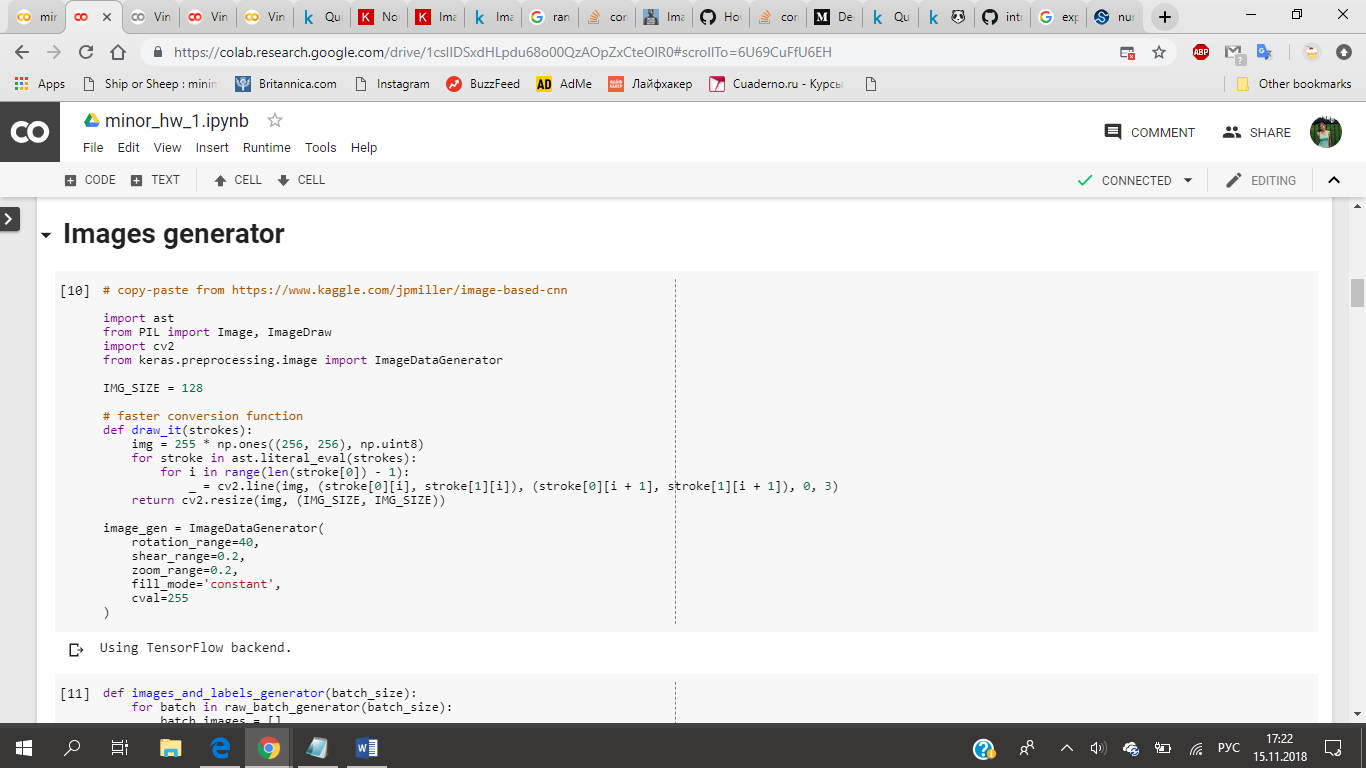
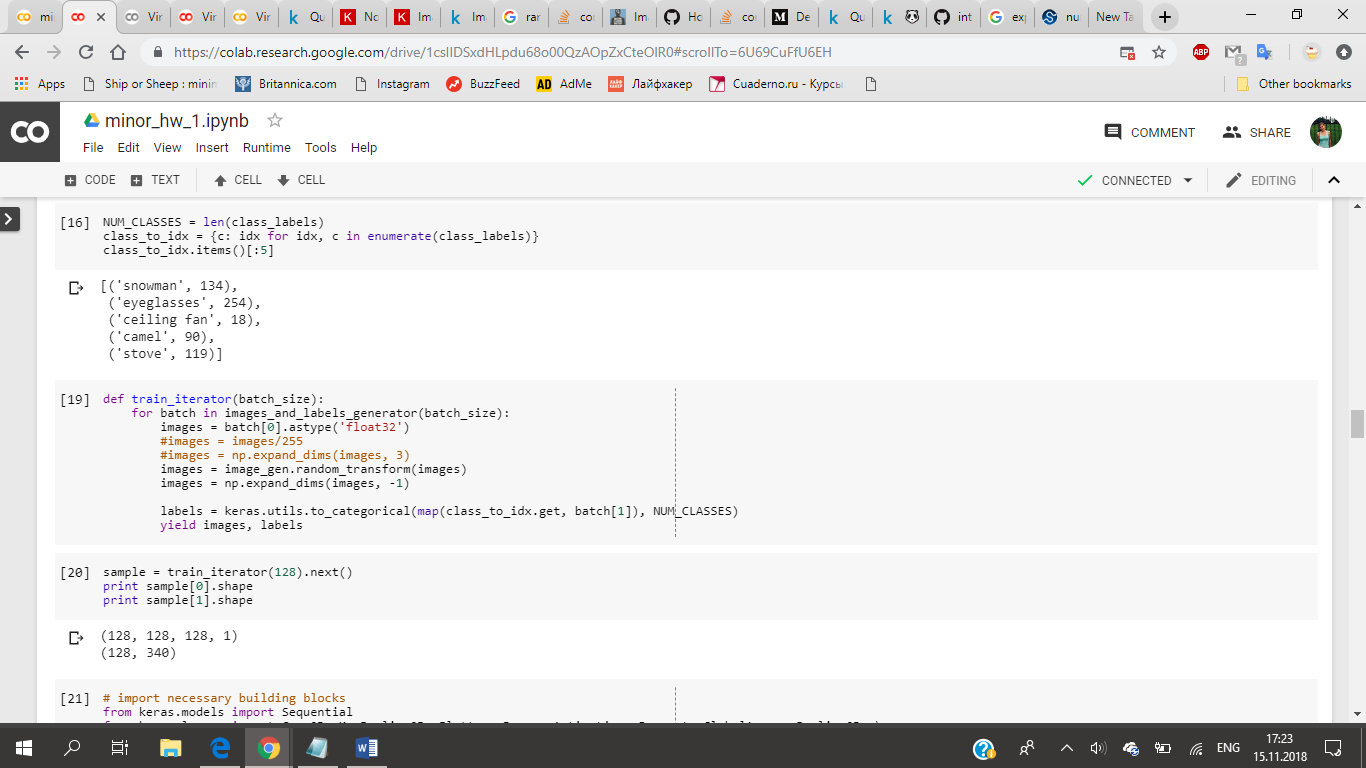
Я не сразу поняла, как лучше строить архитектуру, какие слои добавлять и в какой последовательности, поэтому много работ добавила на Kaggle с очень низкими результатами. Потом только поняла, что модель переобучалась на train, а на тестовой выборке показывала плохие результаты.

В итоге, после многих проб и попыток, дополнительного чтения и помощи от лектора в виде ноутбука составила неплохую модель: Conv2D-Conv2D-MaxPooling-Dropout(0,25)-Conv2D-Conv2D-MaxPooling-Dropout(0,25)-Conv2D-Conv2D-MaxPooling-Dropout(0,25)-GlobalAveragePooling2D-Dense(128)-Dropout(0,5)-Dense(340, ’softmax’). У convolutional layers использовала kernel size **(3, 3)** и возрастающее количество фильтров **(16, 16, 32, 32, 64, 64), использовала Leaky ReLU(0,1). Pooling layers взяла (2,2). В итоге было 123972 параметра, а размер картинок = 128, и при небольших batch\_size, steps\_per\_epoch и epoch модель уже показала хороший результат на тестовой выборке в 0.574. Вообще, я, конечно, начала с еще более простой модели Conv2D(16, (3,3))-Conv2D(16, (3,3))-GlobalAveragePooling2D-Dense(64)-Dense(340), использовала активацию elu, кроме последней – там softmax.На тестовой выборке результат получился 0.208, при размере картинок = 128 и 25668 параметрах, но здесь я просто проверяла, правильно понимаю структуру слоев и вообще, как все работает, потому что до этого результаты были 0.005. При этом я так сконцентрировалась на архитектуре модели, что забыла нормализовать выборку, а это нужно было сделать, судя по комментариям лектора в выложенном им ноутбуке. Но результаты без нормализации были отличными, а с ней процесс обучения почему-то застопорился на score меньшем 0,01, поэтому решила это пока не применять.**

Дальше была выложена новая версия ноутбука, в которой предлагалось поупражняться с аугментацией картинок. Я пыталась, сделала, как на скринах, но модель совсем не училась, поэтому я решила отложить аугментацию и усложнить предыдущую модель, потому что было куда расти.





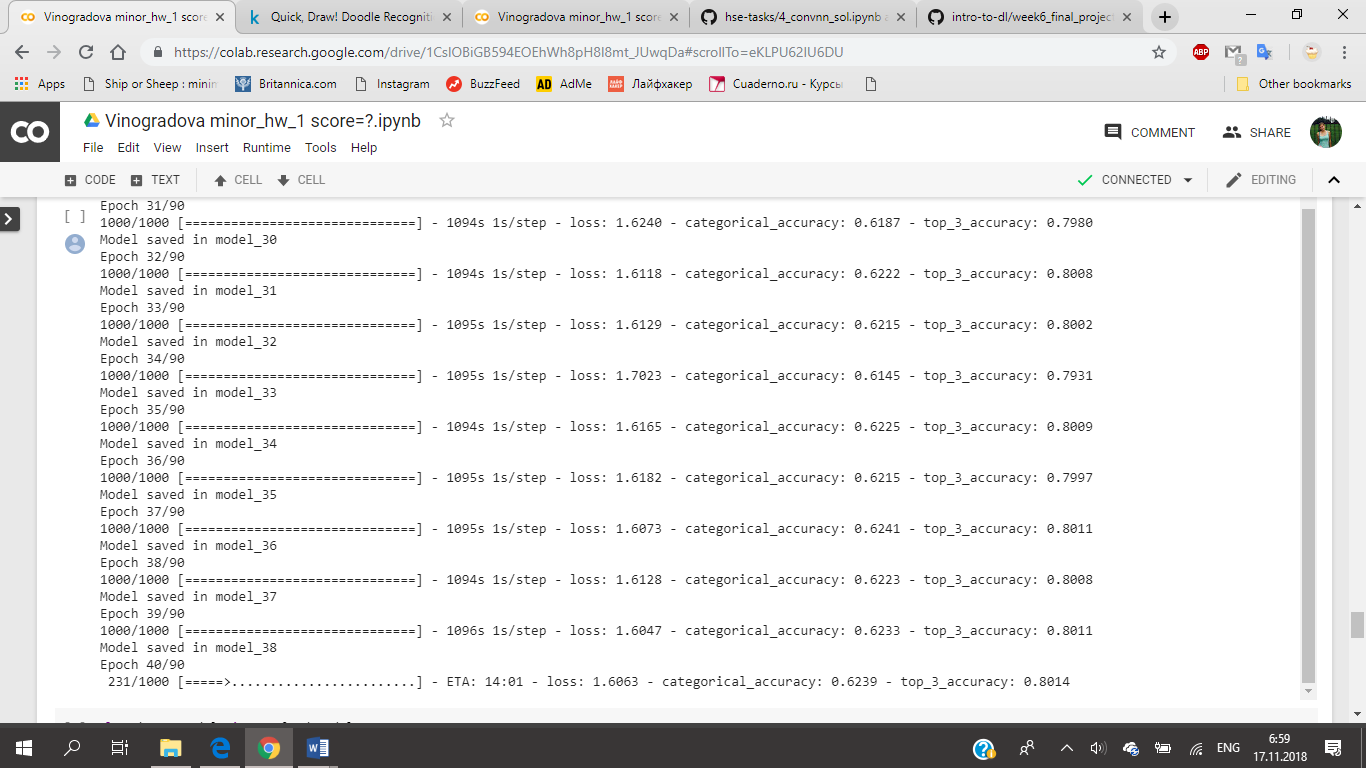
Расширила предыдущую модель, теперь стало вот так: Conv2D-Conv2D-MaxPooling-Dropout(0,25)-Conv2D-Conv2D-MaxPooling-Dropout(0,25)-Conv2D-Conv2D-MaxPooling-Dropout(0,25)- Conv2D-Conv2D-MaxPooling-Dropout(0,25)-GlobalAveragePooling2D-Dense(128)-Dropout(0,5)-Dense(340, ’softmax’). У convolutional layers использовала kernel size **(3, 3)** и возрастающее количество фильтров **(16, 16, 32, 32, 64, 64, 128, 128), использовала Leaky ReLU(0,1). Pooling layers взяла (2,2). В итоге было 353604 параметра, а размер картинок = 128, и при batch\_size=512, steps\_per\_epoch=512 и epoch=32 при обучении модель достаточно легко дошла до 0.7, а дольше застопорилась, на тестовой выборке score = 0,693 (от чего мне стало даже обидно).**

Решила все же добавить нормализацию изображений в комбинации с предыдущей архитектурой. В этот раз обучение идет гораздо лучше (по сравнению с прошлым добавлением нормализации), видимо, до этого я косячила. Обучение идет более равномерно. Но результат на тестовой выборке оказался хуже! 0,685 (снова обидно, что не дотянула до 0,7)

Все же убрала нормализацию изображений. Усложнила модель с помощью Flatten и заменила последний GlobalAveragePooling2D на MaxPooling. Размеры батча, шагов и эпох пока оставила как раньше. Параметров теперь 1385796. За три эпохи на обучении моделька дошла до 0,5. Переобучится? Нет, не переобучилась! На тесте получилось 0.724. Flatten хорошо помог.

Запустила предыдущую модель с **batch\_size = 512, steps\_per\_epoch = 800 и epoch = 40. За 10 часов (я успела на учебу и тренировку съездить) модель набрала 0.791. ОБИДНО. А модель потом отключилась.**

Ради эксперимента заменила Dense(128) на Dense(256). Модель тут же переобучилась. Возвращаюсь к прежней с большим количеством **steps\_per\_epoch и epoch. Соответственно, 1000 и 90. Оставила ноутбук на ночь, он доучился до 40 эпохи и дольше силы кончились, когда проснулась восстановить уже не получилось. Top\_3\_accuracy была больше 0.8 несколько эпох.**



Дальше несколько дней я все же пыталась обучить на большом количестве **steps\_per\_epoch и epoch свою модель. Но мне не хватило мозгов освоить чекпоинты на гугл драйве, поэтому не хватало времени дообучать модель до достойного результата.**

**Добавила вот такую конструкцию:**

**from keras.callbacks import ReduceLROnPlateau**

**callbacks = [**

**ReduceLROnPlateau(monitor='val\_loss', factor=0.1, patience=3, min\_lr=10e-5, verbose=1)]**

**Она была призвана уменьшать лернинг рейт, когда лосс на каждом шаге уменьшается слишком слабо, и это должно помочь медленнее и точнее спускаться к локальному минимуму.**

Модель с LeakyReLU(0.2) у меня тут же переобучилась, хотя я их даже не везде увеличила => такая архитектура точно не подойдет.

Пробовала менять LeakyReLU на обычный relu, обучается медленнее и тоже не имеет смысла.

