Нижегородский государственный университет им. Н.И. Лобачевского
Институт информационных технологий, математики и механики
Кафедра Математического обеспечения и суперкомпьютерных технологий

Образовательный курс «Введение в глубокое обучение с использованием Intel® neon™ Framework»

Практическая работа №0 Предварительная обработка и конвертация данных в формат HDF5 для работы с Intel® neon™ Framework

При поддержке компании Intel

Жильцов М.С.

Нижний Новгород
2018

Содержание

1 Введение....3
2 Методические указания....3
2.1 Цели и задачи работы....3
2.2 Структура работы....3
2.3 Рекомендации по проведению занятий....3
3 Инструкция по выполнению работы....4
3.1 Установка Intel® neon™ Framework и зависимостей....4
3.1.1 Перечень зависимостей....4
3.1.2 Установка Python 3....4
3.1.3 Установка Intel® neon™ Framework....4
3.1.4 Установка дополнительных модулей Python....5
3.2 Настройка окружения....5
3.2.1 Использование Intel® Math Kernel Library....5
3.2.2 Использование NVIDIA CUDA....5
3.2.3 Разрешение доступа к компонентам....5
3.3 Подготовка данных для выполнения практических работ....5
3.3.1 Предварительная обработка визуальных данных и организации работы с данными в Intel® neon™ Framework....5
3.3.2 Подготовка набора IMDB-WIKI для решения задачи распознавания пола....6
4 Литература....9
4.1 Основная литература....9
4.2 Ресурсы сети Интернет....9

1 Введение

Настоящая практическая работа является подготовительной и позволяет создать инфраструктуру для выполнения последующих работ. Здесь дается описание процедуры установки инструмента Intel® neon™ Framework [4] и настройки окружения для запуска примеров обучения и тестирования глубоких нейросетевых моделей.

Выполнение всех практических работ демонстрируется на примере задачи классификации пола (мужской, женский) человека по фотографии. Таким образом, далее решается задача классификации с двумя категориями. В качестве тренировочных и тестовых данных используется набор IMDB-WIKI [5]. Для указанного набора в текущей работе выполняется предварительная обработка изображений, а также преобразование данных и разметки в формат HDF5, принимаемый инструментом Neon. В следующих работах предполагается, что набор данных подготовлен и его необходимо только загрузить. Процедура предварительной обработки данных и преобразования в формат HDF5 может быть перенесена на случай другой задачи и соответственно других данных.

2 Методические указания

2.1 Цели и задачи работы

Цель данной работы состоит в получении общего представления о работе с инструментом Intel® neon™ Framework и подготовке рабочего окружения для выполнения последующих практических работ.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Установить Intel® neon™ Framework и зависимости.
2. Настроить рабочее окружение.
3. Изучить структуру и состав Intel® neon™ Framework.
4. Подготовить данные для выполнения последующих работ.

2.2 Структура работы

В работе приводится руководство по установке Intel® neon™ Framework и необходимых зависимостей. Руководство включает описание последовательности действий, которую следует выполнить из командной строки для установки Neon и настройки рабочего окружения для проведения экспериментов по обучению и тестированию глубоких моделей. Далее поэтапно разрабатывается программный код, обеспечивающий загрузку данных IMDB-WIKI [5] для решения задачи классификации пола человека по фотографии.

2.3 Рекомендации по проведению занятий

При выполнении данной практической работы рекомендуется следующая последовательность действий:

1. Выполнить установку Intel® neon™ Framework и зависимостей.
2. Настроить рабочее окружение.
3. Изучить структуру и состав Intel® neon™ Framework, опираясь на лекционный материал курса и дополнительные источники.
4. Разработать программный код для подготовки данных с целью выполнения последующих работ, проверить его работоспособность.

3 Инструкция по выполнению работы

3.1 Установка Intel® neon™ Framework и зависимостей

3.1.1 Перечень зависимостей

Intel® neon™ Framework предназначен для языка Python 2 и 3, функционирует на платформах Linux и Mac OS. Допускает работу на CPU с возможным использованием библиотеки Intel® Math Kernel Library и на GPU с использованием NVIDIA CUDA.

В перечень зависимостей инструмента входят следующие библиотеки и модули:

- Python 2.7+ / 3.4+ [6],
- Python-pip [7],
- Python-virtualenv [8],
- libhdf5-dev [9],
- (опционально) NVIDIA CUDA (8.0) [10],
- (опционально) Intel® Math Kernel Library (Intel® MKL) [11].

Далее приводится последовательность команд для установки Neon под Linux. Дополнительные инструкции по установке можно найти в официальной документации Intel® neon™ Framework [12].

3.1.2 Установка Python 3

Для установки и сборки Python 3 потребуется скачать исходный код интерпретатора [13]. Сборка выполняется стандартным набором шагов.

```
./configure
make
make install
```

Различные параметры сборки можно посмотреть, вызвав команду:

```
./configure --help
```

Также необходимо обновить модуль **`pip`** и установить модуль **`virtualenv`**. Для этого можно использовать следующие команды:

```
pip install --upgrade pip
pip install virtualenv
```

3.1.3 Установка Intel® neon™ Framework

Первым шагом установки Neon является скачивание и сборка исходных кодов. Для этого можно использовать нижеперечисленные команды.

```
git clone https://github.com/NervanaSystems/neon.git
cd neon; git checkout latest; make python3
```

После сборки и установки в директории сборки будет создана директория виртуального окружения **`.venv`**, содержащего все необходимые для работы библиотеки модули Python. Для удобства в дальнейшей работе рекомендуется создать в директории практических работ копию или символьную ссылку для директории виртуального окружения.

```
cp neon_source_path/.venv .venv # для создания копии
ln -s neon_source_path/.venv .venv # для создания ссылки
```

Для использования виртуального окружения его необходимо активировать. Для этого следует выполнить команду, приведенную ниже.

```
. .venv/bin/activate
```

При этом, вид командной строки должен измениться. С этого момента команды **`python`**, **`pip`** и другие относятся к интерпретатору, расположенному в виртуальном окружении. Модули Python будут скопированы из исходного интерпретатора. Завершение использования виртуального окружения выполняется командой деактивации.

```
deactivate
```

3.1.4 Установка дополнительных модулей Python

Для выполнения практических работ потребуется несколько дополнительных модулей. Их можно установить командой, представленной ниже. Предварительно необходимо активировать виртуальное окружение.

```
pip install -r Practice/requirements.txt
```

3.2 Настройка окружения

3.2.1 Использование Intel® Math Kernel Library

Если планируется использование Intel® MKL [11], то для улучшения производительности могут быть полезны следующие переменные окружения:

```
export KMP_AFFINITY=compact,1,0,granularity=fine
export OMP_NUM_THREADS=<Number of Physical Cores>
```

Переменная **`KMP_AFFINITY`** задает способ назначения рабочих потоков библиотеки MKL на физические ядра процессора. Переменная **`OMP_NUM_THREADS`** фиксирует максимальное число рабочих потоков, используемых OpenMP. Дополнительную информацию по этим параметрам можно найти на сайте Intel® MKL [14].

3.2.2 Использование NVIDIA CUDA

Если планируется использование NVIDIA CUDA [10] и GPU, то требуется установить следующие переменные окружения:

```
export PATH=/usr/local/cuda/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:\
         /usr/local/cuda/lib:/usr/local/lib:$LD_LIBRARY_PATH
```

Данные переменные отвечают за пути поиска исполняемых файлов и библиотек. Для удобства, указанные переменные можно добавить в файл **`~./bashrc`**, чтобы они автоматически устанавливались при входе пользователя.

3.2.3 Разрешение доступа к компонентам

Для использования компонентов практических работ необходимо добавить их в пути поиска модулей Python.

```
export PYTHONPATH=/<full_path_to_practice>:$PYTHONPATH
```

3.3 Подготовка данных для выполнения практических работ

3.3.1 Предварительная обработка визуальных данных и организации работы с данными в Intel® neon™ Framework

Предварительная обработка визуальных данных может предполагать выполнение следующих преобразований:

- Приведение изображений к одному размеру посредством выреза, масштабирования или дополнения.
- Вычитание среднего значения пикселей, деление на среднеквадратичное отклонение.
- Удаление некорректных входных и выходных данных.
- Искусственное расширение набора посредством отражений, поворотов, сдвигов, масштабирования.
- Добавление шума к изображениям.
- Балансировка классов в задаче классификации.

В процессе подготовки данных также выполняется разделение набор на тренировочное и тестовое множества, если они изначально не заданы.

Результатом данного этапа являются данные в формате, совместимом с используемой библиотекой. Intel® neon™ Framework позволяет работать с данными несколькими способами [15]:

- Хранение всех данных в памяти устройства (тип **`ArrayIterator`**).

– Хранение данных во внешней памяти, загрузка небольших порций данных в память устройства (тип **`HDF5Iterator`** и загрузчик Aeon [16]).

Также для широко известных наборов данных предоставляются встроенные загрузчики [17]. Данные передаются в Neon посредством итераторов. В рамках данной работы приводятся примеры использования типа **`HDF5Iterator`** и создания собственного итератора.

3.3.2 Подготовка набора IMDB-WIKI для решения задачи распознавания пола

В процессе выполнения практических работ используется набор данных IMDB-WIKI [5]. Набор содержит около 60000 изображений. Наряду с изображениями набор содержит метаданные, в состав которых входит различная дополнительная информация, такая, как позиция лица, пол человека на фотографии, время съемки. Размер изображений в наборе не фиксирован и варьируется от 32x32 до 512x512, поэтому необходимы дополнительные действия по подготовке изображений.

1. Загрузка набора данных. Выполняется посредством вызова двух команд, приведенных ниже.

```
# ~1 ГБ данных
wget https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/static/wiki_crop.tar
tar -xf wiki_crop.tar
```

2. Приведение изображений к одному размеру. На этом шаге используется комбинация выреза из центра изображения для больших изображений и дополнения нейтральными данными для маленьких изображений. Подход не гарантирует, что в результате выреза лицо останется на изображении, но при достаточно большом вырезе по отношению к размеру изображения на присутствие лица можно рассчитывать с высокой вероятностью. Более правильным подходом могло бы быть использование данных о положении лица из метаданных. Файл метаданных **`wiki_crop/wiki.mat`** является файлом Matlab и может быть открыт с помощью модуля SciPy [18]. Файл содержит ассоциативный массив переменных, в состав которого входит переменная **`wiki`**. Переменная является структурой, содержащей поля, определенные в метаданных [5]. С точки зрения задачи классификации пола представляют интерес следующие поля:
   – Поле с индексом 2 содержит массив путей до входных изображений. Форма массива $1xN$, где N – количество изображений.
   – Поле с индексом 3 содержит массив с метками пола. Значения – 0, 1 и **`NaN`** типа **`float`**. Форма массива $1xN$.

   Для работы с изображениями используется модуль Pillow [19]. Работа с массивами данных осуществляется посредством модуля NumPy [20].

```
from scipy.io import loadmat
import numpy as np
from PIL import Image
```

Используя описание файла с метаданными [5], загрузим изображение и метку пола:

```
metadata = loadmat(dataset_root + '/wiki_crop.mat')['wiki'][0][0]

i = 0 # номер элемента набора данных
image = Image.open(dataset_root + '/' + metadata[2][0][i][0])
gender = metadata[3][0][i].astype(int)
```

Преобразуем все данные к формату RGB:

```
if (len(image.getbands()) != 3):
    image = image.convert("RGB")
```

Сделаем вырез из центра или дополнение изображения нулями:

```
def center_crop(image, crop_size):
    return image.crop((
        (image.size[0] - crop_size[0]) / 2,
        (image.size[1] - crop_size[1]) / 2,
```

```
        (image.size[0] + crop_size[0]) / 2,
        (image.size[1] + crop_size[1]) / 2,
        ))

image_size = (128, 128) # используем вырез размера 128x128
image = center_crop(image, image_size)
```

3. Создание обучающей и тестовой выборок изображений. На этом шаге выполняется разделение набора данных на обучающую и тестовую выборку. Это делается вручную, т.к. авторами набора подмножества не регламентируются. Сформируем набор индексов элементов набора:

```
genders = metadata[3][0]
indices = []
```

В описании набора указывается, что некоторые изображения могут содержать неопределенные метки пола. Устраним такие записи из набора данных:

```
import math

for i, gender in enumerate(genders):
    if (math.isnan(gender) == True):
        continue
    indices.append(i)
```

Перемешаем индексы случайным образом и выполним разделение набора:

```
import random

random.shuffle(indices)

train_ratio = 0.66 # разделим данные в соотношении 2:1
threshold_index = int(train_ratio * len(indices))
train_indices = indices[: threshold_index]
test_indices = indices[threshold_index :]
```

4. Подсчет среднего значения пикселей. Для улучшения свойств сходимости алгоритмов оптимизации приведем данные к нулевому среднему значению. Вычислим среднее по набору данных для каждого из каналов изображения.

```
def compute_mean(indices, metadata):
    channel_count = 3 # RGB
    channels_mean = np.zeros(channel_count)

    for pos, i in enumerate(indices):
        image = Image.open(dataset_root + '/' + metadata[2][0][i][0])
        if (len(image.getbands()) != 3):
            image = image.convert("RGB")
        image = center_crop(image, image_size)

        # преобразуем представление из (H, W, C) в (C, H, W)
        image_array = np.array(image, dtype=np.int8).transpose((2, 0, 1))
        channels_mean += np.mean(image_array, axis=(1, 2))

    channels_mean = channels_mean / len(indices)
    return channels_mean
```

5. Сохранение данных. К этому моменту данные прошли предобработку, но для использования в процессе обучения глубоких моделей необходимо сохранить их в необходимом формате. Рассмотрим вариант хранения данных во внешней памяти и загрузки фрагментов по необходимости. Neon поддерживает работу с данными в формате HDF5, предоставляя итератор **`HDF5Iterator`**. Преобразуем данные набора к этому формату, используя модуль h5py. Следуя документации [21], сохраним каждое подмножество данных в отдельном файле формата HDF5. Файлы должны иметь следующую структуру:

- Массив **input**, содержащий входные изображения в формате $(N, C \times H \times W)$ типа **float** или **numpy.int8**, где N – количество записей в наборе данных, C – число каналов изображений, $H \times W$ – высота и ширина изображений.
- Атрибут **lshape** массива **input**, описывающий способ интерпретации данных. Представляет собой кортеж (C, H, W).
- Опциональный атрибут **mean** массива **input**, содержащий средние значения пикселей по каналам. Форма данных атрибута: $(C, 1)$.
- Массив **output**, содержащий выходные значения. В нашем случае это метки пола размера $(N, 1)$ типа **numpy.int8**.
- Опциональный атрибут **nclass** массива **output**, содержащий количество классов. Используется в задачах классификации.

Пример кода сохранения данных для обучающей выборки:

```
import h5py as h5

channels_mean = compute_mean(train_indices, metadata)
indices = train_indices
dataset_file = h5.File(save_dir + '/train.h5', 'w')

input_channels_count = 3
input_channel_size = image_size[0] * image_size[1]
input_size = input_channels_count * input_channel_size

dataset_inputs = dataset_file.create_dataset('input',
    (len(indices), input_size), dtype=np.int8)
dataset_inputs.attrs['lshape'] = (input_channels_count,
    image_size[1], image_size[0])
dataset_outputs = dataset_file.create_dataset('output',
    (len(indices), 1), dtype=int)
dataset_outputs.attrs['nclass'] = 2

for pos, i in enumerate(indices):
    image = Image.open(dataset_root + '/' + metadata[2][0][i][0])
    if (len(image.getbands()) != 3):
        image = image.convert("RGB")
    image = center_crop(image, image_size)

    image_array = np.array(image, dtype=np.int8).transpose((2, 0, 1))
    dataset_inputs[pos] = image_array.flatten()
    dataset_outputs[pos] = gender

dataset_inputs.attrs['mean'] = channels_mean / len(indices)

dataset_file.close()
```

6. Проверка сохраненных данных. К этому моменту данные сохранены в нужном формате. Осталось проверить, что Neon может их загрузить.

```
import neon.backends
neon.backends.gen_backend('cpu', batch_size=1) # инициализация библиотеки

from neon.data import HDF5Iterator
train_iter = HDF5Iterator(save_dir + '/train.h5')
```

Попытаемся вывести первый элемент набора:

```
import numpy as np
import PIL.Image
```

```
train_it = iter(train_iter) # создаем итератор Python
entry = next(train_it) # пара (input, target), данные в памяти устройства
image_data = entry[0].get() # копируем в основную память
# image_data.shape == (C, H, W, batch_size)
gender_data = entry[1].get() # копируем в основную память
# gender_data.shape == (1, batch_size)

image = PIL.Image.fromarray(image_data.astype('i1') \
   .reshape((3, 128, 128)).transpose((1, 2, 0)), 'RGB')
gender = gender_data[0]

print(gender)
image.show()
```

После успешной проверки данные можно считать подготовленными к дальнейшему использованию во время обучения сетей.

Полные исходные коды данного примера находятся в материалах курса соответственно в скриптах **`Practice0_intro/imdb_wiki_converter.py`** и **`datasets/imdb_wiki_face_dataset.py`**.

4 Литература

4.1 Основная литература

1. Хайкин С. Нейронные сети. Полный курс. – М.: Издательский дом «Вильямс». – 2006. – 1104 с.
2. Осовский С. Нейронные сети для обработки информации. – М.: Финансы и статистика. – 2002. – 344 с.
3. Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press. – 2016. – [http://www.deeplearningbook.org].

4.2 Ресурсы сети Интернет

4. Документация Intel® neon™ Framework: обзор возможностей [http://neon.nervanasys.com/docs/latest/overview.html].
5. Домашняя страница набор данных IMDB-WIKI [https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki].
6. Домашняя страница проекта Python [https://www.python.org].
7. Домашняя страница проекта python-pip [https://pypi.org/project/pip].
8. Домашняя страница проекта python-virtualenv [https://virtualenv.pypa.io/en/stable].
9. Домашняя страница проекта HDF5 [https://support.hdfgroup.org/HDF5].
10. Домашняя страница проекта NVIDIA CUDA [https://developer.nvidia.com/cuda-downloads].
11. Домашняя страница проекта Intel® Math Kernel Library [https://software.intel.com/en-us/mkl].
12. Документация Intel® neon™ Framework: установка [http://neon.nervanasys.com/docs/latest/installation.html].
13. Страница загрузки Python 3.5.3 [https://www.python.org/downloads/release/python-353].
14. Документация Intel® Math Kernel Library: параметр KMP Affinity [https://software.intel.com/en-us/node/522691].
15. Документация Intel® neon™ Framework: загрузка данных [http://neon.nervanasys.com/docs/latest/loading_data.html].
16. Документация Intel® neon™ Framework: загрузчик данных Aeon [http://neon.nervanasys.com/docs/latest/loading_data.html#aeon-dataloader].
17. Документация Intel® neon™ Framework: встроенные наборы данных [http://neon.nervanasys.com/docs/latest/datasets.html].
18. Домашняя страница проекта SciPy [https://www.scipy.org].
19. Домашняя страница проекта Pillow [https://pillow.readthedocs.io/en/3.1.x/index.html].

20. Домашняя страница проекта NumPy [http://www.numpy.org].
21. Документация Intel® neon™ Framework: тип HDF5Iterator [http://neon.nervanasys.com/docs/latest/generated/neon.data.hdf5iterator.HDF5Iterator.html#neon.data.hdf5iterator.HDF5Iterator].