

Der Code hinter KI

Dr. Alexander Rachmann

Fachbereich Technology

Professur für Anwendungsorientierte Informatik

E-Mail: a.rachmann@cbs.de

- Ausbildung zum Industriekaufmann in einer Krawattenfabrik
- Studium der Wirtschaftsinformatik, Informatik & Informationswissenschaft
- Promotion zur Entwicklung von IT-Dienstleistung zur Betreuung von alleinlebenden älteren Personen
- Beratertätigkeit in der Softwareentwicklung, oft im E-Commerce
- Verschiedene Positionen im Handel bei real.digital & Fressnapf
- Seit 2021 an der CBS / EUFH an der Professur für anwendungsorientierte Informatik

Agenda „Der Code hinter KI“

Was ist KI?

Codebeispiel „Weizenbaum“

Übung

Abschluss

Agenda „Der Code hinter KI“

Was ist KI?

Codebeispiel „Weizenbaum“

Übung

Abschluss

Für die Übung benötigt ihr gleich einen Google-Zugang (<https://accounts.google.com/>) und Leap.ai-Zugang (<https://www.tryleap.ai>). Legt ihn jetzt schon mal an, damit wir nachher gut in die Übung starten können.

Was ist KI?

Was ist KI?

Definition nach Zielsetzung

- Das Deutsche Forschungszentrum für künstliche Intelligenz schreibt: „Künstliche Intelligenz ist die Eigenschaft eines IT-Systems, »menschenähnliche«, intelligente Verhaltensweisen zu zeigen.“

Was ist KI?

Definition nach Funktionsbereichen

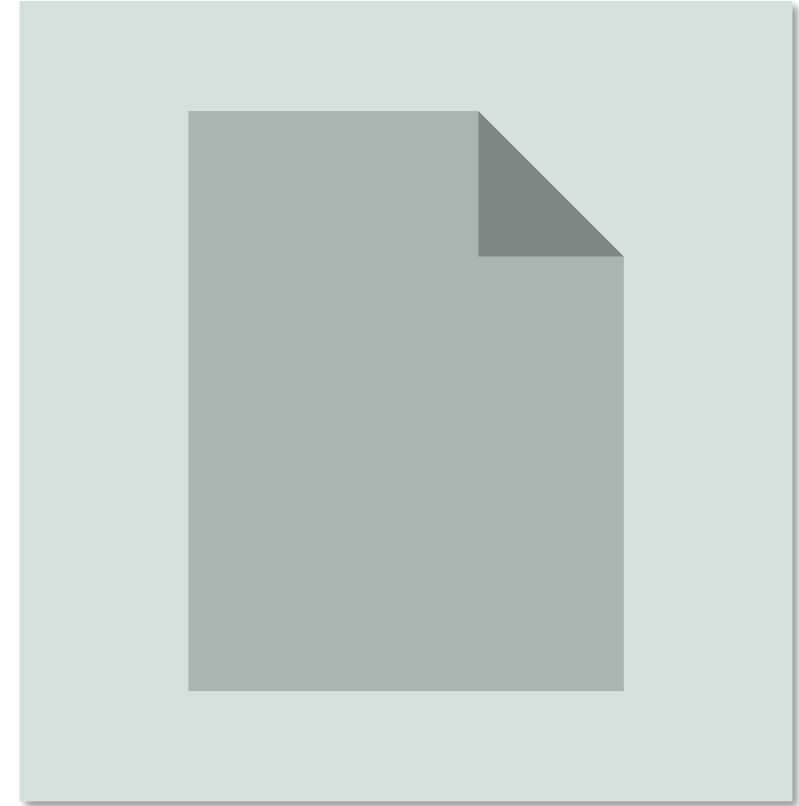
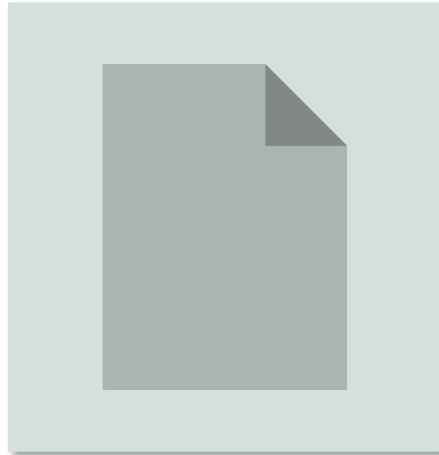
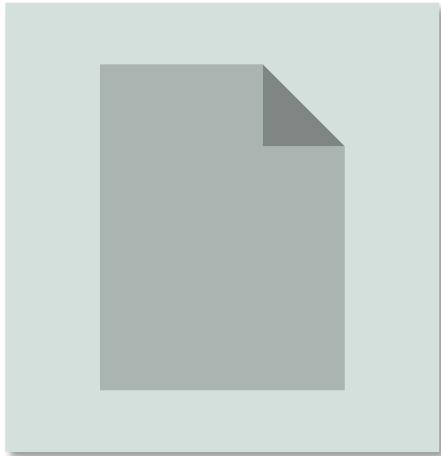
- **Verarbeitung natürlicher Sprache**, damit er einer natürlichen Sprache kommunizieren kann
- **Wissensrepräsentation**, damit er speichern kann, was er weiß und hört
- **Automatisches logisches Schließen**, um anhand der gespeicherten Informationen Fragen zu beantworten
- **Maschinenlernen**, um sich an neue Umstände anzupassen

- **Sensorik**, um Objekte wahrzunehmen
- **Aktorik**, um Objekte zu manipulieren

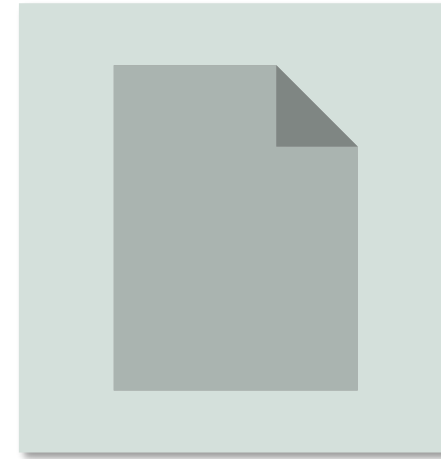
In Anlehnung an Russel, Norvig 2012

Datengetriebene KI

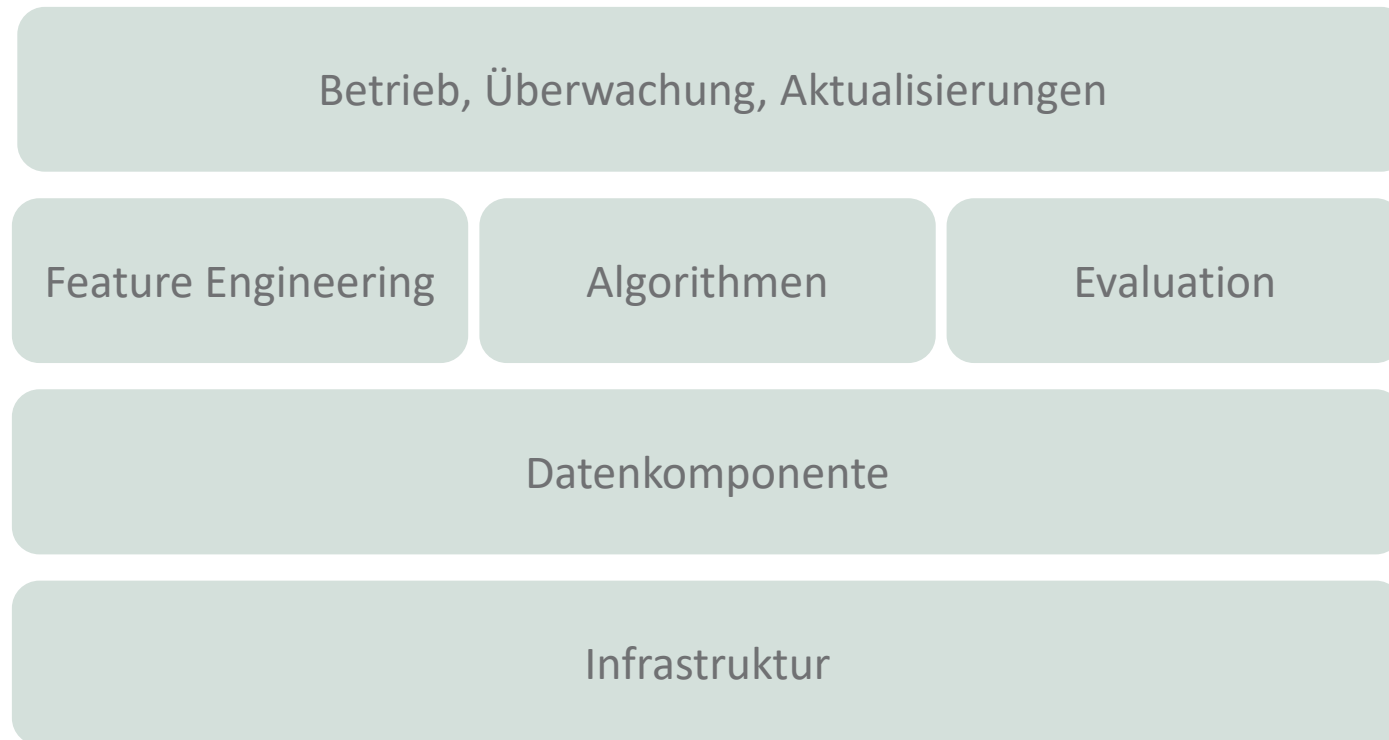
- Unter „datengetriebener KI“ verstehen wir solche Ansätze der KI, die aus sehr großen Datenmenge Informationen ziehen.



- Mit „generativer KI“ meinen wir solche (datengetriebenen) KI-Systeme, die aus den vorhandenen Daten „neue“ Informationen generieren
 - Texte (Text-to-text, Image-to-text etc.)
 - Bilder (Text-to-Image, Image-to-Image etc.)
 - Ton (Text-to-speech etc.)
- Bedenken:
 - Urheberschaft: Wem gehören die „neu“ generierten Informationen?
 - Deep Fakes: Wie können Fälschung erkannt werden?



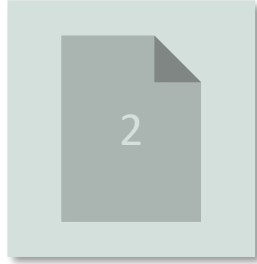
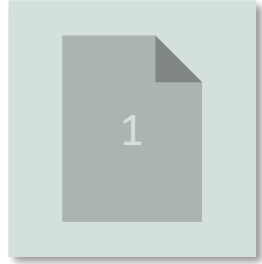
Software für moderne Machine Learning Systeme



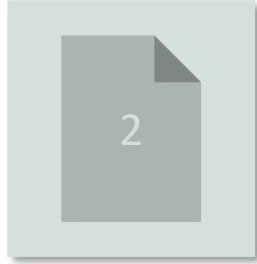
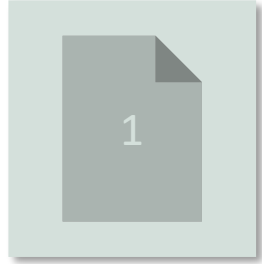
In Anlehnung an Huyen (2022)

Codebeispiel „Weizenbaum“

Was machen wir?

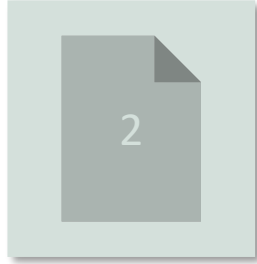
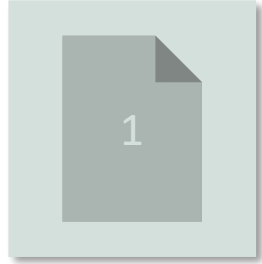


Was machen wir?

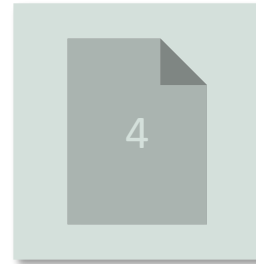


„A photo of @me with a
tall black hat and
sunglasses“

Was machen wir?

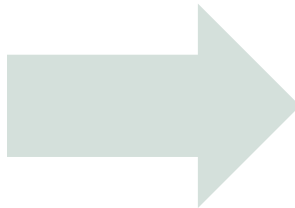


„A photo of @me with a
tall black hat and
sunglasses“



Vorlage des Beispiels

- Das hier gezeigte Beispiel stammt von „Santiago“
 - Geschäftsführung ml.school (Lehre von Machine Learning)
 - Autor bei Substack (<https://substack.com/profile/27043436-santiago>)
 - Autor bei Twitter (<https://twitter.com/0xbnomial>)
- Das Beispiel wurde unter <https://twitter.com/svpino/status/1635611742008147968> veröffentlicht.



Vier Schritte des Beispiels

- Schritt 1: Rahmenbedingungen
- Schritt 2: Unsere Funktionen definieren
- Schritt 3: Model bauen
- Schritt 4: Model anwenden & Bild generieren

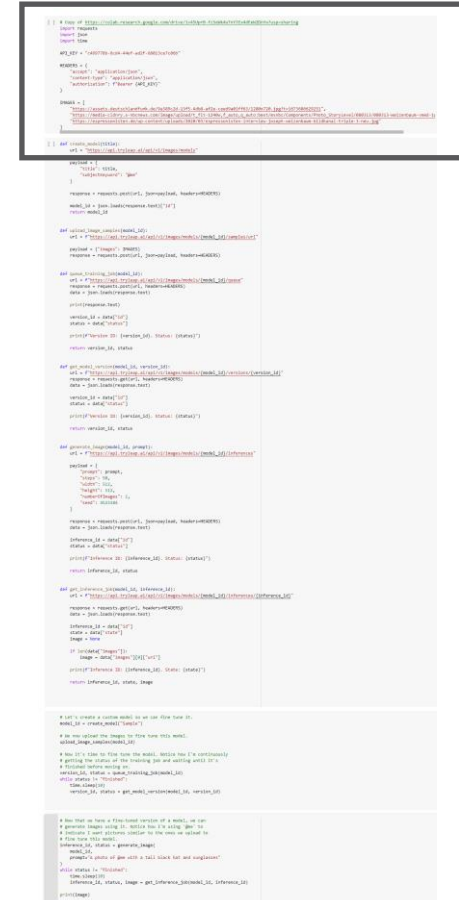
```
1 # Step 1: Import the required packages and define the model structure
2 import requests
3 import time
4 API_URL = "https://api.openai.com/v1/chat/completions"
5
6 # Step 2: Define the functions for the model
7 def get_model_response(prompt, model="gpt-4", temperature=0.7, max_tokens=100):
8     """
9     Get a response from the OpenAI API.
10    """
11    headers = {
12        "Content-Type": "application/json",
13        "Authorization": f"Bearer {API_KEY}"
14    }
15    payload = {
16        "model": model,
17        "prompt": prompt,
18        "temperature": temperature,
19        "max_tokens": max_tokens
20    }
21    response = requests.post(API_URL, headers=headers, json=payload)
22    return response.json()
23
24 # Step 3: Build the model
25 def build_model(prompt, model="gpt-4", temperature=0.7, max_tokens=100):
26     """
27     Build the model by calling the get_model_response function.
28    """
29    response = get_model_response(prompt, model, temperature, max_tokens)
30    return response
31
32 # Step 4: Apply the model to generate an image
33 def generate_image(prompt, model="gpt-4", temperature=0.7, max_tokens=100):
34     """
35     Generate an image by calling the build_model function.
36    """
37    response = build_model(prompt, model, temperature, max_tokens)
38    image_url = response["image_url"]
39    return image_url
40
41 # Step 5: Main function to run the model
42 def main():
43     # Step 5.1: Define the prompt
44     prompt = "A landscape with a large tree and a small house."
45     # Step 5.2: Call the generate_image function
46     image_url = generate_image(prompt, model="gpt-4", temperature=0.7, max_tokens=100)
47     # Step 5.3: Print the image URL
48     print(image_url)
49
50 # Step 6: Run the main function
51 if __name__ == "__main__":
52     main()
```


- 4

17

Schritt 1: Rahmenbedingungen

- Wir nutzen Bibliotheken, d.h. Programme, die andere Personen geschrieben und veröffentlicht haben.
- Wir legen Parameter fest, die die Rahmenbedingungen für unser Programm festlegen:
 - Welchen KI-Dienst wollen wir nutzen? Wie authentifizieren wir uns gegen diesen Dienst?
 - Welche Bilder wollen wir als Grundlagen nutzen?



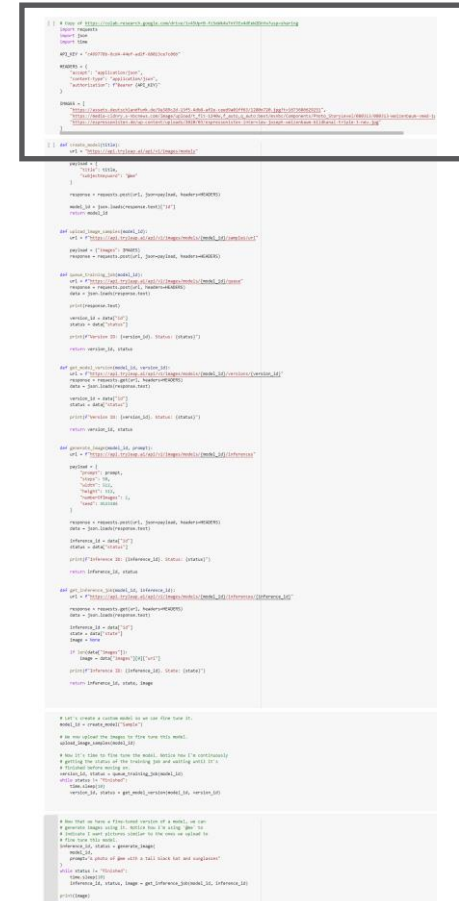
Schritt 1: Rahmenbedingungen

```
import requests
import json
import time
```

```
API_KEY = "c499778b-6cd4-44ef-ad2f-68013ce7c06b"
```

```
HEADERS = {
    "accept": "application/json",
    "content-type": "application/json",
    "authorization": f"Bearer {API_KEY}"
}
```

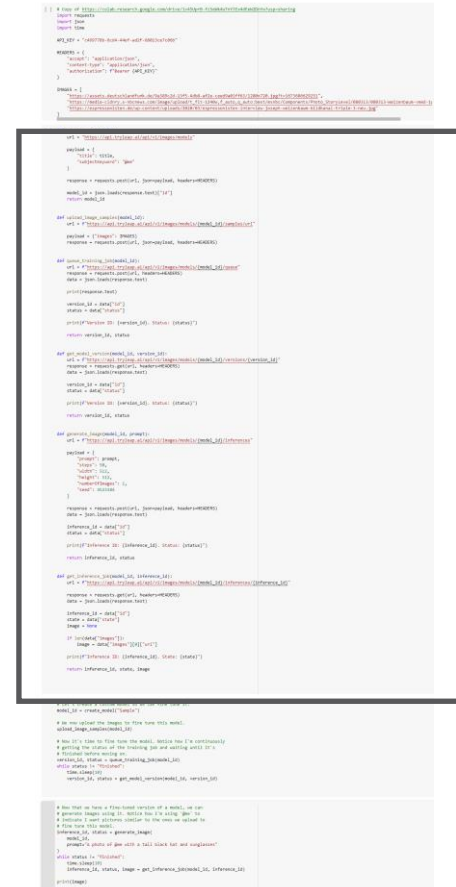
```
IMAGES = [
    "https://assets.deutschlandfunk.de/9a389c2d-13f5-4db8-af2e-ceed9a01ff63/1280x720.jpg?t=1673686629251",
    "https://media-cldnry.s-nbcnews.com/image/upload/t_fit-1240w,f_auto,q_auto:best/msnbc/Components/Photo_StoryLevel/080313/080313-Weizenbaum-vmed-1p.jpg",
    "https://espressonisten.de/wp-content/uploads/2020/03/espressonisten-interview-joseph-weizenbaum-bildkanal-triple-3-neu.jpg"
]
```



Schritt 2: Unsere Funktionen definieren

- Wir müssen ein neues Model bei unserem KI-Dienst anlegen (create_model).
- Wir müssen die Bilder beim KI-Dienst hochladen (upload_image_samples).
- Wir müssen unser Model in eine Warteschlange bei unserem Dienstleister einreihen können (queue_training_job).
- Wir müssen unser Model und unsere Bildgenerierungen über jeweils eine Kennziffer in der Datenbank des Dienstleisters ausfindig machen (get_model_version, get_inference_job).
- Wir müssen ein Bild aus unserem Model generieren (generate_image).

```
def create_model(title)
def upload_image_samples(model_id)
def queue_training_job(model_id)
def get_model_version(model_id, version_id)
def generate_image(model_id, prompt)
def get_inference_job(model_id, inference_id)
```



- [illegible]

```
"prompt": prompt,  
"steps": 50,  
"width": 512,  
"height": 512,  
"numberOfImages": 1,  
"seed": 4523184
```

```
return inference_id, status
```

22

Schritt 3: Model bauen

- Unser Dienstleister stellt uns ein **Foundation Model** zur Verfügung, welches wir **fine-tunen** können für unsere Zwecke.

```
1 # Fine-tune the model using the provided data and the foundation model
2 import requests
3 import json
4 import time
5 API_KEY = "sk-12345678901234567890123456789012"
6 BASE_URL = "https://api.openai.com/v1"
7 MODEL = "gpt-4o-mini"
8 # Create a new OpenAI client
9 client = OpenAI(api_key=API_KEY, base_url=BASE_URL)
10 # Create a new chat completion request
11 messages = [
12     {"role": "system", "content": "You are a helpful assistant."},
13     {"role": "user", "content": "Hello, how are you?"}
14 ]
15 # Create a new chat completion request
16 response = client.chat.completions.create(
17     model=MODEL,
18     messages=messages,
19     temperature=0.7,
20     max_tokens=100
21 )
22 # Print the response
23 print(response.choices[0].message.content)
```

```
1 # Fine-tune the model using the provided data and the foundation model
2 import requests
3 import json
4 import time
5 API_KEY = "sk-12345678901234567890123456789012"
6 BASE_URL = "https://api.openai.com/v1"
7 MODEL = "gpt-4o-mini"
8 # Create a new OpenAI client
9 client = OpenAI(api_key=API_KEY, base_url=BASE_URL)
10 # Create a new chat completion request
11 messages = [
12     {"role": "system", "content": "You are a helpful assistant."},
13     {"role": "user", "content": "Hello, how are you?"}
14 ]
15 # Create a new chat completion request
16 response = client.chat.completions.create(
17     model=MODEL,
18     messages=messages,
19     temperature=0.7,
20     max_tokens=100
21 )
22 # Print the response
23 print(response.choices[0].message.content)
```

```
1 # Fine-tune the model using the provided data and the foundation model
2 import requests
3 import json
4 import time
5 API_KEY = "sk-12345678901234567890123456789012"
6 BASE_URL = "https://api.openai.com/v1"
7 MODEL = "gpt-4o-mini"
8 # Create a new OpenAI client
9 client = OpenAI(api_key=API_KEY, base_url=BASE_URL)
10 # Create a new chat completion request
11 messages = [
12     {"role": "system", "content": "You are a helpful assistant."},
13     {"role": "user", "content": "Hello, how are you?"}
14 ]
15 # Create a new chat completion request
16 response = client.chat.completions.create(
17     model=MODEL,
18     messages=messages,
19     temperature=0.7,
20     max_tokens=100
21 )
22 # Print the response
23 print(response.choices[0].message.content)
```

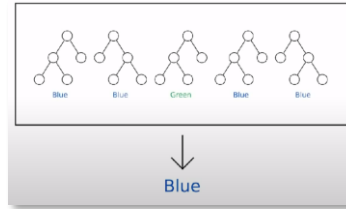
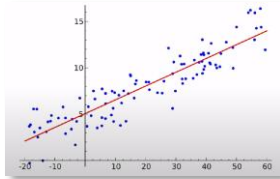
```
1 # Fine-tune the model using the provided data and the foundation model
2 import requests
3 import json
4 import time
5 API_KEY = "sk-12345678901234567890123456789012"
6 BASE_URL = "https://api.openai.com/v1"
7 MODEL = "gpt-4o-mini"
8 # Create a new OpenAI client
9 client = OpenAI(api_key=API_KEY, base_url=BASE_URL)
10 # Create a new chat completion request
11 messages = [
12     {"role": "system", "content": "You are a helpful assistant."},
13     {"role": "user", "content": "Hello, how are you?"}
14 ]
15 # Create a new chat completion request
16 response = client.chat.completions.create(
17     model=MODEL,
18     messages=messages,
19     temperature=0.7,
20     max_tokens=100
21 )
22 # Print the response
23 print(response.choices[0].message.content)
```

Schritt 3: Model bauen

Foundation Model

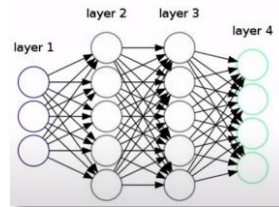
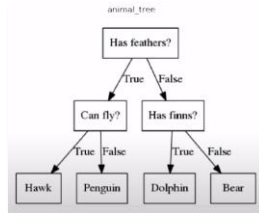
- Ein Model im Wortgebrauch der KI, ist eine Sammlung von statistischen Funktionen, anhand derer Voraussagen getroffen werden können.
- Ein Foundation Model wird genutzt, um ein möglichst breite Anwendungsvielfalt zu gewährleisten.

Lineare
Regression



Random Forest

Decision Tree



Neuronales Netz

```
1 # Importing the dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor

2 # Loading the dataset
dataset = pd.read_csv('data/linear_regression_data.csv')
X = dataset[['x1', 'x2', 'x3', 'x4', 'x5']]
y = dataset['y']

3 # Splitting the dataset into the training set and the test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

4 # Standardizing the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

5 # Fitting the Linear Regression model to the training set
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

6 # Predicting the y values of the test set using the linear model
y_pred_linear = linear_model.predict(X_test)

7 # Calculating the R-squared value and the Mean Squared Error for the linear model
r2_linear = r2_score(y_test, y_pred_linear)
mse_linear = mean_squared_error(y_test, y_pred_linear)

8 # Fitting the Random Forest model to the training set
rf_model = RandomForestRegressor()
rf_model.fit(X_train, y_train)

9 # Predicting the y values of the test set using the Random Forest model
y_pred_rf = rf_model.predict(X_test)

10 # Calculating the R-squared value and the Mean Squared Error for the Random Forest model
r2_rf = r2_score(y_test, y_pred_rf)
mse_rf = mean_squared_error(y_test, y_pred_rf)

11 # Fitting the Neural Network model to the training set
mlp_model = MLPRegressor()
mlp_model.fit(X_train, y_train)

12 # Predicting the y values of the test set using the Neural Network model
y_pred_mlp = mlp_model.predict(X_test)

13 # Calculating the R-squared value and the Mean Squared Error for the Neural Network model
r2_mlp = r2_score(y_test, y_pred_mlp)
mse_mlp = mean_squared_error(y_test, y_pred_mlp)

14 # Comparing the results of the three models
print('Linear Regression: R-squared =', r2_linear, 'MSE =', mse_linear)
print('Random Forest: R-squared =', r2_rf, 'MSE =', mse_rf)
print('Neural Network: R-squared =', r2_mlp, 'MSE =', mse_mlp)
```



```

# Let's create a custom model to do our own thing.
model = keras.models.Sequential([
    # an input layer that maps to 1000 units (this is the input image)
    keras.layers.Dense(1000, activation='sigmoid'),
    # an output layer that maps to 10 units (this is the output)
    keras.layers.Dense(10, activation='sigmoid')
])

# compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

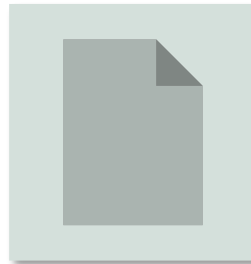
# train the model
model.fit(train_data, train_labels, validation_data=(test_data, test_labels),
        epochs=100, verbose=1)

```

Schritt 3: Model bauen

DALL-E mini

- Ein Model bringt eine „Training“-Pipeline mit.



```
1 # This file is part of the DALL-E mini project, which is a simplified version of the DALL-E model.
2 # It is designed to be used as a reference for how to build a simple GPT-2 model.
3 # The code is written in Python and uses the PyTorch library for deep learning.
4 # The model is trained on a dataset of images and their corresponding captions.
5 # The training process is split into two parts: a pre-training phase and a fine-tuning phase.
6 # The pre-training phase is used to learn the general structure of the data, while the fine-tuning phase is used to learn the specific details of the task.
7 # The model is evaluated on a separate dataset to measure its performance.
8 # The results of the training and evaluation are printed to the console.
9 # The code is organized into several functions, which are called in a specific order.
10 # The main function is responsible for setting up the environment and running the training pipeline.
11 # The other functions handle the data loading, model building, training, and evaluation steps.
12 # The code is written in a modular way, making it easy to modify and extend.
13 # The comments provide detailed explanations of each step and the underlying concepts.
14 # The code is intended to be used as a learning tool for those interested in deep learning and GPT-2.
15 # The project is licensed under the MIT license, which allows for free use and distribution.
16 # The code is written in a clean and readable style, following best practices for Python development.
17 # The comments are written in a clear and concise manner, providing helpful information for the reader.
18 # The code is well-documented, making it easy to understand and use.
19 # The project is a good example of how to build a simple GPT-2 model using PyTorch.
20 # The code is a valuable resource for anyone looking to learn more about GPT-2 and deep learning.
21 # The project is a great starting point for those interested in building their own GPT-2 model.
22 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
23 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
24 # The code is a great resource for anyone looking to build a simple GPT-2 model.
25 # The project is a useful reference for understanding the DALL-E mini model and its training process.
26 # The code is a great starting point for those interested in building their own GPT-2 model.
27 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
28 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
29 # The project is a great resource for anyone looking to build a simple GPT-2 model.
30 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
31 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
32 # The project is a great resource for anyone looking to build a simple GPT-2 model.
33 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
34 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
35 # The project is a great resource for anyone looking to build a simple GPT-2 model.
36 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
37 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
38 # The project is a great resource for anyone looking to build a simple GPT-2 model.
39 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
40 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
41 # The project is a great resource for anyone looking to build a simple GPT-2 model.
42 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
43 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
44 # The project is a great resource for anyone looking to build a simple GPT-2 model.
45 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
46 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
47 # The project is a great resource for anyone looking to build a simple GPT-2 model.
48 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
49 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
50 # The project is a great resource for anyone looking to build a simple GPT-2 model.
51 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
52 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
53 # The project is a great resource for anyone looking to build a simple GPT-2 model.
54 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
55 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
56 # The project is a great resource for anyone looking to build a simple GPT-2 model.
57 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
58 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
59 # The project is a great resource for anyone looking to build a simple GPT-2 model.
60 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
61 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
62 # The project is a great resource for anyone looking to build a simple GPT-2 model.
63 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
64 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
65 # The project is a great resource for anyone looking to build a simple GPT-2 model.
66 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
67 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
68 # The project is a great resource for anyone looking to build a simple GPT-2 model.
69 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
70 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
71 # The project is a great resource for anyone looking to build a simple GPT-2 model.
72 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
73 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
74 # The project is a great resource for anyone looking to build a simple GPT-2 model.
75 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
76 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
77 # The project is a great resource for anyone looking to build a simple GPT-2 model.
78 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
79 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
80 # The project is a great resource for anyone looking to build a simple GPT-2 model.
81 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
82 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
83 # The project is a great resource for anyone looking to build a simple GPT-2 model.
84 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
85 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
86 # The project is a great resource for anyone looking to build a simple GPT-2 model.
87 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
88 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
89 # The project is a great resource for anyone looking to build a simple GPT-2 model.
90 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
91 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
92 # The project is a great resource for anyone looking to build a simple GPT-2 model.
93 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
94 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
95 # The project is a great resource for anyone looking to build a simple GPT-2 model.
96 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
97 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
98 # The project is a great resource for anyone looking to build a simple GPT-2 model.
99 # The project is a helpful tool for learning about the DALL-E mini model and its training process.
100 # The code is a useful reference for understanding the inner workings of the DALL-E mini model.
```

```

# let's create a custom model to use our flow from 2d
model_2d = create_model_2d(layers)

# so we upload the design to flow from this model
upload_design(model_2d)

# let's try to fit flow from the model, since it's continuously
# getting the status of flow training log and saving until 20 s
# finished before saving on
version_2d = train_flow_training_version_2d()

# after saving it to "flow2d"
flow_train_2d()

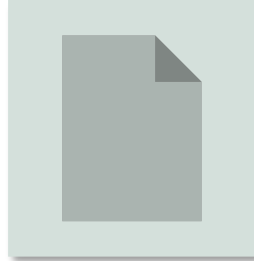
version_2d.status = get_model_version_2d()
version_2d

```

Schritt 3: Model bauen

DALL-E mini

- Ein Model bringt eine „Inferenz“-Pipeline mit.



```
1 # Run if you want to train the model on your own data
2 import requests
3 import json
4 import time
5
6 # The URL of the OpenAI API
7 URL = "https://api.openai.com/v1/images/generations"
8
9 # The API key
10 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
11
12 # The prompt
13 PROMPT = "A cute baby sea otter floating in the ocean."
14
15 # The number of images to generate
16 N_IMAGES = 4
17
18 # The number of variations to generate for each image
19 N_VARIATIONS = 3
20
21 # The number of steps to use for each image
22 N_STEPS = 50
23
24 # The temperature to use for each image
25 TEMPERATURE = 0.7
26
27 # The top_p to use for each image
28 TOP_P = 0.5
29
30 # The top_k to use for each image
31 TOP_K = 5
32
33 # The batch_size to use for each image
34 BATCH_SIZE = 1
35
36 # The timeout to use for each image
37 TIMEOUT = 60
38
39 # The headers to use for each image
40 HEADERS = {
41     "Content-Type": "application/json",
42     "Authorization": f"Bearer {API_KEY}"
43 }
44
45 # The data to use for each image
46 DATA = {
47     "prompt": PROMPT,
48     "n": N_IMAGES,
49     "size": "1024x1024",
50     "response_format": "url",
51     "user": "user"
52 }
53
54 # The URL of the OpenAI API
55 URL = "https://api.openai.com/v1/images/generations"
56
57 # The API key
58 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
59
60 # The prompt
61 PROMPT = "A cute baby sea otter floating in the ocean."
62
63 # The number of images to generate
64 N_IMAGES = 4
65
66 # The number of variations to generate for each image
67 N_VARIATIONS = 3
68
69 # The number of steps to use for each image
70 N_STEPS = 50
71
72 # The temperature to use for each image
73 TEMPERATURE = 0.7
74
75 # The top_p to use for each image
76 TOP_P = 0.5
77
78 # The top_k to use for each image
79 TOP_K = 5
80
81 # The batch_size to use for each image
82 BATCH_SIZE = 1
83
84 # The timeout to use for each image
85 TIMEOUT = 60
86
87 # The headers to use for each image
88 HEADERS = {
89     "Content-Type": "application/json",
90     "Authorization": f"Bearer {API_KEY}"
91 }
92
93 # The data to use for each image
94 DATA = {
95     "prompt": PROMPT,
96     "n": N_IMAGES,
97     "size": "1024x1024",
98     "response_format": "url",
99     "user": "user"
100 }
101
102 # The URL of the OpenAI API
103 URL = "https://api.openai.com/v1/images/generations"
104
105 # The API key
106 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
107
108 # The prompt
109 PROMPT = "A cute baby sea otter floating in the ocean."
110
111 # The number of images to generate
112 N_IMAGES = 4
113
114 # The number of variations to generate for each image
115 N_VARIATIONS = 3
116
117 # The number of steps to use for each image
118 N_STEPS = 50
119
120 # The temperature to use for each image
121 TEMPERATURE = 0.7
122
123 # The top_p to use for each image
124 TOP_P = 0.5
125
126 # The top_k to use for each image
127 TOP_K = 5
128
129 # The batch_size to use for each image
130 BATCH_SIZE = 1
131
132 # The timeout to use for each image
133 TIMEOUT = 60
134
135 # The headers to use for each image
136 HEADERS = {
137     "Content-Type": "application/json",
138     "Authorization": f"Bearer {API_KEY}"
139 }
140
141 # The data to use for each image
142 DATA = {
143     "prompt": PROMPT,
144     "n": N_IMAGES,
145     "size": "1024x1024",
146     "response_format": "url",
147     "user": "user"
148 }
149
150 # The URL of the OpenAI API
151 URL = "https://api.openai.com/v1/images/generations"
152
153 # The API key
154 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
155
156 # The prompt
157 PROMPT = "A cute baby sea otter floating in the ocean."
158
159 # The number of images to generate
160 N_IMAGES = 4
161
162 # The number of variations to generate for each image
163 N_VARIATIONS = 3
164
165 # The number of steps to use for each image
166 N_STEPS = 50
167
168 # The temperature to use for each image
169 TEMPERATURE = 0.7
170
171 # The top_p to use for each image
172 TOP_P = 0.5
173
174 # The top_k to use for each image
175 TOP_K = 5
176
177 # The batch_size to use for each image
178 BATCH_SIZE = 1
179
180 # The timeout to use for each image
181 TIMEOUT = 60
182
183 # The headers to use for each image
184 HEADERS = {
185     "Content-Type": "application/json",
186     "Authorization": f"Bearer {API_KEY}"
187 }
188
189 # The data to use for each image
190 DATA = {
191     "prompt": PROMPT,
192     "n": N_IMAGES,
193     "size": "1024x1024",
194     "response_format": "url",
195     "user": "user"
196 }
197
198 # The URL of the OpenAI API
199 URL = "https://api.openai.com/v1/images/generations"
200
201 # The API key
202 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
203
204 # The prompt
205 PROMPT = "A cute baby sea otter floating in the ocean."
206
207 # The number of images to generate
208 N_IMAGES = 4
209
210 # The number of variations to generate for each image
211 N_VARIATIONS = 3
212
213 # The number of steps to use for each image
214 N_STEPS = 50
215
216 # The temperature to use for each image
217 TEMPERATURE = 0.7
218
219 # The top_p to use for each image
220 TOP_P = 0.5
221
222 # The top_k to use for each image
223 TOP_K = 5
224
225 # The batch_size to use for each image
226 BATCH_SIZE = 1
227
228 # The timeout to use for each image
229 TIMEOUT = 60
230
231 # The headers to use for each image
232 HEADERS = {
233     "Content-Type": "application/json",
234     "Authorization": f"Bearer {API_KEY}"
235 }
236
237 # The data to use for each image
238 DATA = {
239     "prompt": PROMPT,
240     "n": N_IMAGES,
241     "size": "1024x1024",
242     "response_format": "url",
243     "user": "user"
244 }
245
246 # The URL of the OpenAI API
247 URL = "https://api.openai.com/v1/images/generations"
248
249 # The API key
250 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
251
252 # The prompt
253 PROMPT = "A cute baby sea otter floating in the ocean."
254
255 # The number of images to generate
256 N_IMAGES = 4
257
258 # The number of variations to generate for each image
259 N_VARIATIONS = 3
260
261 # The number of steps to use for each image
262 N_STEPS = 50
263
264 # The temperature to use for each image
265 TEMPERATURE = 0.7
266
267 # The top_p to use for each image
268 TOP_P = 0.5
269
270 # The top_k to use for each image
271 TOP_K = 5
272
273 # The batch_size to use for each image
274 BATCH_SIZE = 1
275
276 # The timeout to use for each image
277 TIMEOUT = 60
278
279 # The headers to use for each image
280 HEADERS = {
281     "Content-Type": "application/json",
282     "Authorization": f"Bearer {API_KEY}"
283 }
284
285 # The data to use for each image
286 DATA = {
287     "prompt": PROMPT,
288     "n": N_IMAGES,
289     "size": "1024x1024",
290     "response_format": "url",
291     "user": "user"
292 }
293
294 # The URL of the OpenAI API
295 URL = "https://api.openai.com/v1/images/generations"
296
297 # The API key
298 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
299
300 # The prompt
301 PROMPT = "A cute baby sea otter floating in the ocean."
302
303 # The number of images to generate
304 N_IMAGES = 4
305
306 # The number of variations to generate for each image
307 N_VARIATIONS = 3
308
309 # The number of steps to use for each image
310 N_STEPS = 50
311
312 # The temperature to use for each image
313 TEMPERATURE = 0.7
314
315 # The top_p to use for each image
316 TOP_P = 0.5
317
318 # The top_k to use for each image
319 TOP_K = 5
320
321 # The batch_size to use for each image
322 BATCH_SIZE = 1
323
324 # The timeout to use for each image
325 TIMEOUT = 60
326
327 # The headers to use for each image
328 HEADERS = {
329     "Content-Type": "application/json",
330     "Authorization": f"Bearer {API_KEY}"
331 }
332
333 # The data to use for each image
334 DATA = {
335     "prompt": PROMPT,
336     "n": N_IMAGES,
337     "size": "1024x1024",
338     "response_format": "url",
339     "user": "user"
340 }
341
342 # The URL of the OpenAI API
343 URL = "https://api.openai.com/v1/images/generations"
344
345 # The API key
346 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
347
348 # The prompt
349 PROMPT = "A cute baby sea otter floating in the ocean."
350
351 # The number of images to generate
352 N_IMAGES = 4
353
354 # The number of variations to generate for each image
355 N_VARIATIONS = 3
356
357 # The number of steps to use for each image
358 N_STEPS = 50
359
360 # The temperature to use for each image
361 TEMPERATURE = 0.7
362
363 # The top_p to use for each image
364 TOP_P = 0.5
365
366 # The top_k to use for each image
367 TOP_K = 5
368
369 # The batch_size to use for each image
370 BATCH_SIZE = 1
371
372 # The timeout to use for each image
373 TIMEOUT = 60
374
375 # The headers to use for each image
376 HEADERS = {
377     "Content-Type": "application/json",
378     "Authorization": f"Bearer {API_KEY}"
379 }
380
381 # The data to use for each image
382 DATA = {
383     "prompt": PROMPT,
384     "n": N_IMAGES,
385     "size": "1024x1024",
386     "response_format": "url",
387     "user": "user"
388 }
389
390 # The URL of the OpenAI API
391 URL = "https://api.openai.com/v1/images/generations"
392
393 # The API key
394 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
395
396 # The prompt
397 PROMPT = "A cute baby sea otter floating in the ocean."
398
399 # The number of images to generate
400 N_IMAGES = 4
401
402 # The number of variations to generate for each image
403 N_VARIATIONS = 3
404
405 # The number of steps to use for each image
406 N_STEPS = 50
407
408 # The temperature to use for each image
409 TEMPERATURE = 0.7
410
411 # The top_p to use for each image
412 TOP_P = 0.5
413
414 # The top_k to use for each image
415 TOP_K = 5
416
417 # The batch_size to use for each image
418 BATCH_SIZE = 1
419
420 # The timeout to use for each image
421 TIMEOUT = 60
422
423 # The headers to use for each image
424 HEADERS = {
425     "Content-Type": "application/json",
426     "Authorization": f"Bearer {API_KEY}"
427 }
428
429 # The data to use for each image
430 DATA = {
431     "prompt": PROMPT,
432     "n": N_IMAGES,
433     "size": "1024x1024",
434     "response_format": "url",
435     "user": "user"
436 }
437
438 # The URL of the OpenAI API
439 URL = "https://api.openai.com/v1/images/generations"
440
441 # The API key
442 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
443
444 # The prompt
445 PROMPT = "A cute baby sea otter floating in the ocean."
446
447 # The number of images to generate
448 N_IMAGES = 4
449
450 # The number of variations to generate for each image
451 N_VARIATIONS = 3
452
453 # The number of steps to use for each image
454 N_STEPS = 50
455
456 # The temperature to use for each image
457 TEMPERATURE = 0.7
458
459 # The top_p to use for each image
460 TOP_P = 0.5
461
462 # The top_k to use for each image
463 TOP_K = 5
464
465 # The batch_size to use for each image
466 BATCH_SIZE = 1
467
468 # The timeout to use for each image
469 TIMEOUT = 60
470
471 # The headers to use for each image
472 HEADERS = {
473     "Content-Type": "application/json",
474     "Authorization": f"Bearer {API_KEY}"
475 }
476
477 # The data to use for each image
478 DATA = {
479     "prompt": PROMPT,
480     "n": N_IMAGES,
481     "size": "1024x1024",
482     "response_format": "url",
483     "user": "user"
484 }
485
486 # The URL of the OpenAI API
487 URL = "https://api.openai.com/v1/images/generations"
488
489 # The API key
490 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
491
492 # The prompt
493 PROMPT = "A cute baby sea otter floating in the ocean."
494
495 # The number of images to generate
496 N_IMAGES = 4
497
498 # The number of variations to generate for each image
499 N_VARIATIONS = 3
500
501 # The number of steps to use for each image
502 N_STEPS = 50
503
504 # The temperature to use for each image
505 TEMPERATURE = 0.7
506
507 # The top_p to use for each image
508 TOP_P = 0.5
509
510 # The top_k to use for each image
511 TOP_K = 5
512
513 # The batch_size to use for each image
514 BATCH_SIZE = 1
515
516 # The timeout to use for each image
517 TIMEOUT = 60
518
519 # The headers to use for each image
520 HEADERS = {
521     "Content-Type": "application/json",
522     "Authorization": f"Bearer {API_KEY}"
523 }
524
525 # The data to use for each image
526 DATA = {
527     "prompt": PROMPT,
528     "n": N_IMAGES,
529     "size": "1024x1024",
530     "response_format": "url",
531     "user": "user"
532 }
533
534 # The URL of the OpenAI API
535 URL = "https://api.openai.com/v1/images/generations"
536
537 # The API key
538 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
539
540 # The prompt
541 PROMPT = "A cute baby sea otter floating in the ocean."
542
543 # The number of images to generate
544 N_IMAGES = 4
545
546 # The number of variations to generate for each image
547 N_VARIATIONS = 3
548
549 # The number of steps to use for each image
550 N_STEPS = 50
551
552 # The temperature to use for each image
553 TEMPERATURE = 0.7
554
555 # The top_p to use for each image
556 TOP_P = 0.5
557
558 # The top_k to use for each image
559 TOP_K = 5
560
561 # The batch_size to use for each image
562 BATCH_SIZE = 1
563
564 # The timeout to use for each image
565 TIMEOUT = 60
566
567 # The headers to use for each image
568 HEADERS = {
569     "Content-Type": "application/json",
570     "Authorization": f"Bearer {API_KEY}"
571 }
572
573 # The data to use for each image
574 DATA = {
575     "prompt": PROMPT,
576     "n": N_IMAGES,
577     "size": "1024x1024",
578     "response_format": "url",
579     "user": "user"
580 }
581
582 # The URL of the OpenAI API
583 URL = "https://api.openai.com/v1/images/generations"
584
585 # The API key
586 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
587
588 # The prompt
589 PROMPT = "A cute baby sea otter floating in the ocean."
590
591 # The number of images to generate
592 N_IMAGES = 4
593
594 # The number of variations to generate for each image
595 N_VARIATIONS = 3
596
597 # The number of steps to use for each image
598 N_STEPS = 50
599
600 # The temperature to use for each image
601 TEMPERATURE = 0.7
602
603 # The top_p to use for each image
604 TOP_P = 0.5
605
606 # The top_k to use for each image
607 TOP_K = 5
608
609 # The batch_size to use for each image
610 BATCH_SIZE = 1
611
612 # The timeout to use for each image
613 TIMEOUT = 60
614
615 # The headers to use for each image
616 HEADERS = {
617     "Content-Type": "application/json",
618     "Authorization": f"Bearer {API_KEY}"
619 }
620
621 # The data to use for each image
622 DATA = {
623     "prompt": PROMPT,
624     "n": N_IMAGES,
625     "size": "1024x1024",
626     "response_format": "url",
627     "user": "user"
628 }
629
630 # The URL of the OpenAI API
631 URL = "https://api.openai.com/v1/images/generations"
632
633 # The API key
634 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
635
636 # The prompt
637 PROMPT = "A cute baby sea otter floating in the ocean."
638
639 # The number of images to generate
640 N_IMAGES = 4
641
642 # The number of variations to generate for each image
643 N_VARIATIONS = 3
644
645 # The number of steps to use for each image
646 N_STEPS = 50
647
648 # The temperature to use for each image
649 TEMPERATURE = 0.7
650
651 # The top_p to use for each image
652 TOP_P = 0.5
653
654 # The top_k to use for each image
655 TOP_K = 5
656
657 # The batch_size to use for each image
658 BATCH_SIZE = 1
659
660 # The timeout to use for each image
661 TIMEOUT = 60
662
663 # The headers to use for each image
664 HEADERS = {
665     "Content-Type": "application/json",
666     "Authorization": f"Bearer {API_KEY}"
667 }
668
669 # The data to use for each image
670 DATA = {
671     "prompt": PROMPT,
672     "n": N_IMAGES,
673     "size": "1024x1024",
674     "response_format": "url",
675     "user": "user"
676 }
677
678 # The URL of the OpenAI API
679 URL = "https://api.openai.com/v1/images/generations"
680
681 # The API key
682 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
683
684 # The prompt
685 PROMPT = "A cute baby sea otter floating in the ocean."
686
687 # The number of images to generate
688 N_IMAGES = 4
689
690 # The number of variations to generate for each image
691 N_VARIATIONS = 3
692
693 # The number of steps to use for each image
694 N_STEPS = 50
695
696 # The temperature to use for each image
697 TEMPERATURE = 0.7
698
699 # The top_p to use for each image
700 TOP_P = 0.5
701
702 # The top_k to use for each image
703 TOP_K = 5
704
705 # The batch_size to use for each image
706 BATCH_SIZE = 1
707
708 # The timeout to use for each image
709 TIMEOUT = 60
710
711 # The headers to use for each image
712 HEADERS = {
713     "Content-Type": "application/json",
714     "Authorization": f"Bearer {API_KEY}"
715 }
716
717 # The data to use for each image
718 DATA = {
719     "prompt": PROMPT,
720     "n": N_IMAGES,
721     "size": "1024x1024",
722     "response_format": "url",
723     "user": "user"
724 }
725
726 # The URL of the OpenAI API
727 URL = "https://api.openai.com/v1/images/generations"
728
729 # The API key
730 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
731
732 # The prompt
733 PROMPT = "A cute baby sea otter floating in the ocean."
734
735 # The number of images to generate
736 N_IMAGES = 4
737
738 # The number of variations to generate for each image
739 N_VARIATIONS = 3
740
741 # The number of steps to use for each image
742 N_STEPS = 50
743
744 # The temperature to use for each image
745 TEMPERATURE = 0.7
746
747 # The top_p to use for each image
748 TOP_P = 0.5
749
750 # The top_k to use for each image
751 TOP_K = 5
752
753 # The batch_size to use for each image
754 BATCH_SIZE = 1
755
756 # The timeout to use for each image
757 TIMEOUT = 60
758
759 # The headers to use for each image
760 HEADERS = {
761     "Content-Type": "application/json",
762     "Authorization": f"Bearer {API_KEY}"
763 }
764
765 # The data to use for each image
766 DATA = {
767     "prompt": PROMPT,
768     "n": N_IMAGES,
769     "size": "1024x1024",
770     "response_format": "url",
771     "user": "user"
772 }
773
774 # The URL of the OpenAI API
775 URL = "https://api.openai.com/v1/images/generations"
776
777 # The API key
778 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
779
780 # The prompt
781 PROMPT = "A cute baby sea otter floating in the ocean."
782
783 # The number of images to generate
784 N_IMAGES = 4
785
786 # The number of variations to generate for each image
787 N_VARIATIONS = 3
788
789 # The number of steps to use for each image
790 N_STEPS = 50
791
792 # The temperature to use for each image
793 TEMPERATURE = 0.7
794
795 # The top_p to use for each image
796 TOP_P = 0.5
797
798 # The top_k to use for each image
799 TOP_K = 5
800
801 # The batch_size to use for each image
802 BATCH_SIZE = 1
803
804 # The timeout to use for each image
805 TIMEOUT = 60
806
807 # The headers to use for each image
808 HEADERS = {
809     "Content-Type": "application/json",
810     "Authorization": f"Bearer {API_KEY}"
811 }
812
813 # The data to use for each image
814 DATA = {
815     "prompt": PROMPT,
816     "n": N_IMAGES,
817     "size": "1024x1024",
818     "response_format": "url",
819     "user": "user"
820 }
821
822 # The URL of the OpenAI API
823 URL = "https://api.openai.com/v1/images/generations"
824
825 # The API key
826 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
827
828 # The prompt
829 PROMPT = "A cute baby sea otter floating in the ocean."
830
831 # The number of images to generate
832 N_IMAGES = 4
833
834 # The number of variations to generate for each image
835 N_VARIATIONS = 3
836
837 # The number of steps to use for each image
838 N_STEPS = 50
839
840 # The temperature to use for each image
841 TEMPERATURE = 0.7
842
843 # The top_p to use for each image
844 TOP_P = 0.5
845
846 # The top_k to use for each image
847 TOP_K = 5
848
849 # The batch_size to use for each image
850 BATCH_SIZE = 1
851
852 # The timeout to use for each image
853 TIMEOUT = 60
854
855 # The headers to use for each image
856 HEADERS = {
857     "Content-Type": "application/json",
858     "Authorization": f"Bearer {API_KEY}"
859 }
860
861 # The data to use for each image
862 DATA = {
863     "prompt": PROMPT,
864     "n": N_IMAGES,
865     "size": "1024x1024",
866     "response_format": "url",
867     "user": "user"
868 }
869
870 # The URL of the OpenAI API
871 URL = "https://api.openai.com/v1/images/generations"
872
873 # The API key
874 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
875
876 # The prompt
877 PROMPT = "A cute baby sea otter floating in the ocean."
878
879 # The number of images to generate
880 N_IMAGES = 4
881
882 # The number of variations to generate for each image
883 N_VARIATIONS = 3
884
885 # The number of steps to use for each image
886 N_STEPS = 50
887
888 # The temperature to use for each image
889 TEMPERATURE = 0.7
890
891 # The top_p to use for each image
892 TOP_P = 0.5
893
894 # The top_k to use for each image
895 TOP_K = 5
896
897 # The batch_size to use for each image
898 BATCH_SIZE = 1
899
900 # The timeout to use for each image
901 TIMEOUT = 60
902
903 # The headers to use for each image
904 HEADERS = {
905     "Content-Type": "application/json",
906     "Authorization": f"Bearer {API_KEY}"
907 }
908
909 # The data to use for each image
910 DATA = {
911     "prompt": PROMPT,
912     "n": N_IMAGES,
913     "size": "1024x1024",
914     "response_format": "url",
915     "user": "user"
916 }
917
918 # The URL of the OpenAI API
919 URL = "https://api.openai.com/v1/images/generations"
920
921 # The API key
922 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
923
924 # The prompt
925 PROMPT = "A cute baby sea otter floating in the ocean."
926
927 # The number of images to generate
928 N_IMAGES = 4
929
930 # The number of variations to generate for each image
931 N_VARIATIONS = 3
932
933 # The number of steps to use for each image
934 N_STEPS = 50
935
936 # The temperature to use for each image
937 TEMPERATURE = 0.7
938
939 # The top_p to use for each image
940 TOP_P = 0.5
941
942 # The top_k to use for each image
943 TOP_K = 5
944
945 # The batch_size to use for each image
946 BATCH_SIZE = 1
947
948 # The timeout to use for each image
949 TIMEOUT = 60
950
951 # The headers to use for each image
952 HEADERS = {
953     "Content-Type": "application/json",
954     "Authorization": f"Bearer {API_KEY}"
955 }
956
957 # The data to use for each image
958 DATA = {
959     "prompt": PROMPT,
960     "n": N_IMAGES,
961     "size": "1024x1024",
962     "response_format": "url",
963     "user": "user"
964 }
965
966 # The URL of the OpenAI API
967 URL = "https://api.openai.com/v1/images/generations"
968
969 # The API key
970 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
971
972 # The prompt
973 PROMPT = "A cute baby sea otter floating in the ocean."
974
975 # The number of images to generate
976 N_IMAGES = 4
977
978 # The number of variations to generate for each image
979 N_VARIATIONS = 3
980
981 # The number of steps to use for each image
982 N_STEPS = 50
983
984 # The temperature to use for each image
985 TEMPERATURE = 0.7
986
987 # The top_p to use for each image
988 TOP_P = 0.5
989
990 # The top_k to use for each image
991 TOP_K = 5
992
993 # The batch_size to use for each image
994 BATCH_SIZE = 1
995
996 # The timeout to use for each image
997 TIMEOUT = 60
998
999 # The headers to use for each image
1000 HEADERS = {
1001     "Content-Type": "application/json",
1002     "Authorization": f"Bearer {API_KEY}"
1003 }
1004
1005 # The data to use for each image
1006 DATA = {
1007     "prompt": PROMPT,
1008     "n": N_IMAGES,
1009     "size": "1024x1024",
1010     "response_format": "url",
1011     "user": "user"
1012 }
1013
1014 # The URL of the OpenAI API
1015 URL = "https://api.openai.com/v1/images/generations"
1016
1017 # The API key
1018 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
1019
1020 # The prompt
1021 PROMPT = "A cute baby sea otter floating in the ocean."
1022
1023 # The number of images to generate
1024 N_IMAGES = 4
1025
1026 # The number of variations to generate for each image
1027 N_VARIATIONS = 3
1028
1029 # The number of steps to use for each image
1030 N_STEPS = 50
1031
1032 # The temperature to use for each image
1033 TEMPERATURE = 0.7
1034
1035 # The top_p to use for each image
1036 TOP_P = 0.5
1037
1038 # The top_k to use for each image
1039 TOP_K = 5
1040
1041 # The batch_size to use for each image
1042 BATCH_SIZE = 1
1043
1044 # The timeout to use for each image
1045 TIMEOUT = 60
1046
1047 # The headers to use for each image
1048 HEADERS = {
1049     "Content-Type": "application/json",
1050     "Authorization": f"Bearer {API_KEY}"
1051 }
1052
1053 # The data to use for each image
1054 DATA = {
1055     "prompt": PROMPT,
1056     "n": N_IMAGES,
1057     "size": "1024x1024",
1058     "response_format": "url",
1059     "user": "user"
1060 }
1061
1062 # The URL of the OpenAI API
1063 URL = "https://api.openai.com/v1/images/generations"
1064
1065 # The API key
1066 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
1067
1068 # The prompt
1069 PROMPT = "A cute baby sea otter floating in the ocean."
1070
1071 # The number of images to generate
1072 N_IMAGES = 4
1073
1074 # The number of variations to generate for each image
1075 N_VARIATIONS = 3
1076
1077 # The number of steps to use for each image
1078 N_STEPS = 50
1079
1080 # The temperature to use for each image
1081 TEMPERATURE = 0.7
1082
1083 # The top_p to use for each image
1084 TOP_P = 0.5
1085
1086 # The top_k to use for each image
1087 TOP_K = 5
1088
1089 # The batch_size to use for each image
1090 BATCH_SIZE = 1
1091
1092 # The timeout to use for each image
1093 TIMEOUT = 60
1094
1095 # The headers to use for each image
1096 HEADERS = {
1097     "Content-Type": "application/json",
1098     "Authorization": f"Bearer {API_KEY}"
1099 }
1100
1101 # The data to use for each image
1102 DATA = {
1103     "prompt": PROMPT,
1104     "n": N_IMAGES,
1105     "size": "1024x1024",
1106     "response_format": "url",
1107     "user": "user"
1108 }
1109
1110 # The URL of the OpenAI API
1111 URL = "https://api.openai.com/v1/images/generations"
1112
1113 # The API key
1114 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
1115
1116 # The prompt
1117 PROMPT = "A cute baby sea otter floating in the ocean."
1118
1119 # The number of images to generate
1120 N_IMAGES = 4
1121
1122 # The number of variations to generate for each image
1123 N_VARIATIONS = 3
1124
1125 # The number of steps to use for each image
1126 N_STEPS = 50
1127
1128 # The temperature to use for each image
1129 TEMPERATURE = 0.7
1130
1131 # The top_p to use for each image
1132 TOP_P = 0.5
1133
1134 # The top_k to use for each image
1135 TOP_K = 5
1136
1137 # The batch_size to use for each image
1138 BATCH_SIZE = 1
1139
1140 # The timeout to use for each image
1141 TIMEOUT = 60
1142
1143 # The headers to use for each image
1144 HEADERS = {
1145     "Content-Type": "application/json",
1146     "Authorization": f"Bearer {API_KEY}"
1147 }
1148
1149 # The data to use for each image
1150 DATA = {
1151     "prompt": PROMPT,
1152     "n": N_IMAGES,
1153     "size": "1024x1024",
1154     "response_format": "url",
1155     "user": "user"
1156 }
1157
1158 # The URL of the OpenAI API
1159 URL = "https://api.openai.com/v1/images/generations"
1160
1161 # The API key
1162 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
1163
1164 # The prompt
1165 PROMPT = "A cute baby sea otter floating in the ocean."
1166
1167 # The number of images to generate
1168 N_IMAGES = 4
1169
1170 # The number of variations to generate for each image
1171 N_VARIATIONS = 3
1172
1173 # The number of steps to use for each image
1174 N_STEPS = 50
1175
1176 # The temperature to use for each image
1177 TEMPERATURE = 0.7
1178
1179 # The top_p to use for each image
1180 TOP_P = 0.5
1181
1182 # The top_k to use for each image
1183 TOP_K = 5
1184
1185 # The batch_size to use for each image
1186 BATCH_SIZE = 1
1187
1188 # The timeout to use for each image
1189 TIMEOUT = 60
1190
1191 # The headers to use for each image
1192 HEADERS = {
1193     "Content-Type": "application/json",
1194     "Authorization": f"Bearer {API_KEY}"
1195 }
1196
1197 # The data to use for each image
1198 DATA = {
1199     "prompt": PROMPT,
1200     "n": N_IMAGES,
1201     "size": "1024x1024",
1202     "response_format": "url",
1203     "user": "user"
1204 }
1205
1206 # The URL of the OpenAI API
1207 URL = "https://api.openai.com/v1/images/generations"
1208
1209 # The API key
1210 API_KEY = "sk-1234567890abcdefghijklmnopqrstuvwxyz0123456789"
1211
1212 # The prompt
1213 PROMPT = "A cute baby sea otter floating in the ocean."
1214
1215 # The number of images to generate
1216 N_IMAGES = 4
1217
1218 # The number of variations to generate for each image
1219 N_VARIATIONS = 3
1220
1221 # The number of steps to use for each image
1
```

Schritt 3: Model bauen

Fine-tuning

- Im Fine-Tuning lernt das Foundation Model Dinge, die für den spezifischen Anwendungsfall wichtig sind.

Elemente des fine-tuned Models

Elemente des Foundation Models

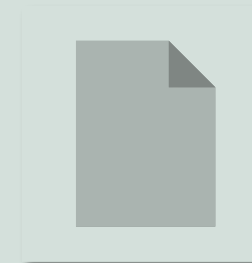
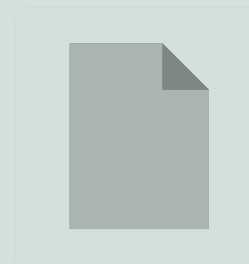
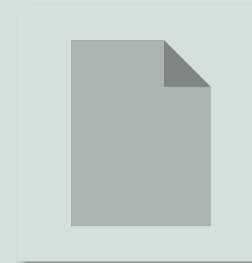


@me

„Schwarzer Hut“

„Sonnenbrille“

„Die Person, die auf den Bildern zu sehen ist“



```
1 # Fine-tune the model on the dataset
2 # Import the necessary libraries
3 import torch
4 import torchvision
5 from torch.utils.data import Dataset, DataLoader
6 from torchvision import transforms
7 from torch.nn import LstmCell, LSTM
8 from torch.optim import Adam
9 from tqdm import tqdm
10
11 # Define the dataset
12 class CustomDataset(Dataset):
13     def __init__(self, data_loader):
14         self.data_loader = data_loader
15
16     def __len__(self):
17         return len(self.data_loader)
18
19     def __getitem__(self, index):
20         data = self.data_loader[index]
21         return data
22
23 # Define the model
24 class CustomModel(LSTM):
25     def __init__(self, input_size, hidden_size, num_layers):
26         super().__init__(input_size, hidden_size, num_layers)
27
28     def forward(self, x):
29         _, (h, c) = self._init_states()
30         for i in range(x.size()[0]):
31             h, c = self._lstm_cell(x[i], h, c)
32         return h
33
34 # Define the training loop
35 def train(model, data_loader, optimizer):
36     for epoch in range(10):
37         model.train()
38         for data in data_loader:
39             optimizer.zero_grad()
40             output = model(data)
41             loss = criterion(output, target)
42             loss.backward()
43             optimizer.step()
44         print(f'Epoch {epoch} Loss: {loss.item()}')
```

```
# We now upload the images to fine tune this model.
upload_image_samples(model_id)
```

```
version_id, status = queue_training_job(model_id)
while status != "finished":
    time.sleep(10)
    version_id, status = get_model_version(model_id, version_id)
```

Der Code hinter KI | März 2023 | © Dr. Alexander Rachmann

Schritt 4: Model anwenden & Bild generieren

- Wir geben dem Dienstleister einen Prompt mit, mit einer Beschreibung was wir möchten.
- Der Dienstleister soll uns regelmäßig (alle 10 Sekunden) Bescheid geben, wie weit er ist.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This script is a simple example of how to use the OpenAI API to generate images.
# It uses the openai package to interact with the API.
# The script is designed to be run from the command line.
# It takes a single argument, the prompt, which is used to generate the image.
# The script will generate a single image and save it to a file named 'image.png'.
# The script will also print the URL of the generated image to the console.
# The script is designed to be run from the command line.
# It takes a single argument, the prompt, which is used to generate the image.
# The script will generate a single image and save it to a file named 'image.png'.
# The script will also print the URL of the generated image to the console.

import openai
import sys
import time
import os

# Set the OpenAI API key
openai.api_key = 'sk-1234567890abcdefghijklmnopqrstuvwxyz'

# Set the prompt
prompt = sys.argv[1]

# Generate the image
response = openai.Image.create(
    prompt=prompt,
    n=1,
    size='1024x1024'
)

# Save the image to a file
image_url = response[0]['url']
image_data = openai.Image.create_image_url(image_url)

# Print the URL of the generated image
print(image_url)
```

Schritt 4: Model anwenden & Bild generieren

```
# Now that we have a fine-tuned version of a model, we can  
# generate images using it. Notice how I'm using '@me' to  
# indicate I want pictures similar to the ones we upload to  
# fine tune this model.
```

```
inference_id, status = generate_image(  
    model_id,  
    prompt="A photo of @me with a tall black hat and sunglasses"  
)  
while status != "finished":  
    time.sleep(10)  
    inference_id, status, image = get_inference_job(model_id, inferen  
ce_id)  
  
print(image)
```



Übung

Nun seid ihr dran....

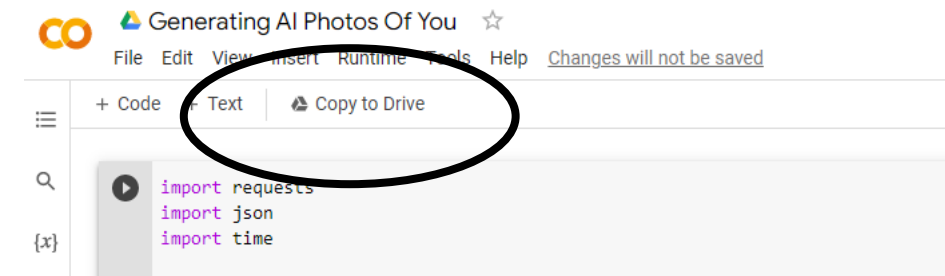
- 1. Account bei Google ist notwendig um Google Collab zu nutzen.

Nun seid ihr dran....

- 1. Account bei Google ist notwendig um Google Collab zu nutzen.
- 2. Account bei <https://www.tryleap.ai/> ist notwendig um ein Model zu bauen.

Nun seid ihr dran....

- 1. Account bei Google ist notwendig um Google Collab zu nutzen.
- 2. Account bei <https://www.tryleap.ai/> ist notwendig um ein Model zu bauen.
- 3. Code kopieren von <https://colab.research.google.com/drive/1v45UprB-fzSeWk4wTnYJEx4dEeW2DnYw?usp=sharing>

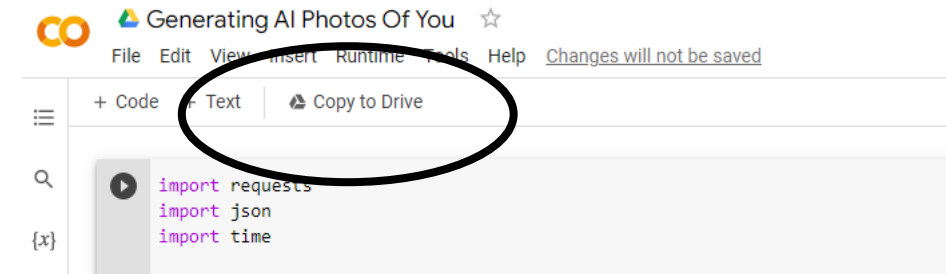


Nun seid ihr dran....

- 1. Account bei Google ist notwendig um Google Collab zu nutzen.
- 2. Account bei <https://www.tryleap.ai/> ist notwendig um ein Model zu bauen.
- 3. Code kopieren von <https://colab.research.google.com/drive/1v45UprB-fzSeWk4wTnYJEx4dEeW2DnYw?usp=sharing>

- 4. Bilder anpassen

```
IMAGES = [  
    //... "  
    //... "  
    //... "  
    //... "  
]
```



Nun seid ihr dran....

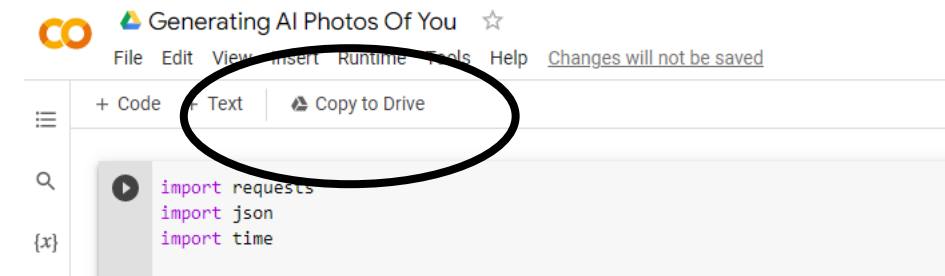
- 1. Account bei Google ist notwendig um Google Collab zu nutzen.
- 2. Account bei <https://www.tryleap.ai/> ist notwendig um ein Model zu bauen.
- 3. Code kopieren von <https://colab.research.google.com/drive/1v45UprB-fzSeWk4wTnYJEx4dEeW2DnYw?usp=sharing>

- 4. Bilder anpassen

```
IMAGES = [
    //...",
    //...",
    //..."]
```

- 5. Prompt anpassen

```
prompt="A photo of @me with a tall  
black hat and sunglasses"
```



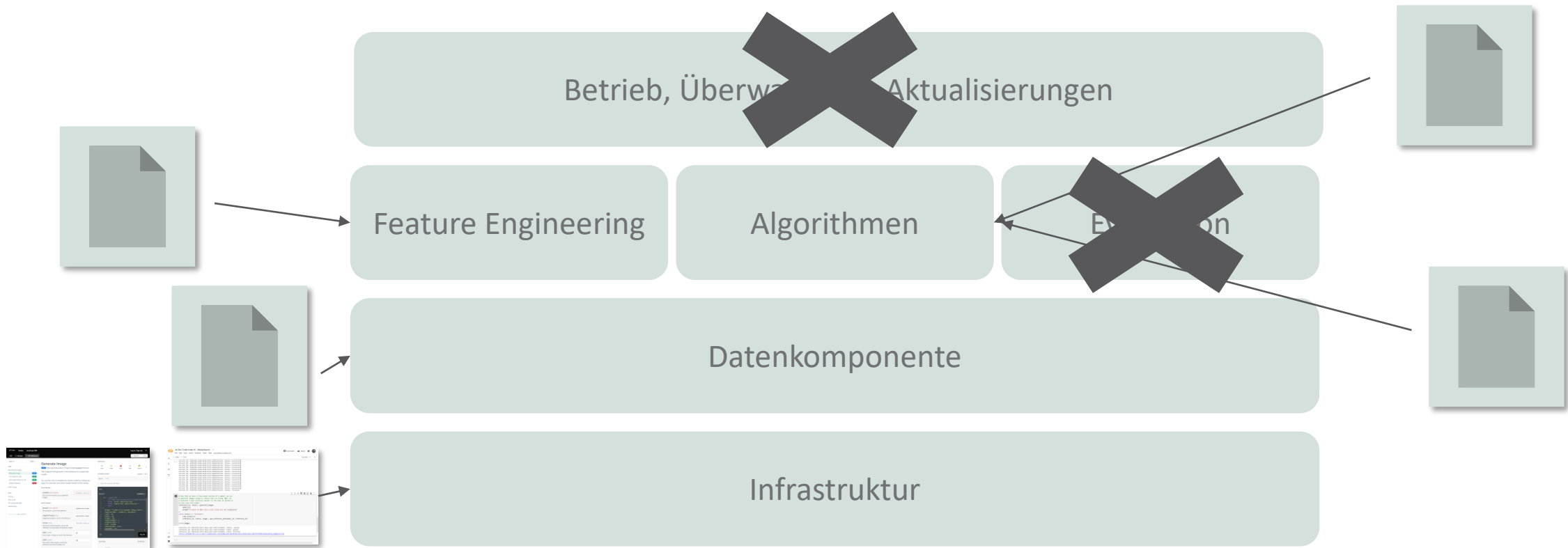
- 6. Sich selbst und die Gruppe überraschen lassen 😊

```
prompt="A photo of @me with a tall  
black hat and sunglasses"
```



Abschluss

Was haben wir heute gesehen?



In Anlehnung an Huyen (2022)

Literatur

- Chip Huyen: Designing Machine Learning Systems. O'Reilly 2022.
- Stuart Russel, Peter Norvig: Künstlicher Intelligenz. 3. aktualisierte Auflage. Pearson Studium 2012.