Андреев Александр 6233

Маленькое замечание: в репозитории отсутствует файл X_train.csv, т.к. он имеет размер 480 МБ, а GitHub позволяет загружать файлы размером до 25 МБ.

Первый pipeline

Работа началась стандартно с использованием рекомендованных image'ей преподавателем.

Для начала были настроены connections в airflow, далее был получен токен для записи на сайте Hugging Face, а впоследствии с использованием открытых images для докера были проведены выделение аудио из видео, его обработка и запись в pdf файл (пришлось написать свой image).

Выделение аудио из видео прошло успешно.

```
extract_audio = DockerOperator(
    task_id='extract_audio',
    image='jrottenberg/ffmpeg',
    command='-i /data/input_video.mp4 -vn -acodec copy /data/audio.aac',
    mounts=[Mount(source='/data', target='/data', type='bind')],
    docker_url="tcp://docker-proxy:2375",
    dag=dag,
)
```

Рисунок 1 – Выделение аудио из видео

Преобразование из аудио в текст прошло успешно лишь отчасти, так как по какой-то причине распознавание идет толи не полностью, толи пропускаются слова. Но я считаю, что это не проблема в рамках данной ЛР, ведь я должен использовать сторонний инструмент, а не написать свой. Поэтому имеем то, что имеем – распознается лишь часть текста, но она в видео действительно есть, это не случайный набор слов.

Скрипты, которые используются на скрине ниже представлены в репозитории.

```
transform_audio_to_text = DockerOperator(
    task_id='transform_audio_to_text',
    image='nyurik/alpine-python3-requests',
    command='python /data/audio_to_text.py',
    mounts=[Mount(source='/data', target='/data', type='bind')],
    docker_url="tcp://docker-proxy:2375",
    dag=dag,
)

summarize_text = DockerOperator(
    task_id='summarize_text',
    image='nyurik/alpine-python3-requests',
    command='python /data/text_to_summ.py',
    mounts=[Mount(source='/data', target='/data', type='bind')],
    docker_url="tcp://docker-proxy:2375",
    dag=dag,
)
```

Рисунок 2 – Преобразование аудио в текст

Далее была проблема с преобразованием текста в pdf. Стандартные средства, которые были использовались в этом коде для преобразования в формат pdf не работали.

```
save_to_pdf = DockerOperator(
    task_id='save_to_pdf',
    image='mashupmill/text2pdf',
    command='text2pdf /data/summ.txt > /data/summ.pdf',
    mounts=[Mount(source='/data', target='/data', type='bind')],
    docker_url="tcp://docker-proxy:2375",
    dag=dag,
)
```

Рисунок 3 – Неработающий код преобразования в pdf

Чтобы это исправить был написан свой скрипт с использованием библиотеки fpdf и создан свой образ (так как мне не удалось быстро найти образ на DockerHub с fpdf библиотекой) для решения этой задачи, который также используется в DAG файле.

C: > Users > sasha > Desktop > Mara > 2 год > DE > Prerequisites > airflow > data > save_to_pdf.py > ...

```
from fpdf import FPDF

file = open("/data/summ.txt","r")
pdf = FPDF()
pdf.add_page()
for text in file:
    if len(text) <= 20:
        pdf.set_font("Arial","B",size=18)
        pdf .cell(w=200,h=10, txt=text, Ln=1,align="C")
    else:
        pdf.set_font("Arial",size=15)
        pdf .multi_cell(w=0,h=10, txt=text,align="L")
    pdf. output ( "/data/output.pdf")
```

Рисунок 4 – Скрипт для преобразования файла в pdf

Dockerfile – Блокнот

Файл Правка Формат Вид Справка

```
#Deriving the latest base image
FROM python:latest

RUN pip install numpy fpdf

#Labels as key value pair
LABEL Maintainer="sasha151299.fpdf"


# Any working directory can be chosen as per choice like '/' or '/home' etc
# i have chosen /usr/app/src
WORKDIR /usr/app/src
```

Рисунок 5 – Dockerfile для создание своего image файла с библиотекой fpdf

```
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  wait_for_new_file >> extract_audio >> transform_audio_to_text >> summarize_text >> save_to_pdf
```

Рисунок 6 – Использование скрипта в dag файле

Итого мы имеем запущенный файл в airflow и несколько файлов в входной-выходной папке (в конце все файлы, кроме output.pdf можно было бы удалить, но я этого не делал).

Скрин со всеми файлами в папке будет приложен в самом конце отчета, так как в папке есть скрипты и файла для второй части ЛР.

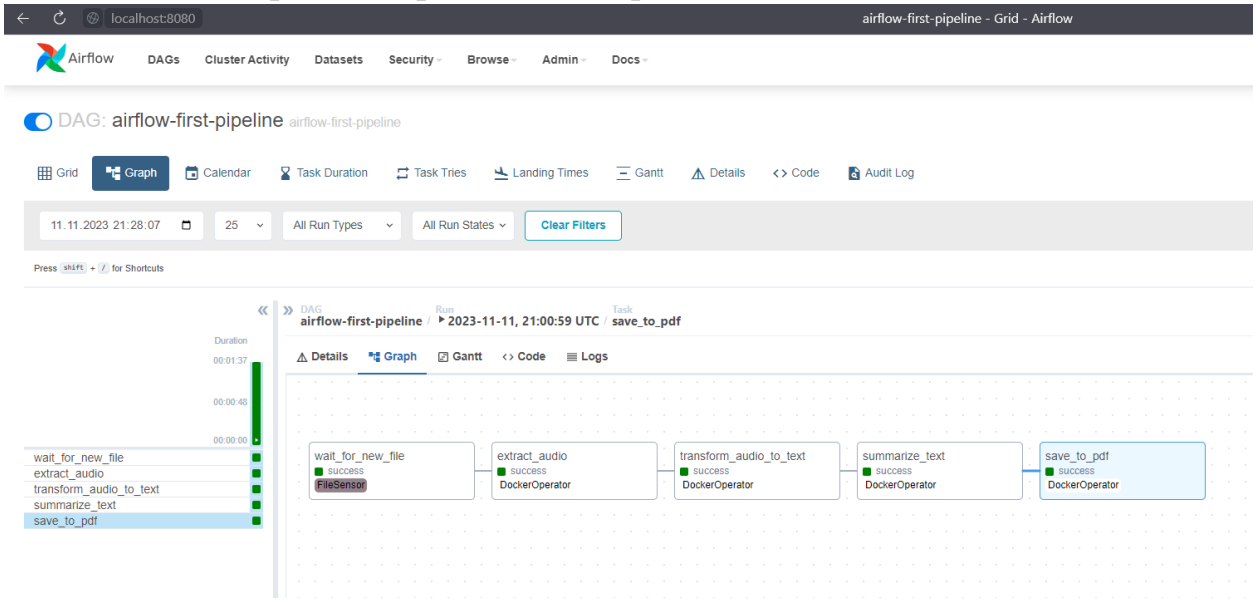

Рисунок 7 – Запущенный DAG в airflow

Второй pipeline

В качестве решения второй задачи нам не предлагается никакой датасет и так как моя НИР не связана с сетями мною был выбран код с другого предмета этого семестра, где мы сами писали простую нейронную сеть с использованием TensorFlow и пытались ее обучить на двух датасетах (я обучал ее на mnist). Коды для чтения данных и обучения приложены в репозитории.

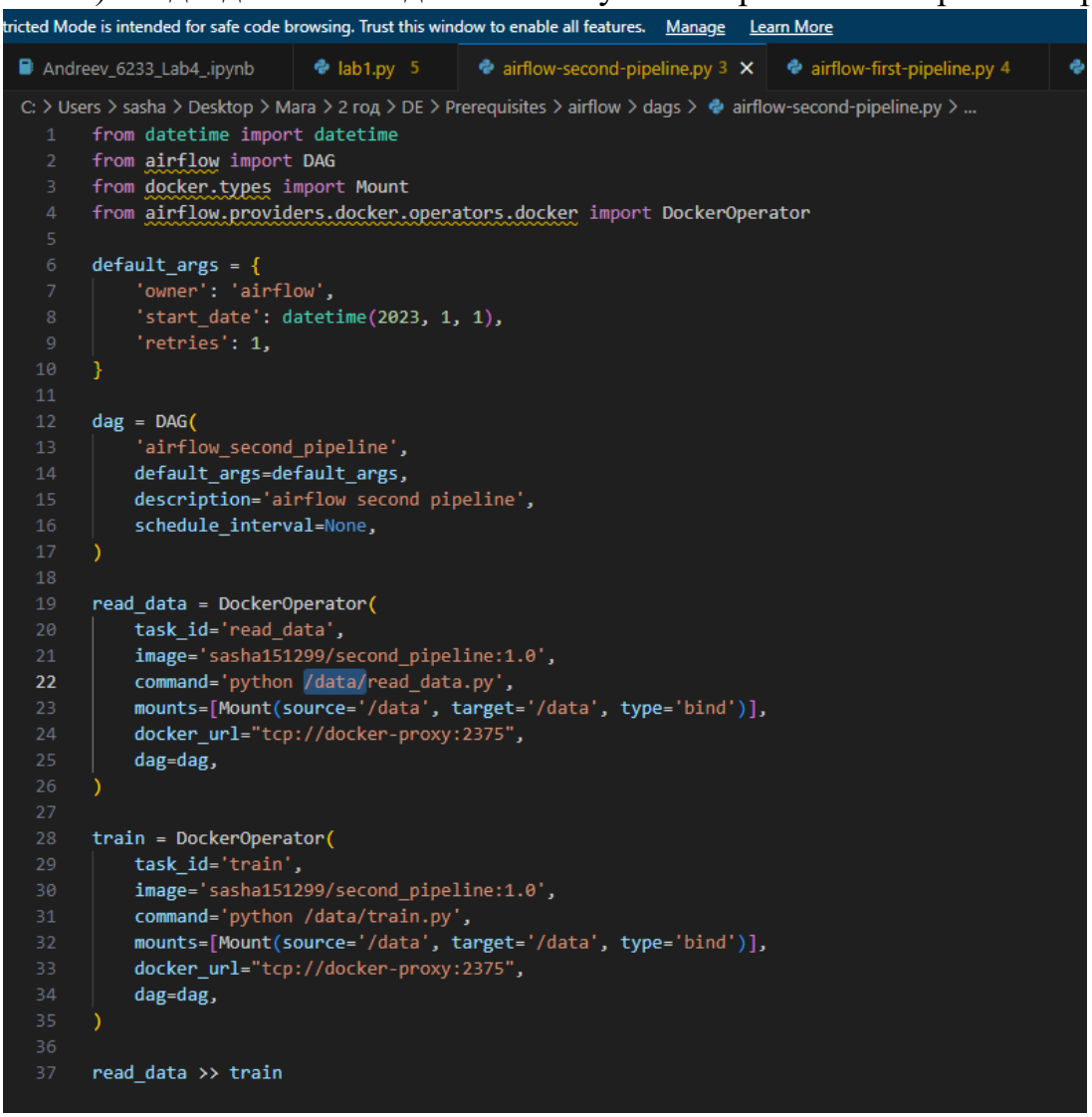

Рисунок 8 – Код DAG файла для второй части работы

Для этой части работы на просторах DockerHub мне не удалось образ, который содержал бы в себе TensorFlow, pandas и numpy, поэтому я снова сделал свой образ (использовал следующую инструкцию https://linuxhint.com/push-and-pull-docker-image-to-docker-hub/#b).

Dockerfile – Блокнот

Файл Правка Формат Вид Справка

```
#Deriving the latest base image
FROM tensorflow/tensorflow:latest

RUN pip install numpy pandas

#Labels as key value pair
LABEL Maintainer="sasha151299.second_pipeline"


# Any working directory can be chosen as per choice like '/' or '/home' etc
# i have chosen /usr/app/src
WORKDIR /usr/app/src
```

Рисунок 9 – Dockerfile для второй части работы

Удивительно, но в этой части лабораторной работы проблем не возникло (кроме нейминга файлов – failed запуски). DAG файл также успешно запустился и отработал.

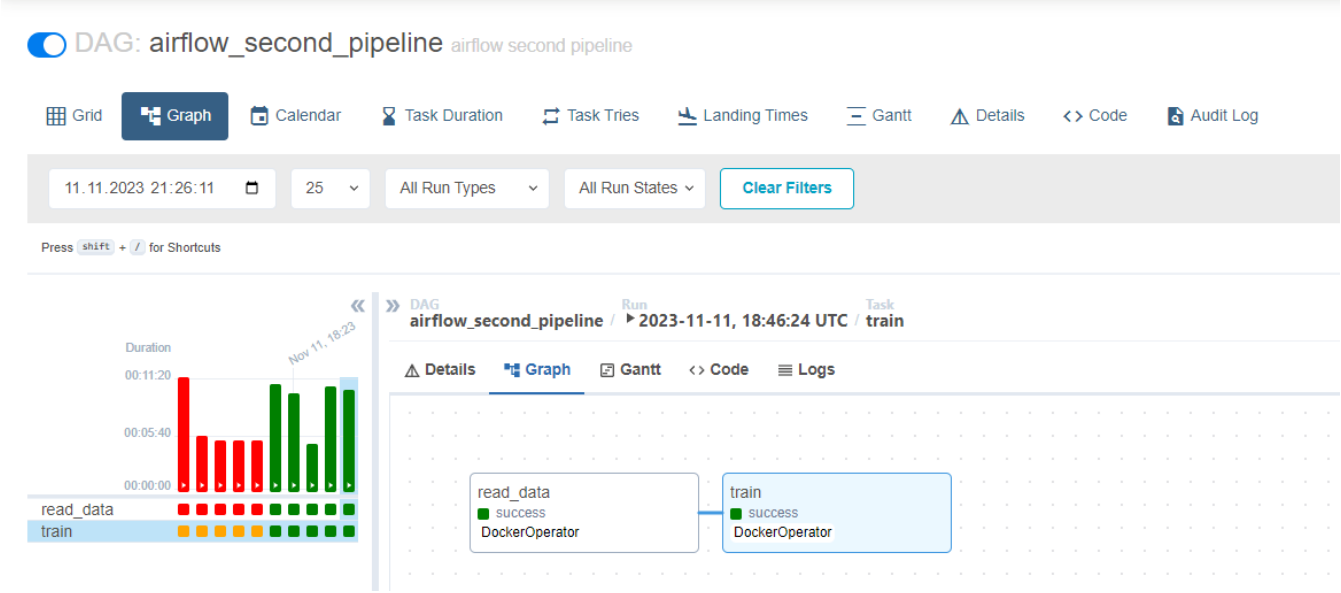

Рисунок 10 – Запущенный DAG в airflow для обучения нейронной сети

Результаты обучения нейронной сети – accuracy на тренировочной и валидационной выборках и соответствующие loss представлены в выходном файле после обучения (как требовалось в лабораторной работе).

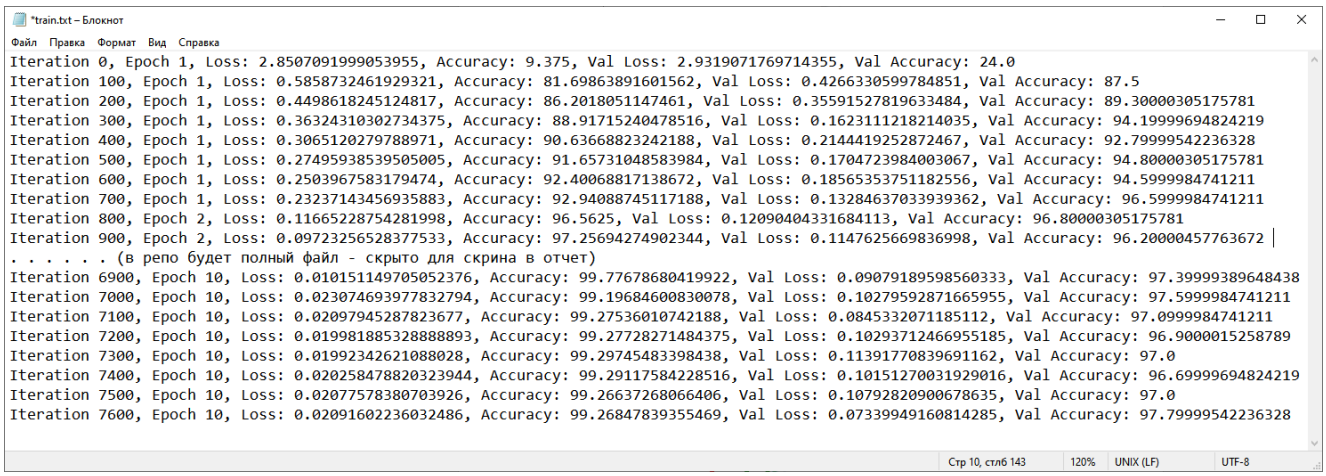

Рисунок 11 – Результаты обучения нейронной сети

По итогу выполнения лабораторной работы имеется два созданных образа, два созданных и отработавших DAG файла и несколько промежуточных и финальных файлов в папке.

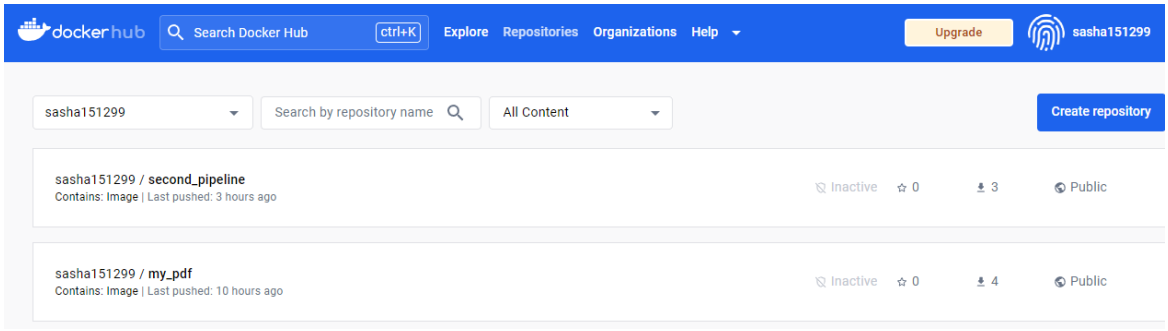

Рисунок 12 – Images для лабораторной работы

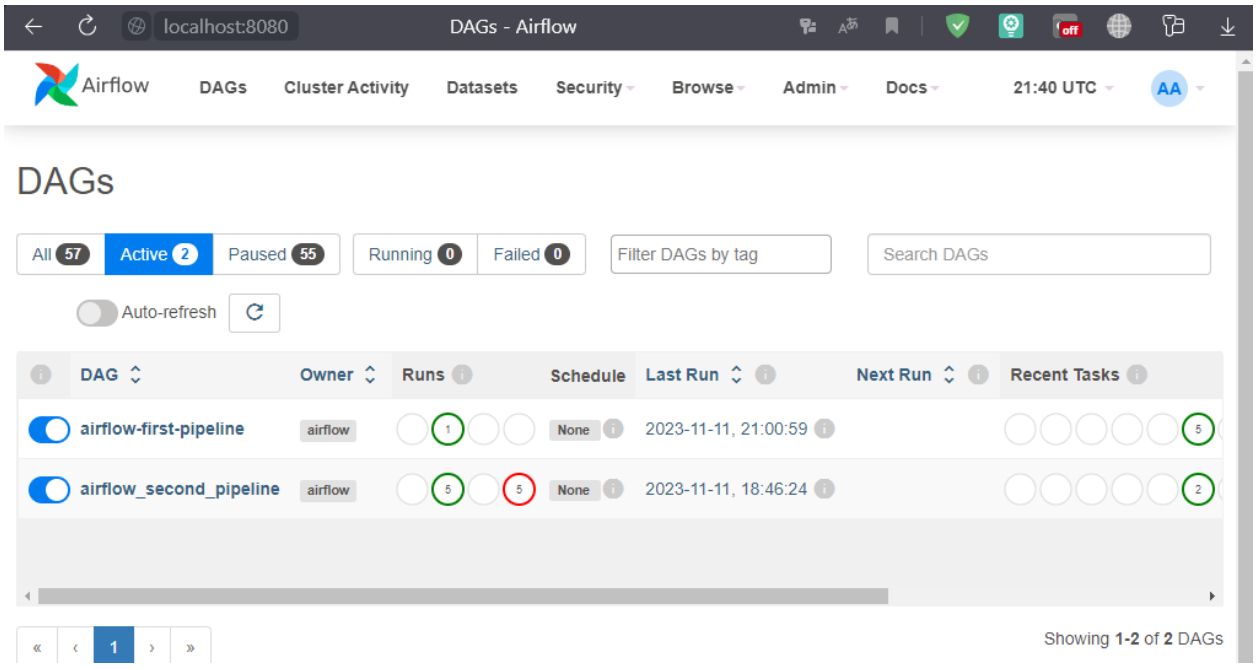

Рисунок 13 – DAG файлы в airflow

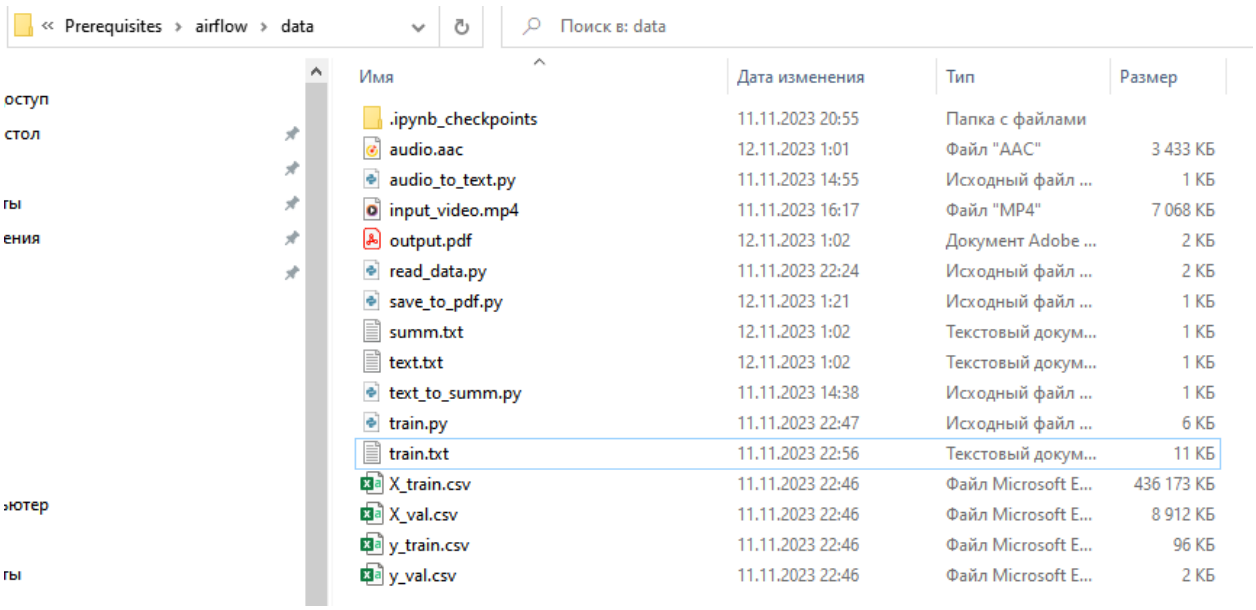

Рисунок 17 – Все файлы кроме DAG для ЛР