МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра прикладной математики**

**КУРСОВАЯ РАБОТА**

**РАСПОЗНАВАНИЕ ЦИФР НА ОСНОВЕ МЕТОДА НЕЙРОННОЙ СЕТИ**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д.С.Корниенко

(подпись)

Направление подготовки 01.03.02 Прикладная математика и информатика

Направленность (профиль) Системный анализ, исследование операций и управление (Математическое и информационное обеспечение экономической деятельности)

Научный руководитель

канд. техн. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.Ю. Пелипенко

(подпись)

Нормоконтролер

канд. физ.-мат. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.В. Калайдина

(подпись)

Краснодар

2018

**РЕФЕРАТ**

Курсовая работа содержит 37 страниц, 9 изображений. Ключевые слова: нейронная сеть, обучение, связь, вес связи, нейрон, точность результатов.

Цель курсовой работы – изучение искусственных нейронных сетей с последующей программной реализацией и анализом отдельных параметров.

В ходе курсовой работы была изучена формальная модель искусственной нейронной сети и методы ее обучения.

В работе рассмотрено решение задачи распознавания образов на основе метода нейронных сетей. В процессе решения задачи был получен программный модуль, позволяющий автоматизировать процесс распознавания цифр. Программа имеет удобный и простой интерфейс для взаимодействия пользователем. В приложении к работе представлен листинг программы, реализованной на языке программирования C#.

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc501975828)

[1 Общие сведения о нейронных сетях 6](#_Toc501975829)

[1.1 Искусственные нейронные сети 6](#_Toc501975830)

[1.2 Анализ существующих нейросетевых моделей распознавания образов 9](#_Toc501975832)

[1.3 Модель естественных и искусственных нейронных сетей 12](#_Toc501975832)

[1.4 Обучение нейронных сетей 11](#_Toc501975833)

[1.5 Нейронная сеть Перцептрон 16](#_Toc501975834)

[2 Постановка задачи и проектирование нейронные сети 18](#_Toc501975835)

[2.1 Постановка задачи 18](#_Toc501975836)

[3 Описание реализации нейронной сети и алгоритма распознавания 19](#_Toc501975837)

[3.1 Описание основных методов 19](#_Toc501975838)

[ЗАКЛЮЧЕНИЕ 26](#_Toc501975840)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc501975841)

[Приложение А Реализация класса, отвечающего за нейрон 28](#_Toc501975842)

[Приложение Б Реализация класса, отвечающего за нейронную сеть 31](#_Toc501975843)

[Приложение В Реализация класса, отвечающего за обработку изображения рукописного символа 35](#_Toc501975844)

ВВЕДЕНИЕ

Одними из важнейших областей исследований и разработок современной кибернетики является область компьютерного зрения, которая заключается в машинном распознавании образов. Ускоряющиеся темпы развития технологий информационного общества, развитие робототехники, концепций «умный дом» и «умный город» определяют особое место разработке систем искусственного интеллекта в современном научном знании. При решении прикладных задач распознавания образов в практике современного программирования используются методы сбора данных, кластерного и классификационного анализа, методы статистического вывода данных.

В повседневную жизнь, как и в корпоративную и промышленную среды, начинают внедряться технологии, постепенно стирающие грань между реальным и виртуальным пространством. Это требует нового качественного уровня повсеместно внедряемых технологий распознавания образов, чья область применения в последние годы выросла колоссально: считавшиеся сложнейшими ранее задачи распознавания сегодня круглосуточно решаются мобильными устройствами рядовых граждан.

Компьютеризированные пространства с выраженной топологией, такие как «умный дом» рядового пользователя, многопользовательская дополненная и многопользовательская виртуальная реальности различной степени погружения, усложняющийся искусственный интеллект в компьютерных играх различного назначения требуют новых идей и подходов, нового уровня точности и скорости распознавания образов.

Эффективность распознавания различных образов в процессе роботизации экономики и производства, роботизации спасательных мероприятий, роботизации вооружённых сил и частных армий, действующих в интересах государства и других корпоратократических объединений, становится вопросом как сохранения человеческих жизней так и улучшения ее качества. В курсовой работе рассмотрена задача распознавания рукописных символов. Нейронная сеть, распознающая символы, может быть применена в различных устройствах для людей с ограниченными возможностями, чтобы они могли писать слова и тем самым, выражать свои мысли не только вербально. А так же при идентификации личности по подчерку.

Целью курсовой работы является изучение метода нейронных сетей на примере решения задачи распознавания рукописных символов.

# **1 Общие сведения о нейронных сетях**

## **1.1 Искусственные нейронные сети**

Искусственная нейронная сеть (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 [[1]](#_СПИСОК_ИСПОЛЬЗОВАННЫХ_ИСТОЧНИКОВ). После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в задачах прогнозирования задачах прогнозирования, для распознавания образов различного происхождения, 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, регрессионных, прогнозных задач и задач аппроксимации. С математической точки зрения, обучение нейронных сетей − это многопараметрическая задача нелинейной оптимизации. С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники. С точки зрения развития вычислительной техники и программирования, нейронная сеть − способ решения проблемы эффективного параллелизма. А с точки зрения искусственного интеллекта, ИНС является основой философского течения коннективизма и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения − одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искажённых данных.

Активные работы над созданием нейронных сетей начались в 60-е годы, когда нейробиологами было установлено, что мозг − это миллиарды нейронов, соединенных между собой, а мозг человека способен выполнять сложнейшие задачи.

Нейронные сети имеют следующие свойства:

1) возможность параллельной обработки информации;

2) способность к классификации, обобщению, абстрагированию;

3) способность к самоорганизации;

4) надежность.

Рассмотрим задачи, решаемые с помощью нейронных вычислений.

Классификация образов. Задача состоит в указании принадлежности входного образа (например, речевого сигнала или рукописного символа), представленного вектором признаков, одному или нескольким предварительно определенным классам. Для решения задачи используется обучение с учителем − сети предъявляется обучающая выборка, состоящая из пар (входной образ, метка класса), и применяются определенные правила для того, чтобы научить сеть определять метку класса для входного образа, который на этапе функционирования может и не входить в обучающую выборку.

Кластеризация/категоризация. При решении задачи кластеризации, (классификация образов без учителя), отсутствует обучающая выборка с метками классов. Алгоритм кластеризации основан на подобии образов и размещает близкие образы в один кластер.

Аппроксимация функций. Необходима при решении многочисленных инженерных и научных задач моделирования.

Предсказание/прогноз. Задача состоит в предсказании значения в некоторый будущий момент времени.

Оптимизация. Задачей алгоритма оптимизации является отыскание такого решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию.

Память, адресуемая по содержанию. Содержимое памяти может быть вызвано даже по частичным данным или искаженному содержанию.

## **1.2 Анализ существующих нейросетевых моделей распознавания образов**

На распознавание сложного объекта в комплексных изменяющихся условиях, мозг тратит около 100-200 миллисекунд. С. Хайкин в своей книге «Нейронные сети. Полный курс» дает такое определение данного понятия: «Нейронная сеть − это громадный распределенный параллельный процессор, состоящий из элементарных единиц обработки информации, накапливающих экспериментальные знания и предоставляющих их для последующей обработки»[[2]](#_СПИСОК_ИСПОЛЬЗОВАННЫХ_ИСТОЧНИКОВ). В первую очередь нейронные сети характеризуют такие их свойства как массивное распараллеливание процесса обработки информации и способность к обобщению. Несмотря на это, на сегодняшний день, нейронные сети еще не способны давать сложные готовые решения, из-за чего их приходится интегрировать в многосоставные системы для решения конкретных, пусть и не тривиальных задач. Искусственные нейронные сети имеют ряд полезных свойств.

1. Нелинейность всей сети и отдельных нейронов, из которых она состоит;
2. Адаптивность. Нейронные сети могут изменять свою структуру, приспосабливаясь к изменениям в окружающей среде;
3. Устойчивость к отказам. При повреждении отдельных нейронов или связей в аппаратной реализации нейронной сети, работоспособность всей системы сохраняется;
4. Единый подход к анализу и проектированию, универсальность в решении задач, относящихся к различным предметным областям.

Нейронная сеть состоит из большого числа элементарных процессоров − нейронов, выполняющих простые вычисления, но связанных друг с другом и, таким образом, в совокупности способных решать сложные, плохо формализованные задачи. Нейроны группируются в слои таким образом, что на вход каждого нейрона в следующем слое поступают выходные значения всех нейронов предыдущего слоя (такие сети называют полносвязными). При этом каждой связи, соединяющей два нейрона, соответствует определенный вес, определяющий силу взаимодействия между ними. Связи между нейронами могут быть возбуждающими или тормозящими. При всех своих объективных достоинствах нейронные сети остаются чрезвычайно сложным механизмом обработки информации. Очень важно заранее серьезно подойти к выбору модели сети, чтобы хотя бы примерно определить, какие результаты мы можем ожидать от системы при решении конкретной поставленной задачи. «Стандартные» сети прямого распространения способны выполнять очень сложные задачи по распознаванию объектов. Однако этот вид архитектуры нейронных сетей имеет некоторые ограничения. В ситуациях, когда исследуемый объект появляется вместе с другими объектами, сети прямого распространения могут быть перегружены и не способны корректно осуществить распознавание. Сверточные нейронные сети обрабатывают исходное изображение не полностью, а отдельными «порциями», последовательно уменьшая его размер или выделяя характерные наиболее важные признаки, уходя на новый уровень абстракции. В этих сетях формируются так называемые карты признаков, которые стороннему наблюдателю кажутся размытыми, искаженными копиями исходного изображения, но для нейронной сети имеют совершенно иной смысл, содержат инварианты и характерные признаки.

Однако нельзя полагаться на излишне оптимистичные показатели распознавания, если проверять оценку качества функционирования нейронной сети по той же выборке m X, с помощью которой и был произведен процесс обучения. Ситуацию, когда нейронная сеть просто «запоминает» прецеденты, предоставленные ей для обучения, но не способна обобщить полученный опыт на новые данные, называется переобучением.

Ключевое значение имеет способность сети к обобщению на наборе данных, выходящих за рамки обучающей выборки, поэтому существует понятие тестовой (или контрольной) выборки, по которой оценивается качество построенной модели.

Для того, чтобы сравнить несколько моделей, обученных на обучающей выборке, по критерию качества распознавания, необходимо создать еще одну, так называемую, проверочную или вариационную выборку, позволяющую избежать смещения при сравнении и получения нереалистично оптимистичных результатов.

Таким образом, среди всех подходов к распознаванию образов нейронные сети выгодно отличает их свойство адаптивности и универсальность. Это мощный инструмент, обеспечивающий высокое качество распознавания образов.

## 

## **1.3 Модель естественных и искусственных нейронных сетей**

Модель искусственной нейронной сети является прототипом естественной. Естественная нейронная сеть состоит из связанных между собой нейронов − специализированных клеток, которые способны к передаче, обработке и хранению информации, необходимой для реализации физиологических функций организма. Биологический нейрон состоит из тела и входных и выходных каналов – дендритов и аксонов. Для хранения информации в нейронах имеются синапсы. Еще одна особенность биологических нейронных сетей − естественный параллелизм и коллективизм их работы. Обучение естественных нейронных систем происходит путем изменения силы межнейронных связей.

Чтобы отразить суть биологических нейронных систем, искусственный [нейрон](http://statsoft.ru/home/textbook/glossary/gloss_n.html#Neuron) обладает следующими свойствами:

а) он получает входные сигналы (исходные данные либо выходные сигналы других нейронов [нейронной сети](http://statsoft.ru/home/textbook/glossary/gloss_n.html#Neural Networks)) через несколько входных каналов. Каждый входной сигнал проходит через соединение, имеющее определенную интенсивность (или вес); этот вес соответствует синаптической активности биологического нейрона. С каждым нейроном связано определенное пороговое значение. Вычисляется взвешенная сумма входов, из нее вычитается пороговое значение и в результате получается величина активации нейрона.

б) сигнал активации преобразуется с помощью [функции активации](http://statsoft.ru/home/textbook/glossary/gloss_f.html#Activation Function) (или передаточной функции) и в результате получается выходной сигнал нейрона.

ИНС чрезвычайно разнообразны по своей архитектуре. Все они имеет простой общий базовый элемент − искусственный нейрон, который имитирует свойства своего биологического родителя. На вход искусственного нейрона (рисунок 1) подается набор входных сигналов x1, x2, ..., xn (входной вектор X), представляющий собой выходные сигналы других нейроподобных элементов или входные сигналы нейросети.

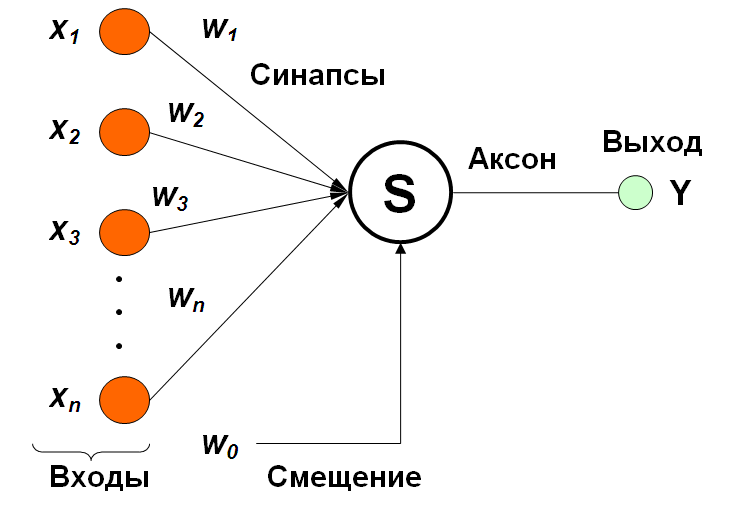


Рисунок 1 ‒ Модель нейрона

Каждый входной сигнал умножается на соответствующий вес связи w1, w2, ..., wn − аналог эффективности синапса. Вес этой связи является положительным для возбуждающей и отрицательным для тормозящей связи нейрона. Взвешенные весами связей, входные сигналы поступают на блок суммации (по своей сути являющийся аналогом тела клетки в биологическом нейроне), где вычисляется их алгебраическая сумма и определяется уровень возбуждения нейрона S:

(1)

где: θ − порог возбуждения нейрона.

Затем над уровнем возбуждения S производится необходимое нелинейное преобразование:

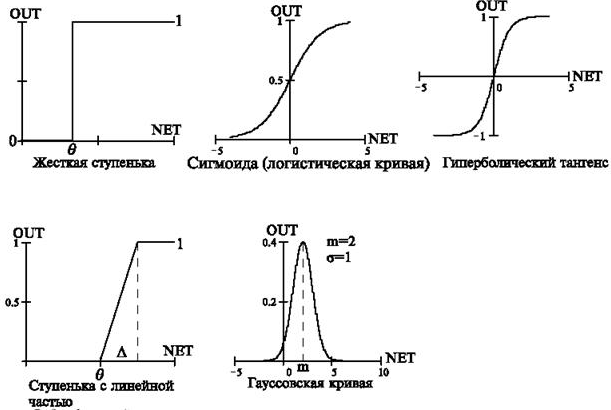
(2)

Эта функция называется активационной (рисунок 2). Наиболее часто на практике применяется сигмоидальная (логистическая) функция вида:

. (3)

Эта функция позволяет работать со слабыми и сильными уровнями возбуждения нейрона. При сильных происходит насыщение активационной функции, и таким образом, нейрон функционирует в широком диапазоне входных сигналов. Коэффициент α определяет крутизну сигмоида [3].

Таким образом, работа ИНС заключается в последовательности следующих этапов. При использовании ИНС во входные элементы подаются значения входных переменных, затем последовательно отрабатывают нейроны промежуточных и выходного слоев. Каждый из них вычисляет свое значение активации, беря взвешенную сумму выходов элементов предыдущего слоя и вычитая из нее пороговое значение. Затем значение активации преобразуются с помощью функции активации, и в результате получается выход нейрона. После того, как вся сеть отработает, выходные значения элементов выходного слоя принимаются за выход всей сети в целом.

Рисунок 2 ‒ Функции активации

Все многообразие сетей основано на их отличиях в топологии (конфигурация нейронов), характере связей между нейронами (с прямыми, обратными связями, рекуррентные сети), функциях активации нейронов, значениями весовых коэффициентов.

## **1.4 Обучение нейронных сетей**

Процесс обучения нейронных сетей заключается в выборе весовых коэффициентов на основе выбранного алгоритма. Коэффициенты подбираются таким образом, чтобы заданное выходное пространство соответствовало существующим реальным данным – обучающей выборке, или формировало выходное пространство на основе существующих данных. Заметим, что важным является объем данных, используемых при обучении сети. Чем больше объем, тем лучше будет работать сеть, обученная на этих данных. Выделяют следующие виды обучения нейронных сетей:

1. обучение без учителя – сеть формирует выходное пространство на основе входных воздействий;
2. обучение с учителем – выходное пространство решений сети известно;
3. обучение с подкреплением – в случае правильного ответа сеть получает “поощрение”, иначе “штраф”.

В процессе обучения сеть в заданном порядке просматривает некоторый набор данных. Заданный порядок может быт последовательным, случайным и т.д. Некоторые сети, такие как сеть Хопфилда, просматривают выборку один раз. Другие, такие как сеть Конохена, а также нейронные сети, обучающиеся с учителем, просматривают выборку множество раз. Один полный проход по выборке называется эпохой обучения. При обучении с учителем исходные данные произвольно делятся на две части – обучающую выборку и тестовые данные. Тестовые данные используются для расчета ошибки сети. Также с их помощью можно определить, происходит ли переобучение сети. В этом случае обучение прекращается. Тестирование качества обучения нейронных сетей производится на примерах, не участвовавших в обучении. Задача тестирования качества является трудной, поскольку, если сеть имеет низкую вероятность ошибки, то доказать это можно многочисленными тестами.

## **1.5 Нейронная сеть Перцептрон**

## 

При конструировании нейронной сети важным является выбор топологии сети, числа нейронов, связей между ними. Согласно следствию из теоремы Колмогорова−Хет−Нильсона [7], существует оптимальная нейронная сеть типа Перцептрон, формирующая максимально верное выходное пространство при выборе некоторых функций активации нейронов и определенного количества нейронов в слоях. Поэтому, для решения задачи распознавания символов была выбрана сеть Перцептрон.

Перцептрон был изобретен в 1958 году Ф. Розенблаттом[[3]](#_СПИСОК_ИСПОЛЬЗОВАННЫХ_ИСТОЧНИКОВ). Данная нейронная сеть обучается с учителем. Элементарный перцептрон состоит из элементов трёх типов: S−элементов, A−элементов и одного R−элемента. S-элементы − это слой сенсоров или рецепторов. В физическом воплощении они соответствуют, например, светочувствительным клеткам сетчатки глаза или фоторезисторам матрицы камеры. Каждый рецептор может находиться в одном из двух состояний − покоя или возбуждения, и только в последнем случае он передаёт единичный сигнал в следующий слой, ассоциативным элементам.

A−элементы называются ассоциативными, потому что каждому такому элементу, как правило, соответствует целый набор (ассоциация) S−элементов. A-элемент активизируется, как только количество сигналов от S−элементов на его входе превысило некоторую величину θ. Таким образом, если набор соответствующих S−элементов располагается на сенсорном поле в форме буквы «Д», A−элемент активизируется, если достаточное количество рецепторов сообщило о появлении «белого пятна света» в их окрестности, то есть A−элемент будет как бы ассоциирован с наличием/отсутствием буквы «Д» в некоторой области.

Сигналы от возбудившихся A−элементов, в свою очередь, передаются в сумматор R, причём сигнал от i-го ассоциативного элемента передаётся с коэффициентом. Этот коэффициент называется весом A−R связи.

Также, как и A-элементы, R−элемент подсчитывает сумму значений входных сигналов, помноженных на веса (линейную форму). R−элемент, а вместе с ним и элементарный перцептрон, выдаёт «1», если линейная форма превышает порог θ, иначе на выходе будет «−1».

Обучение элементарного перцептрона состоит в изменении весовых коэффициентов связей A−R. Веса связей S−A (которые могут принимать значения {−1; 0; +1}) и значения порогов A−элементов выбираются случайным образом в самом начале и затем не изменяются. После обучения перцептрон готов работать в режиме распознавания или обобщения. В этом режиме перцептрону предъявляются ранее неизвестные ему объекты, и перцептрон должен установить, к какому классу они принадлежат. Работа перцептрона состоит в следующем: при предъявлении объекта возбудившиеся A−элементы передают сигнал R−элементу, равный сумме соответствующих коэффициентов. Если эта сумма положительна, то принимается решение, что данный объект принадлежит к первому классу, а если она отрицательна − то ко второму [[4]](#_СПИСОК_ИСПОЛЬЗОВАННЫХ_ИСТОЧНИКОВ). Известно, что перцептрон является достаточно популярным решением задач такого плана, как:

1. распознавание образов;
2. принятие решения и управления;
3. кластеризация;
4. прогнозирование;
5. аппроксимация;
6. анализ данных.

# **2 Постановка задачи и проектирование нейронные сети**

## **2.1 Постановка задачи**

Требуется реализовать нейронную сеть, которая будет анализировать изображения с рукописными символами и определять, какой рукописный символ изображен. Сеть должна будет работать следующим образом: на вход нейронной сети подается изображение с рукописным символом, созданное пользователем при помощи различных средств манипуляции курсором (сенсорная панель, компьютерная мышь и т.д). Требуется распознать, какой символ изображен и показать пользователю. В случае правильного ответа сеть может запомнить нарисованный символ, чтобы в случае повторного распознавания быстрее определить нарисованный символ. Внесение изображения нарисованного символа в память нейрона должно происходить с разрешения пользователя.

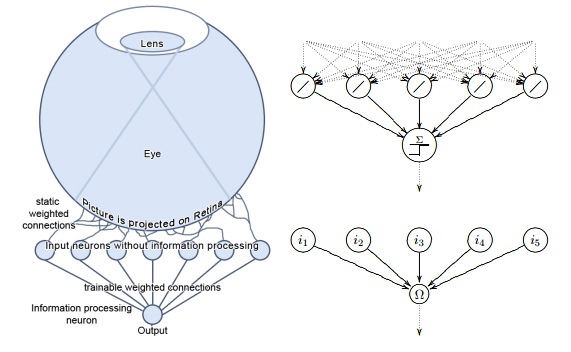


Рисунок 3 – Нейронная сеть типа Перцептрон для задачи распознавания символов

Для построения сети выбрана топология двухслойного Перцептрона, которая по своей работе напоминает глаз человека (рисунок 4) [[7]](#_СПИСОК_ИСПОЛЬЗОВАННЫХ_ИСТОЧНИКОВ)

# **3 Описание реализации нейронной сети и алгоритма распознавания**

Для решения задачи распознавания цифр выбрана и обучена нейронная сеть типа двухслойный Перцептрон с 782 нейронами во входном слое, 20 нейронами в скрытом слое и N нейронами − в выходном, где N − количество рукописных символов, которые сеть должна уметь распознавать. Для задачи распознавания цифр выходной слой содержит 10 нейронов. Каждый нейрон имеет имя, память, количество запомненных образов и отвечает за определенный символ. Для нейрона создан специальный класс, в котором будут храниться имя нейрона, память в качестве атрибутов и методы для работы с атрибутами − распознавание поступившего изображения рукописного символа, а также обучение нейрона. Для активации нейрона в процессе обучения была выбрана пороговая функция активации. Сеть учитывает ошибку при пересчете памяти нейрона, в чем ей помогает счетчик числа обучений. Для распознавания символа был выбран алгоритм, отыскивающий в памяти нейронов максимально похожий образ. Имя нейрона, чьи образы в памяти максимально совпадают с входным изображением, определяет название символа.

## **3.1 Описание основных методов**

Для реализации был выбран язык программирования C# − высокоуровневый язык программирования. Синтаксис ядра C# объёмен. Тем не менее, встроенные библиотеки С# содержат много полезных функций.

Алгоритм:

Для визуализации работы нейронной сети используется библиотека Windows Forms. В главной форме (рисунок 5) имеются такие элементы как:

1. поле PictureBox. Именно то, что будет нарисовано на этом поле, будет использовано сетью для обучения, если включен режим обучения, и для распознания рукописного символа. Изображение создается пользователем при помощи компьютерной мыши;
2. поле ComboBox. Являет собой список нейронов, которые принадлежат нейронной сети;
3. поле TextBox. Используется для создания нейронов вручную, то есть, без загрузки значений из указанного текстового файла. Нейрон создает при нажатии кнопки “Добавить значение”;
4. меню ToolStrip − меню команд, доступных сети. Включает в себя команды:
5. выход; (Память всех созданных нейронов сохраняется в текстовом файле. Файл с памятью будет использован при повторном запуске сети);
6. создать нейроны, используя значения из указанного текстового файла; (Нейроны создаются с пустой памятью);
7. добавить образ в память; (Нарисованный образ добавляется в память нейрона. Для работы этой команды необходим включенный режим обучения);
8. включить режим обучения; (Нейронная сеть будет запоминать нарисованный образ, если это позволит пользователь);
9. прекратить обучение; (Нейронная сеть перестает запоминать образы и может только распознавать);
10. очистить рисунок.
11. кнопка, нажатие которой вызывает метод класса NeuroNet, отвечающий за распознание нарисованного образа.

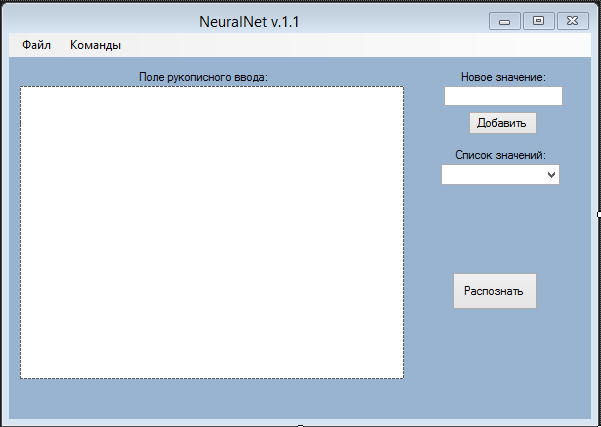


Рисунок 4 – Главная форма

Для работы нейронной сети имеются классы:

1. Neuron − класс нейрона, имеющий методы для создания нейрона, его обучения и предсказания символов; (Каждый созданный нейрон отвечает за определенный символ);
2. NeuroNet – класс нейронной сети, имеющий методы для создания сети, предсказания символов; (В качестве атрибута имеется список нейронов, отвечающих за разнообразные символы);
3. NeuroGraphics − класс, отвечающий за обработку входного изображения. (Имеются методы очистки поля PictureBox, обрезки изображения, преобразования изображения);

При начале работы НС не в состоянии ничего распознавать, поскольку память пуста. Поэтому, НС требуется обучать определенным символам. В список значений, который пуст, вносится значение. На поле PictureBox рисуется рукописный символ, которому нужно обучить сеть. (рисунок 5)

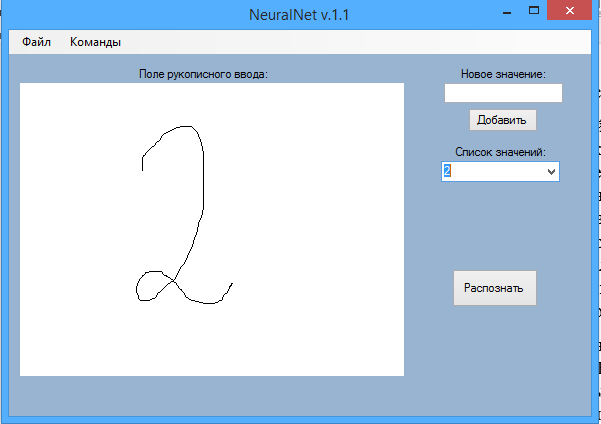


Рисунок 5 – Пример входного изображения

Далее создается нейрон с именем, соответствующему значению в списке значений. В программе нейрону посвящен отдельный класс. В память созданного нейрона, заносится нарисованное изображение в пиксельном виде и нейрон активируется с помощью пороговой функции. В то же время, счетчик, отвечающий за число образов, известных данному нейрону, увеличивается на 1, поскольку нейрон обучен новому образу. После многочисленных повторений обучения НС готова к работе. Пользователь может пользоваться любым координатным устройством для управления курсором (сенсорная панель, компьютерная мышь, трекбол).

Класс для нейрона. (рисунок 6)

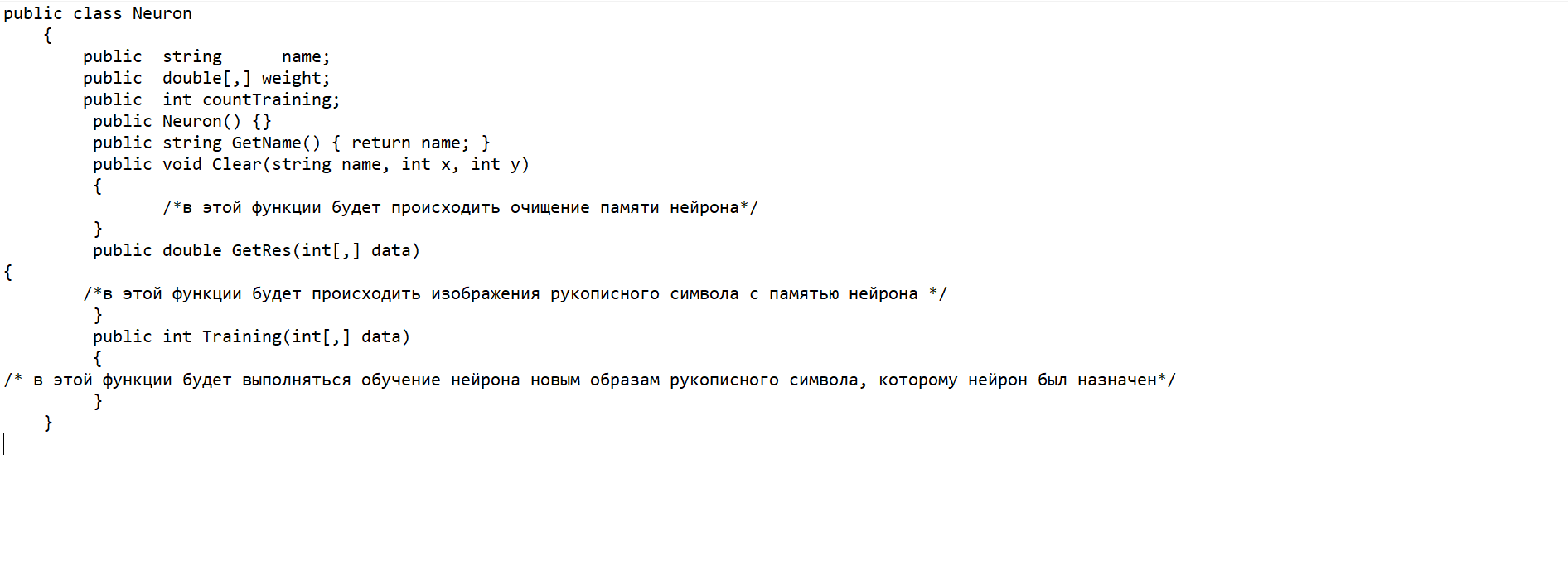


Рисунок 6 − Реализация класса Neuron

Сети выделен класс NeuroNet. Нарисованный в PictureBox рукописный символ поступает на вход сети в виде изображения. За обработку изображения отвечает класс NeuroGraphics. Изображение обрезается и приводится к стандартному размеру. Далее, преобразованное изображение показывается всем нейронам сети. Ответом будет имя нейрона, в памяти которого будет максимально похожий образ. Если включен режим обучения и ответ будет верным, будет предложено запомнить нарисованный образ.

Класс для нейронной сети. (рисунок 7)

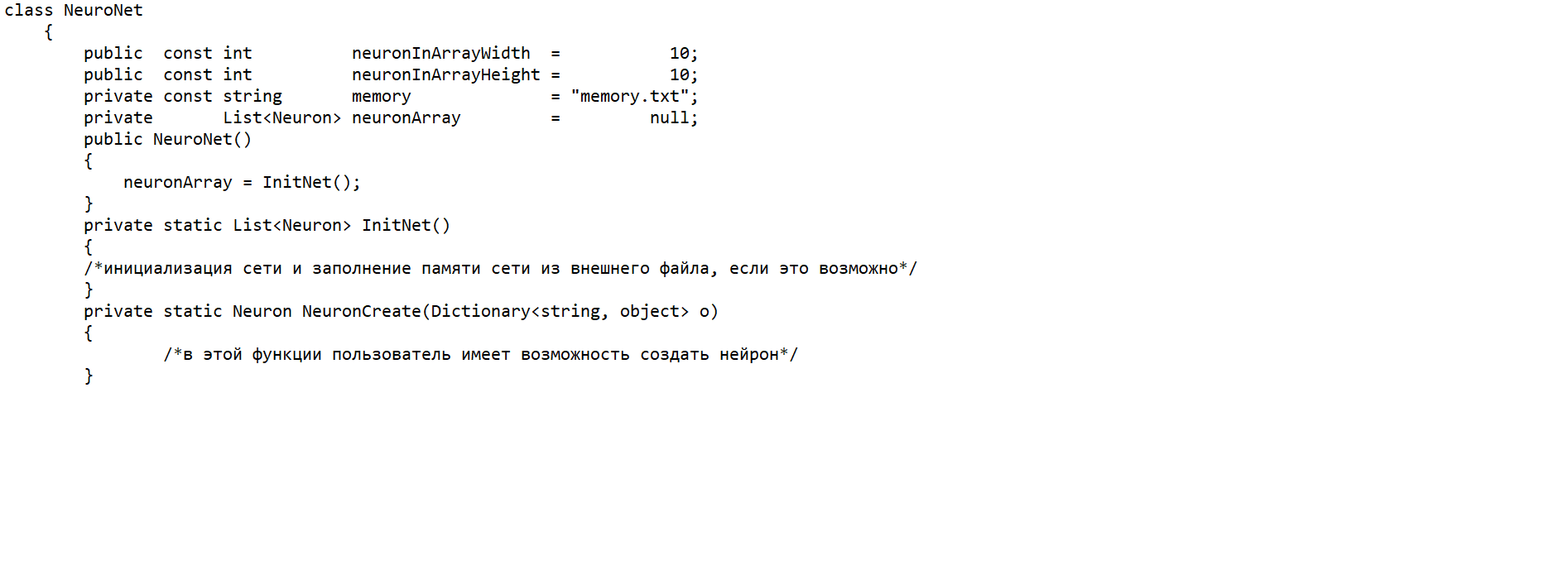
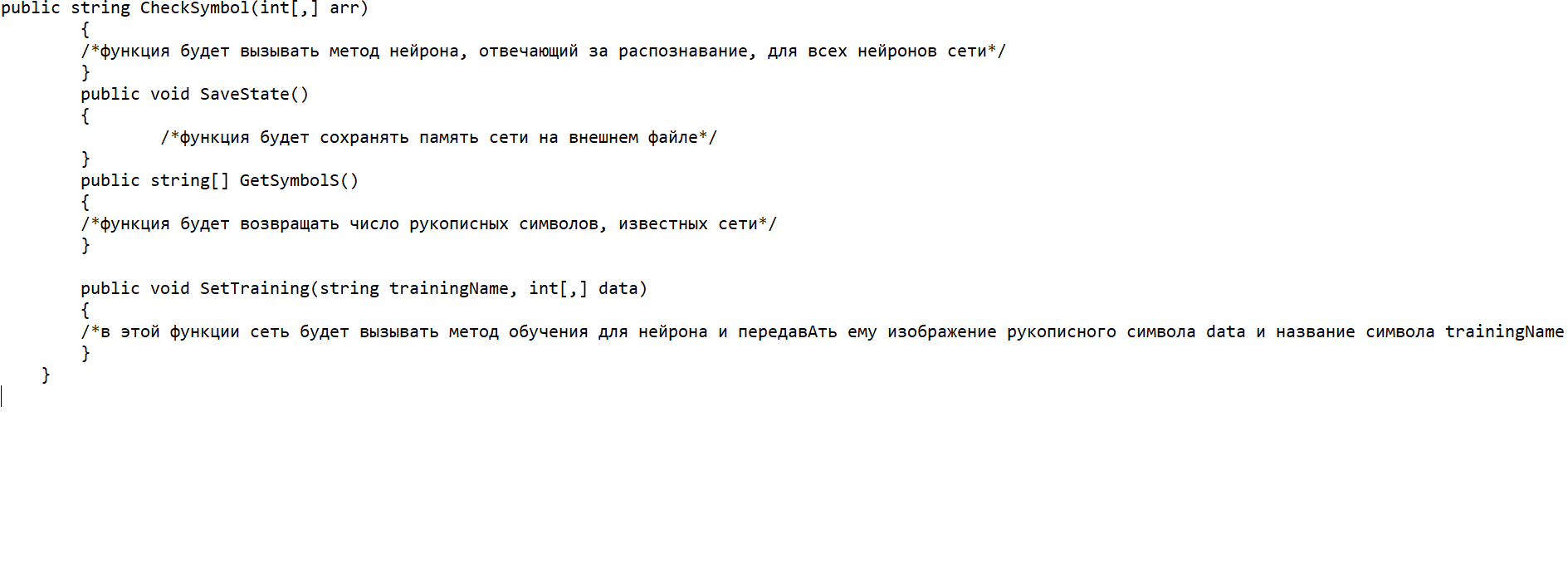


Рисунок 7 − Реализация класса NeuroNet

Класс для CheckSymbol (рисунок 8)

 Рисунок 8 – Реализация класса CheckSybol

Класс для обработки входного изображения рукописного символа. (рисунок 9)

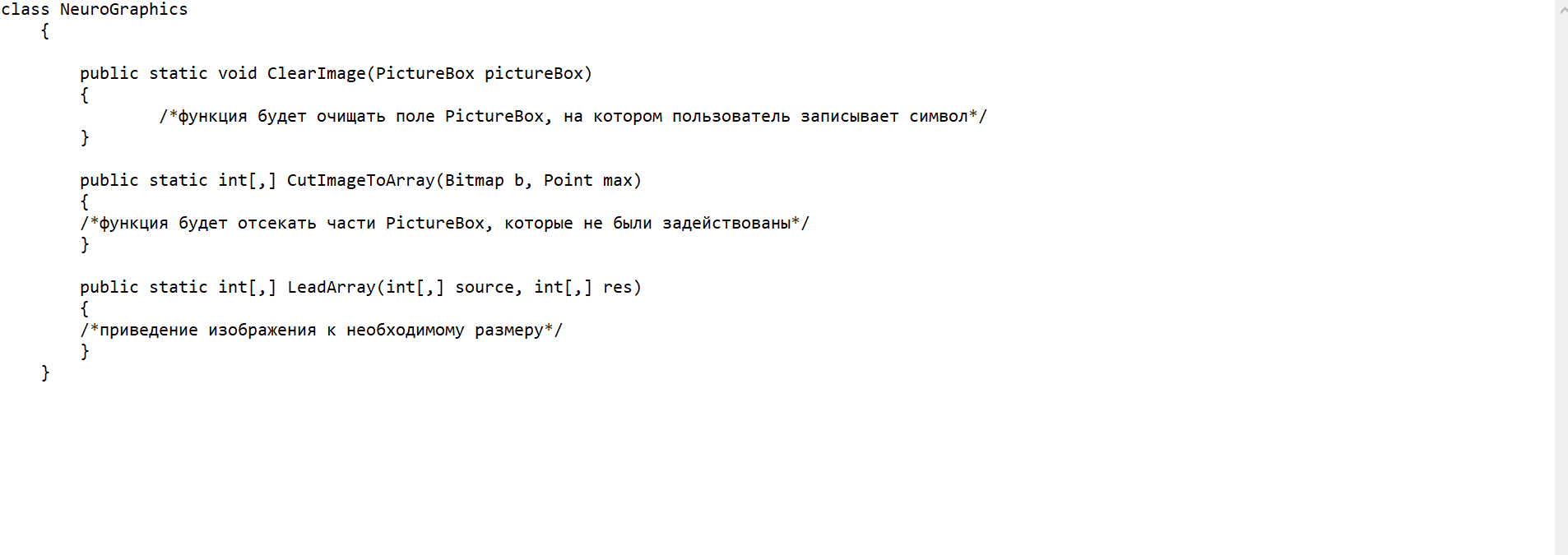


Рисунок 9 − Реализация класса ClearImage

При желании пользователя для построенной сети возможно переобучение. Если сеть верно распознала рукописные символы, пользователю предлагается внести изображение рукописного символа в память. Количество образов символов в памяти является атрибутом в классе, отвечающем за нейрон. После переобучения пользователю доступно количество образов в памяти, чтобы знать, насколько сеть обучена. Для реализации переобучения сети программа выдает запрос о степени верности ответа после каждой попытки распознавания. Если пользователь будет переобучать сеть распознавать рукописный символ на основе неверных данных, то сеть будет беспрекословно верить пользователю. Но это может привести к некорректному распознаванию.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы были изучены общие сведения о нейронных сетях и некоторые их возможности, а также реализована нейронная сеть Перцептрон на языке программирования C#.

Нейронная сеть получала на вход символы, производила анализ и заполняла память нейронов новыми образами. Результаты распознавания оказывались успешными в большинстве случаев и ошибки были лишь в том случае, когда символ был изображен слишком небрежно или количество образов в памяти нейронов было мало. В ходе тестирования программы были выявлены преимущества и недостатки реализованной сети и алгоритма. К преимуществам можно отнести универсальность сети и алгоритма распознавания, а также явно выраженная эффективность Перцептрона по сравнению с когнитроном, в процессе обучения которого нужно указывать все возможные образы распознаваемого символа. Алгоритм и его модификации могут применяться для решения широкого круга задач, в том числе и задач, связанных с безопасностью. Одним из недостатков является то, что обучающую выборку пользователь вынужден вводить вручную, что приводит к большим временным затратам. Также определено, что нейронная сеть Перцептрон не может распознать нарисованный символ, если он нарисован под другим углом.

Так как Перцептрон − одна из самых ранних реализаций нейронной сети, более современные виды нейронных сетей, такие как сеть Хопфилда, сеть Конохена и сверточные сети, должны справляться с задачей распознавания эффективнее и обучаться самостоятельно. В дальнейшем планируется реализация алгоритма распознавания рукописных символов при помощи более современной нейронной сети с использованием алгоритмов глубокого обучения, призванных повысить эффективность работы нейронных сетей.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Мак-Каллок У. С., Питтс В. Логическое исчисление идей, относящихся к нервной активности. Автоматы. Под ред. К. Э. Шеннона и Дж. Маккарти., стереотип. –М.: Изд-во иностр. лит., 1956. ––С. 363-384.
2. С. Хайкин. Нейронные сети. Полный курс 2-е изд., испр.: Пер. с англ. – М.: ООО «И. Д. Вильямс», 2006. – 1104 с.
3. F. Rosenblat. The perceptron: a probabilistic model for information storage and organization in the brain. Psychology Review, 65: 386-408, 1958.
4. Персептрон [Электронный ресурс]. –<http://www.machinelearning.ru/wiki/index.php?title=Персептрон>(29.10.2018)
5. Нейронные сети для начинающих [Электронный ресурс]. –<https://habrahabr.ru/post/312450> (3.11.2018).
6. Круглов В.В., Борисов В.В., Искуственные нейронные сети. Теория и практика. – 2-е изд., стереотип. –М.: Горячая линия-Телеком, 2002. 382 с.
7. R. Hecht-Nielson. Neurocomputing: Picking the human brain. IEEE March 1986, Spectrum, pp. 36-41,
8. A Brief Introduction to Neural Networks [Электронный ресурс].–<http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-1col-dkrieselcom.pdf> (6.11.2018)

# **ПРИЛОЖЕНИЕ А**

# **Реализация класса, отвечающего за нейрон**

using System;

namespace NeuroNet1

{

public class Neuron

{

public string name; // имя - текстовое значение образа который хранит нейрон

public double[,] weight; // массив весов - память нейрона

public int countTraining; // количество вариантов образа в памяти

// необходимо для правильного пересчёта весов при обучении

public Neuron() {}

public string GetName() { return name; }

// очистить память нейрона и присвоить ему новое имя

public void Clear(string name, int x, int y)

{

this.name = name;

weight = new double[x,y];

for (int n = 0; n < weight.GetLength(0); n++)

for (int m = 0; m < weight.GetLength(1); m++) weight[n, m] = 0;

countTraining = 0;

}

// функция возвращает сумму величин отклонения входного массива от эталонного

public double GetRes(int[,] data){

if (weight.GetLength(0) != data.GetLength(0) || weight.GetLength(1) != data.GetLength(1)) return -1;

double res = 0;

for (int n = 0; n < weight.GetLength(0); n++)

for (int m = 0; m < weight.GetLength(1); m++)

res += 1 - Math.Abs(weight[n, m] - data[n, m]); // в этой строке мы считаем отклонения каждого элемента входного массива от усреднённого значения из памяти

return res / (weight.GetLength(0) \* weight.GetLength(1));// возвращем среднее арифметическое отклонение по массиву

}

// добавить входной образ в память массива

public int Training(int[,] data)

{

// проверим что массив существует и тех же размеров что и массив памяти

if (data == null || weight.GetLength(0) != data.GetLength(0) || weight.GetLength(1) != data.GetLength(1)) return countTraining;

countTraining++;

for (int n = 0; n < weight.GetLength(0); n++)

for (int m = 0; m < weight.GetLength(1); m++)

{

// на всякий случай приведём значение элемента входного массива к дискретному

double v = data[n, m] == 0 ? 0 : 1;

// каждый элемент в памяти пересчитывается с учетом значения из data

weight[n, m] += 2 \* (v - 0.5f) / countTraining;

if (weight[n, m] >= 1) weight[n, m] = 1; // значение памяти не может быть больше 1

if (weight[n, m] < 0) weight[n, m] = 0; // значение памяти не может быть меньше 0

}

return countTraining; // вернуть количество обучений

}

}

}

# **ПРИЛОЖЕНИЕ Б**

# **Реализация класса, отвечающего за нейронную сеть**

using System;

using System.Collections.Generic;

using System.IO;

using System.Web.Script.Serialization;

using System.Windows.Forms;

namespace NeuroNet1

{

class NeuroNet

{

public const int neuronInArrayWidth = 10; // количество по горизонтали

public const int neuronInArrayHeight = 10; // количество по вертикали

private const string memory = "memory.txt"; // имя файла хранения сети

private List<Neuron> neuronArray = null; // массив нейронов

// конструктор

public NeuroNet()

{

neuronArray = InitNet();

}

// функция открывает текстовой файл и преобразовывает его в массив нейронов

private static List<Neuron> InitNet()

{

if (!File.Exists(memory)) return new List<Neuron>();

string[] lines = File.ReadAllLines(memory);

if (lines.Length == 0) return new List<Neuron>();

string jStr = lines[0];

JavaScriptSerializer json = new JavaScriptSerializer();

List<Object> objects = json.Deserialize<List<Object>>(jStr);

List<Neuron> res = new List<Neuron>();

foreach (var o in objects) res.Add(NeuronCreate((Dictionary<string,Object>)o));

return res;

}

// преобразовать структуру данных в класc нейрона

private static Neuron NeuronCreate(Dictionary<string, object> o)

{

Neuron res = new Neuron();

res.name = (string)o["name"];

res.countTraining = (int)o["countTraining"];

Object[] weightData = (Object[])o["weight"];

int arrSize = (int)Math.Sqrt(weightData.Length);

res.weight = new double[arrSize, arrSize];

int index = 0;

for (int n = 0; n < res.weight.GetLength(0); n++)

for (int m = 0; m < res.weight.GetLength(1); m++)

{

res.weight[n, m] = Double.Parse(weightData[index].ToString());

index++;

}

return res;

}

// функция сравнивает входной массив с каждым нейроном из сети и

// возвращает имя нейрона наиболее похожего на него

// именно эта функция отвечает за распознавание образа

public string CheckSymbol(int[,] arr)

{

string res = null;

double max = 0;

foreach (var n in neuronArray)

{

double d = n.GetRes(arr);

if (d > max)

{

max = d;

res = n.GetName();

}

}

return res;

}

// функция сохраняет массив нейронов в файл

public void SaveState()

{

JavaScriptSerializer json = new JavaScriptSerializer();

string jStr = json.Serialize(neuronArray);

System.IO.StreamWriter file = new System.IO.StreamWriter(memory);

file.WriteLine(jStr);

file.Close();

}

// получить список имён образов, имеющихся в памяти

public string[] GetSymbolS()

{

var res = new List<string>();

for (int i = 0; i < neuronArray.Count; i++) res.Add(neuronArray[i].GetName());

res.Sort();

return res.ToArray();

}

// эта функция заносит в память нейрона с именем trainingName

// новый вариант образа data

public void SetTraining(string trainingName, int[,] data)

{

Neuron neuron = neuronArray.Find(v => v.name.Equals(trainingName));

if (neuron == null) // если нейрона с таким именем не существует, создадим новый и добавим его в массив нейронов

{

neuron = new Neuron();

neuron.Clear(trainingName, neuronInArrayWidth, neuronInArrayHeight);

neuronArray.Add(neuron);

}

int countTrainig = neuron.Training(data); // обучим нейрон новому образу

string messageStr = "Имя образа - " + neuron.GetName() +

", вариантов образа в памяти - " + countTrainig.ToString();

MessageBox.Show(messageStr);

}

}

}

# **ПРИЛОЖЕНИЕ В**

# **Реализация класса, отвечающего за обработку изображения рукописного символа**

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Windows.Forms;

namespace NeuroNet1

{

// в этом классе собраны функции для преобразования изображений

class NeuroGraphics

{

// очищает рисунок

public static void ClearImage(PictureBox pictureBox)

{

pictureBox.Image = (Image)new Bitmap(pictureBox.Width, pictureBox.Height);

}

// отпечатать строковый символ на графическом рисунке

public static Image DrawLitera(Image bmp, string l)

{

Font myFont = new Font("Arial", 80f);

using (Graphics g = Graphics.FromImage(bmp))

{

SizeF size = g.MeasureString(l, myFont);

g.TextRenderingHint = System.Drawing.Text.TextRenderingHint.AntiAlias;

g.InterpolationMode = InterpolationMode.HighQualityBicubic;

g.PixelOffsetMode = PixelOffsetMode.HighQuality;

g.DrawString(l, myFont, new SolidBrush(Color.Black), Point.Empty);

}

return bmp;

}

// обрезать рисунок по краям и преобразовать в массив

public static int[,] CutImageToArray(Bitmap b, Point max)

{

int x1 = 0;

int y1 = 0;

int x2 = max.X;

int y2 = max.Y;

for (int y = 0; y < b.Height && y1 == 0; y++)

for (int x = 0; x < b.Width && y1 == 0; x++)

if (b.GetPixel(x, y).ToArgb() != 0) y1 = y;

for (int y = b.Height - 1; y >= 0 && y2 == max.Y; y--)

for (int x = 0; x < b.Width && y2 == max.Y; x++)

if (b.GetPixel(x, y).ToArgb() != 0) y2 = y;

for (int x = 0; x < b.Width && x1 == 0; x++)

for (int y = 0; y < b.Height && x1 == 0; y++)

if (b.GetPixel(x, y).ToArgb() != 0) x1 = x;

for (int x = b.Width - 1; x >= 0 && x2 == max.X; x--)

for (int y = 0; y < b.Height && x2 == max.X; y++)

if (b.GetPixel(x, y).ToArgb() != 0) x2 = x;

if (x1 == 0 && y1 == 0 && x2 == max.X && y2 == max.Y) return null;

int size = x2 - x1 > y2 - y1 ? x2 - x1 + 1 : y2 - y1 + 1;

int dx = y2 - y1 > x2 - x1 ? ((y2 - y1) - (x2 - x1)) / 2 : 0;

int dy = y2 - y1 < x2 - x1 ? ((x2 - x1) - (y2 - y1)) / 2 : 0;

int[,] res = new int[size, size];

for (int x = 0; x < res.GetLength(0); x++)

for (int y = 0; y < res.GetLength(1); y++)

{

int pX = x + x1 - dx;

int pY = y + y1 - dy;

if (pX < 0 || pX >= max.X || pY < 0 || pY >= max.Y)

res[x, y] = 0;

else

res[x, y] = b.GetPixel(x + x1 - dx, y + y1 - dy).ToArgb() == 0 ? 0 : 1;

}

return res;

}

// пересчитать массив source в массив res - используется для

// приведения произвольного массива данных к массиву стандартных размеров

public static int[,] LeadArray(int[,] source, int[,] res)

{

for (int n = 0; n < res.GetLength(0); n++)

for (int m = 0; m < res.GetLength(1); m++) res[n, m] = 0;

double pX = (double)res.GetLength(0) / (double)source.GetLength(0);

double pY = (double)res.GetLength(1) / (double)source.GetLength(1);

for (int n = 0; n < source.GetLength(0); n++)

for (int m = 0; m < source.GetLength(1); m++)

{

int posX = (int)(n \* pX);

int posY = (int)(m \* pY);

if (res[posX, posY] == 0) res[posX, posY] = source[n, m];

}

return res;

}

}

}