Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГООБРАЗОВАНИЯ

«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙУНИВЕРСИТЕТ»

Факультет математики и информационных технологий

Кафедра компьютерной безопасности и математического

обеспечения информационных систем

**КУРСОВАЯ РАБОТА**

по дисциплине «Языки программирования»

**Проектирование и разработка программы для распознавания рукописных цифр с помощью нейронной сети**

ОГУ 10.05.01.3017.094 ОО

Руководитель

кандидат

физико-математических наук

\_\_\_\_\_\_\_ К.Р. Джукашев

подпись инициалы фамилия

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2022г.

Студент группы

21КБ(с)РЗПО

\_\_\_\_\_\_\_ Н.С. Фот

подпись инициалы фамилия

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2022г.

Оренбург 2022

**Содержание**

[Приложение И 3](#_Toc122911822)

[Введение 4](#_Toc122911824)

[1 Теоретическая основа курсовой работы 5](#_Toc122911825)

[1.1 Теория о всех вариантов алгоритмов: нейросети 5](#_Toc122911826)

[1.2 Выбор методов и алгоритмов. Варианты активационных функций. 6](#_Toc122911827)

[1.3 Типы нейронных сетей 9](#_Toc122911828)

[1.4 Типы задач, которые решают нейронные сети 11](#_Toc122911829)

[1.5 Виды нейронных сетей 11](#_Toc122911830)

[1.6 Обучение нейронной сети 13](#_Toc122911831)

[1.7 Ошибки нейронной сети 15](#_Toc122911832)

[2 Практическая основа курсовой работы 16](#_Toc122911833)

[2.1 Выбор типа нейронной сети и определение задачи 16](#_Toc122911834)

[2.2 Выбор активационной функции 17](#_Toc122911835)

[2.3 Выбор типа обучения нейронной сети 18](#_Toc122911836)

[2.4 Структура выбранной нейронной сети 20](#_Toc122911837)

[3 Описание программного продукта 22](#_Toc122911843)

[3.1 Файлы программного продукта и их назначение 22](#_Toc122911844)

[3.2 Описание модулей программного продукта 24](#_Toc122911845)

[4 Руководство пользователя 31](#_Toc122911859)

[4.1 Сведения о файлах программы и способе их установки 31](#_Toc122911860)

[4.2 Интерфейс и функциональность программного продукта 31](#_Toc122911861)

[Заключение 40](#_Toc122911862)

[Список использованных источников 41](#_Toc122911863)

[Приложение A 42](#_Toc122911864)

# **Приложение И**

***(обязательное)***

Данная курсовая работа посвящена нейросетевому программирования и вопросу актуальности последнего в современном мире. В ней рассматриваются разные подходы к решению аналитических задач путём использования нейронных сетей, и решается непосредственная задача по реализации оной.

Большое место в работе уделено анализу всех типов и вариантов реализации нейронных сетей, необходимых для решения поставленных заданий. Так же рассмотрены разные варианты применения нейросетевого программирования в настоящее время.

В работе приводятся результаты работы по изучению и созданию конечного программного продукта, реализующего демонстрацию работы нейронных сетей на примере определения рукописного ввода пользователя.

Работа представляет интерес с точки зрения демонстрации создания нейронной сети без использования сторонних библиотек, что в свою очередь даёт понимание работы последней с точки зрения, как и математики, так и программирования.

Работа содержит 57 листов текста, 20 рисунков, 13 таблиц, 2 приложения.

# **Введение**

На данный момент можно точно сказать одно - количество информации, генерируемое человечеством за определённый промежуток времени, невозможно обработать традиционными алгоритмическими конструкциями. В связи с этим, смотря на невероятно эффективную работу мозга, программистами был разработан специальный алгоритм, который смог с эмулировать работу отдельных нейронов.

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.

Актуальность работы обусловлена тем, что в современном мире всё больше людей взаимодействуют с компьютером, заведомо не имея даже начальных навыков компьютерной грамотности, из-за чего уменьшение порога для входа за счёт понимания компьютером более высокоуровневых языков является одной из главных задач перед программистами.

Таким образом мы пришли к выводу о том, что использование выше описанных нейронных сетей является большой необходимостью из-за чего целью данной курсовой работы стала разработка программного продукта, реализующего работу и демонстрацию работы нейронной сети на примере распознавании рукописных цифр и символов.

Задачами же можно назвать:

* реализацию работы самой нейронной сети
* реализацию механизма её обучения
* реализацию пользовательского интерфейса

1 Теоретическая основа курсовой работы

1.1 Теория о всех вариантов алгоритмов: нейросети

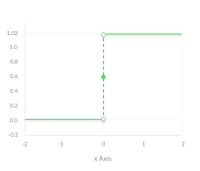
Нейронная сеть (также искусственная нейронная сеть, ИНС) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов,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

В качестве образов могут выступать различные по своей природе объекты: символы текста, изображения, образцы звуков и т. д. При обучении сети предлагаются различные образцы образов с указанием того, к какому классу они относятся. Образец, как правило, представляется как вектор значений признаков. При этом совокупность всех признаков должна *однозначно определять класс*, к которому относится образец. В случае, если признаков недостаточно, сеть может соотнести один и тот же образец с несколькими классами, что неверно. По окончании обучения сети ей можно предъявлять неизвестные ранее образы и получать ответ о принадлежности к определённому классу.

Топология такой сети характеризуется тем, что количество нейронов в выходном слое, как правило, равно количеству определяемых классов. При этом устанавливается соответствие между выходом нейронной сети и классом, который он представляет. Когда сети предъявляется некий образ, на одном из её выходов должен появиться признак того, что образ принадлежит этому классу. В то же время на других выходах должен быть признак того, что образ данному классу не принадлежит. Если на двух или более выходах есть признак принадлежности к классу, считается, что сеть «не уверена» в своём ответе.

1.2 Выбор методов и алгоритмов. Варианты активационных функций.

1) Ступенчатая функция (англ. *binary step function*) является пороговой функцией активации. То есть если «z» больше или меньше некоторого значения, то нейрон становится активированным. Такая функция отлично работает для бинарной классификации. Но она не работает, когда для классификации требуется большее число нейронов и количество возможных классов больше двух.

Рис. 1. - Ступенчатая функция

2) Линейная функция (англ. *linear function*) представляет собой прямую линию, а это значит, что результат этой функции активации пропорционален переданному аргументу (1). В отличии от предыдущей функции, она позволяет получить диапазон значений на выходе, а не только бинарные 0 и 1, что решает проблему классификации с большим количеством классов.

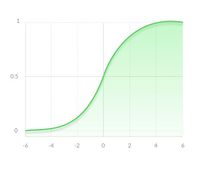
Рис. 2. - Линейная функция

(1)

3) Сигмоидная функция (англ. *sigmoid function*), которую также называет логистической (англ. *logistic function*), является гладкой монотонно возрастающей нелинейной функцией. И так как эта функция нелинейна, то ее можно использовать в нейронных сетях с множеством слоев, а также обучать эти сети методом обратного распространения ошибки. Сигмоида ограничена двумя горизонтальными асимптотами:

Что дает нормализацию выходного значения каждого нейрона. Кроме того, для сигмоидной функции характерен гладкий градиент, который предотвращает "прыжки" при подсчете выходного значения. Помимо всего этого, у этой функции есть еще одно преимущество, для значений:

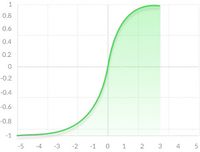
«Y» "прижимается" к одной из асимптот, что позволяет делать четкие предсказания классов.

Рис. 3. - Сигмоидная функция

(2)

4) Функция гиперболического тангенса (англ. *hyperbolic tangent*). Эта функция является скорректированной сигмоидной функцией (2) и записывается следующим образом:

То есть она сохраняет те же преимущества и недостатки, но уже для диапазона значений (−1;1). Ниже, на рисунке 3, можно увидеть график данной функции. Формула (3) является той самой сигмоидной функцией.

Рис. 3. - Сигмоидная функция

(3)

4) Данная функция возвращает 0, если принимает отрицательный аргумент, в случае же положительного аргумента, функция возвращает само число.

На первый взгляд может показаться, что она линейна (4) и имеет те же проблемы что и линейная функция, но это не так и ее можно использовать в нейронных сетях с множеством слоев.

На рисунке 5 представлен график данной функции.

Рис. 4. - ReLU

(4)

Преимущества ReLU (4) перед сигмоидой (2) и гиперболическим тангенсом (3):

1. Очень быстро и просто считается производная. Для отрицательных значений - 0, для положительных - 1.
2. Разреженность активации. В сетях с очень большим количеством нейронов использование сигмоидной функции или гиперболического тангенса в качестве активационный функции влечет активацию почти всех нейронов, что может сказаться на производительности обучения модели. Если же использовать ReLU, то количество включаемых нейронов станет меньше, в силу характеристик функции, и сама сеть станет легче.

1.3 Типы нейронных сетей

Есть десятки видов нейросетей, которые отличаются архитектурой, особенностями функционирования и сферами применения. При этом чаще других встречаются сети трех видов.

1) Нейронные сети прямого распространения (Feed forward neural networks, FFNN). Прямолинейный вид нейросетей, при котором соседние узлы слоя не связаны, а передача информации осуществляется напрямую от входного слоя к выходному. FFNN имеют малую функциональность, поэтому часто используются в комбинации с сетями других видов. Данные нейронные сети состоят из трёх типов слоёв:

* Входного
* Скрытого
* Выходного

2) Сверточные нейронные сети (Convolutional neural network, CNN). Состоят из слоев пяти типов:

* Входного
* Свертывающего
* Объединяющего
* Подключенного
* Выходного

Каждый из вышеописанных слоёв, вне зависимости от типа нейронной сети выполняет определенную задачу: например, обобщает или соединяет данные. Передаёт их далее после обработки и т.п.

Сверточные нейросети применяются для классификации изображений, распознавания объектов, прогнозирования, обработки естественного языка и других задач.

3) Рекуррентные нейронные сети (Recurrent neural network, RNN). Используют направленную последовательность связи между узлами. В RNN результат вычислений на каждом этапе используется в качестве исходных данных для следующего. Благодаря этому, рекуррентные нейронные сети могут обрабатывать серии событий во времени или последовательности для получения результата вычислений.

RNN применяют для языкового моделирования и генерации текстов, машинного перевода, распознавания речи и других задач.

1.4 Типы задач, которые решают нейронные сети

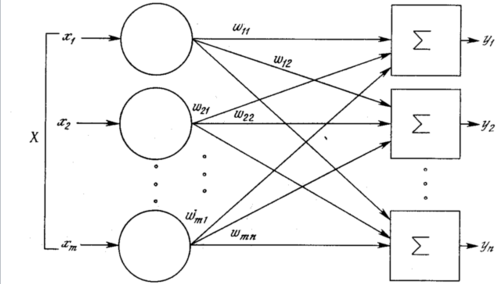
Выделяют несколько базовых типов задач, для решения которых могут использоваться нейросети.

1. Классификация. Для распознавания лиц, эмоций, типов объектов: например, квадратов, кругов, треугольников. Также для распознавания образов, то есть выбора конкретного объекта из предложенного множества: например, выбор квадрата среди треугольников.
2. Регрессия. Для определения возраста по фотографии, составления прогноза биржевых курсов, оценки стоимости имущества и других задач, требующих получения в результате обработки конкретного числа.
3. Прогнозирования временных рядов. Для составления долгосрочных прогнозов на основе динамического временного ряда значений. Например, нейросети применяются для предсказания цен, физических явлений, объема потребления и других показателей. По сути, даже работу автопилота Tesla можно отнести к процессу прогнозирования временных рядов.
4. Кластеризация. Для изучения и сортировки большого объема неразмеченных данных в условиях, когда неизвестно количество классов на выходе, то есть для объединения данных по признакам. Например, кластеризация применяется для выявления классов картинок и сегментации клиентов.
5. Генерация. Для автоматизированного создания контента или его трансформации. Генерация с помощью нейросетей применяется для создания уникальных текстов, аудиофайлов, видео, раскрашивания черно-белых фильмов и даже изменения окружающей среды на фото.

1.5 Виды нейронных сетей

1) Однослойная нейронная сеть (англ. Single-layer neural network) — сеть, в которой сигналы от входного слоя сразу подаются на выходной слой, который и преобразует сигнал и сразу же выдает ответ.

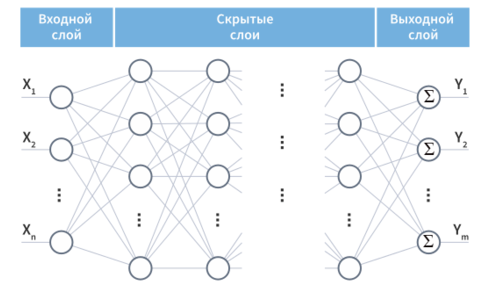
Как видно из схемы однослойной нейронной сети, представленной справа, сигналы поступают на входной слой (который не считается за слой нейронной сети), а затем сигналы распределяются на выходной слой обычных нейронов. На каждом ребре от нейрона входного слоя к нейрону выходного слоя написано число — вес соответствующей связи.

Рис. 1. - Схема однослойной нейронной сети

2) Многослойная нейронная сеть (англ. Multilayer neural network) — нейронная сеть, состоящая из входного, выходного и расположенного(ых) между ними одного (нескольких) скрытых слоев нейронов.

Помимо входного и выходного слоев эти нейронные сети содержат промежуточные, скрытые слои. Такие сети обладают гораздо большими возможностями, чем однослойные нейронные сети, однако методы обучения нейронов скрытого слоя были разработаны относительно недавно.

Работу скрытых слоев нейронов можно сравнить с работой большого завода. Продукт (выходной сигнал) на заводе собирается по стадиям на станках. После каждого станка получается какой-то промежуточный результат. Скрытые слои тоже преобразуют входные сигналы в некоторые промежуточные результаты.

Рис. 2. - Схема многослойной нейронной сети

1.6 Обучение нейронной сети

Обучение нейронной сети — это процесс обучения нейронной сети выполнению задачи. Нейронные сети обучаются путем первичной обработки нескольких больших наборов размеченных или неразмеченных данных. На основе этих примеров сети могут более точно обрабатывать неизвестные входные данные.

При контролируемом обучении специалисты по работе с данными предлагают искусственным нейронным сетям помеченные наборы данных, которые заранее дают правильный ответ. Например, сеть глубокого обучения, обучающаяся распознаванию лиц, обрабатывает сотни тысяч изображений человеческих лиц с различными терминами, связанными с этническим происхождением, страной или эмоциями, описывающими каждое изображение.

Нейронная сеть медленно накапливает знания из этих наборов данных, которые заранее дают правильный ответ. После обучения сеть начинает делать предположения об этническом происхождении или эмоциях нового изображения человеческого лица, которое она никогда раньше не обрабатывала.

Искусственный интеллект — это область компьютерных наук, которая исследует методы предоставления машинам возможности выполнять задачи, требующие человеческого интеллекта. Машинное обучение — это метод искусственного интеллекта, который дает компьютерам доступ к очень большим наборам данных для дальнейшего обучения. Программное обеспечение для машинного обучения находит шаблоны в существующих данных и применяет эти шаблоны к новым данным для принятия разумных решений. Глубокое обучение — это разновидность машинного обучения, в котором для обработки данных используются сети глубокого обучения.

Традиционные методы машинного обучения требуют участия человека, чтобы программное обеспечение работало должным образом. Специалист по работе с данными вручную определяет набор соответствующих функций, которые должно анализировать программное обеспечение. Это ограничение делает создание и управление программным обеспечением утомительным и трудозатратным процессом.

С другой стороны, при глубоком обучении специалист по работе с данными предоставляет программному обеспечению только необработанные данные. Сеть глубокого обучения извлекает функции самостоятельно и обучается более независимо. Она может анализировать неструктурированные наборы данных (например, текстовые документы), определять приоритеты атрибутов, данных и решать более сложные задачи.

Например, при обучении программного обеспечения с алгоритмами машинного обучения правильно идентифицировать изображение домашнего животного вам потребуется выполнить следующие шаги:

Найти и вручную отметить тысячи изображений домашних животных: кошек, собак, лошадей, хомяков, попугаев и т. д.

Сообщить программному обеспечению с алгоритмами машинного обучения, какие функции необходимо найти, чтобы оно могло идентифицировать изображение методом исключения. Например, оно может подсчитать количество ног, а затем проверить форму глаз, ушей, хвоста, цвет меха и так далее.

Вручную оценить и изменить помеченные наборы данных, чтобы повысить точность программного обеспечения. Например, если в вашем тренировочном наборе слишком много изображений черных кошек, программное обеспечение правильно определит черную кошку, но не белую.

При глубоком обучении нейронные сети будут обрабатывать все изображения и автоматически определять, что сначала им требуется проанализировать количество ног и форму морды, а уже после посмотреть на хвосты, чтобы правильно идентифицировать животное на изображении.

1.7 Ошибки нейронной сети

1) Для расчета ошибки выходного слоя необходимо найти производную потерь по входу выходному слою: . Это отвечает на вопрос: как веса последнего слоя влияют на общую ошибку в сети? Формула производной принимает следующий вид (1).

(1)

Чтобы упростить запись, практикующие МО обычно заменяют последовательность термином . Итак, наша формула для ошибки выходного примет такой вид (2).

(2)

2) Для вычисления ошибки скрытого слоя нужно найти производную потерь по входу скрытого слоя, (3).

(3)

Далее мы можем поменять местами элемент выше, чтобы избежать дублирования и создать новое упрощенное уравнение для ошибки скрытого слоя (4).

(4)

Эта формула лежит в основе обратного распространения. Мы вычисляем ошибку текущего слоя и передаем взвешенную ошибку обратно на предыдущий слой, продолжая процесс, пока не достигнем нашего первого скрытого слоя. Попутно мы обновляем веса, используя производную потерь по каждому весу.

3) Также теперь у нас есть возможность получить производную потерь по любому весу. Формула для получения необходимых данных будет выглядеть следующим образом:

(5)

Мы знаем, что можем заменить первую часть уравнением для ошибки выходного слоя . H представляет собой активацию скрытого слоя. Полученная формула будет выглядеть подобным образом:

(6)

Таким образом, чтобы найти производную потерь по любому весу в нашей сети (7), мы просто умножаем ошибку соответствующего слоя на его вход (выход предыдущего слоя).

(7)

2 Практическая основа курсовой работы

2.1 Выбор типа нейронной сети и определение задачи

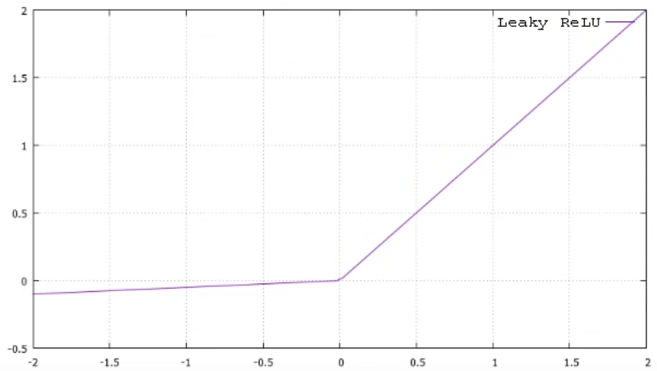
Задачей на данную курсовую работу, как было выше описано, стала разработка программного продукта, способного к распознаванию рукописных цифр. Данное задание относится к группе задач по классификации. настоящее время разработано большое количество различных видов классификаторов, для построения которых используются как статистические методы (логистическая регрессия, дискриминантный анализ), так и методы машинного обучения (нейронные сети, деревья решений, метод k-ближайших соседей, машины опорных векторов и др.), но самым прогрессивным и многообещающим, по моему мнению, являются нейронные сети.

Наиболее часто используемой для классификации архитектурой нейронных сетей являются сети прямого распространения, на входные нейроны которых подаются значения признаков классифицируемого объекта, а на выходе формируется метка или числовой код класса. Обычно используются многослойные персептроны.

2.2 Выбор активационной функции

Для реализации работы нейронной сети изначально было необходимо выбрать одну из выше представленных активационных функций нейрона. Основным претендентом стала функция, выраженная на основе «Rectified Linear Unit», или в сокращении «ReLU». Говоря про «родительскую» функцию, можно отметить то, что именно она наиболее часто используется для активации при глубоком обучении.

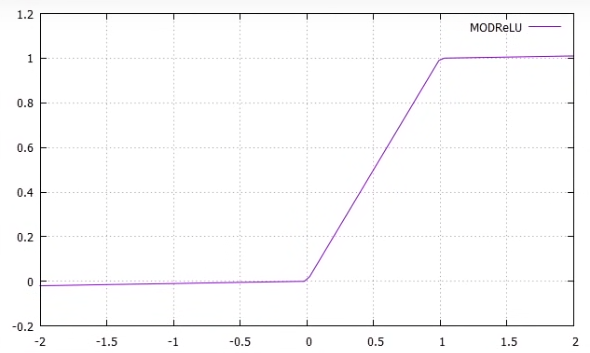
Используемая в данной курсовой работе функция является модифицированной функцией «Leaky ReLU» (2), которая в свою очередь имеет такой вид:

Рис. 4. - Функция Leaky ReLU

(1)

Причиной использования именно модифицированной активационной функции можно назвать специфику поставленной задачи. Алгоритм, при считывании пикселей, будет получать на вход его цвет, и так как это значение имеет верхний и нижние пределы, а именно 255 и 1 соответственно, «Leaky ReLU» нуждается в доработке (2).

Итоговый вид функции будет таковым:

Рис. 5. - Функция MOD ReLU

(2)

2.3 Выбор типа обучения нейронной сети

Нейронная сеть способна к обучению как человеческий мозг. Происходит этот процесс следующим образом: Обучающие образцы передаются по сети, и выходные данные, полученные от сети, сравниваются с фактическими выходными данными. Эта ошибка используется для изменения веса нейронов таким образом, чтобы ошибка постепенно уменьшалась. Это делается с помощью алгоритма обратного распространения ошибки, также называемого обратным распространением. Итеративная передача пакетов, данных по сети и обновление весов для уменьшения ошибки называется стохастический градиентный спуск (SGD). Величина, на которую изменяются вес «а», определяется параметром, называемым «Скорость обучения».

1) Обучение с учителем (supervised learning) предполагает наличие полного набора размеченных данных для тренировки модели на всех этапах ее построения.

Наличие полностью размеченного датасета означает, что каждому примеру в обучающем наборе соответствует ответ, который алгоритм и должен получить. Таким образом, размеченный датасет из фотографий цветов обучит нейронную сеть, где изображены розы, ромашки или нарциссы. Когда сеть получит новое фото, она сравнит его с примерами из обучающего датасета, чтобы предсказать ответ.

В основном обучение с учителем применяется для решения двух типов задач: классификации и регрессии.

В задачах классификации алгоритм предсказывает дискретные значения, соответствующие номерам классов, к которым принадлежат объекты. В обучающей выборке данных с фотографиями животных каждое изображение будет иметь соответствующую метку, например — «кошка», «коала» или «черепаха». Качество алгоритма оценивается тем, насколько точно он может правильно классифицировать новые фото с коалами и черепахами.

А вот задачи регрессии связаны с непрерывными данными. Один из примеров, линейная регрессия, вычисляет ожидаемое значение переменной «Y», учитывая конкретные значения «X». (Кстати, говоря, методы линейной регрессии активно используются в статистике).

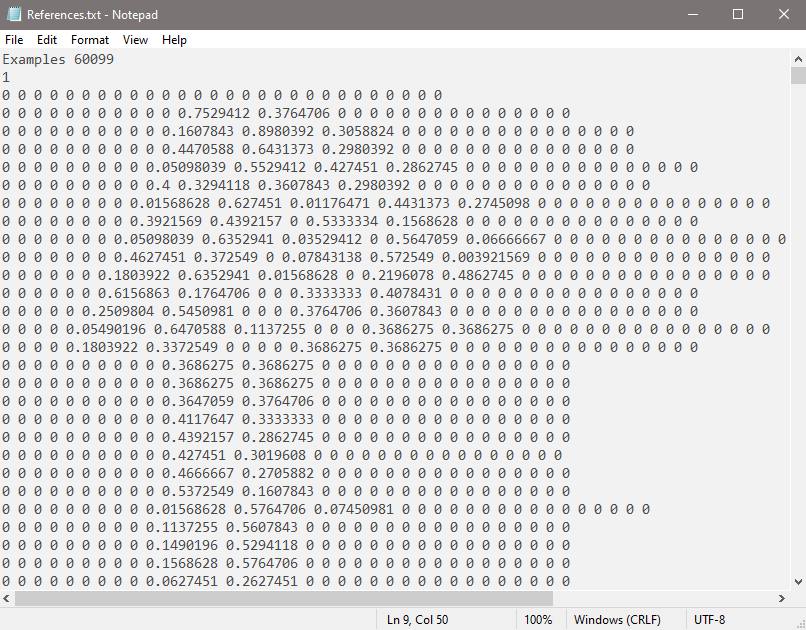
Более утилитарные задачи машинного обучения задействуют большое число переменных. Как пример, нейронная сеть, предсказывающая цену квартиры в городе на основе ее площади, местоположения и доступности общественного транспорта. Алгоритм выполняет работу эксперта (экспертов), который рассчитывает цену квартиры исходя из полученных данных.

Можно сделать вывод, что нейронные сети, обучающиеся с учителем, наиболее эффективны при наличии большой выборки достоверных данных.

2) Получить большой набор чистых и достоверных данных порой бывает нелегко. Поэтому перед алгоритмом встает задача найти заранее неизвестные ответы. В данной ситуации мы имеем дело с нейронными сетями, обучающимися без учителя.

В обучении без учителя у модели есть набор данных, и нет явных указаний, что с ним делать. Нейронная сеть пытается самостоятельно найти корреляции в данных, извлекая полезные признаки и анализируя их.

В моём случае будут реализованы оба варианта обучения нейронной сети с наличием двух датасетов для обучения и последующего теста, и так же наличием механизма обучения с учителем. Датасеты были получены из открытого источника с сайта MNIST и далее конвертированы в .txt формат.

****Рис. 6. – Фрагмент конвертированного

датасета

2.4 Структура выбранной нейронной сети

2.4.1 Основные сведения

Нейронная сеть, рассматриваемая в данной курсовой работе является многослойной нейронной сетью прямого распространения. Это говорит о том, что она имеет как минимум три типа слоёв в своём составе. А в моём случае она как раз имеет по одному слою каждого типа, а именно один входной слой, который принимает входные пользовательские данные, скрытый слой, благодаря которому происходит обработка поступивших данных, и выходной слой, анализируя который программа предполагает какое число ввёл пользователь.

2.4.2 Входной слой

В рамках данной курсовой работы, созданная мною нейронная сеть обрабатывает ручной ввод пользователя принимая на вход одномерный массив значений цветов холста размером 28 на 28 пикселей. Представить математически это можно следующим образом:

, 0 1

Получив на вход ровно 784 значений, мы тем самым заполнили первый слой нашей нейронной сети, всего который может насчитываться от 3 штук в числе который входной слой, который выступает приёмником вводимых данных, скрытый слой, количество которых и определяет глубину нейронной сети, и выходной слой, в данных которых находится ответ нейронной сети.

2.4.3 Скрытый слой

Скрытые слои нейронной сети содержат ненаблюдаемые обрабатывающие блоки (нейроны). Значение каждого скрытого нейрона — это некоторая функция предикторов. Точная форма этой функции частично зависит от типа сети и в моём случае является формой MOD ReLU.

Каждый предыдущий слой нейронной сети, вне зависимости от своего типа, исключая разве что выходной слой, так или иначе передаёт свои значения далее посредством перемножения собственных нейронов на их веса прибавляя смещение. Выглядит такая формула примерно таким образом:

2.4.4 Веса

Вес представляет силу связи между нейронами. Например, если вес соединения узлов 1 и 3 больше, чем узлов 2 и 3, это значит, что нейрон 1 оказывает на нейрон 3 большее влияние. Нулевой вес означает, что изменения входа не повлияют на выход. Отрицательный вес показывает, что увеличение входа уменьшит выход. Вес определяет влияние ввода на вывод.

В самом начале обучения веса нейронной сети заполняются либо случайными значениями и процессе обучения меняются тем самым реализуя механизм работы сети.

2.4.5 Смещение

Смещение – это дополнительный вход для нейрона, который всегда равен 1 и, следовательно, имеет собственный вес соединения. Это гарантирует, что даже когда все входы будут равны нулю, нейрон будет активен.

3 Описание программного продукта

3.1 Файлы программного продукта и их назначение

|  |  |
| --- | --- |
| Имя файла | Содержимое и назначение |
| Папка “DATA” | Содержит конфигурационные файлы и файлы весов, созданных во внутреннем конструкторе, стандартных нейронных сетей. |
| Папка “GUI” | Содержит статические классы, возвращающие элементы интерфейса. |
| Configuration.cs | Статический класс содержащий статический метод, возвращающий элемент пользовательского интерфейса. |
| Structure.cs | Статический класс содержащий статический метод, возвращающий элемент пользовательского интерфейса. |
| Папка “IMAGES” | Содержит визуальные элементы графического интерфейса программного продукта. |
| Папка “OBJECTS” | Содержит объекты необходимые для работы программного продукта. |
| Matrix.cs | Объект матрицы с методами по транспонированию и заполнению самой себя случайными числами. |
| Network.cs | Объект нейронной сети с методами по инициализации слоёв, получению информации от пользователя, обработки данных и вывода ответа. |
| Number.cs | Объект, в котором программный продукт представляет, как данные для обучения, так и данные вводимые пользователем. |
| Vector.cs | Объект, заменяющий в некоторых моментам одномерный массив. |
| Папка “Properties” | Содержит ресурсы для программного продукта, а именно стандартную конфигурацию и стандартные веса нейронной сети. |
| Папка “SCRIPTS” | Содержит вынесенные функции для обработки данных и работы с ними. |
| DataWorker.cs | Статический класс устанавливающий текущую конфигурацию нейронной сети считывая их из файлов конфигурации. |
| NeuronActivate.cs | Статический класс содержащий функции активации нейронов. |
| Prediction.cs | Статический класс по работе с нейронной сетью. Позволяет напрямую внести данные и получить ответ. |
| Teaching.cs | Статический класс реализующий два способа обучения. |
| Папка “WINDOWS” | Содержит основные окна приложения. |
| TeacherWindow.xaml | Содержит разметку окна конструктора конфигураций и обучения. |
| TeacherWindow.xaml.cs | Окно приложения, предоставляющее пользователю инструменты по созданию своих конфигураций нейронных сетей и их последующее обучение по одному из двух методов. |
| UserWindow.xaml | Содержит разметку основного окна в котором происходит работа с нейронной сетью. |
| UserWindow.xaml.cs | Окно, предоставляющее пользователю инструменты для взаимодействия с нейронной сетью. Такие как ввод данных и обратное распространение ошибки. |
| MainWindow.xaml | Содержит разметку окна приветствия. |
| MainWindow.xaml.cs | Окно приложения, в котором пользователь имеет выбор между конструктором и работой с нейронной сетью. |
| App.xaml | Объявления, определяющие поведение приложения. |

3.2 Описание модулей программного продукта

3.2.1 Модуль Configuration.cs

Данный модуль программы содержит один статический метод, считывающий конструктор.

|  |  |
| --- | --- |
| Метод | Назначение |
| public static void WriteConfig(Grid configGrid, int size) | Метод, создающий текстовый файл с созданной пользователем конфигурацией. |

3.2.2 Модуль Structure.cs

Данный модуль программы содержит один статический метод возвращающий элемент графического интерфейса.

|  |  |
| --- | --- |
| Метод | Назначение |
| public static Grid GetStructure(Teacher teacher, int size) | Метод, возвращающий сетку элементов для их встраивания в графический интерфейс. |

3.2.3 Модуль Matrix.cs

Данный объект реализует удобное взаимодействие с двумерными матрицами и содержит методы и переопределения операндов.

|  |  |
| --- | --- |
| Метод или переопределение операнда | Назначение |
| public Matrix GetTranspose() | Транспонирует матрицу. |
| public static double[] operator \*(Matrix matrix, double[] neuron) | Переназначение оператора умножения двух объектов матриц. |
| public void FillRandom() | Заполняет матрицу случайными числами. |
| public string GetValues() | Возвращает строку значений матрицы. |
| public void SetValues(string value, int x, int y) | Вставляет значение в определённые координаты. |

3.2.4 Модуль Network.cs

Данный объект является нейронной сетью и содержит основные методы взаимодействия с ней.

|  |  |
| --- | --- |
| Метод | Назначение |
| public void InsertInformation(List<double> values) | Заполняет входной слой нейронной сети значениями. |
| private int GetMaxIndex(IReadOnlyList<double> values) | Получает максимальный индекс элемента в выходном слое тем самым получая ответ. |
| public double ForwardFeed() | Метод прямого распространения, проводящий через все слои с преобразованием входные данные и возвращающий максимальный индекс элемента. |
| public void BackPropagation(double expectedAnswer) | Метод обратного распространения ошибки записывающий в значения нейронной ошибки необходимое смещение. |
| public void SetWeights(double learningRange) | Метод изменяющий веса нейронов в зависимости от ошибки и интенсивности обучения. |
| private static string GetWeights() | Метод, возвращающий веса из текстового файла. |
| public void SaveWeights() | Метод, сохраняющий веса в текстовый файл. |
| public void ReadWeights() | Метод, считывающий веса с текстового файла и записывающий их в веса нейронной сети. |

3.2.5 Модуль Number.cs

Данный объект является представлением числа программой и содержит основные данные о пикселях, значением числа и методы взаимодействия с ним.

|  |  |
| --- | --- |
| Метод | Назначение |
| public string PrintNumber() | Метод, возвращающий строку числа. |

3.2.6 Модуль Vector.cs

Данный объект является представлением одномерного массива чисел с плавающей точкой программой и содержит основные данные о нём вместе с методами взаимодействия и переопределениями операндов.

|  |  |
| --- | --- |
| Метод или переопределение операнда | Назначение |
| private double this[int key] | Возвращает значение по индексу из массива чисел с плавающей точкой. |
| private void SetElement(int index, double value) | Метод вставляющий в элемент в заданный индекс массива. |
| public static double[] operator +(Vector vector1, Vector vector2) | Переопределение операнда умножения двух объектов. |

3.2.6 Модуль DataWorker.cs

Данный модуль программы содержит методы, обрабатывающие данные и возвращающие объекты необходимые для работы нейронной сети.

|  |  |
| --- | --- |
| Метод | Назначение |
| public static Configuration ReadNetworkConfig(string config) | Возвращает структуру конфигурации нейронной сети считывая текстовый файл. |
| public static Number ReadData(string pixelsValue, Configuration configuration) | Возвращает объект числа считывая введённые пользователем данные. |
| public static List<Number> ReadData(string config, Configuration configuration, ref int examples) | Возвращает лист объектов числа считывая текстовый файл. |

3.2.7 Модуль NeuronActivate.cs

Данный модуль программы содержит активационные функции.

|  |  |
| --- | --- |
| Метод | Назначение |
| public static double[] Activation(double[] value) | Метод активации одномерного массива нейронов. |
| public static double GetDerivative(double value) | Метод нахождения производной из входного значения после активации нейрона. |

3.2.8 Модуль Prediction.cs

Данный модуль программы содержит статический метод для удобного взаимодействия с нейронной сетью, а именно для вызова функции прямого распространения с введением пользовательских данных.

|  |  |
| --- | --- |
| Метод | Назначение |
| public static int Predict(Network network, string number) | Метод, возвращающий определённое число, которое считается нейронной сетью ответом к поставленной задаче. |

3.2.9 Модуль Teaching.cs

Данный модуль программы содержат методы для взаимодействия с нейронной сетью, а именно оба варианта в качестве методов обучения последней.

|  |  |
| --- | --- |
| Метод | Назначение |
| public static void LightStudying(Network network, string number, int expected) | Метод, реализующий обучение с Учителем. |
| public static void HardStudying(Network network, int teachingCounts) | Метод, реализующий обучение без учителя по датасетам. |

3.2.10 Модуль TeacherWindow.xaml.cs

Данный модуль программы содержат методы обработчики для взаимодействия пользователя с программным продуктом.

|  |  |
| --- | --- |
| Метод | Назначение |
| public void DecreaseStructure(object sender, MouseButtonEventArgs e) | Обработчик, обрабатывающий нажатия по кнопке уменьшения структуры. |
| public void IncreaseStructure(object sender, MouseButtonEventArgs e) | Обработчик, обрабатывающий нажатия по кнопке увеличения структуры. |
| private void CloseTeacher(object sender, MouseButtonEventArgs e) | Обработчик, вызывающийся при закрытие окна. |
| private void FastTeaching(object sender, MouseButtonEventArgs e) | Обработчик, запускающий режим обучения с учителем без стандартных весов. |
| private void HardTeaching(object sender, MouseButtonEventArgs e) | Обработчик, запускающий режим обучения без учителем датасетами. |

3.2.11 Модуль UserWindow.xaml.cs

Данный модуль программы содержат методы и обработчики для взаимодействия пользователя с программным продуктом и нейронной сетью.

|  |  |
| --- | --- |
| Обработчики и методы | Назначение |
| private void AnalyzeUserInput(object sender, EventArgs eventArgs) | Метод, собирающий зарисованные пиксели холста в одномерный массив силы их закрашивания, делённый на 255, и вызываемый таймером. |
| private void UserMoveMouse(object sender, MouseEventArgs e) | Обработчик, отслеживающий движение мышки пользователя и рисующий линии. |
| private void UserClick(object sender, MouseButtonEventArgs e) | Обработчик, отслеживающий клик мыши и начинающий рисование. |
| private void BackPropagation(object sender, RoutedEventArgs e) | Обработчик, отслеживающий нажатие кнопки обратного обучения. |
| private void Clear(object sender, RoutedEventArgs e) | Обработчик, отслеживающий нажатие кнопки очистки холста. |
| private void Exit(object sender, RoutedEventArgs e) | Обработчик, отслеживающий нажатие кнопки выход. |
| private void SaveWeights(object sender, RoutedEventArgs e) | Обработчик, отслеживающий нажатие кнопки сохранения текущих весов нейронной сети в текстовый файл. |
| private void LoadWeights(object sender, RoutedEventArgs e) | Обработчик, отслеживающий нажатие кнопки загрузки весов из текстового файла. |
| private void DragWindow(object sender, MouseButtonEventArgs e) | Обработчик, обрабатывающий нажатия по области перетаскивания окна, и позволяющий перетаскивать это самое окно. |

3.2.12 Модуль MainWindow.xaml.cs

Данный модуль программы содержат методы и обработчики для взаимодействия пользователя с программным продуктом и нейронной сетью.

|  |  |
| --- | --- |
| Метод или обработчик | Назначение |
| private void StartAnimation() | Метод, запускающий таймеры для анимации. |
| private void UpdateInput(object sender, EventArgs eventArgs) | Метод, запускаемый таймером, для анимации курсора. |
| private void UpdateText(object sender, EventArgs eventArgs) | Метод, запускаемый таймером, для анимации приветственного текста. |
| private void UserPressKey(object sender, KeyEventArgs e) | Обработчик, отслеживающий нажатия кнопок пользователя и записывающий символы с них в окно ввода. |
| private void SelectOption(string optionName) | Метод, вызываемый после ввода пользователем команды и открывающий соответствующее окно. |

4 Руководство пользователя

4.1 Сведения о файлах программы и способе их установки

Программа включает в себя следующий файл: «Network.exe».

Требование к ресурсам компьютера, на который устанавливается программа:

‒ ОС Windows, начиная с версии Windows 7;

‒ .NET Framework 7.0;

‒ Разрешение экрана монитора минимум 800х600, 256 цветов;

‒ Стандартная мыши и клавиатура;

‒ Жесткий диск со свободным пространством на 4 Мб;

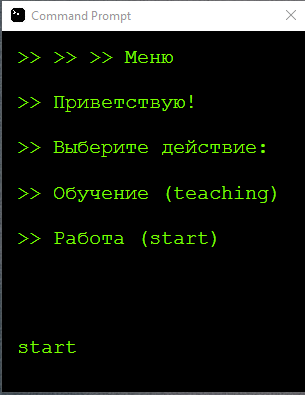
‒ Процессор с тактовой частотой от 500 МГц.

Программа устанавливается на компьютер пользователя. Порядок установки программы следующий:

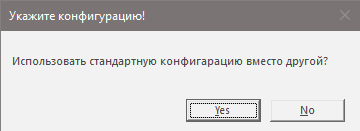
Скопировать исполнительный файл «Network.exe» на персональный компьютер и запустить его.

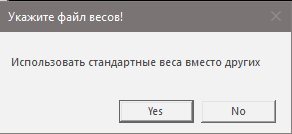
4.2 Интерфейс и функциональность программного продукта

При запуске приложения отображается стартовое окно программы, которое предлагает пользователю выбор действия над файлом (рисунок 6). Пользователю предлагается два варианта – обучение и работа с нейронной сетью.

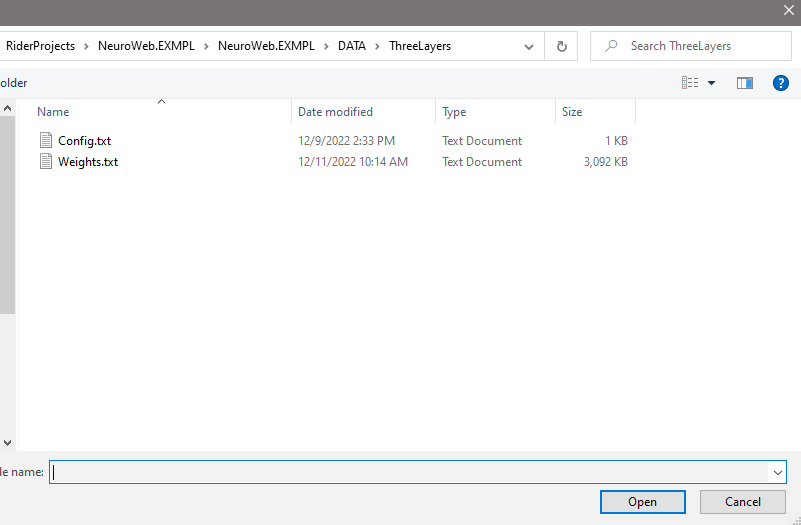
 Рис. 6. – Стартовое окно программы

После прописывания команды «start», пользователю сначала предлагают загрузить конфигурации нейронной сети и веса нейронов, или же воспользоваться встроенными в программу файлами, как показано на рисунках под номером 7 и 8.

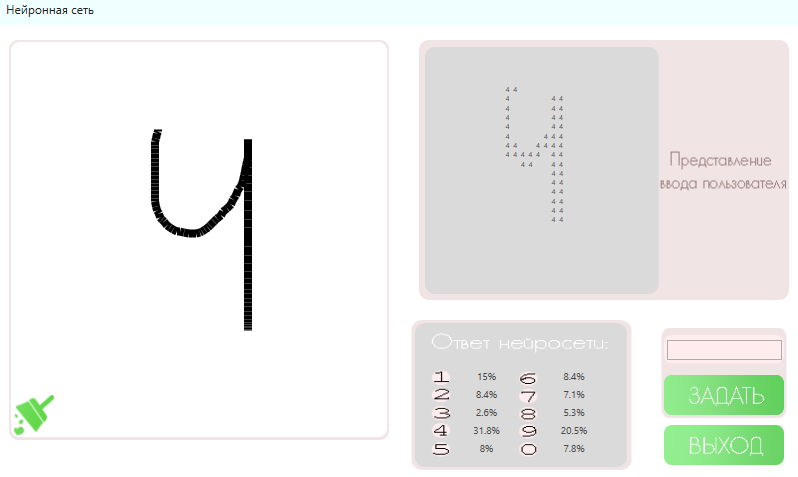
Рис. 7. – Выбор конфигурации

Рис. 8. – Выбор весов

При отрицательном ответе на любой из них перед пользователем открывается проводник, с помощью которого последний должен выбрать необходимый файл.

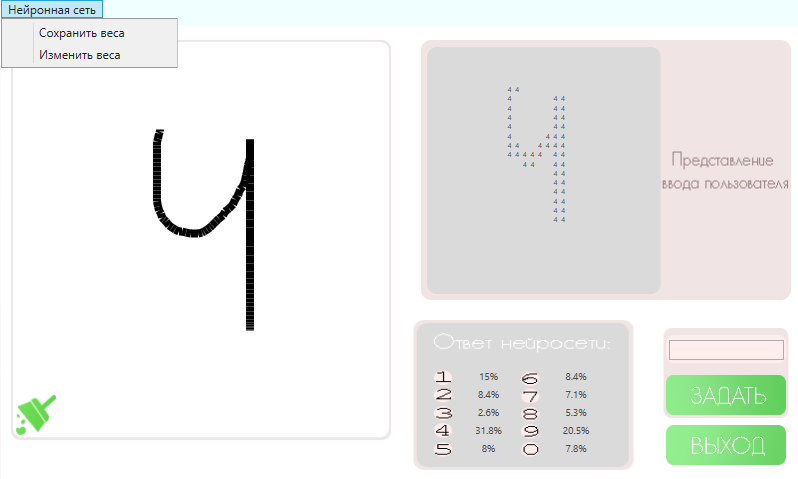
Рис. 9. – Интерфейс выбора файла

После задания всех необходимых настроек пользователь попадает в основное окно приложения где происходит основное взаимодействие с нейронной сетью.

Рис. 10. – Интерфейс главного окна

С помощью этого интерфейса, пользователь имеет возможность рукописного ввода в выделенную область слева. Справа в окне представления ввода можно увидеть область числа, которая была отправлена в нейронную сеть как входные данные, и то, из каких чисел она состоит, для удобства, является ответом нейронной сети.

Снизу имеется две панели, одна из которой является представлением всех вариантов и их вероятность. Вторая панель представляет из себя две кнопки, одна из которых – кнопка обратного обучения, которая необходима при неправильном ответе нейронной сети, и кнопка «Выход», которая закрывает программу.

 Рис. 11. – Интерфейс главного окна и

панель инструментов

На рисунке 11 продемонстрирована панель инструментов с помощью которой можно сохранить текущие веса и открыть другие.

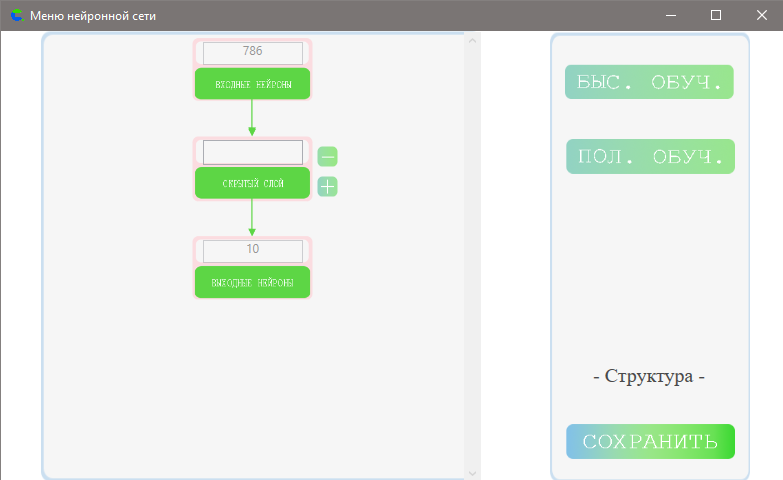
Вернёмся к стартовому меню, и теперь при использовании команды «teaching», пользователь получает доступ к окну обучения и конструктору конфигураций.

Рис. 12. – Интерфейс окна конструктора

и обучения

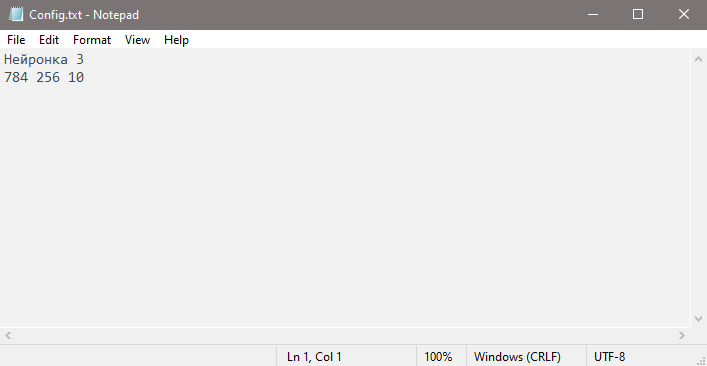
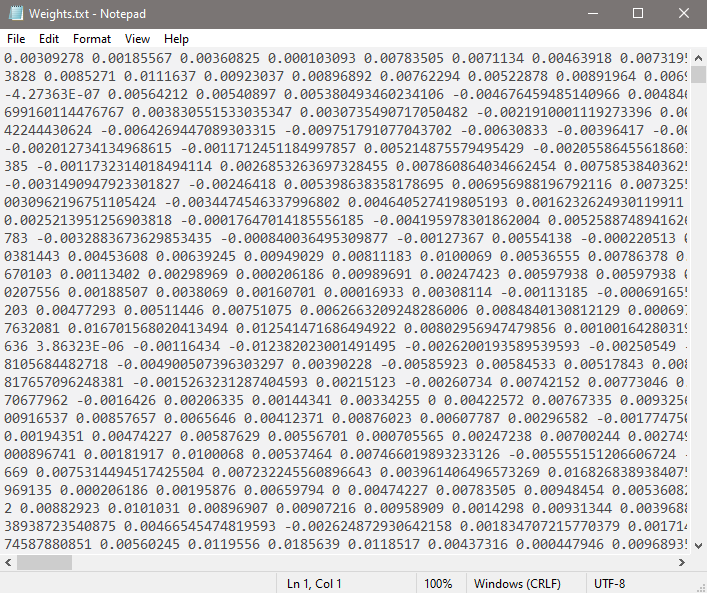
С помощью этого окна пользователь может как создать конфигурационный файл своей нейронной сети, который выглядит следующим образом и задаёт основные характеристики, такие как количество слоёв и их размерность,

Рис. 13. – Файл конфигурации нейронной сети

так и возможность загрузки конфигурации для создания соответствующих весов с помощью двух типов обучения:

1. Обучение с учителем, где пользователь с помощью стандартного интерфейса создаёт веса через функцию обратного распространения ошибки.
2. Обучение без учителя где будут задействованы два датасета данных. Один основной, взятый с сайта MNIST для обучения нейронных сетей и содержащий 60000 экземпляров чисел, и второй, так же взятый с сайта MNIST для проверки нейронной сети.

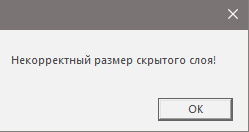
Результатом этого обучения станет файл весов, выглядящий примерно следующим образом:

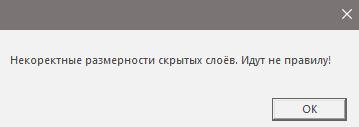
Рис. 14. – Файл весов нейронной сети

Файл весов – это строка в которой последовательно записаны все значения матриц весов нейронной сети через пробел.

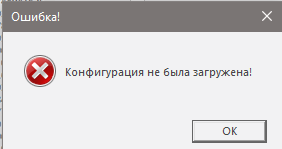
Во время работы программы, при ряде факторов, возможны сообщения о ряде ошибок и предупреждений, которые расписаны ниже:

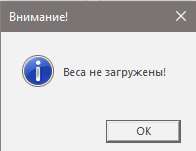
1. Предупреждение о некорректной структуре нейронной сети при попутке сохранения пользовательской структуры:

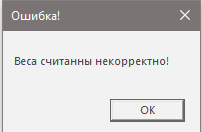
Рис. 15. – Предупреждение о некорректной записи размеров слоёв

Рис. 16. – Предупреждение о некорректной структуре нейронной сети

2) При запуске рабочего режима программы, пользователь имеет возможность выбора других конфигураций и весов нейронной сети, и при их некорректности возникают следующие ошибки:

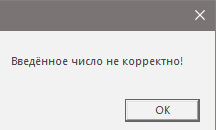
Рис. 17. – Ошибка отсутствия файла конфигурации

Рис. 18. – Ошибка отсутствия файла весов

Рис. 19. – Ошибка некорректности файла весов

для данной конфигурации

3) Во время работы программы, пользователь, пользуясь функцией обратного распространения ошибки, может указать некорректное число, или не число вовсе:

Рис. 20. – Ошибка некорректности введённого числа

Заключение

Разработанный программный продукт демонстрирует работу нейронной сети на примере определения графического ввода пользователя используя комбинацию популярных методов и функций для работы подобных алгоритмов.

В дальнейшем планируется развитие программы путём улучшения конструктора нейронной сети, оптимизацией активационной функции или полной замены последней на более подходящий вариант, каковой может стать сигмоидальная функция активации.

Данный программный продукт будет полезна, как и обычным пользователям, так и другим программистам как хороший пример работы подобных алгоритмов. Так же подобная нейронная сеть, которая предназначена для задач по классификации данных, отлично подойдёт для веб-хостинга, где сможет выполнять задачи по анализу данных пользователей.

Список использованных источников

1 THE MNIST DATABASE of handwritten digits [Электронный ресурс] // 21.12.2022 – сайт с базой данных датасетов которые были конвертированы в .txt формат. – Режим доступа: http://yann.lecun.com/exdb/mnist/

2 Функции активации нейросети: сигмоида, линейная, ступенчатая, ReLu, tahn [Электронный ресурс] // 22.12.2022 – информация о функция активации. – Режим доступа: https://neurohive.io/ru/osnovy-data-science/activation-functions/

3 Все о нейронных сетях – SberCloud [Электронный ресурс] // © 2022 Cloud, 22.12.2022 – информация об основах нейро-сетевого программирования – Режим доступа: https://sbercloud.ru/ru/services/neural-networks

4 Нейронные сети для начинающих. - информация об основах нейро-сетевого программирования [Электронный ресурс] // © 2006–2022, Habr, 22.12.2022 – Режим доступа: https://habr.com/ru/post/312450/

5 Нейронная сеть – Википедия - информация об основах нейро-сетевого программирования [Электронный ресурс] // 22.12.2022 – Режим доступа: https://ru.wikipedia.org/wiki/Нейронная\_сеть

6 Практики реализации нейронных сетей – общая информация об алгоритмах работы нейронных сетей. [Электронный ресурс] // 22.12.2022 – Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=Практики\_реализации\_нейронных\_сетей#:~:text=Функция%20активации%20англ.,ого%20входа%2C%20а%20—%20смещение

7 Активационная функция нейрона (Activation function) – общая информация о работе активационной функции в нейро-сетевом программировании. [Электронный ресурс] // 22.12.2022 – Режим доступа: https://wiki.loginom.ru/articles/activation-function.html

8 Что такое нейронная сеть? - AWS - Amazon.com – общая информация о методах обучения нейронных сетей. [Электронный ресурс] // 22.12.2022 – Режим доступа: https://aws.amazon.com/ru/what-is/neural-network/#:~:text Обучение%20нейронной%20сети%20 —%20это%20процесс,точно%20обрабатывать%20неизвестные%20входные%20данные

Приложение A

***(обязательное)***

**Текст программы**

Файл Configuration.cs

using System.IO;

using System.Windows;

using System.Windows.Controls;

using System.Collections.Generic;

using Microsoft.Win32;

namespace NeuroWeb.EXMPL.Gui {

public static class Configuration {

public static void WriteConfig(Grid configGrid, int size) {

var tempConfig = $"Нейронка {size}\n";

var tempArray = new List<int>();

foreach (var element in configGrid.Children) {

if (element.GetType() != typeof(TextBox)) continue;

var text = (element as TextBox)!.Text;

if (int.TryParse(text, out var layerSize)) {

tempConfig += $"{layerSize} ";

tempArray.Add(layerSize);

}

else {

MessageBox.Show("Некорректный размер скрытого слоя!");

tempConfig += "0 ";

tempArray.Add(0);

}

}

for (var i = 0; i < tempArray.Count - 1; i++) {

if (tempArray[i] >= tempArray[i + 1] / 3) continue;

MessageBox.Show("Некоректные размерности скрытыхte слоёв. Идут не правилу!");

break;

}

var openFile = new SaveFileDialog();

if (openFile.ShowDialog() == true) File.WriteAllText(openFile.FileName, tempConfig);

}

}

}

Файл Structure.cs

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using NeuroWeb.EXMPL.WINDOWS;

namespace NeuroWeb.EXMPL.Gui {

public static class Structure {

public static Grid GetStructure(Teacher teacher, int size) {

try {

var tempGrid = new Grid();

tempGrid.Children.Add(new Image {

Height = 100,

Width = 120,

HorizontalAlignment = HorizontalAlignment.Center,

VerticalAlignment = VerticalAlignment.Top,

Stretch = Stretch.Fill,

Margin = new Thickness(0, 5, 0, 0),

Source = new BitmapImage(new Uri("../IMAGES/TeacherWindow/Входные\_Структура.png",UriKind.Relative))

});

tempGrid.Children.Add(new TextBox {

Height = 23,

Width = 100,

HorizontalAlignment = HorizontalAlignment.Center,

VerticalAlignment = VerticalAlignment.Top,

TextAlignment = TextAlignment.Center,

Margin = new Thickness(0,10,0,0),

Background = Brushes.Transparent,

Text = "786",

IsEnabled = false

});

for (var i = 0; i < size; i++) {

tempGrid.Children.Add(new Image {

Height = 100,

Width = 120,

HorizontalAlignment = HorizontalAlignment.Center,

VerticalAlignment = VerticalAlignment.Top,

Stretch = Stretch.Fill,

Margin = new Thickness(0, 105 + 105 \* i, 0, 0),

Source = new BitmapImage(new Uri("../IMAGES/TeacherWindow/Скрытый\_Структура.png",UriKind.Relative))

});

tempGrid.Children.Add(new TextBox {

Height = 25,

Width = 100,

HorizontalAlignment = HorizontalAlignment.Center,

VerticalAlignment = VerticalAlignment.Top,

Margin = new Thickness(0, 105 + 105 \* i + 3, 0, 0),

Background = Brushes.Transparent

});

}

var img = new Image {

Height = 20,

Width = 20,

HorizontalAlignment = HorizontalAlignment.Center,

VerticalAlignment = VerticalAlignment.Top,

Stretch = Stretch.Fill,

Margin = new Thickness(150, 105 \* size + 45, 0, 0),

Source = new BitmapImage(new Uri("../IMAGES/TeacherWindow/Увеличить\_Структура.png",UriKind.Relative)),

Cursor = Cursors.Hand,

ToolTip = "Добавить скрытый слой"

};

img.MouseDown += teacher.IncreaseStructure;

tempGrid.Children.Add(img);

var img1 = new Image {

Height = 20,

Width = 20,

HorizontalAlignment = HorizontalAlignment.Center,

VerticalAlignment = VerticalAlignment.Top,

Stretch = Stretch.Fill,

Margin = new Thickness(150, 105 \* size + 15, 0, 0),

Source = new BitmapImage(new Uri("../IMAGES/TeacherWindow/Уменьшить\_Структура.png",UriKind.Relative)),

Cursor = Cursors.Hand,

ToolTip = "Убрать скрытый слой"

};

img1.MouseDown += teacher.DecreaseStructure;

tempGrid.Children.Add(img1);

tempGrid.Children.Add(new Image {

Height = 65,

Width = 120,

HorizontalAlignment = HorizontalAlignment.Center,

VerticalAlignment = VerticalAlignment.Top,

Stretch = Stretch.Fill,

Margin = new Thickness(0, 102.6 + 102.6 \* size + 3, 0, 0),

Source = new BitmapImage(new Uri("../IMAGES/TeacherWindow/Выходные\_Структура.png",UriKind.Relative))

});

tempGrid.Children.Add(new TextBox {

Height = 23,

Width = 100,

HorizontalAlignment = HorizontalAlignment.Center,

VerticalAlignment = VerticalAlignment.Top,

TextAlignment = TextAlignment.Center,

Margin = new Thickness(0,102.6 + 102.6 \* size + 8,0,0),

Background = Brushes.Transparent,

Text = "10",

IsEnabled = false

});

return tempGrid;

}

catch (Exception exception) {

MessageBox.Show($"{exception}", "Ошибка создания интерфейса!", MessageBoxButton.OK,

MessageBoxImage.Error);

}

return null;

}

}

}

Файл Matrix.cs

using System;

using System.Diagnostics.CodeAnalysis;

using System.Globalization;

namespace NeuroWeb.EXMPL.OBJECTS {

/// <summary>

/// Обьект матрицы

/// </summary>

public class Matrix {

/// <summary>

/// Конструктор матрицы, который принимает уже готовый двумерный массив

/// </summary>

/// <param name="body"> Тело матрицы, двумерный массив </param>

private Matrix(double[,] body) {

Row = body.GetLength(0);

Col = body.GetLength(1);

Body = body;

}

/// <summary>

/// Конструктор матрицы, который принимает только размерности матрицы

/// </summary>

/// <param name="row">Строки</param>

/// <param name="col">Колонки</param>

public Matrix(int row, int col) {

Row = row;

Col = col;

Body = new double[row, col];

}

private int Row { get; }

private int Col { get; }

public double[,] Body { get; }

/// <summary>

/// Метод возвращающий транспонированную матрицу

/// </summary>

/// <returns> Транспонированная матрица </returns>

public Matrix GetTranspose() {

var rows = Body.GetLength(0);

var columns = Body.GetLength(1);

var temp = new double[columns, rows];

for (var i = 0; i < temp.GetLength(0); i++) {

for (var j = 0; j < temp.GetLength(1); j++) {

temp[i, j] = Body[j, i];

}

}

return new Matrix(temp);

}

public static double[] operator \*(Matrix matrix, double[] neuron) {

if (matrix.Col != neuron.Length) throw new Exception();

var c = new double[matrix.Row];

for (var x = 0; x < matrix.Row; ++x) {

double tmp = 0;

for (var y = 0; y < matrix.Col; ++y)

tmp += matrix.Body[x, y] \* neuron[y];

c[x] = tmp;

}

return c;

}

/// <summary>

/// Метод, заполняющий матрицу случайными числами

/// </summary>

public void FillRandom() {

for (var i = 0; i < Row; i++)

for (var j = 0; j < Col; j++) {

Body[i, j] = new Random().Next() % 100 \* 0.03 / (Row + 35);

}

}

/// <summary>

/// Компанует все значения матрицы в строку

/// </summary>

/// <returns> Строка значений </returns>

[SuppressMessage("ReSharper.DPA", "DPA0000: DPA issues")]

public string GetValues() {

var tempValues = "";

for (var i = 0; i < Row; i++)

for (var j = 0; j < Col; j++)

tempValues += Body[i, j] + " ";

return tempValues;

}

public void SetValues(string value, int x, int y) {

Body[x, y] = double.Parse(value, CultureInfo.InvariantCulture);

}

}

}

Файл Network.cs

using System;

using System.IO;

using System.Linq;

using System.Windows;

using System.Globalization;

using System.Collections.Generic;

using Microsoft.Win32;

using NeuroWeb.EXMPL.SCRIPTS;

namespace NeuroWeb.EXMPL.OBJECTS {

public class Network {

public Network(Configuration configuration) {

try {

Layouts = configuration.Layout;

Neurons = new int[Layouts];

for (var i = 0; i < Layouts; i++) Neurons[i] = configuration.NeuronsLayer[i];

Configuration = configuration;

Weights = new Matrix[Layouts - 1];

Bias = new double[Layouts - 1][];

for (var i = 0; i < Layouts - 1; i++) {

Bias[i] = new double[Neurons[i + 1]];

Weights[i] = new Matrix(Neurons[i + 1], Neurons[i]);

Weights[i].FillRandom();

for (var j = 0; j < Neurons[i + 1]; j++)

Bias[i][j] = new Random().Next() % 50 \* .06 / (Neurons[i] + 15);

}

NeuronsValue = new double[Layouts][];

NeuronsError = new double[Layouts][];

for (var i = 0; i < Layouts; i++) {

NeuronsValue[i] = new double[Neurons[i]];

NeuronsError[i] = new double[Neurons[i]];

}

NeuronsBios = new double[Layouts - 1];

for (var i = 0; i < NeuronsBios.Length; i++) NeuronsBios[i] = 1;

}

catch (Exception e) {

MessageBox.Show($"{e}","Сбой инициализации сети!", MessageBoxButton.OK,

MessageBoxImage.Error);

throw;

}

}

public Configuration Configuration { get; }

private int Layouts { get; }

private int[] Neurons { get; }

private Matrix[] Weights { get; }

private double[][] Bias { get; }

public double[][] NeuronsValue { get; }

private double[][] NeuronsError { get; }

private double[] NeuronsBios { get; }

public void InsertInformation(List<double> values) {

for (var i = 0; i < values.Count; i++) NeuronsValue[0][i] = values[i];

}

private int GetMaxIndex(IReadOnlyList<double> values) {

try {

var max = values[0];

var prediction = 0;

for (var j = 1; j < Neurons[Layouts - 1]; j++) {

var temp = values[j];

if (!(temp > max)) continue;

prediction = j;

max = temp;

}

return prediction;

}

catch (Exception e) {

MessageBox.Show($"{e}","Сбой получения максимального индекса!", MessageBoxButton.OK,

MessageBoxImage.Error);

throw;

}

}

public double ForwardFeed() {

try {

for (var k = 1; k < Layouts; ++k) {

NeuronsValue[k] = new Vector(Weights[k - 1] \* NeuronsValue[k - 1]) + new Vector(Bias[k - 1]);

NeuronsValue[k] = NeuronActivate.Activation(NeuronsValue[k]);

}

return GetMaxIndex(NeuronsValue[Layouts - 1]);

}

catch (Exception e) {

MessageBox.Show($"{e}","Сбой активации нейронов!", MessageBoxButton.OK,

MessageBoxImage.Error);

throw;

}

}

public void BackPropagation(double expectedAnswer) {

try {

for (var i = 0; i < Neurons[Layouts - 1]; i++)

if (i != (int)expectedAnswer)

NeuronsError[Layouts - 1][i] = -NeuronsValue[Layouts - 1][i] \*

NeuronActivate.GetDerivative(NeuronsValue[Layouts - 1][i]);

else NeuronsError[Layouts - 1][i] = (1.0 - NeuronsValue[Layouts - 1][i]) \*

NeuronActivate.GetDerivative(NeuronsValue[Layouts - 1][i]);

for (var i = Layouts - 2; i > 0; i--) {

NeuronsError[i] = Weights[i].GetTranspose() \* NeuronsError[i + 1];

for (var j = 0; j < Neurons[i]; j++)

NeuronsError[i][j] \*= NeuronActivate.GetDerivative(NeuronsValue[i][j]);

}

}

catch (Exception e) {

MessageBox.Show($"{e}","Сбой обратного обучения!", MessageBoxButton.OK,

MessageBoxImage.Error);

throw;

}

}

public void SetWeights(double learningRange) {

for (var i = 0; i < Layouts - 1; ++i)

for (var j = 0; j < Neurons[i + 1]; ++j)

for (var k = 0; k < Neurons[i]; ++k)

Weights[i].Body[j, k] += NeuronsValue[i][k] \* NeuronsError[i + 1][j] \* learningRange;

for (var i = 0; i < Layouts - 1; i++)

for (var j = 0; j < Neurons[i + 1]; j++)

Bias[i][j] += NeuronsError[i + 1][j] \* learningRange;

}

private static string \_weights;

private static string GetWeights() {

var defaultWeights = Properties.Resources.defaultWeights;

var file = new OpenFileDialog();

var message = MessageBox.Show("Использовать стандартные веса вместо " +

"других", "Укажите файл весов!", MessageBoxButton.YesNo);

if (message == MessageBoxResult.Yes) return defaultWeights;

\_weights = file.FileName;

return file.ShowDialog() != true ? "" : File.ReadAllText(file.FileName);

}

public void SaveWeights() {

try {

MessageBox.Show("Начата запись весов!");

var temp = Weights.Aggregate("", (current, weight) => current + weight.GetValues());

for (var i = 0; i < Layouts - 1; i++)

for (var j = 0; j < Neurons[i + 1]; ++j)

temp += Bias[i][j] + " ";

if (File.Exists(\_weights)) File.WriteAllText(\_weights, temp);

else {

var file = new SaveFileDialog();

MessageBox.Show("Укажите место для сохранения весов!");

if (file.ShowDialog() == true) File.WriteAllText(file.FileName, temp);

return;

}

MessageBox.Show("Веса обновлены!");

}

catch (Exception e) {

MessageBox.Show($"{e}","Сбой при записи весов!", MessageBoxButton.OK,

MessageBoxImage.Error);

throw;

}

}

public void ReadWeights() {

try {

var tempValues = GetWeights().Split(" ",

StringSplitOptions.RemoveEmptyEntries);

if (tempValues.Length < 10) return;

var position = 0;

for (var l = 0; l < Layouts - 1; l++)

for (var i = 0; i < Weights[l].Body.GetLength(0); i++)

for (var j = 0; j < Weights[l].Body.GetLength(1); j++)

Weights[l].SetValues(tempValues[position++], i, j);

for (var l = 0; l < Layouts - 1; l++)

for (var i = 0; i < Neurons[l + 1]; i++)

Bias[l][i] = double.Parse(tempValues[position++], CultureInfo.InvariantCulture);

if (position < tempValues.Length) MessageBox.Show("Веса считанны некорректно!", "Ошибка!");

}

catch (Exception e) {

MessageBox.Show($"{e}","Сбой при чтении весов!", MessageBoxButton.OK,

MessageBoxImage.Error);

throw;

}

}

}

public struct Configuration {

public int Layout;

public int[] NeuronsLayer;

}

}

Файл Number.cs

using System.Collections.Generic;

namespace NeuroWeb.EXMPL.OBJECTS {

public class Number {

public Number() {

Pixels = new List<double>();

}

public List<double> Pixels { get; set; }

public int Digit { get; set; }

public string PrintNumber()

{

var temp = "";

var position = 0;

for (var i = 0; i < 28; i++) {

for (var j = 0; j < 28; j++) {

temp += Pixels[position++] + " ";

}

temp += "\n";

}

return temp;

}

}

}

Файл Vector.cs

namespace NeuroWeb.EXMPL.OBJECTS {

public class Vector {

public Vector(double[] array) {

Body = array;

Size = array.Length;

}

private double[] Body { get; }

private int Size { get; }

private double this[int key] {

get => Body[key];

set => SetElement(key, value);

}

private void SetElement(int index, double value) {

Body[index] = value;

}

public static double[] operator +(Vector vector1, Vector vector2) {

for (var i = 0; i < vector1.Size; i++) {

vector1[i] += vector2[i];

}

return vector1.Body;

}

}

}

Файл DataWorker.cs

using System;

using System.Windows;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using NeuroWeb.EXMPL.OBJECTS;

namespace NeuroWeb.EXMPL.SCRIPTS {

public static class DataWorker {

public static Configuration ReadNetworkConfig(string config) {

try {

var data = new Configuration();

var tempData = config.Split(new[] {' ', '\n'},

StringSplitOptions.RemoveEmptyEntries);

for (var i = 0; i < tempData.Length; i++) {

if (tempData[i] != "Нейронка") continue;

var layouts = int.Parse(tempData[i + 1]);

data.Layout = layouts;

data.NeuronsLayer = new int[layouts];

for (var j = 1; j < layouts + 1; j++)

data.NeuronsLayer[j - 1] = int.Parse(tempData[i + 1 + j]);

break;

}

return data;

}

catch (Exception e) {

MessageBox.Show($"{e}", "Ошибка считывания конфиг. файла",MessageBoxButton.OK,

MessageBoxImage.Error);

throw;

}

}

[SuppressMessage("ReSharper.DPA", "DPA0000: DPA issues")]

public static Number ReadData(string pixelsValue, Configuration configuration) {

try {

var number = new Number();

for (var i = 0; i < configuration.NeuronsLayer[0]; i++) number.Pixels.Add(0);

var pixels = pixelsValue.Split(" ", StringSplitOptions.RemoveEmptyEntries);

for (var j = 0; j < configuration.NeuronsLayer[0]; j++)

if (double.TryParse(pixels[j], out var db)) number.Pixels[j] = db;

else number.Pixels[j] = 0d;

return number;

}

catch (Exception e) {

MessageBox.Show($"{e}","Ошибка создания обьекта числа", MessageBoxButton.OK,

MessageBoxImage.Error);

throw;

}

}

public static List<Number> ReadData(string config, Configuration configuration, ref int examples) {

try {

var numbers = new List<Number>();

var tempValues = config.Split(new[] {' ', '\n'},

StringSplitOptions.RemoveEmptyEntries);

var position = 0;

if (tempValues[position++] != "Examples") return numbers;

examples = int.Parse(tempValues[position++]);

for (var i = 0; i < examples; i++)

numbers.Add(new Number());

for (var i = 0; i < examples; i++)

for (var j = 0; j < configuration.NeuronsLayer[0]; j++)

numbers[i].Pixels.Add(0);

for (var i = 0; i < examples; i++) {

numbers[i].Digit = int.TryParse(tempValues[position++], out var it) ? it : 0;

for (var j = 0; j < configuration.NeuronsLayer[0]; j++)

if (double.TryParse(tempValues[position++], out var db)) numbers[i].Pixels[j] = db;

else numbers[i].Pixels[j] = 0d;

}

return numbers;

}

catch (Exception e) {

MessageBox.Show($"{e}","Ошибка создания обьекта числа", MessageBoxButton.OK,

MessageBoxImage.Error);

throw;

}

}

}

}

Файл NeuronActivate.cs

namespace NeuroWeb.EXMPL.SCRIPTS {

public static class NeuronActivate {

public static double[] Activation(double[] value) {

for (var i = 0; i < value.Length; i++)

switch (value[i]) {

case < 0:

value[i] \*= 0.01d;

break;

case > 1:

value[i] = 1d + .01d \* (value[i] - 1d);

break;

}

return value;

}

public static double GetDerivative(double value) => value is < 0 or > 1 ? 0.01d : value;

}

}

Файл Prediction.cs

using NeuroWeb.EXMPL.OBJECTS;

namespace NeuroWeb.EXMPL.SCRIPTS {

public static class Prediction {

public static int Predict(Network network, string number) {

var dataInformation = DataWorker.ReadData(number, network.Configuration);

network.InsertInformation(dataInformation.Pixels);

return (int)network.ForwardFeed();

}

}

}

Файл Teaching.cs

using System;

using System.Windows;

using System.Diagnostics.CodeAnalysis;

using Microsoft.Win32;

using NeuroWeb.EXMPL.OBJECTS;

namespace NeuroWeb.EXMPL.SCRIPTS {

public static class Teaching {

public static void LightStudying(Network network, string number, int expected) {

try {

var dataInformation = DataWorker.ReadData(number, network.Configuration);

network.InsertInformation(dataInformation.Pixels);

var prediction = network.ForwardFeed();

if (expected.Equals((int)prediction)) return;

network.BackPropagation(expected);

network.SetWeights(.08);

}

catch (Exception e) {

MessageBox.Show($"{e}", "Ошибка при обучении!", MessageBoxButton.OK,

MessageBoxImage.Error);

throw;

}

}

[SuppressMessage("ReSharper.DPA", "DPA0000: DPA issues")]

public static void HardStudying(Network network, int teachingCounts) {

try {

double rightAnswersCount = 0d, maxRightAnswers = 0d;

var era = 0;

var examples = 0;

MessageBox.Show("Укажите файл обучения!");

var file = new OpenFileDialog();

if (file.ShowDialog() != true) return;

var dataInformation = DataWorker.ReadData(file.FileName, network.Configuration, ref examples);

MessageBox.Show($"Загруженно приверов: {examples}\n");

while (rightAnswersCount / examples \* 100 < 100) {

rightAnswersCount = 0;

for (var i = 0; i < examples; ++i) {

network.InsertInformation(dataInformation[i].Pixels);

double right = dataInformation[i].Digit;

var prediction = network.ForwardFeed();

if (!prediction.Equals(right)) {

network.BackPropagation(right);

network.SetWeights(.15d \* Math.Exp(-era / 20d));

}

else rightAnswersCount++;

}

if (rightAnswersCount > maxRightAnswers) maxRightAnswers = rightAnswersCount;

MessageBox.Show($"Правильно: {Math.Round(rightAnswersCount / examples \* 100, 3)}%\n" +

$"Максимум правильных: {Math.Round(maxRightAnswers / examples \* 100, 3)}%\n" +

$"Цикл обучения №{era}");

if (++era == teachingCounts) break;

}

network.SaveWeights();

}

catch (Exception e) {

MessageBox.Show($"{e}", "Ошибка при глубоком обучении!", MessageBoxButton.OK,

MessageBoxImage.Error);

throw;

}

}

}

}

Файл TeacherWindow.xaml.cs

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using Microsoft.Win32;

using NeuroWeb.EXMPL.Gui;

using NeuroWeb.EXMPL.OBJECTS;

using NeuroWeb.EXMPL.SCRIPTS;

using Configuration = NeuroWeb.EXMPL.Gui.Configuration;

namespace NeuroWeb.EXMPL.WINDOWS {

public partial class Teacher {

public Teacher() => InitializeComponent();

private int \_size = 1;

public void DecreaseStructure(object sender, MouseButtonEventArgs e) {

if (\_size <= 1) return;

NetworkStructure.Content = Structure.GetStructure(this, --\_size);

}

public void IncreaseStructure(object sender, MouseButtonEventArgs e) =>

NetworkStructure.Content = Structure.GetStructure(this, ++\_size);

private void CloseTeacher(object sender, MouseButtonEventArgs e) {

Configuration.WriteConfig(NetworkStructure.Content as Grid, \_size + 2);

Close();

}

private void FastTeaching(object sender, MouseButtonEventArgs e) {

new User().Show();

Close();

}

private void HardTeaching(object sender, MouseButtonEventArgs e) {

MessageBox.Show("Укажите файл конфигурации сети!");

var file = new OpenFileDialog();

if (file.ShowDialog() == true)

Teaching.HardStudying(new Network(DataWorker.ReadNetworkConfig(file.FileName)), 20);

}

}

}

Файл UserWindow.xaml.cs

using System;

using System.IO;

using System.Windows;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Shapes;

using System.Windows.Controls;

using System.Windows.Threading;

using System.Collections.Generic;

using System.Windows.Media.Imaging;

using System.Diagnostics.CodeAnalysis;

using Microsoft.Win32;

using NeuroWeb.EXMPL.OBJECTS;

using NeuroWeb.EXMPL.SCRIPTS;

namespace NeuroWeb.EXMPL.WINDOWS {

public partial class User {

public User() {

var defaultConfig = Properties.Resources.defaultConfig;

if (MessageBox.Show("Использовать стандартную конфигарацию вместо другой?",

"Укажите конфигурацию!", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

Network = new Network(DataWorker.ReadNetworkConfig(defaultConfig));

else {

var file = new OpenFileDialog();

if (file.ShowDialog() == true)

Network = new Network(DataWorker.ReadNetworkConfig(File.ReadAllText(file.FileName)));

}

InitializeComponent();

Answers = new List<Label> {

Zero, One, Two, Three, Four, Five, Six, Seven, Eight, Nine

};

Network.ReadWeights();

Update = new DispatcherTimer {

Interval = new TimeSpan(0,0,0,1)

};

Update.Tick += AnalyzeUserInput;

Update.IsEnabled = true;

}

private Network Network { get; }

private DispatcherTimer Update { get; }

private string Number { get; set; }

private List<Label> Answers { get; }

private readonly Brush \_userBrush = Brushes.Black;

private int \_pred = 1;

[SuppressMessage("ReSharper.DPA", "DPA0000: DPA issues")]

private void AnalyzeUserInput(object sender, EventArgs eventArgs) {

var renderTargetBitmap = new RenderTargetBitmap(28,28, 6.5d, 6.5d,

PixelFormats.Pbgra32);

renderTargetBitmap.Render(UserCanvas);

var writeableBitmap = new WriteableBitmap(renderTargetBitmap);

var matrix = new double[28,28];

var temp = "";

var numberValue = "";

for (var i = 0; i < 28; i++) {

for (var j = 0; j < 28; j++) {

matrix[i,j] = writeableBitmap.GetPixel(j, i).A / 255d;

if (matrix[i, j] > 0) temp += \_pred + " ";

else temp += " " + " ";

numberValue += matrix[i, j]+ " ";

}

temp += "\n";

}

for (var i = 0; i < Answers.Count; i++) Answers[i].Content =

$"{Math.Abs(Math.Round(Network.NeuronsValue[2][i] \* 100, 1))}%";

Matrix.Content = temp;

Number = numberValue;

\_pred = Prediction.Predict(Network, numberValue);

}

private Point \_currentPoint;

private void UserMoveMouse(object sender, MouseEventArgs e) {

AnalyzeUserInput(null, null);

if (e.LeftButton != MouseButtonState.Pressed) return;

var line = new Line {

Stroke = \_userBrush,

X1 = \_currentPoint.X,

Y1 = \_currentPoint.Y,

X2 = e.GetPosition(UserCanvas).X,

Y2 = e.GetPosition(UserCanvas).Y,

StrokeThickness = 8

};

\_currentPoint = e.GetPosition(UserCanvas);

UserCanvas.Children.Add(line);

}

private void UserClick(object sender, MouseButtonEventArgs e) {

if (e.ButtonState == MouseButtonState.Pressed)

\_currentPoint = e.GetPosition(UserCanvas);

}

private void BackPropagation(object sender, RoutedEventArgs e) {

Update.IsEnabled = false;

if (int.TryParse(ExpectedAnswer.Text, out var number)) {

ExpectedAnswer.Text = "";

Teaching.LightStudying(Network, Number, number);

}

else MessageBox.Show("Введённое число не корректно!");

Update.IsEnabled = true;

}

private void Clear(object sender, RoutedEventArgs e) => UserCanvas.Children.Clear();

private void Exit(object sender, RoutedEventArgs e) => Close();

private void SaveWeights(object sender, RoutedEventArgs e) {

MessageBox.Show("Сохранение начато...");

Network.SaveWeights();

}

private void LoadWeights(object sender, RoutedEventArgs e) => Network.ReadWeights();

private void DragWindow(object sender, MouseButtonEventArgs e) {

base.OnMouseLeftButtonDown(e);

DragMove();

}

}

}

Файл MainWindow.xaml.cs

using System;

using System.Windows.Input;

using System.Windows.Threading;

using NeuroWeb.EXMPL.WINDOWS;

namespace NeuroWeb.EXMPL {

public partial class MainWindow

{

public MainWindow() {

InitializeComponent();

StartAnimation();

}

private DispatcherTimer TextUpdate { get; set; }

private DispatcherTimer UserInputAnimation { get; set; }

private const string Text = ">> >> >> Меню\n\n>> Приветствую!\n\n>> Выберите действие:\n\n" +

">> Обучение (teaching)\n\n>> Работа (start)";

private int \_position;

private void StartAnimation() {

TextUpdate = new DispatcherTimer {

Interval = new TimeSpan(900000)

};

TextUpdate.Tick += UpdateText;

TextUpdate.IsEnabled = true;

UserInputAnimation = new DispatcherTimer {

Interval = new TimeSpan(0,0,0,1)

};

UserInputAnimation.Tick += UpdateInput;

UserInputAnimation.IsEnabled = true;

}

private void UpdateInput(object sender, EventArgs eventArgs) {

if (UserInput.Content.ToString()!.Contains("|")) UserInput.Content =

UserInput.Content.ToString()!.Replace("|", "");

else UserInput.Content += "|";

}

private void UpdateText(object sender, EventArgs eventArgs) {

if (\_position >= Text.Length) {

TextUpdate.IsEnabled = false;

return;

}

MainText.Content += Text[\_position++].ToString();

}

private void UserPressKey(object sender, KeyEventArgs e) {

if (UserInput.Content.ToString()!.Contains("|")) UserInput.Content =

UserInput.Content.ToString()!.Replace("|", "");

var temp = e.Key.ToString().ToLower();

switch (temp) {

case "return":

SelectOption(UserInput.Content.ToString());

return;

case "back":

var str = UserInput.Content.ToString();

if (str!.Length <= 0) return;

UserInput.Content = UserInput.Content.ToString()!.Remove(str!.Length - 1);

return;

}

if (UserInput.Content.ToString()!.Length < 20) UserInput.Content += temp;

else UserInput.Content = "";

}

private void SelectOption(string optionName) {

switch (optionName) {

case "teaching":

new Teacher().Show();

break;

case "start":

new User().Show();

break;

default:

return;

}

Close();

}

}

}

Файл TeacherWindow.xaml

<Window x:Class="NeuroWeb.EXMPL.WINDOWS.Teacher"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

mc:Ignorable="d"

Title="Меню нейронной сети" Height="490" Width="800"

Background="White"

Icon="../IMAGES/Icons/Иконка.png">

<Grid>

<Image Source="../IMAGES/TeacherWindow/Фон\_Структура.png"

Stretch="Fill"

Height="450"

Width="440"

HorizontalAlignment="Left"

Margin="40,0,0,0"></Image>

<ScrollViewer x:Name="NetworkStructure"

Height="450"

Width="440"

HorizontalAlignment="Left"

Margin="40,0,0,0">

<Grid>

<Image Source="../IMAGES/TeacherWindow/Входные\_Структура.png"

Stretch="Fill"

Height="100"

Width="120"

HorizontalAlignment="Center"

VerticalAlignment="Top"

Margin="0,5,0,0"></Image>

<TextBox Height="23"

Width="100"

HorizontalAlignment="Center"

VerticalAlignment="Top"

Margin="0,10,0,0"

Background="Transparent"

Text="786"

IsEnabled="False"

TextAlignment="Center"></TextBox>

<Image Source="../IMAGES/TeacherWindow/Скрытый\_Структура.png"

Stretch="Fill"

Height="100"

Width="120"

HorizontalAlignment="Center"

VerticalAlignment="Top"

Margin="0,105,0,0"></Image>

<TextBox Height="25"

Width="100"

HorizontalAlignment="Center"

VerticalAlignment="Top"

Margin="0,108,0,0"

Background="Transparent"

TextAlignment="Center"></TextBox>

<Image Source="../IMAGES/TeacherWindow/Увеличить\_Структура.png"

Stretch="Fill"

Height="20"

Width="20"

HorizontalAlignment="Center"

VerticalAlignment="Top"

Margin="150,145,0,0"

Cursor="Hand"

ToolTip="Добавить скрытый слой"

MouseDown="IncreaseStructure"></Image>

<Image Source="../IMAGES/TeacherWindow/Уменьшить\_Структура.png"

Stretch="Fill"

Height="20"

Width="20"

HorizontalAlignment="Center"

VerticalAlignment="Top"

Margin="150,115,0,0"

Cursor="Hand"

ToolTip="Убрать скрытый слой"

MouseDown="DecreaseStructure"></Image>

<Image Source="../IMAGES/TeacherWindow/Выходные\_Структура.png"

Stretch="Fill"

Height="65"

Width="120"

HorizontalAlignment="Center"

VerticalAlignment="Top"

Margin="0,203,0,0"></Image>

<TextBox Height="23"

Width="100"

HorizontalAlignment="Center"

VerticalAlignment="Top"

Margin="0,208,0,0"

Background="Transparent"

Text="10"

IsEnabled="False"

TextAlignment="Center"></TextBox>

</Grid>

</ScrollViewer>

<Image Source="../IMAGES/TeacherWindow/Фон\_Управление.png"

Stretch="Fill"

Height="450"

Width="200"

HorizontalAlignment="Right"

Margin="0,0,35,0"></Image>

<Image Source="../IMAGES/TeacherWindow/БысОбуч\_Управление.png"

Stretch="Fill"

Height="35"

Width="170"

HorizontalAlignment="Right"

Margin="0,0,50,350"

Cursor="Hand"

ToolTip="Начать обучение нейронной сети без стороннего файла"

MouseDown="FastTeaching"></Image>

<Image Source="../IMAGES/TeacherWindow/ПолноеОбучение\_Управление.png"

Stretch="Fill"

Height="35"

Width="170"

HorizontalAlignment="Right"

Margin="0,0,50,200"

Cursor="Hand"

ToolTip="Загрузить файл и начать полное обучение"

MouseDown="HardTeaching"></Image>

<Image Source="../IMAGES/TeacherWindow/Кнопка\_Закрыть.png"

Stretch="Fill"

Height="35"

Width="170"

HorizontalAlignment="Right"

Margin="0,370,50,0"

Cursor="Hand"

ToolTip="Закрыть и сохранить конфигурацию"

MouseDown="CloseTeacher"></Image>

</Grid>

</Window>

Файл UserWindow.xaml

<Window x:Class="NeuroWeb.EXMPL.WINDOWS.User"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

mc:Ignorable="d"

Title="Нейронная сеть" Height="480" Width="800"

ResizeMode="NoResize"

Background="White"

WindowStyle="None"

Icon="../IMAGES/Icons/Иконка.png">

<Grid>

<Image Source="../IMAGES/UserWindow/Холст\_Рисования.png"

Stretch="Fill"

Height="400"

Width="380"

HorizontalAlignment="Left"

Margin="10,0,0,0">

</Image>

<Image Source="../IMAGES/UserWindow/Окно\_представления.png"

HorizontalAlignment="Right"

Stretch="Fill"

Height="260"

Width="370"

Margin="0,0,10,140">

</Image>

<Image Source="../IMAGES/UserWindow/Окно\_ответ.png"

Stretch="Fill"

Height="150"

Width="220"

HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Margin="245,0,0,10"></Image>

<Canvas x:Name = "UserCanvas"

MouseMove="UserMoveMouse"

MouseDown="UserClick"

Height="400"

Width="380"

Margin="0,0,410,0">

<Canvas.Background>

<SolidColorBrush Color="Transparent" Opacity="0"/>

</Canvas.Background>

</Canvas>

<Grid Width="385"

Height="450"

HorizontalAlignment="Right"

VerticalAlignment="Top">

<Label HorizontalAlignment="Left"

VerticalAlignment="Top"

Margin="10,5,0,0"

x:Name="Matrix"

FontSize="7">

</Label>

<Label HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Margin="0,0,240,62"

x:Name="One"

FontSize="10">

100%

</Label>

<Label HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Margin="0,0,240,44"

x:Name="Two"

FontSize="10">

100%

</Label>

<Label HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Margin="0,0,240,26"

x:Name="Three"

FontSize="10">

100%

</Label>

<Label HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Margin="0,0,240,8"

x:Name="Four"

FontSize="10">

100%

</Label>

<Label HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Margin="0,0,240,-10"

x:Name="Five"

FontSize="10">

100%

</Label>

<Label HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Margin="0,0,65,62"

x:Name="Six"

FontSize="10">

100%

</Label>

<Label HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Margin="0,0,65,44"

x:Name="Seven"

FontSize="10">

100%

</Label>

<Label HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Margin="0,0,65,26"

x:Name="Eight"

FontSize="10">

100%

</Label>

<Label HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Margin="0,0,65,8"

x:Name="Nine"

FontSize="10">

100%

</Label>

<Label HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Margin="0,0,65,-10"

x:Name="Zero"

FontSize="10">

100%

</Label>

</Grid>

<Image Source="../IMAGES/UserWindow/Фон\_ответа.png"

Stretch="Fill"

Height="90"

Width="125"

HorizontalAlignment="Right"

VerticalAlignment="Bottom"

Margin="0,0,12.5,62"></Image>

<Image Source="../IMAGES/UserWindow/Кнопка\_Задать.png"

Stretch="Fill"

Height="40"

Width="120"

HorizontalAlignment="Right"

VerticalAlignment="Bottom"

Margin="0,0,15,65"

MouseDown="BackPropagation"

Cursor="Hand"

ToolTip="Задать нужный ответ"></Image>

<Image Source="../IMAGES/UserWindow/Поле\_Ответа.png"

Stretch="Fill"

Height="30"

Width="120"

HorizontalAlignment="Right"

VerticalAlignment="Bottom"

Margin="0,0,15,115"></Image>

<Image Source="../IMAGES/UserWindow/Кнопка\_Выход.png"

Stretch="Fill"

Height="40"

Width="120"

HorizontalAlignment="Right"

VerticalAlignment="Bottom"

Margin="0,0,15,15"

MouseDown="Exit"

Cursor="Hand"

ToolTip="Выйти"></Image>

<TextBox Height="20"

Width="115"

x:Name="ExpectedAnswer"

HorizontalAlignment="Right"

VerticalAlignment="Bottom"

Margin="0,0,17,120"

Background="Transparent"

TextAlignment="Center"></TextBox>

<Image Source="../IMAGES/UserWindow/Кнопка\_Очистка.png"

Stretch="Fill"

Height="40"

Width="40"

HorizontalAlignment="Left"

VerticalAlignment="Bottom"

Margin="15,0,0,45"

MouseDown="Clear"

Cursor="Hand"

ToolTip="Очистка холста"></Image>

<Menu Height="25"

Background="Azure"

VerticalAlignment="Top">

<MenuItem Header="Нейронная сеть">

<MenuItem Click="SaveWeights"

Header="Сохранить веса">

</MenuItem>

<MenuItem Click="LoadWeights"

Header="Изменить веса">

</MenuItem>

</MenuItem>

</Menu>

<Canvas Height="25"

Width="400"

MouseLeftButtonDown="DragWindow"

VerticalAlignment="Top"

Background="Azure"></Canvas>

</Grid>

</Window>

Файл MainWindow.xaml

<Window x:Class="NeuroWeb.EXMPL.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

Icon="IMAGES/Icons/command-line-icon-1.png"

Title="Command Prompt" Height="400" Width="320"

ResizeMode="NoResize"

KeyDown="UserPressKey">

<Grid>

<Canvas Background="Black"></Canvas>

<Label x:Name="MainText"

FontSize="20"

FontFamily="Courier New"

HorizontalAlignment="Left"

VerticalAlignment="Top"

Margin="10,10,0,0"

Foreground="LawnGreen">

</Label>

<Label x:Name="UserInput"

FontSize="20"

FontFamily="Courier New"

HorizontalAlignment="Left"

VerticalAlignment="Top"

Margin="10,300,0,0"

Foreground="LawnGreen">

|

</Label>

</Grid>

</Window>