

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: «Фундаментальная информатика и
информационные технологии»

Магистерская программа: «Компьютерная графика»

Образовательный курс «Глубокое обучение»

ОТЧЕТ

по лабораторной работе №2

Разработка полностью связанной нейронной сети

Выполнили:

студенты группы 381706-2м

Привалов Даниил

Бабаев Иван

Зубарева Екатерина

Фадеев Алексей

Нижний Новгород
2018

Содержание

Цели.....	3
Задачи.....	4
Решаемая задача.....	5
Выбор библиотеки.....	6
Метрика качества решения задачи.....	6
Тренировочные и тестовые наборы данных.....	6
Конфигурации нейронных сетей.....	7
Разработанные программы/скрипты.....	14
Результаты экспериментов.....	14
Анализ результатов.....	16

Цели

Цель настоящей работы состоит в том, чтобы получить базовые навыки работы с одной из библиотек глубокого обучения (Caffe, Torch, TensorFlow, MXNet или какая-либо другая библиотека на выбор студента) 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 классификации дорожных знаков. Количество классов - 43. Датасет: <http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset>

Архив обучающего набора имеет следующую структуру:

- Один каталог на класс
- Каждый каталог содержит один CSV-файл с аннотациями («GT- <ClassID> .csv») и обучающими изображениями.
- Обучающие изображения сгруппированы по наборам
- Каждый набор содержит 30 изображений одного дорожного знака

Commented [IB1]:

Commented [PW2R1]:

Формат изображения

- Каждое изображение содержит один дорожный знак
- Изображения содержат границу в 10% вокруг фактического дорожного знака (не менее 5 пикселей), чтобы обеспечить подходы по краям
- Изображения хранятся в формате PPM (Portable Pixmap, P6)
- Размеры изображения варьируются от 15x15 до 250x250 пикселей.
- Изображения не обязательно имеют квадратную форму
- Дорожный знак необязательно находится по центру изображения.
- Ограничительная рамка дорожного знака является частью описания.

Описания предоставляются в файлах CSV. Поля разделены знаком ";" (точка с запятой). Описания содержат следующую информацию:

- Имя файла: Имя файла соответствующего изображения
- Ширина: ширина изображения
- Высота: высота изображения
- ROI.x1: X-координата верхнего левого угла ограничительной рамки дорожного знака
- ROI.y1: Y-координата верхнего левого угла ограничительной рамки дорожного знака
- ROI.x2: X-координата нижнего правого угла ограничительной рамки дорожного знака
- ROI.y2: Y-координата нижнего правого угла ограничительной рамки дорожного знака
- ClassId: номер класса дорожного знака

Предобработка данных

- Автоматическое выравнивание яркости (histogram equalization).
- Обрезка по центру.
- Перевод в черно-белое изображение.
- Увеличение до заданного размера (48x48).

Выбор библиотеки

Для выполнения лабораторных работ выбрана библиотека Keras 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 0.89.

Метрика качества решения задачи

В качестве метрики точности решения используется отношение правильно классифицированных знаков ко всем знакам в тестовой выборке:

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

Тренировочные и тестовые наборы данных

39203 изображений различных знаков для тренировочных данных.

12630 изображений используется при финальном тестировании модели.

Конфигурации нейронных сетей

В данной работе были рассмотрены шесть конфигураций полностью связанных нейронных сетей с 3-мя и 4-мя скрытыми слоями.

Активационная функция на слоях выбирается из следующих:

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

На выходном слое:

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

Для обучения использован модифицированный метод градиентного спуска. Adam — adaptive moment estimation, оптимизационный алгоритм. Он сочетает в себе идею накопления движения и идею более слабого обновления весов для типичных признаков. Его реализация имеется в библиотеке Keras.

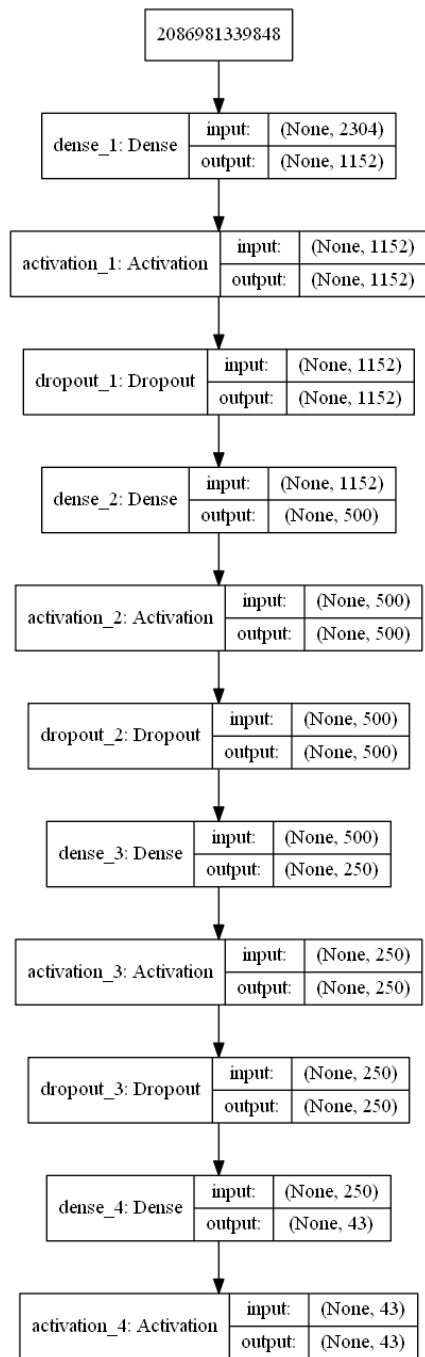


Рисунок 1. Сеть 1

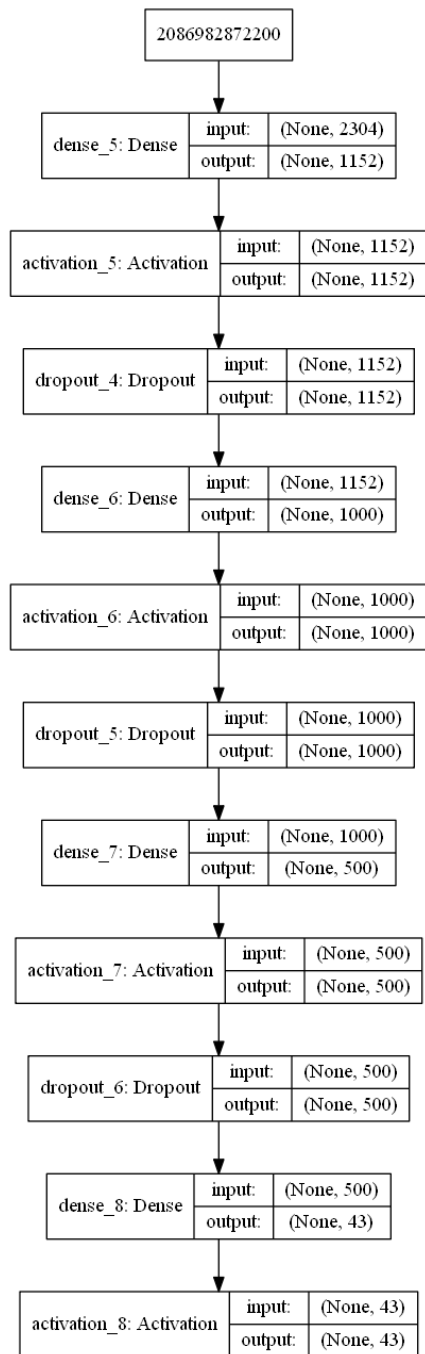


Рисунок 2. Сеть 2

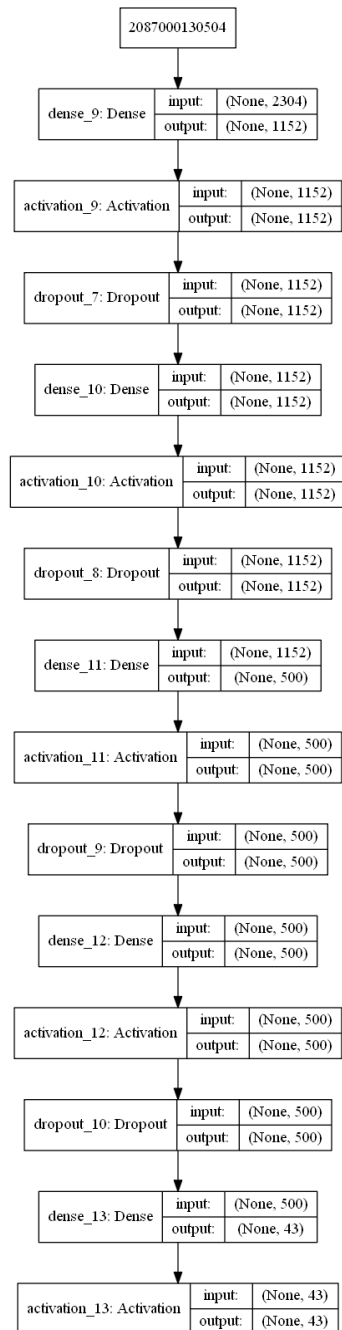


Рисунок 3. Сеть 3

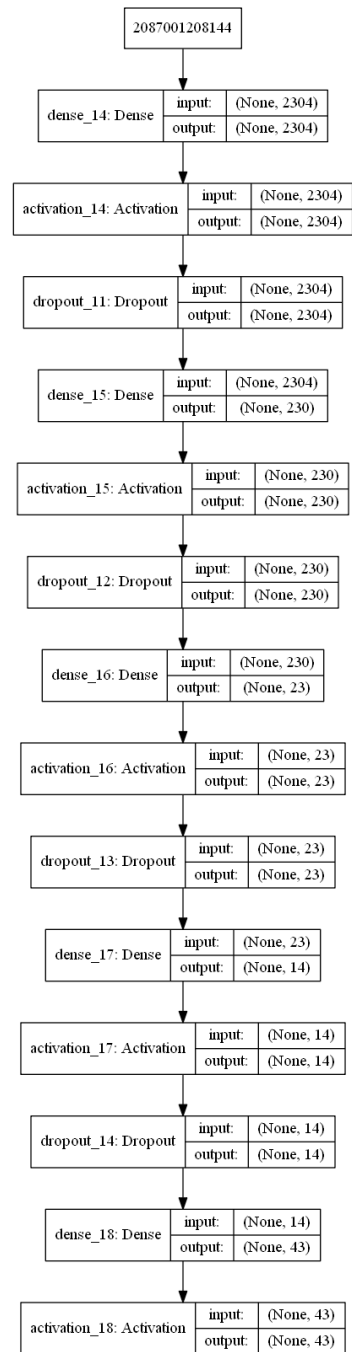


Рисунок 4. Сеть 4

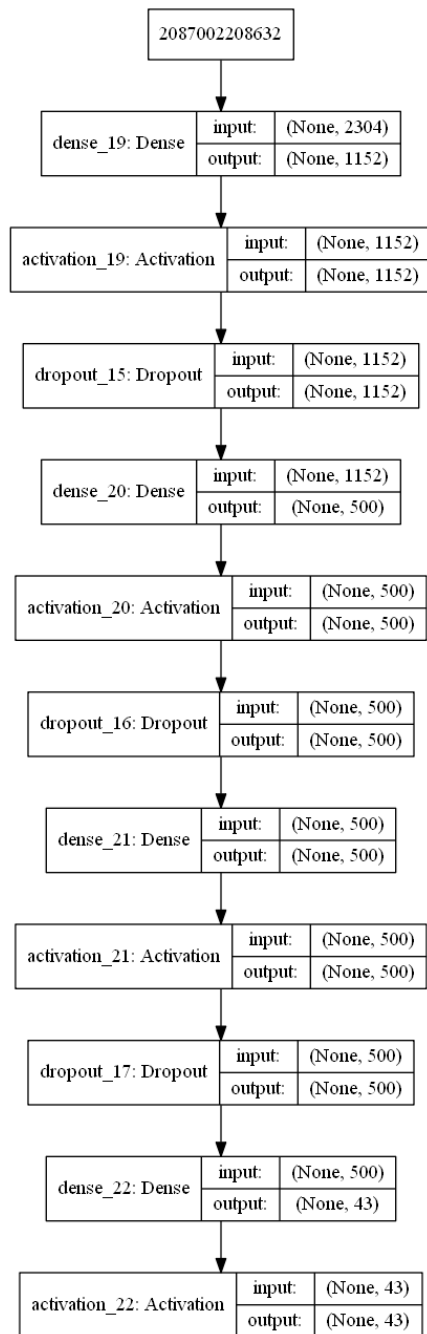


Рисунок 5. Сеть 5

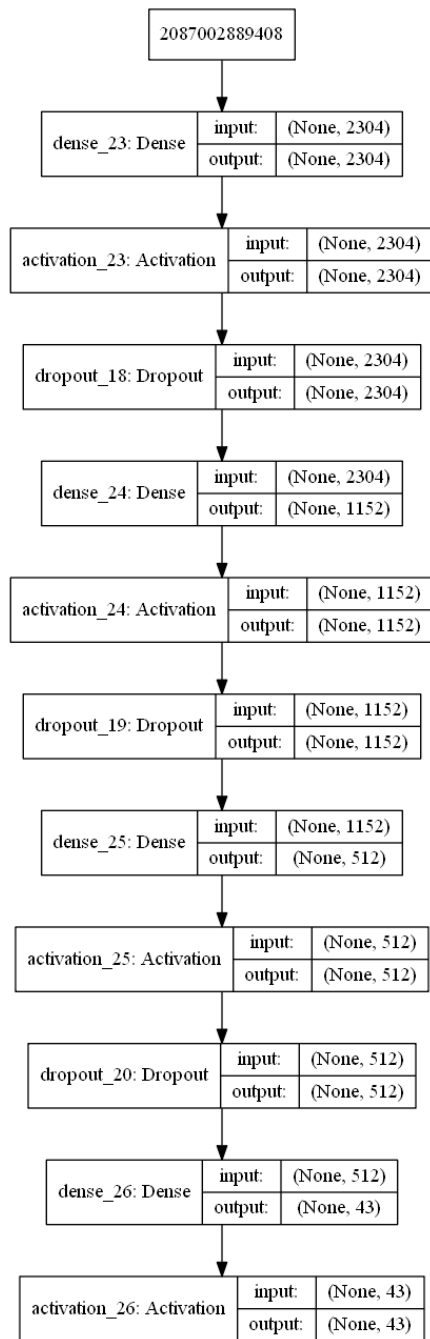


Рисунок 6. Сеть 6

Разработанные программы/скрипты

В репозитории в папке lab2 находится скрипт seq_all.py, в нем содержится функция seq_model(model_num) генерирующая вышеописанные модели и запускающая их обучение и тестирование в цикле.

Подготовка файлов датасета к работе подробно описана в README репозитория.

Результаты экспериментов

Параметры обучения:

Функция ошибки='categorical_crossentropy',

Оптимизационный алгоритм='adam' - <https://keras.io/optimizers/>,

batch_size=128,

Количество эпох – 10,

Скорость обучения – 0.001.

Параметры PC:

OS: Windows 10

CPU: Intel Core i5-5200U (2.2 GHz)

GPU: NVidia 920M (1 GB)

RAM: 4 GB

Commented [B3]:

№	Количество скрытых слоев	Количество нейронов на скрытых слоях	Функции активации	Результат		
				Точность на тренировочном множестве	Точность на тестовом множестве	Время эпохи, с
1	3	1152-500-250	tanh-tanh-sigmoid	0.84	0.795	46
2	3	1152-1000-500	relu-relu-sigmoid	0.91	0.832	62
3	4	1152-1152-500-500	tanh-tanh-tanh- sigmoid	0.721	0.723	69
4	4	2304 -230-23-14	relu-relu-relu- relu	0.582	0.684	82
5	3	1152-500-500	relu-relu-sigmoid	0.91	0.80	50
6	3	2304-1152-512	relu-relu-sigmoid	0.907	0.847	123

Анализ результатов

Лучший результат 0.85% точности был получен в 6 модели, получен он был за счет подбора оптимального количества нейронов для данных параметров обучения. Если взять большее количество нейронов, то результат по времени обучения и проценту ошибок сильно падал.

Результат полностью связанных сетей сильно лучше не сделать, так как когда мы преобразуем изображение в линейную цепочку байт, мы что-то безвозвратно теряем. Причем с каждым слоем эта потеря только усугубляется. Мы теряем топологию изображения, т.е. взаимосвязь между отдельными его частями. Кроме того, задача распознавания подразумевает умение нейросети быть устойчивой к небольшим сдвигам, поворотам и изменению масштаба изображения, т.е. она должна извлекать из данных некие инварианты, не зависящие от углов под которым сделано фото знака.

При тестировании с большим количеством нейронов было немного проще использовать ReLU функцию активации, но в целом на время данная функция не сильно влияет, время обучения сильно зависит от количества нейронов в сети.

Все же стоит отметить преимущества ReLU:

1. Вычисление сигмиды 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 сигмидой и гиперболическим тангенсом. Это обусловлено линейным характером и отсутствием насыщения данной функции.