

# Отчет. Лабораторная работа №2.

## ПОСТАНОВКА ЗАДАЧИ

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

Таковыми параметрами являются:

- Количество слоев
- Количество эпох
- Количество нейронов в скрытых слоях
- Функция активаций в скрытых слоях
- Функция инициализации весов в скрытых слоях
- Функция оптимизации

## 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  $D = \{(x_i, y_i), x_i \in X, y_i \in Y, i = [1, \dots, N]\}$ . 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  $D = \{(x_i, y_i), x_i \in X, y_i \in Y, i = [1, \dots, N]\}$  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  $p(y|x)$  при данном сэмпле  $D = \{(x_i, y_i), x_i \in X, y_i \in Y, i = [1, \dots, N]\}$ .

# 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	Images	Class Id	Images	Class Id	Images
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



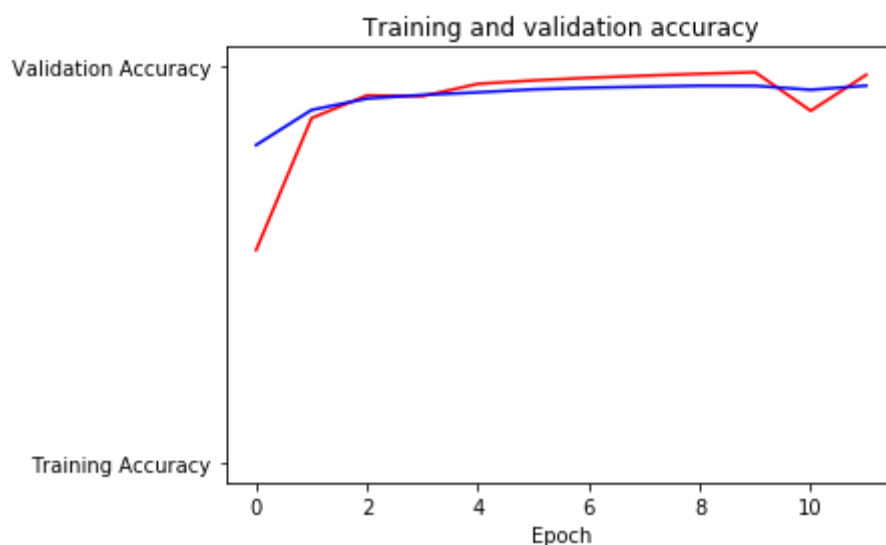
# МЕТРИКА КАЧЕСТВА РЕШЕНИЯ ЗАДАЧИ

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 до величины 40x40
2. Изображения раскладываются из матрицы в вектор

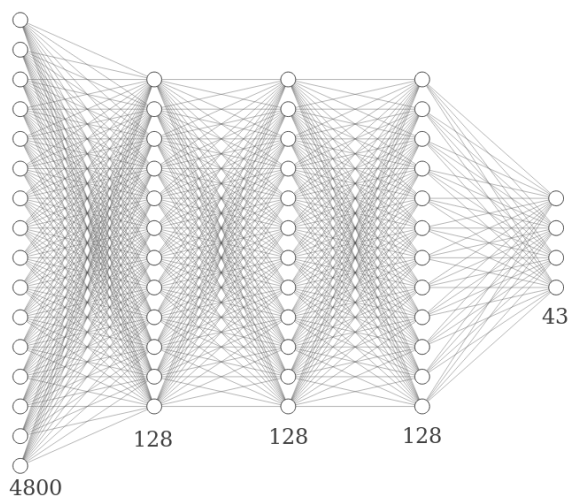
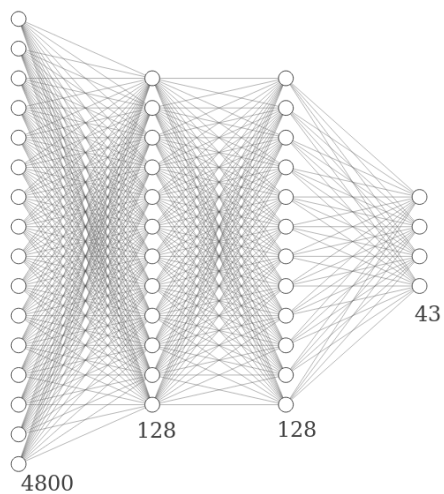
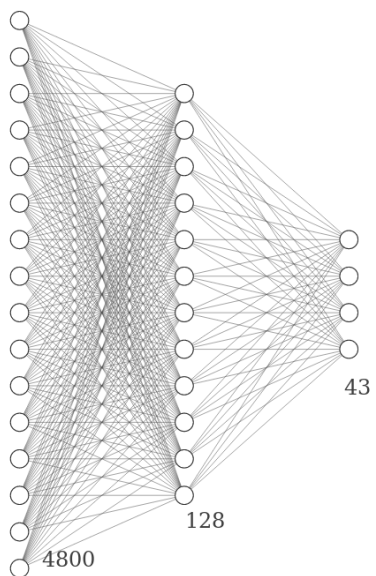


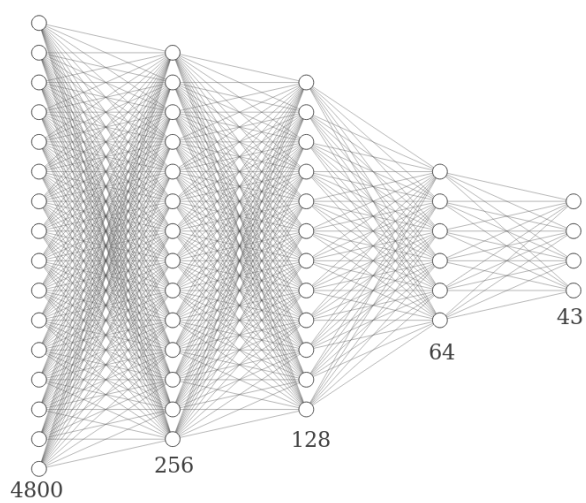
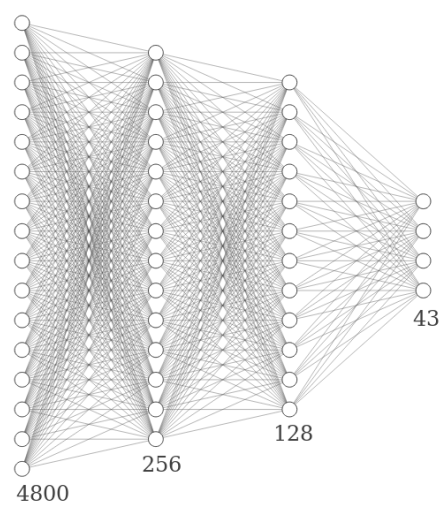
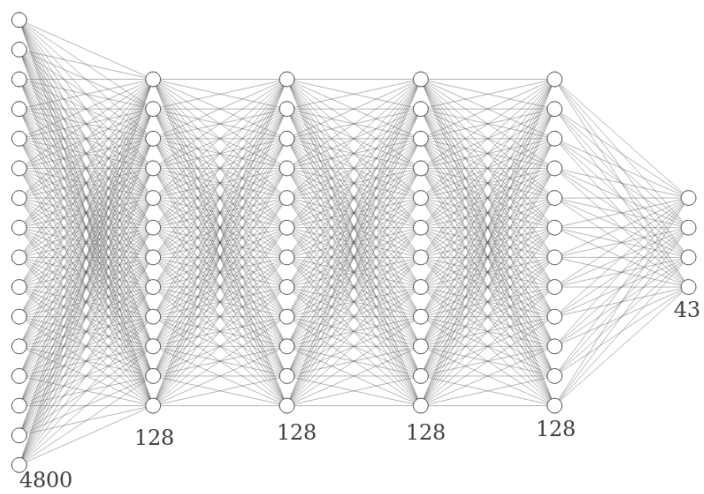
# РАЗРАБОТАННЫЕ ПРОГРАММЫ/СКРИПТЫ

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 ознакомлении с данными (показывает изображения, статистику и прочее). Выводы дополняются комментариями для более простого понимания кода.

# ТЕСТОВЫЕ КОНФИГУРАЦИИ СЕТЕЙ





# РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

В процессе экспериментов запускалось обучение нейронной сети с различным набором параметров. (В таблице присутствуют только скрытые слои)

Таблица с результатами

№	Ф-ия оптимизации	Скорость обучения	Количество эпох	Ф-ия инициализации	Параметры	Dense Layer 1	Dense Layer 2	Тренировочная точность	Валидационная точность	Время (мин)
0	RMSprop	0.001	20	glorot_uniform	Neuron number	128	-	90.31 %	89.15 %	4.83
					Activation function	relu	-			
Попробуем добавлять слои										
1	RMSprop	0.001	20	glorot_uniform	NN	128	128	90.14 %	88.55 %	4.83
					AF	relu	relu			
Результат не улучшился. Но ошибка на валидационной выборке стала нестабильной. Попробуем уменьшить параметр скорости обучения.										
2	RMSprop	0.0001	20	glorot_uniform	NN	128	128	91.58 %	90.02 %	4.74
					AF	relu	relu			
Все стало лучше. Попробуем добавить еще один слой.										
3	RMSprop	0.0001	20	glorot_uniform	NN	128	128(+1)	90.26 %	90.21 %	4.80
					AF	relu	relu(+1)			
Стало хуже. Проверим есть у этого динамика, добавим еще один слой.										
4	RMSprop	0.0001	20	glorot_uniform	NN	128	128(+2)	87.39 %	87.53 %	4.79
					AF	relu	relu(+2)			
Динамика сохранилась, добавление слоев не помогает. Попробуем изменить количество нейронов в сети с двумя и тремя слоями.										
5	RMSprop	0.0001	20	glorot_uniform	NN	256	128	92.07 %	89.95 %	5.10
					AF	relu	relu			
6	RMSprop	0.0001	20	glorot_uniform	NN	256	128	89.95	88.84	5.12

	op			uniform	AF	relu	relu	%	%	
Точность увеличилась, как и время обучения. Два слоя лучше трех. Попробуем исследовать влияние алгоритма оптимизации на такой сетке. Возьмем Adam и понизим для него скорость обучения, так как он обычно быстрее сходится, а для SGD наоборот, повысим.										
7	Adam	0.00005	20	glorot_uniform	NN	256	128	98.94 %	94.63 %	5.08
					AF	relu	relu			
8	SGD	0.001	20	glorot_uniform	NN	256	128	32.83 %	33.28 %	5.03
					AF	relu	relu			
Adam дает лучший результат. Продолжим его использовать. Исследуем влияние функции активации.										
9	Adam	0.00005	20	glorot_uniform	NN	256	128	76.72 %	75.43 %	5.07
					AF	sigmoid	sigmoid			
10	Adam	0.00005	20	glorot_uniform	NN	256	128	98.60 %	95.63 %	5.04
					AF	tanh	tanh			
Гиперболический тангенс дает лучшие результаты. Продолжим его использовать. Исследуем влияние функции инициализации весов.										
11	Adam	0.00005	20	glorot_normal	NN	256	128	98.65 %	95.65 %	5.07
					AF	tanh	tanh			
12	Adam	0.00005	20	he_normal	NN	256	128	98.69 %	95.58 %	5.06
					AF	tanh	tanh			
13	Adam	0.00005	20	he_uniform	NN	256	128	98.63 %	95.73 %	5.05
					AF	tanh	tanh			
Особой разницы в точности нет. Попробуем увеличить количество эпох.										
14	Adam	0.00005	50	he_uniform	NN	256	128	99.96 %	96.58 %	10.90
					AF	tanh	tanh			

## Запуск лучшего на тестовой выборке

Best	Accuracy
Network #14	86.84%

Сеть не очень хорошо обобщается на тестовую выборку. Методы регуляризации могут помочь, но сначала необходимо попробовать применить сверточные нейронные сети.