Данная статья не претендует на полноту. Господи, о чем говорить, если ее название даже не претендует на полноту. Полное название было бы таким:

Как натренировать SSDmobilenet на кошкособачьем датасете локально на CPU под Windows 10 в половине пятого утра понедельника, чтобы потом в голову пришла гениальная идея написать гайд в виде статьи на фикбуке, Карл.

Кроме полноты данная статья не претендует на полезность. Действительно, с чего бы вам понадобилось делать transfer learning, в полпятого утра. Под Windows. На кошкособачьем датасете. Не, ну я бы поняла, если бы это был датасет с енотами. Но кошкособачий…

Единственное, на что претендует мое творение – самоирония. Она тонкая, но ее можно заметить. В шапке. В рейтинге.

Спасибо за внимание, можете закрывать.

1. Установим tensorflow-master на машину.
2. Git clone ее в любое удобное место и качаем protoc.exe туда же.

Теперь делаем так:

# From tensorflow/models/research/

protoc object\_detection/protos/\*.proto --python\_out=.

Мне очень импонирует позиция разработчиков по этому поводу:

**If you are on linux: блаблабла**

**If you are on MacOS: ляляля**

**???????????**

**Ну а если вы на Windows, просто умрите ля, я так понимаю.**

**Не всегда под Windows команда выше нормально выполняется. Следует запустить protoc.exe отдельно для каждого прото-файла. Да. Я видела их количество. Можно написать скрипт, но…**

1. Импортируем (под Виндоус!) object-detection, если no module named оно самое:

SET PYTHONPATH=%cd%;%cd%\slim (и запускаем это из папки research, конечно)

1. Installing pycocotools for любимый (о, да) Windows:

pip install git+https://github.com/philferriere/cocoapi.git#subdirectory=PythonAPI

1. Исходные данные: кошкособачий датасет http://www.robots.ox.ac.uk/~vgg/data/pets/

Фотки jpg и png в images, ей соответствует annotations, в которой для каждой картинки существует индивидуальный xml файл с координатами bbox-ов и номерами классов.

Если хочется собрать свой, то кроме картинок, на которые у нас (о, боже) есть авторские права (либо лицензия разрешает их использовать для данных целей), предполагается, что у нас есть некоторая программа, которая при помощи специально обученного человека к сырым jpg файлам создает xml-ки.

1. Создадим LABEL MAP. Ой, ля, погуглим. Там все просто.

Для кошкособачьего датасета она уже создана и хранится в object-detection/data, поэтому сегодня руки мы не мараем.

1. Преобразуем это сырое датасетное мясо в удобоваримый для tf формат TFRecords. Разработчики приготовили специальный скрипт, лежащий в research. Конечно, чтобы использовать его с минимальными потерями, все должно лежать в тех путях, которые указали разработчики:

В директории tensorflow/models/research/ объекты вида

- images.tar.gz

- annotations.tar.gz

+ images/

+ annotations/

+ object\_detection/

... other files and directories

Отлично. Запускаем скрипт — вот так:

python object\_detection/dataset\_tools/create\_pet\_tf\_record.py --label\_map\_path=object\_detection/data/pet\_label\_map.pbtxt --data\_dir=. --output\_dir=.

Предполагается, что мы сидим в папке research. Сюда же созданные файлы и выгрузятся. Так как мы используем прекрасный и волшебный Windows 10, то линуксовское pwd мы заменяем на . (точечку).

Я думаю, если залезть в скрипт create\_pet\_tf\_record.py, его довольно просто отредактировать для кастомно собранного датасета. Но делать мы это, конечно, не будем.

1. Теперь пора создать чистую папку для обучения. Пусть это будет какая-нибудь a в object-detection. Хотя я не буду против, если вы назовете ее eyafjallajökull и поместите в центр своего чистейшего Desktop-а. Хотя все-таки eyafjallajökull я бы не советовала – кодировка, чтоб ее.

В этой классной папке делаем data – и копируем туда все TFRecord’ы и Label Map.

1. Вы же в курсе, что мы хотим transfer learning? Т.е. дообучаем предобученную модельку. Конкретно я брала SSD mobilenet из их набора. Вот ее и качаем, распаковываем и берем оттуда все, что содержит загадочные ckpt – чекпойнты – берем и копируем в нашу dat’у.

Все, жить можно.

1. Идем из даты в обратно в «a» – и создаем в ней models. А в ней уже еще три – model, train и eval.

Последние две оставляем пустыми – они заполнятся сами в процессе, а вот в первую положим кое-что. Конфиг модельки.

В config’е хранится вся-вся важная информация (кхе-хе, если реально интересно – никто не запрещает его открыть и почитать. Спойлеры!). Берем его из папки скачанной модели, копируем сюда и открывает через какой-нибудь Notepad++.

Тут есть PATH\_TO\_BE\_CONFIGURED, которую мы, собственно, и делаем be configured. Нужен просто абсолютный путь к папке data вместо этого замечательного слова. В пяти местах.

Но раз мы сидим под Windows и не хотим проблем с кодировкой tf, все \ меняем на /. И радуемся, радуемся жизни.

1. Время запуска!

Есть два пути – воспользоваться старым train.py, упоминание которого встречается во всех туториалах до 2018 года, перенеся его из legacy в object-detection. Либо же model\_main.py, который сразу и тренит, и исполняет, а лежит уже здесь – в object-detection.

PIPELINE\_CONFIG\_PATH={path to pipeline config file}

MODEL\_DIR={path to model directory}

NUM\_TRAIN\_STEPS=50000

SAMPLE\_1\_OF\_N\_EVAL\_EXAMPLES=1

python object\_detection/model\_main.py \

--pipeline\_config\_path=${PIPELINE\_CONFIG\_PATH} \

--model\_dir=${MODEL\_DIR} \

--num\_train\_steps=${NUM\_TRAIN\_STEPS} \

--sample\_1\_of\_n\_eval\_examples=$SAMPLE\_1\_OF\_N\_EVAL\_EXAMPLES \

--alsologtostderr

Так выглядит запуск здорового человека.

А так для тех, кто сидит на Windows 10 (и слушается меня в плане прописывания путей).

python object\_detection/model\_main.py --pipeline\_config\_path= a\models\model\ssd\_mobilenet\_v1\_pets.config --model\_dir=a\models\model\

--num\_train\_steps=50000 --sample\_1\_of\_n\_eval\_examples=1 --alsologtostderr

Старый вариант запускается, соответственно, так:

python train.py --logtostderr --train\_dir=a\data --pipeline\_config\_path=a\models\model\ssd\_mobilenet\_v1\_pets.config

По идее, мы должны радоваться и плясать (без бубна, он был только у меня). Конечно, если не выскочат никакие:

ошибки импорта модулей (привет, pycocotools),

ошибки кодировки (слеши, бекслеши, но никаких фемслешей, хо-хо),

ошибки доступа и существования файлов – проверяйте пути и экспериментируйте.

А, и о да, если у вас по \_счастливому\_ совпадению точно так же сгорела видео-карта, и располагаете вы только CPU ресурсами, посмотрите-ка еще раз на num\_train\_steps=50000 *подумойте*!!

Оригиналы и никакого Windows:

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md>

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/running_pets.md>

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/running_locally.md>

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/configuring_jobs.md>

И еще немного, зато с Windows: <https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10>

Датасет с енотами: <https://github.com/datitran/raccoon_dataset>

Чекпойнты с руками: <https://github.com/victordibia/handtracking>

C:\Users\karin.sadovs\Desktop\tf\models-master\research\object\_detection>python train.py --logtostderr --train\_dir=training\ --pipeline\_config\_path=training\ssd\_mobilenet\_v1\_coco.config