Отчет о работе над проектом

Задание

Написать консольную утилиту для многопоточного скачивания изображений, их последующей обработки и классификации. Обработка файлов заключается в изменении размера (требуемый размер 224x224) и перекодировании в формат PNG. Классификация заключается в сохранении в соответствующую изображению директорию.

Алгоритм решения

Блок-схема алгоритма решения поставленной задачи приведена на рис. 1.

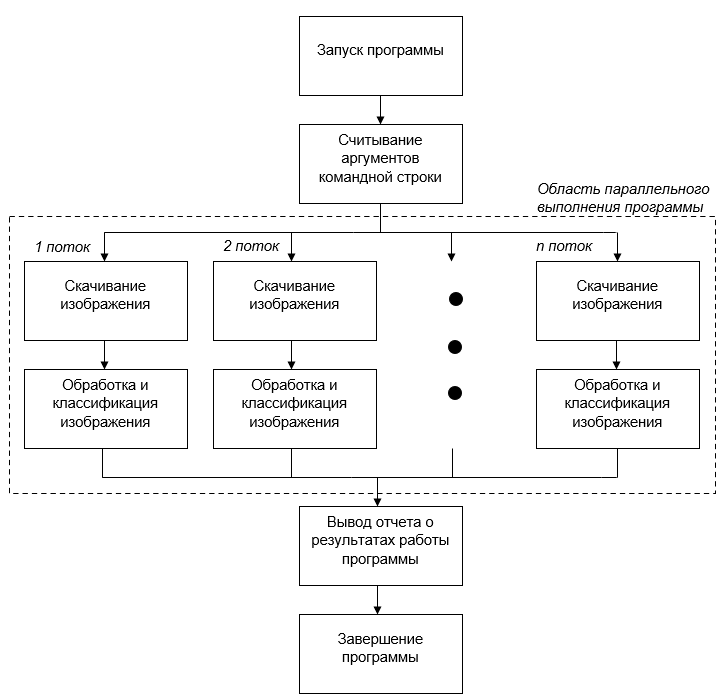


Рис. 1. Блок-схема алгоритма

Решение программы состоит из следующих этапов:

1. Считывание аргументов командной строки с помощью библиотеки **argparse.py**. В качестве обязательного параметра считывается название файла, содержащего список ссылок на изображения для скачивания. Необязательным аргументов является число потоков для решения задачи. В случае, когда число потоков отсутствует, программа выполняется последовательно.
2. Параллельное скачивание изображений. Получение изображений по ссылкам происходит при помощи объекта **request** библиотеки **urllib**. Реализация параллелизма осуществляется использования **ThreadPoolExecutor** библиотеки **concurrent**.
3. Обработка изображений согласно требованиям классификатора. Для хранения и работы с изображениями используется объект **Image** библиотеки **Pillow**. Классификатор принимает изображения размера 224x224 RGB. Для изменения размеров изображения используется метод **resize**, для удовлетворения второго требования – метод mode с параметром RGB. Преобразования картинок реализовано в функции standardize\_img
4. Классификация модели при помощи классификатора. Классификатор предоставляется заранее и представляет из себя обученную модель библиотеки **tensorflow**, умеющую распознавать изображения котов. Классифицируемые объекты сохраняются в соответствующие папки: в директорию с котиками, если на изображении кот, и в директорию с собаками иначе.
5. По завершению скачивания, обработки и классификации изображений выводится подробный отчет о результатах работы программы. Для подсчета числа котов используется класс **Counter** модуля **collections**

Функции тестирования

Реализованы тесты для проверки работы функции преобразования. Функции скачивания или работы с классификаторами тестами не покрывались. Для выполнения тестов создается папка с тестовыми изображениями и проверяется формат обработанных после вызова функций standardize\_img картинок. Запуск тестов осуществляется посредством вызова pytest из корневого каталога проекта.

Трудности, возникшие при реализации

1. При реализации параллелизма посредством ThreadPoolExecutor требуется, чтобы работа функции, выполнение которой требуется распараллелить, принимала лишь один аргумент. Однако, помимо ссылок для скачивания изображений, считываемых из файла, необходимо в качестве аргумента принимать заранее подгруженную модель для классификации. Для решения использовалась функция **partial** модуля **functools.** Использование данной функции позволяет создать функцию подкачки изображений с заранее загруженной модель классификатора
2. Классификатор изначально не работал на машине, где выполнялись разработка и тестирование программы. Для работы классификатора помимо установки пакетов **tensorflow** и **keras** необходимо наличие **CUDA 10**. Данный факт отражен в README.md отдельно от приведенного списка зависимостей requirements.txt
3. Файлы сохраняются под именами, соответствующими их номеру строки в списке URL, незначащие разряды заполняются нулями. Для удовлетворения условия необходимо считать файл заранее для подсчета общего числа ссылок с целью определения числа нулей.

Резюме

Реализована программа для параллельного скачивания и классификации изображений. Изучены средства библиотеки argparse для работы с аргументами командной строки: задания параметра по умолчанию, определение типа подаваемого аргумента, задания параметра по умолчанию, задание короткого и длинного названий опциональных параметров. В ходе реализации параллелизма получены базовые навыки при работе c ThreadPoolExecutor. Исследованы возможность применения классификатора.