Отчет. Лабораторная работа №3.

ПОСТАНОВКА ЗАДАЧИ

В лабораторной работе №2 и 3 в качестве входных данных берется GTSRB - немецкий тест распознавания дорожных знаков. Ставится задача - разработать полностью связанную нейронную сетей с использованием одной из библиотек глубокого обучения для решения выбранной задачи. Цель экспериментов - поиск оптимальных параметров настройки сети.

Такими параметрами являются:

- Количество слоев
- Количество эпох
- Функция активаций в скрытых слоях
- Функция инициализации весов в скрытых слоях
- Функция оптимизации
- Размер ядра свертки
- Количество входных фильтров в свертке

Классификация

Задача классификации — это задача присвоения меток объектам. Например, если объекты — это фотографии, то метками может быть содержание фотографий: содержит или изображение пешехода или нет, изображен ли мужчина или женщина, какой породы собака изображена на фотографии. Обычно есть набор взаимоисключающих меток и сборник объектов, для которых эти метки известны. Имея такую коллекцию данных, необходимо автоматически расставлять метки на произвольных объектах того же типа, что были в изначальной коллекции. Давайте формализуем это определение.

Допустим, есть множество объектов X. Это могут быть точки на плоскости, рукописные цифры, фотографии или музыкальные произведения. Допустим также, что есть конечное множество меток Y. Эти метки могут быть пронумерованы. Мы будем отождествлять метки и их номера. Таким образом $Y = \{red, green, blue\}$ в нашей нотации будет обозначаться как $Y = \{1, 2, 3\}$. Если $Y = \{0, 1\}$, то задача называется задачей бинарной классификации, если меток больше двух, то обычно говорят, что это просто задача классификации. Дополнительно, у нас есть входная выборка $D = \{(x_i, y_i), x_i \in X,\ y_i \in Y,\ i = [1, ..., N]\}$. Это те самые размеченные примеры, на которых мы и будем обучаться проставлять метки автоматически. Так как мы не знаем классов всех объектов точно, мы считаем, что класс объекта — это случайная величина, которую мы для простоты тоже будем обозначать . Например, фотография собаки может классифицироваться как собака с вероятностью 0.99 и как кошка с

вероятностью 0.01. Таким образом, чтобы классифицировать объект, нам нужно знать условное распределение этой случайной величины на этом объекте $p(y|x)$.

Задача нахождения $p(y|x)$ при данном множестве меток Y и данном наборе размеченных примеров $D = \{(x_i, y_i), x_i \in X,\ y_i \in Y,\ i = [1, ..., N]\}$ называется задачей классификации.

Вероятностная постановка задачи классификации

Чтобы решить эту задачу, удобно переформулировать ее на вероятностном языке. Итак, есть множество объектов X и множество меток Y. $\xi : \Omega \rightarrow X$ — случайная величина, представляющая собой случайный объект из X. $\eta : \Omega \rightarrow Y$ — случайная величина, представляющая собой случайную метку из Y. Рассмотрим случайную величину $(\xi, \eta) : \Omega \rightarrow (X, Y)$ с распределением $p(x|y)$, которое является совместным распределением объектов и их классов. Тогда, размеченная выборка — это сэмплы из этого распределения $(x_i, y_i) \sim p(x, y)$. Мы будем предполагать, что все сэмплы независимо и одинаково распределены.

Задача классификации теперь может быть переформулирована как задача нахождения $p(x|y)$ при данном сэмпле $D = \{(x_i, y_i), x_i \in X,\ y_i \in Y,\ i = [1, ..., N]\}$.

ТРЕНИРОВОЧНЫЕ И ТЕСТОВЫЕ НАБОРЫ ДАННЫХ

Характеристики примеров

Количество классов	43
Максимальная ширина	243
Максимальная высота	225
Минимальная ширина, высота	25
Медиана для ширины, высоты	43

Распределение количества изображений по классам

Для тренировочного набора данных:

Class Id	Image number	Class Id	Image number	Class Id	Image number
0	210	15	630	30	450
1	2220	16	420	31	780
2	2250	17	1110	32	240
3	1410	18	1200	33	689
4	1980	19	210	34	420
5	1860	20	360	35	1200
6	420	21	330	36	390
7	1440	22	390	37	210
8	1410	23	510	38	2070
9	1470	24	270	39	300
10	2010	25	1500	40	360
11	1320	26	600	41	240

12	2100	27	240	42	240
13	2160	28	540		
14	780	29	270		

Для тестового набора данных:

Class Id	Image number	Class Id	Image number	Class Id	Image number
0	60	15	210	30	150
1	720	16	150	31	270
2	750	17	360	32	60
3	450	18	390	33	210
4	660	19	60	34	120
5	630	20	90	35	390
6	150	21	90	36	120
7	450	22	120	37	60
8	450	23	150	38	690
9	480	24	90	39	90
10	660	25	480	40	90
11	420	26	180	41	90
12	690	27	60	42	90
13	720	28	150		
14	270	29	90		

Примеры изображений из каждого класса

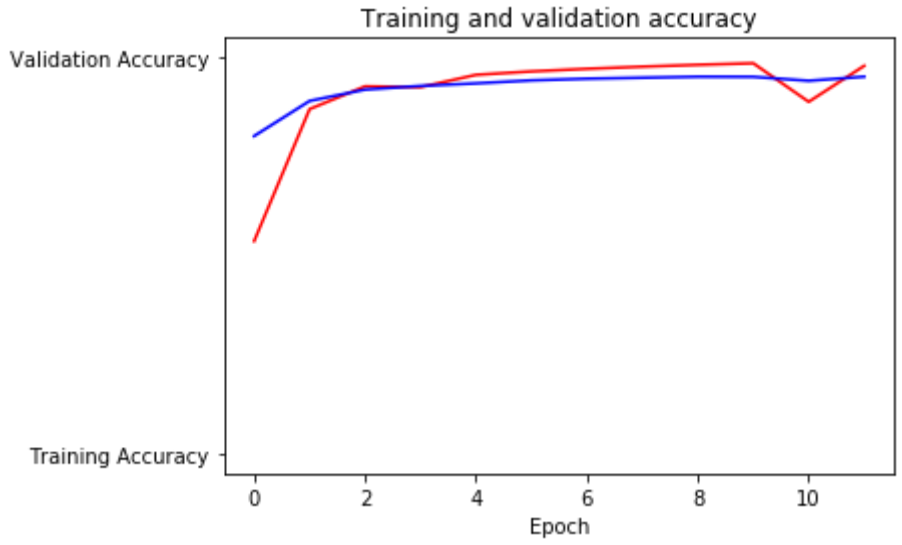
МЕТРИКА КАЧЕСТВА РЕШЕНИЯ ЗАДАЧИ

Тренировочная выборка была разделена на основную тренировочную и проверочную(**validation**), для улучшения качества обучения и лучшей оценки получившегося результата, при использовании в реальных условиях.

За **метрику качества решения** взят максимальный процент, правильно определенных, дорожных знаков на проверочной выборке за время обучения.

Пример обучения

Epoch	Accuracy	Validation accuracy	Epoch	Accuracy	Validation accuracy
1	0.5367	0.8020	7	0.9717 ▲	0.9467 ▲
2	0.8700 ▲	0.8906 ▲	8	0.9771 ▲	0.9492 ▲
3	0.9274 ▲	0.9193 ▲	9	0.9818 ▲	0.9517 ▲
4	0.9254 ▼	0.9285 ▲	10	0.9857 ▲	0.9514 ▼
5	0.9565 ▲	0.9350 ▲	11	0.8884 ▼	0.9415 ▲
6	0.9651 ▲	0.9425 ▲	12	0.9793 ▲	**0.9518** ▲



ИСХОДНЫЙ ФОРМАТ ХРАНЕНИЯ ДАННЫХ

Данные представляют собой набор изображений имеющие набор параметров

№	Path	Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId
0	Test/00000.png	53	54	6	5	48	49	16
1	Test/00001.png	42	45	5	5	36	40	1
2	Test/00002.png	48	52	6	6	43	47	38
3	Test/00003.png	27	29	5	5	22	24	33
n	...	...	...	...	...	...	...	...

Roi - отступы от левого верхнего угла для определения положения дорожного знака на изображении.

ФОРМАТ, В КОТОРОМ ДАННЫЕ ПРЕДОСТАВЛЯЮТСЯ НА ВХОД СЕТИ

Перед загрузкой в сеть входные данные обрабатываются:

1. Размер изображений уменьшается, или увеличивается до величины 40x40
2. Изображения раскладываются из матрицы в вектор

РАЗРАБОТАННЫЕ ПРОГРАММЫ/СКРИПТЫ

Программа состоит из одного файла *Jupyter Notebook*. Код написан на языке программирования *Python*, с помощью *Keras* - оболочки для фреймворка *TensorFlow*.

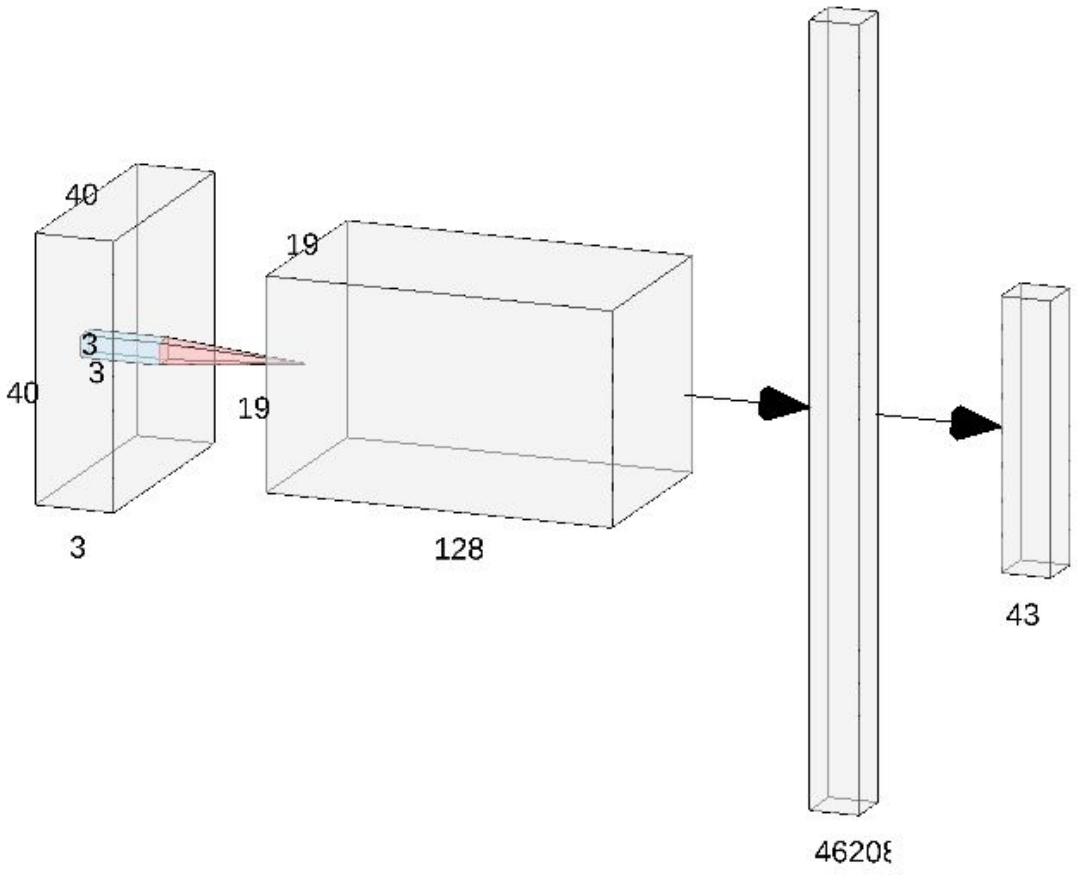
Jupyter Notebook – это крайне удобный инструмент для создания красивых аналитических отчетов, так как он позволяет хранить вместе код, изображения, комментарии, формулы и графики. В *Jupyter Notebook* сразу видно, что возвращает та или иная функция, что особенно важно в начале, при ознакомляемся с данными (показывает изображения, статистику и прочее). Выводы дополняются комментариями для более простого понимания кода.

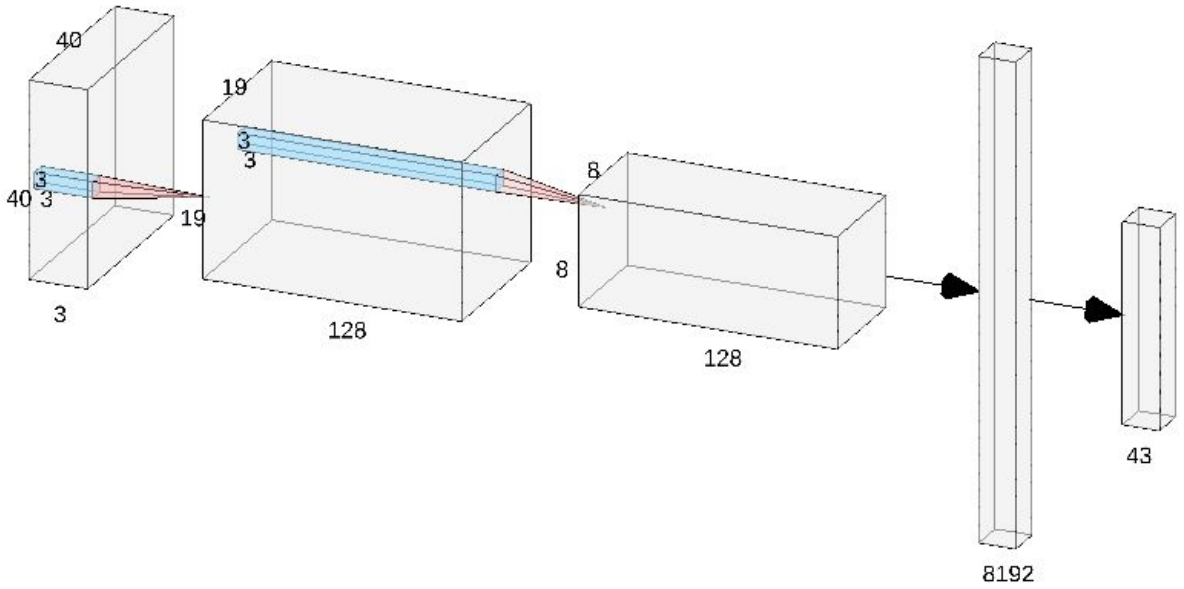
ТЕСТОВЫЕ КОНФИГУРАЦИИ СЕТЕЙ

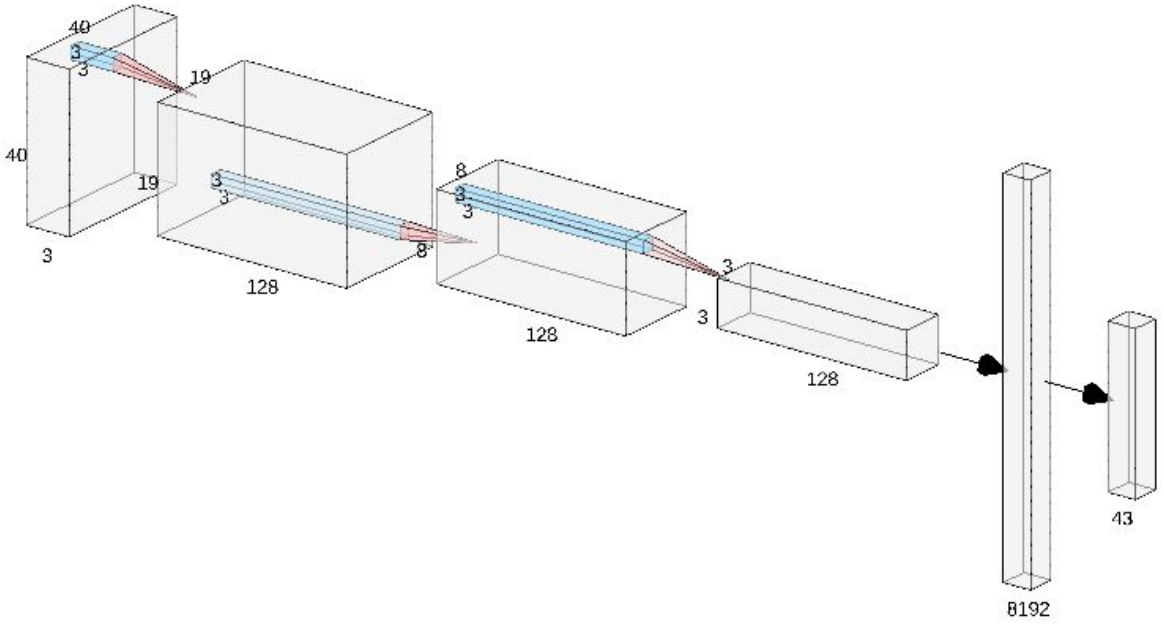
Начальные параметры

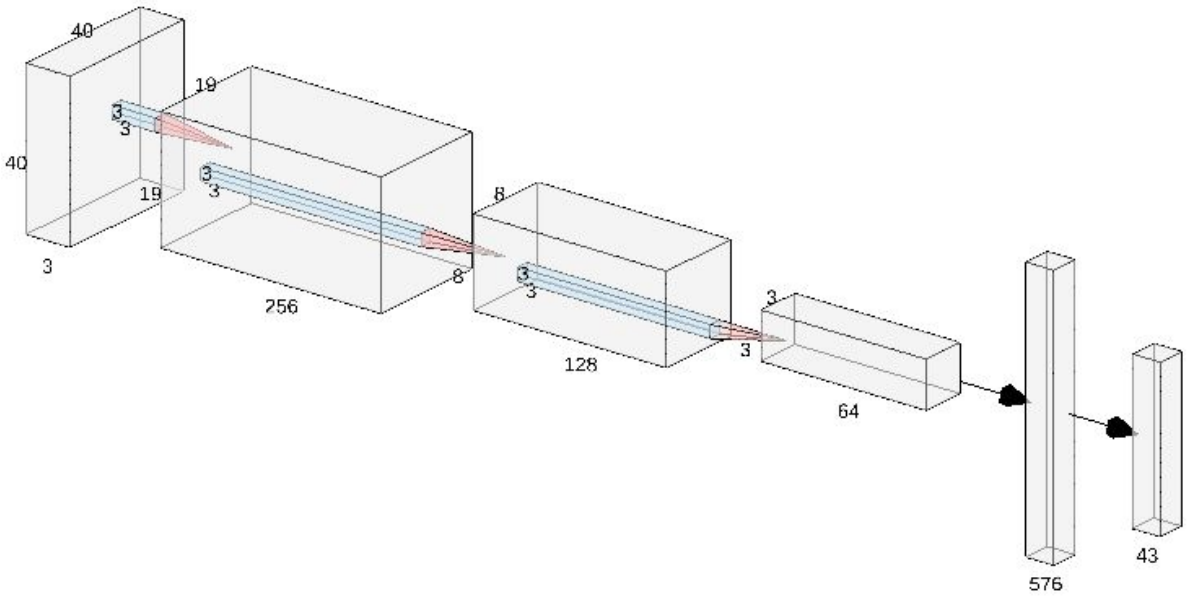
Parameters	Input Layer	Conv2D	MaxPooling2D	Flatten	Dense
Input	[40, 40, 3]	(40, 40, 3)	(38, 38, 100)	(19, 19, 100)	36100
Output	[40, 40, 3]	(38, 38, 100)	(19, 19, 100)	36100	43
Activation function	-	relu	-	-	softmax

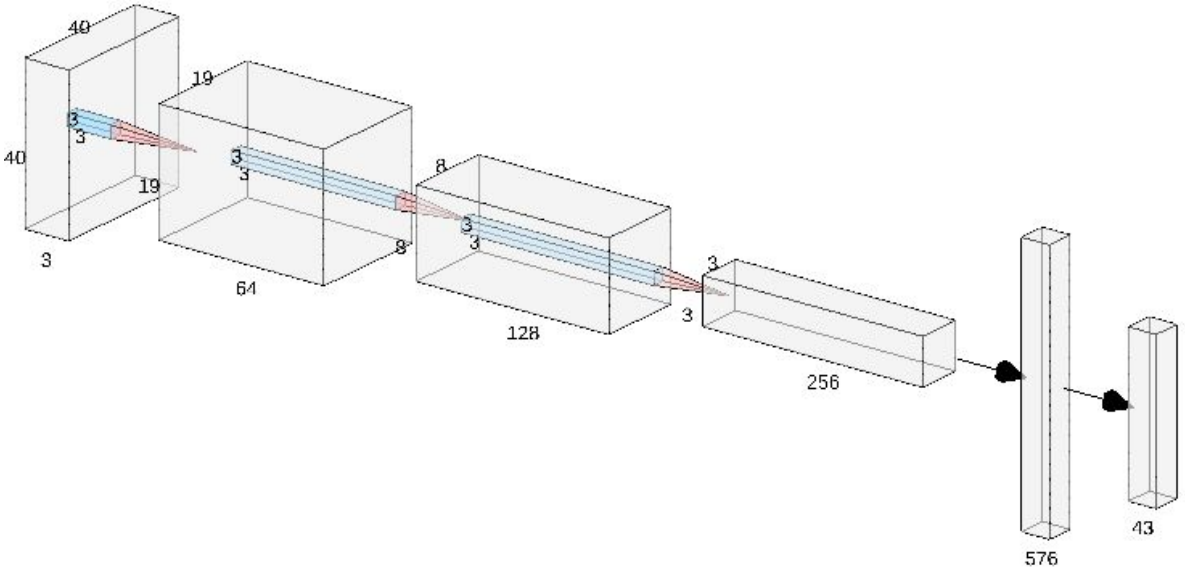
Схемы использованных сетей

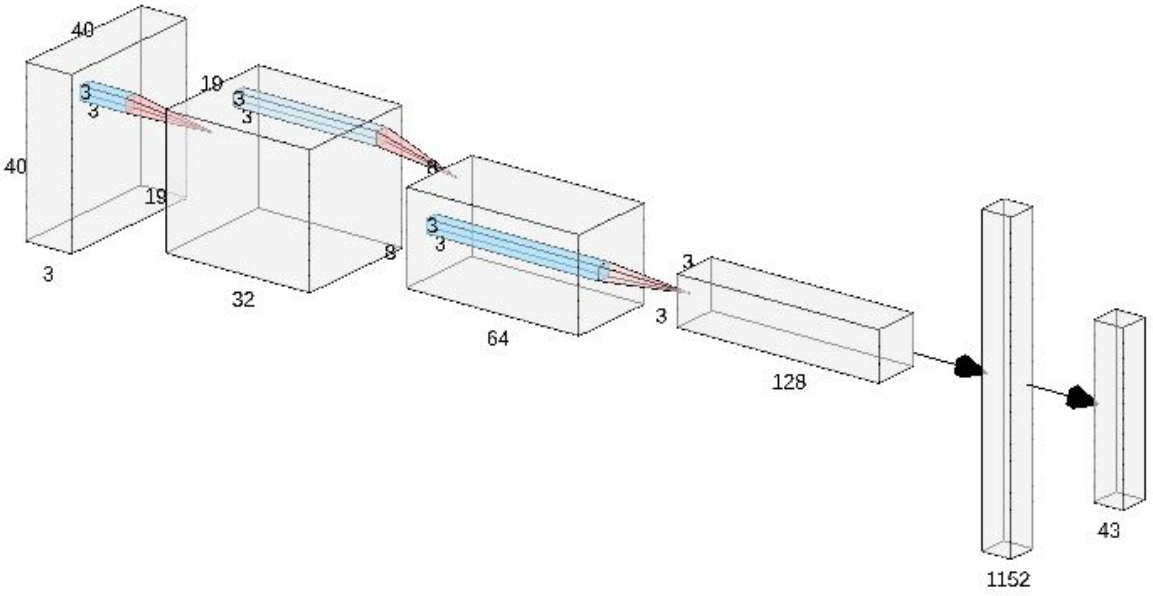


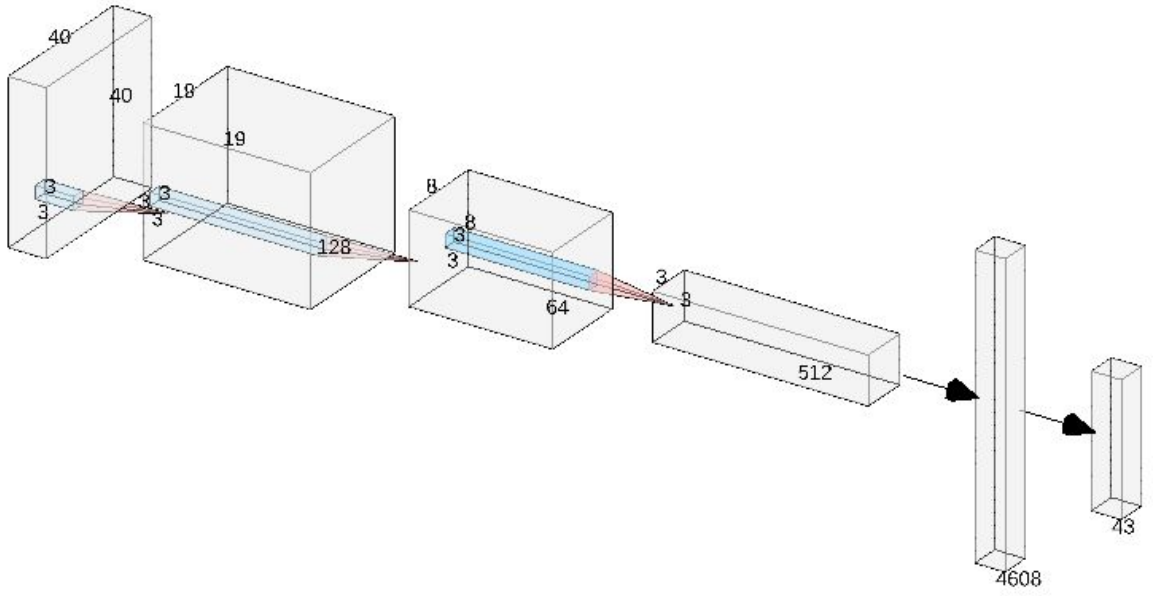

40
40
3
3
3
19
19
3
3
128
8
8
128
8192
43

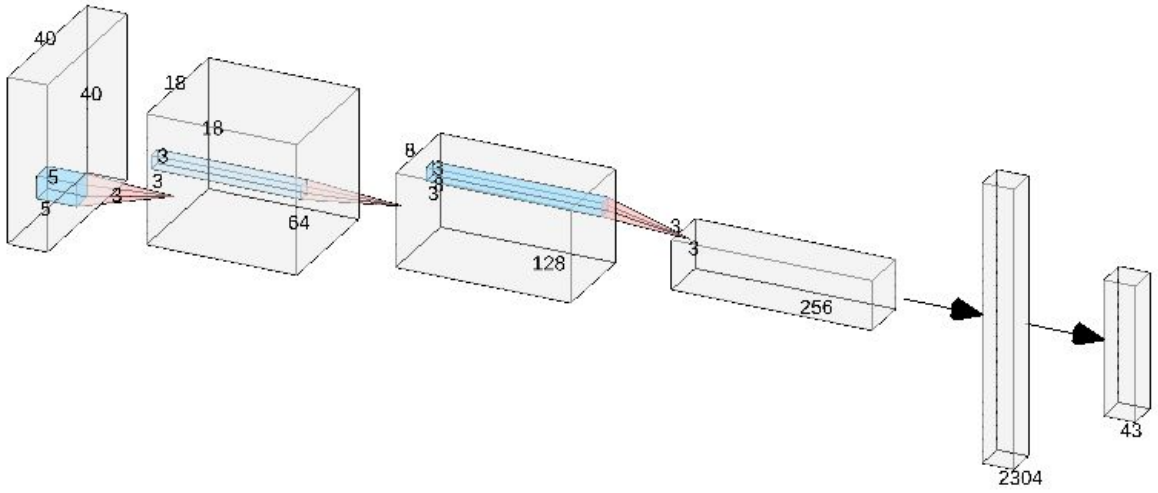

40
3
3
40
3
19
19
3
3
128
8
3
3
8
128
3
3
128
8192
43


40
40
3
19
19
3
3
3
3
256
8
8
3
3
128
3
3
64
576
43


40
40
3
19
19
3
3
3
3
64
8
8
3
3
128
3
3
256
576
43


40
40
3
3
3
19
19
3
3
32
8
8
3
3
64
3
3
128
1152
43


40
40
3
3
3
19
19
3
3
128
8
8
3
3
64
3
3
512
4608
43


40
40
5
5
18
18
3
3
64
8
3
3
128
3
3
256
2304
43

РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

В процессе экспериментов запускалось обучение нейронной сети с различным набором параметров.

Таблица с результатами

Для всех схем нейронных сетей заданы следующие параметры:

Функция оптимизации	Adam
Скорость обучения	0.0005
Количество эпох	20
Функция инициализации	glorot_uniform

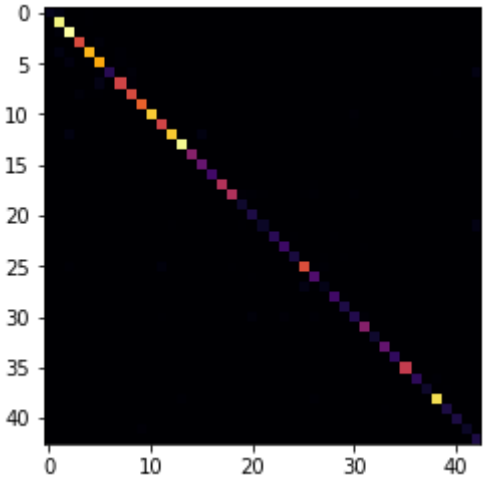
№	Layers Conv2D(filters, kernel size, activation function) MaxPooling2D(Pool size)	Тренировочная точность	Валидационная точность	Время (мин)
0	Conv2D(128, (3, 3), relu) MaxPooling2D(2, 2)	99.41%	97.13%	0.83
Попробуем добавить слои				
1	Conv2D(128, (3, 3), relu) MaxPooling2D(2, 2) Conv2D(128, (3, 3), relu) MaxPooling2D(2, 2)	99.87%	98.47%	0.97
2	Conv2D(128, (3, 3), relu) MaxPooling2D(2, 2) Conv2D(128, (3, 3), relu) MaxPooling2D(2, 2) Conv2D(256, (3, 3), relu) MaxPooling2D(2, 2)	99.71%	98.58%	0.88
Двух слоев достаточно, далее точность начинает падать. Попробуем изменять количество фильтров.				
3	Conv2D(256, (3, 3), relu) MaxPooling2D(2, 2) Conv2D(128, (3, 3), relu) MaxPooling2D(2, 2) Conv2D(64, (3, 3), relu) MaxPooling2D(2, 2)	99.59%	98.02%	1.49
Попробуем обратный подход. Ожидаются более хорошие результаты.				
4	Conv2D(64, (3, 3), relu) MaxPooling2D(2, 2) Conv2D(128, (3, 3), relu) MaxPooling2D(2, 2)	99.95%	98.93%	0.85

	Conv2D(256, (3, 3), relu) MaxPooling2D(2, 2)			
Стало лучше. Попробуем увеличить и уменьшить количество фильтров.				
5	Conv2D(32, (3, 3), relu) MaxPooling2D(2, 2) Conv2D(64, (3, 3), relu) MaxPooling2D(2, 2) Conv2D(128, (3, 3), relu) MaxPooling2D(2, 2)	99.47%	97.59%	0.64
6	Conv2D(128, (3, 3), relu) MaxPooling2D(2, 2) Conv2D(256, (3, 3), relu) MaxPooling2D(2, 2) Conv2D(512, (3, 3), relu) MaxPooling2D(2, 2)	98.76%	98.38%	1.41
Не стоит добавлять больше фильтров. Сеть становится слишком сложной для данной задачи. Попробуем увеличить размер ядра в первом сверточном слое, так как тут размер карты признаков достаточно большой.				
7	Conv2D(64, (5, 5), relu) MaxPooling2D(2, 2) Conv2D(128, (3, 3), relu) MaxPooling2D(2, 2) Conv2D(256, (3, 3), relu) MaxPooling2D(2, 2)	99.67%	98.55%	0.82
Не помогло. Попробуем использовать tanh как функцию активации на втором и третьем слоях. На первом сверточном слое оставим relu, чтобы бороться с исчезающим градиентом.				
8	Conv2D(64, (5, 5), relu) MaxPooling2D(2, 2) Conv2D(128, (3, 3), tanh) MaxPooling2D(2, 2) Conv2D(256, (3, 3), tanh) MaxPooling2D(2, 2)	**99.99%**	**99.53%**	0.71

Запуск лучшего на тестовой выборке

Best	Accuracy
Network #8	94.63%

Посмотрим матрицу ошибок, чтобы увидеть, если есть какие-то странные проблемы, и чтобы наглядно понять, как классифицирует наш классификатор.



Результат, очевидно, хуже по сравнению с валидационной выборкой, но тем не менее, хороший. Причин может быть много. Разница между тестовыми и тренировочными данными может быть большой. Возможно стоит добавлять регуляризацию, Dropout, а также искусственно изменять и добавлять данные в тренировочную выборку (augmentation).