Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»

Институт Информационных Технологий, Математики и Механики

Направление: Прикладная математика и информатика

Магистерская программа: Компьютерные науки и приложения

**ОТЧЕТ**

по лабораторной работе №3

Тема:

**«Разработка свёрточных нейронных сетей»**

Выполнили: студенты группы 381803-4м

Котова О.А.

\_\_\_\_\_\_\_\_\_\_\_

Подпись

Лицов А.

\_\_\_\_\_\_\_\_\_\_\_

Подпись

Синицкая О.

\_\_\_\_\_\_\_\_\_\_\_

Подпись

Преподаватель: доцент, к.т.н. Кустикова В.Д.

\_\_\_\_\_\_\_\_\_\_\_

Подпись

Нижний Новгород

2019

Оглавление

[1. Постановка задачи 3](file:///C:\Users\user\Downloads\Kursovaya.docx#_Toc420187372)

[2. Тренировочные и тестовые наборы данных 4](file:///C:\Users\user\Downloads\Kursovaya.docx#_Toc420187373)

[3. Метрика качества решения 5](file:///C:\Users\user\Downloads\Kursovaya.docx#_Toc420187374)

[4. Разработанные программы 5](file:///C:\Users\user\Downloads\Kursovaya.docx#_Toc420187374)

[5. Тестовые конфигурации сетей 5](file:///C:\Users\user\Downloads\Kursovaya.docx#_Toc420187375)

[6. Результаты экспериментов 9](file:///C:\Users\user\Downloads\Kursovaya.docx#_Toc420187375)

[7. Анализ результатов 11](file:///C:\Users\user\Downloads\Kursovaya.docx#_Toc420187375)

**Постановка задачи**

**Цели**

Цельнастоящей работы состоит в том, чтобы построить архитектуру сверточной нейронной сети, которая позволяет решать практическую задачу с высокими показателями качества.

**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.

**Тренировочные и тестовые наборы данных**

Выбранная задача - Intel Image Classification: <https://www.kaggle.com/puneet6060/intel-image-classification>.

Исходные данные хранятся в директориях seg\_pred, seg\_test, seg\_train в формате jpg и размера 150x150.

* seg\_pred содержит 7301 изображений
* seg\_test - 3000 изображений, которые распределены по папкам
  + buildings
  + forest
  + glacier
  + mountain
  + sea
  + street
* seg\_train - 14034 изображений, которые распределены по папкам
  + buildings
  + forest
  + glacier
  + mountain
  + sea
  + street

Данные содержат около 25 тыс. цветных изображений размером 150x150, распределенных по 6 категориям: здания, лес, ледник, гора, море, улица. Изображения хранятся в формате jpg.

Тренировочная выборка содержит 14034 изображений.

Тестовая выборка содержит 3000 изображений.

Размер каждого изображения 150x150.

|  |  |  |  |
| --- | --- | --- | --- |
| *№* | *Категории* | *Размер тренировочной выборки* | *Размер тестовой выборки* |
| 1 | mountain | 2512 | 525 |
| 2 | street | 2382 | 501 |
| 3 | glasier | 2404 | 553 |
| 4 | buildings | 2191 | 437 |
| 5 | sea | 2274 | 510 |
| 6 | forest | 2271 | 474 |

Процентное соотношение категорий. Тренировочная выборка:



Процентное соотношение категорий. Тестовая выборка:



**Метрика качества решения**

Для оценки качества решения задачи выбрана метрика "Точность" ("Accuracy"). Она вычисляет, как часто прогнозы соответствуют меткам. Иными словами, частота с которой y\_pred совпадает с y\_true.

[](https://github.com/a-litsov/deep-learning/blob/master/lab2/img/accuracy.png)

**Разработанные программы**

Lab3.ipynb – скрипт для обучения свёрточных нейронных сетей.

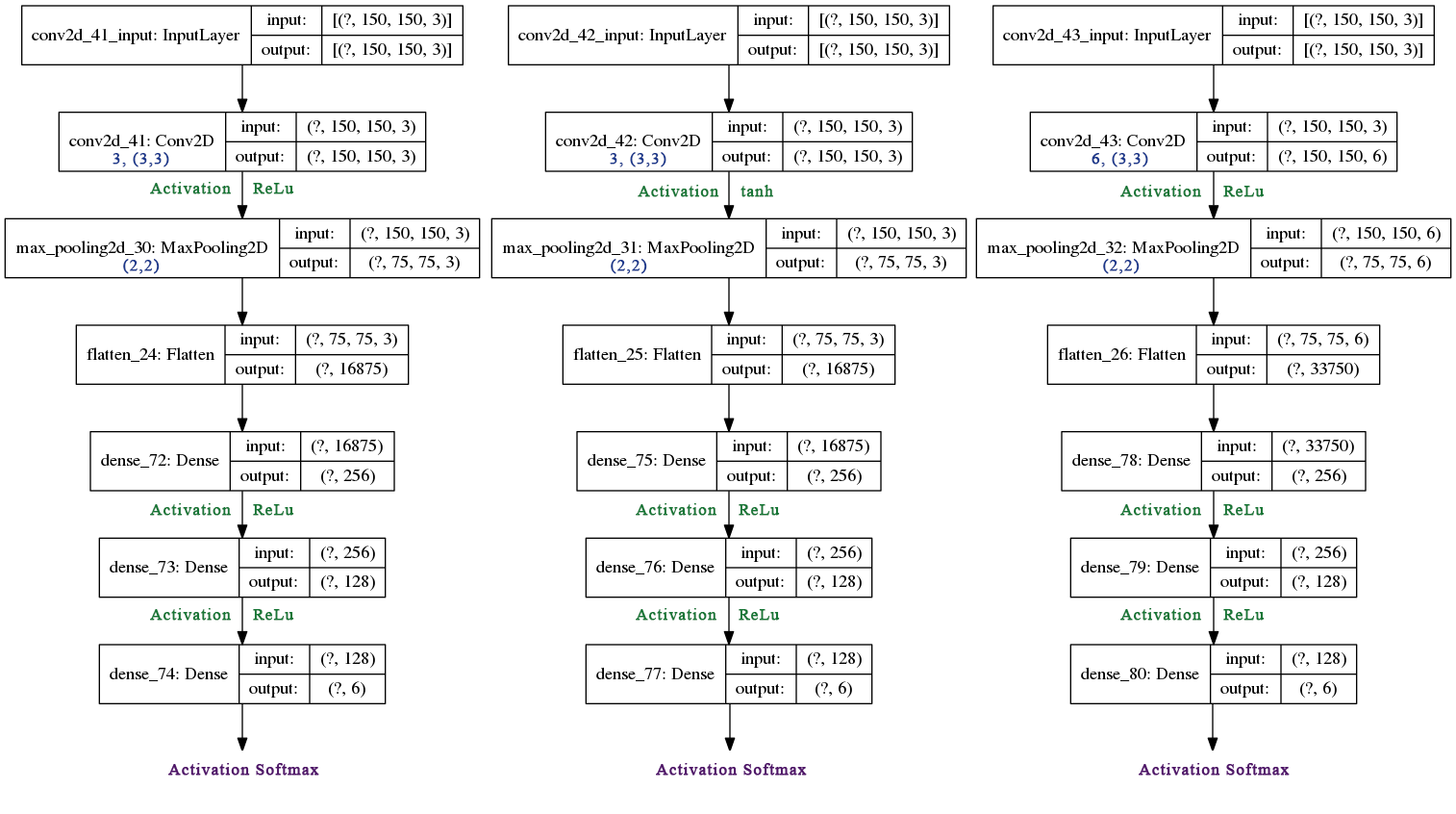
**Тестовые конфигурации сетей**

С помощью класса ImageDataGenerator и его метода flow\_from\_directory() генерируем пакеты. Данные возвращаются в формате (x, y), где x, y - numpy массивы.

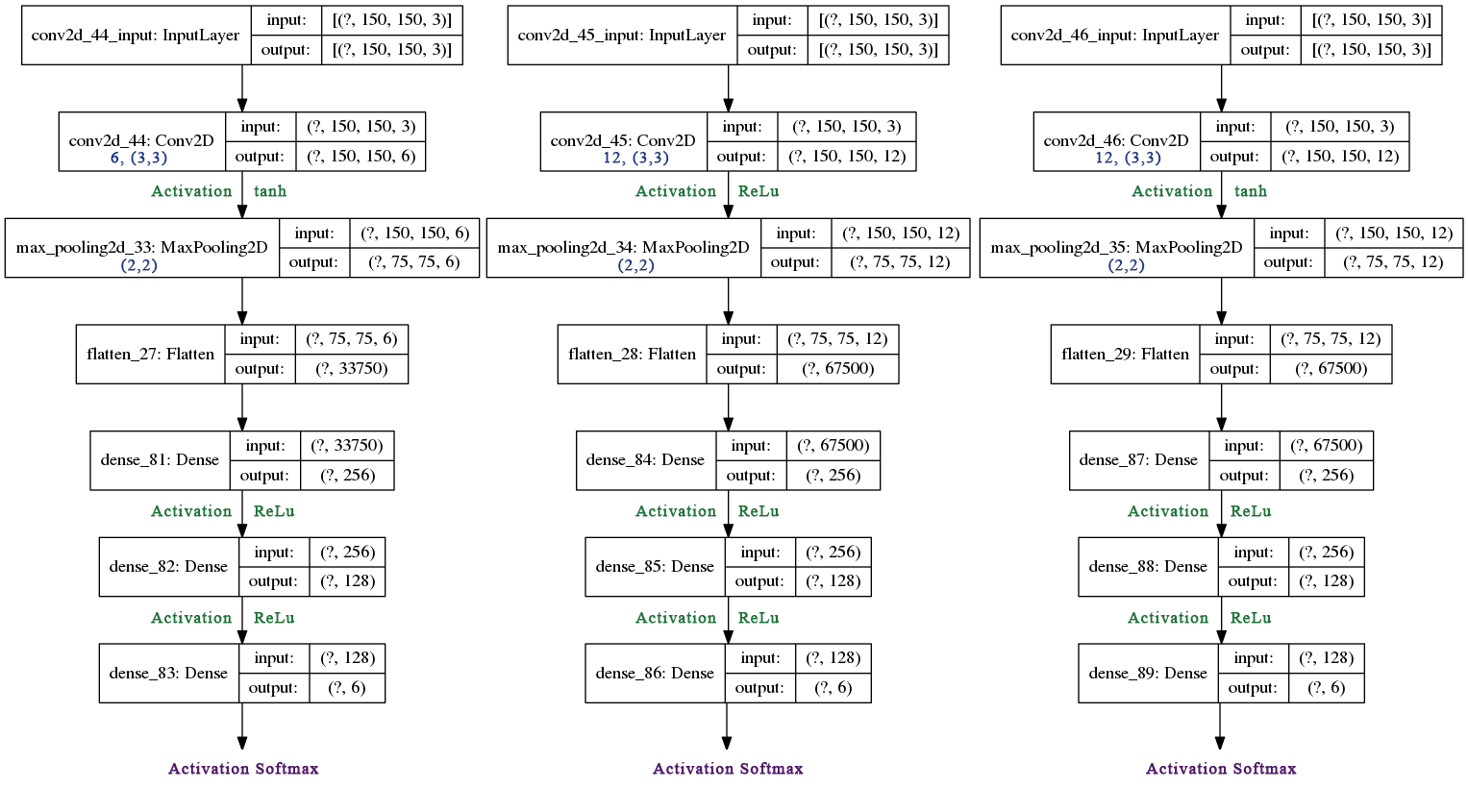
Форма x: (batch\_size, 150, 150, 3).

Форма y: (batch\_size, 6).

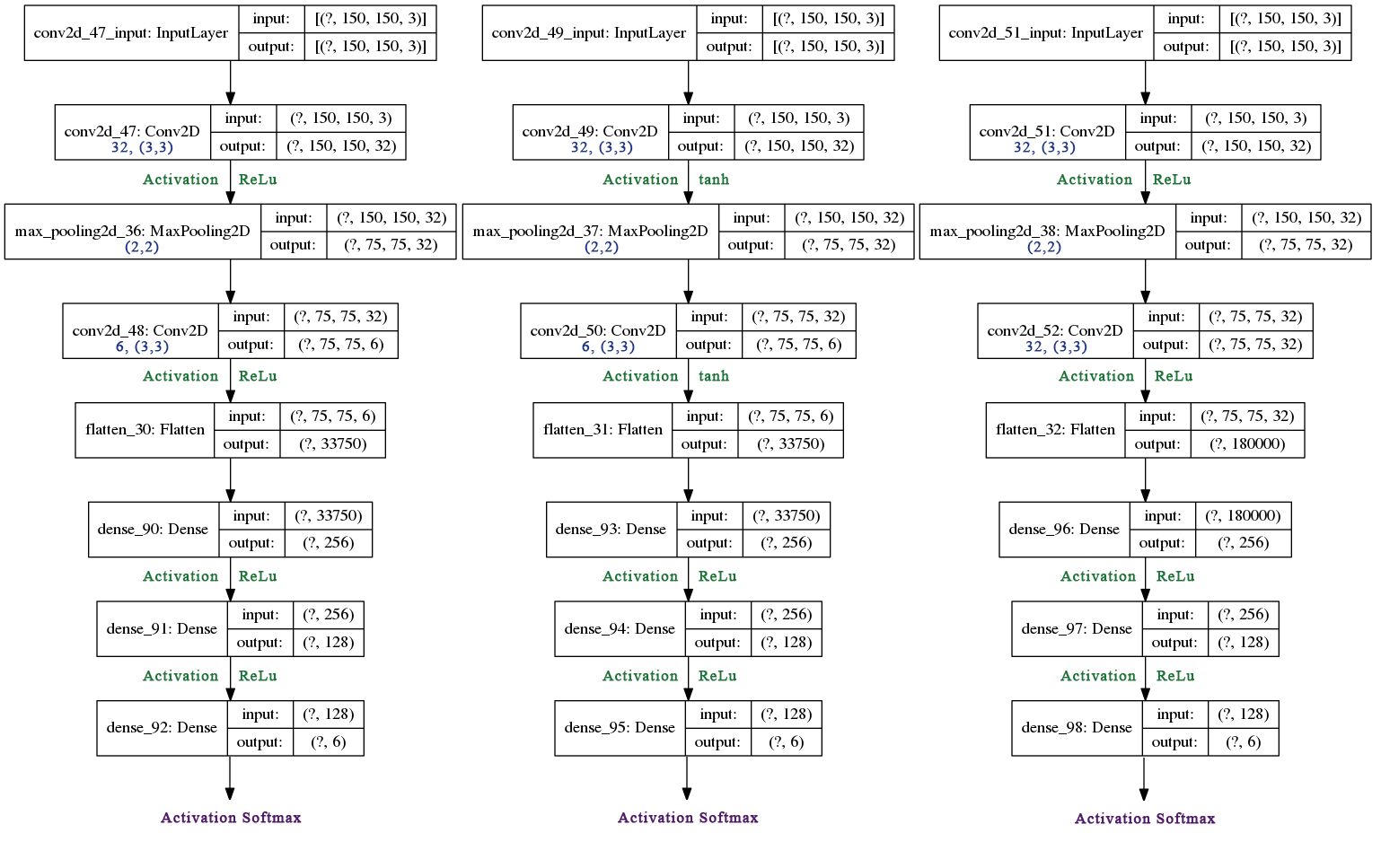
Методу fit\_generator подается на вход генератор данных в формате (x, y). Сети подается на вход массив numpy формата (150, 150, 3), который "сглаживается" сетью с помощью метода Flatten().



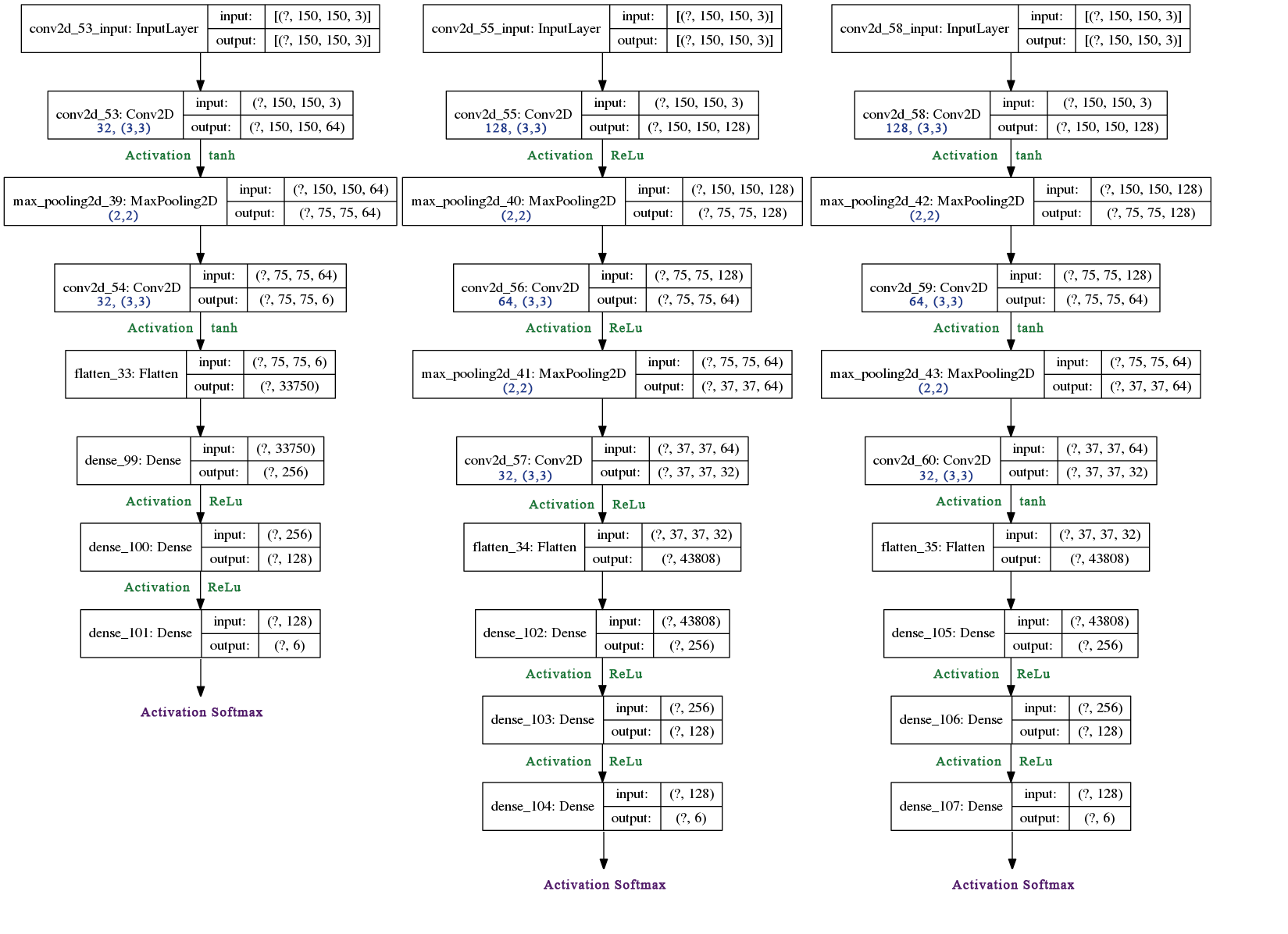
Конфигурации 1-3



Конфигурации 4-6



Конфигурации 7-9



Конфигурации 10-12

**Результаты экспериментов**

В таблице приведены конфигурация системы и программное обеспечение, с помощью которых проводилось обучение и тестирование построенных моделей.

|  |  |
| --- | --- |
| *Параметры* | *Версия* |
| GPU | Tesla P100, having 3584 CUDA cores, 16GB(16.28GB Usable) GDDR6 VRAM Tesla P100 Spec Sheet |
| Python | 3.7.5 |
| TensorFlow | 2.0.0 |

Параметры обучения:

|  |  |
| --- | --- |
| Количество эпох | 20 |
| Размер пачки | 128 |

Результаты экспериментов:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер сети | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Количество скрытых нейронов | 128  6 | 128  6 | 128  6 | 180  128  6 | 180  128  50  6 | 180  128  50  6 | 180  128  50  6 |
| Количество скрытых слоев | 4 | 5 | 6 | 8 | 10 | 10 | 10 |
| Функция активации | relu | tanh | relu | relu | relu | tanh | tanh |
| Батч | 128 | 128 | 128 | 128 | 128 | 128 | 128 |
| Количество эпох | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| Скорость обучения | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 |
| Оптимизатор | adam | adam | adam | adam | adam | adam | adam |
| Общее время | 11:39 | 13:38 | 15:06 | 14:28 | 37:41 | 39:08 | 38:15 |
| Точность (Accuracy) на тренировочном наборе, % | 99.80 | **99.91** | 85.55 | 90.97 | 91.09 | 85.24 | 77.61 |
| Ошибка на тренировочном наборе | 0.0135 | 0.071 | 0.401 | 0.2511 | 0.2606 | 0.4344 | 0.6398 |
| Точность (Accuracy) на тестовом наборе, % | 79.47 | 80.33 | 82.37 | **84.07** | 84.03 | 82.13 | 75.57 |
| Ошибка на тестовом наборе | 0.5915 | 0.6261 | 0.5070 | 0.4767 | 0.5015 | 0.5179 | 0.6688 |

**Анализ результатов**

Варьируемые параметры:

* Количество фильтров (32, 64, 128)
* Количество свёрточных слоёв (троек, состоящих из свёртки, функции активации и пространственного объединения)
* Вид функции активации

Рассмотрим конфигурацию №1 с одним свёрточным слоем и числом фильтров равным 3 (в два раза меньше, чем число классов), с функцией активации relu. На этом примере замечаем, что при малом количестве фильтров в сверточном слое (меньше, чем число классов 6), сетка быстро переобучается.

Рассмотрим конфигурацию №2 с одним свёрточным слоем, числом фильтров равным 3, функцией активации tanh, и получим, что:

* Точность на тестовой выборке на 20% выше, чем лучший результат, полученный в первой лабораторной работе (полносвязные сети);
* Тем не менее, есть определённая уверенность, что полученный результат можно улучшить, поскольку наблюдается довольно быстрое переобучение (уже на четвёртой итерации точность на обучающей выборке больше, на 30% выше, чем на тестовой и почти достигает максимального значения.
* Тестовую точность можно улучшить за счёт увеличения количества фильтров в сети как минимум до числа классов и дальше. Это объясняется тем, что полученный на выходе свёрточного слоя трёхмерный тензор можно рассматривать как пиксель исходного изображения и соответствующая ему вероятность принадлежности его к одному из классов. Для такой интерпретации потребуется как минимум 6 фильтров (число классов в задаче). Проверим нашу теорию в следующей конфигурации №3 (с relu), №4 (с tanh).

Конфигурации №3, №4 дают нам такие результаты:

* Мы улучшили точность на тестовой выборке на 5-6 процентов
* Удалось снизить скорость переобучения, так на четвёртой итерации разница между точностью на обучающей и тестовой выборке сократилась на 10% по сравнению с прошлой конфигурацией
* Наша теория о том, что число фильтров должно быть как минимум равно числу классов подтвердилась и дала закономерную прибавку в точности.

В конфигурации №5 (с relu), №6 (с tanh) продолжим увеличивать число фильтров и оставим пока всё так же один свёрточный слой. Гипотеза: заметной прибавки в тестовой точности это не даст. Результаты:

* Как и ожидалось, сколь-нибудь ощутимой прибавки в точности на тестовой выборке по сравнению с предыдущей конфигурацией мы не получили
* Дальнейшее увеличение числа фильтров при одном свёрточном слое особо смысла не имеет: используя 12 фильтров вместо 6 (число классов), мы начинаем классифицировать пиксели уже не на 6 классов, а на 12, но данных для такой классификации недостаточно (поскольку в действительности у нас всего 6 классов).

В конфигурации №7 (с relu), №8 (с tanh) введём ещё один свёрточный слой. Первый слой будет иметь 32 фильтра, второй - 6 фильтров. Такая "двухуровневая" свёртка имеет следующую интерпретацию: сперва мы попробуем распознать более сложные элементы изображения, а затем на основании полученного выхода проведём уже привычную свёртку с 6 фильтрами, выход который можно будет интерпретировать так: пикселю исходного изображение и соответствующей ему вероятности принадлежности к одному из классов. Результат: получили небольшой рост точности.

В конфигурации №9, №10 увеличим число фильтров во второй свёртке до 32. И убедились, что лучше не стало.

Поскольку переобучение всё так же имеет место быть, попробуем снизить его влияние, добавив ещё один свёрточный слой в конфигурации №11, №12. Таким образом, сделав сеть глубже мы добились еще немного прироста точности.

**Выводы**

В ходе выполнения лабораторной работы была получена архитектура сверточной нейронной сети №11, которая позволяет решать выбранную практическую задачу с высокими показателями качества. Была спроектирована и разработана программная реализация, позволяющая обучать различные конфигурации нейронных сетей. С помощью полученной реализации были произведены эксперименты на выбранном наборе данных. Во время экспериментов была измерена ошибка классификации, а также произведен сбор результатов при разном наборе параметров. Полученные результаты отражены в настоящем отчете.