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

Это можно было бы записать проще (работает только в IPython-средах):

!unzip ../input/train.zip -d {PATH}
!unzip ../input/test.zip -d {PATH}

```
augmentations = a.Compose([
    a.GaussianBlur(p=0.05),
    ...
```

Здесь можно было бы еще добавить cutout – вырезание случайных участков из изображения.

Кроме того полезно было бы замерить время, затрачиваемое на выполнение каждой выбранных вами аугментаций. Может случиться такое, что какая-то из них слишком долго выполняется, и тогда либо стоит от нее отказаться, либо заранее сгенерировать несколько копий каждого изображения с этой аугментацией и пользоваться ими.

И наконец 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, 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.

```
test_datagen = datagen.flow_from_directory(
```

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.

```
# Add a fully-connected layer
model.add(Layer.Dense(256,
                      activation='relu',
                      bias_regularizer=l2(1e-4),
                      activity_regularizer=l2(1e-5)))

# Add batch normalization
model.add(Layer.BatchNormalization())
```

Здесь не очень понятно из каких соображений добавлены эти слои. Я бы посоветовал провести предварительные эксперименты на небольшом размере изображения (чтобы быстрее шло обучение) с разными вариантами реализации последних слоев («головы»). Обычно в голову добавляют только выходной слой и не добавляют скрытых полносвязных слоев. Но не факт что это всегда верное решение.

```
ModelCheckpoint('best_model.hdf5',
```

```
                                monitor = ['val_accuracy'],
                                verbose = 1,
                                mode = 'max')
```

Если указываете значения monitor и mode, то также нужно указывать параметр save_best_only=True, иначе модель будет сохраняться каждую эпоху.

```
monitor='val_loss',
                                 factor=0.2, #let's reduce LR 5 times
                                 patience=2
```

Возможно стоит выбрать patience побольше. Приведу следующую цитату: «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").»

```
STEPS = 5
predictions = []

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

Чем более сильные аугментации, тем больше они ухудшают точность, и следовательно тем больше шагов TTA надо делать. 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.

«Techniques and functions used for the project:

- Transfer learning with fine-tuning
- 2 base models: Xception and EfficientNetB5
- Batch Normalization added and callbacks
- Advanced Albumentations package for augmentation
- Leraning rate managing techique
- Test Time Augmentation (just tested)»

В целом хорошая работа, но есть важное замечание. Если вы делаете какое-то действие, то важно его обосновывать – либо ссылками на какие-то внушающие доверие статьи, либо собственными экспериментами со сравнением разных подходов. В DL много спорного. То, что работает в одних случаях, не сработает в других. Я могу например написать так: «Использованы продвинутые техники: в голову добавлено 20 слоев, использовано 20 аугментаций, послойная разморозка модели, очень сложная стратегия learning rate и т. д.». Но потом окажется, что если использовать только один слой в голове, всего 2 вида аугментаций и всю модель обучать сразу целиком не замораживая, то результат получится лучше.