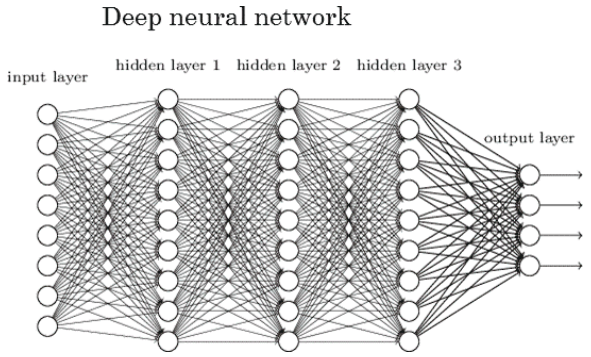
# TensorFlow - Erstellen eines Neuronalen Netzes

Jan Popko

Python Advanced

Business Trends Academy (BTA) GmbH
Nestorstraße 36
D-10709 Berlin
Tel.: +49 (0) 30 894 087 57
Fax: +49 (0) 30 895 429 94
Geschäftsführer:
Gabriele Fleischmann-Hahn
Maxi-Marlen Fleischmann
Hauptsitz des Unternehmens:
Nestorstraße 36, D-10709 Berlin
HRB 115251 B / Amtsgericht Berlin Charlottenburg
Steuer-Nr. 27/248/31179

# Imports und Wahl des Models




Imports:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Conv2D, MaxPooling2D,
                                    BatchNormalization
```

Erstellen eines Models:

```
model = Sequential()
```

# Wahl der Layer

Flatten:

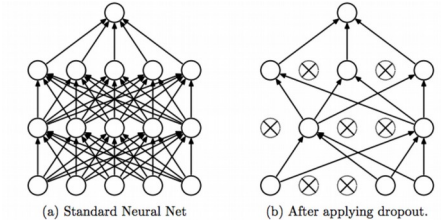
- flacht den Input von shape = (x, y) auf shape = (1, x*y) ab

Dropout:

- bekommt eine Zahl zwischen 0 und 1
- deaktiviert Input für nächstes Layer um diese Menge (zufällig)
- beugt overfitting vor, verlangsamt das Konvergieren

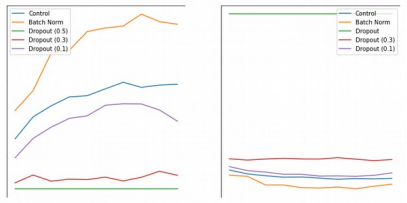
BatchNormalization:

- bei Convolutional (Faltenden) Netzen nutzen statt Dropout da räumliche zusammenhänge existieren können



(a) Standard Neural Net (b) After applying dropout.

accuracy loss



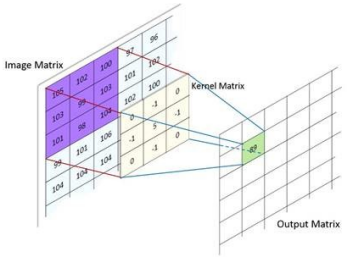
# Wahl der Layer

Dense:

- jeder Knoten ist mit jedem Knoten aus den anderen Layern verbunden
- erwartet flachen Input
- erstellt eine Matrix von Gewichten und wendet diese auf Input an

Conv2D:

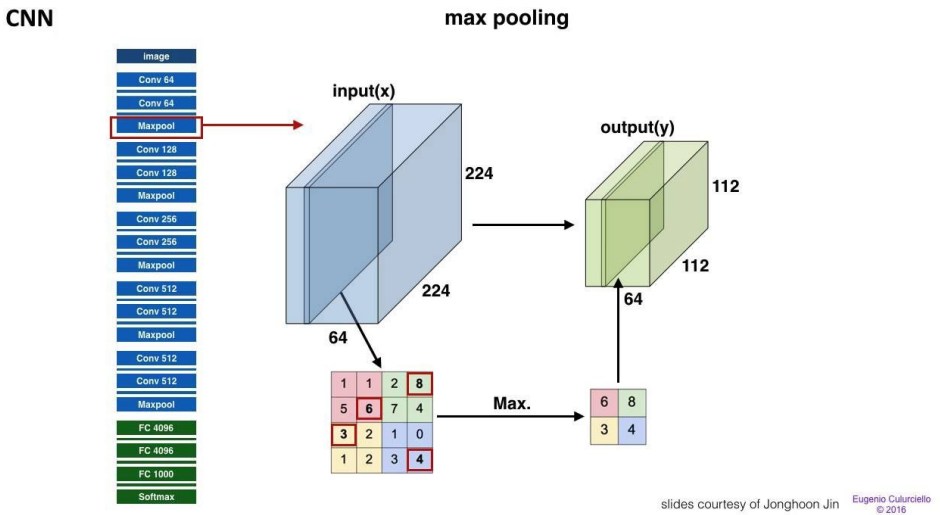
- Convolutional Layer
- Faltet das Bild
- wendet eine (x, y) Matrix auf ein Bild an (Gewichtung)
- verringert Daten



# Wahl der Layer

MaxPooling2D:

- wählt aus einem (x, y) Grid das Maximum
- folgt gewöhnlich einem Convolutional Layer



# Spezielle Layer

## Input Layer:

- Beliebiges Layer
- Bekommt alle Werte x aus dem Datensatz X und die Targets y
- input_shape von x sollte spezifiziert werden

## Output Layer:

- für Klassifizierung:
  - 2 Klassen:
    - Dense
    - 1 Knoten
    - activation_function = sigmoid
  - mehr als 2 Klassen:
    - Dense
    - # Knoten = # Klassen
    - activation_function = softmax
- für Berechnungen:
  - Dense
  - 1 Knoten
  - activation_function = linear (== default Funktion)

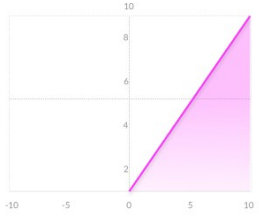
# Aktivierungsfunktionen

Softmax:

- transformiert den Output, so, daß alle Werte im Bereich zwischen 0 -1 liegen, wobei die Summe aller Werte gleich 1 ist
- Resultat kann als Wahrscheinlichkeitsverteilung interpretiert werden
- wird häufig als Output bei Klassifizierungsnetzwerken genutzt

Relu:

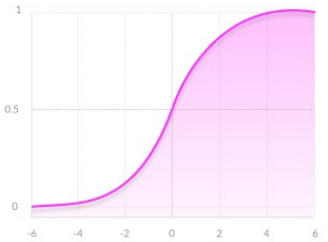
- gibt das maximum von (x, 0) aus
- wird gängigerweise bei Convolutional Layern genutzt
- relativ geringer Rechenaufwand



# Aktivierungsfunktionen

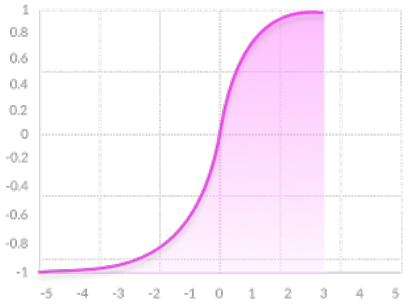
Sigmoid:

- höherer Rechenaufwand
- für sehr große und sehr kleine Werte ändern sich die Gewichtungen kaum
- meist genutzt zur Klassifizierung




Tanh:

- Hyperbolic Tangens, ähnlich Sigmoid
- konvergiert häufig schneller als Sigmoid



# Compilieren

Nach der Wahl der Layer muß das Model Kompiliert werden.
Man kann bestimmte Parameter übergeben:

Optimizer:

- Bestimmt Optimisierungswerte des Models (Lernrate)
- Man kann nicht generalisieren, wann ein Optimizer besser als der andere ist
- Aber häufig gilt:
    - Adam- erzielt gute für alle Modelle
    - AdaGrad, RMSProp – sind meistens schneller, Speicherintensiver
    - SGD – langsamere Konvergierung, weniger Speicherverbrauch, häufig bessere Accuracy und Loss Werte



Business Trends Academy (BTA) GmbH
Nestorstraße 36
D-10709 Berlin
Tel.: +49 (0) 30 894 087 57
Fax: +49 (0) 30 895 429 94
Geschäftsführer:
Gabriele Fleischmann-Hahn
Maxi-Marlen Fleischmann
Hauptsitz des Unternehmens:
Nestorstraße 36, D-10709 Berlin
HRB 115251 B / Amtsgericht Berlin Charlottenburg
Steuer-Nr. 27/248/31179

# Compilieren

Loss:
- Bestimmt, nach welcher Berechnung der Loss berechnet wird

BinaryCrossentropy
- name = 'binary_crossentropy'
- wenn zwischen zwei Labeln entschieden werden soll

SparseCategoricalCrossentropy
- name='sparse_categorical_crossentropy'
- wenn zwischen mehr als zwei Labeln entschieden werden soll
- Label sollten als Integer übergebenwerden

MeanSquaredError
- name='mean_squared_error'
- Es wird der mittlere quadrierte Fehler zwischen y_true und y_pred berechnet



Business Trends Academy (BTA) GmbH
Nestorstraße 36
D-10709 Berlin
Tel.: +49 (0) 30 894 087 57
Fax: +49 (0) 30 895 429 94
Geschäftsführer:
Gabriele Fleischmann-Hahn
Maxi-Marlen Fleischmann
Hauptsitz des Unternehmens:
Nestorstraße 36, D-10709 Berlin
HRB 115251 B / Amtsgericht Berlin Charlottenburg
Steuer-Nr. 27/248/31179

# MNIST Model

```
model = Sequential()

model.add(Flatten(input_shape=(28,28))
model.add(Dense(128, activation = 'relu')
model.add(Dropout(0.2))
model.add(Dense(128, activation = 'relu')
model.add(Dense(10, activation = 'softmax')
```

**Oder:**

```
model = Sequential([
     Flatten(input_shape=(28,28)),
     Dense(128, activation = 'relu'),
     Dropout(0.2),
     Dense(128, activation = 'relu'),
     Dense(10, activation = 'softmax'
])

model.compile(optimizer='Adam',
              loss=['sparse_categorical_crossentropy'],
              metrics=['accuracy'])
```

# Speichern/Laden von Modellen

Speichern:

```
model.save('pfad/model_name.h5')
```

Laden:

```
from tensorflow.keras.models import load_model

load_model('pfad/model_name.h5')
```