ТЗ проекта classi_face_cation

Тарасов Роман, Комаров Артём, Павливский Сергей

1 Кратко о проекте

На наборе размеченных данных мы исследовали точность различных известных моделей Сверточных Нейронных Сетей (CNN). Большая часть моделей в силу исследовательского характера проекта была реализована на Python3.7, из-за требований к проекту также некоторые из моделей были реализованы на C++ с использованием сторонних библиотек.

2 Python3.7 документация

Для использования необходимо:

- 32-х или 64-х разрядная операционная система Linux/Windows/Mac. Код обладает полной переносимостью
- Не менее 8 GB оперативной памяти. В случае желания перенести вычисления на видеокарту также не менее 2 GB видеопамяти, видеокарта Nvidia поколения G8x и выше.
- Требуется наличие библиотек:
  - NumPy
  - Torch, Torch.nn
  - Pandas
  - Matplotlib
  - tqdm
  - skimage
  - PIL
  - torchvision, torchvision.transforms
  - Возможна необходимость в установлении еще других библиотек, в зависимости от предустановленных в рабочей среде

Работа над проектом велась в Jupyter Notebook, для возможности просматривать его исходные коды требуется установить Miniconda с официального сайта `https://conda.io/en/latest/miniconda.html`, все вышеописанные библиотеки устанавливаются из командной строки с использованием conda.

2.1 Запуск

Запускать файлы jpynb можно в Google Colaboratory `https://colab.research.google.com`. Перед запуском нужно загрузить на Google Drive файл fer2013.csv.

3 Датасет

Набор данных можно скачать по ссылке: `https://www.kaggle.com/ahmedmoorsy/facial-expression/download`.

Датасет состоит из 35887 фотографий людей размера 48x48, по одному каналу на изображения (черно-белые изображения). На каждой фотографии по одному человеку, и также указана 7 классовая разметка по типам настроения человека на фотографии.

4 C++ документация

Для запуска необходима ОС Linux.

4.1 Библиотека Libtorch

Нужно загрузить библиотеку Libtorch. Для Linux это можно сделать по ссылке `https://download.pytorch.org/libtorch/cpu/libtorch-cxx11-abi-shared-with-d5.0%2Bcpu.zip`. Распаковываем скачанный архив в некоторую директорию /absolute/path/to/libtorch.

4.2 Сборка

Для сборки необходима программа CMake.

В некоторую папку /current/dir загружаем файлы проекта. Здесь же создаём папку build.

В папку /current/dir нужно поместить CMakeLists.txt. На репозитории он также выложен. Затем для сборки и компиляции из папки build нужно выполнить команды:

```
cmake -DCMAKE_PREFIX_PATH=/absolute/path/to/libtorch ..
cmake --build . --config Release
```

Инструкция по установке также есть на официальном сайте `https://pytorch.org/cppdocs/installing.html`

4.3 Запуск

Загружаем в папку build файл с данными fer2013.csv

Из папки build запускаем программу

```
./classi_face_cation
```

Затем после обработки датасета программа предложит ввести число от 1 до 5. В зависимости от него будет выбрана модель нейронной сети, на которой будет происходить обучение.

- 1 — Слегка изменённая модель AlexNet.
- 2 — Модель OurNet
- 3 — Модель SmallNet с малым количеством параметров
- 4 — VGG16
- 5 — Упрощённая модель ResNet18

Затем начнётся обучение, во время которого после каждого обработанного батча на экран будут выводиться номер эпохи, номер батча и значение лосс-функции.

После окончания обучения будет проведена валидация на тестовом датасете. Для каждого изображения из test_images будут выведены 7 вероятностей принадлежности каждому из типов и правильный ответ. А затем будет подсчитана доля правильных ответов сети.

5 Ссылка на репозиторий

`https://github.com/ArtKomarov/classi_face_cation`

6 Авторство

Реализация моделей на Python: Сергей

Эксперименты с архитектурой AlexNet: Роман и Артём

Модули nets.cpp, nets.hpp: первая модель – Артём, а остальное Роман

Модуль main.cpp: обобщение на все нейросети – Роман, остальное Артём

Модули customdataset.cpp, customdataset.hpp, clfc.hpp: Артём

Обработка ошибок: Артём

ТЗ: Роман, Сергей

7 Структура проекта

nets.cpp, nets.hpp содержат описания и реализацию классов AlexNet, Ournet, SmallNet, VGG16, ResNet18 и их базового класса Net, который в свою очередь унаследован от torch::nn::Module. В каждом из этих классов содержится архитектура сети и функция forward, которая прогоняет входной тензор через сеть.

customdataset.cpp, customdataset.hpp содержат класс CustomDataset, который считывает данные из датасета и заносит их в тензоры.

clfc.hpp содержит константы, характеризующие датасет.

В main.cpp реализовано обучение сети и проверка на тестовой выборке.