Хакатон

биометрия

Кейс_2, Сборная команда №4

Описание задачи

Описание: С развитием технологий машинного обучения, особенно искусственных нейронных сетей, появились инструменты, позволяющие создавать реалистичные поддельные изображения, видео и записи голоса. Дипфейки, могут использоваться не только для дезинформации, манипуляции мнением общественности, но и для мошеннических действий в биометрических системах. Важной становится задача распознавания дипфейков с целью недопущения мошеннических операций, краж данных и взломов систем.

Цель: Разработать инструмент, способный определять поддельные (сгенерированные с помощью AI) видеоизображения с высокой точностью. Сервис должен включать в себя алгоритм для обнаружения видео-дипфейков с изображением лица человека.

Стеки: На язык программирования ограничений нет. [SEP] Дистрибутив должен соответствовать разделу 4 Методических рекомендаций по подключению биометрических процессоров к Единой биометрической системе (далее — МР), API должен соответствовать приложению Б МР.

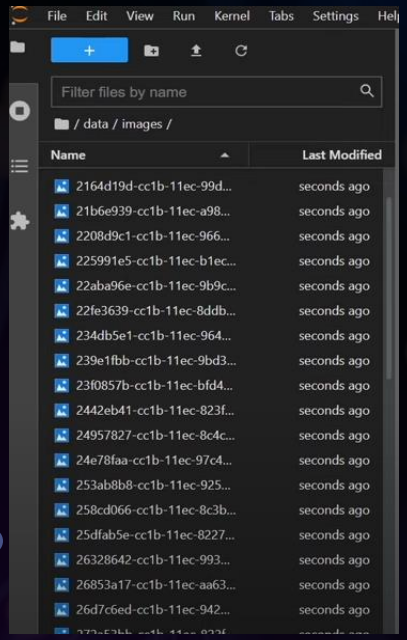
Сбор дипфейков

Для того, чтобы собрать различные дипфейки, я воспользовалась инструментами tensorflow-gpu, opencv.

```
[39]: !pip install labelme tensorflow tensorflow-gpu opencv-python matplotlib albumentations
```

1.2 Собирайте изображения с помощью OpenCV

```
[32]: import os
      import time
      import uuid
      import cv2

[33]: uuid.uuid1()

[33]: UUID('ea9843a8-80c2-11ee-b8df-3c7c3f2fc724')

[34]: IMAGES_PATH = os.path.join('data','images')
      number_images = 30

[38]: cap = cv2.VideoCapture(1)
      for imgnum in range(number_images):
          print('Collecting image {}'.format(imgnum))
          ret, frame = cap.read()
          imgname = os.path.join(IMAGES_PATH,f'{str(uuid.uuid1())}.jpg')
          cv2.imwrite(imgname, frame)
          cv2.imshow('frame', frame)
          time.sleep(0.5)

          if cv2.waitKey(1) & 0xFF == ord('q'):
              break
      cap.release()
      cv2.destroyAllWindows()
```



Просмотр набора данных и создание функции загрузки изображений

2.1 Импорт TF и Deps

```
import tensorflow as tf
import json
import numpy as np
from matplotlib import pyplot as plt
```

2.2 Ограничьте рост памяти графического процессора

```
# Избегайте ошибок OOM, установив увеличение потребления памяти графическим процессором
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
```

```
tf.config.list_physical_devices('GPU')
```

2.3 Загрузить изображение в конвейер передачи данных TF

```
images = tf.data.Dataset.list_files('data\\images\\*.jpg')
```

```
images.as_numpy_iterator().next()
```

```
def load_image(x):
    byte_img = tf.io.read_file(x)
    img = tf.io.decode_jpeg(byte_img)
    return img
```

```
images = images.map(load_image)
```

```
images.as_numpy_iterator().next()
```

```
type(images)
```

2.4 Просмотр необработанных изображений с помощью Matplotlib

```
image_generator = images.batch(4).as_numpy_iterator()
```

```
plot_images = image_generator.next()
```

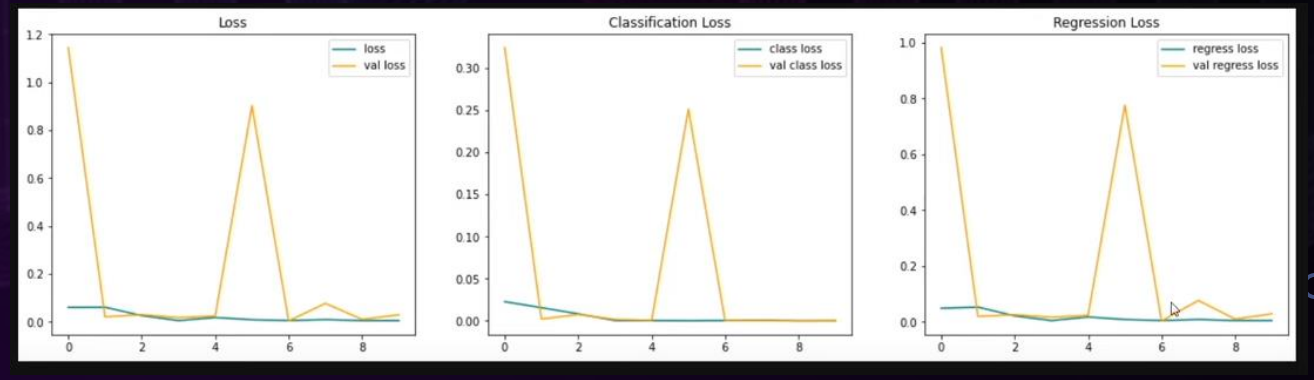
```
fig, ax = plt.subplots(ncols=4, figsize=(20,20))
for idx, image in enumerate(plot_images):
    ax[idx].imshow(image)
plt.show()
```

Обучение нейронной сети

10.3 Создание Plot Performance

```
]: hist.history
```

```
]: fig, ax = plt.subplots(ncols=3, figsize=(20,5))

ax[0].plot(hist.history['total_loss'], color='teal', label='loss')
ax[0].plot(hist.history['val_total_loss'], color='orange', label='val loss')
ax[0].title.set_text('Loss')
ax[0].legend()

ax[1].plot(hist.history['class_loss'], color='teal', label='class loss')
ax[1].plot(hist.history['val_class_loss'], color='orange', label='val class loss')
ax[1].title.set_text('Classification Loss')
ax[1].legend()

ax[2].plot(hist.history['regress_loss'], color='teal', label='regress loss')
ax[2].plot(hist.history['val_regress_loss'], color='orange', label='val regress loss')
ax[2].title.set_text('Regression Loss')
ax[2].legend()

plt.show()
```



Прогнозирование

```
Epoch 7/10
100/100 [==============================] - 9s 58ms/step - total_loss: 0.1014 - class_loss: 0.0338 - regress_loss: 0.0845 - val_tot
al_loss: 0.0413 - val_class_loss: 0.0059 - val_regress_loss: 0.0384
Epoch 8/10
100/100 [==============================] - 9s 58ms/step - total_loss: 0.0549 - class_loss: 0.0168 - regress_loss: 0.0465 - val_tot
al_loss: 0.0420 - val_class_loss: 0.0018 - val_regress_loss: 0.0411
Epoch 9/10
100/100 [==============================] - 9s 58ms/step - total_loss: 0.0518 - class_loss: 0.0140 - regress_loss: 0.0448 - val_tot
al_loss: 0.1082 - val_class_loss: 0.0412 - val_regress_loss: 0.0876
Epoch 10/10
```

Спасибо!

Кейс_2

Сборная команда №4