Непонятна мотивация работать с Tensorflow 2.1. Можно использовать последнюю версию 2.6, в ней есть efficientNet в модуле keras.applications, так что нет необходимости в установке pip-пакета efficientnet. Возможно вы работали в TF 2.1 потому что так делается в бейзлайне на Kaggle. Этот бейзлайн написан давно, и версии библиотек сейчас стали не актуальны. Но ничего страшного, и версия TF 2.1 должна работать нормально.

```
import tensorflow.keras.layers as Layer
```

Такой импорт немного сбивает с толку. Обычно с большой буквы пишутся имена классов. Кроме того в keras есть класс с таким же именем: keras.layers.Layer.

```
for data_zip in ['train.zip', 'test.zip']:
    with zipfile.ZipFile("../input/"+data_zip,"r") as z:
        z.extractall(PATH)
```

Это можно заменить командой !unzip train.zip -d {PATH}

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

Обратите внимание, что генератор изображений для валидации у вас делается на основе train_datagen, в котором делаются аугментации. Поэтому валидация у вас делается с аугментациями. Это типичная ошибка, на которую почему-то никто не обращает внимания. Так не должно быть. Представьте, что одну модель вы обучили на слабых аугментациях, другую на сильных. Если валидация также 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.

```
checkpoint = ModelCheckpoint('best_model.hdf5',
                             monitor = ['val_accuracy'],
                             verbose = 1,
                             mode = 'max')
```

Насколько я помню, здесь надо добавить параметр save_weights_only=True, иначе модель будет сохраняться после каждой эпохи.

```
model.fit_generator
```

Такой метод является устаревшим, сейчас вызов fit_generator эквивалентен вызову fit.

Не совсем понял смысла вашего кода, который идет после «Увеличим размер изображения и понизим уровень аугментации»:

```
base_model = efn.EfficientNetB6(weights='imagenet',
                                include_top=False,
                                input_shape=input_shape)
base_model.trainable = True
```

Здесь вы создаете новую модель EfficientNet, но нигде далее ее не используете. По сути в этом коде вы просто переопределяете переменную base_model. Чтобы это имело эффект, нужно переменную base_model далее где-то использовать. Но и смысл действия тоже непонятен: зачем создавать новую модель? Вы ведь хотите дообучать старую на новом размере изображений? Сверточные сети исходя из принципа своей работы способны принимать на вход изображения разного размера, поэтому никаких действий делать не нужно.

В целом у вас достигнута хорошая точность. В основном это обусловлено хорошим выбором модели и высоким разрешением подаваемого на вход изображения.

Оценивание по критериям:

Качество и понятность кода

3 балла из 3

Метрика качества

Оценивается по лидерборду на Kaggle.

0.97243, 2 балла из 3

Работа с обучающими данными

Проведите несколько экспериментов с разными аугментациями изображений, выберите оптимальные аугментации на основе чего-то из перечисленного:
- Точность на валидационной выборке или сабмите
- Научные статьи и статьи из интернета
Всегда проверяйте аугментации визуально, отрисовывая изображения.

У вас выполняются аугментации, но валидация делается с аугментациями, по-моему это ошибка. Кроме того у вас не проверяется визуально результат выполняемых аугментаций.

1 балл из 3

Работа с архитектурой модели

Проведите несколько экспериментов с разными архитектурами моделей, выберите оптимальную архитектуру (или несколько архитектур) на основе чего-то из перечисленного:
- Точность на валидационной выборке или сабмите
- Научные статьи и статьи из интернета

Вы выбрали архитектуру EfficientNet и добавили несколько слоев в «голову» модели (dense, dropout, batchNorm). Но этот выбор ничем не обоснован. Стоило провести сравнение разных методов (например запустить несколько экспериментов на изображениях с маленьким размером длоя ускорения вычислений), либо руководствоваться статьями из интернета, где советуют делать так или иначе. Тогда нужно дать ссылки на эти статьи.

1 балл из 3

Работа с процессом обучения

Проведите несколько экспериментов с разными оптимизаторами, learning rate и (опционально) продвинутыми техниками, такими как постепенная разморозка слоев, progressive resizing или изменение степени аугментаций при обучении. Выберите оптимальный вариант на основе чего-то из перечисленного:
- Точность на валидационной выборке или сабмите
- Научные статьи и статьи из интернета

Здесь можно повторить то же самое. Вы выполняете обучение сначала головы, затем половины сети, затем всей сети, затем всей сети на увеличенном размере изображений. Но ничем не

обосновывается что именно такой подход наилучший. Вдруг окажется, что если бы вы обучали сеть всю изначально целиком и сразу на большом размере, то точность оказалась бы еще выше? Нужно либо привести ссылки на статьи, либо сравнить несколько подходов экспериментально.

1 балл из 3

Работа с процессом инференса

Инференсом называется получение предсказаний модели на данных.
Делайте ансамблирование и TTA с целью улучшить точность предсказаний. Всегда проверяйте, насколько улучшилась (и улучшилась ли) точность от использования ансамблирования и TTA.

Ансамблирования и TTA у вас не нашел

Создание работающего прототипа

Реализуйте один из вариантов:
Веб-сервер, принимающий изображение (можно по URL) и возвращающий название модели автомобиля
Скрипт .py, который печатает название модели автомобиля, присутствующего на выбранном изображении

Этого пункта у вас не нашел

Сумма

3+2+1+1+1=8 баллов