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

Направление подготовки: «Прикладная математика и информатика»
Магистерская программа: «Вычислительные методы и суперкомпьютерные технологии»

Образовательный курс «Глубокое обучение»

ОТЧЕТ

по лабораторной работе №2

Разработка полностью связанных нейронных сетей

Выполнили:
студенты группы 381703-3м
Гладкова Татьяна
Крутоборежская Ирина
Крюкова Полина
Подчищаева Мария

Нижний Новгород
2018

Содержание

Цели

Цель настоящей работы состоит в том, чтобы получить базовые навыки работы с одной из библиотек глубокого обучения (в данном случае, MXNet) на примере полностью связанных нейронных сетей.

Задачи

Выполнение практической работы предполагает решение ***следующих задач***:

1. Выбор библиотеки для выполнения практических работ курса.
2. Установка выбранной библиотеки на кластере (параметры аутентификации и инструкция по работе с кластером выложена в отдельной задаче в системе redmine).
3. Проверка корректности установки библиотеки. Разработка и запуск тестового примера сети, соответствующей логистической регрессии, для решения задачи классификации рукописных цифр набора данных MNIST (пример разобран в лекционных материалах).
4. Выбор практической задачи компьютерного зрения для выполнения практических работ.
5. Разработка программ/скриптов для подготовки тренировочных и тестовых данных в формате, который обрабатывается выбранной библиотекой.
6. Разработка нескольких архитектур полностью связанных нейронных сетей (варьируются количество слоев и виды функций активации на каждом слое) в формате, который принимается выбранной библиотекой.
7. Обучение разработанных глубоких моделей.
8. Тестирование обученных глубоких моделей.
9. Публикация разработанных программ/скриптов в репозитории на GitHub.
10. Подготовка отчета, содержащего минимальный объем информации по каждому этапу выполнения работы.

Решаемая задача

Была выбрана задача бинарной классификации: «кошки» - «собаки». Были использованы картинки из наборов данных https://www.kaggle.com/tongpython/cat-and-dog и https://www.kaggle.com/c/dogs-vs-cats/data. Получившийся набор состоит из 35029 изображений.

Рис. 1 Пример изображения из класса «кошки»

Рис. 2 Пример изображения из класса «собаки»

С помощью скрипта на python данные были преобразованы к размеру 128×128. С помощью скрипта im2rec.py, который входит в библиотеку MXNet, изображения были сконвертированы в формат .rec.

Выбор библиотеки

Для выполнения лабораторных работ выбрана библиотека MXNet для языка программирования Python.

На этапе проверки корректности установки библиотеки выполнена разработка и запуск тестового примера сети для решения задачи классификации рукописных цифр набора данных MNIST. Достигнута точность 0.9225.

Метрика качества решения задачи

В качестве метрики точности решения используется отношение угаданных животных ко всем в тестовой выборке:

$$Accuracy = \frac{CorrectAnswersCount}{ImagesCount}$$

Тренировочные и тестовые наборы данных

В качестве тренировочной выборки используем тренировочную выборку первого и второго наборов данных, всего 16500 изображений котов и 16505 изображений собак. В качестве тестовой выборки используем тестовую выборку только из первого набора данных, т.к. во втором наборе данных тестовая выборка не разбита на изображения котов и собак. Всего в тестовой выборке 2042 изображения, котов и собак поровну.

Конфигурации нейронных сетей

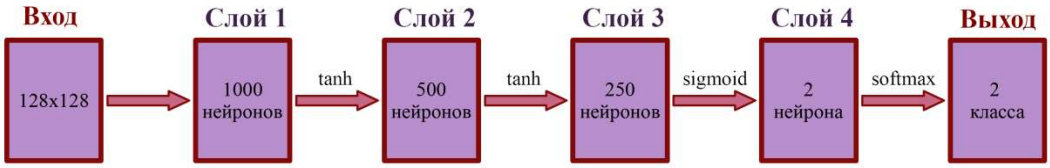
В данной работе были рассмотрены четыре конфигурации полносвязных нейронных сетей с 4-мя и 5-мя скрытыми слоями.
Активационная функция на слоях выбирается из следующих:

- $tanh, f = \frac{e^s - e^{-s}}{e^v + e^{-s}}$
- $sigmoid, f \frac{1}{e^v + e^{-s}}$
- $relu, fmax(x, 0)$

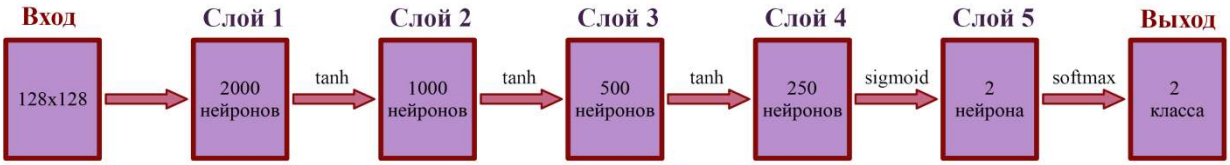
На выходном слое:

- $softmax, f \frac{e^{s_j}}{\sum_{j=1}^{n} e^{s_j}}$

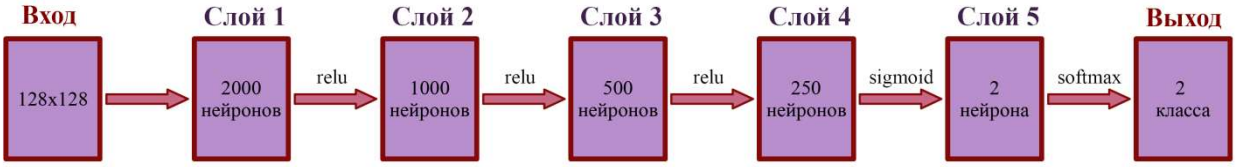
Конфигурация №1



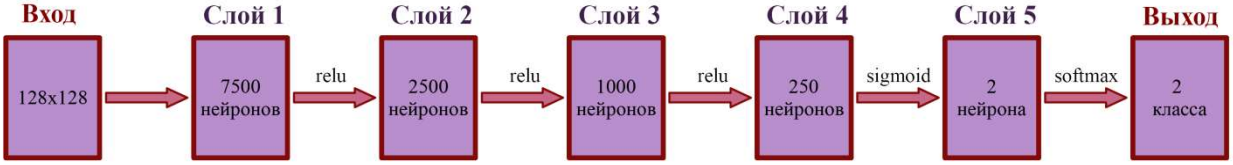
Конфигурация №2



Конфигурация №3



Конфигурация №4



Разработанные программы/скрипты

В директории расположены четыре конфигурации построенных полносвязных нейронных сетей. Соответствия построенных конфигураций и конфигураций в директории:

- fcnn_tts.py – первая нейронная сеть tanh-tanh-sigmoid, 1000-500-250-2
- fcnn_ttts.py – вторая нейронная сеть tanh-tanh-tanh-sigmoid, 1000-500-250-2
- fcnn_rrrs1.py – третья нейронная сеть relu-relu-relu-sigmoid, 2000-1000-500-250-2
- fcnn_rrrs2.py – четвертая нейронная сеть relu-relu-relu-sigmoid, 7500-2500-1000-250-2

Результаты экспериментов

В работе рассмотрены 4 конфигурации.

Тестовая инфраструктура
Вычисления производились на машине со следующими характеристиками:

- Процессор: i7-6700k
- Видеокарта: gtx1070
- Оперативная память: 16 Гб

Параметры обучения:

- количество эпох – 10,
- скорость обучения – 0.001.

№	Количество скрытых слоев	Количество нейронов на скрытых слоях	Функции активации	Результат		
				Точность на тренировочном множестве	Точность на тестовом множестве	Время, с
1	4	1000-500-250-2	tanh-tanh-sigmoid	0.56	0.5	514.38
2	5	2000-1000-500-250-2	tanh-tanh-tanh-sigmoid	0.57	0.5	524.61
3	5	2000-1000-500-250-2	relu-relu-relu-sigmoid	0.79	0.74	496.42
4	5	7500-2500-1000-250-2	relu-relu-relu-sigmoid	0.83	0.78	1739.67

Анализ результатов

Нейронные сети с функцией активации ReLU показывают результат лучше, чем нейронные сети с функцией сигмоидальной функцией активации.

Рассмотрим функцию активации, которая представляется суммой нескольких логистических сигмоидов:

$$f(x) = \sigma\left(x + \frac{1}{2}\right) + \sigma\left(x - \frac{1}{2}\right) + \sigma\left(x - \frac{3}{2}\right) + \sigma\left(x - \frac{5}{2}\right) + \cdots$$

Построенную сумму можно приблизить интегралом:

$$f(x) = \sum_{i=0}^{\infty} \sigma\left(x + \frac{1}{2} - i\right) \approx$$

$$\approx \int_{\frac{1}{2}}^{\infty} \sigma\left(x + \frac{1}{2} - y\right) dy = -ln\left(1 + e^{x+\frac{1}{2}-y}\right)_{y=\frac{1}{2}}^{y=\infty} = ln(1 + e^x)$$

Приведенный ряд сигмоидальных функций более выразителен и может быть приближен $ln(1 + e^x)$.

Указанный логарифм похож на ReLU.

Преимущества ReLU:

1. Вычисление сигмоиды и гиперболического тангенса требует ресурсоёмких операций, таких как возведение в степень, в то время как ReLU не подвержен насыщению.
2. Применение ReLU существенно повышает скорость стохастического градиентного спуска по сравнению с сигмоидой и гиперболическим тангенсом. Это обусловлено линейным характером и отсутствием насыщения данной функции.

Недостатки ReLU:

К сожалению, ReLU не всегда достаточно надёжны и в процессе обучения могут выходить из строя. Например, большой градиент, проходящий через ReLU, может привести к такому обновлению весов, что данный нейрон никогда больше не активируется. Если это произойдет, то, начиная с данного момента, градиент, проходящий через этот нейрон, всегда будет равен нулю. Соответственно, данный нейрон будет необратимо выведен из строя. Например, при слишком большой скорости обучения, может оказаться, что до 40% ReLU никогда не активируются. Эта проблема решается посредством выбора надлежащей скорости обучения.

Выводы

Наилучший результат был получен на нейронной сети с конфигурацией №4. В ходе экспериментов было установлено, что нейронные сети с функцией активации relu показывают более точные результаты. Так же увеличить точность помогло увеличение нейронов на всех слоях. Однако, точность увеличилась не сильно, а время работы более чем в 3 раза.