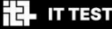



Документация может немного поменяться, ссылка на нее:

<https://docs.google.com/document/d/1O8OAleC5983xtY-U-2huoawDm4DjtXkQwxIJRuyKRjk/edit>



## Сервис по распознаванию уровня "надетости" защитных масок

Придумайте сервис, который будет определять по изображению насколько "правильно" человек носит маску, т.е. закрывает ли она лицо полностью, натянута на подбородок или не закрывает нос.



by Cucumber

github: <https://github.com/kirrya95/tulahack>

Мы подошли к решению задачи следующим образом: надо как-то определить лица и части лица (глаза, нос, рот, подбородок) и маски. Далее по взаимному расположению этих позиций можем дать ответ на поставленную задачу.

### 1. Определения масок и соответствующих им лиц.

Для начала мы поискали в открытом доступе, сталкивался ли кто-то с такой проблемой. Результаты нас немного удивили - никто ничего подобного не делал. Делать нечего, пришлось самим искать датасет, немного редактировать его, а затем обучать модель. Весь цикл обучения можно посмотреть в одном из ноутбуков на github. Сначала мы взяли самую легковесную модель из доступным - Faster R-CNN MobileNetV3-Large 320 FPN. Однако она показала плохие результаты и "отказалась" обучаться. Проблема в том, что ей выгоднее было выдавать пустые boxes.

Следующим шагом мы взяли модель посложнее - Faster R-CNN MobileNetV3-Large FPN. Это отличное сочетание хорошего качества, небольшой памяти и высокой производительности. После нескольких эпох обучения модель показывала отличные результаты.

Датасет для обучения:

<https://drive.google.com/file/d/1-0TjTRNHuiz5e2FIDqF4Uw6T4JVsXxMq/view>

Кроме того неплохим оказался датасет с kaggle

<https://www.kaggle.com/andrewmvd/face-mask-detection>

Однако ввиду его маленького размера было решено взять именно первый.

### 2. Определения частей лица.

Тут мы попробовали сразу несколько моделей: dlib,

[https://github.com/cunjian/pytorch\\_face\\_landmark](https://github.com/cunjian/pytorch_face_landmark)

<https://github.com/nicehuster/pytorch-facial-landmark>

**Документация может немного поменяться, ссылка на нее:**

<https://docs.google.com/document/d/1O8OAleC5983xtY-U-2huoawDm4DjtXkQwxIJRuyKRjk/edit>

<https://github.com/braindotai/Facial-Landmarks-Detection-Pytorch>

Однако, как ни странно, нам показалось, что dlib выигрывала по качеству все остальные, поэтому решили использовать именно её.

### **3. Объединение**

И теперь самое главное - как же объединять эти результаты в один? У нас было две идеи:

- Смотреть какие точки лежат в пределах квадрата маски. Например, если внутри квадрата с маской лежат 4 точки, относящиеся к носу, то тогда маска полностью надета. В противном случае она лежит либо на подбородке либо на её вообще нет. Аналогичным способом рассматриваются эти случаи.
- Мы заметили, что рот распознаётся хорошо, однако с детекцией носа зачастую бывают проблемы. Поэтому мы стали смотреть не на количество точек, а на длину носа, которая покрывается маской.

Затем мы объединили всё в один класс. Весь код можно посмотреть на github.