КОМИТЕТ ПО ОБРАЗОВАНИЮ

АДМИНИСТРАЦИИ ГОРОДСКОГО ОКРУГА

«ГОРОД КАЛИНИНГРАД»

Открытая XXV ученическая научно-практическая конференция «Поиск и творчество»

Секция: информатика

**Исследовательская работа**

Нейронные сети и машинное обучение

Работу выполнил:

Гончаров Сергей Максимович

Ученик 9 класса

МАОУ гимназии №40 им. Ю.А. Гагарина

Руководитель:

Семеновых Татьяна Игоревна

учитель информатики и ИКТ

МАОУ гимназия №40 им. Ю.А. Гагарина

Калининград 2017

Оглавление

[Оглавление 2](#_Toc499980705)

[Введение 4](#_Toc499980706)

[1. Теоретическая часть 6](#_Toc499980707)

[1.1. Общее определение машинного обучения 6](#_Toc499980708)

[1.2. Общее определение нейросетей 7](#_Toc499980709)

[1.3. Способы МО и классические задачи, решаемые с помощью МО 7](#_Toc499980710)

[1.4. Применение нейросетей и используемые для этого архитектуры 9](#_Toc499980711)

[1.4.1. Распознавание образов и классификация 9](#_Toc499980712)

[1.4.2. Принятие решений и управление 9](#_Toc499980713)

[1.4.3. Кластеризация 9](#_Toc499980714)

[1.4.4. Прогнозирование 9](#_Toc499980715)

[1.4.5. Аппроксимация 9](#_Toc499980716)

[1.4.6. Сжатие данных и Ассоциативная память 10](#_Toc499980717)

[1.4.7. Анализ данных 10](#_Toc499980718)

[1.4.8. Оптимизация 10](#_Toc499980719)

[1.5. Схема работы нейросети 10](#_Toc499980720)

[1.6. Все виды нейросетей 11](#_Toc499980721)

[1.7. Примеры приложений 11](#_Toc499980722)

[1.7.1. Предсказание финансовых временных рядов 11](#_Toc499980723)

[1.7.2. Психодиагностика 12](#_Toc499980724)

[1.7.3. Хемоинформатика 12](#_Toc499980725)

[1.7.4. Нейроуправление 13](#_Toc499980726)

[1.7.5. Экономика 13](#_Toc499980727)

[2. Практическая часть 14](#_Toc499980728)

[2.1. Описание структуры работы программы 14](#_Toc499980729)

[2.2. Входной слой (Input Layer) 14](#_Toc499980730)

[2.3. Скрытый слой (Hidden Layer) 14](#_Toc499980731)

[2.4. Выходной слой(Output Layer) 15](#_Toc499980732)

[2.5. Обучение нейронной сети (Метод Обратного Распространения Ошибки) 16](#_Toc499980733)

[2.5.1. Ошибка (Error) 16](#_Toc499980734)

[2.5.2. Дельта выходного слоя(Delt Output) и скрытого слоя(Delt Hidden) 16](#_Toc499980735)

[2.5.3. Градиент синапсов, соединяющих скрытый и выходной слои и градиент синапсов, соединяющих входной и скрытый слои 17](#_Toc499980736)

[Список используемой литературы 21](#_Toc499980737)

[ПРИЛОЖЕНИЕ A 22](#_Toc499980738)

# Введение

*” Машинное обучение принимает много разных форм и скрывается под разными именами: распознавание паттернов, статистическое моделирование, добыча данных, выявление знаний, предсказательная аналитика, наука о данных, адаптивные и самоорганизующиеся системы и так далее “.*

Педро Домингос

«Верховный алгоритм.

Как машинное обучение изменит наш мир»

Педро Домингос «Верховный алгоритм. Как машинное обучение изменит наш мир» я сейчас творится что-то невообразимое. Когда мы входим в эту область и начинаем считать деньги, которые она может дать, то оказывается, что нам необходимы знания, материалы и опыт из различных других областей. Мы с вами, как правило, разработчики. Кто-то из нас может быть с неким математическим бэкграундом, кто-то пишет хороший код, кто-то, например, профессионал в математической статистике. И машинное обучение как раз требует знания и навыков в этих сферах. В 2016 году машинное обучение заняло высокую позицию в рейтинге способа заработка денег, саморазвития и даже просто баловства. В этом году набрали популярность многие приложения, связанные например, с преобразованием изображения в стилизованное изображение с использованием одним из способов машинного обучения – нейросети(Prisma), распознавание лица в камере смартфона, голосовой поиск(Google), поиск мелодии по отрывку(Shazam). Предположительно, машинное обучение будет развиваться с каждым годом все интенсивней и интенсивней. Так что можно считать, что за машинным обучением стоит будущее.

Цель работы – Изучить работу нейросетей на примере создания собственной нейронной сети для распознавания цифр с изображения.

Задачи:

1. Изучение основных понятий о машинном обучении и нейронных сетях
2. Изучение видов и алгоритмов машинного обучения
3. Реализация нейросети в среде разработки Visual Studio на языке программирования C#
4. Тестирование на приготовленных данных

Сокращения:

МО - машинное обучение

# Теоретическая часть

# Общее определение машинного обучения

Машинное обучение (англ. Machine Learning) — обширный подраздел искусственного интеллекта, математическая дисциплина, использующая разделы математической статистики, численных методов оптимизации, теории вероятностей, дискретного анализа, и извлекающая знания из данных.

Различают два типа обучения:

1. Обучение по прецедентам, или индуктивное обучение, основано на выявлении закономерностей в эмпирических данных.
2. Дедуктивное обучение предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний.

Дедуктивное обучение принято относить к области экспертных систем, поэтому термины машинное обучение и обучение по прецедентам можно считать синонимами.

Многие методы индуктивного обучения разрабатывались как альтернатива классическим статистическим подходам. Многие методы тесно связаны с извлечением информации (Information Extraction), интеллектуальным анализом данных (Data mining).

Имеется множество объектов (ситуаций) и множество возможных ответов (откликов, реакций). Существует некоторая зависимость между ответами и объектами, но она неизвестна. Известна только конечная совокупность прецедентов — пар «объект, ответ», называемая обучающей выборкой. На основе этих данных требуется восстановить зависимость, то есть построить алгоритм, способный для любого объекта выдать достаточно точный ответ. Для измерения точности ответов определённым образом вводится функционал качества.

Данная постановка является обобщением классических задач аппроксимации функций. В классических задачах аппроксимации объектами являются действительные числа или векторы. В реальных прикладных задачах входные данные об объектах могут быть неполными, неточными, нечисловыми, разнородными. Эти особенности приводят к большому разнообразию методов машинного обучения.

# Общее определение нейросетей

Иску́сственная нейро́нная се́ть (ИНС) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети У. Маккалока и У. Питтса. После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

ИНС представляют собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты (особенно в сравнении с процессорами, используемыми в персональных компьютерах). Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И, тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие по отдельности простые процессоры вместе способны выполнять довольно сложные задачи. С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п.

# Способы МО и классические задачи, решаемые с помощью МО

Способы:

Обучение с учителем — для каждого прецедента задаётся пара «ситуация, требуемое решение»:

1. Метод коррекции ошибки
2. Метод обратного распространения ошибки

Обучение без учителя — для каждого прецедента задаётся только «ситуация», требуется сгруппировать объекты в кластеры, используя данные о попарном сходстве объектов, и/или понизить размерность данных:

1. Альфа-система подкрепления
2. Гамма-система подкрепления
3. Метод ближайших соседей

Обучение с подкреплением — для каждого прецедента имеется пара «ситуация, принятое решение»:

1. Генетический алгоритм.

Активное обучение — отличается тем, что обучаемый алгоритм имеет возможность самостоятельно назначать следующую исследуемую ситуацию, на которой станет известен верный ответ:

Обучение с частичным привлечением учителя (Semi-supervised learning) — для части прецедентов задается пара «ситуация, требуемое решение», а для части — только «ситуация»

Трансдуктивное обучение (англ. transduction (machine learning)) — обучение с частичным привлечением учителя, когда прогноз предполагается делать только для прецедентов из тестовой выборки

Многозадачное обучение (multi-task learning) — одновременное обучение группе взаимосвязанных задач, для каждой из которых задаются свои пары «ситуация, требуемое решение»

Многовариантное обучение (multiple-instance learning) — обучение, когда прецеденты могут быть объединены в группы, в каждой из которых для всех прецедентов имеется «ситуация», но только для одного из них (причем, неизвестно какого) имеется пара «ситуация, требуемое решение»

*Классические задачи:*

* Классификация как правило, выполняется с помощью обучения с учителем на этапе собственно обучения.
* Кластеризация как правило, выполняется с помощью обучения без учителя
* Регрессия как правило, выполняется с помощью обучения с учителем на этапе тестирования, является частным случаем задач прогнозирования.
* Понижение размерности данных и их визуализация выполняется с помощью обучения без учителя
* Восстановление плотности распределения вероятности по набору данных
* Одноклассовая классификация и выявление новизны
* Построение ранговых зависимостей

# Применение нейросетей и используемые для этого архитектуры

# Распознавание образов и классификация

#### Используемые архитектуры нейросетей

* Обучение с учителем:
  + Перцептрон
* Обучение без учителя:
  + Сети адаптивного резонанса
* Смешанное обучение:
  + Сеть радиально-базисных функций

# Принятие решений и управление

#### Используемые архитектуры нейросетей

* Обучение с учителем:
  + Перцептрон
  + Смешанное обучение:
  + Сеть радиально-базисных функций

# Кластеризация

#### Используемые архитектуры нейросетей

* Обучение без учителя:
  + Перцептрон
  + Самоорганизующаяся карта Кохонена
  + Нейронная сеть Кохонена
  + Сети адаптивного резонанса

# Прогнозирование

#### Используемые архитектуры нейросетей

* Обучение с учителем:
  + Перцептрон
* Смешанное обучение:
  + Сеть радиально-базисных функций

# Аппроксимация

#### Используемые архитектуры нейросетей

* Обучение с учителем:
  + Перцептрон
* Смешанное обучение:
  + Сеть радиально-базисных функций

# Сжатие данных и Ассоциативная память

#### Используемые архитектуры нейросетей

* Обучение с учителем:
  + Перцептрон
* Обучение без учителя:
  + Нейронная сеть Хопфилда

# Анализ данных

#### Используемые архитектуры нейросетей

* Обучение с учителем:
  + Перцептрон
* Обучение без учителя:
  + Перцептрон
  + Самоорганизующаяся карта Кохонена
  + Нейронная сеть Кохонена

# Оптимизация

#### Используемые архитектуры нейросетей

* Обучение без учителя:
  + Самоорганизующаяся карта Кохонена
  + Нейронная сеть Кохонена

# Схема работы нейросети

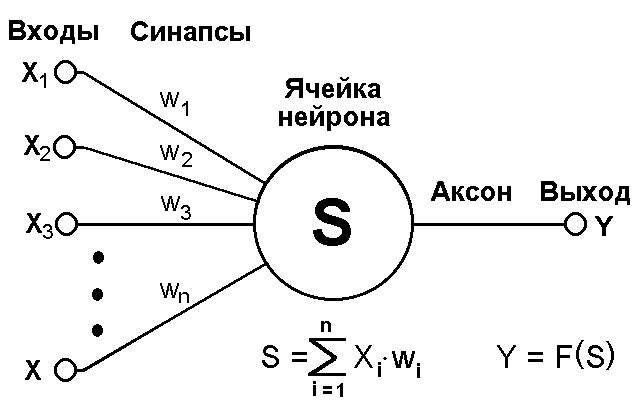


Рисунок 1. – Схема работы нейросети.

# Все виды нейросетей

* Перцептрон Розенблатта;
* Сплайн-модель Хакимова;
* Многослойный перцептрон Розенблатта;
* Многослойный перцептрон Румельхарта;
* Сеть Джордана;
* Сеть Элмана;
* Сеть Хэмминга;
* Сеть Ворда;
* Сеть Хопфилда;
* Сеть Кохонена;
* Нейронный газ;
* Когнитрон;
* Неокогнитрон;
* Хаотическая нейронная сеть;
* Осцилляторная нейронная сеть;
* Сеть встречного распространения;
* Сеть радиально-базисных функций (RBF-сеть);
* Сеть обобщенной регрессии;
* Сеть Д.Смирнова;
* Вероятностная сеть;
* Вероятностная нейронная сеть Решетова;
* Сиамская нейронная сеть;
* Сети адаптивного резонанса;
* Свёрточная нейронная сеть (англ. convolutional neural network);
* Нечеткий многослойный перцептрон;
* Импульсная нейронная сеть.

# Примеры приложений

# Предсказание финансовых временных рядов

Входные данные — курс акций за год. Задача — определить завтрашний курс. Проводится следующее преобразование — выстраивается в ряд курс за сегодня, вчера, за позавчера. Следующий ряд — смещается по дате на один день и так далее. На полученном наборе обучается сеть с 3 входами и одним выходом — то есть выход: курс на дату, входы: курс на дату минус 1 день, минус 2 дня, минус 3 дня. Обученной сети подаем на вход курс за сегодня, вчера, позавчера и получаем ответ на завтра. Нетрудно заметить, что в этом случае сеть просто выведет зависимость одного параметра от трёх предыдущих. Если желательно учитывать ещё какой-то параметр (например, общий индекс по отрасли), то его надо добавить как вход (и включить в примеры), переобучить сеть и получить новые результаты. Для наиболее точного обучения стоит использовать метод ОРО, как наиболее предсказуемый и несложный в реализации.

# Психодиагностика

Серия работ М. Г. Доррера с соавторами посвящена исследованию вопроса о возможности развития психологической интуиции у нейросетевых экспертных систем. Полученные результаты дают подход к раскрытию механизма интуиции нейронных сетей, проявляющейся при решении ими психодиагностических задач. Создан нестандартный для компьютерных методик интуитивный подход к психодиагностике, заключающийся в исключении построения описанной реальности. Он позволяет сократить и упростить работу над психодиагностическими методиками.

# Хемоинформатика

Нейронные сети широко используются в химических и биохимических исследованиях. В настоящее время нейронные сети являются одним из самых распространенных методов хемоинформатики для поиска количественных соотношений структура-свойство, благодаря чему они активно используются как для прогнозирования физико-химических свойств и биологической активности химических соединений, так и для направленного дизайна химических соединений и материалов с заранее заданными свойствами, в том числе при разработке новых лекарственных препаратов.

# Нейроуправление

Нейронные сети успешно применяются для синтеза [систем управления](https://ru.wikipedia.org/wiki/Регулятор_(теория_управления)) динамическими объектами. Нейросети обладают рядом уникальных свойств, которые делают их мощным инструментом для создания систем управления: способностью к обучению на примерах и обобщению данных, способностью адаптироваться к изменению свойств объекта управления и внешней среды, пригодностью для синтеза нелинейных регуляторов, высокой устойчивостью к повреждениям своих элементов в силу изначально заложенного в нейросетевую архитектуру параллелизма.

# Экономика

Алгоритмы искусственных нейронных сетей нашли широкое применение в экономике. С помощью нейронных сетей решается задача разработки алгоритмов нахождения аналитического описания закономерностей функционирования экономических объектов (предприятие, отрасль, регион). Эти алгоритмы применяются к прогнозированию некоторых «выходных» показателей объектов. Применение нейросетевых методов позволяет решить некоторые проблемы экономико-статистического моделирования, повысить адекватность математических моделей, приблизить их к экономической реальности. Поскольку экономические, финансовые и социальные системы очень сложны и являются результатом человеческих действий и противодействий, создание полной математической модели с учётом всех возможных действий и противодействий является очень сложной (если разрешимой) задачей. В системах подобной сложности естественным и наиболее эффективным является использование моделей, которые напрямую имитируют поведение общества и экономики. Именно это способна предложить методология нейронных сетей.

# Практическая часть

# Описание структуры работы программы

Нейросеть реализована в среде разработки Visual Studio на языке программирования C#. Для нейросети я использовал архитектуру “Перцептрон” с использованием метода обратного распространения ошибки. Нейронная сеть состоит из трех слоев: входной, скрытый и выходной.

# Входной слой (Input Layer)

Входные данные – черно-белая картинка формата BMP. Переработка изображения происходит с помощью встроенного класса Bitmap, который разделяет картинку на пиксели. Программа по очереди берет пиксель и определяет его цветовую насыщенность. При показателе насыщенности более 250 на выход входного нейрона подается 0, иначе 1.

int Input(int x, int y)

{

Bitmap im = pictureBox1.Image as Bitmap;

int n = (im.GetPixel(x, y).R);

if (n >= 250) n = 0;

else n = 1;

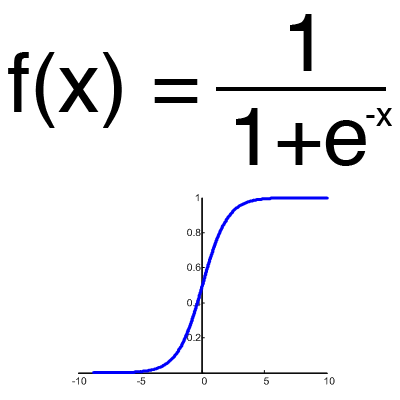
input[x, y] = n;

return input[x, y];

}

Нейроны входного и скрытого слоев соединены синапсами, каждый из которых имеет свой вес. Веса входного слоя хранятся в CSV файле.

# Скрытый слой (Hidden Layer)



На вход скрытого слоя передается сумма произведений значений входного нейрона и весов синапсов, соединенных с этим нейроном. В качестве функции активации я выбрал сигмодальную функцию.

На выходе скрытого слоя подается значение функции активации, которое может принимать значение от 0 до 1.

double sigmo(double x)

{

double y;

y = 1 / (1 +Math.Pow(e,-x));

return y;

}

void neuronSum()

{

double[] NeuronInput = new double[HiddenNeuronsCount];

for (int i = 0; i <= HiddenNeuronsCount-1; i++)

{

for (int j = 0; j <= pictureHeight - 1; j++)

{

for (int m = 0; m <= pictureWidth - 1; m++)

{

NeuronInput[i] = NeuronInput[i]+publweights[i, j, m] \* Input(m,j);

}

}

NeuronOutput[i]=sigmo(NeuronInput[i]);

}

}

# Выходной слой(Output Layer)

На вход выходного нейрона подается сумма произведений значений скрытого слоя и весов синапсов, соединенных с этим нейроном. В качестве функции активации я выбрал, опять же, сигмоидальную функцию. Всего я использовал 10 выходных нейронов. Каждый выходной нейрон может принимать значение от 0 до 1. Индекс нейрона, принимающего наибольшее значение, и будет ответом нейросети.

void outInput()

{

double[] OutSum = new double[outputNeuronsCount];

for (int j = 0; j < outputNeuronsCount; j++)

{

OutSum[0] = 0;

for (int m = 0; m < HiddenNeuronsCount; m++)

{

OutSum[j] = OutSum[j] + NeuronOutput[m] \* publicSecondWeights[j,m];

}

outOutput[j] = sigmo(OutSum[j]);

}

}

# Обучение нейронной сети (Метод Обратного Распространения Ошибки)

# Ошибка (Error)

Ошибка выходного нейрона – это разница между идеальным значением нейрона и фактическим. Для обучения принял ошибку, равную 20%.

void Error()

{

for (int i=0;i<outputNeuronsCount;i++)

{

error[i] = ideal[i] - outOutput[i];

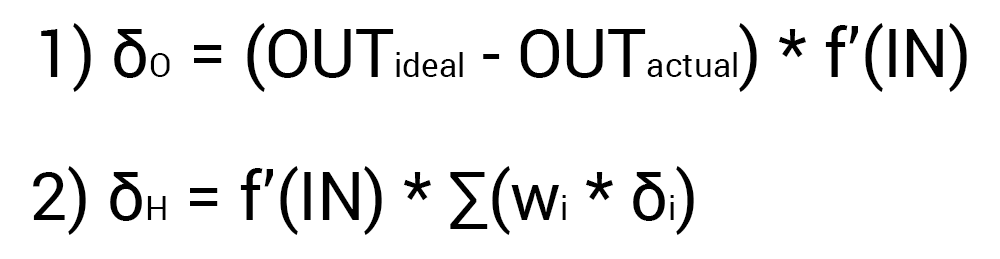
}

}

Для реализации обратного распространения ошибки, я использовал способ градиентного спуска.

# Дельта выходного слоя(Delt Output) и скрытого слоя(Delt Hidden)

После вычисления ошибки необходимо высчитать значение дельты. Дельта используется в формуле подсчета градиентного спуска для упрощения вычислений.



Так как у выходного нейрона нет исходящих синапсов, то будем пользоваться первой формулой (δ output), следственно для скрытых нейронов мы уже будем брать вторую формулу (δ hidden). Считаем разницу между желаемым и полученным результатом и умножаем на производную функции активации от входного значения данного нейрона.

void deltOutput()

{

for(int i = 0; i < outputNeuronsCount; i++)

{

deltOut[i] = error[i]\*(1-outOutput[i])\*outOutput[i];

}

}

void DeltHidden()

{

double sum=0;

for (int i = 0; i < HiddenNeuronsCount; i++)

{

for(int j=0;j<outputNeuronsCount;j++)

{

sum = sum + publicSecondWeights[j, i] \* deltOut[j];

}

deltHidden[i] =((1 - NeuronOutput[i]) \* NeuronOutput[i]) \* sum;

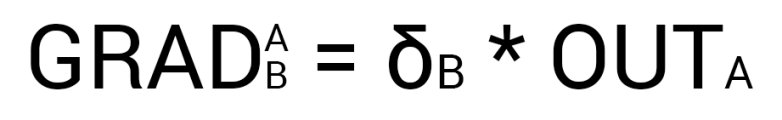
sum = 0;

}

}

# Градиент синапсов, соединяющих скрытый и выходной слои (Gradient Output) и градиент синапсов, соединяющих входной и скрытый слои (Gradient Hidden)

Теперь нам нужно найти градиент для каждого исходящего синапса. Здесь точка A это точка в начале синапса, а точка B на конце синапса.



В моем примере у меня сразу же подсчитывается дельта изменения весов(Delt Weights).



E-постоянная(скорость обучения)

α-постоянная(момент)

void gradientOut()

{

double[,] gradientHO = new double[outputNeuronsCount,HiddenNeuronsCount];

for(int i = 0; i < outputNeuronsCount; i++)

{

for(int j = 0; j < HiddenNeuronsCount; j++)

{

gradientHO[i,j]=NeuronOutput[j]\*deltOut[i];

deltWHO[i, j] = learnSpeed \* gradientHO[i, j] + (moment \* previousDeltWHO[i, j]);

previousDeltWHO[i, j] = deltWHO[i, j];

publicSecondWeights[i, j] = publicSecondWeights[i, j]+deltWHO[i, j];

}

}

}

void gradientHidden()

{

double[,,] gradientIH = new double[HiddenNeuronsCount,pictureHeight,pictureWidth];

for (int i = 0; i < HiddenNeuronsCount; i++)

{

for (int j = 0; j < pictureHeight; j++)

{

for (int m=0;m<pictureWidth;m++)

{

gradientIH[i, j,m] = Input(m,j) \* deltHidden[i];

deltWIH[i, j,m] = learnSpeed \* gradientIH[i, j,m] + (moment \* previousDeltWIH[i, j,m]);

previousDeltWIH[i, j,m] = deltWIH[i, j,m];

publweights [i, j,m] = publweights[i, j, m] + deltWIH[i, j,m];

}

}

}

}

После подсчета дельты изменения весов, веса изменяются на эту дельту.

После изменения весов, происходит повторное вычисление скрытого и выходного слоя и подсчет ошибки. Эта процедура выполняется, пока ошибка на каждом выходном нейроне не будет меньше минимально допустимой.

Для автоматического обучения я использовал по 3 картинки каждой цифры и провел обучение на 500 эпох.

**Заключение**

В процессе исследования была изучена работа простейшей архитектуры нейросетей - перцептрона на примере создания собственной нейронной сети для распознавания цифр с рукописного изображения. Результаты, в целом, оказались положительными, но процент распознавания картинок оказался ниже ожидаемого.

После обучения нейросети, тестируя ее, я столкнулся с некоторыми проблемами.

Во-первых, долгое обучение. Все функции, написанные мной, выполняются быстро, но долго происходит ввод-вывод из файла.

Во-вторых, относительно небольшая точность распознавания - 50%.

В-третьих, программа работает только для 64-битной Windows, а также требует ручной настройки некоторых параметров системы.

# Список используемой литературы

* 1. Домингос П. Верховный алгоритм. Как машинное обучение изменит наш мир. ЛитРес, ООО «Манн, Иванов и Фербер», 2016.
  2. 1С-Битрикс. Машинное обучение, предсказание будущего и анализ причин успеха в электронной коммерции [Электронный ресурс]: <https://habrahabr.ru/company/bitrix/blog/268971/>. (дата обращения: 03.10.2017).
  3. Арнис. Нейронные сети для начинающих. Часть 1. [Электронный ресурс]: <https://habrahabr.ru/post/312450/> (дата обращения: 10.09.2017).

# ПРИЛОЖЕНИЕ A

