```
# сбалансируем данные по категориям
cat_max = int(train.Category.value_counts().max())
…
```

Вы балансируете данные методом осверсэмплинга, хорошая идея, обычно на несбалансированность закрывают глаза. Но с другой стороны, если целевой метрикой является accuracy (а не F-мера например), то необходимость балансировать классы сомнительна. Вообще это интересный вопрос, *если целевой метрикой является accuracy, имеет ли смысл балансировать классы, повысит ли это метрику*? Поизучал научные работы по этой теме, но ни в одной из работ по этому вопросу ничего не нашел. Везде говорят только что accuracy использовать не надо, но это уже другой вопрос. Я думаю что при балансировке классов accuracy не повысится.

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

Почему бы не делать randomCrop (а не CenterCrop), если мы делаем аугментации?

Очень важно проверять, правильный ли формат изображения вы используете. В документации к библиотеке albumentations сказано следующее:

«Under the hood, Albumentations supports two data types that describe the intensity of pixels:
- np.uint8, an unsigned 8-bit integer that can define values between 0 and 255.
- np.float32, a floating-point number with single precision. For np.float32 input, Albumentations expects that value will lie in the range between 0.0 and 1.0.»

Есть большая вероятность (это не точно, в документации про это ничего нет), что ImageDataAugmentor использует формат float с диапазоном 0-255. В этом случае часть аугментаций в albumentations может работать неправильно, если не преобразовывать изображение к формату uint8 в функции augment.

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

Но EfficientNet принимает изображение со значениями пикселей от 0 до 255. Если подавать значения от 0 до 1, то это будет отличаться от того, что использовалось при обучении этой сети на ImageNet, что может сказаться на качестве.

```
test_generator = train_datagen.flow_from_directory(
    train_path,
```

У вас валидация делается с аугментациями. Практически все студенты так делают, но по-моему 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.

Насколько я понимаю, вы сначала использовали новый бейзлайн, а потом решили переключиться на старый? В новом валидация делалась без аугментаций (в начале вашего ноутбука код из нового бейзлайна, далее из старого).

```
pip install -U git+https://github.com/qubvel/efficientnet
```

Это нужно было в старых версиях keras. В новых efficientNet есть в модуле keras.applications. В начале вашего ноутбука она как раз загружается из этого модуля, а в середине ноутбука вы почему-то устанавливаете pip-пакет efficientnet.

```
ModelCheckpoint(f'{MODEL}_best.hdf5' , monitor = ['val_acc'] , verbose = 1
, mode = 'max')
```

Здесь надо добавить save_best_only=True, иначе модель будет сохраняться каждую эпоху независимо от точности.

```
LearningRateScheduler(lambda x: 1e-3 * 0.95 ** x),
```

Понадобится очень много эпох, чтобы затухание learning rate стало существенным. Вообще я бы советовал использовать PiecewiseConstantDecay, он часто дает лучшие результаты.

```
# значение метрики может колебаться в процессе обучения, поэтому иногда
стоит продолжить
# обучение несмотря на ухудшение метрики
```

Согласен, мне нравится следующая цитата, она совпадает с моими наблюдениями: «leave it training. I've often seen people tempted to stop the model training when the validation loss seems to be leveling off. In my experience networks keep training for unintuitively long time. One time I accidentally left a model training during the winter break and when I got back in January it was SOTA ("state of the art").»

Но это верно только если датасет большой. Если небольшой (а 15 тысяч изображений – это скорее небольшой), то лучше все-таки не обучать чрезмерно долго, может начаться переобучение.

```
# Заморозим веса EfficientNetB5 и обучим только "голову".
# Делаем это для того, чтобы хорошо обученные признаки на Imagenet не
затирались в самом начале нашего обучения
```

Нейронные сети несколько сложнее, чем часто может показаться. Подолжные рассуждения выглядят логично, но часто оказываются неверны на практике. Я пробовал сравнивать послойный файн-тюнинг и файн-тюнинг сразу целиком, и второй вариант давал лучшую точность. Но конечно я провел недостаточно много экспериментов чтобы утверждать это наверняка. Пишу это к тому, что в DL надо во всем сомневаться.

То же самое насчет архитектуры головы. Почти все учасники данного соревнования добавляют в голову полносвязный слой и batch-norm, и почти никто не проверяет – а дает ли это прирост точности по сравнению с головой, состоящей только из GlobalMaxPool2D и выходного слоя? Болше слоев не значит лучше. Если посмотреть на научные работы по сверточным сетям, то там практически всегда делают голову только из GlobalMaxPool2D и выходного слоя. Значит это хорошо работает при обучении на ImageNet. Но с другой стороны это еще не гарантирует, что это наилучший выбор при файн-тюнинге.

```
Dropout(0.25),
```

По моим наблюдениям дропаут 0.5 обычно дает наилучший результат.

```
model.save(f'{MODEL}_last.hdf5')
model.load_weights(f'{MODEL}_best.hdf5')
```

Это одна и та же модель, потому что в ModelCheckpoint вы не указали save_best_only=True.

```
model.evaluate_generator(
```

Это устаревший метод, он сейчас эквивалентен evaluate.

```
for file in files:
        Cat = submission[submission.Id == file]['Category']
```

Эта техника называется pseudo-labeling. По идее в тестовый датасет добавляют те изображения, в которых нейросеть уверена (например все предсказания в TTA совпали). Если же в train добавить все изображения из теста, с соответствующими предсказаниями, то новая модель обучится давать точно такие же предсказания, что и предыдущая. Поэтому вряд ли это даст прирост точности.

- Псевдо разметка данных сабмишн (версия 10) выигрыша в точности не дала.

Причина этого как раз и описана выше. Надо было брать не все изображения из теста, а те, где все или почти все предсказания TTA совпали. А для того, в первую очередь, надо пофиксить TTA, у вас в нем ошибка, об этом читайте ниже.

```
for _ in range(5):
  predictions.append(model.predict(test_sub_generator, verbose=1))
```

Вы делаете предсказания на test_sub_generator, в котором *не делаются аугментации*. А значит все итерации цикла TTA дают в точности одинаковые предсказания. Я бы посоветовал визуализировать предсказания (predictions) с помощью plt.imshow чтобы увидеть, дают ли разные шаги TTA разные предсказания или одни и те же. Скорее всего вы бы увидели, что одни и те же.

- Аугментация при тестировании (версия 19) не привела к повышению точности.

Как раз причина этого и описана выше.

Но если бы вы исправили test_sub_generator так, чтобы он делал аугментации, то столкнулись бы с другой проблемой. У вас в аугментациях указан угол поворота до 30 градусов и cutout. Это сильные аугментации, а чем более сильные аугментации, тем больше они ухудшают точность, и следовательно тем больше шагов TTA надо делать. У вас делается всего 5 – это может дать точность хуже, чем без TTA.

Еще можно делать так: сделать сначала одинаковое количество шагов TTA на каждом изображении, а затем делать дополнительные шаги на тех изображениях, в которых результат остался спорным.

И кроме того у вас нигде не написаны результаты сравнения сабмита с TTA и без TTA. Верно ли, что они получились одинаковые?

```
CoarseDropout(p=1, max_height=18, max_width=18)
```

Cutout размером всего 18x18 – это наверное слишком мало для изображений 250x250.

- На бейзлан была пролучена точность 0,64299.

Возможно вы решили перейти на старый бейзлайн из-за того, что точность на новом была слишком низкой. В нем (для примера, чтобы быстрее считалось) используется размер изображений 90x120, а обычно чем больше размер – тем выше точность при использовании сверточных сетей. Надо было просто изменить размер изображения и обучать болше эпох. Даже на размере 90x120 можно достичь точности 95% и выше. Но все же такой размер имеет смысл

использовать только для подбора гиперпараметров (для ускорения обучения), а финальную модель обучать на изображениях большего размера.

В целом можно сказать, что при правильной релизации TTA и pseudo-labeling практически всегда дают прирост точности.