```
# построим модель Xception, введем нормализацию
model = Sequential([
  Lambda(lambda x: x/255),
```

В разных моделях нужна разная нормализация, но почти нигда она не равна делению на 255. Например, для Xception требуется нормализация tf.keras.applications.xception.preprocess_input – это не то же самое, что деление на 255. Правильная нормализация (то есть такая же, какая была при предобучении) может немного улучшить точность, хотя это не принципиальный момент.

```
Dense(256, activation='relu'),
BatchNormalization(),
```

Эти слои в голову обычно не добавляют, но вы можете проверить – вдруг они повысят точность?

```
Dense(10, activation='softmax')

model.compile(
    loss=CategoricalCrossentropy(from_logits=True),
```

Нужно добавлять либо softmax, либо параметр from_logits=True, но не вместе – иначе получится двойной softmax, что может негативно сказаться на точности.

```
optimizer=Adam(ExponentialDecay(1e-3, 100, 0.9)),
```

Каждые 100 шагов LR умножается на 0.9, и в итоге после нескольких эпох он станет такой маленький, что сеть перестанет обучаться. Чтобы была возможность обучать дольше, можно использовать другие способы, например ReduceLROnPlateau.

```
ModelCheckpoint('best_model.hdf5' , monitor = ['val_accuracy'] , verbose =
 1   , mode = 'max')
```

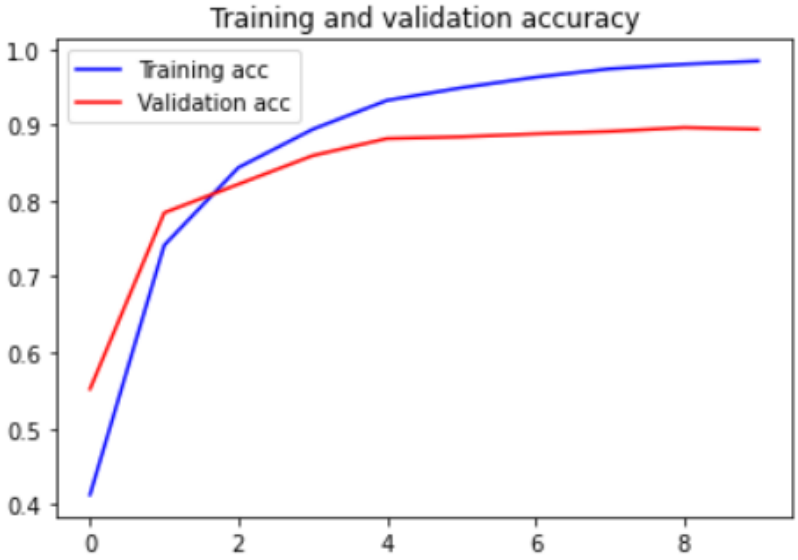
Здесь нужно добавить параметр save_best_only=True, иначе будет сохраняться каждую эпоху.

```
model.fit_generator(
```

Этот legacy-метод сейчас эквивалентент .fit().

> По графикам видим переобучение модели на втором шаге.

Обычно под переобучением понимают ситуацию, когда точность на валидации перестает расти. У вас же она растет, хоть и медленно, так что это еще не переобучение. Можно спокойно обучать дальше.




```
A.CLAHE(),
      A.HorizontalFlip(p=0.5),
      OneOf([
            MotionBlur(p=0.2),
            MedianBlur(blur_limit=3, p=0.1),
            Blur(blur_limit=3, p=0.1),
        ], p=0.2),
```

Обычно используют более простые аугментации, такие как сдвиг и случайный кроп. Сложный аугментации, как ни странно, часто не повышают точность, но при этом замедляют обучение из-за того, что на их выполнение также требуется время.

```
!pip install -q efficientnet
```

Модель EfficientNet встроена в модуль keras.applications, так что не обязательно устанавливать отдельную библиотеку.

Модель EfficientNet у вас плохо обучается (точность долго не растет выше 10%) потому что вы не делаете нормализацию изображений. Здесь есть тонкость: если брать EfficientNet из pip-пакета, то нужно делать нормализацию, если же брать из keras.applications, то не нужно.

Так как вы обучили несколько моделей, то вы могли бы усреднить их предсказания и тем самым повысить точность на сабмите.

В целом проблема в двойном softmax и отсутствии нормализации, а при обучении Xception – в слишком маленьком размере изображения. С размером 90x120 точность сложно поднять выше 90%. Но для учебных целей это не принципиально, а для продакшена нужно использовать изображения большего размера. Но обучаться при этом, конечно, будет дольше.