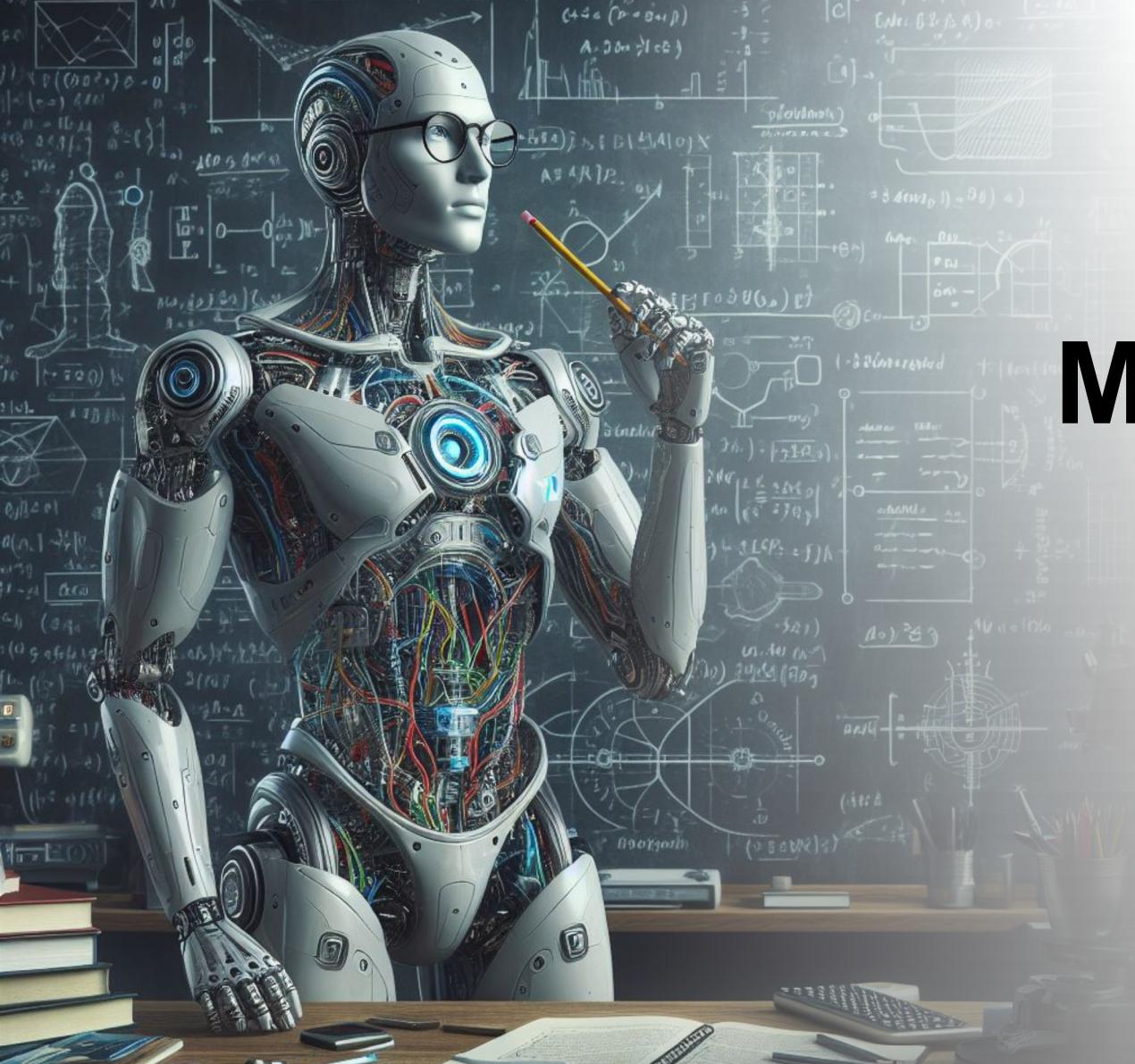


# Maschinelles Lernen

Untertitel



# Was ist Machine Learning?

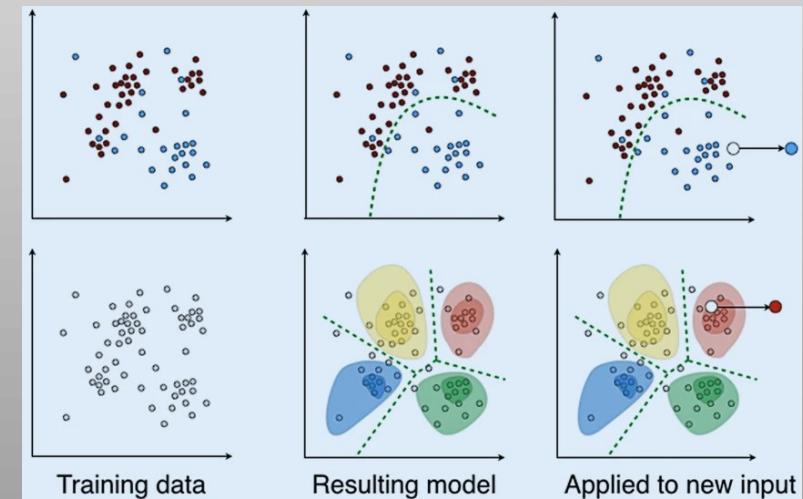
- Teilgebiet der künstlichen Intelligenz
- System erkennt Muster und Zusammenhänge
- keine explizite Programmierung
- Automatische Verbesserung über Zeit
- Vorhersagen von Werten, Berechnen von Wahrscheinlichkeiten, Dimensionen reduzieren, Geschäftsprozesse optimieren

# Wie funktioniert ML?

- nimmt Daten auf
  - Verarbeitung der Daten, Zusammenhänge erkennen
  - Ausgabe eines Wertes
- 
- Trainieren eines Modells mit Trainingsdatensatz (richtige Ausgabe bekannt)
  - Validierungsdatensatz zur Überprüfung des Modells
- 
- keine klassische Programmierung

# Arten des Machine Learnings

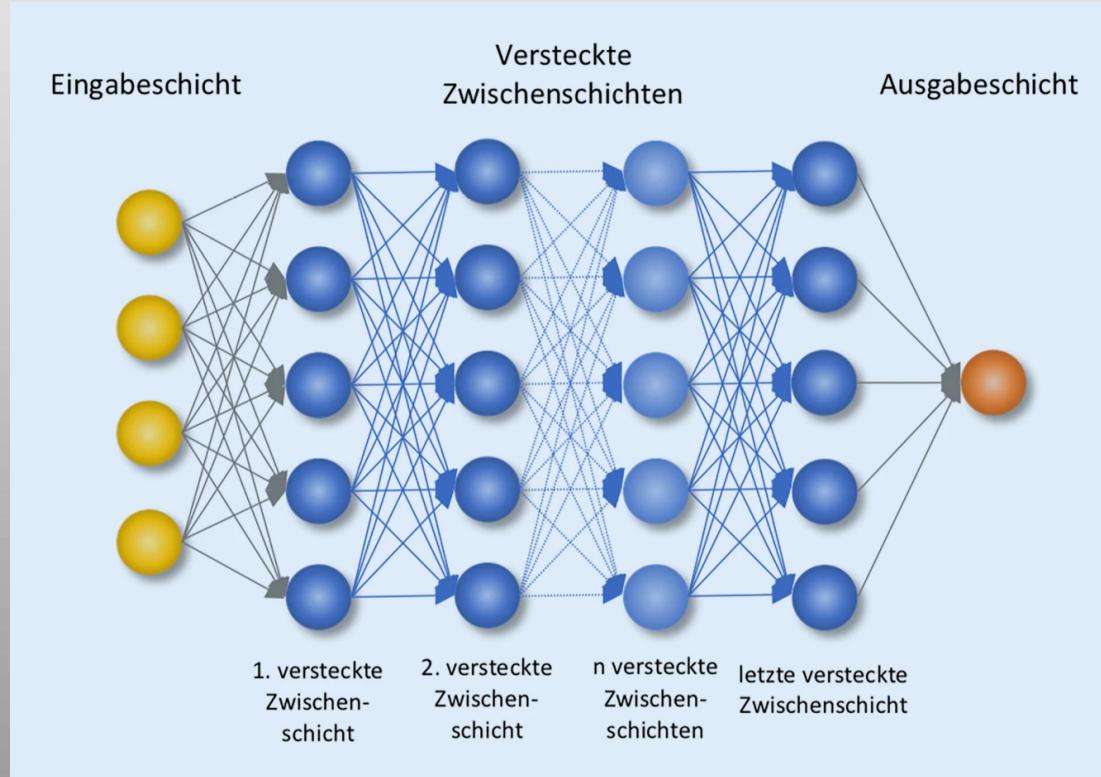
- Überwachtes Lernen
  - bekannte Daten, lernt Zusammenhang zwischen Eingabe und Ausgabe
  - Klassifizierung und Vorhersage von Daten
- Unüberwachtes Lernen
  - eigenständiges Erkennen von Mustern
  - Clusteranalyse und Dimensionsreduktion



- Teilüberwachtes Lernen
  - Mischung aus überwachtem und unüberwachtem Lernen
  - geringe Menge an bekannten Daten, viele unbekannte
  - günstiger, da bekannte Daten manuell zugewiesen werden müssen
- Verstärkendes Lernen
  - Belohnung bei guten Aktionen, Bestrafung bei schlechten Aktionen
  - Algorithmus lernt, indem möglichst hohe Belohnung erzielt wird

# Neuronale Netze

- besteht aus Neuronen in Schichten



# Erstellen des Programms

- 1) Datensatz auswählen
- 2) Neuronales Netz erstellen
- 3) Modell trainieren
- 4) Modell speichern und anwenden

# Auswählen des Datensatzes

- Genügend Daten
- Gelabelt
- Verwendet: Grocery Store Dataset

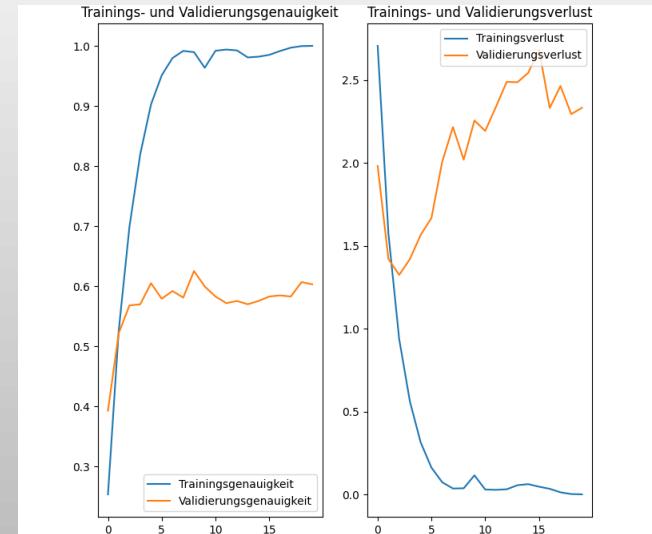


# Neuronales Netz erstellen

```
model = Sequential([
    layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
    layers.Conv2D(filters: 16, kernel_size: 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(filters: 32, kernel_size: 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(filters: 64, kernel_size: 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(units: 128, activation='relu'),
    layers.Dense(num_classes)
])
```

# Modell trainieren

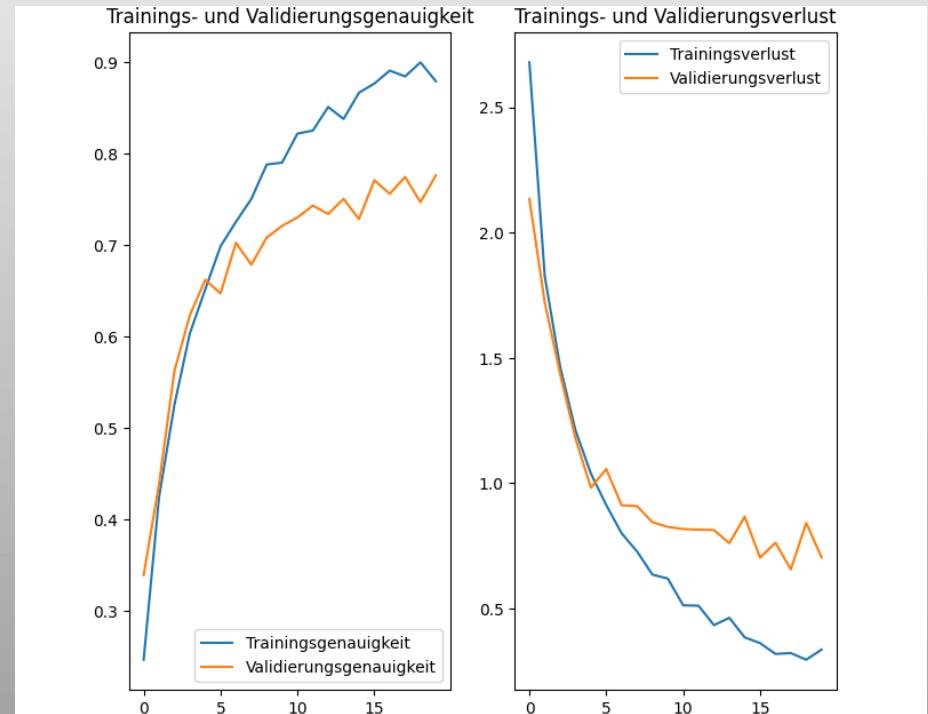
```
epochs = 15  
history = model.fit(  
    train_ds,  
    validation_data=val_ds,  
    epochs=epochs  
)
```



```
129/129 [=====] - 36s 264ms/step - loss: 3.0396 - accuracy: 0.1949 - val_loss: 5.8812 - val_accuracy: 0.0678  
Epoch 2/15  
129/129 [=====] - 33s 252ms/step - loss: 2.3325 - accuracy: 0.3266 - val_loss: 7.0514 - val_accuracy: 0.0339  
Epoch 3/15  
129/129 [=====] - 33s 255ms/step - loss: 1.8756 - accuracy: 0.4307 - val_loss: 8.7323 - val_accuracy: 0.0678  
Epoch 4/15  
129/129 [=====] - 32s 252ms/step - loss: 1.5274 - accuracy: 0.5202 - val_loss: 10.0023 - val_accuracy: 0.1017  
Epoch 5/15  
129/129 [=====] - 32s 244ms/step - loss: 1.3019 - accuracy: 0.5768 - val_loss: 10.4406 - val_accuracy: 0.1017  
Epoch 6/15  
129/129 [=====] - 31s 243ms/step - loss: 1.1211 - accuracy: 0.6259 - val_loss: 12.1131 - val_accuracy: 0.1017  
Epoch 7/15  
62/129 [=====>.....] - ETA: 17s - loss: 0.9446 - accuracy: 0.6825
```

# Overfitting

```
data_augmentation = keras.Sequential(  
[  
    layers.RandomFlip(mode="horizontal",  
                      input_shape=(img_height, img_width, 3)),  
    layers.RandomRotation(0.1),  
    layers.RandomZoom(0.1),  
]  
)  
  
model = Sequential([  
    data_augmentation,  
    layers.Rescaling(1./255),  
    layers.Conv2D(filters: 16, kernel_size: 3, padding='same', activation='relu'),  
    layers.MaxPooling2D(),  
    layers.Conv2D(filters: 32, kernel_size: 3, padding='same', activation='relu'),  
    layers.MaxPooling2D(),  
    layers.Conv2D(filters: 64, kernel_size: 3, padding='same', activation='relu'),  
    layers.MaxPooling2D(),  
    layers.Dropout(0.2),  
    layers.Flatten(),  
    layers.Dense(units: 128, activation='relu'),  
    layers.Dense(num_classes, name="outputs")  
])
```



# Modell speichern und aufrufen

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

date = str(date.today())

with open(date+'.tflite', 'wb') as f:
    f.write(tflite_model)
```

```
interpreter = tf.lite.Interpreter(model_path=".//2024-02-25.tflite")

classify_lite = interpreter.get_signature_runner('serving_default')
```

# Werte voraussagen

```
print("Enter the URL of an image to classify: ", end="")
url = input()
filename = "".join(random.choices(string.ascii_uppercase + string.digits, k=10))
path = tf.keras.utils.get_file(filename, origin=url)

img = tf.keras.utils.load_img(
    path, target_size=(img_height, img_width)
)

img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions_lite = classify_lite(sequential_input=img_array)['outputs']
score_lite = tf.nn.softmax(predictions_lite)

print(
    "This image most likely belongs to {name} with a {percent} percent confidence."
    .format(name=class_names[np.argmax(score_lite)], percent=100 * np.max(score_lite))
)
```

# Beispiel



```
Enter the URL of an image to classify: https://imgix.obi.de/api/disc/cms/public/dam/DE-AT-As
Downloading data from https://imgix.obi.de/api/disc/cms/public/dam/DE-AT-Assets/pflanzen/apf
90603/90603 [=====] - 0s 0us/step
This image most likely belongs to Apple with a 99.77620840072632 percent confidence.
```

# Beispiel



shutterstock.com · 2210553929

```
Enter the URL of an image to classify: https://www.shutterstock.com/image-photo/close-pile-kiwi-grocery-store-46652/46652 [=====] - 0s 0us/step
This image most likely belongs to Kiwi with a 78.82998585700989 percent confidence.
```

# Beispiel



```
Enter the URL of an image to classify: https://upload.wikimedia.org/wikipedia/commons/thumb/e/ec/Oranges\_in\_a\_grocery\_store.jpg/166871px-Oranges\_in\_a\_grocery\_store.jpg
Downloading data from https://upload.wikimedia.org/wikipedia/commons/thumb/e/ec/Oranges\_in\_a\_grocery\_store.jpg/166871px-Oranges\_in\_a\_grocery\_store.jpg [=====] - 0s 1us/step
This image most likely belongs to Pepper with a 71.57952785491943 percent confidence.
```