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 №3

Разработка сверточных нейронных сетей

Выполнили:
студенты группы 381706-2м

Привалов Даниил
Бабаев Иван
Зубарева Екатерина
Фадеев Алексей

Нижний Новгород
2018

Содержание

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 сверточных нейронных сетей.

Задачи

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

1. Разработка нескольких архитектур сверточных нейронных сетей (варьируются количество слоев и виды функций активации на каждом слое) в формате, который принимается выбранной библиотекой глубокого обучения.
2. Обучение разработанных глубоких моделей.
3. Тестирование обученных глубоких моделей.
4. Публикация разработанных программ/скриптов в репозитории на GitHub.
5. 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

Состав:

- 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

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).

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 три конфигураций сверточных нейронных сетей с 8-15 скрытыми слоями.

Активационная функция на слоях выбирается из следующих:

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

На выходном слое:

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

Ниже на картинках приведены схемы построенных сетей:

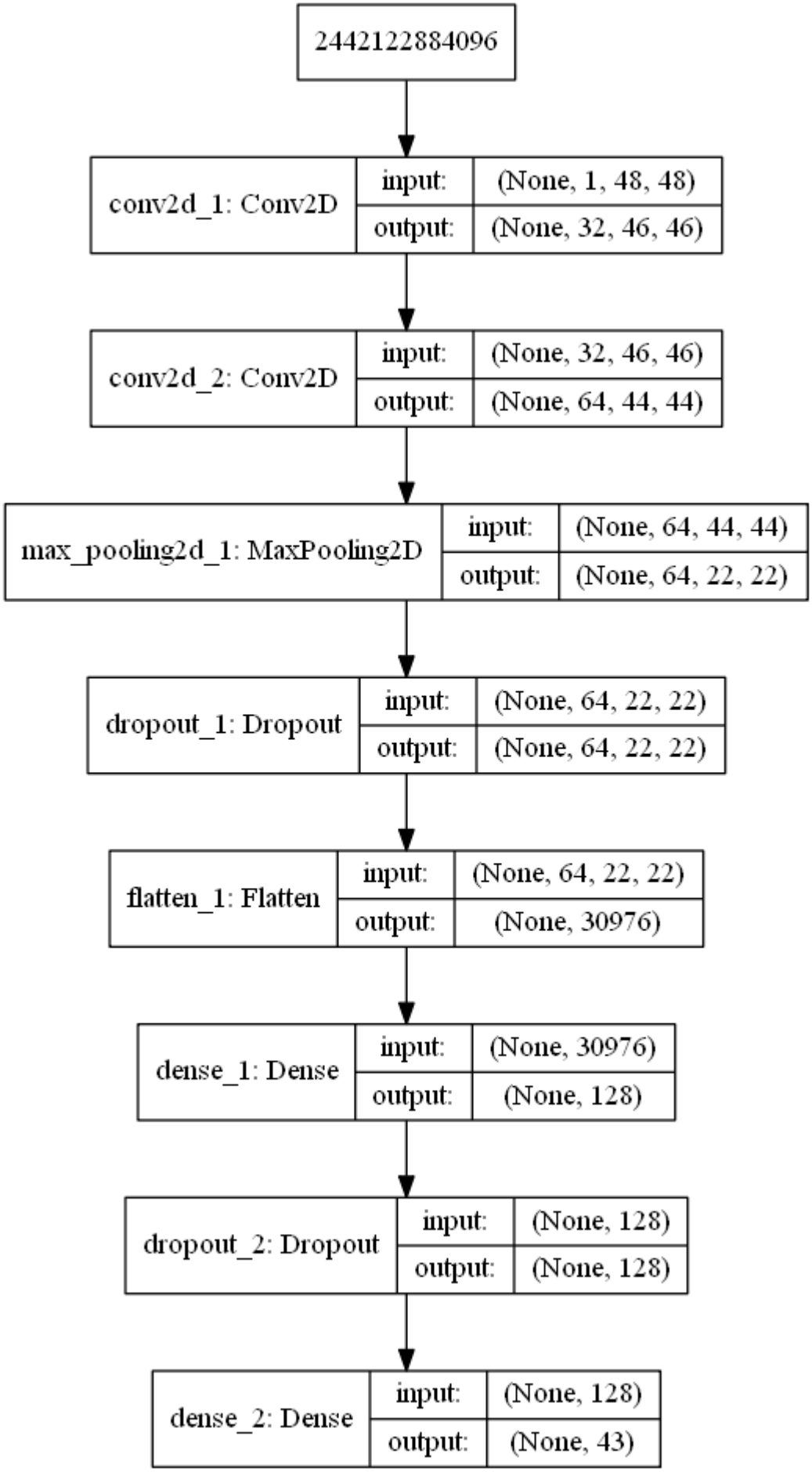

Рисунок 1. Сеть 1

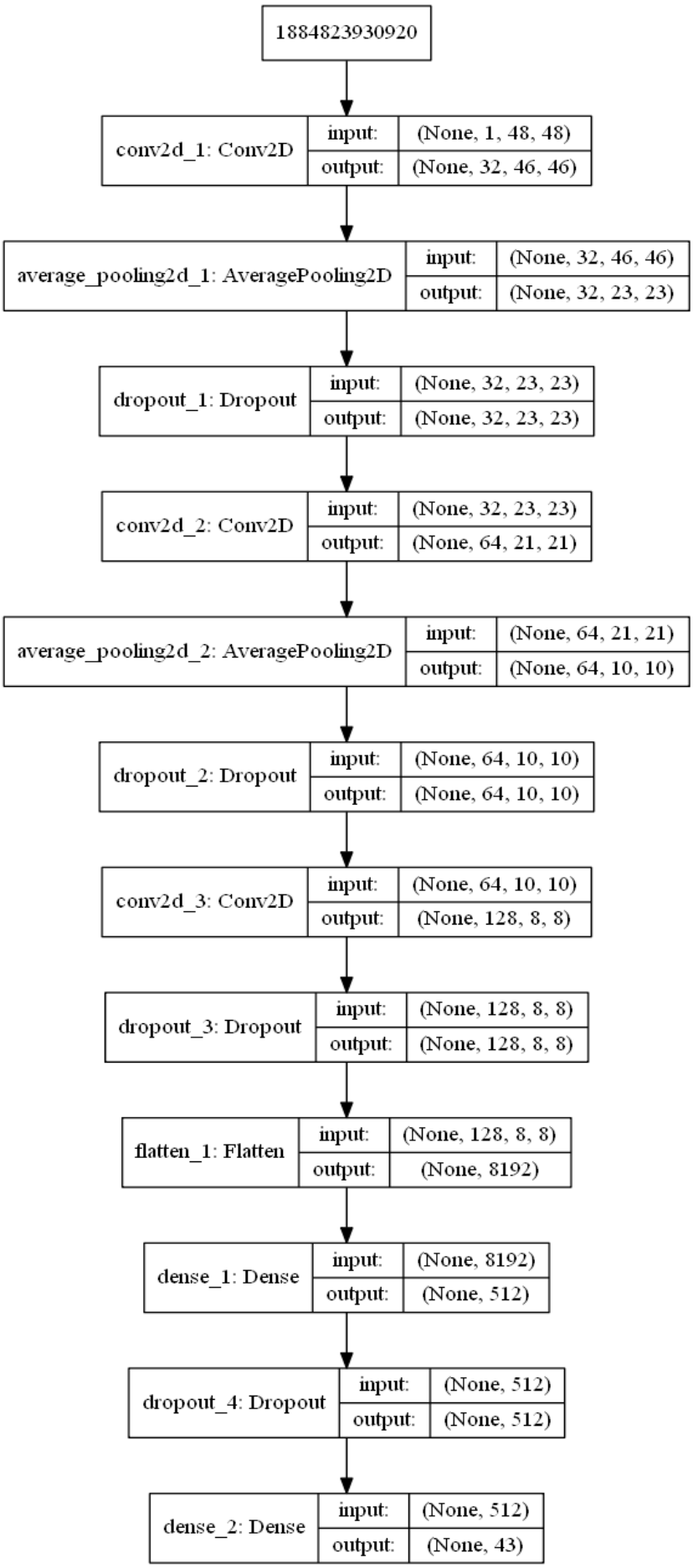

Рисунок 2. Сеть 2.

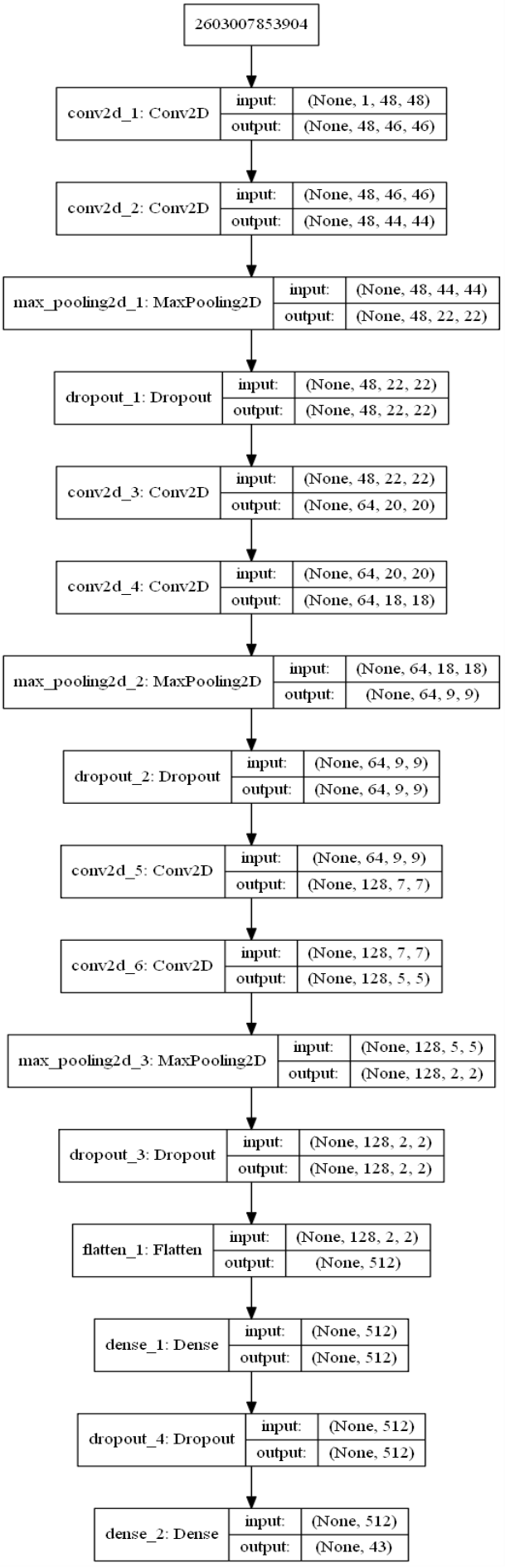

Рисунок 3. Сеть 3.

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

Разработанные программы/скрипты

В репозитории в папке lab3 находятся все три конфигурации модели. Скрипты генерируют модели и запускают их обучение и тестирование.

Подготовка набора данных 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 – 20,
Скорость обучения – 0.01.

Параметры PC:
OS: Windows 10
CPU: Intel Core i5-3470 3.2Ghz (4CPUs)
RAM: 16384Mb, DDR3
GPU: Nvidia GeForce GTX 970 1050 MHz, 4058Mb, 256-bit GDDR5
Cuda: 9.0.176
Python: 3.5.4
Keras: 2.2.4
Tensorflow-gpu: 1.12.0

Наименование сети	**Функции активации**	**Результат**		
		Точность на тренировочном множестве	Точность на тестовом множестве	Время, с
Сеть 1	relu - relu - relu	0.9375	0.937529	184
Сеть 2	relu-tanh-sigmoid – relu-tanh-sigmoid-linear	0.9498	0.949802	323
Сеть 3	relu - relu – relu- relu - relu – relu- relu - relu	0.9624	0.96239	415

Анализ результатов

Лучший результат 0.96% точности был достигнут в 3 модели, получен он был за счет подбора оптимального количества нейронов с ReLU функцией активации.

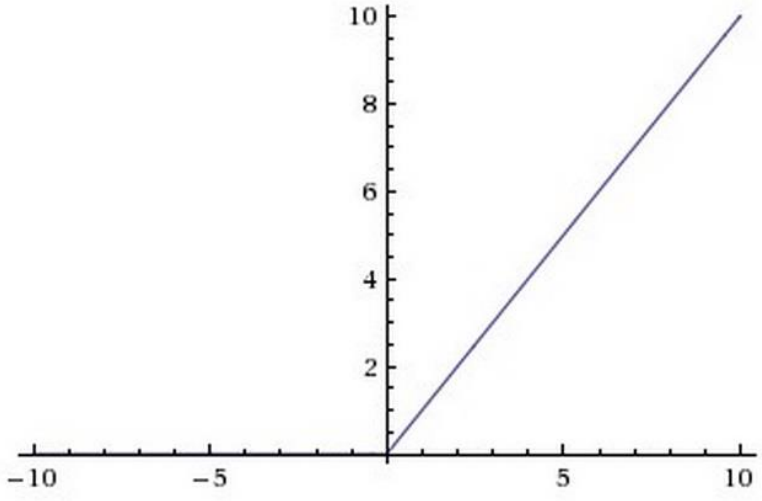

Рисунок 4. ReLU функция активации.

Почему ReLU? Чисто из-за практических соображений. Эта функция не требует пожирающих ресурсов операций и является обычным преобразованием $f(x)=max(0, x)$, которая реализует простой пороговый переход в нуле. При этом, используя эту функцию, наблюдается увеличение скорости стохастического градиентного спуска по сравнению с сигмоидой и гиперболическим тангенсом. Таким образом именно с ReLU есть возможность обучения сравнительно большой сети за разумное время и приемлемую точность результатов.

К сожалению, ReLU не всегда достаточно надежны 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 (learning rate), может оказаться, что до 40% ReLU никогда не активируются.