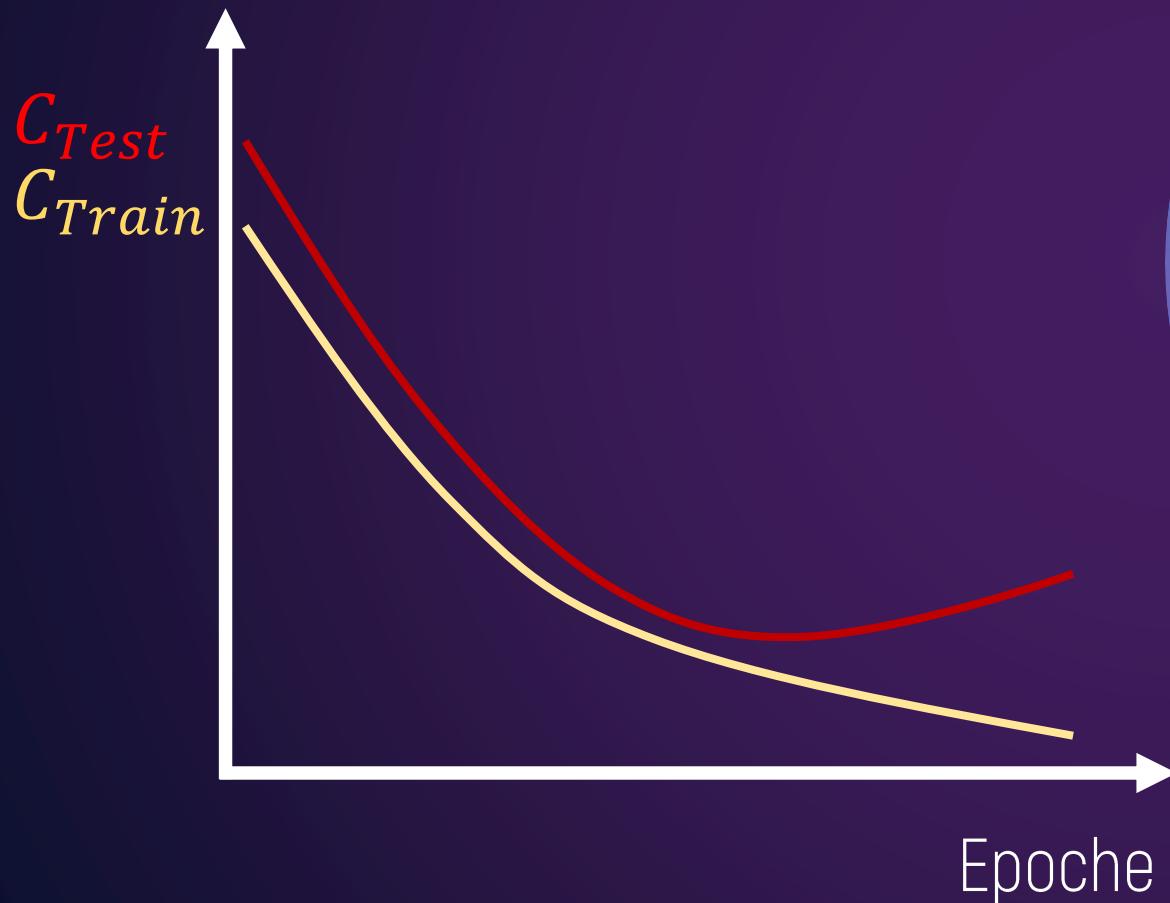
Epochen

# Lernalgorithmus



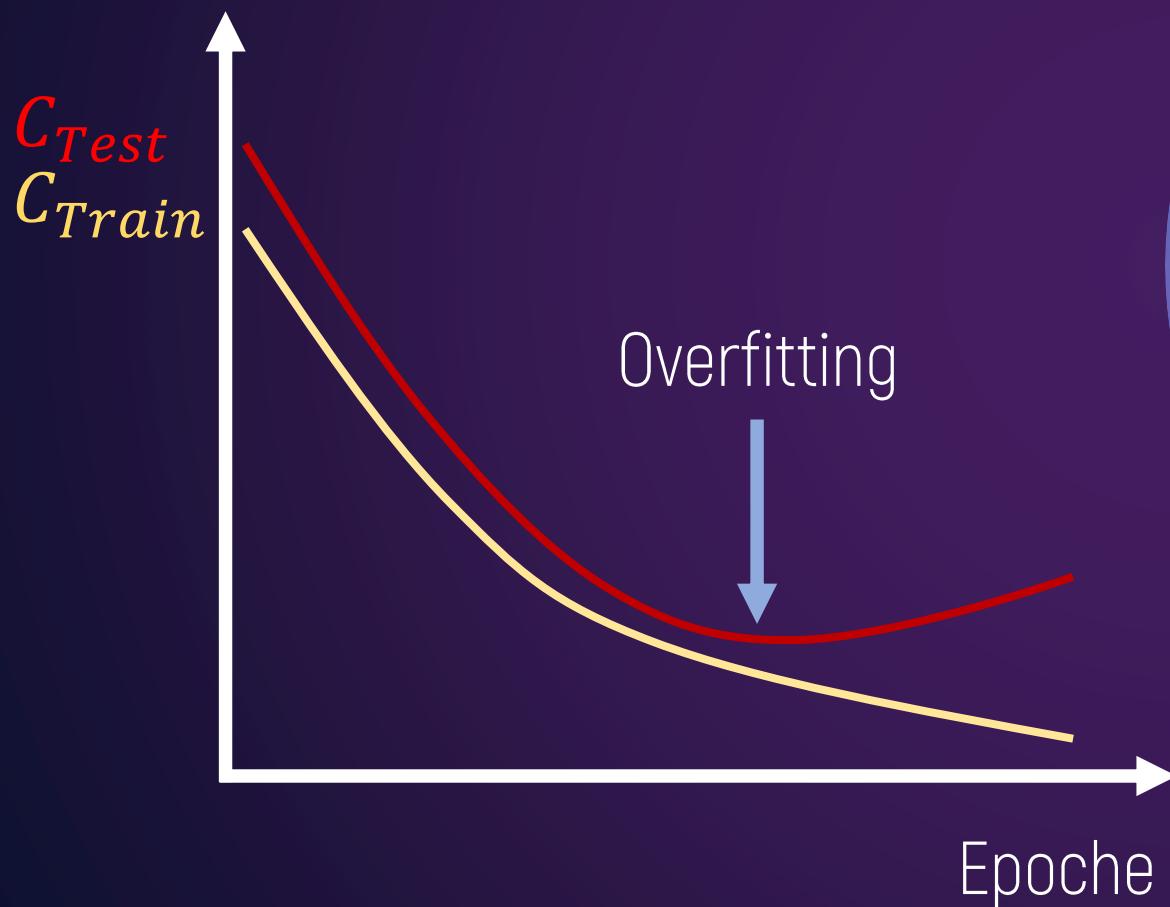
Epochen

# Lernalgorithmus



Epochen

# Lernalgorithmus



## Epochen

# Lernalgorithmus

Hyper-  
Parameter



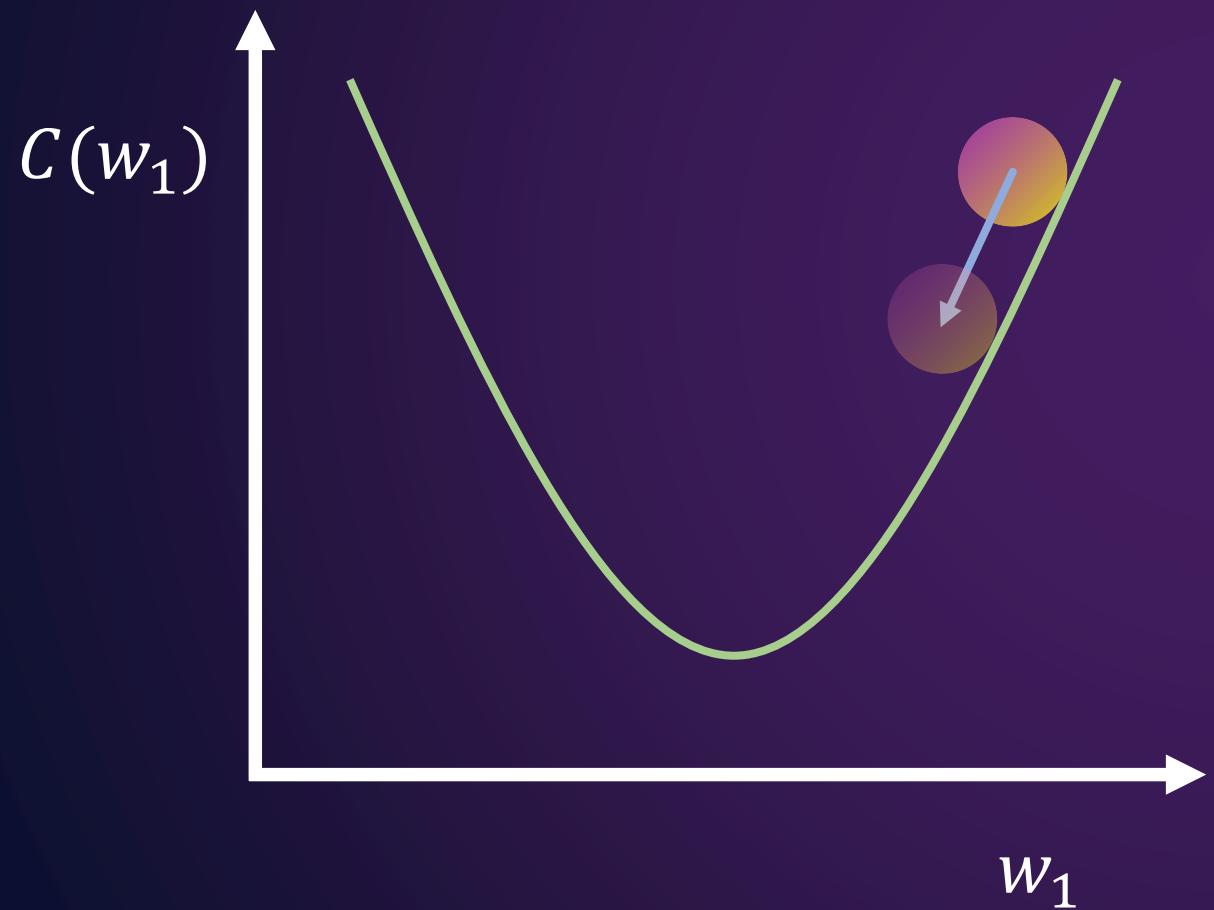
Epochen

# Lernalgorithmus

Hyper-  
Parameter

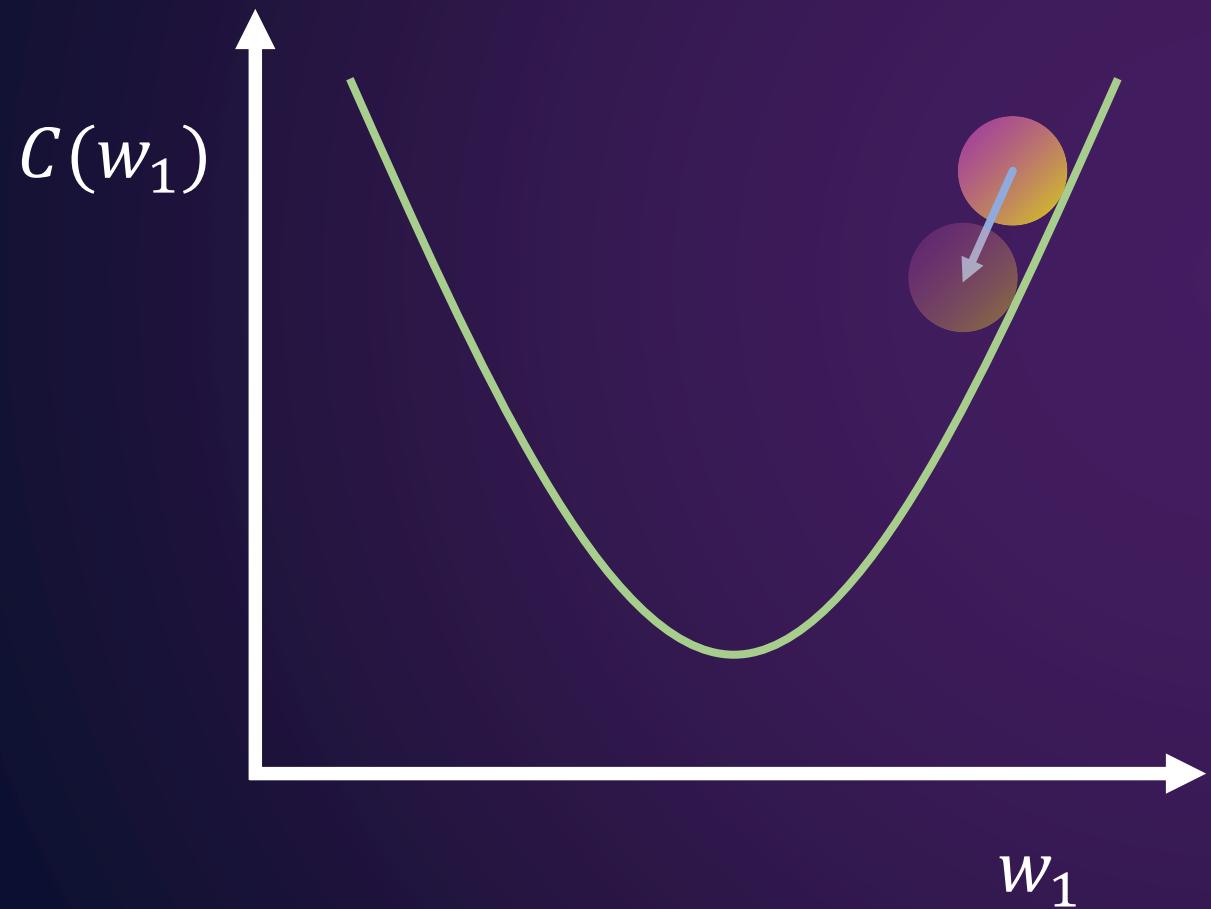
Lernrate

# Lernalgorithmus



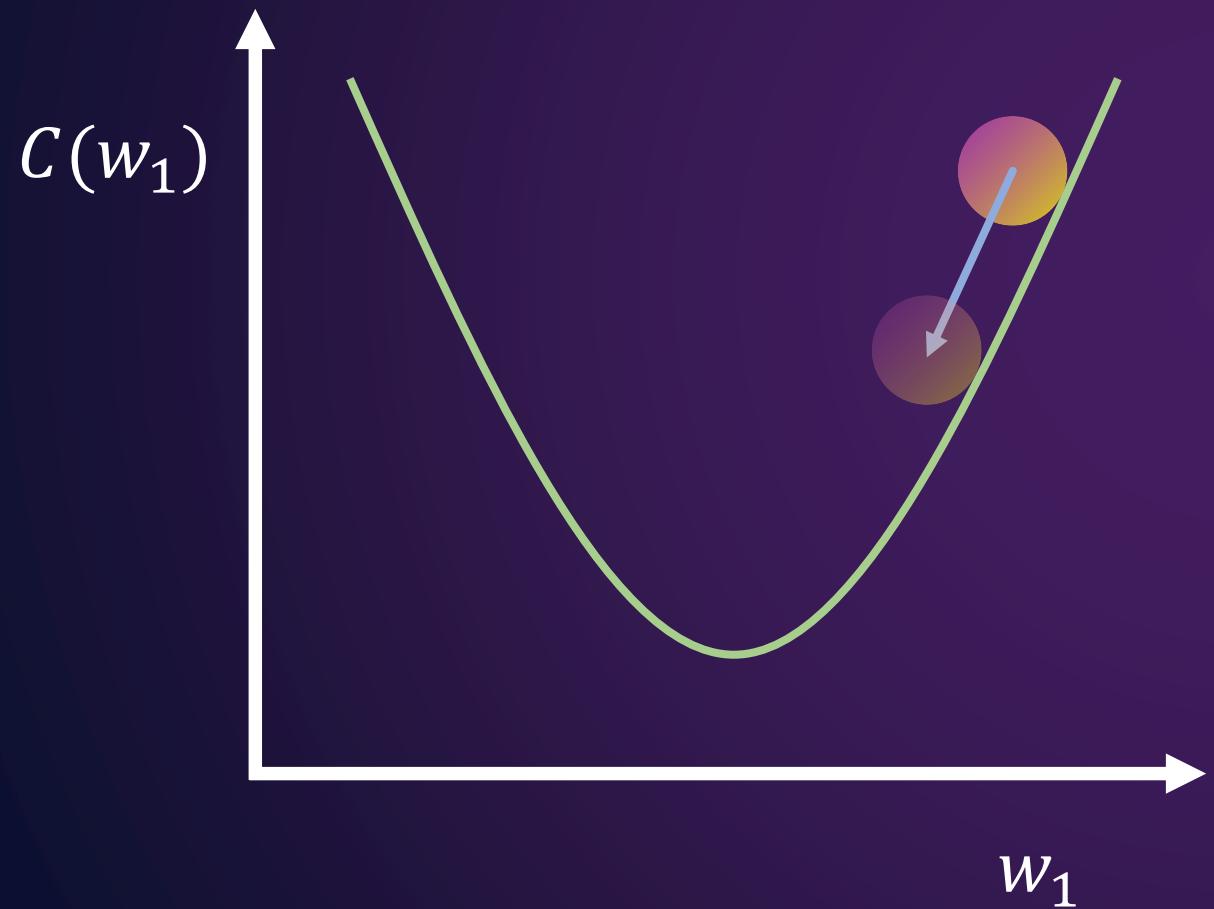
Lernrate

# Lernalgorithmus



Lernrate

# Lernalgorithmus



Lernrate

# Lernalgorithmus

Hyper-  
Parameter

Lernrate

# *Lernalgorithmen*

Hyper-  
Parameter



Batch Size

# Lernalgorithmen



## Batch Size

# Lernalgorithmus

Daten

Batch Size

# Lernalgorithmen

Batch



Batch Size

# *Lernalgorithmen*

Hyper-  
Parameter



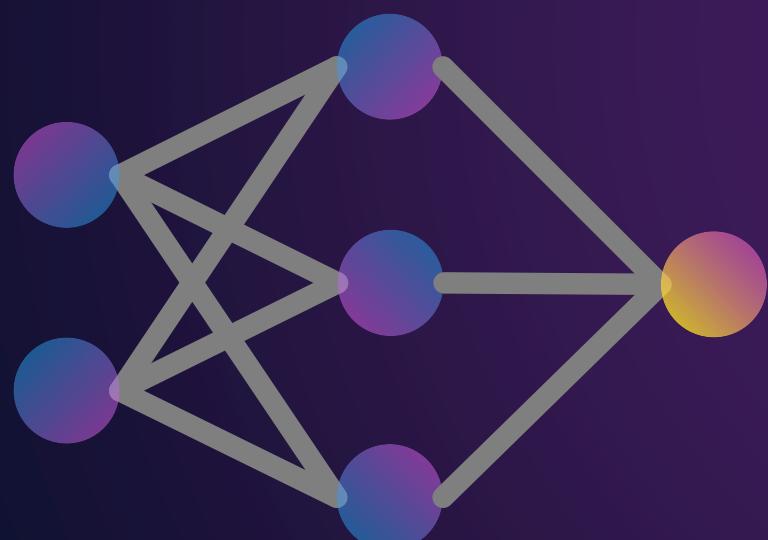
Batch Size

# Lernalgorithmus

Hyper-  
Parameter

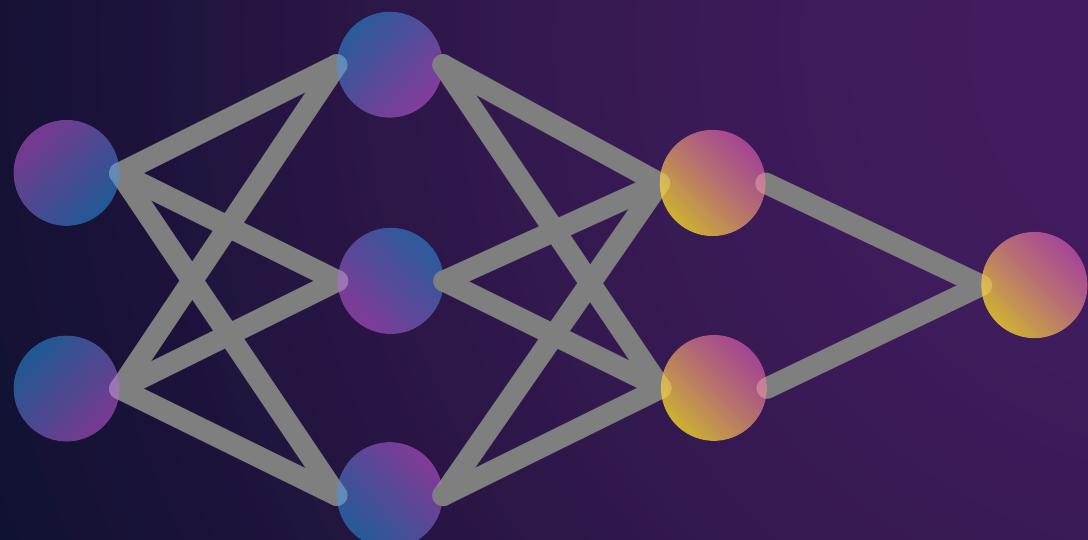
Anzahl der  
Schichten

# Lernalgorithmus



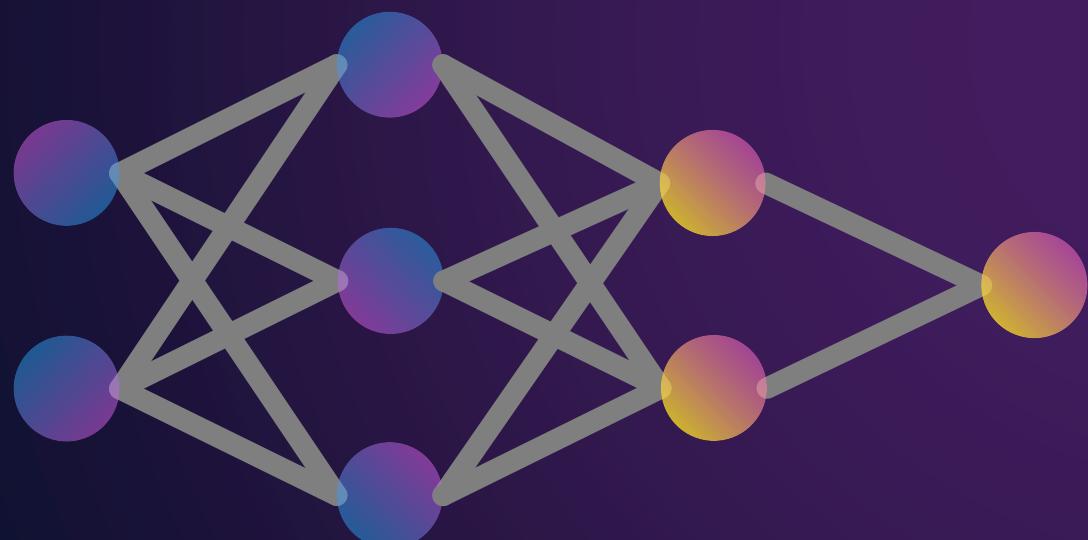
Anzahl der  
Schichten

# Lernalgorithmen



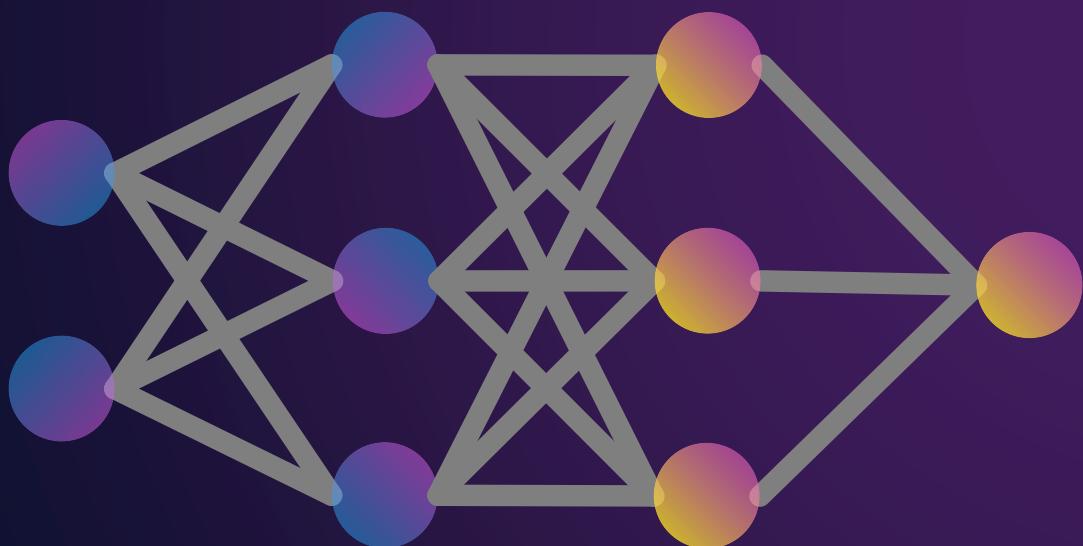
Anzahl der  
Schichten

# Lernalgorithmus



Anzahl der  
Neuronen

# Lernalgorithmen



Anzahl der  
Neuronen

# Lernalgorithmus

Hyper-  
Parameter

Anzahl der  
Neuronen

# *Lernalgorithmen*

Hyper-  
Parameter





Tension  
Flow



Tension



Flow

# Was ist ein Tensor



Tensor

# Was ist ein Tensor

Eine Zahl

$$s = a$$



# Tensor<sup>n</sup>

# Was ist ein Tensor

Ein Vektor

$$\vec{v} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$



Tensor<sup>n</sup>

# Was ist ein Tensor

Eine Matrix

$$M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$



Tensor<sup>n</sup>

# Was ist ein Tensor

N-Dimensional

$$T = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$



Tensor<sup>1</sup>



Tension



Flow



Tension



Flow



Flow

Wie fließen  
Tensor

# Tensor Operation

Wie fließen  
Tensor



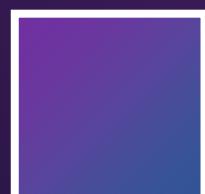
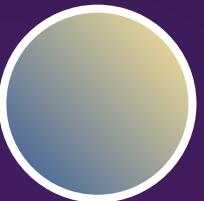
flow



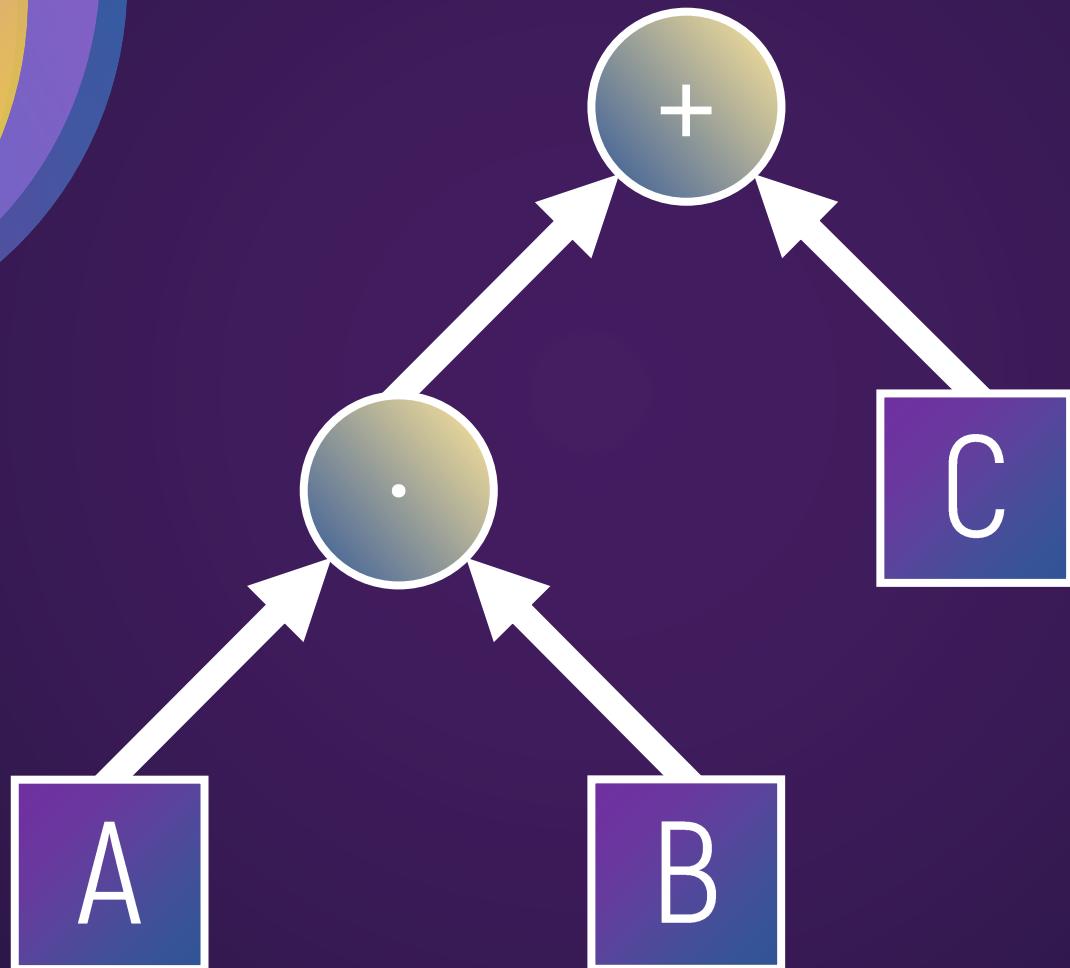


Flow

Wie fließen  
Tensor



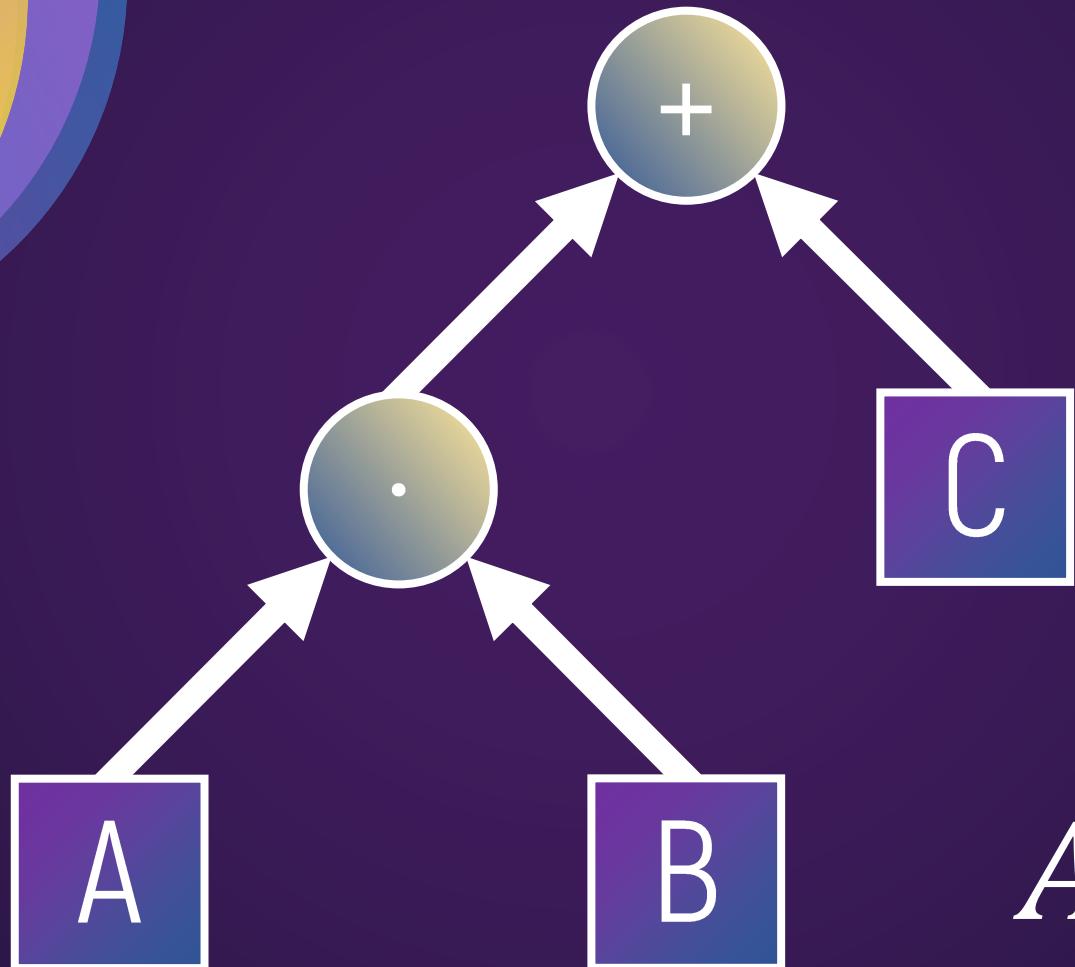
# Wie fließen Tensor



flow



Flow



Wie fließen  
Tensor

$$A \cdot B + C$$



*Tensor*

Was  
Tensoren



*Flow*

Wie  
Operationen



*Tensor*

**Was**  
Tensoren



*Flow*

**Wie**  
Operationen



Keras

The TensorFlow logo is displayed on a large, semi-transparent circular background. The background has concentric rings in shades of yellow, orange, pink, and purple. The text "TensorFlow" is written in a bold, black, sans-serif font.

TensorFlow

The Keras logo is displayed on a large, semi-transparent circular background. The background has concentric rings in shades of blue, purple, pink, orange, and yellow. The text "Keras" is written in a bold, black, sans-serif font.

Keras

Tensor  
Flow

Keras

Tensor  
Flow



Keras

Tensor  
Flow

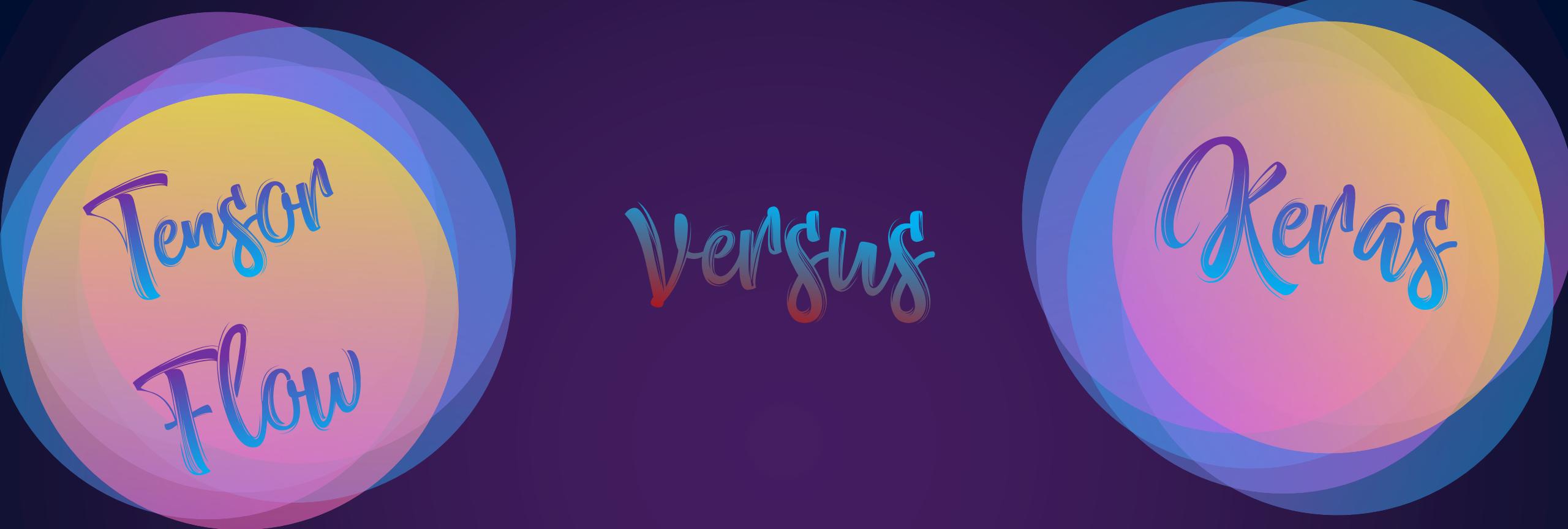


Keras

Tensor  
Flow



Keras



Tensor  
Flow

- + Schneller
- + Flexibler

Versus



Keras

- + Leserlicher
- + Abstrakter

# In der Sprache Python

- Neuronales Netz
- Quellcode der das Netz erzeugt und trainiert
- Vergleich von Keras und TensorFlow Code

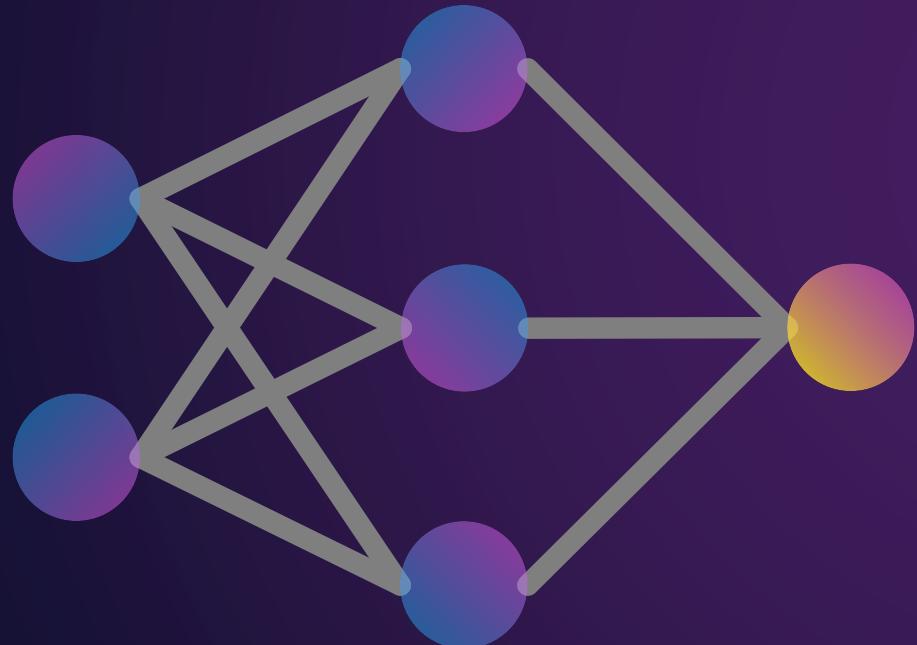


Umsetzung

Modell

Keras

# Modell



# Keras

Modell

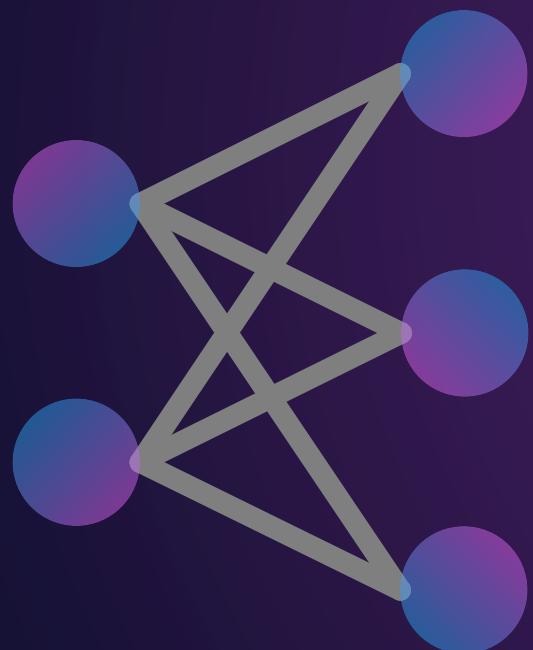
Keras

# Modell

# Keras

```
model = Sequential()
```

# Modell

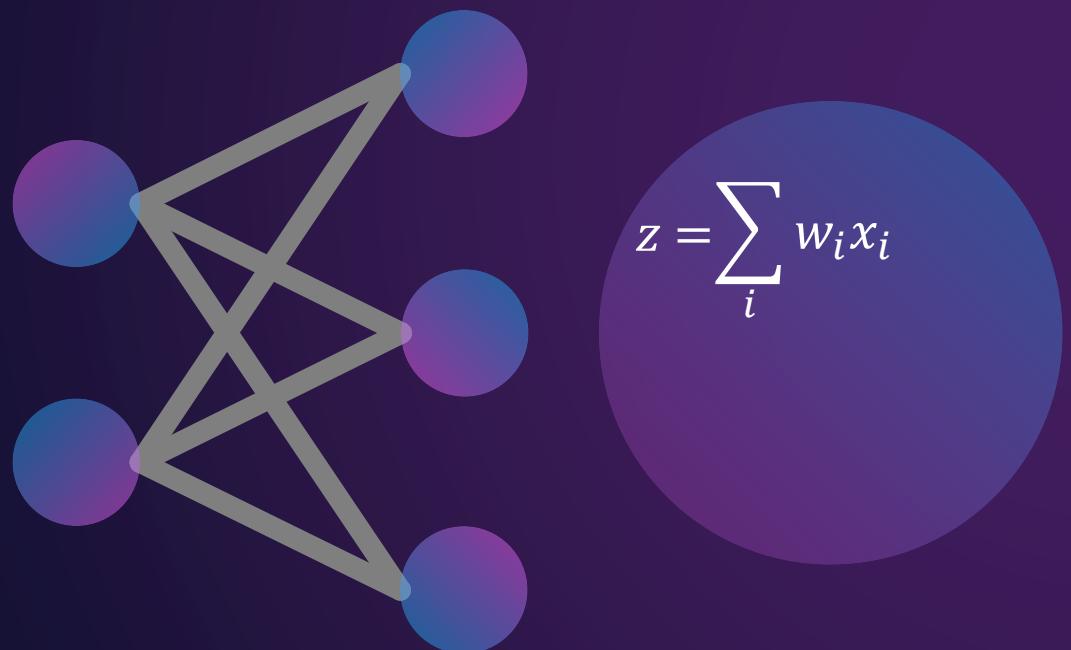


# Keras

```
model = Sequential()
```

```
model.add(Dense(3, input_dim=2))
```

# Modell

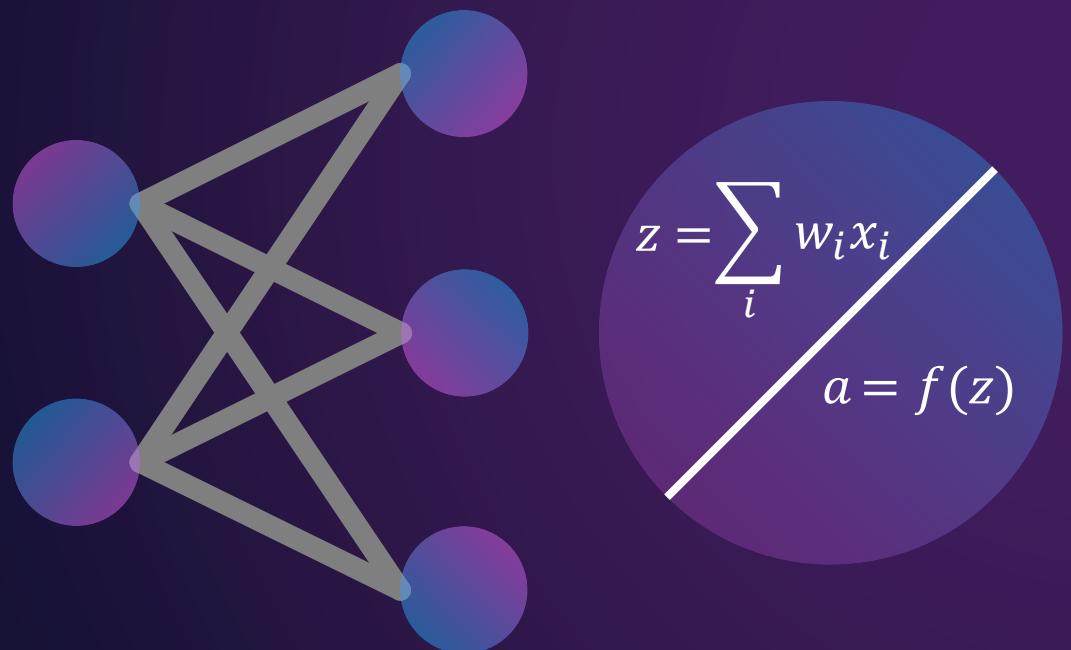


# Keras

```
model = Sequential()
```

```
model.add(Dense(3, input_dim=2))
```

# Modell

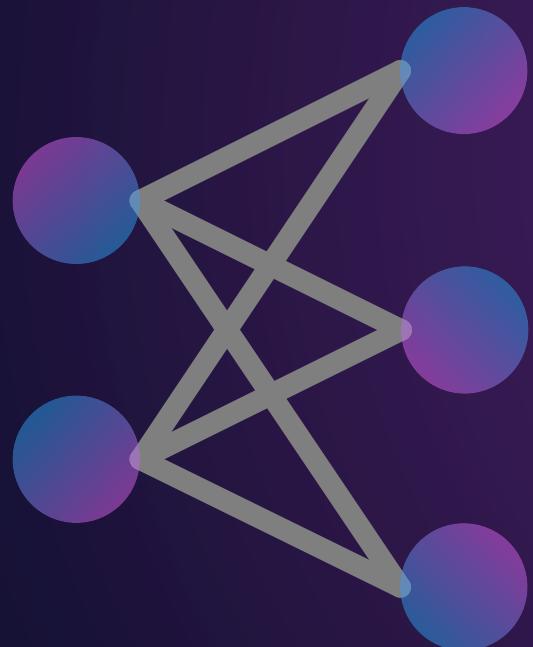


# Keras

```
model = Sequential()
```

```
model.add(Dense(3, input_dim=2))  
model.add(Activation('sigmoid'))
```

# Modell

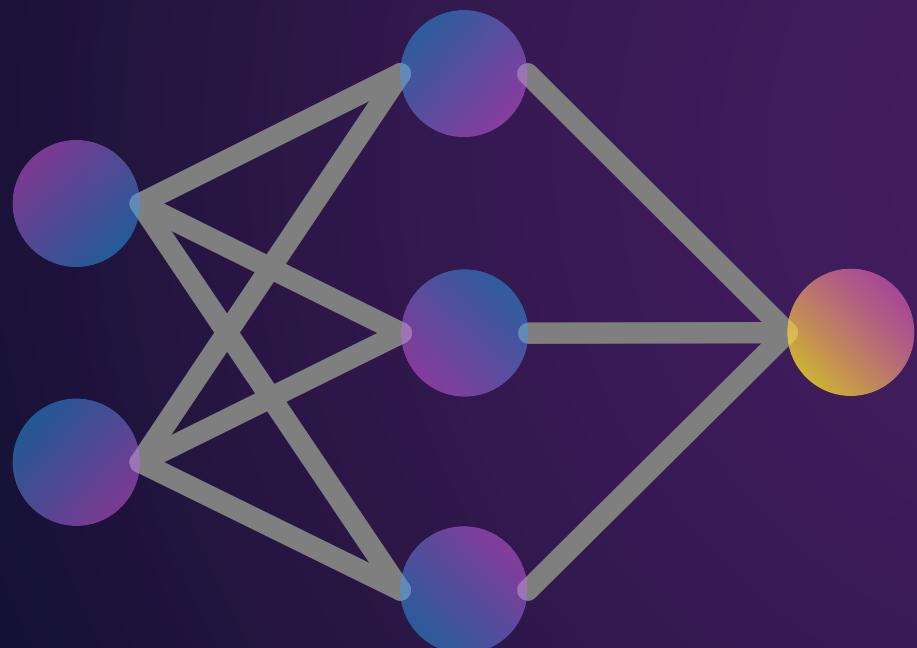


# Keras

```
model = Sequential()
```

```
model.add(Dense(3, input_dim=2))  
model.add(Activation('sigmoid'))
```

# Modell



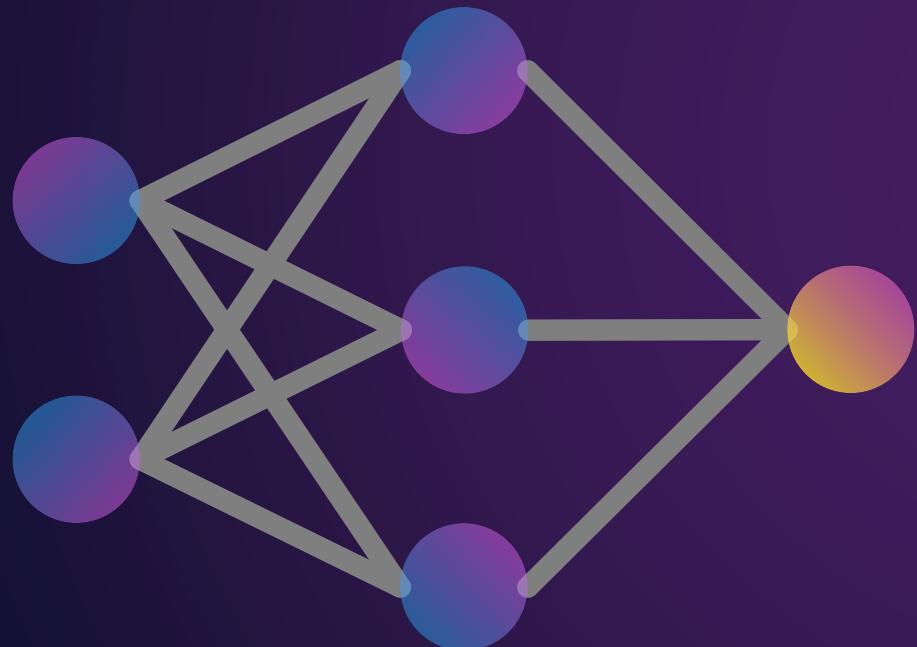
# Keras

```
model = Sequential()
```

```
model.add(Dense(3, input_dim=2))  
model.add(Activation('sigmoid'))
```

```
model.add(Dense(1))
```

# Modell



# Keras

```
model = Sequential()
```

```
model.add(Dense(3, input_dim=2))  
model.add(Activation('sigmoid'))
```

```
model.add(Dense(1))  
model.add(Activation('sigmoid'))
```

# Keras

```
model = Sequential()
```

```
model.add(Dense(3, input_dim=2))  
model.add(Activation('sigmoid'))
```

```
model.add(Dense(1))  
model.add(Activation('sigmoid'))
```

# TensorFlow

# Keras

```
model = Sequential()  
  
model.add(Dense(3, input_dim=2))  
model.add(Activation('sigmoid'))
```

```
model.add(Dense(1))  
model.add(Activation('sigmoid'))
```

# TensorFlow

```
x = tf.placeholder(  
    shape=[None, 2],  
    dtype=tf.float32)
```

# Keras

```
model = Sequential()  
  
model.add(Dense(3, input_dim=2))  
model.add(Activation('sigmoid'))  
  
model.add(Dense(1))  
model.add(Activation('sigmoid'))
```

# TensorFlow

```
x = tf.placeholder(  
    shape=[None, 2],  
    dtype=tf.float32)  
  
l1 = tf.layers.dense(x, 3,  
    activation=tf.nn.sigmoid)
```

# Keras

```
model = Sequential()  
  
model.add(Dense(3, input_dim=2))  
model.add(Activation('sigmoid'))  
  
model.add(Dense(1))  
model.add(Activation('sigmoid'))
```

# TensorFlow

```
x = tf.placeholder(  
    shape=[None, 2],  
    dtype=tf.float32)  
  
l1 = tf.layers.dense(x, 3,  
    activation=tf.nn.sigmoid)  
  
l2 = tf.layers.dense(l1, 1,  
    activation=tf.nn.sigmoid)
```

# Keras

```
model = Sequential()  
  
model.add(Dense(3, input_dim=2))  
model.add(Activation('sigmoid'))  
  
model.add(Dense(1))  
model.add(Activation('sigmoid'))
```

# TensorFlow

```
x = tf.placeholder(  
    shape=[None, 2],  
    dtype=tf.float32)  
  
l1 = tf.layers.dense(x, 3,  
    activation=tf.nn.sigmoid)  
  
l2 = tf.layers.dense(l1, 1,  
    activation=tf.nn.sigmoid)
```

# Keras

```
model = Sequential()  
  
model.add(Dense(3, input_dim=2))  
model.add(Activation('sigmoid'))  
  
model.add(Dense(1))  
model.add(Activation('sigmoid'))
```

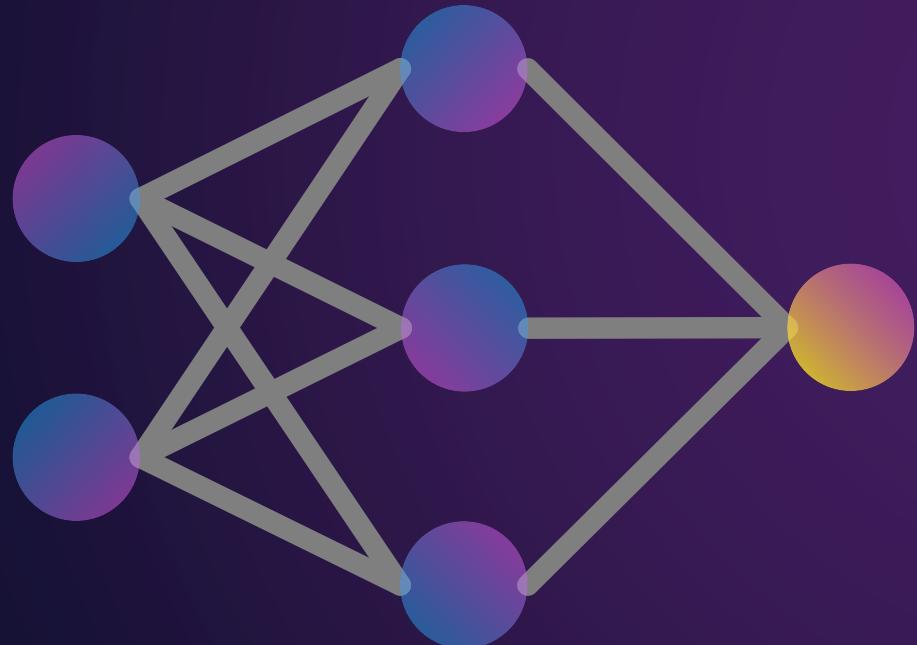
# TensorFlow

```
x = tf.placeholder(  
    shape=[None, 2],  
    dtype=tf.float32)  
  
l1 = tf.layers.dense(x, 3,  
    activation=tf.nn.sigmoid)  
  
l2 = tf.layers.dense(l1, 1,  
    activation=tf.nn.sigmoid)
```

# Keras

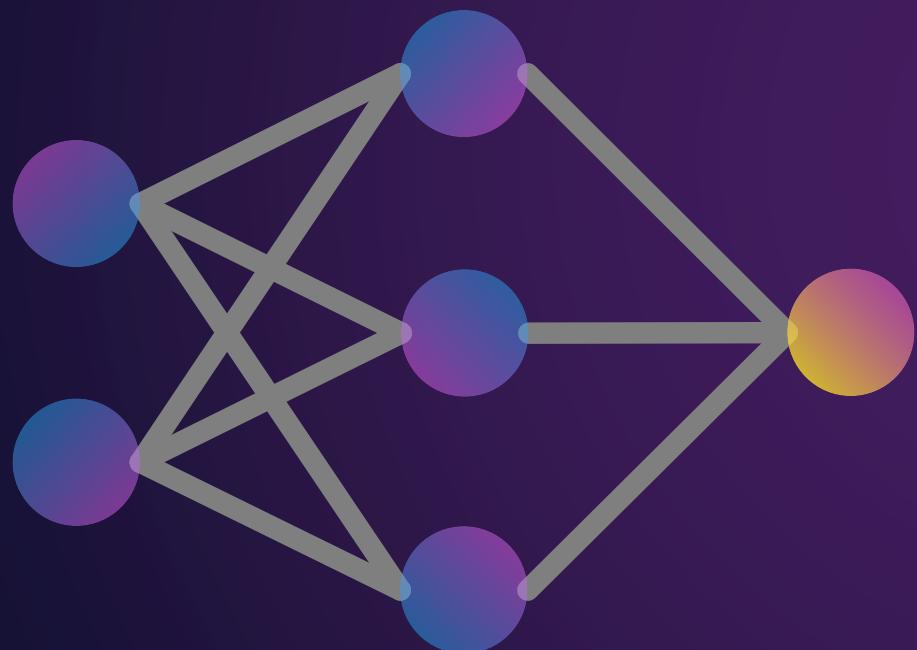
# TensorFlow

# Modell



# Keras

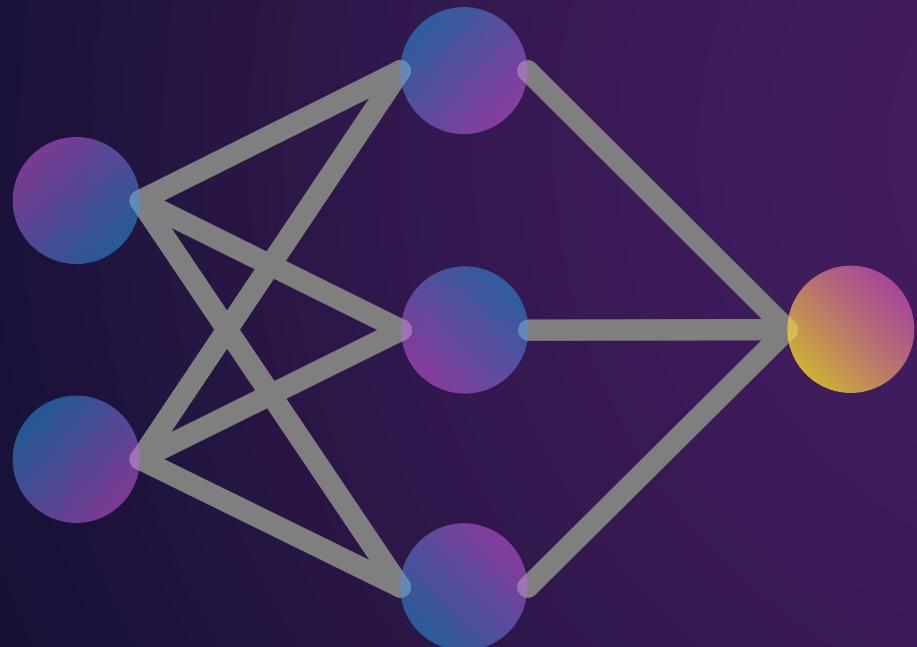
# Modell



# Keras

```
model.compile(loss='mse',  
              optimizer=adam(lr=0.001))
```

# Modell

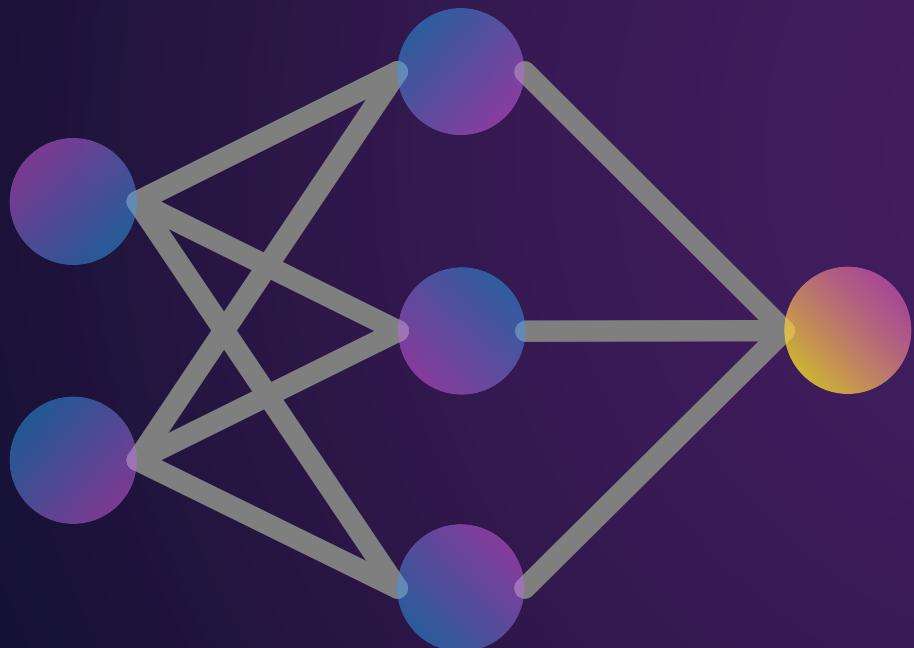


# Keras

```
model.compile(loss='mse',  
optimizer=adam(lr=0.001))
```

```
model.fit(x=x_train, y=y_train,  
batch_size=128,  
epochs=10)
```

# Modell



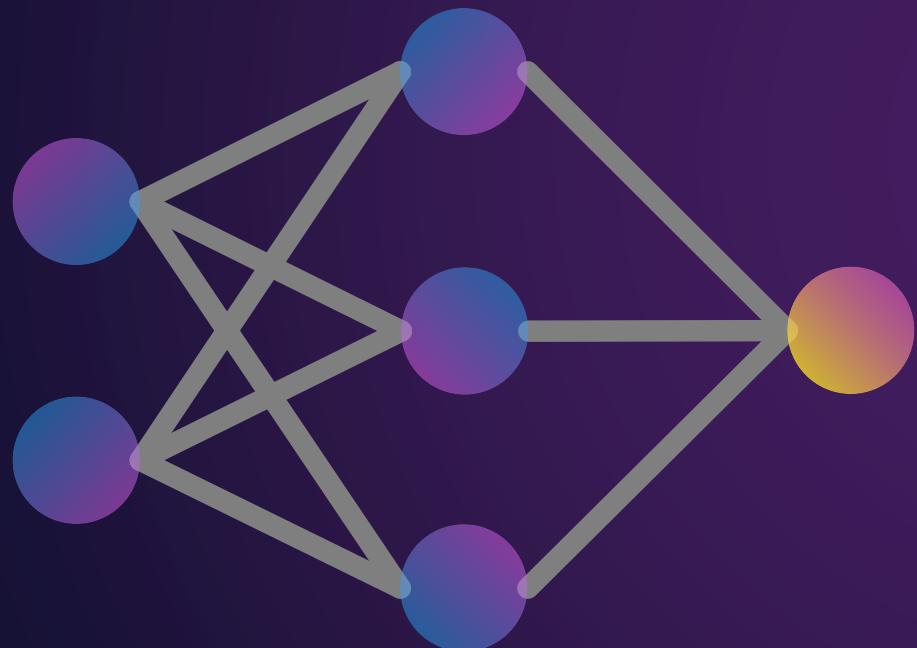
# Keras

```
model.compile(loss='mse',  
optimizer=adam(lr=0.001))
```

```
model.fit(x=x_train, y=y_train,  
batch_size=128,  
epochs=10)
```

```
model.evaluate(x=x_test, y=y_test)
```

# Modell



# Keras

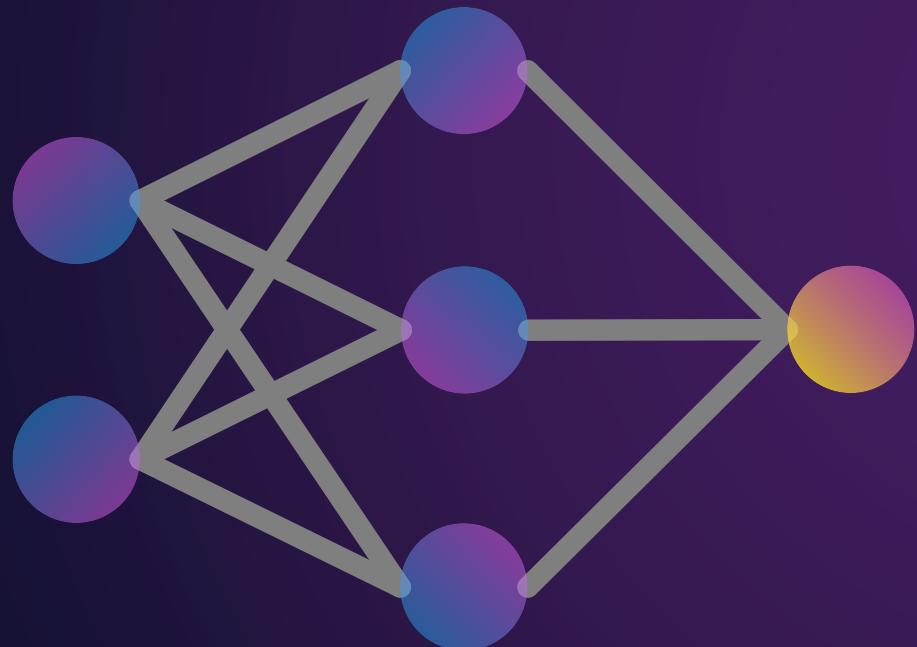
```
model.compile(loss='mse',  
optimizer=adam(lr=0.001))
```

```
model.fit(x=x_train, y=y_train,  
batch_size=128,  
epochs=10)
```

```
model.evaluate(x=x_test, y=y_test)
```

```
model.predict(x=x_new)
```

# Modell



# Keras

```
model.compile(loss='mse',  
optimizer=adam(lr=0.001))
```

```
model.fit(x=x_train, y=y_train,  
batch_size=128,  
epochs=10)
```



Problem

Handgeschriebene  
Zahlen  
klassifizieren

0 1 2 3 4  
5 6 7 8 9

0

1

2

3

4

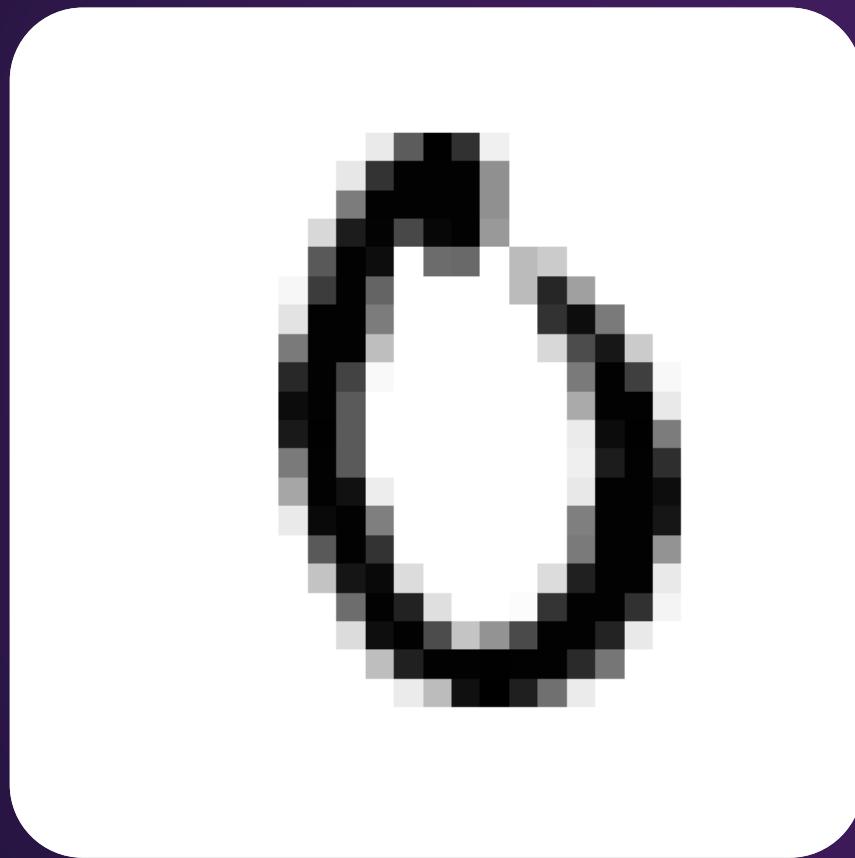
5

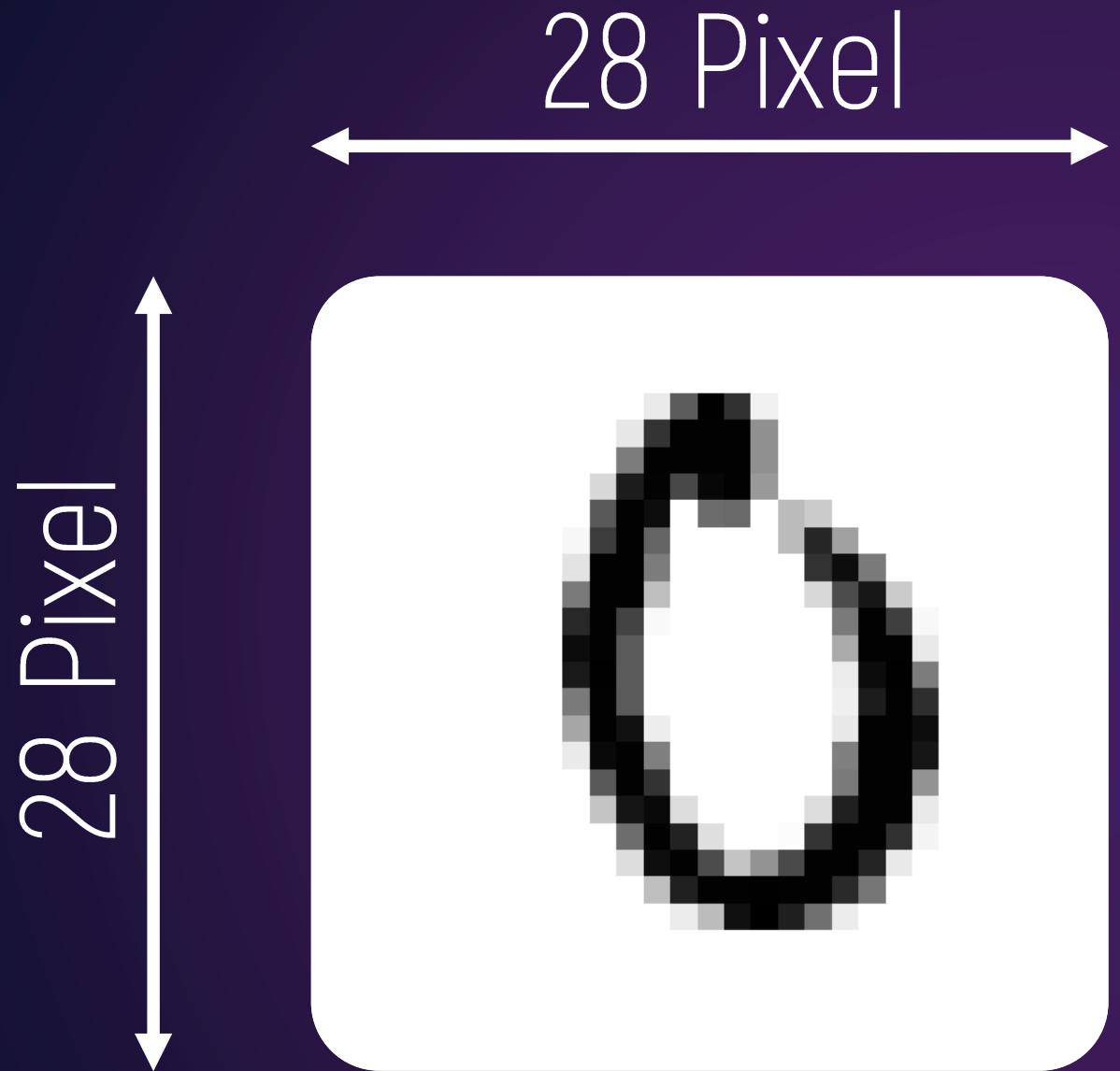
6

7

8

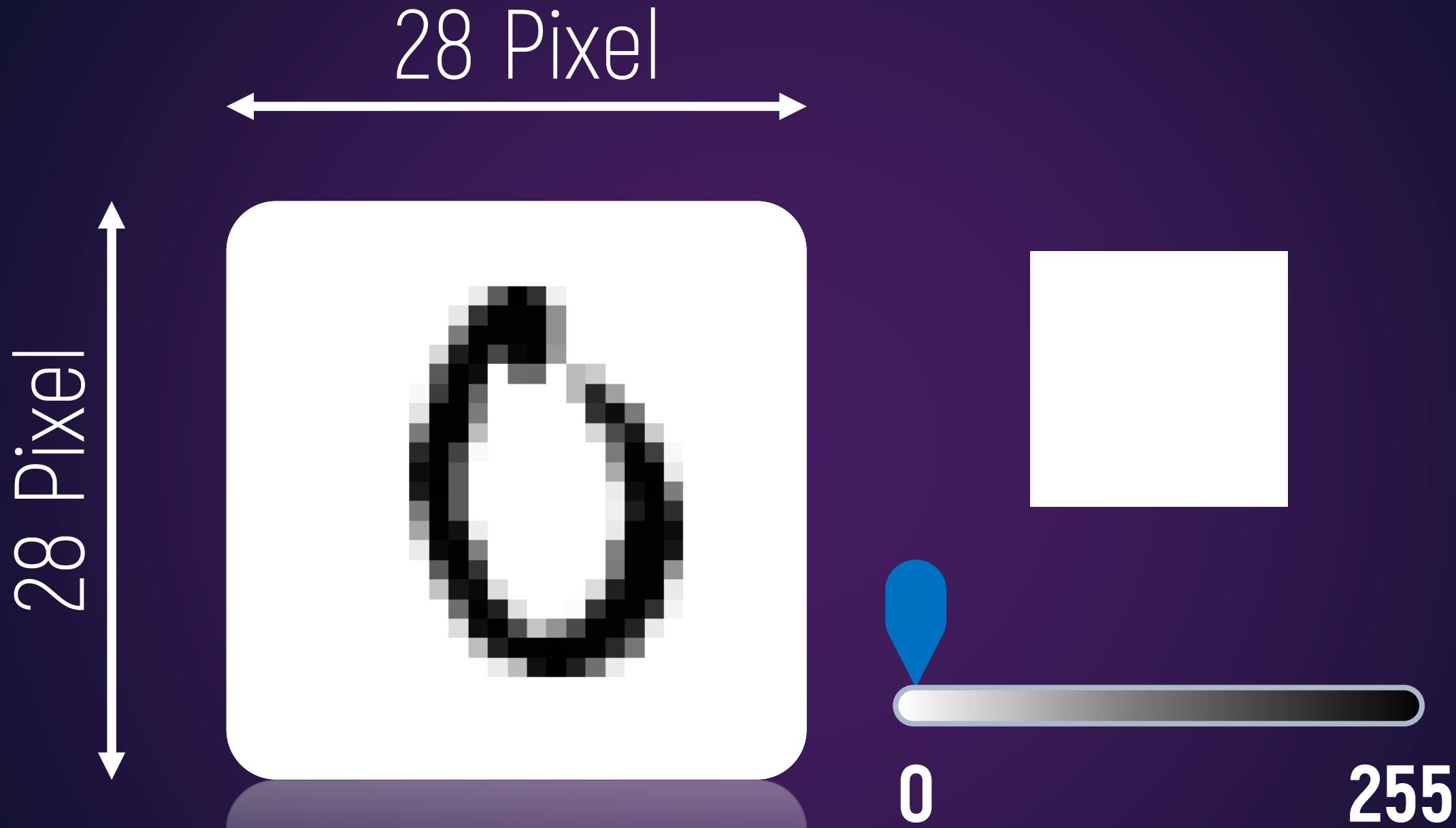
9

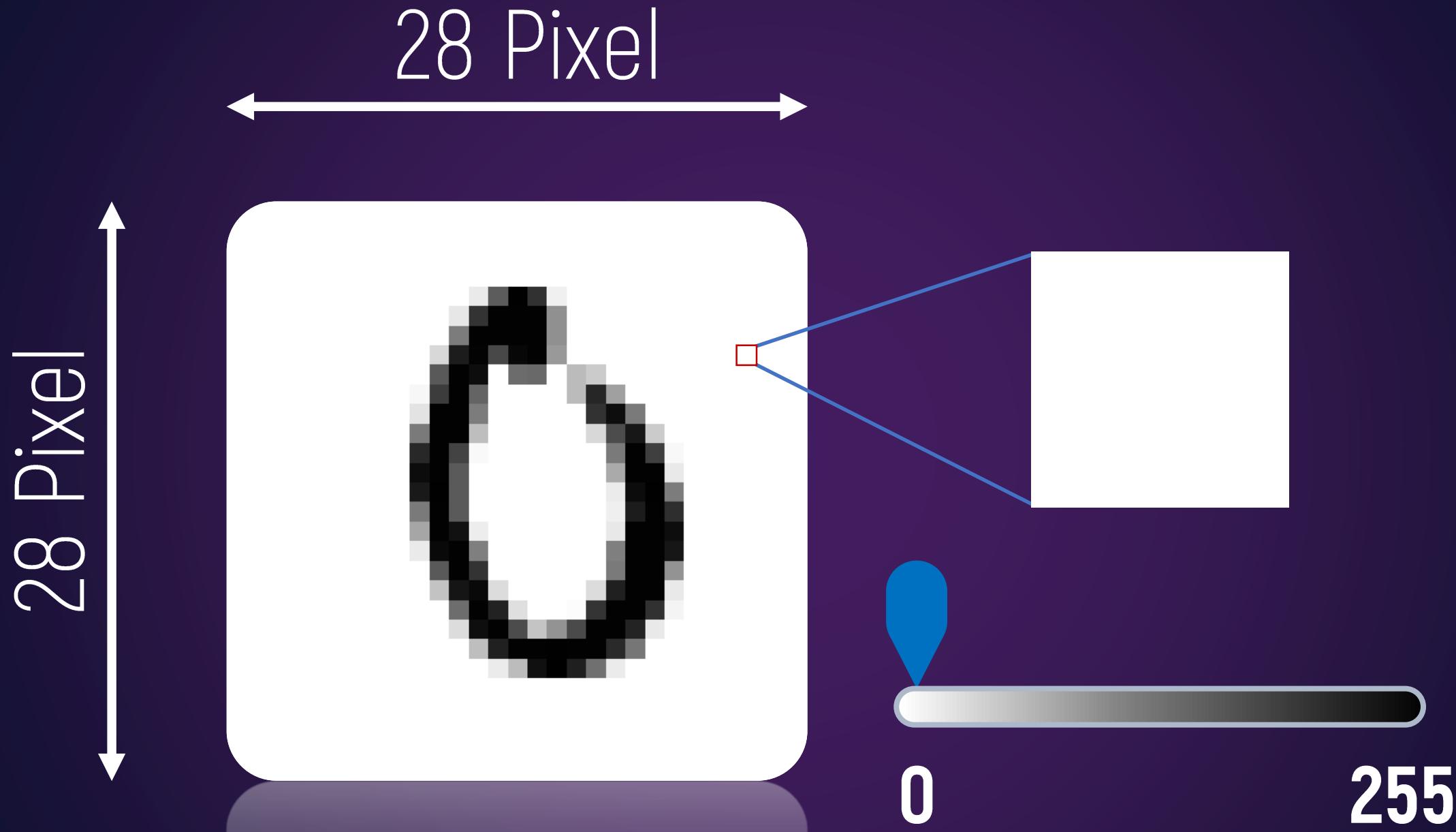


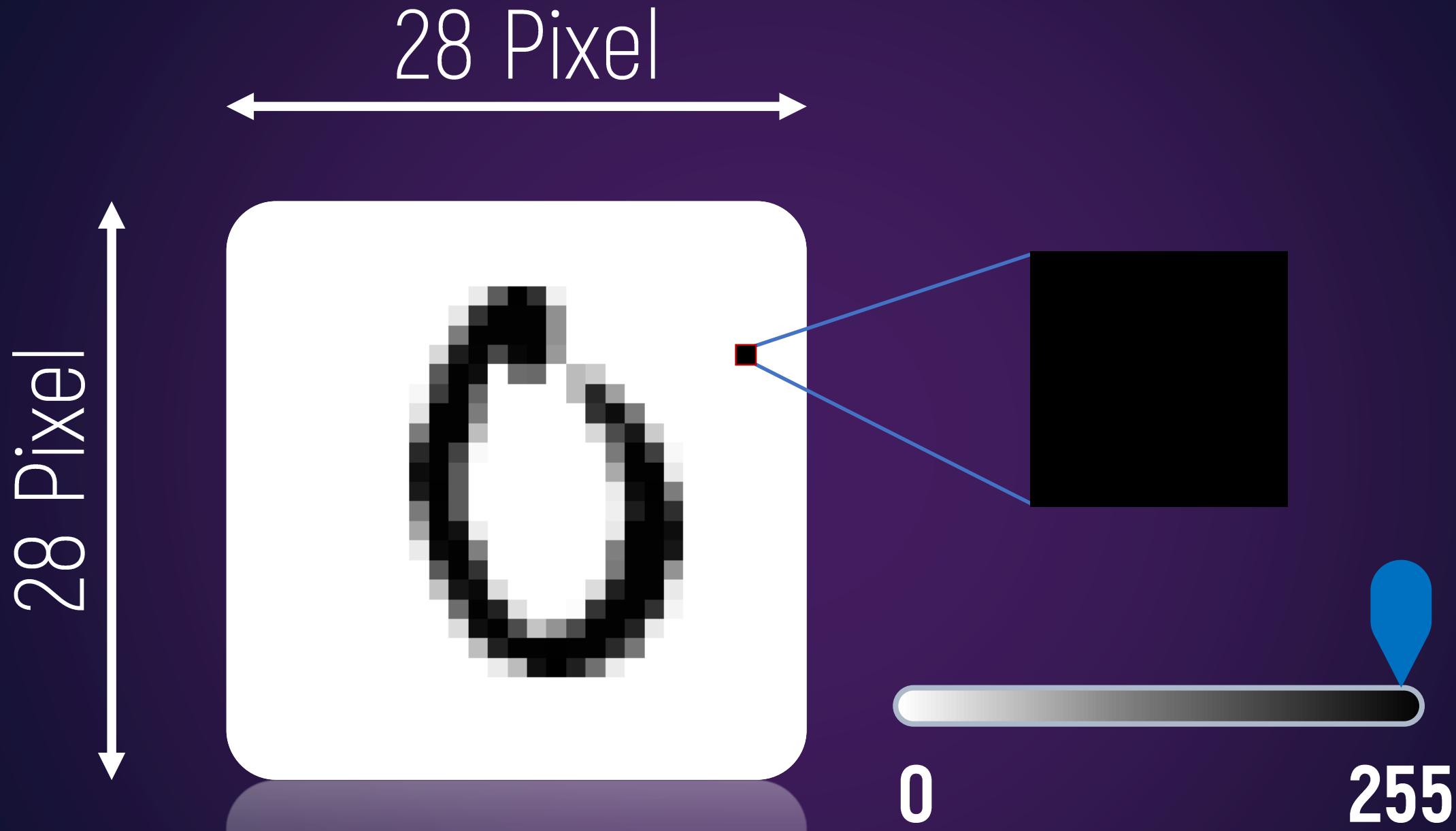


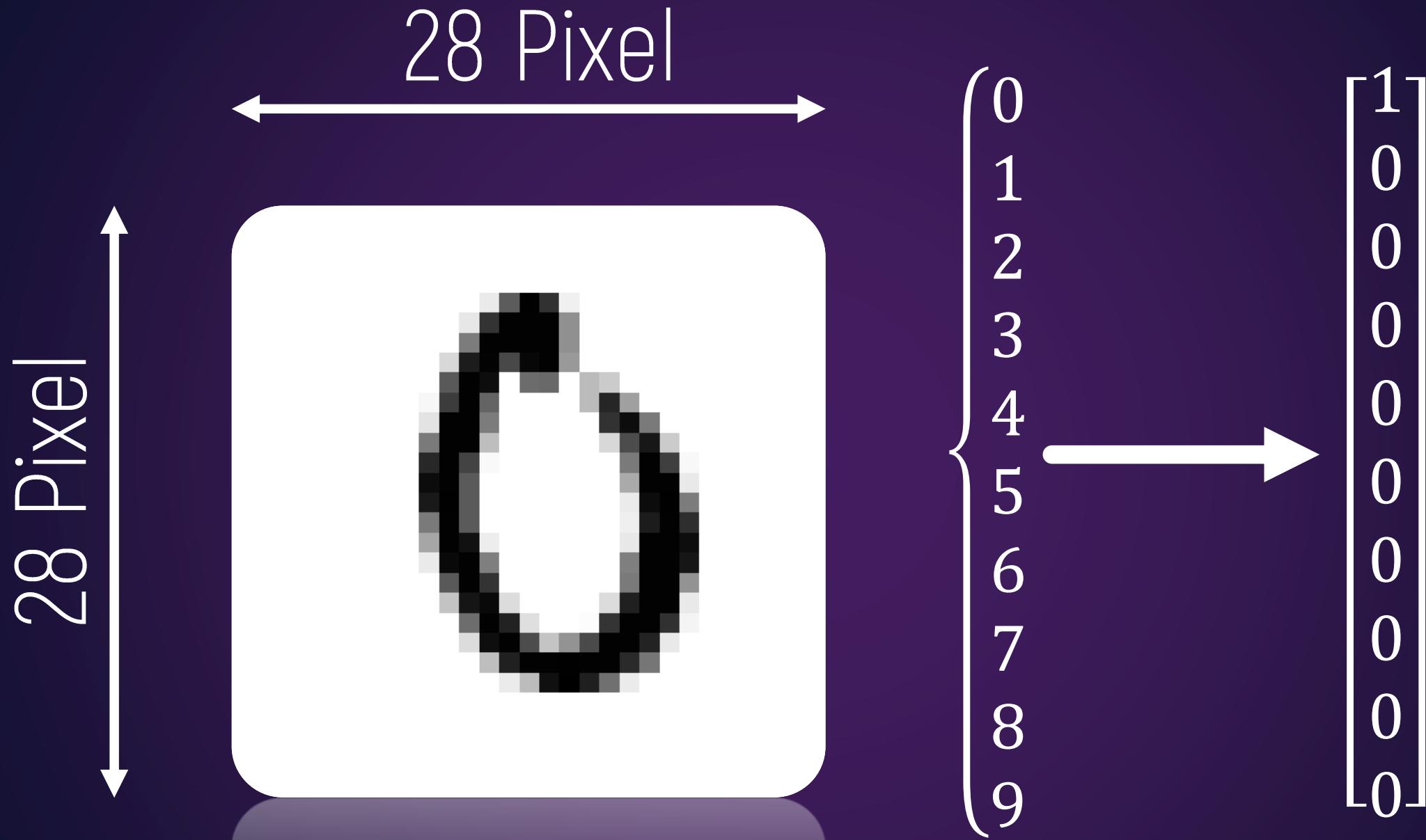
28 Pixel

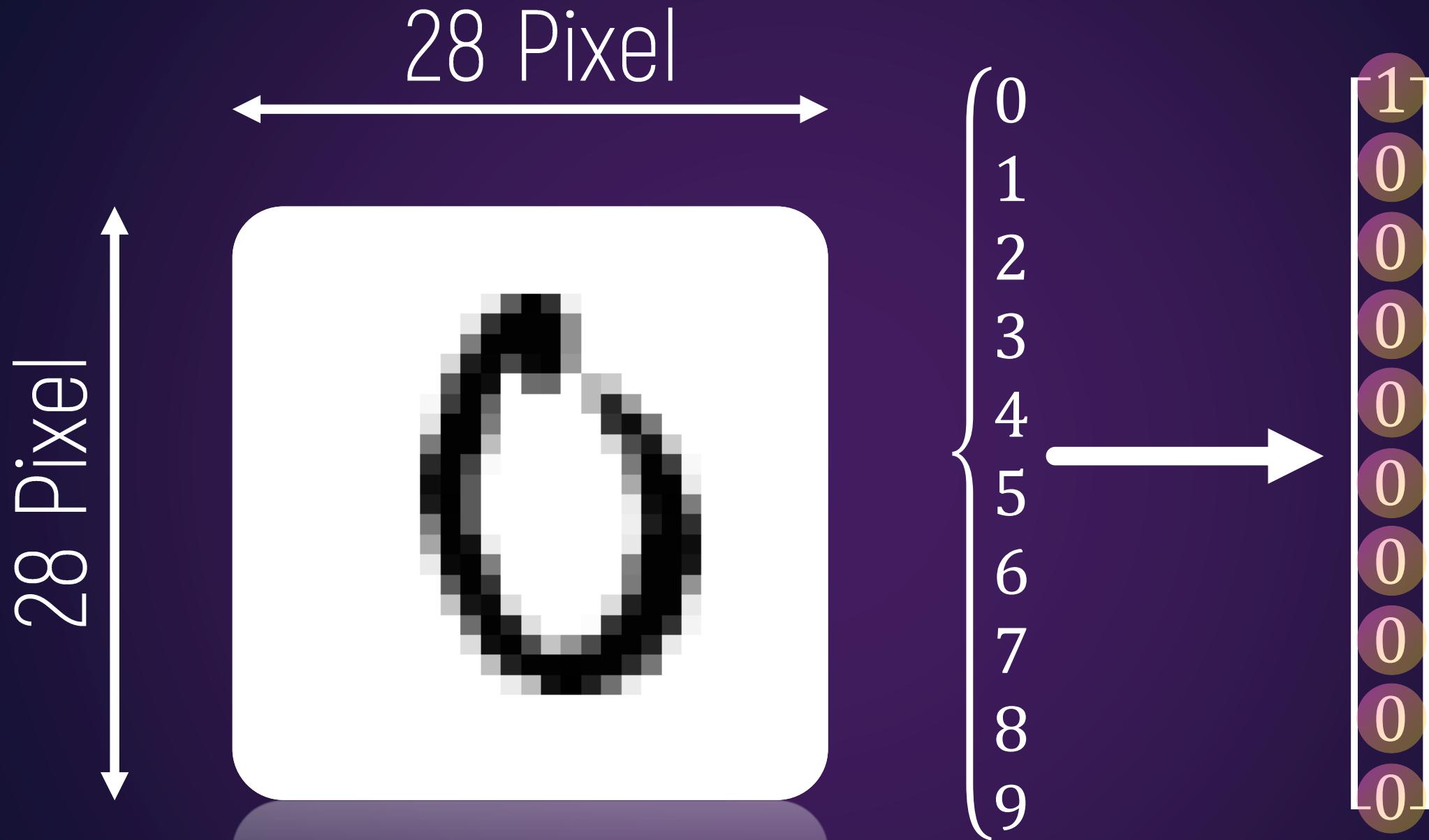
28 Pixel

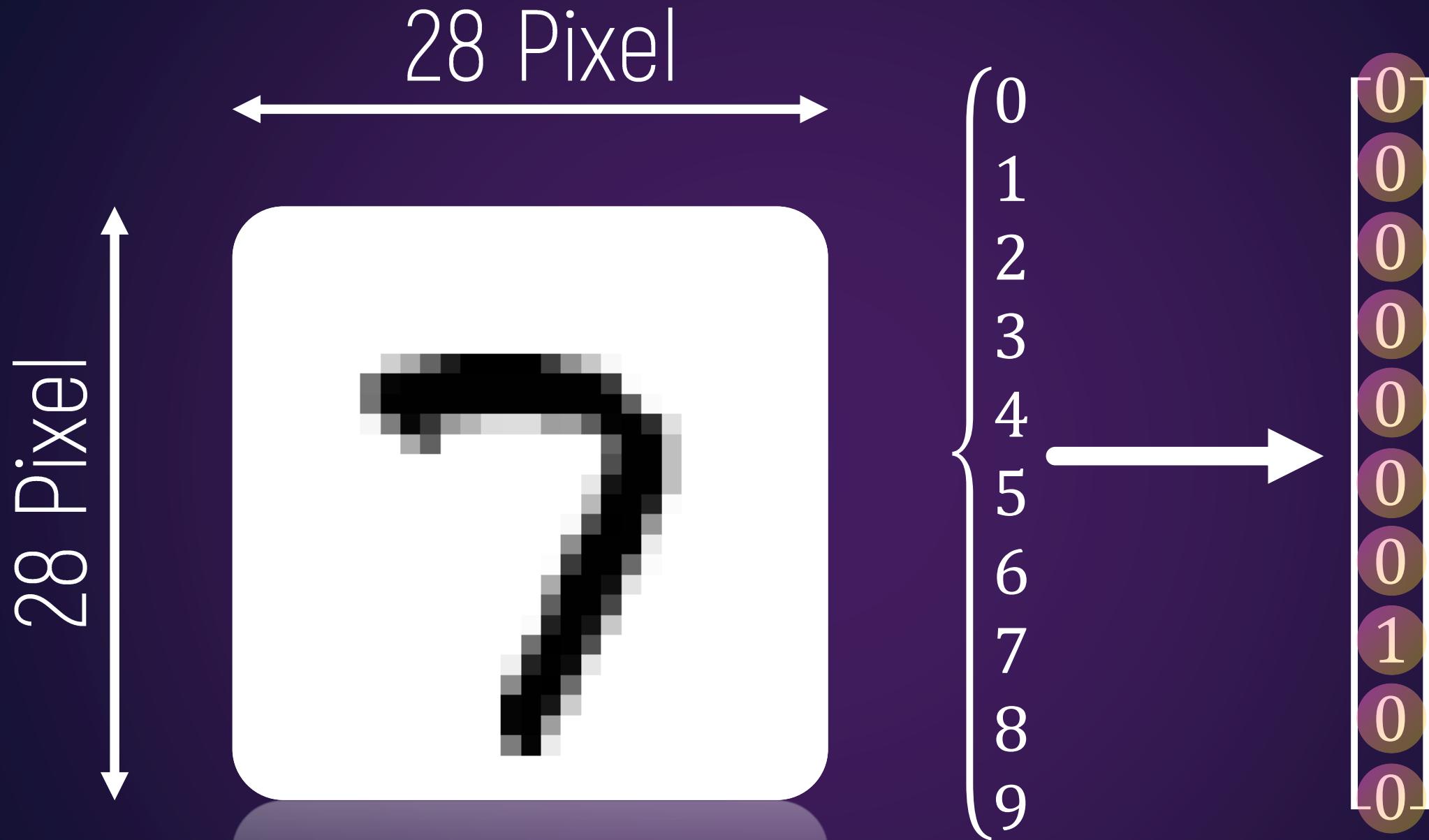


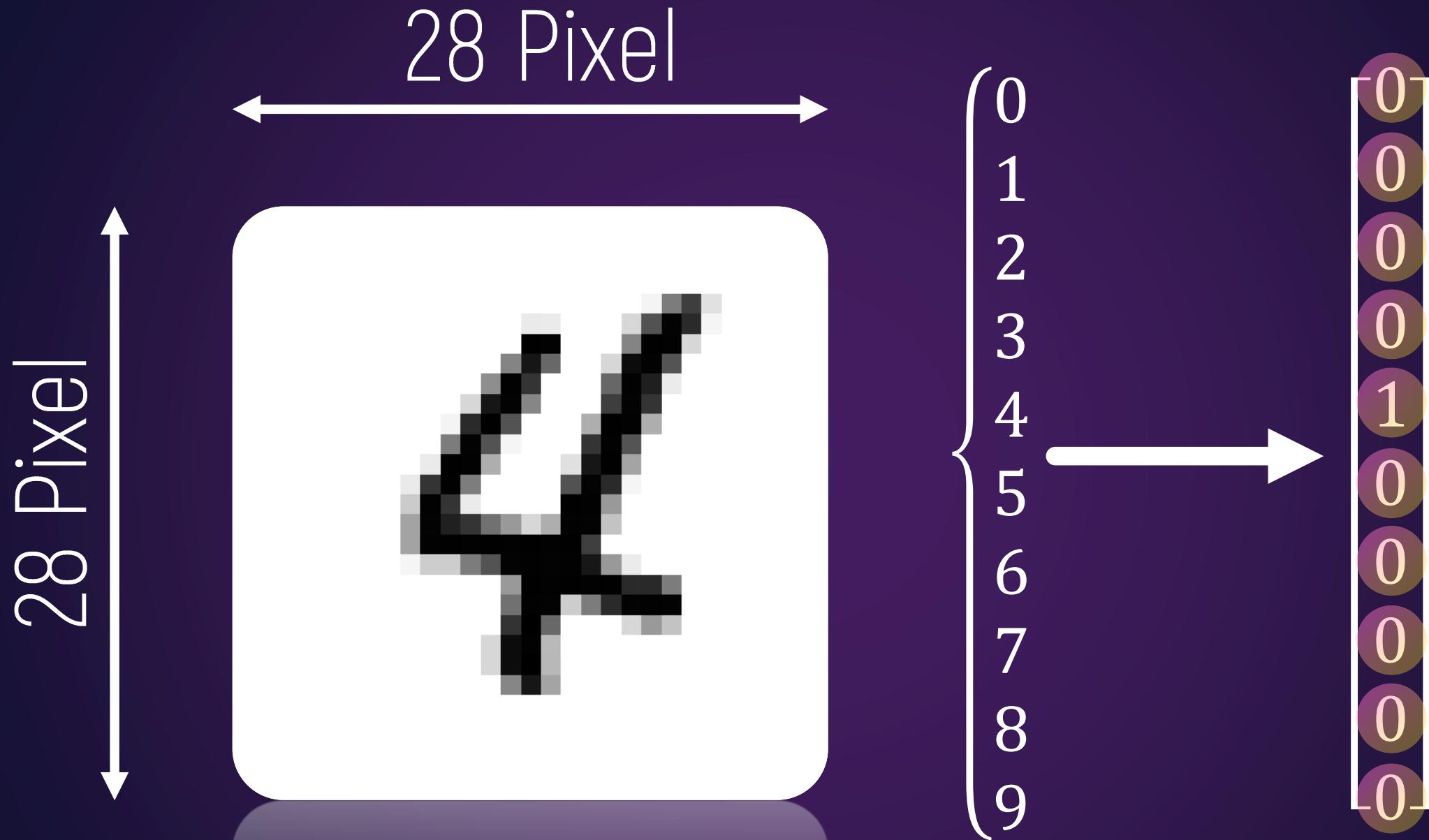














Live Demo

# Deep Learning

- Was ist Deep Learning
- Warum brauchen wir Deep Learning
- Typische Deep Learning Probleme
- Wie funktioniert Deep Learning



Summary

# Neuronales Netze

- Geschichte
- Aufbau
  - Schichten
  - Neuronen



Summary

# Lernalgorithmus

- Supervised Learning
- Gradient Descent
- Hyperparameter



Summary

# Keras und TensorFlow

- TensorFlow - Namensgebung
- Keras baut auf TensorFlow auf
- Vor- und Nachteile von Keras und TensorFlow



Summary

# Programmieren einer KI in Keras

- Quellcode
- Keras und TensorFlow Codevergleich
- Neuronales Netz in Keras trainieren



Summary

# Inhalt

- Resourcen für Selbststudium
- Das eben gezeigt Jupyter Notebook
- Diese Präsentation im PDF-Format



# Fragen





