

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

Нижегород  
2019

# Оглавление

1. Постановка задачи .....	3
2. Тренировочные и тестовые наборы данных .....	4
3. Метрика качества решения .....	5
4. Разработанные программы .....	5
5. Тестовые конфигурации сетей .....	5
6. Результаты экспериментов .....	9
7. Анализ результатов .....	11

# **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.

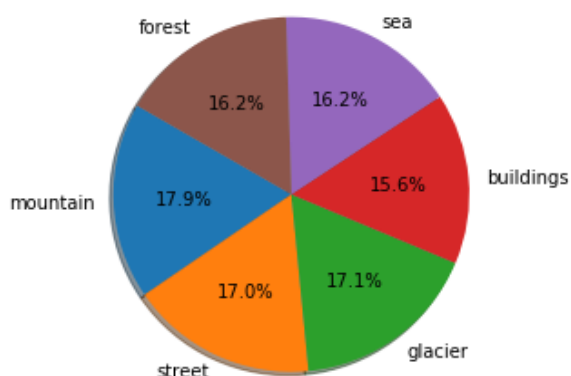
Тестовая выборка содержит 3000 изображений.

Размер каждого изображения 150x150.

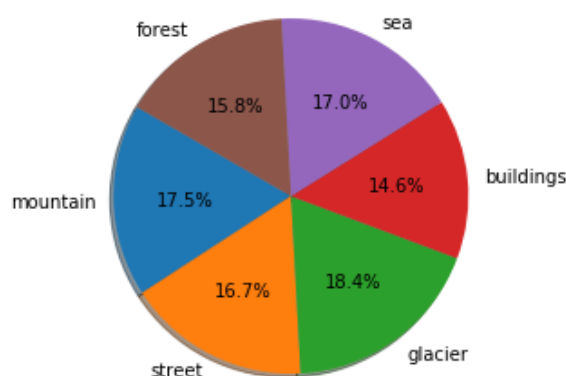
<i>№</i>	<i>Категории</i>	<i>Размер тренировочной выборки</i>	<i>Размер тестовой выборки</i>
1	mountain	2512	525

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.

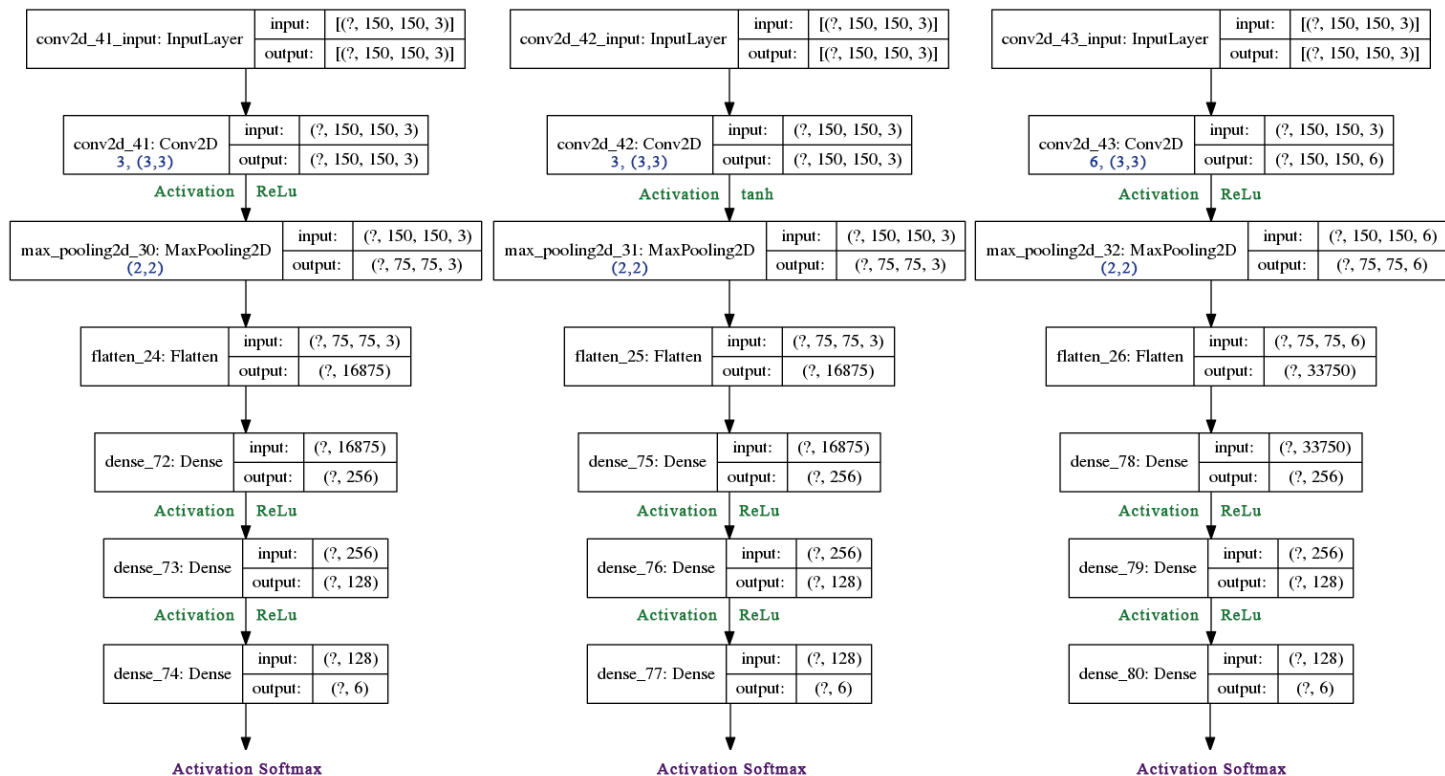
## Тестовые конфигурации сетей

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

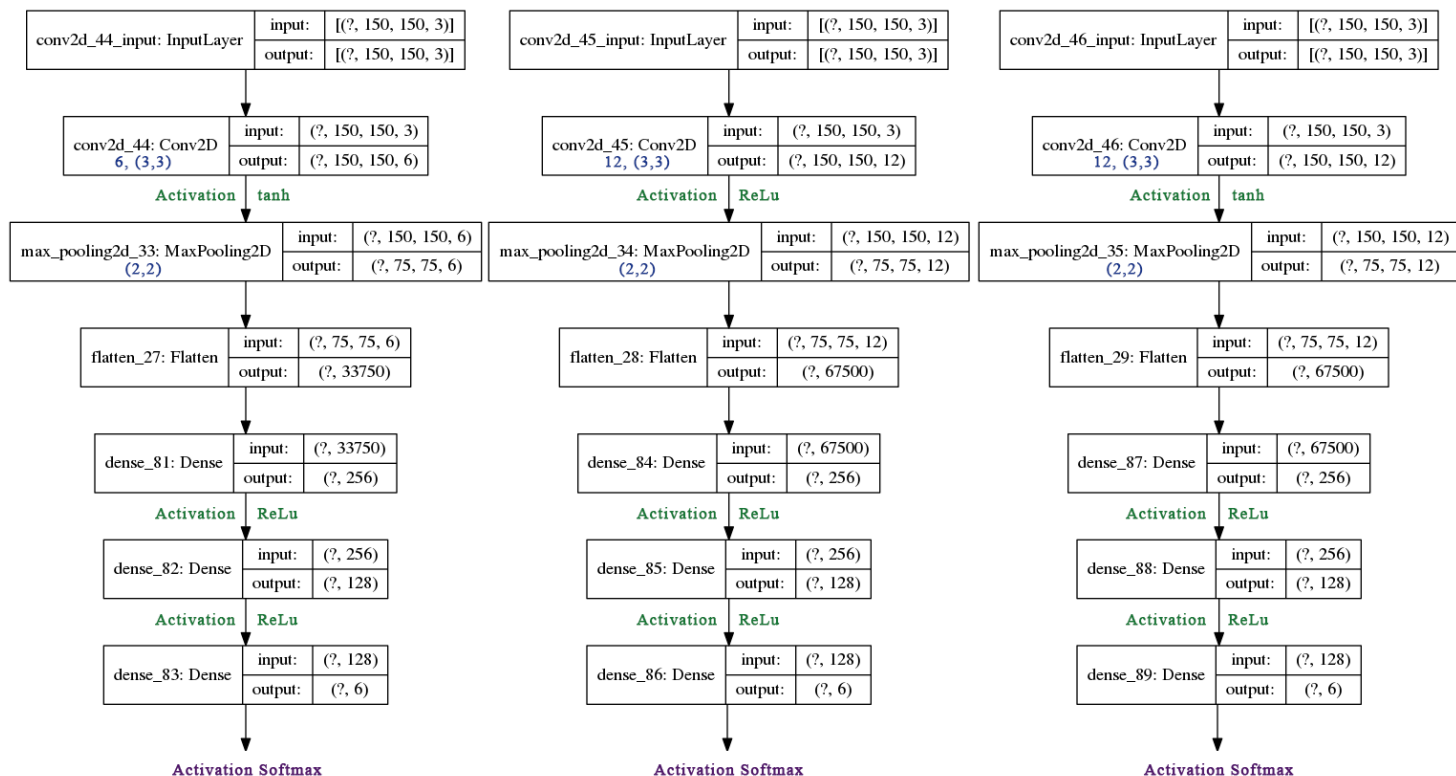
Форма x: (batch\_size, 150, 150, 3).

Форма y: (batch\_size, 6).

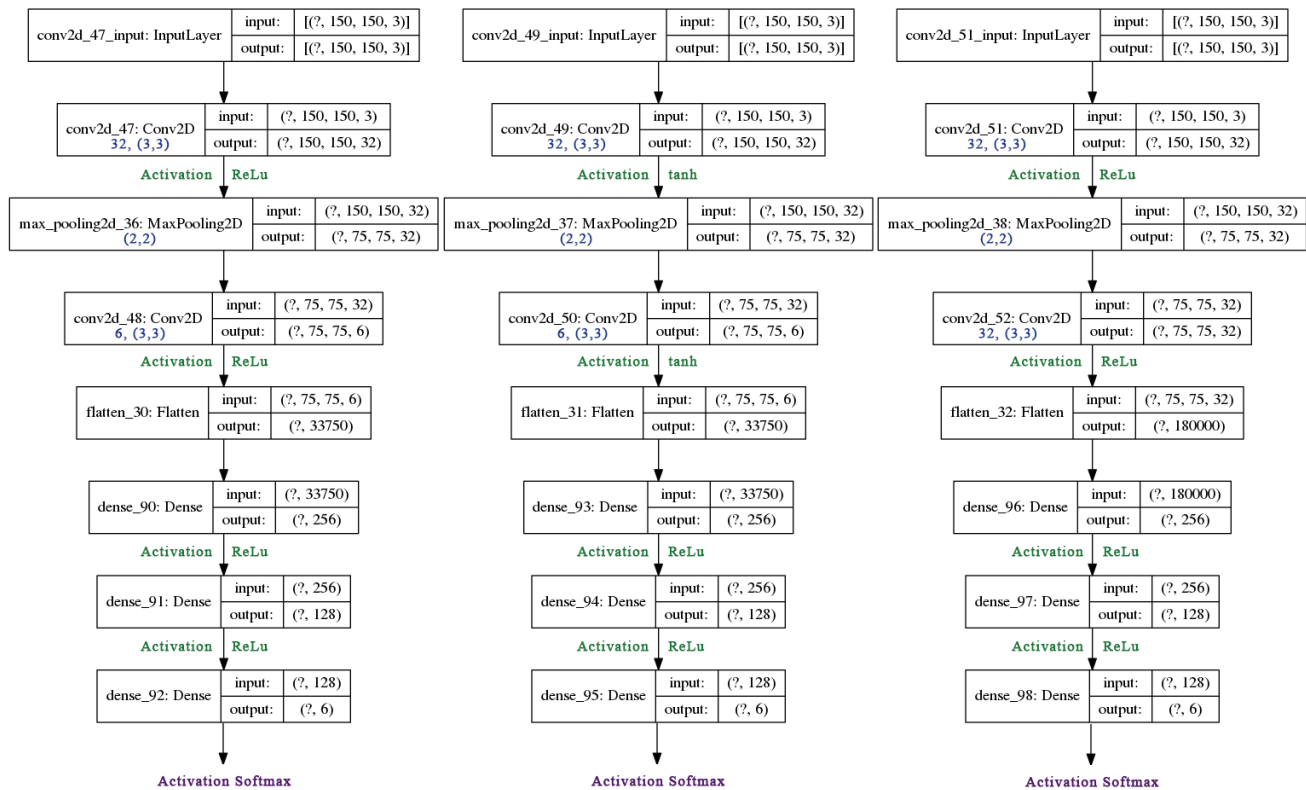
Методу `fit_generator` подается на вход генератор данных в формате (x, y). Сети подается на вход массив numpy формата (150, 150, 3), который "сглаживается" сетью с помощью метода `Flatten()`.



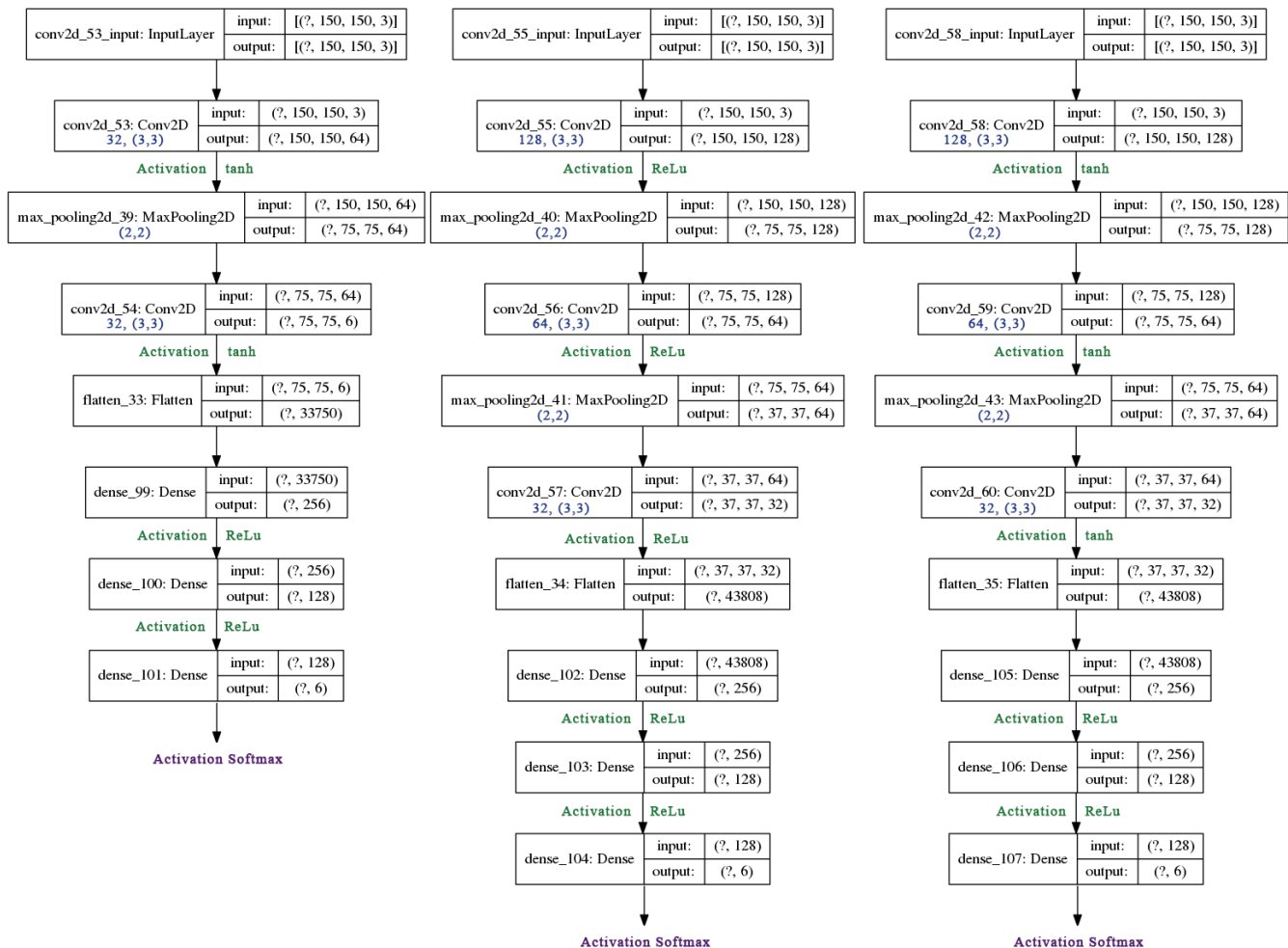
### Конфигурации 1-3



### 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.

<i>Параметры</i>	<i>Версия</i>
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	15
Размер пачки	128

## Результаты экспериментов:

Номер сети	1	2	3	4	5	6
Количество скрытых нейронов	256 128 6	256 128 6	256 128 6	256 128 6	256 128 6	256 128 6
Количество фильтров	3	3	6	6	12	12
Количество свёрточных слоёв	1	1	1	1	1	1
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	15
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 (Ассурасу) 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 (Ассурасу) 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	256 128 6	256 128 6	256 128 6	256 128 6	256 128 6	256 128 6
Количество фильтров	32 6	32 6	32 32	32 32	128 64 32	128 64 32
Количество свёрточных слоёв	2	2	2	2	3	3
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	<b>76.97</b>	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` и конфигурацию №2 с одним свёрточным слоем, числом фильтров равным 3, функцией активации `tanh`.

На этом примере замечаем, что при малом количестве фильтров в сверточном слое (меньше, чем число классов 6), сетка быстро переобучается.

- 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 (на 6-7 итерации точность на обучающей выборке более чем на 20% выше, чем на тестовой, и достигает довольно близкое к максимальному значение.).
- 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 гипотезу 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, это неплохо (учитывая, что она и до этого была не низкой и мы не вводили новых свёрточных слоёв - не углубляли сеть)
- Удалось снизить скорость переобучения, так на четвёртой итерации разница между точностью на обучающей и тестовой выборке сократилась на 5% по сравнению с прошлой конфигурацией (функция активации - тангенс).
- Наша гипотеза 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, значительной 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). Если мы повысим число фильтров, к примеру, до 32, но оставим при этом один свёрточный слой, получим довольно сильную просадку по точности.

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 изображения ставятся в соответствие вероятности принадлежности его к каждому 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 небольшого прироста точности. Продолжая дальше углублять сеть за счёт добавления новых свёрточных слоёв, вполне вероятно, можно добиться и более высокой точности.

## **Выводы**

В ходе выполнения лабораторной работы была получена архитектура свёрточной нейронной сети №11, которая позволяет решать выбранную практическую задачу с достаточно 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.