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

# Постановка задачи

## Цели

Цель настоящей 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.

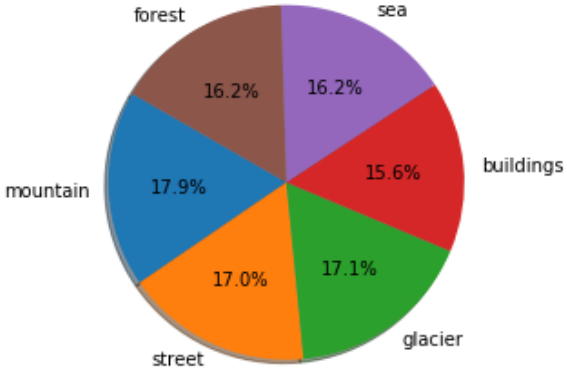
Тестовая выборка содержит 3000 изображений.

Размер каждого изображения 150x150.

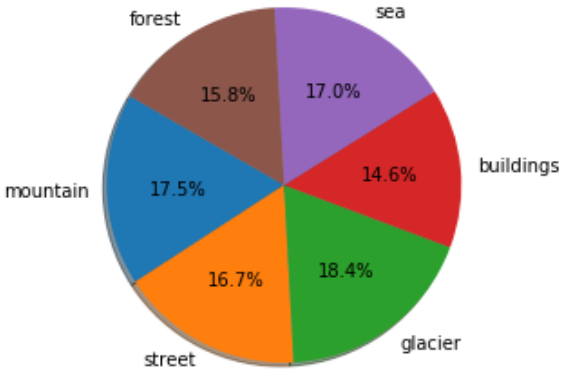
| *№* | *Категории* | *Размер тренировочной выборки* | *Размер тестовой выборки* |
|---|---|---|---|
| 1 | mountain | 2512 | 525 |

| | | | |
|---|---|---|---|
| 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.

$$accuracy(y_{pred}, y_{true}) = \frac{1}{N}\sum_{i=1}^{N} 1(y_{pred_i} == y_{true_i})$$

## Разработанные программы

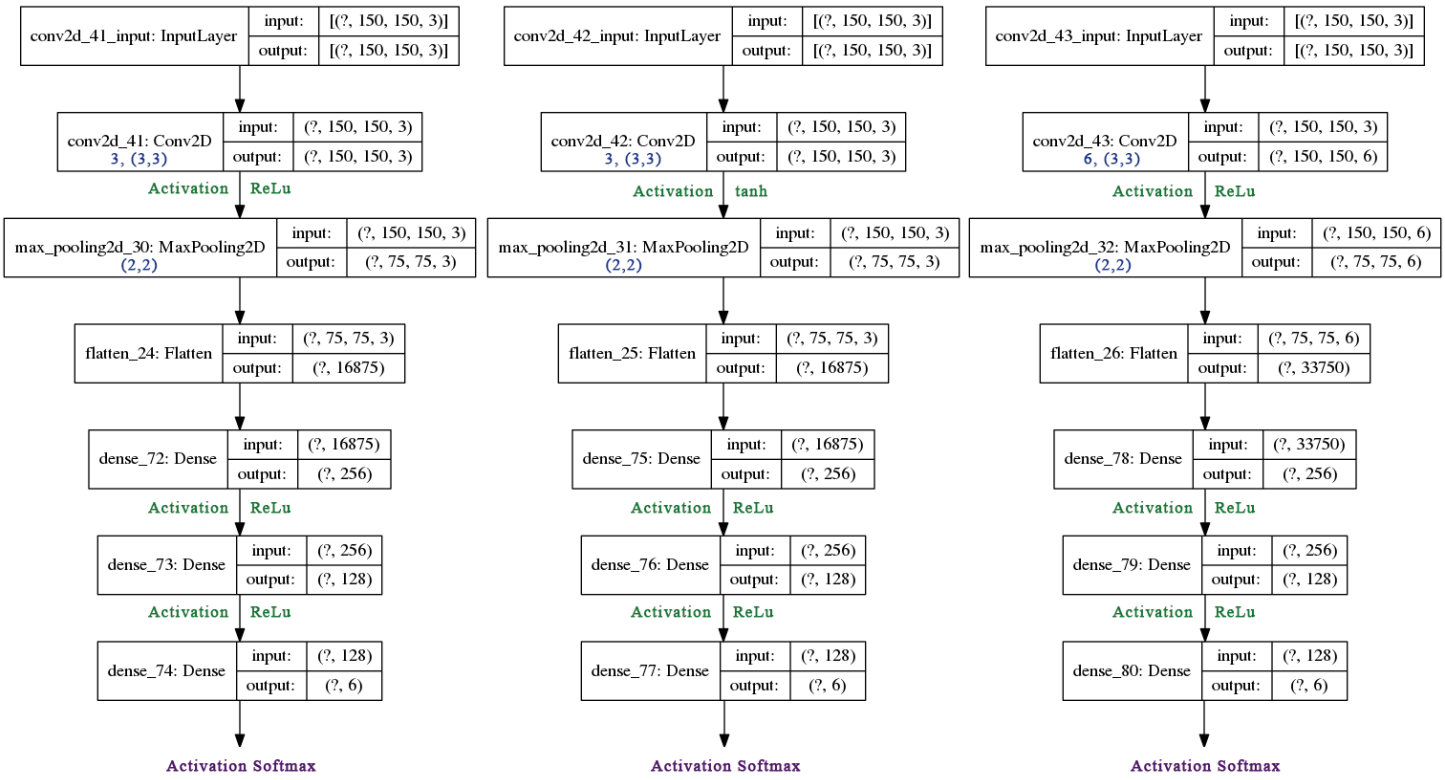
Lab3.ipynb – скрипт для обучения свёрточных нейронных сетей.

## Тестовые конфигурации сетей

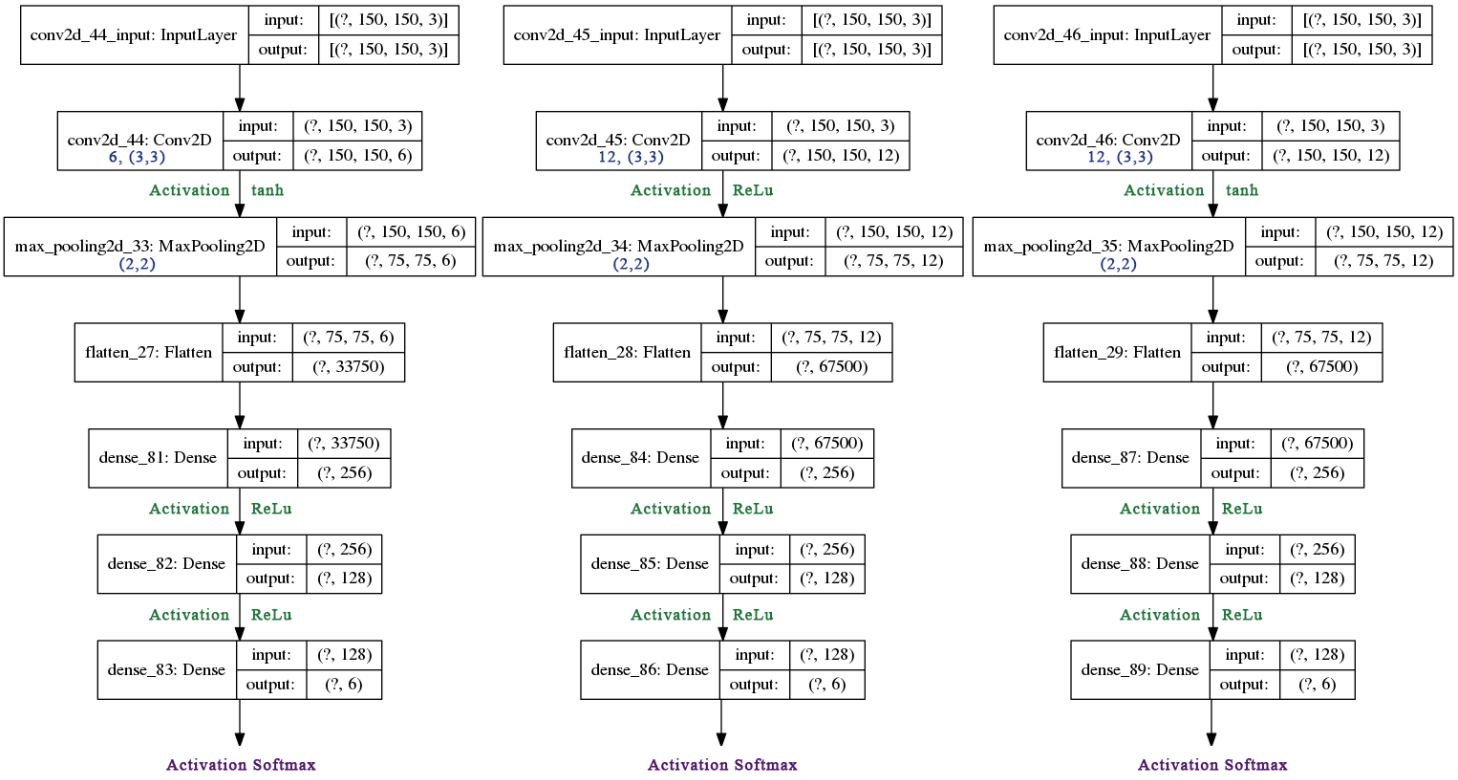
С помощью класса ImageDataGenerator и его метода flow_from_directory() генерируем пакеты. Данные возвращаются в формате (x, y), где x, y - numpy массивы.

Форма x:  (batch_size, 150, 150, 3).
Форма y: (batch_size, 6).

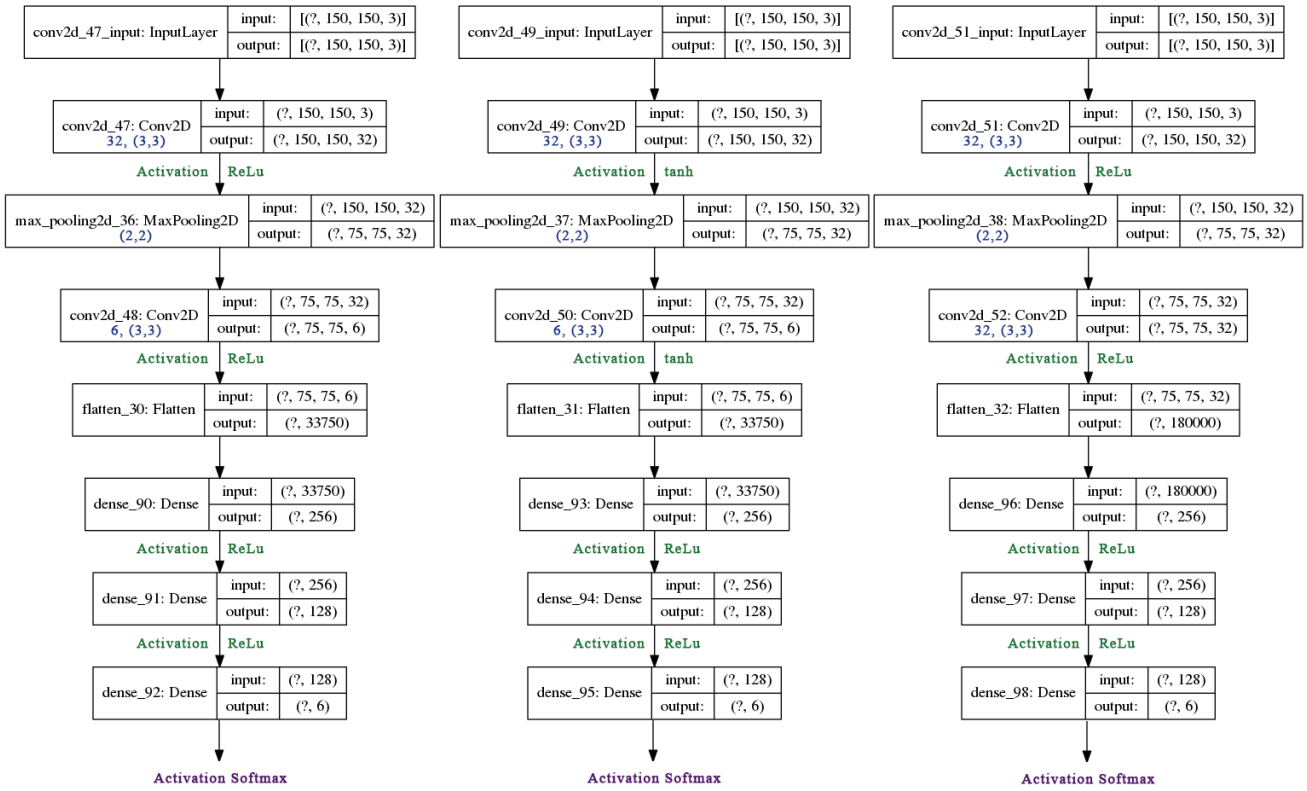
Методу fit_generator подается на вход генератор данных в формате (x, y). Сети подается на вход массив numpy формата (150, 150, 3), который "сглаживается" сетью с помощью метода Flatten().



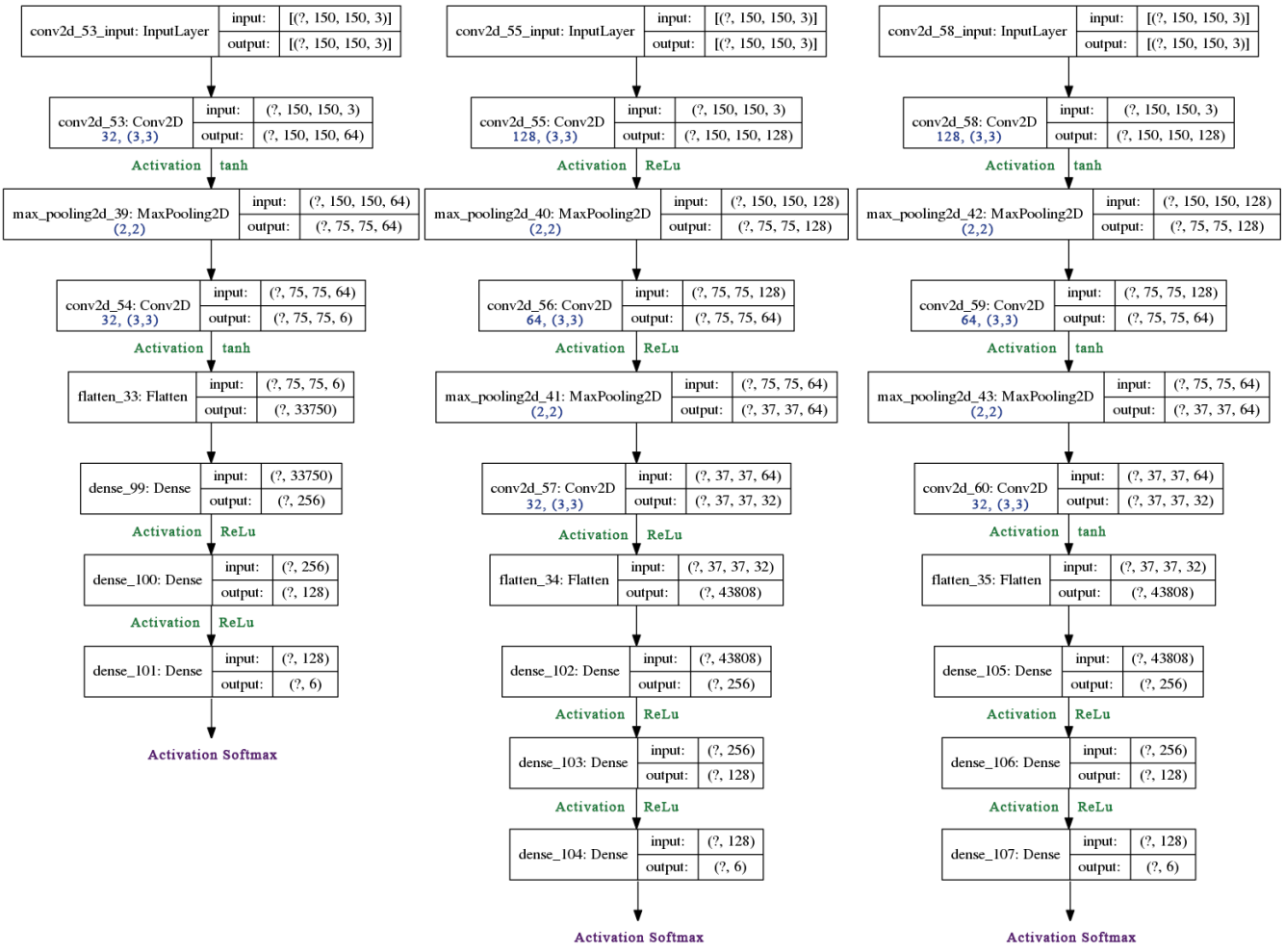

Конфигурации 1-3




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 |
|---|---|---|---|---|---|---|
| Количество скрытых нейронов | 256<br>128<br>6 | 256<br>128<br>6 | 256<br>128<br>6 | 256<br>128<br>6 | 256<br>128<br>6 | 256<br>128<br>6 |
| Количество скрытых слоев | 4 | 4 | 4 | 4 | 4 | 4 |
| Функция активации | relu | tanh | relu | tanh | relu | tanh |
| Батч | 128 | 128 | 128 | 128 | 128 | 128 |
| Количество эпох | 15 | 15 | 15 | 15 | 15 | 5 |
| Скорость обучения | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 |
| Оптимизатор | adam | adam | adam | adam | adam | adam |
| Общее время | 8:57 | 8:52 | 8:53 | 8:49 | 8:49 | 8:48 |
| Точность (Accuracy) на тренировочном наборе, % | 99.66 | 98.07 | 99.87 | 99.64 | 99.11 | 98.50 |
| Ошибка на тренировочном наборе | 0.0296 | 0.0701 | 0.0121 | 0.0209 | 0.0320 | 0.0564 |
| Точность (Accuracy) на тестовом наборе, % | 64.40 | 64.67 | 69.10 | 70.40 | 72.00 | 68.60 |
| Ошибка на тестовом наборе | 1.7531 | 1.8091 | 1.6104 | 1.3845 | 1.3150 | 1.3766 |

| Номер сети | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|
| Количество скрытых нейронов | 256<br>128<br>6 | 256<br>128<br>6 | 256<br>128<br>6 | 256<br>128<br>6 | 256<br>128<br>6 | 256<br>128<br>6 |
| Количество скрытых слоев | 5 | 5 | 5 | 5 | 6 | 6 |
| Функция активации | relu | tanh | relu | tanh | relu | tanh |
| Батч | 128 | 128 | 128 | 128 | 128 | 128 |
| Количество эпох | 15 | 15 | 15 | 15 | 15 | 15 |
| Скорость обучения | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 |
| Оптимизатор | adam | adam | adam | adam | adam | adam |
| Общее время | 09:02 | 09:03 | 09:08 | 09:05 | 10:18 | 10:39 |
| Точность (Accuracy) на тренировочном наборе, % | 99.48 | 99.88 | 98.52 | 99.87 | 99.37 | 97.09 |
| Ошибка на тренировочном наборе | 0.0202 | 0.0069 | 0.0463 | 0.0080 | 0.0243 | 0.0872 |
| Точность (Accuracy) на тестовом наборе, % | 74.03 | 68.60 | 68.73 | 73.43 | **76.97** | 74.30 |
| Ошибка на тестовом наборе | 1.5115 | 1.6224 | 1.6969 | 1.3248 | 1.2341 | 1.2976 |

# 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 которой 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.