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

Это можно заменить командой !unzip train.zip -d {PATH}

```
train_datagen = ImageDataAugmentor(
    preprocess_input=efficientnet.preprocess_input,
```

Очень здорово, что вы нашли эту функцию, действительно это так и делается: для каждой сети свой алгоритм препроцессинга. Насколько я знаю, efficientnet.preprocess_input вообще ничего не делает с данными, поскольку нормализация встроена внутрь самой сети.

```
test_generator = train_datagen.flow_from_directory(
```

Обратите внимание, что генератор изображений для валидации у вас делается на основе train_datagen, в котором делаются аугментации. Поэтому валидация у вас делается с аугментациями. Это типичная ошибка, на которую почему-то никто не обращает внимания. Так не должно быть. Представьте, что одну модель вы обучили на слабых аугментациях, другую на сильных. Если валидация также 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.

```
base_model = EfficientNetB3(
```

Я думаю правильно, что вы не взяли EfficientNetB7, она слишком тяжелая для этой задачи. По-моему можно было даже взять EfficientNetB0, ее можно было бы обучить в течение большего количества эпох и получить более высокую точность. Даже если обучать одинаковое количество эпох, не факт что EfficientNetB7 даст результат лучше EfficientNetB3. Дело в том, что глубокие сети сильнее переобучаются, и поэтому требуют больше обучающих данных. А сеть EfficientNetB7 более глубокая, чем EfficientNetB3.

Вы выполняете обучение сначала головы, затем половины сети, затем всей сети, затем собираетесь обучать всю сети на увеличенном размере изображений. А вы уверены, что именно такой подход наилучший? Вдруг окажется, что если бы вы обучали сеть всю изначально целиком и сразу на большом размере, то точность оказалась бы еще выше? Я понимаю что так было написано в бейзлайне, но например по моим экспериментам это не подтвердилось. Я не утверждаю что это точно не лучший подход, просто здесь важно понять: большая часть утверждений в DL спорны, надо все подвергать сомнению.

В целом мне очень понравился ваш код. У нас с вами очень похожий code style.

```
for i in range(NUM_PREDICTIONS_FOR_TTA):
    temp_predict = model.predict(test_generator, verbose=1)
    predictions_TTA.append(temp_predict)
```

Тут можно еще делать так: сделать сначала 10 попыток на каждом изображении, а затем делать дополнительные попытки на тех изображениях, в которых результат остался спорным.