МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)

Институт информационных технологий, математики и механики

Кафедра математического обеспечения и суперкомпьютерных технологий

Направления подготовки: «Прикладная математика и информатика», «Фундаментальная информатика и информационные технологии»

Магистерские программы: «Системное программирование», «Компьютерная графика и моделирование живых и технических систем»

Отчет по лабораторной работе

«Разработка полностью связанной нейронной сети с помощью библиотеки MXNet»

Выполнили:

_____

Нижний Новгород

2017

Оглавление

Постановка задачи:

Получить базовые навыки работы с библиотекой MXNet и разработать полносвязную нейронную сеть.

Цели работы:

- Установить библиотеку MXNet и проверить корректность установки.
- Выбрать практическую задачу распознавания образов для выполнения практических работ
- Разработать программ/скриптов для подготовки тренировочных и тестовых данных в формате, который обрабатывается библиотекой.
- Разработать нескольких архитектур полностью связанных нейронных сетей в формате, который принимается библиотекой.
- Обучить и протестировать сеть

Этапы работы.

Для выполнения практических работ была выбрана версия библиотеки **MXNet** для языка программировани Python.

MXNet - масштабируемая система для машинного обучения, созданная Amazon и несколькими университетами. Библиотека позволяет максимально эффективно задействовать как аппаратные средства (она работает и на CPU, и на GPU, на одной рабочей станции и их кластере.

Причины выбора библиотеки:

- наличие удобной документации с примерами;
- символические вычисления и умная автоматическая оптимизация с учётом аппаратного обеспечения;
- кроссплатформенность.

Алгоритм установки достаточно прост и есть на сайте
https://mxnet.incubator.apache.org/get_started/install.html

1. **Практическое задание**: распознавание эмоций персонажей (ссылка на данные: http://grail.cs.washington.edu/projects/deepexpr/ferg-db.html)

   Набор данных состоит из изображений лиц 6-ти персонажей. Изображения для каждого персонажа сгруппированы по 7 типовых выражений лица: гнев, отвращение, страх, радость, нейтральное выражение, печаль и удивление. Всего база содержит 55767 изображений лиц шести стилизованных персонажей.

   Были реализованы нейронные сети с 1-3 полносвязными скрытыми слоями. В качестве варьируемого параметра использовалась функция активации нейронов внутренних слоёв, выбор осуществлялся между ReLu и сигмоидальной функцией. На выходном слое при решении задач классификации обычно используется функция Softmax, данная функция всегда остаётся без изменений.

2. В качестве тренировочного набора данных используем изображения с эмоциями 5 персонажей. В качестве тестового – изображения с эмоциями 6го персонажа. Такое разделение выбрано потму, что разнообразие входных данных невелико – лица персонажей смоделированны с помощью 3-D инструмента, многие изображения датасета отличаются друг от друга незначительно. Было бы интересно посмотреть, насколько хорошо сеть справится с неизвестным ей персонажем.

3. Метрика. В качестве метрики точности решения используем отношение угаданных эмоций ко всем в тестовой выборке, т.е

$$Accuracy = \frac{Correctly\ answers\ count}{Images\ count}$$

4. Данные создаются на основе изображений в формате .png. Каждая папка, в которой размещены файлы, трактуется как признак. Все изображения конвертируются в специальный сжатый rec-формат, в котором они случайным образом заранее перемешаны и поставлены в соответствие с lable - меткой признака, который следует распознать на этом изображении (в нашем случае - эмоция).

5. Данные на вход передаются в формате rec. Конвертация происходит с помощью скрипта im2rec.py.

6. Директория src содержит слдеующие файлы:
   a. main.py - файл с конфигурацией нейронной сети
      Так как программный интерфейс весьма удобен и позволяет написание приложения в форме скрипта, изменения конфигурации мы встраиваем непосредственно в main.py по мере надобности.
7. Тестовые конфигурации сетей. Раздел содержит визуальные схемы конфигураций построенных нейронных сетей (не содержит)

8. Результаты экспериментов:

Таблица 1. Результаты экспериментов

№ конфигурации	Число скрытых слоев	Число нейронов скрыт.	Функции активации	Точность трен.	Точность тест.
1	1	150	sigmoid	0,15	0,16965116
2	1	150	relu	0,16	0,169651
3	1	250	sigmoid	0,16	0,169701
4	1	250	relu	0,15	0,169604
5	2	100-100	Sigmoid-sigmoid	0,15	0,16961
6	2	100-100	relu-relu	0,16	0,17012
7	3	100-100-100	Sigmoid-relu-sigmoid	0,13	0,13061

Если исключить применение активационной функции sigma/relu даже в случае с одним скрытым слоем, то точность на обучающей выборке достигнет 1.0, а точность на тестовой будет порядка 0.55 - 0.65 *(почему?)*.

Вывод

Поподробнее остановимся на результатах экспериментов.

Во-первых, сразу бросается в глаза крайне низкая точность предсказания. По сути, нейросеть предсказывает ненамного лучше случайного угадывания: $\frac{1.0}{7} \approx 0.14$. Во-вторых, изменение количества слоёв или числа нейронов на них почти никак не влияет на результат; более того, на четырёх слоях нейросеть даёт даже худшие предсказания, чем выбор наугад. Наиболее вероятное объяснение – неспособность полносвязной сети выявлять устойчивые признаки, как это делает, например, свёрточная нейронная сеть, способная акцентировать эти «фичи» с помощью свёрточных фильтров, а затем сохранить и передать эти значения дальше, выделив их из незначащей информации с помощью, например, max-pooling-операции. Соответственно, при увеличении числа слоёв мы лишь усложняем неэффективную модель.