Качество и понятность кода
Оценка: 3 балла из 3.

Метрика качества 0.94921

Работа с обучающими данными
В первом варианте вы делали более сложные аугментации, сейчас решили делать более простые. Я согласен с вами в этом, простые аугментации часто дают результат не хуже сложных, а работают быстрее. Очень понравилось как вы делаете визуализацию с помощью .flow из батча изображений. Кстати вместо x.reshape((1,) + x.shape) можно писать x[None, :] – добавление новой оси размерностью 1. В train_generator вы забыли указать seed, возможно это приведет к тому, что обучающий и валидационный датасеты будут пересекаться. Но это не точно.
Оценка: 3 балла из 3.

Работа с архитектурой модели
Архитектуры сравниваются друг с другом, это хорошо, но есть проблема. Xception и VGG сравниваются после 5 эпох, на которых эти сверточные сети фактически не обучались (trainable = False), обучались только последние полносвязные слои. Неудивительно что точность получается низкой, а такое сравнение (без обучения) вряд ли корректно. Но в целом VGG действительно хуже Xception, это устаревшая архитектура.
Оценка: 2 балла из 3.

Работа с процессом обучения
В первой версии вы использовали lr_find() и plot_lr(), значит приобрели навыки использования этих функций, что хорошо. Код обучения теперь построен верно: сначала обучаются последние слои, затем половина модели, затем вся модель. Есть и другой вариант: всю модель можно было начинать обучать с нуля.
Оценка: 3 балла из 3.

Работа с процессом инференса
Вижу что вы делали аугментацию тестовых данных (TTA), для этого создали test_datagen с аугментациями, но потом ее закомментировали. Что-то не получилось? Нет никаких комментариев. Но вижу что вы это пробовали. Нет ансамблирования, поскольку вы обучали только одну модель.
Оценка: 2 балла из 3.

Создание работающего прототипа
Этого пункта у вас не нашел. Его никто не делает практически :)
Сумма: 3+3+2+3+2=13 баллов.