

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Кафедра суперкомпьютеров и общей информатики

Отчет по лабораторной работе №2

Дисциплина: «Инженерия данных»

Тема: «**Инференс и обучение НС**»

Выполнил: Неженский М.С.

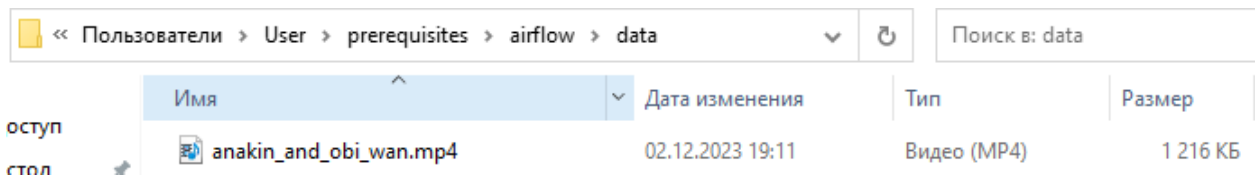
Группа: 6233-010402D

Самара 2023

1. Подготовка

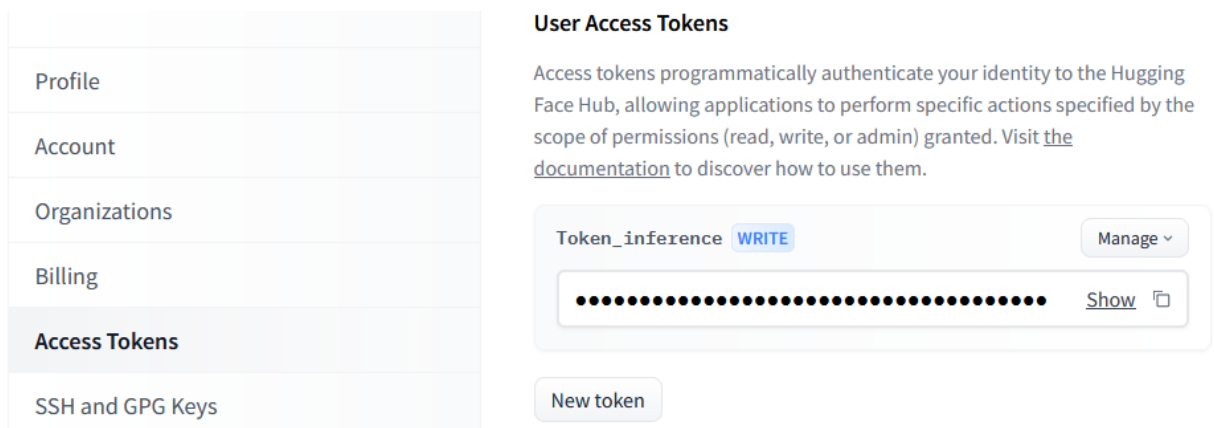
В рамках первого задания построен пайплайн, который реализует систему "Автоматического распознавания речи" для видеофайлов.

Сначала был найден видеоролик на просторах Youtube и скачан через специализированный сайт. Исходный файл можно найти в директории с названием



« Пользователи » User » prerequisites » airflow » data				
Поиск в: data				
	Имя	Дата изменения	Тип	Размер
оступ стол	anakin_and_obi_wan.mp4	02.12.2023 19:11	Видео (MP4)	1 216 КБ

Далее стоит зарегистрироваться на сайте <https://api-inference.huggingface.co> для получения токена. Проходим тест с машинкой. Подтверждаем электронную почту, письмо будет на почте. Нажимаем на изображение профиля -> Settings -> Access Tokens как показано ниже.



Profile

Account

Organizations

Billing

Access Tokens

SSH and GPG Keys

User Access Tokens

Access tokens programmatically authenticate your identity to the Hugging Face Hub, allowing applications to perform specific actions specified by the scope of permissions (read, write, or admin) granted. Visit [the documentation](#) to discover how to use them.

Token_inference WRITE Manage

..... Show

New token

Нажимаем New token.

🔑 Create a new access token

×

Name

my_token

Role

write

Generate a token

Вписываем любое название и выбираем роль: write.

Далее копируем наш токен. Он нам понадобится позже.

Profile

Account

Organizations

Billing

Access Tokens

SSH and GPG Keys

User Access Tokens

Access tokens programmatically authenticate your identity to the Hugging Face Hub, allowing applications to perform specific actions specified by the scope of permissions (read, write, or admin) granted. Visit [the documentation](#) to discover how to use them.

Token_inference

WRITE

Manage

.....

Show

New token

Далее включаем контейнер airflow через docker – работа будет выполняться в нем. На следующем скрине видно, что включено два контейнера, но в данной работе нам нужен только airflow.

Containers

Images

Volumes

Dev Environments BETA

Docker Scout EARLY ACCESS

Learning center

Extensions

Add Extensions

Containers [Give feedback](#)

Container CPU usage ⓘ

NaN% / 100% (1 cores allocated)

Container memory usage ⓘ

0B / 0B

Show charts ▾

Search

Only show running containers

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	> airflow		Running (7/8)	193.62%		1 hour ago	■ : 🗑
<input type="checkbox"/>	> mflow		Running (3/4)	0.14%		1 hour ago	■ : 🗑
<input type="checkbox"/>	> postgresql		Exited	0%		4 days ago	▶ : 🗑
<input type="checkbox"/>	> elasticsearch		Exited	0%		2 days ago	▶ : 🗑
<input type="checkbox"/>	> nifi		Exited	0%		4 days ago	▶ : 🗑

Showing 5 items

RAM 0.00 GB CPU 0.00% Connected to Hub

v4.22.0

Важно! Нужно обязательно следить за тем, чтобы контейнер не отключился сам, так любит делать airflow-docker-проху. На рисунке он представлен пятым. Если “вдруг” он отключился, то надо нажать в данном окне на “плей”.

Containers

Images

Volumes

Dev Environments BETA

Docker Scout EARLY ACCESS

Learning center

Extensions

Add Extensions

< airflow

C:\Users\User\prerequisites

Open

▶

■

🗑

airflow-airflow-sc...

airflow-airflow-sche

Running

■

:

🗑

airflow-airflow-w...

airflow-airflow-webs

Running

■

:

🗑

airflow-airflow-w...

airflow-airflow-work

Running

■

:

🗑

airflow-airflow-tri...

airflow-airflow-trigg

Running

■

:

🗑

airflow-docker-pr...

docker:24-dind

Running

■

:

🗑

airflow-airflow-ini...

airflow-airflow-init

Exited

▶

:

🗑

airflow-postgres-1

postgres:13

Running

■

:

🗑

Search

🕒

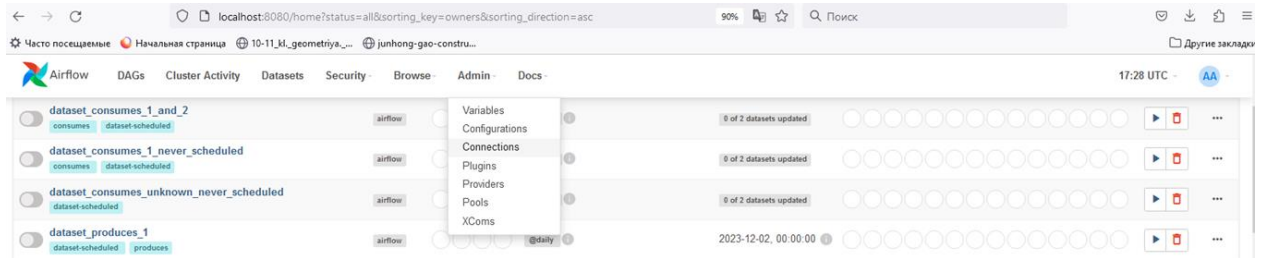
🗑

RAM 0.00 GB CPU 0.00% Connected to Hub

v4.22.0

4

Заходим по адресу <http://localhost:8080/> в airflow. Выбираем в верхней панели Admin -> Connections



Далее на этом моменте airflow у меня завис, поэтому картинки сделать не получилось. Нажимаем плюсики, название выбираем какое хотим, я выбрал connection_inference, тип выбираем – docker, url – tcp://docker-proxy, port – 2375 и нажимаем кнопку сохранить. Подготовка завершена!

2. Инференс и обучение НС

Для выполнения данного задания был создан DAG для Airflow и три файла sound_to_txt, txt_to_pdf и save_to_pdf. Код представлен ниже. В 24 строчке вписываем название подключения.

```

airflow > dags > first-task.py > ...
1  from datetime import datetime
2  from airflow import DAG
3  from docker.types import Mount
4  from airflow.providers.docker.operators.docker import DockerOperator
5  from airflow.sensors.filesystem import FileSensor
6
7  default_args = {
8      'owner': 'airflow',
9      'start_date': datetime(2023, 1, 1),
10     'retries': 1,
11 }
12
13 dag = DAG(
14     'translate_words_into_text',
15     default_args=default_args,
16     description='DAG , which allows you to translate words into text',
17     schedule_interval=None,
18 )
19
20 monitoring = FileSensor(
21     task_id='monitoring',
22     poke_interval=20,
23     filepath='/opt/airflow/data',
24     fs_conn_id='connection_inference',
25     dag=dag,
26 )
27
28 converting_audio_mp4_to_aac = DockerOperator(
29     task_id='converting_audio_mp4_to_aac',
30     image='jrottenberg/ffmpeg',
31     command='-i /data/anakin_and_obi_wan.mp4 -vn -acodec copy /data/received_video.aac',
32     mounts=[Mount(source='/data', target='/data', type='bind')],
33     docker_url="tcp://docker-proxy:2375",
34     dag=dag,
35 )

```

```

36
37 converting_audio_to_text = DockerOperator(
38     task_id='converting_audio_to_text',
39     image='nyurik/alpine-python3-requests',
40     command='python /data/sound_to_txt.py',
41     mounts=[Mount(source='/data', target='/data', type='bind')],
42     docker_url="tcp://docker-proxy:2375",
43     dag=dag,
44 )
45
46 converting_text_to_pdf = DockerOperator(
47     task_id='converting_text_to_pdf',
48     image='nyurik/alpine-python3-requests',
49     command='python /data/txt_to_pdf.py',
50     mounts=[Mount(source='/data', target='/data', type='bind')],
51     docker_url="tcp://docker-proxy:2375",
52     dag=dag,
53 )
54
55 save_to_pdf = DockerOperator(
56     task_id='save_to_pdf',
57     image='sasha151299/my_pdf:1.0',
58     command='python /data/save_to_pdf.py',
59     mounts=[Mount(source='/data', target='/data', type='bind')],
60     docker_url="tcp://docker-proxy:2375",
61     dag=dag,
62 )
63
64 monitoring >> converting_audio_mp4_to_aac >> converting_audio_to_text >> converting_text_to_pdf >> save_to_pdf

```

Тут вписываем наш сохраненный токен.

```

airflow > data > sound_to_txt.py > ...
1  import requests
2  API_URL = "https://api-inference.huggingface.co/models/openai/whisper-small"
3  API_TOKEN = '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'
4  headers = {"Authorization": f"Bearer {API_TOKEN}"}
5
6  with open('/data/received_video.aac', "rb") as f:
7      data = f.read()
8      response = requests.post(API_URL, headers=headers, data=data)
9      result = response.json()
10     text_file = open("/data/text.txt", "w+")
11     text_file.write(result['text'])
12     text_file.close()

```

Подготовка данных к переводу в pdf. Сюда тоже вписываем токен.

```

airflow > data > txt_to_pdf.py > ...
1  import requests
2  API_URL = "https://api-inference.huggingface.co/models/slauw87/bart_summarisation"
3  API_TOKEN = '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'
4  headers = {"Authorization": f"Bearer {API_TOKEN}"}
5
6  with open('/data/text.txt', "rb") as f:
7      data = f.read()
8      response = requests.post(API_URL, headers=headers, json={'inputs': f"{data}"})
9      result = response.json()
10     text_file = open("/data/summ.txt", "w+")
11     text_file.write(result[0]['summary_text'])
12     text_file.close()

```

Сохранение в pdf.

```

airflow > data > save_to_pdf.py > ...
1  from fpdf import FPDF
2
3  file = open("/data/summ.txt","r")
4  pdf = FPDF()
5  pdf.add_page()
6  for text in file:
7      pdf.set_font("Arial", size=15)
8      pdf.multi_cell(0, 5, txt=text, align="J")
9      pdf.output("/data/output.pdf")

```

В данной работе были использованы различные image. Все файлы кроме DAG лежат в папке data, а сам DAG в папке dags. И все это внутри папки airflow.

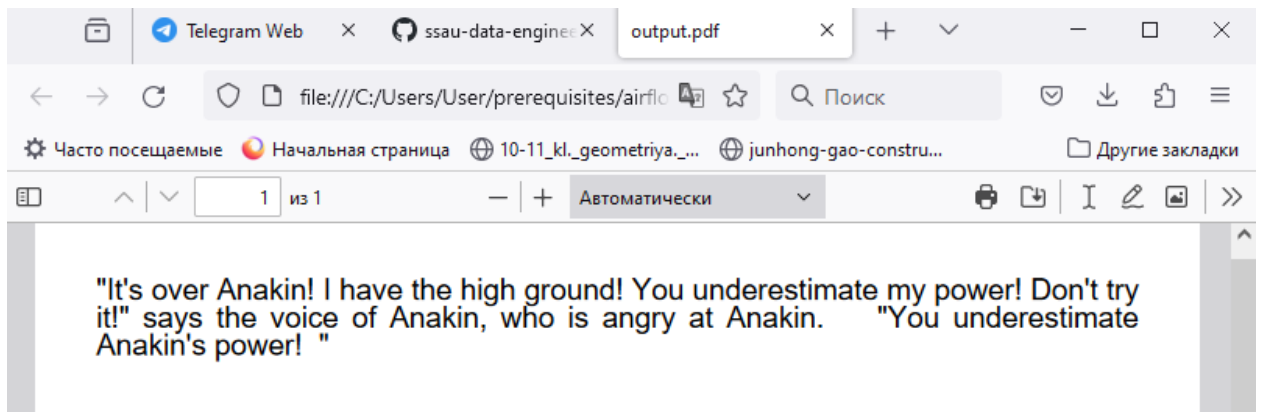
Запущенный процесс в airflow:

<input type="checkbox"/>	tutorial_taskflow_api_virtualenv	airflow	...	None	
<input checked="" type="checkbox"/>	translate_words_into_text	airflow	...	None	2023-12-03, 15:45:31
<input type="checkbox"/>	example_dynamic_task_mapping	airflow	...	1 day, 0:00:00	
<input type="checkbox"/>	example_dynamic_task_mapping_with_no_taskflow_operators	airflow	...	1 day, 0:00:00	
<input type="checkbox"/>	example_sla_dag	airflow	...	* / 2 * * * *	
<input type="checkbox"/>	validation_model	airflow	...	@hourly	

Папка с результатами:

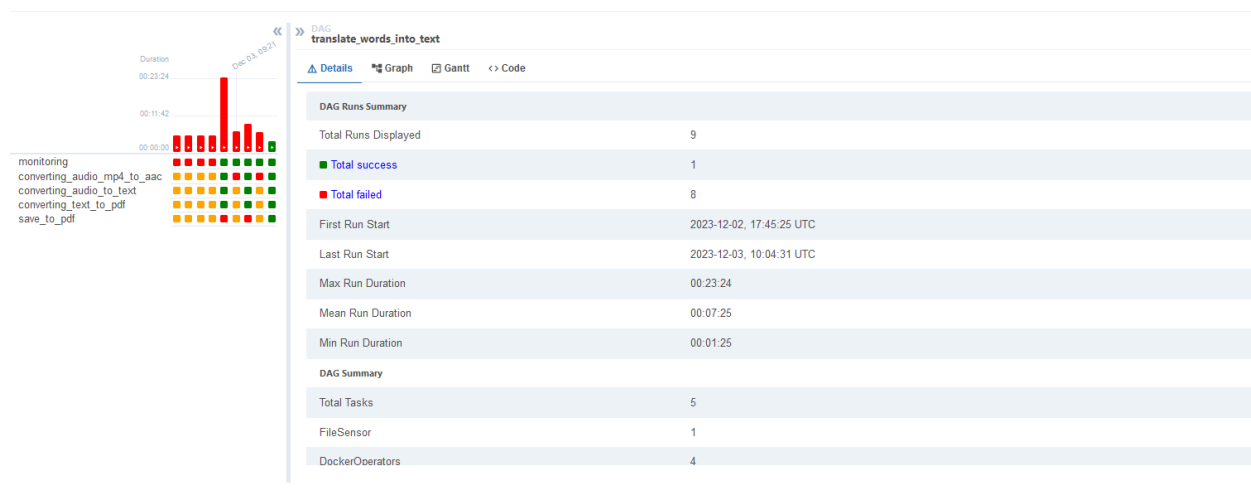
ватели > User > prerequisites > airflow > data				Поиск в: data
Имя	Дата изменения	Тип	Размер	
anakin_and_obi_wan.mp4	02.12.2023 19:11	Видео (MP4)	1 216 КБ	
mnist_test.csv	03.12.2023 0:23	Файл Microsoft O...	17 875 КБ	
mnist_train.csv	03.12.2023 0:23	Файл Microsoft O...	107 071 КБ	
output.pdf	03.12.2023 14:05	Firefox PDF Docu...	2 КБ	
popitka.py	03.12.2023 1:30	Python File	1 КБ	
read_data.py	03.12.2023 12:38	Python File	1 КБ	
received_video.aac	03.12.2023 14:05	Звук (ADTS)	175 КБ	
save_to_pdf.py	03.12.2023 13:57	Python File	1 КБ	
sound_to_txt.py	02.12.2023 22:27	Python File	1 КБ	
summ.txt	03.12.2023 14:05	Текстовый докум...	1 КБ	
text.txt	03.12.2023 14:05	Текстовый докум...	1 КБ	
train_data.py	03.12.2023 13:25	Python File	2 КБ	
txt_to_pdf.py	03.12.2023 13:57	Python File	1 КБ	

Результат в пдф формате:



Важно! Перед запуском программы удалять все сгенерированные файлы из папки data, иначе будет ошибка.

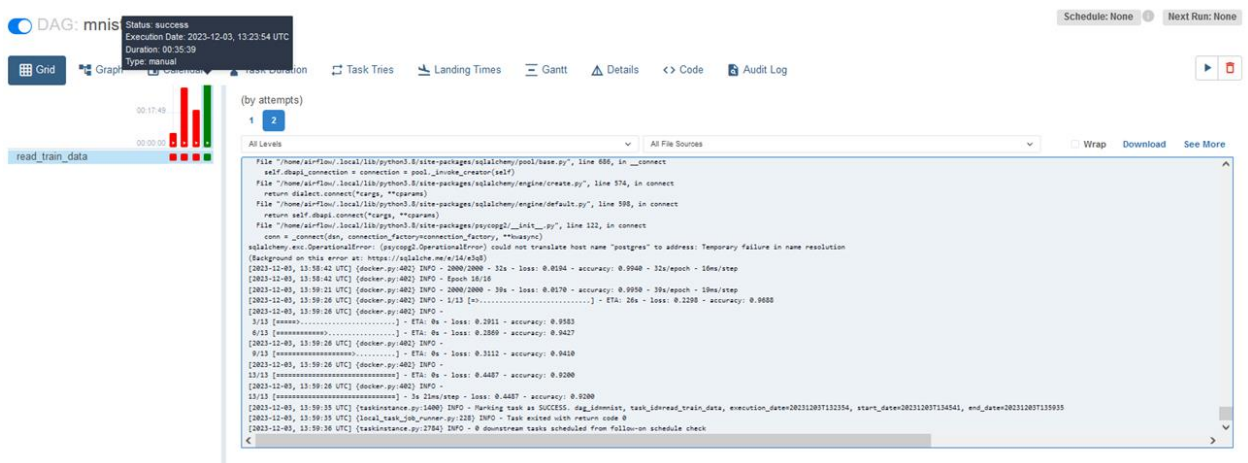
Результат в airflow:



3. Пайплайн для обучения модели

В рамках данного задания предлагается построен пайплайн, который реализует систему автоматического обучения нейросетевой модели. Обучение проводилось на наборе данных MNIST и заняло целых 35 минут для 16 эпох.

Результат в airflow:



Код DAG:

```
airflow > dags > second-task.py > ...
1  from datetime import datetime
2  from airflow import DAG
3  from docker.types import Mount
4  from airflow.providers.docker.operators.docker import DockerOperator
5
6  default_args = {
7      'owner': 'airflow',
8      'start_date': datetime(2023, 1, 1),
9      'retries': 1,
10 }
11
12 dag = DAG(
13     'mnist',
14     default_args=default_args,
15     description='mnist',
16     schedule_interval=None,
17 )
18
19 read_train_data = DockerOperator(
20     task_id='read_train_data',
21     image='sasha151299/second_pipeline:1.0',
22     command='python /data/read_train_data.py',
23     mounts=[Mount(source='/data', target='/data', type='bind')],
24     docker_url="tcp://docker-proxy:2375",
25     dag=dag,
26 )
27
28 read_train_data
```

Код read_train_data.py. В данной коде есть csv_logger, который сохраняет логи программы. Так как компьютер слабый, то пришлось ограничить

количество эпох обучения и количество данных для обучения.

Представленная нейронная модель с четырьмя нейронами.

```
airflow > data > read_train_data.py > read_train_data
1  import pandas as pd
2  import tensorflow as tf
3  import numpy as np
4
5  csv_logger = [tf.keras.callbacks.CSVLogger('/data/log.csv', append=True, separator=';')]
6
7  def read_train_data():
8      (X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()
9      X_train = X_train[:2000]
10     y_train = y_train[:2000]
11     X_test = X_test[:400]
12     y_test = y_test[:400]
13     X_train = X_train / 255
14     X_test = X_test / 255
15     X_train_flat = X_train.reshape(len(X_train), (28 * 28))
16     X_test_flat = X_test.reshape(len(X_test), (28 * 28))
17
18     model = tf.keras.Sequential([
19         tf.keras.layers.Dense(128, input_shape = (784,), activation = 'relu'),
20         tf.keras.layers.Dense(64, activation = 'sigmoid'),
21         tf.keras.layers.Dense(32, activation = 'sigmoid'),
22         tf.keras.layers.Dense(10, activation = 'softmax'),
23     ])
24
25     model.compile(
26         optimizer = 'adam',
27         loss = "sparse_categorical_crossentropy",
28         metrics=['accuracy']
29     )
30
31     logs = model.fit(X_train_flat, y_train, epochs=16, batch_size=1, verbose=2, callbacks=csv_logger)
32     model.evaluate(X_test_flat, y_test)
33 read_train_data()
```

Логги программы:

log.csv - Microsoft Excel

	A1	epoch												
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	epoch	accuracy	loss	val_acc	val_loss									
2	0	0.6909999	1.2185850	NA	NA									
3	1	0.9039999	0.4049921	NA	NA									
4	2	0.9409999	0.2346139	NA	NA									
5	3	0.9599999	0.1579155	NA	NA									
6	0	0.6740000	1.2631888	NA	NA									
7	1	0.8995000	0.4446087	NA	NA									
8	2	0.9365000	0.2505277	NA	NA									
9	3	0.9484999	0.1817753	NA	NA									
10	4	0.9725000	0.1091661	NA	NA									
11	5	0.9704999	0.0988163	NA	NA									
12	6	0.9865000	0.0541762	NA	NA									
13	7	0.9909999	0.0398843	NA	NA									
14	8	0.9909999	0.0342937	NA	NA									
15	9	0.9950000	0.0184408	NA	NA									
16	10	0.9925000	0.0298643	NA	NA									
17	11	0.9934999	0.0262407	NA	NA									
18	12	0.9975000	0.0105804	NA	NA									
19	13	0.9950000	0.0159599	NA	NA									

Выполнение данного кода в Google Colab:

second_task.ipynb

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменения сохранены

Комментировать Поделиться

Файлы

- sample_data
- log.csv

```

1 #!python3
2 import pandas as pd
3 import tensorflow as tf
4 import numpy as np
5 logger = [tf.keras.callbacks.CSVLogger('log.csv', append=True, separator=';')]
6
7 def train_data():
8     (X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()
9     X_train = X_train[:2000]
10    y_train = y_train[:2000]
11    X_test = X_test[:400]
12    y_test = y_test[:400]
13    X_train = X_train / 255
14    X_test = X_test / 255
15    X_train_flat = X_train.reshape(len(X_train), (28 * 28))
16    X_test_flat = X_test.reshape(len(X_test), (28 * 28))
17
18    model = tf.keras.Sequential([
19        tf.keras.layers.Dense(128, input_shape = (784,), activation = 'relu'),
20        tf.keras.layers.Dense(64, activation = 'sigmoid'),
21        tf.keras.layers.Dense(32, activation = 'sigmoid'),
22        tf.keras.layers.Dense(10, activation = 'softmax'),
23    ])
24
25    model.compile(
26        optimizer = 'adam',
27        loss = "sparse_categorical_crossentropy",
28        metrics=['accuracy']
29    )
30
31    logs = model.fit(X_train_flat, y_train, epochs=10, callbacks=csv_logger)

```

log.csv

1 to 10 of 20 entries

epoch;accuracy;loss;val_accuracy;val_loss
0.0;6880000233650208;1.2060576677322388;NA;NA
1.0;9054999947547913;0.4196309745311737;NA;NA
2.0;9380000233650208;0.2504310905933338;NA;NA
3.0;9520000219345093;0.17086486518383026;NA;NA
4.0;973999977118164;0.10720445960760117;NA;NA
5.0;9819999933242798;0.0748123899102211;NA;NA
6.0;9819999933242798;0.06371814757585626;NA;NA
7.0;9884999990463257;0.04726141318678856;NA;NA
8.0;9919999837875366;0.03248242661356926;NA;NA
9.0;9965000152587891;0.02443413808941841;NA;NA

Show 10 per page

6 сек. выполнено в 15:27

```
Epoch 1/10
63/63 [=====] - 1s 4ms/step - loss: 2.1516 - accuracy: 0.3100
Epoch 2/10
63/63 [=====] - 0s 4ms/step - loss: 1.6382 - accuracy: 0.6100
Epoch 3/10
63/63 [=====] - 0s 4ms/step - loss: 1.2582 - accuracy: 0.7740
Epoch 4/10
63/63 [=====] - 0s 6ms/step - loss: 0.9691 - accuracy: 0.8415
Epoch 5/10
63/63 [=====] - 0s 6ms/step - loss: 0.7539 - accuracy: 0.8935
Epoch 6/10
63/63 [=====] - 0s 7ms/step - loss: 0.5886 - accuracy: 0.9310
Epoch 7/10
63/63 [=====] - 0s 7ms/step - loss: 0.4663 - accuracy: 0.9450
Epoch 8/10
63/63 [=====] - 0s 6ms/step - loss: 0.3740 - accuracy: 0.9585
Epoch 9/10
63/63 [=====] - 0s 6ms/step - loss: 0.3011 - accuracy: 0.9635
Epoch 10/10
63/63 [=====] - 0s 7ms/step - loss: 0.2490 - accuracy: 0.9680
13/13 [=====] - 0s 2ms/step - loss: 0.3989 - accuracy: 0.8975
```

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы были получены навыки по работе с Docker, Apache Airflow. В данной работе были написаны программы для автоматического распознавания речи и обучения нейронной сети.