```
!pip install -q efficientnet
!pip install git+https://github.com/mjkvaak/ImageDataAugmentor
```

Библиотека efficientnet была актуальна года два назад, но с тех пор EfficientNet появился в модуле keras.applications, так что устанавливать доп. библиотеку не обязательно.

Библиотека ImageDataAugmentor, на мой взгляд, имеет мало смысла, ведь в обычном ImageDataGenerator есть параметр preprocessing_func, куда можно передать функцию из albumentations.

```
# Устаналиваем конкретное значение random seed для воспроизводимости
os.makedirs(PATH,exist_ok=False)
```

Воспроизводимось будет в генераторах, но при обучении моделей все равно ее не будет, так как у tensorflow свой сид, и к тому же обучение сверточных сетей на GPU идет недетерминированно. Поэтому к 100%-й воспроизводимости в DL обычно не стремятся.

```
A.OneOf([
        A.CenterCrop(height=224, width=200),
        A.CenterCrop(height=200, width=224)],
        p=0.1),
```

Если мы делаем аугментации, то почему не использовать randomCrop? То, что при обучении некоторые автомобили целиком не попадут в кадр – ничего страшного. Есть даже способ аугментации, называемый cutout, в котором из изображения вырезаются участки.

```
train_datagen = ImageDataAugmentor(rescale=1./255,
```

В разных моделях нужна разная нормализация, но почти нигда она не равна делению на 255. Например, для Xception требуется нормализация tf.keras.applications.xception.preprocess_input – это не то же самое, что деление на 255. Правильная нормализация (то есть такая же, какая была при предобучении) может немного улучшить точность, хотя это не принципиальный момент.

```
test_generator = train_datagen.flow_from_directory(
```

У вас валидация делается с аугментациями. Так не должно быть. Представьте, что одну модель вы обучили на слабых аугментациях, другую на сильных. Если валидация делается с аугментациями, то модель с сильными аугментациями покажет более низкую точность на валидации. Если же делать валидацию нормально, без аугментаций, то все может быть наоборот. Поскольку валидация влияет на выбор моделей и гиперпараметров, важно делать ее без аугментаций, особенно если вы сравниваете разные модели и значения гиперпараметров.

```
model.add(L.Dense(256,
                  activation='relu'))
model.add(L.BatchNormalization())
```

Эти слои в голову обычно не добавляют, но вы можете проверить – вдруг они повысят точность?

```
model.evaluate_generator(
```

Этот legacy-метод сейчас эквивалентент .evaluate().

В целом получилась хорошая точность, в основном за счет большого размера изображений (456x456). Можно было бы еще повысить точность, применив TTA.

> стоит сделать еще несколько вариантов, чтобы объединить их в ансамбль нейросетей (возможно это улучшит результат). В данном ноутбуке планирую обучить нейросеть на основе EfficientNetB6.

Получается, вы обучаете 2 модели, но я не нашел у вас кода, где бы объединялись эти модели в ансамбль.

> модель EfficientNetB5 показала метрику точности на тесте лучше, чем EfficientNetB6. Хотя в ходе обучения вторая модель обучалась на равне с EfficientNetB6 и показывала результаты на валидации даже иногда лучше.

Если одна модель показывает результаты лучше другой, то можно первую взять с большим весом, чем вторую, при усреднении.