Здравствуйте! Далее комментарии по коду работы:

```
print('Распаковываем картинки')
# Will unzip the files so that you can see them..
for data_zip in ['train.zip', 'test.zip']:
    with zipfile.ZipFile("../input/"+data_zip,"r") as z:
        z.extractall(PATH)
```

Это можно было бы заменить командой !unzip -q /Kaggle/input/train.zip -d {PATH}

- Способ аугментации из бейзлайна показал качество хуже в сравнении с применением albuminations, не будем его использовать.

Здесь важно не то, какую библиотеку вы используете, а то, какие конкретно аугментации делаете. Например, случайный поворот изображения, скорее всего, даст один и тот же результат независимо от того, с помощью какой библиотеки он выполнялся: albumentations, tensorflow и др.

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

Если это аугментации, то почему не использовать randomCrop? Хотя, с другой стороны, иногда в кадре есть и другие автомобили, и при randomCrop у них больше шанс попасть в кадр, так что применение centerCrop тоже обосновано.

```
train_datagen = ImageDataAugmentor(
        rescale=1./255,
        augment = augments,
        validation_split = VAL_SPLIT,
        seed = RANDOM_SEED
        )
```

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_generator = test_datagen.flow_from_directory(
```

Совершенно правильно, что валидацию вы делаете без аугментаций.

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
# x = BatchNormalization()(x)
# let's add a fully-connected layer
x = Dense(256, activation='relu')(x)
# x = BatchNormalization()(x)
x = Dropout(0.25)(x)
# and a logistic layer -- let's say we have 10 classes
predictions = Dense(CLASS_NUM, activation='softmax')(x)
```

Как вариант, можно создать аналогичную модель так:

```
model = Sequential([
  base_model
  GlobalAveragePooling2D(),
  Dense(256, activation='relu'),
  Dropout(0.25),
  Dense(CLASS_NUM, activation='softmax')
])
```

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

Здесь нужно добавить параметр save_best_only=True

earlystop = EarlyStopping(monitor='accuracy', patience=5, restore_best_weights=True)

Как вариант, можно увеличить patience, нейронные сети часто обучаются очень долго. Дажа если точность на валидации идет вниз, то это еще не значит, что через несколько эпох она снова не пойдет вверх.

```
model.fit_generator
```

Этот legacy-метод эквивалентен model.fit().

Можно было бы попробовать сначала обучить всю модель сразу без заморозки слоев, а потом уже обучать с постепенной разморозкой – так легче было бы понять, увеличивает ли постепенная разморозка точность.

Еще можно было попробовать применить TTA, как показано в примере в бейзлайне.

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!