Здравствуйте! Далее комментарии по коду работы:

В ноутбуке по ссылке нет выводов ячеек кода, поэтому если в них было что-то важное (графики, результаты аугментаций), то не могу посмотреть.

```
# Исходные данные можно распаковать в output, используя ниже приведенный код
...
# Однако при такой распаковке возникает проблема сохранения файла submission
# Поэтому распакованные исходные данные добавили из базы каггла
```

На Kaggle вы можете создать папку /kaggle/temp и распаковывать картинки в нее, тогда проблем с сохранением не будет.
Для распаковки можно использовать такую команду:
!unzip -q { DATA_PATH + data_zip} -d /kaggle/temp

```
A.OneOf([
        A.CenterCrop(height=224, width=200),
        A.CenterCrop(height=200, width=224)],
        p=0.5),
```

CenterCrop можно заменить на RandomCrop, если цель – создание случайных аугментаций.

```
train_gen = 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.

```
test_datagen = train_gen.flow_from_directory(DATA_PATH+'train/train',
                                             class_mode='categorical',
                                             batch_size=BATCH_SIZE,
                                             target_size=(IMG_SIZE,
IMG_SIZE),
                                             shuffle=True,
                                             subset='validation'
                                            )
```

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.

Пример того, как делать валидацию без аугментаций, есть в бейзлайне.

```
model.add(Layer.Dense(256,
                      activation='relu',
                      bias_regularizer=l2(1e-4),
                      activity_regularizer=l2(1e-5)))
model.add(Layer.BatchNormalization())
```

Непонятно из каких соображений добавлены эти слои именно с такими параметрами. Например, зачем нужны bias_regularizer и activity_regularizer в данной сети? Если вы пробовали разные виды

голов, и такая оказалась наилучшей, то было бы хорошо сделать таблицу и в ней показать разные архитектуры головы и достигнутые значения точности.

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

Нужно не забыть параметр save_best_only=True.

```
reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.25,
                              patience=2,
                              min_lr=0.0000001,
                              verbose=1,
                              mode='auto')
```

Мне кажется у вас Learning rate убывает слишком быстро, можно было бы увеличить patience. Модели часто обучаются подолгу, и в процессе обучения точность может случайно то расти, то падать. С такой стратегией затухания LR, какая реализована у вас, в итоге LR станет практически нулевой, и сеть перестанет обучаться, даже если у нее был потенциал к обучению.

```
tta_steps = 10
predictions = []

for i in range(tta_steps):
    preds = model.predict(test_sub_generator, verbose=1)
    predictions.append(preds)
```

TTA работает тем лучше, чем больше шагов, и при 10 шагах точность наоборот может оказаться ниже, чем без TTA. Поэтому прежде, чем делать сабмит с TTA, лучше проверить, дает ли TTA повышение точности или нет. Вдруг наоборот снижает точность?

В целом, в итоге у вас достигнуто очень хорошее значение точности!

Отзыв подготовил ментор проекта Олег Зяблов. Если есть вопросы, можете задать их в канале #0-project_7-ford_vs_ferrari, постараюсь помочь разобраться. Успехов в дальнейшем обучении! Обязательно подключайтесь на итоговый созвон-вебинар по проекту **29 января**. Анонс вебинара появится позже в канале #0-project_7-ford_vs_ferrari.