Здравствуйте! Далее комментарии по коду работы, журналу и отчету:

- был обработан и создан новый набор данных - в котором картинки сжаты по размеру до 224 пикс по длинной стороне и дополнены черным цветом до квадрата.
- Обработка первого каталога показала, что 194 из 1610 были вертикальными.

Если большинство фотографий горизонтальны, то можно было рассмотреть такой вариант: в качестве стандартного размера взять не 224x244, а скажем 360x240. Тогда в вертикальных фотографиях будет больше черных полей, а в горизонтальных их будет меньше.

```
def stretch_image(src, dst, nm, size):
```

Есть и стандартные функции для таких действий, например, можно использовать:
- tf.image.resize_with_crop_or_pad
- tf.image.resize_with_pad

```
# from tensorflow.keras.applications.efficientnet_v2 import EfficientNetV2
B0
```

- К сожалению новая версия EfficientNetV2B0 не захотела устанавливаться и этот runtime завершился с ошибкой. Надо будет пробовать.

Не знал что в keras.applications добавили такую модель. Если не работает, можно взять аналогичную модель из tensorflow hub.

- Модель Xception принимает на вход только 3х канальное изображение и в сравнении не участвовала.

EfficientNet тоже требует три канала, то что она работает с одним каналом – наверное случайность, и так не задумано. Но из одного канала всегда можно сделать три одинаковых.

- **Подход 4.** Изучение аугментации.

У вас приводятся таблицы точности моделей с разными аугментациями, но непонятно сколько эпох обучались эти модели (или я не заметил, но кажется там нигде не написано).

- Складывается впечатление, что НС Xception отличается хорошим качеством распознавания - но обучаться с ней дольше. EfficientNetB0 - оправдывает названия, быстрая, эффективная, но качество чуточку хуже.

B0 не очень большая модель, вот B3 дает уже более высокую точность, что видно из ваших экспериментов. По моим экспериментам Xception и B6 давали примерно одинаковую точность.

- Правильнее делать аугментацию в слоях. Вариантов аугментации в стандартном пакете не слишком много.

Аугментацию можно делать с помощью tf.data.Dataset и операций tensorflow, там очень много возможностей и быстрая скорость выполнения.

- Оказывается, что больше всего ошибок между классами 7 и 9: ошибочные классификации в количестве 32 и 40. Оказывается 7 класс - это 2109, а 9 класс - 21099. Модели

отличаются типом кузова и формой задней части - хэтчбек и седан. Соответственно они не могут быть определены по фотографии капота, где не видно формы багажника.

Совершенно верно, эти две модели вносят основной вклад в ошибку. Без них можно было бы поднять точность еще выше. Еще одня проблема в том, что на фото часто несколько автомобилей.

- Kaggle Score до TTA: 0.96659 Kaggle Score после TTA - 0.96629 С учетом TTA стало чуточку хуже!

Можно попробовать сделать TTA на валидационной выборке и измерить, как зависит точность от числа шагов TTA (и сравнить с точностью без TTA). Скорее всего было бы видно, что при увеличении числа шагов точность повышается, и можно было бы спрогнозировать точность при большем числе шагов, и подобрать оптимальное число шагов. Также можно делать доп. шаги TTA только на тех изображениях, где модель сомневается (то есть выдает разный ответ на разных шагах TTA).

- Короткий хвостик из одного плотного слоя на 10 нейронов и чуть более длинный с промежуточным плотным слоем в 256 нейронов. Чуть более длинный хвостик показал чуть более качественный результат.

Более длинный быстрее обучается, но и сильнее переобучается. В моих экспериментах получалось, что если обучать сеть большое количество эпох, то сеть без дополнительных полносвязных скрытых слоев дает лучшую точность. Но на среднем числе эпох доп. скрытый слой, наоброот, повышает точность.

- Для adam был подобран вариант экспоненциального затухания `optimizer=Adam(ExponentialDecay(1e-3, 100, 0.93))`.

Здесь очень важно, чтобы learning rate не опускался слишком быстро, иначе сеть перестанет обучаться, даже если у нее есть потенциал к дальнейшему обучению. Как вариант, можно использовать InverseTimeDecay.

В целом у вас, пожалуй, самая подробная и лучшая работа,
которую я проверял за последний год :)

Отзыв подготовил ментор проекта Олег Седухин. Если есть вопросы, можете задать их в канале #0-project_7-ford_vs_ferrari, постараюсь помочь разобраться. Успехов в дальнейшем обучении!