Мы обучали модель с параметрами batch_size=500, epochs=3, input_shape=(299,299,3)

Получили точность 70,07% на обучающем отрезке и 74,14% на валидационном.

```
# Тренировка модели
history = model.fit(train_data, batch_size=500, verbose=1, epochs= 3,
                    validation_data=val_data,
                    callbacks=[model_checkpoint_callback])
```

```
optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.RMSprop.

] - 140s 94ms/step - loss: 2.1852 - binary_accuracy: 0.6269 - val_loss: 0.6936 - val_binary_accuracy: 0.6663

eras/src/engine/training.py:3000: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This

] - 127s 88ms/step - loss: 1.0930 - binary_accuracy: 0.6532 - val_loss: 0.8638 - val_binary_accuracy: 0.6736

] - 129s 90ms/step - loss: 1.3552 - binary_accuracy: 0.7007 - val_loss: 0.5490 - val_binary_accuracy: 0.7414
```

Изображение из демонстрационных данных модель определила верно

```
[30] image_path = '/content/faces_splited/test/Male Faces/1 (148).jpg'
```

```
[22] # Классификация нового изображения
     from tensorflow.keras.preprocessing import image

     def classify_image(image_path):
         img = image.load_img(image_path, target_size=(299, 299))
         x = image.img_to_array(img)
         x = x / 255.0
         x = x.reshape((1, 299, 299, 3))
         pred = model.predict(x)
         print(pred)
         if pred[0][0] > 0.5:
             return 'men'
         else:
             return 'woman'
```

```
# Пример использования функции classify_image для классификации новых изображений


predicted_class = classify_image(image_path)
print(str(image_path), f'Predicted Class: {predicted_class}')
```

```
1/1 [==============================] - 0s 29ms/step
[[0.8054291]]
/content/faces_splited/test/Male Faces/1 (148).jpg Predicted Class: men
```

Менять будем гиперпараметры:

- batch_size (размер тренировочного окна)
- epochs (количество эпох обучения)
- input_shape (размерность)

На первом этапе изменим batch_size=550, epochs=4

(параметр input_shape=(299,299,3) оставляем неизменным от первоначального варианта)

Результат был значительно лучше: 82% точности на обучающей выборке и 87,68% точности на валидационной.

```
# Тренировка модели
history = model.fit(train_data, batch_size=550, verbose=1, epochs= 4,
                    validation_data=val_data,
                   callbacks=[model_checkpoint_callback])
```

```
 Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.RMSprop.

=======] - 133s 90ms/step - loss: 0.6310 - binary_accuracy: 0.7457 - val_loss: 0.5814 - val_binary_accuracy: 0.7611

=======] - 132s 91ms/step - loss: 0.5182 - binary_accuracy: 0.7868 - val_loss: 0.3816 - val_binary_accuracy: 0.8719

=======] - 131s 90ms/step - loss: 0.4974 - binary_accuracy: 0.8027 - val_loss: 0.9915 - val_binary_accuracy: 0.7845

=======] - 129s 89ms/step - loss: 0.4650 - binary_accuracy: 0.8200 - val_loss: 0.4766 - val_binary_accuracy: 0.8768
```

Изображение из демонстрационных данных модель определила верно:

```
[36] image_path = '/content/faces_splited/test/Female Faces/0 (1015).jpg'
```

```
[34] # Классификация нового изображения
     from tensorflow.keras.preprocessing import image

     def classify_image(image_path):
         img = image.load_img(image_path, target_size=(299, 299))
         x = image.img_to_array(img)
         x = x / 255.0
         x = x.reshape((1, 299, 299, 3))
         pred = model.predict(x)
         print(pred)
         if pred[0][0] > 0.5:
             return 'men'
         else:
             return 'woman'
```

```
# Пример использования функции classify_image для классификации новых изображений


predicted_class = classify_image(image_path)
print(str(image_path), f'Predicted Class: {predicted_class}')
```

```
1/1 [==============================] - 0s 20ms/step
[[3.6919734e-09]]
/content/faces_splited/test/Female Faces/0 (1015).jpg Predicted Class: woman
```

На втором этапе мы изменим параметры epochs=5, input_shape= (250,250,3)

(параметр batch_size=500 оставим, как в начальном варианте)

Результат следующий: 77,48% на обучающей выборке и 87,32% на валидационной.

```
# Тренировка модели
history = model.fit(train_data, batch_size=500, verbose=1, epochs= 5,
                    validation_data=val_data,
                    callbacks=[model_checkpoint_callback])
```

```
Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.RMSprop.
=======] - 129s 86ms/step - loss: 1.6534 - binary_accuracy: 0.5734 - val_loss: 0.5761 - val_binary_accuracy: 0.6675
=======] - 124s 86ms/step - loss: 1.4716 - binary_accuracy: 0.6470 - val_loss: 0.5250 - val_binary_accuracy: 0.7131
=======] - 124s 86ms/step - loss: 0.8402 - binary_accuracy: 0.6920 - val_loss: 0.5087 - val_binary_accuracy: 0.7254
=======] - 121s 83ms/step - loss: 0.6541 - binary_accuracy: 0.7464 - val_loss: 0.4173 - val_binary_accuracy: 0.8177
=======] - 123s 85ms/step - loss: 0.5467 - binary_accuracy: 0.7748 - val_loss: 0.4127 - val_binary_accuracy: 0.8732
```

Результат на демонстрационном изображении – верный:

```
[45] image_path = '/content/faces_splited/test/Female Faces/0 (1017).jpg'
```

```
[43] # Классификация нового изображения
     from tensorflow.keras.preprocessing import image

     def classify_image(image_path):
         img = image.load_img(image_path, target_size=(250, 250))
         x = image.img_to_array(img)
         x = x / 255.0
         x = x.reshape((1, 250, 250, 3))
         pred = model.predict(x)
         print(pred)
         if pred[0][0] > 0.5:
             return 'men'
         else:
             return 'woman'
```

```
# Пример использования функции classify_image для классификации новых изображений


predicted_class = classify_image(image_path)
print(str(image_path), f'Predicted Class: {predicted_class}')
```

```
1/1 [==============================] - 0s 30ms/step
[[0.11799034]]
/content/faces_splited/test/Female Faces/0 (1017).jpg Predicted Class: woman
```

На третьем этапе мы изменим параметры batch_size=600 и input_shape=(200,200,3)

(параметр epochs=3 оставим, как в начальном варианте)

Результат следующий: 74,67% - обучающая, 80,91% - валидационная.

```
# Тренировка модели
history = model.fit(train_data, batch_size=600, verbose=1, epochs= 3,
                    validation_data=val_data,
                   callbacks=[model_checkpoint_callback])
```

```
Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.RMSprop.
=======] - 126s 85ms/step - loss: 1.4583 - binary_accuracy: 0.5865 - val_loss: 0.6500 - val_binary_accuracy: 0.6810
=======] - 120s 83ms/step - loss: 0.7573 - binary_accuracy: 0.6802 - val_loss: 0.4541 - val_binary_accuracy: 0.7845
=======] - 122s 84ms/step - loss: 0.6112 - binary_accuracy: 0.7467 - val_loss: 0.4222 - val_binary_accuracy: 0.8091
```

Результат предсказания демонстрационного изображения – верный.

```
[51] image_path = '/content/faces_splited/test/Female Faces/0 (1019).jpg'
```

```
[52] # Классификация нового изображения
     from tensorflow.keras.preprocessing import image

     def classify_image(image_path):
         img = image.load_img(image_path, target_size=(200, 200))
         x = image.img_to_array(img)
         x = x / 255.0
         x = x.reshape((1, 200, 200, 3))
         pred = model.predict(x)
         print(pred)
         if pred[0][0] > 0.5:
             return 'men'
         else:
             return 'woman'
```

```
# Пример использования функции classify_image для классификации новых изображений


predicted_class = classify_image(image_path)
print(str(image_path), f'Predicted Class: {predicted_class}')
```

```
1/1 [==============================] - 0s 252ms/step
[[0.02515546]]
/content/faces_splited/test/Female Faces/0 (1019).jpg Predicted Class: woman
```

Выводы

Из полученных данных следует вывод, что увеличение количества эпох положительно сказывается на точности модели.

Увеличение значений параметра batch_size так же давал положительный результат.

А вот снижение размерности влияет скорее всего не слишком хорошо и ведёт к упрощению модели.