```
albumentations.ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.01
```

При таком scale_limit изображения почти не меняются в размере, но их качество может слегка ухудшиться из-за интерполяции, и к тому же на эту операцию затрачивается время. Поэтому непонятен смысл указания scale_limit=0.01, а не 0.

```
albumentations.Flip(p=0.5),
```

А есть ли какой-то смысл отражать изображения по вертикали?

```
rescale=1/255,              # поскольку беру из pip-пакета efficientnet, то
нормализация нужна
```

Нормализация нужна, но посмотрите пример из документации
https://github.com/qubvel/efficientnet/blob/master/examples/inference_example.ipynb
Нормализация делается с помощью efficientnet.keras.preprocess_input, которая в свою очередь вызывает tf.keras.applications.imagenet_utils.preprocess_input. Там нормализация делается с разными коэффициентами по разному каналу изображения, а не просто делением на 255.

```
return(train_generator,test_generator,test_sub_generator)
(train_generator,test_generator,test_sub_generator) = building_generators()
```

Тут можно и не ставить скобки, синтаксис python это позволяет

> На данный момент лучшим оптимайзером является ADAM. Он обучается за меньшее количество времени и более эффективно.

А из чего это следует? Хотелось бы увидеть ссылки на источники. Потому что по-моему в этом вопросе нет однозначности. Говорят, что иногда SGD при хорошо подобранных гиперпараметрах дает лучшую метрику на валидации. Кроме того есть оптимизатор AdaBound, который также хорошо работает себя в задачах CV. Про сравнение оптимизаторов можно почитать здесь https://arxiv.org/abs/2007.01547

> Step 4: добавим технику по увеличению самой картинки

Много закомментированного кода, непонятно нужно его смотреть или нет.

> Попробую улучшить метрику с помощью Test Time Augmentation
```
albumentations.ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.01,
                        interpolation=1, border_mode=4, rotate_limit=40
```

Если обучение шло с rotate_limit 20, а TTA проводится с rotate_limit=40, то точность может наоборот ухудшиться, потому что модель при обучении не видела настолько сильно повернутых изображений. И в целом чем сильнее аугментации, тем больше нужно шагов TTA, потому что качество каждого предсказания падает из-за искажений изображения.

> Проведя ряд экспериментов,сделала вывод ,что обучение всей модели целиком не приводит к очень хорошим результатам. Поэтому я попробовала применить Fine-tuning, разделив обучение на 3 шага. В первой части было обучение верхних слоев, во второй - половина и в третьей - разморожены все слои.

> применить Fine-tuning, разделив обучение на 3 шага

Дообучение всей модели сразу это ведь тоже fine-tuning. В термине «fine-tuning» не конкретизируется как именно дообучается модель - послойно или вся целиком.

> Большое преимущество у этой библиотеки в том, что она быстрее (судя по статьям из интернета)

Интересно какие это статьи, я такой информации не нашел. По-моему аугментации через Tensorflow и Keras будут быстрее, так как выполняются на GPU, хотя это не точно. Например с помощью функций Tensorflow:
https://www.tensorflow.org/addons/api_docs/python/tfa/image/random_cutout
Или слоев Keras:
https://keras.io/api/layers/preprocessing_layers/image_augmentation/random_rotation/

Отлично что вы занесли результаты в таблицу, так намного удобнее

Раздел «1 step (обучение верхних слоев)»: странно что добавление батч-нормализации без скрытого полносвязного слоя вызывает падение точности с 69% до 53%. Здесь что-то не так, возможно обучение шло недостаточно долго.

Конечно чтобы надежно установить нужны ли скрытый полносвязный слой и батч-нормализация, нужно намного больше экспериментов. Может, например, оказаться так, что при более долгом обучении батч-нормализация перестанет давать отрицательное влияние на обучение или даже даст положительное влияние. Было бы интересно еще посмотреть на графики точности при обучении, потому что по этим графикам лучше видно, было ли достаточным количество эпох. Также выбор learning rate может повлиять на результаты всех экспериментов, и наконец есть случайная погрешность, вызванная случайной инициализацией весов и тем, что процесс обучения также содержит элементы случайности.

Также в таблице вы сравниваете послойное обучение и обучение сразу целиком, и послойное обучение дает точность выше. Здесь важно чтобы все остальные условия (learning rate, суммарное кол-во эпох и т. д.) были одинаковыми, из таблицы это непонятно. Например если вы обучали до того, как сработает EarlyStopping, то могло получиться так: при послойном обучении вышло суммарно 30 эпох, а при обучении целиком суммарно лишь 10 (на графиках это было бы видно, а по таблице не видно). Тогда и сравнивать будет неправильно. И еще: непонятно какая стратегия изменения LR использовалась в экспериментах, приведенных в таблице. Если это например был ExponentialDecay, то скорость затухания могла быть выбрана слишком большой.

В целом очень здорово что вы сделали таблицу результатов экспериментов