Распознавание породы кошек V2

Изменения:

1. Добавлена аугментация изображений
2. Использование Fine-tuning
3. Изменен формат обрабатываемых изображений(float64 -> uint8)

Добавление аугментации

Функция, с помощью которой была проведена аугментация

```
def augment_image(image_path, target_width = 224, target_height = 224):
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    seq = iaa.Sequential([
        iaa.Fliplr(0.5),
        iaa.Affine(rotate=(-20, 20)),
        iaa.GaussianBlur(sigma=(0, 1.0)),
        iaa.AdditiveGaussianNoise(scale=(0, 0.05*255))
    ], random_order=True)
    new_image = seq(image=image)
    resized_image = cv2.resize(new_image, (target_width, target_height))
    return resized_image
```

```
for files in train_files:
    preprocessed_image = preprocess_image(dataset_path + '\\' + file_name + '\\' + files)
    train_data.append(preprocessed_image)
    train_labels.append([class_mapping[file_name]])
    augmented_image = augment_image(dataset_path + '\\' + file_name + '\\' + files)
    train_data.append(augmented_image)
    train_labels.append([class_mapping[file_name]])

    favorite_files.remove(files)
```

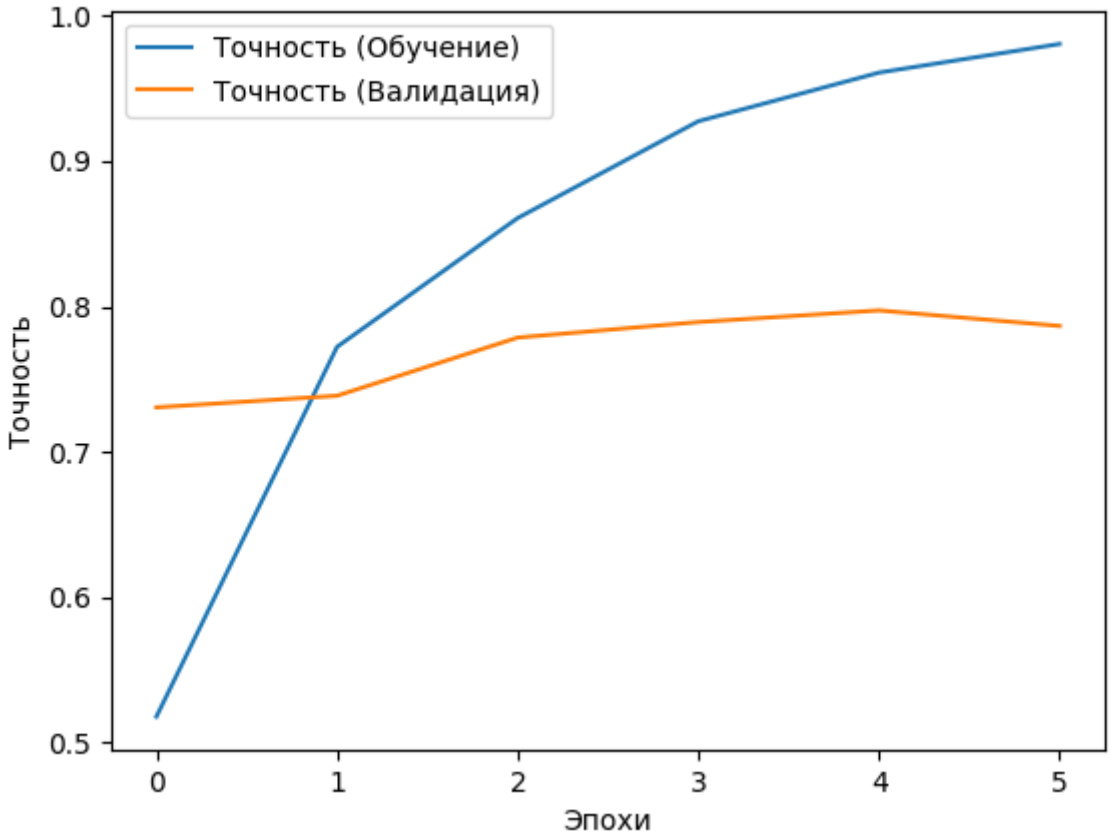
Fine-tuning

Размораживаем несколько последних слоев. В своей задаче я разморозил 15 слоев.

```
fine_tune_at = 15
for layer in base_model.layers[:fine_tune_at]:
        layer.trainable = False
```

Анализ результатов:

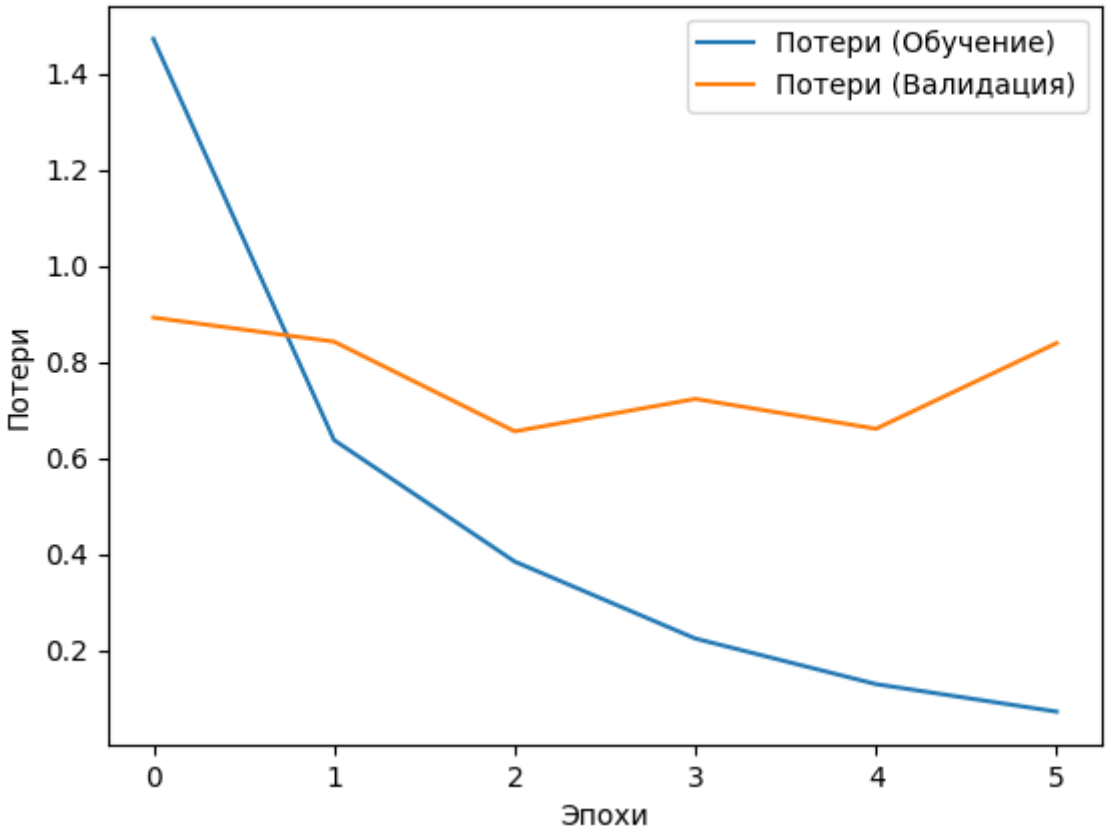
После внесения данных изменений были получены следующие результаты для 500 фотографий для каждой породы.
Процесс обучения
Epoch 1/10
110/110 ━━ 1150s 10s/step - accuracy: 0.3959 - loss: 2.0268 - val_accuracy: 0.7307 - val_loss: 0.8928
Epoch 2/10
110/110 ━━ 924s 8s/step - accuracy: 0.7569 - loss: 0.6797 - val_accuracy: 0.7387 - val_loss: 0.8428
Epoch 3/10
110/110 ━━ 930s 8s/step - accuracy: 0.8502 - loss: 0.4119 - val_accuracy: 0.7787 - val_loss: 0.6559
Epoch 4/10
110/110 ━━ 904s 8s/step - accuracy: 0.9330 - loss: 0.2095 - val_accuracy: 0.7893 - val_loss: 0.7235
Epoch 5/10
110/110 ━━ 886s 8s/step - accuracy: 0.9580 - loss: 0.1344 - val_accuracy: 0.7973 - val_loss: 0.6613
Epoch 6/10
110/110 ━━ 885s 8s/step - accuracy: 0.9789 - loss: 0.0758 - val_accuracy: 0.7867 - val_loss: 0.8396
12/12 ━━ 72s 6s/step - accuracy: 0.7795 - loss: 0.6791
Test Accuracy: 76.80%
Test Loss: 70.19%



Как мы видим по графику, точность на обучающем наборе данных продолжает расти, а точность валидационном наборе данных остается примерно **const.**

Это говорит нам о том, что происходит переобучение модели. Можно также рассмотреть график потерь, который тоже свидетельствует о переобучении

модели.



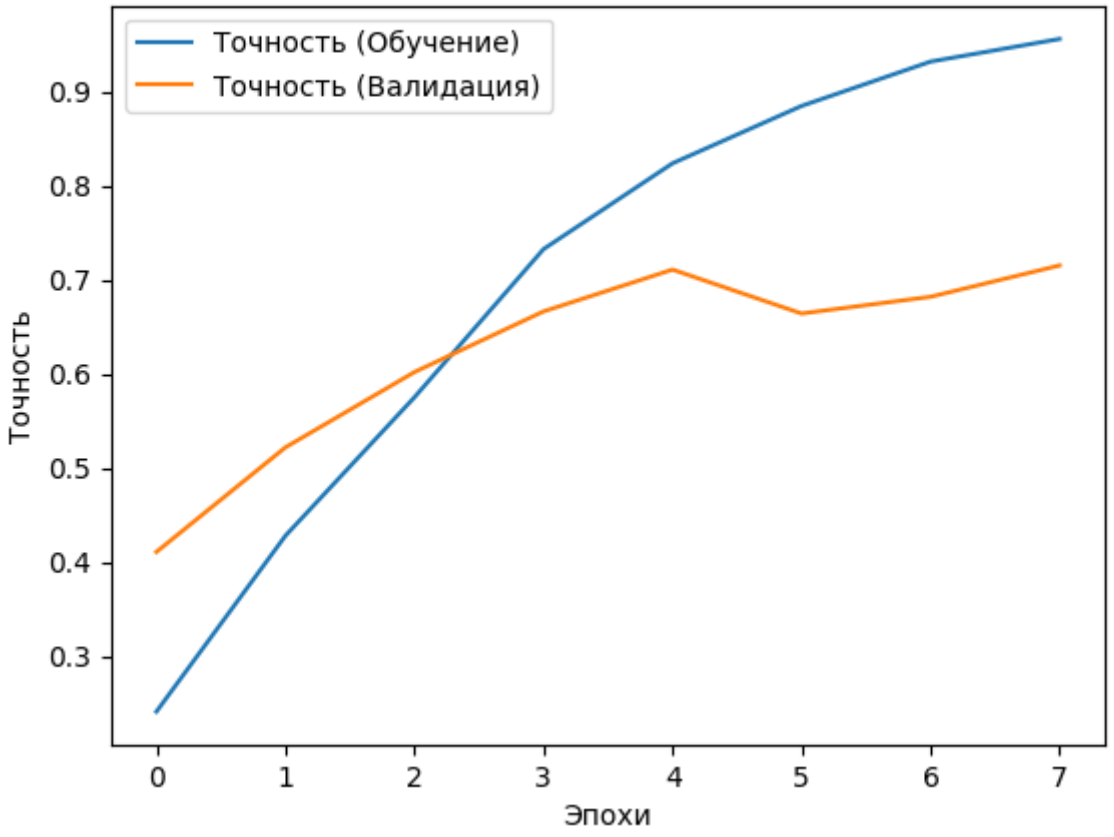
Изменения:

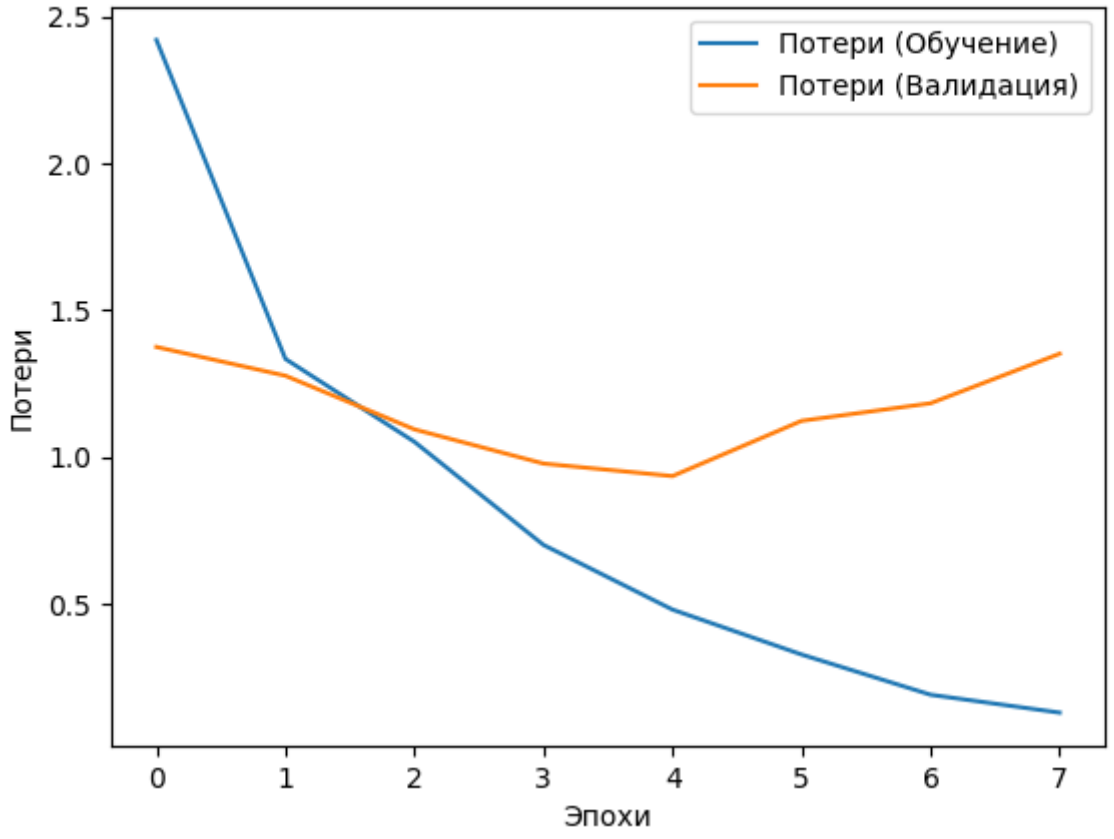
1. feature-exctracting
2. чуть-чуть увеличили количество данных(на 100 шт. на одну породу)

Анализ результатов:

Процесс обучения
Epoch 1/10
132/132 ——————————————————————— 1133s 9s/step - accuracy: 0.2148 - loss: 4.9069 - val_accuracy: 0.4111 - val_loss: 1.3756
Epoch 2/10
132/132 ——————————————————————— 1210s 9s/step - accuracy: 0.3952 - loss: 1.3944 - val_accuracy: 0.5222 - val_loss: 1.2776
Epoch 3/10
132/132 ——————————————————————— 1118s 8s/step - accuracy: 0.5486 - loss: 1.1020 - val_accuracy: 0.6022 - val_loss: 1.0950
Epoch 4/10
132/132 ——————————————————————— 1151s 9s/step - accuracy:

0.7167 - loss: 0.7212 - val_accuracy: 0.6667 - val_loss: 0.9786
Epoch 5/10
132/132 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1138s 9s/step - accuracy:
0.8048 - loss: 0.4993 - val_accuracy: 0.7111 - val_loss: 0.9360
Epoch 6/10
132/132 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1059s 8s/step - accuracy:
0.8909 - loss: 0.3131 - val_accuracy: 0.6644 - val_loss: 1.1240
Epoch 7/10
132/132 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1060s 8s/step - accuracy:
0.9305 - loss: 0.1954 - val_accuracy: 0.6822 - val_loss: 1.1838
Epoch 8/10
132/132 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1056s 8s/step - accuracy:
0.9612 - loss: 0.1254 - val_accuracy: 0.7156 - val_loss: 1.3524
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85s 6s/step - accuracy:
0.6860 - loss: 0.9494
Test Accuracy: 68.89%
Test Loss: 96.35%





Исходя из результатов мы видим, что до сих пор имеется проблема переобучения модели.

Изменения:

1. Использование ImageDataGenerator
2. Замена прямой загрузки массивов данных на использование генераторов данных 'flow()' для обучения, валидации и тестирования модели
   Анализ результатов:
   Точность на валидационном наборе данных увеличивается с каждой эпохой и достигает 71.68% к концу обучения. Точность на тестовом наборе данных составляет 71.68%. Это свидетельствует о том. что модель не переобучилась и хорошо обобщает данные.

Процесс обучения:

```
Epoch 1/10
C:\python\Lib\site-
packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:120:
UserWarning: Your `PyDataset` class should call
`super().__init__(**kwargs)` in its constructor. `**kwargs` can include
`workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these
arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
76/76 ━━━━━━━━━━━━━━━━━━━━ 687s 9s/step - accuracy: 0.3114 - loss:
1.6339 - val_accuracy: 0.5977 - val_loss: 0.9903
Epoch 2/10
 1/76 ━━━━━━━━━━━━━━━━━━━━ 9:47 8s/step - accuracy: 0.5938 - loss:
0.98552024-04-14 23:03:56.616161: W
tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is
aborting with status: OUT_OF_RANGE
: End of sequence
         [[{{node IteratorGetNext}}]]
C:\python\Lib\contextlib.py:158: UserWarning: Your input ran out of
data; interrupting training. Make sure that your dataset or generator
can generate at least `steps_per_epoch * epochs` batches. You may need
 to use the `.repeat()` function when building your dataset.
  self.gen.throw(typ, value, traceback)
2024-04-14 23:04:00.094901: W
tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is
aborting with status: OUT_OF_RANGE: End of sequence
```

```
76/76 ━━━━━━━━━━━━━━━━━━━━ 11s 47ms/step - accuracy: 0.5938 - loss:
0.9855 - val_accuracy: 0.7222 - val_loss: 1.0333
Epoch 3/10
76/76 ━━━━━━━━━━━━━━━━━━━━ 656s 9s/step - accuracy: 0.5961 - loss:
1.0151 - val_accuracy: 0.6562 - val_loss: 0.8421
Epoch 4/10
 1/76 ━━━━━━━━━━━━━━━━━━━━ 8:57 7s/step - accuracy: 0.6875 - loss:
0.90382024-04-14 23:15:03.322791: W
tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is
aborting with status: OUT_OF_RANGE
: End of sequence
         [[{{node IteratorGetNext}}]]
2024-04-14 23:15:06.701311: W
tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is
aborting with status: OUT_OF_RANGE: End of sequence
76/76 ━━━━━━━━━━━━━━━━━━━━ 11s 45ms/step - accuracy: 0.6875 - loss:
0.9038 - val_accuracy: 0.6111 - val_loss: 1.4003
Epoch 5/10
76/76 ━━━━━━━━━━━━━━━━━━━━ 653s 9s/step - accuracy: 0.6969 - loss:
0.8391 - val_accuracy: 0.7109 - val_loss: 0.7718
Epoch 6/10
 1/76 ━━━━━━━━━━━━━━━━━━━━ 9:01 7s/step - accuracy: 0.8438 - loss:
0.46552024-04-14 23:26:06.659123: W
tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is
aborting with status: OUT_OF_RANGE
: End of sequence
         [[{{node IteratorGetNext}}]]
2024-04-14 23:26:10.092833: W
tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is
aborting with status: OUT_OF_RANGE: End of sequence
76/76 ━━━━━━━━━━━━━━━━━━━━ 11s 47ms/step - accuracy: 0.8438 - loss:
0.4655 - val_accuracy: 0.7778 - val_loss: 0.6897
Epoch 7/10
76/76 ━━━━━━━━━━━━━━━━━━━━ 652s 9s/step - accuracy: 0.7330 - loss:
0.7206 - val_accuracy: 0.7051 - val_loss: 0.7858
Epoch 8/10
 1/76 ━━━━━━━━━━━━━━━━━━━━ 8:52 7s/step - accuracy: 0.6875 - loss:
0.70182024-04-14 23:37:09.441015: W
```

```
tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is
aborting with status: OUT_OF_RANGE
: End of sequence
         [[{{node IteratorGetNext}}]]
2024-04-14 23:37:12.801500: W
tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is
aborting with status: OUT_OF_RANGE: End of sequence
76/76 ━━━━━━━━━━━━━━━━━━━━ 10s 45ms/step - accuracy: 0.6875 - loss:
0.7018 - val_accuracy: 0.8333 - val_loss: 0.7067
Epoch 9/10
76/76 ━━━━━━━━━━━━━━━━━━━━ 653s 9s/step - accuracy: 0.7490 - loss:
0.6818 - val_accuracy: 0.7168 - val_loss: 0.7682
16/16 ━━━━━━━━━━━━━━━━━━━━ 95s 6s/step - accuracy: 0.7078 - loss: 0.8111
```

Также был испробована предобученная модель EfficientNetB0
Но результаты показаны были хуже, поэтому не будем освещать их.

Итого:
После добавления fine-tuning и аугментации данных, мы получили хорошие результаты по сравнению с теми, что у нас были прежде.
Test accuracy: 63.88% --> 76.80%
Оставалась проблема переобучения модели.
После внесения feature-exctracting:
Test accuracy: 76.80% --> 68.89%
Точность росла, но затем падала =>переобучение

Использование ImageDataGenerator и добавление генераторов данных.
Дальше мы использовали `ImageDataGenerator` из библиотеки Keras, который автоматически применяет аугментацию данных во время обучения модели. Также ввели для этого генераторы данных, которые автоматически выполняют аугментацию изображений и предоставляют данные в модель небольшими пакетами (batch) во время обучения.

Что можно попробовать сделать еще?
Есть идея использовать ансамбль методов, но задача станет тогда более объемной.