**Городской округ Краснотурьинск**

**Муниципальное автономное образовательное учреждение**

**«Средняя общеобразовательная школа № 24»**

**Направление: «Наука в цифрах»**

**Секция:**

**Проект**

**Искусственный интеллект. Модель, проектирование, разработка.**

**Автор проекта:**

***Самонин Евгений Игоревич****, 10 класс*

**Руководитель проекта:** *Ф.И.О., должность (с кем работаете)*

*телефон, эл. почта*

**2025**

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc190794883)

[1. ТЕОРЕТИЧСЕКИЕ ОСНОВЫ ПРОЕКТА 4](#_Toc190794884)

[1.1 История создания нейронных сетей 4](#_Toc190794885)

[1.2 Современные нейронные сети 5](#_Toc190794886)

[1.3 Понятие нейронной сети в современном мире 7](#_Toc190794887)

[1.4 Основные выполняемые нейросетями функции 8](#_Toc190794888)

[1.5 Ключевые для понимания определения 8](#_Toc190794889)

[1.6 Алгоритм работы 9](#_Toc190794890)

[1.6.1 слои перцептрона 9](#_Toc190794891)

[1.6.2 Вычисление предсказания нейронной сети 9](#_Toc190794892)

[1.6.3 Обучение нейронной сети 11](#_Toc190794893)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 15](#_Toc190794894)

[2.1 Код модели нейронной сети 15](#_Toc190794895)

[2.2 Обучение нейронной сети 17](#_Toc190794896)

[2.2.1 Эффективность работы 17](#_Toc190794897)

[2.2.2 Скорость работы 19](#_Toc190794898)

[ЗАКЛЮЧЕНИЕ 20](#_Toc190794899)

[БИБЛИОГРАФИЯ 22](#_Toc190794900)

[ПРИЛОЖЕНИЕ 23](#_Toc190794901)

**ВВЕДЕНИЕ**

**Технологии искусственного интеллекта призваны стать важнейшим ресурсом достижения национальных целей развития**

Президент РФ Путин В.В. [12]

Проектирование и разработка искусственного интеллекта в современности — одна из наиболее быстро развивающихся и перспективных областей как бизнеса, так и науки. В реалиях скачка компьютерных технологий и глобализации, ИИ становится универсальным помощником для достижения преимуществ перед конкурентами и инноваций в различных направлениях экономики.

**Актуальность** проектной работы, определяется тем, что искусственный интеллект используется во многих сферах нашей жизни и позволяет увеличить продуктивность, как в творческих задачах, так и в задачах требующих обработки большого количества данных. В образовательной сфере ИИ открывает новые возможности для личного обучения и улучшения образовательных программ.

Работая над этим проектом, я поставил перед собой **цель:** создать модель нейронной сети.

Для достижения этой цели, требуется решить следующие **задачи**:

* Изучить историю создания искусственного интеллекта.
* Проанализировать современные нейронные сети.
* Рассмотреть алгоритм работы полносвязного перцептрона.
* Создать свою модель нейронной сети.

**Объект:** Искусственный интеллект.

**Предмет:** Модель нейронной сети.

**Гипотеза:** Использование искусственных нейронных сетей позволяет автоматизировать задачи классификации данных, что повышает точность и скорость обработки информации по сравнению с ручными методами.

Для решения этих задач в ходе работы применены **методы исследования**:

* Информационно-поисковый (теоретический) метод: анализ источников.
* Практический метод: моделирование, эксперимент.
  1. **ТЕОРЕТИЧСЕКИЕ ОСНОВЫ ПРОЕКТА**
  2. **История создания нейронных сетей**

Изначально люди пытались воссоздать работу мозга человека с помощью компьютера и для этого выдвинули модели нейронов и связей между ними:

В 1943-м Уорреном Мак-Каллоком и Уолтером Питтсом была предложена математическая модель нейрона, а уже в конце 1950-х Фрэнк Розенблатт представил персептрон – простую модель машинного обучения, созданную для помощи компьютерам в обучении на разных объёмах данных. Именно его можно назвать первой практической реализацией нейросети. [4]

Именно для этого сначала использовать бинарная система активации нейрона. Либо нейрон активирован, либо нет. После этого многие поняли ограниченность нейронных сетей и настала «Зима искусственного интеллекта». Но позже с ростом сложности алгоритмов нейронных сетей, а также вычислительных возможностей компьютеров интерес появился вновь. В это время начали появляться способы «глубокого» обучения сетей, например обратное распространение ошибки, именно этот способ рассмотрен в этом проекте.

В период 1980-2000 стали разрабатываться и появляться первые алгоритмы обучения, сравнения и анализа данных, в XXI направление стало стремительно развиваться. В 2000-х появились мощные графические процессоры, и стали доступны большие объёмы данных, что побудило сообщество начать разработку алгоритмов Deep Learning – совокупность методов машинного обучения, основанных на обучении представлениям, а не на специализированных алгоритмах под конкретные задачи. [4]

В современном времени бум популярности нейронных сетей пришел с выходом Chat-GPT 3, который был способен осознанно отвечать на вопросы пользователя. После чего каждая крупная компания стала ставить перед собой цель конкурирования на рынке искусственного интеллекта, благодаря этому нейронные сети получили свое развитие и в современности достигают новых успехов изо дня в день.

* 1. **Современные нейронные сети**

Благодаря современным вычислительным мощностям, новым архитектурам и количеству данных для обучения, нейронные сети стали не только машинами для предсказания и классификации, но и помощниками для обычных людей. В этом блоке будут рассмотрены современные нейронные сети, которые могут помогать обывателю без знания их устройства и примеры использования нейронных сетей крупными компаниями.

Нейросети для обработки текста:

* Первооткрыватель и главный двигатель прогресса в мире ИИ — Chat GPT. Мощнейший искусственный интеллект способный обрабатывать естественный язык, аудио, видео контент и различные документы. Он способен сделать за вас текст любой стилистики и на любую тему.
* Google Gemini. Чат бот с доступом к режиму разговорного диалога и обработки фотографий. Имеет бесплатный API для интеграции в свои проекты.
* DeepSeek. Полностью бесплатный чат бот с открытым исходным кодом, что позволяет локально установить его на свой компьютер и пользоваться без отправки своих данных в сеть.

Нейросети для обработки изображений:

* DALL E (OpenAI) — это одна из самых известных нейросетей для генерации изображений на основе текстовых описаний. Она способна создавать уникальные изображения. DALL E интегрирована с ChatGPT, что позволяет получать более точные результаты.
* MidJourney — это инструмент для создания художественных изображений. Нейросеть особенно популярна среди художников и дизайнеров благодаря своей способности генерировать изображения с уникальным стилем.
* Российские разработки: Шедеврум и Kandinsky от Яндекса и Сбера для генерации изображений по описанию.

Нейросети для обработки аудио:

* OpenAI Whisper — это нейросеть, предназначенная для распознавания и преобразования речи в текст. Она поддерживает множество языков и диалектов, а также способна работать с шумными аудиозаписями. Но, что является и плюсом, и минусом, доступна только либо для скачивания, либо через API.
* ElevenLabs — это нейросеть для генерации реалистичной речи. Она позволяет создавать аудио с различными голосами, настроением и интонацией. ElevenLabs часто используется для озвучки видео, создания аудиокниг и даже для разработки голосовых помощников. Нейросеть поддерживает API для интеграции в сторонние приложения.

Нейросети для анализа данных:

* TensorFlow — это одна из самых популярных платформ для машинного обучения и анализа данных. Она поддерживает создание и обучение нейронных сетей для решения многих задач. TensorFlow широко используется в научных исследованиях и промышленных приложениях.
* PyTorch — этоготовый набор инструментов для машинного обучения, который особенно популярен среди исследователей. PyTorch активно используется в задачах компьютерного зрения и обработки естественного языка.

Так с помощью этих нейросетей вы можете сделать свою жизнь проще. Они могут стать личным учителем, который может рассказать вам, что угодно и когда угодно или объяснить непонятную тему. Так и помочь учителю составить план урока, презентацию или проверить работы.

Нейросети в большом бизнесе:

* Машины на автопилоте. Дальше всех в этой сфере продвинулась компания «Tesla». В ее машинах установлены мощные камеры и компьютер способный обрабатывать тысячи изображений в секунду и принимать решение на их основе.
* Алгоритмы рекомендаций больших компаний для выдачи таргетированной лично на вас информации, например в YouTube, Spotify, Netflix.
  1. **Понятие нейронной сети в современном мире**

Рассмотрим определение термина нейронной сети и ИИ.

Иску́сственный интелле́кт (ИИ) (англ. artificial intelligence; AI) в самом широком смысле — это интеллект, демонстрируемый машинами, в частности компьютерными системами. Это область исследований в области компьютерных наук, которая разрабатывает и изучает методы и программное обеспечение, позволяющие машинам воспринимать окружающую среду и использовать обучение и интеллект для выполнения действий, которые максимально увеличивают их шансы на достижение поставленных целей. Такие машины можно назвать искусственным интеллектом. ИНС представляет собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры 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 простые по отдельности процессоры вместе способны выполнять довольно сложные задачи [7].

Нейронная сеть (также искусственная нейронная сеть, ИНС, или просто нейросеть) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации 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. [8].

Если проанализировать эти определения, то можно понять, что искусственный интеллект — это то, что демонстрирует обученная нейронная сеть, и что напоминает интеллект человека. А нейронная сеть — это математическая модель, построенная на основе принципа работы человеческого мозга и выполняющая конкретные функции.

* 1. **Основные выполняемые нейросетями функции**
* Классификация — распределение данных по классам на основе определённых параметров. Например, на вход дается набор характеристик человека и от нейросети требуется понять: давать человеку кредит или нет.
* Предсказание — возможность анализировать закономерности и на их основе давать предсказания. Например, рост или падение акций, основываясь на ситуации на фондовом рынке.
* Распознавание — Идентификация и детектирование объектов на изображении.

В этом проекте я буду рассматривать устройство классифицирующей нейросети для дальнейшего ее создания.

* 1. **Ключевые для понимания определения**

СПР — сеть прямого распространения.

Искусственный нейрон или просто нейрон — это составляющая часть слоя нейронной сети, обладающая связями (синапсами) с нейронами другого слоя, сумматор для суммирования всех подающихся на вход значений и смещение.

Синапс — связь между нейронами разных слоев ИНС, имеющая вес.

Полносвязный перцептрон — искусственная нейронная сеть, состоящая из слоев, последовательно связанных друг с другом.

Гиперпараметры нейросети — это параметры, которые не меняются в процессе обучения нейросети и подбираются вручную разработчиком исходя из вычислительных мощностей и потребностей.

Весы ИНС — все значения связей и смещений (все обучаемые характеристики).

Батч (batch) — неполная подвыборка из всего массива данных.

* 1. **Алгоритм работы**

В этом проекте будет рассмотрен принцип работы полносвязного перцептрона — одного из типов нейронных сетей из-за его простоты и наглядности для понимания работы нейросети. Он помогает понять основы устройства искусственного интеллекта, так как полносвязные слои, из которых он состоит, присутствуют почти во всех типах нейронных сетей.

* + 1. **слои перцептрона**

Проанализировав источники [11] [6] и то, как сам Розенблатт предложивший идею перцептрона, говорил о их устройстве, можно понять, что слои делятся на три типа:

1. Входной — нейроны, на которые изначально подается входная информация, будь это слова текста или пиксели фотографии. Входной слой всегда один. Входных нейронов в слое должно быть столько же, сколько у нас есть входных данных. Например, если на вход подается изображение разрешением 28 на 28 пикселей и каждый пиксель закодирован одним числом, то входных нейронов должно быть 28 \* 28 = 784. Значение входного нейрона никак не обрабатывается и передается следующим слоям неизменным.

2. Скрытые слои — нейроны, находящиеся между входными и выходными, слоев этих нейронов может быть сколько угодно. Пользователь никак не контактирует с этими нейронами, поэтому они и называются скрытыми. Количество нейронов в скрытых слоях выбирается заранее разработчиком. Нейроны скрытых слоев берут свое значение исходя из значений всех нейронов предыдущего слоя и передают его на следующий слой.

3. Выходной слой — последний слой нейронов, дающий окончательный результат. Выходной слой тоже только один. Нейронов в этом слое столько же, сколько возможных вариантов ответа. Ответом является нейрон с наибольшим значением.

* + 1. **Вычисление предсказания нейронной сети**

Начнем с того, как нейрон преобразует переданную ему информацию. Как стало понятно из вышесказанного, нейроны входного слоя ничего не делают с этой информацией, тогда нужно понять, как это делают нейроны скрытых и выходного слоев.

Синапс — это связь между двумя нейронами, которая имеет вес. У каждого нейрона одного слоя есть синапсы с каждым нейроном предыдущего и последующего слоя. И так каждый нейрон передает свое значение каждому следующему нейрону, умножая его на вес конкретного синапса. Эта информация поступает на нейрон и суммируется с также полученными значениями других нейронов, после чего к этой сумме прибавляется смещение нейрона. Дальше эту сумму нужно нормализовать, иначе предсказание сети будет линейной функцией неспособной к сложному анализу закономерностей, для этого и используется функция нормализации, в моем случае я буду использовать функцию ReLU, так как она используется чаще всего и требует меньше вычислительных мощностей в отличие от остальных функций активации [10].

ReLU (rectified linear unit) — это функция, превращающая отрицательные значения в 0, а положительные оставляет как есть. [3].

Формула ReLU:

График функции ReLU (приложение №1, рис. 1).

Для этого и нужно смещение, чтобы задать больше, чего должна быть сумма значений предыдущих нейронов, например, чтобы нейрон активировался, только если сумма больше 10 или больше -100.

После нормализации значения сигнал передается дальше по синапсам в следующие слои.

Рассмотрим всю эту схему на примере с двумя входными нейронами и одним скрытым нейроном. Обозначим входные нейроны I1, I2, скрытый как N, веса синапсов как W и смещение B. Тогда входными и выходными данными на нейрон будут:

*Ninput = I1\*W1 + I2\*W2*

*Noutput = ReLU(I1\*W1 + I2\*W2 + B)*

Запишем всё это в алгебраическом виде и сделаем формулы общими. Представим n-ный (отсчет идет с нуля) нейрон L-ого слоя (кроме входного и выходного) нейронной сети, обозначим ALn, нейроны предыдущего слоя AL-1k, весы синапсов как Wk, n, а смещение как Bn, а, где k— это номер нейрона предыдущего слоя, а n— это номер рассматриваемого нейрона. Значение этого нейрона: A = ReLU (W0,0AL-10 + W1,0AL-11+ ... + Wn, 0\*AL-1k + Bn). Если рассмотреть эту формулу без смещения и ReLU поподробнее, можно понять, что это первый элемент вектора, получаемого при умножении вектора значений нейронов предыдущего слоя и матрицы весов (каждая строка — это вектор весов синапсов нейрона предыдущего слоя со всеми нейронами текущего) = W0,0\*A0 + W0,1\*A1 + ... + W0,n\*An. Тогда, чтобы получить наше значение, нужно к этому произведению прибавить вектор смещений и взять сигмоиду. И все это можно красиво записать для первого слоя: A1 = ReLU(WA0 + B) или в общем виде:

L – номер слоя, n – количество нейронов в новом слое, k – количество нейронов в предыдущем слое. Сами формулы и как их выводить я изучил на нескольких источниках для подтверждения достоверности. [2] [9]

Так применяя эту формулу для каждого слоя, нейросеть доходит до последнего, а в нем нужно использовать другую функцию. Функция Softmax преобразует данный ей вектор в вектор вероятностей, так чтобы сумма этих вероятностей была равна единице. Её формула для каждого члена вектора:

Так ответом нейросети является распределение вероятностей ответов.

* + 1. **Обучение нейронной сети**

Нейросеть учится на данных. Эти данные должны быть заранее заготовлены и сгруппированы. Дальше эти данные делятся на два типа: тренировочные и тестовые. Тренировочные используются напрямую при обучении. Тестовые используются для отслеживания динамики обучения.

Обучение нейронной сети происходит по этапам, каждый этап называется эпохой. Данные для обучения для каждой эпохи одинаковые, и эпохой и называется одна итерация обучения на базе данных. Для каждой эпохи желательно каждый раз перемешивать данные. В каждой эпохе происходят следующие действия:

1. Деление данных на батчи (размер батча — это гиперпараметр).
2. Прямое распространение, т. е. получение результатов предсказания нейросети для этих данных.
3. Обратное распространение ошибки — это способ получения изменений весов ИНС.
4. Корректировка значений весов на основе суммарной ошибки за весь батч.

И далее те же действия по итерациям батчей и эпох.

Рассмотрим обучение способом стохастического градиентного спуска. В математике градиентом функции называют «направление» в пространстве функции, в котором она возрастает. Для нейросети это будет функция ошибки. Она показывает, насколько предсказание нейросети отличается от действительности. Так как функцию ошибки необходимо минимизировать, брать градиент нужно со знаком минус. Стохастическим его называют, потому что обучение идет по случайным неполным частям всех данных, а спуском из-за того, что, уменьшая функцию ошибки ее значение опускается в локальный минимум, где и остается.

(Приложение №1, рис. 2). На изображении показан случай с 2-х мерной функцией и одним глобальным минимумом, но в действительности у функции ошибки в нейросети параметров, а следовательно, и пространств сотни тысяч и множество локальных минимумов. Для получения вектора градиента используют метод обратного распространения ошибки. Он называется так, потому что мы сначала получаем градиент по последнему слою, редактируем слой, а после передаем с помощью формул этот градиент на предыдущий слой, и всё повторяется циклично.

Рассмотрев формулы обратного распространения ошибки из разных источников [1] [5] и проанализировав их, можно записать их в матричном виде для функции активации RuLU для скрытх слоев и softmax для послднего и crossentropy как функцию ошибки. Пусть L это номер слоя, T это транспонированная версия вектора или матрицы, WL это матрица весов между слоями L и L – 1, BL это вектор смещений данного слоя, Z то, что передается в функцию активации, F это функция активации, ⊙это поэлементное умножение, AL это вектор значений слоя L. То на сколько нужно изменить вес обозначается производной ошибки по этому весу, но я буду рассматривать формулы относительно матриц и векторов целых слоев для большей производительности программы.

После чего становится для предыдущего слоя и все повторяется по итерациям. Но для первой итерации нужно знать последнего слоя. Для этого есть отдельная формула. Для нее введём еще несколько обозначений. Y это правильный результат, SoftMax(Zmax) = P (результат работы нейросети в вероятностях).

Но так как нейронная сеть работает не с единичными прогнозами, а с батчами, то формулы изменятся. Из-за свойств матричного умножения почти все формулы останутся теми же, кроме одной:

A так же вычислим саму ошибку для отслеживания прогресса обучения нейросети. Для этого воспользуемся функцией CrossEntropy, она показывает, насколько распределение вероятности ответов соответствует действительности. CrossEntropy(P, Y) = . Но так как в истинном распределении вероятности только одна единица и остальные нули, то при умножении останется только индекс правильного результата, от суммы останется только -.

После рассмотрения принципа обучения нейросети, нужно научиться делать это в меру, так как ИНС может переобучиться. Это значит, что она слишком хорошо запомнит тренировочные данные и станет хуже работать на тестовых данных. Оптимальные гиперпараметры для конкретной нейросети подбираются в ходе ее обучения по валидационным данным.

1. **ПРАКТИЧЕСКАЯ ЧАСТЬ**

**Задача:** реализовать полносвязный перцептрон на языке Python без использования заготовленных библиотек вроде TensorFlow, дающих возможность создавать нейронные сети без написания их логики работы. И сделать это парадигме ООП, то есть создать класс нейросети для дальнейшей возможности импортирования этого класса в другие программы и большей универсальности.

* 1. **Код модели нейронной сети**

При обучении, необходимо инициализировать веса и смещения случайными значениями, например, центрированными около нуля и со средним отклонением, равным квадратному корню из количества нейронов в слое.

def init\_weights(self, layers):

    self.weights = []  # Инициализация пустых весов

    self.bias = []

# Присваивание случайных значений для всех слоев

    for num in range(len(layers) - 1):

        self.weights.append(

            np.random.normal(

                0.0, pow(layers[num], -0.5), (layers[num], layers[num + 1])

                )

            )

        self.bias.append(

            np.random.normal(

                0.0, pow(layers[num + 1], -0.5), (layers[num + 1])

                )

            )

И прежде, чем реализовывать обучение нейросети, нужно создать функции нужные для этого. А именно функцию активации, ее производную, функцию для перевода значений в вероятность, функцию ошибки, предсказание нейросети и функцию для ее тестирования.

def ReLU(x):  # Функция активации

    return np.maximum(0, x)

def ReLU\_deriv(Z):  # Производная ReLU

    return np.where(Z > 0, 1, 0)

def SoftMax\_batch(vect):  # Преобразование батча к вероятностям

    exp\_vect = np.exp(vect - np.max(vect, axis=1, keepdims=True))

    return exp\_vect / np.sum(exp\_vect, axis=1, keepdims=True)

def cross\_entropy\_batch(p, y, epsilon=1e-12):  # Вычисление расстояния между вероятностями

    p = np.clip(p, epsilon, 1.0)  # Защита от деления на 0

    return -np.sum(np.log(p[np.arange(len(y)), y])) / len(y)

Функция обучения будет брать на вход данные для обучения и тестирования и гиперпараметры нейросети. После будет получать предсказание и сохранять его для дальнейшего использования. После этого по формулам будет получать векторы градиента и корректировать значения весов. После завершения каждой эпохи будет сохранять значения функции ошибки и аккуратности на тестовых данных.

def learn(self, data, labels, epochs, batch\_size, train\_rate, valid\_data, valid\_labels):

    print('Начало обучения')

    for ep in range(epochs):

        epoch\_loss = 0

        correct\_count = 0

        indices = np.arange(data.shape[0])  # Перемешивание данных

        np.random.shuffle(indices)

        data = data[indices]

        labels = labels[indices]

        for i in range(data.shape[0] // batch\_size):

            # Прямое распространение нейросети

            layer = [data[i \* batch\_size : (i + 1) \* batch\_size]]

            y = labels[i \* batch\_size : (i + 1) \* batch\_size]

            Z = []

            for L in range(len(self.weights) - 1):

                Z.append(layer[L] @ self.weights[L] + self.bias[L])

                layer.append(ReLU(Z[L]))

            layer.append(SoftMax\_batch(layer[-1] @ self.weights[-1] + self.bias[-1]))

            # Обратное распространение

            dE\_dZ = layer[-1] - y\_vector\_batch(y, self.weights[-1].shape[1])

            for L in range(len(self.weights) - 1, -1, -1):

                dE\_dW = np.transpose(layer[L]) @ dE\_dZ

                dE\_dB = np.sum(dE\_dZ, axis=0, keepdims=True)

                dE\_dB.resize(self.bias[L].shape)

                if L > 0:

                    dE\_dA = dE\_dZ @ np.transpose(self.weights[L])

                    dE\_dZ = dE\_dA \* ReLU\_deriv(Z[L - 1])

                # Корректировка весов

                self.weights[L] -= train\_rate \* dE\_dW

                self.bias[L] -= train\_rate \* dE\_dB

            epoch\_loss += np.sum(cross\_entropy\_batch(layer[-1], y))

            correct\_count += num\_correct(layer[-1], y)

        # Вывод номера эпохи и точности нейросети

        accuracy = correct\_count / data.shape[0] \* 100

        print(f'Epoch {ep+1}/{epochs}: Loss = {epoch\_loss:.2f}, Train accuracy = {accuracy:.2f}%, Valid accuracy = {self.test(valid\_data, valid\_labels):.2f}%')

    print(f'Нейросеть обучена')

# Вычисление расстояния между вероятностями

def cross\_entropy\_batch(p, y, epsilon=1e-12):

    p = np.clip(p, epsilon, 1.0)  # Защита от деления на 0

    return -np.sum(np.log(p[np.arange(len(y)), y])) / len(y)

# Преобразование вектора правильных ответов к матрице истинных вероятностей

def y\_vector\_batch(y, num\_classes):

    return np.eye(num\_classes)[y]

def num\_correct(p, y): # Подсчет верных ответов

    return np.sum(np.argmax(p, axis=1) == y)

def predict\_batch(self, data):  # Предсказание нейросети для батча

    for i in range(len(self.bias) - 1):

        data = ReLU(data @ self.weights[i] + self.bias[i])

    return SoftMax\_batch(data @ self.weights[-1] + self.bias[-1])

def test(self, data, labels):

    return num\_correct(self.predict\_batch(data), labels) / data.shape[0] \* 100

def num\_correct(p, y): # Подсчет верных ответов

    return np.sum(np.argmax(p, axis=1) == y)

        for i in range(data.shape[0] // batch\_size):

            # Прямое распространение нейросети

            layer = [data[i \* batch\_size : (i + 1) \* batch\_size]]

            y = labels[i \* batch\_size : (i + 1) \* batch\_size]

            Z = []

            for L in range(len(self.weights) - 1):

                Z.append(layer[L] @ self.weights[L] + self.bias[L])

                layer.append(ReLU(Z[L]))

            layer.append(SoftMax\_batch(layer[-1] @ self.weights[-1] + self.bias[-1]))

            # Обратное распространение

            dE\_dZ = layer[-1] - y\_vector\_batch(y, self.weights[-1].shape[1])

            for L in range(len(self.weights) - 1, -1, -1):

                dE\_dW = np.transpose(layer[L]) @ dE\_dZ

                dE\_dB = np.sum(dE\_dZ, axis=0, keepdims=True)

                dE\_dB.resize(self.bias[L].shape)

                if L > 0:

                    dE\_dA = dE\_dZ @ np.transpose(self.weights[L])

                    dE\_dZ = dE\_dA \* ReLU\_deriv(Z[L - 1])

                # Корректировка весов

                self.weights[L] -= train\_rate \* dE\_dW

                self.bias[L] -= train\_rate \* dE\_dB

            epoch\_loss += np.sum(cross\_entropy\_batch(layer[-1], y))

            correct\_count += num\_correct(layer[-1], y)

        # Вывод номера эпохи и точности нейросети

        accuracy = correct\_count / data.shape[0] \* 100

        print(f'Epoch {ep+1}/{epochs}: Loss = {epoch\_loss:.2f}, Train accuracy = {accuracy:.2f}%, Valid accuracy = {self.test(valid\_data, valid\_labels):.2f}%')

    print(f'Нейросеть обучена')

* 1. **Обучение нейронной сети**
     1. **Эффективность работы**

Проверять эффективность обучения нейросети я буду используя набор данных MNIST. Это набор из 60000 тренировочных и 10000 тестовых черно-белых изображений рукописных цифр, написанных разными людьми в формате 28\*28 пикселей. Пример того, как выглядят изображения цифр: (Приложение №2, рис. 1).

При обучении используется параметр — скорость обучения. Это коэффициент, на который умножается градиент. Можно предположить, что лучше оставить значение таким каким его дают формулы, но при таком способе нейросеть не может попасть в локальный минимум и в итоге не обучается: (Приложение №2, рис. 2). Так для набора данных MNIST и моей архитектуры нейросети, она начинает эффективно обучатся на значении скорости 0,01. Ниже это значение делать нет смысла, так как нейросеть будет слишком медленно обучатся.

Для обучения нейросети нужно выбрать количество слоев и нейронов в них. В первом слоев должно быть 784 нейрона так как пикселей в фотографии 28\*28. В последнем слое должно быть 10 нейронов, потому что выходной результат это распределение вероятностей того, что на изображение та или иная цифра. Размер батча обычно выбирается от 30 до 100, для того чтобы использовать меньше вычислительных ресурсов, так что я выберу 50 изображений для одного батча.

Опытным путем я выяснил, что большинство конфигураций нейронных сетей для распознавания цифр, перестают улучшать результат на 15 эпохе, а некоторые конфигурации достигают 100% точности на тренировочных данных, что не позволяет им расти дальше. Исходя из этого, и чтобы не дать нейросети переобучится использовать я буду значение в 20 эпох.

Рассмотрим результаты обучения нейросети с различными конфигурациями:

* Самая первая в мире нейросеть — однослойный перцептрон.

Реализуем его, запустив нейросеть с слоями состоящими из 784 и 10 нейронов. Графики обучения: (Приложение №3, рис. 1). На них видно, что точность такого перцептрона достигает 92%. Для такой простой архитектуры это уже хороший результат.

Так как первый слой сразу связывается с нейронами, дающими ответ, то если визуализировать матрицу связей первого слоя с 0-ым нейроном второго слоя, то можно увидеть усредненное изображение нуля из этого набора данных, где белым показаны самые большие веса, а черным самые маленькие. (Приложение №3, рис. 2) и так же с остальными цифрами.

* Большой перцептрон.

Обучим сеть с слоями содержащими такое количество нейронов: 784, 512, 256, 10. Обучение такой нейросети длилось 5 минут, когда как однослойный перцептрон обучился меньше чем на минуту. Конечная точность этой конфигурации вышла 98.62% (Приложение №4, рис. 1).

* Средняя конфигурация.

Выберем оптимальные параметры: 784, 256, 10. В таком случае модель обучается минуту, а 100% на тренировочных данных добивается еще быстрее. При этом ее точность почти такая же, что и у более крупной модели (Приложение №4, рис. 2).

При этом если визуализировать весы этой сети, то у первого слоя видно, что веса определяют отдельный элемент цифры, а на втором слое просто шум (Приложение №4, рис. 3).

Если посмотреть на каких изображениях ошиблась нейросеть, то можно понять, что у человека была бы тоже не 100% точность (приложение №5, рис. 1).

* + 1. **Скорость работы**

Для проверки гипотезы измерим время работы нейросети. Для этого засеку время с помощью стандартного модуля time и получу предсказание для тренировочных и тестовых данных. Учитывая, что я не располагаю мощной вычислительной техникой, результаты получились: 0.459 секунды для 60 тысяч тренировочных и 0.086 секунды для 10 тысяч тестовых изображений. Это подтверждает, что искусственный интеллект работает в тысячи раз быстрее человека.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения проекта «Искусственный интеллект. Модель, проектирование, разработка» была поставлена цель — изучить принципы работы нейронных сетей и создать собственную модель полносвязного перцептрона на языке Python без использования готовых библиотек. Эта цель была успешно достигнута.

В теоретической части проекта были рассмотрены:

* История развития нейронных сетей и ключевые понятия, связанные с искусственным интеллектом.
* Современные примеры искусственного интеллекта для применения в практике.
* Принципы работы полносвязного перцептрона, включая алгоритмы прямого и обратного распространения ошибки.

Практическая реализация включала разработку класса нейронной сети по принципам объектно-ориентированного программирования. Модель была протестирована на наборе данных MNIST, что позволило наглядно продемонстрировать, что даже простая архитектура способна достигать высокой точности распознавания рукописных цифр. Экспериментальные результаты (точность от 92% для однослойного перцептрона до 98+% для более сложных моделей) подтвердили эффективность выбранного подхода и корректность реализации.

Благодаря изученной информации и проделанной работе была создана страница проекта на GitHub, где размещен исходный код программы для обучения и работы нейросети, программы для визуализации весов нейросети, кода интерфейса позволяющего самостоятельно рисовать цифры и наглядно видеть результат работы нейросети, и весы обученной мною нейросети (Приложение №5, ссылка на проект).

**Теоретическая значимость:** работа рассказывает об устройстве нейронных сетей и поможет привлечь к этой теме больше людей.

**Практическая значимость проекта**:

* Проект служит отличной базой для дальнейшего изучения и экспериментов в области машинного обучения.
* Разработанный класс нейросети можно использовать в образовательных целях и как основу для создания более сложных моделей, а также может быть использована для обучения нейросети на различных наборах данных, включая MNIST, что демонстрирует возможности ИИ в решении задач классификации.

Таким образом, проделанная работа не только позволила глубже понять внутреннюю логику работы нейронных сетей, но и продемонстрировала, что самостоятельная реализация алгоритмов обучения — эффективный способ освоения основ искусственного интеллекта. Полученные знания и навыки станут прочной базой для дальнейших исследований и разработки более сложных систем в области ИИ.

Считаю, что цель проекта достигнута, поставленные задачи выполнены. Гипотеза о том, что нейронные сети могут автоматизировать задачи классификации данных и делать это лучше и быстрее людей подтверждена. В будущем планирую продолжить исследование в данном направление и создавать более сложные модели искусственного интеллекта.

**Искусственный интеллект может открыть дорогу к обществу изобилия, в котором необходимость работать исчезнет для большинства людей.**

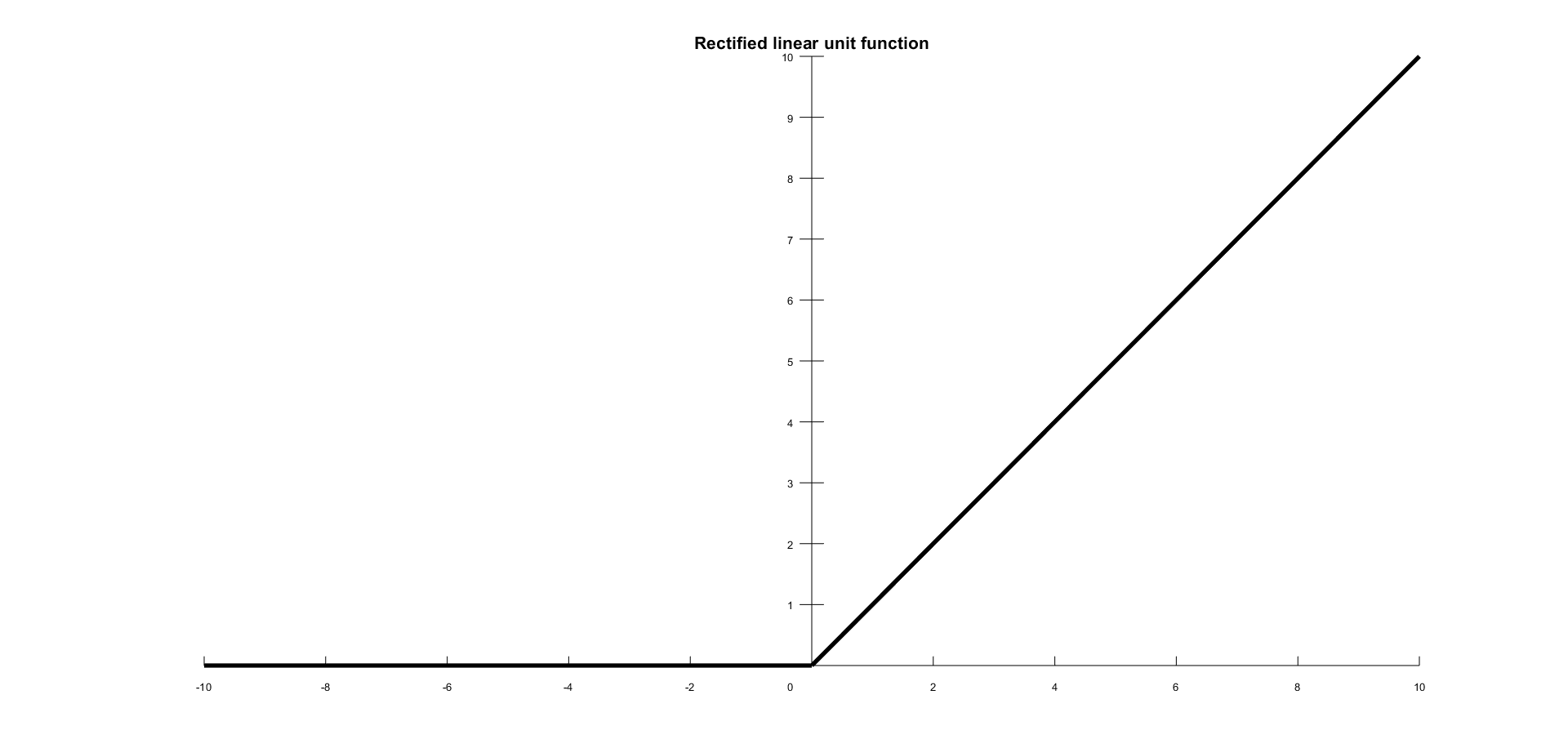
Илон Маск [13]

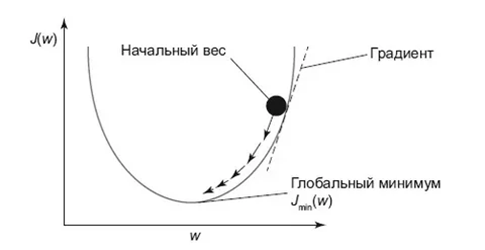
**У нас есть уникальная возможность улучшить жизнь с помощью ИИ. Мы все еще на начальном этапе, но я уже верю, что ИИ станет самым значительным изменением в нашей жизни.**

Глава компании OpenAI Сэм Альтман [14]

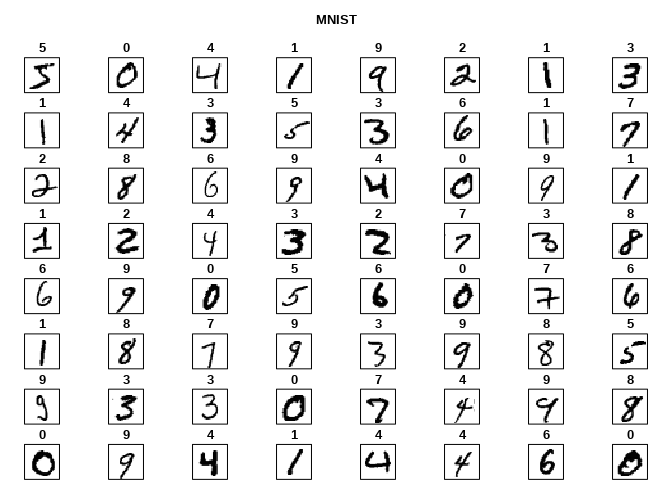
**БИБЛИОГРАФИЯ**

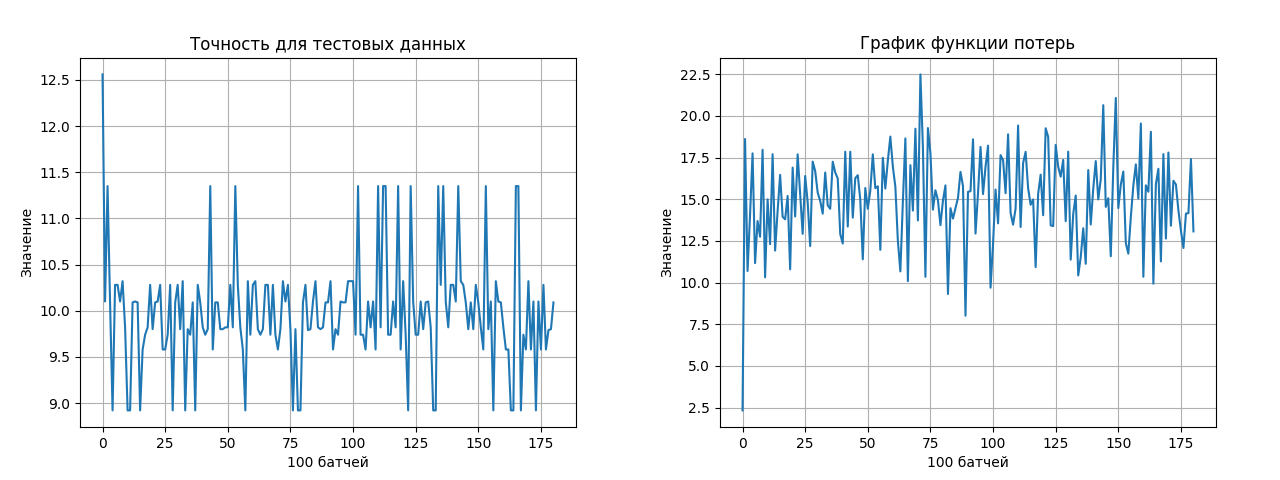
1. Вывод общих матричных формул обратного распространения ошибки. <https://habr.com/ru/companies/ods/articles/344116/>
2. Вывод формул для прямого распространения в перцептроне. [https://habr.com/ru/articles/722422/#anchor\_3](https://habr.com/ru/articles/722422/%23anchor_3)
3. Информация о функции ReLU. <https://en.wikipedia.org/wiki/Rectifier_(neural_networks)>
4. История создания нейронных сетей. <https://habr.com/ru/articles/861888/>
5. Общее представление об обратном распространении ошибки и формулы. <https://en.wikipedia.org/wiki/Backpropagation>
6. Описание модели перцептрона. <https://habr.com/ru/articles/214317/>
7. Определение искусственного интеллекта. <https://ru.wikipedia.org/wiki/ИИ>
8. Определение нейронной сети. <https://ru.wikipedia.org/wiki/Нейронная_сеть>
9. Принципы прямого и обратного распространения ошибки. [https://education.yandex.ru/handbook/ml/article/pervoe-znakomstvo-s-polnosvyaznymi-nejrosetyami#forward-and-backward-propagation](https://education.yandex.ru/handbook/ml/article/pervoe-znakomstvo-s-polnosvyaznymi-nejrosetyami%23forward-and-backward-propagation)
10. Разновидности функций активации. <https://deepmachinelearning.ru/docs/Neural-networks/MLP/Activation-functions>
11. Теоретические объяснения логики перцептронов. <https://neural.radkopeter.ru/chapter/персептроны/>
12. Цитата В. В. Путина президента РФ про Искусственный интеллект. <https://tass.ru/ekonomika/22643091>
13. Цитата Илона Маска об ИИ. <https://tass.ru/obschestvo/19190483>
14. Цитата Сэма Альтмана об ИИ в нашей жизни. [rbc.ru/technology\_and\_media/13/02/2025/67adc1e69a7947a9e82690d7](https://www.rbc.ru/technology_and_media/13/02/2025/67adc1e69a7947a9e82690d7)

**ПРИЛОЖЕНИЕ 1**

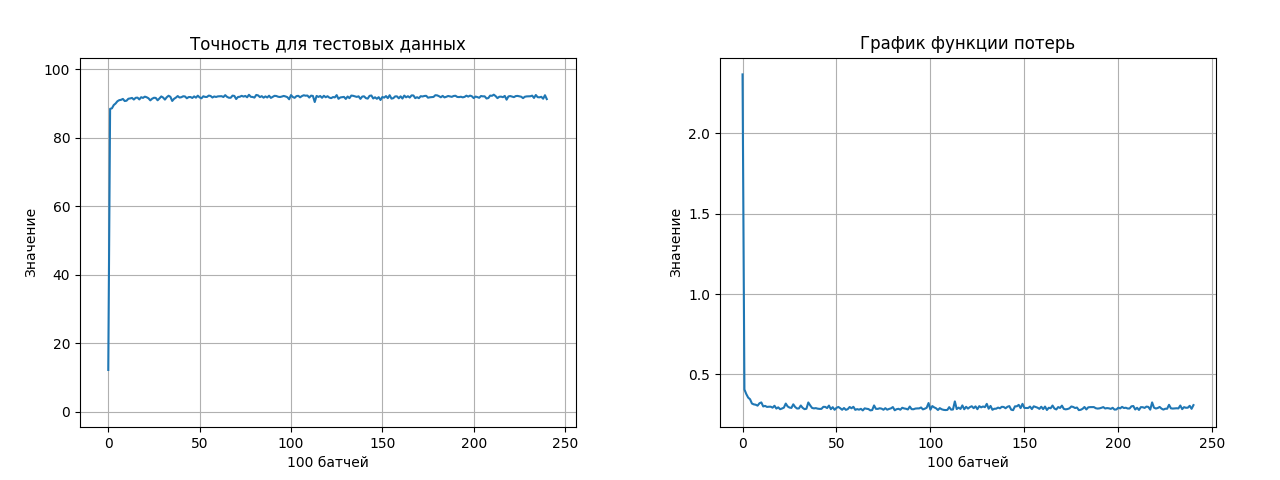
**Рис. 1** График функции ReLU

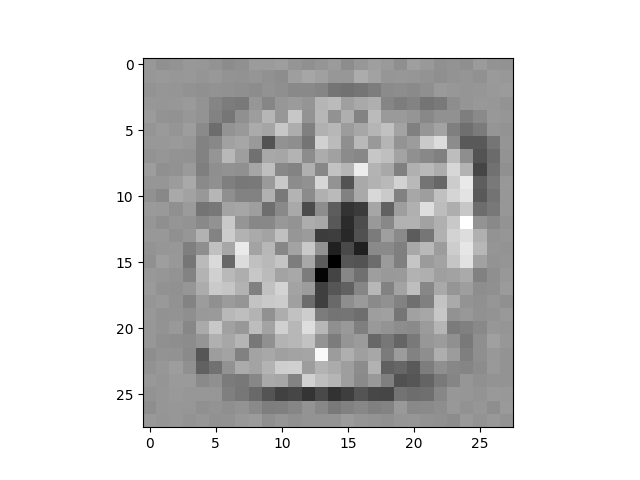
**Рис. 2** Пример графика функции ошибки и его градиента

**ПРИЛОЖЕНИЕ 2**

**Рис. 1** Пример изображений рукописных цифр из набора MNIST

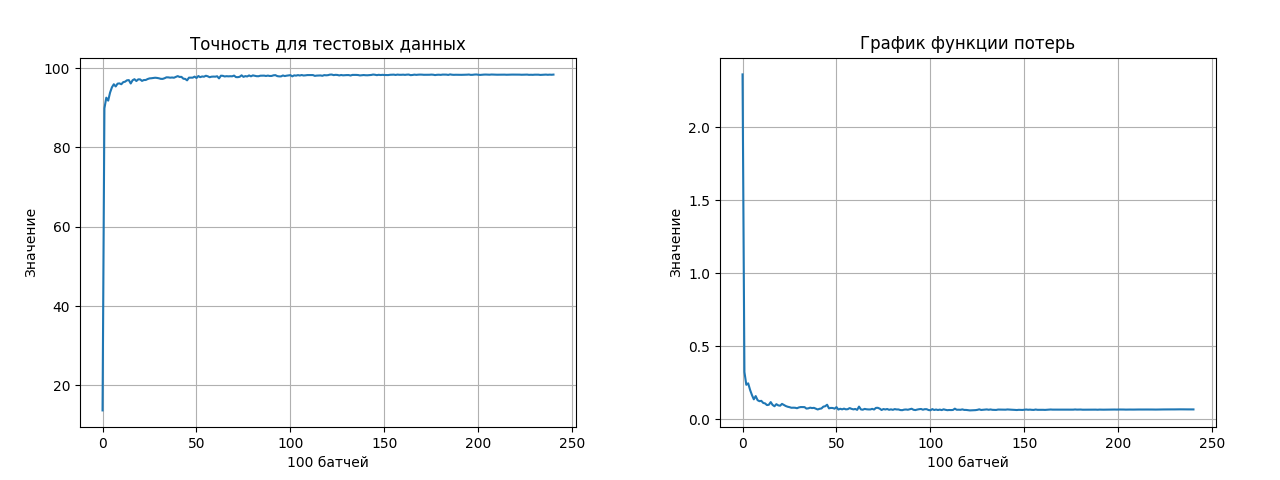
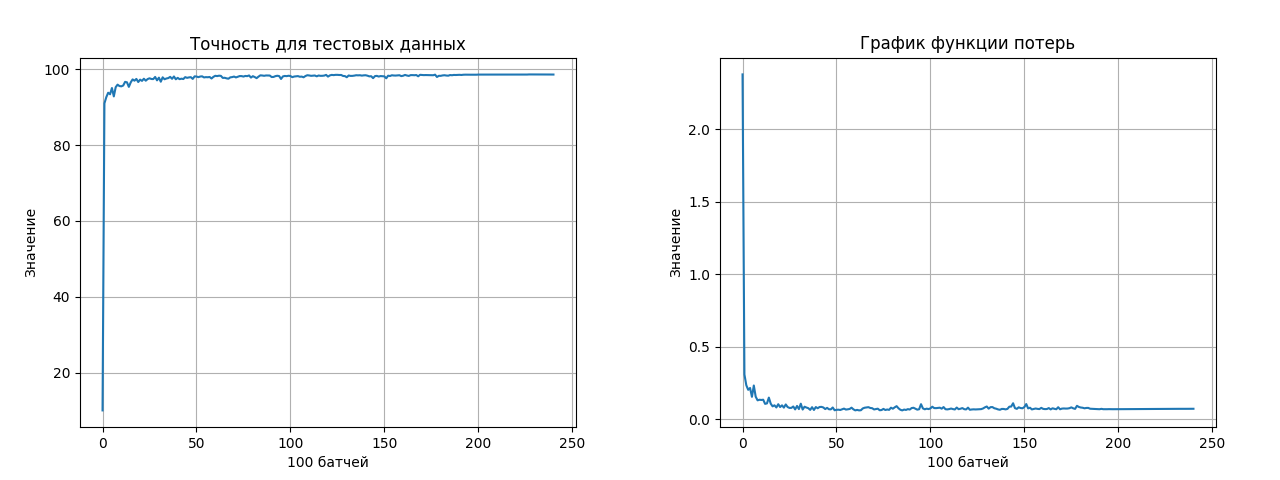
**Рис. 2** Графики обучения со скоростью равной 1

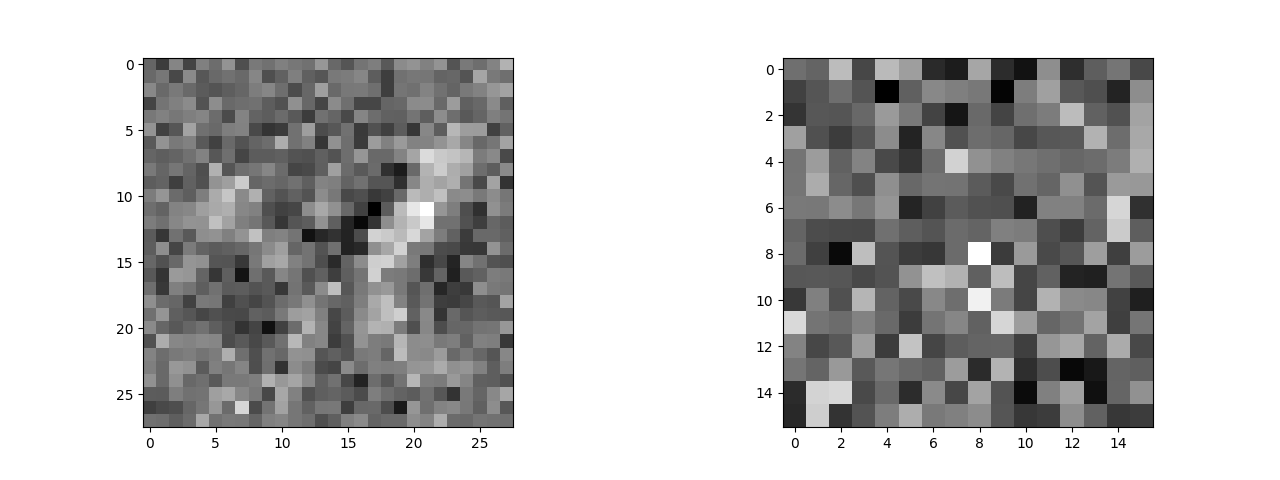
**ПРИЛОЖЕНИЕ 3**

**Рис. 1** Графики обучения однослойного перцептрона

**Рис. 2** Визуализация весов однослойного перцептрона.

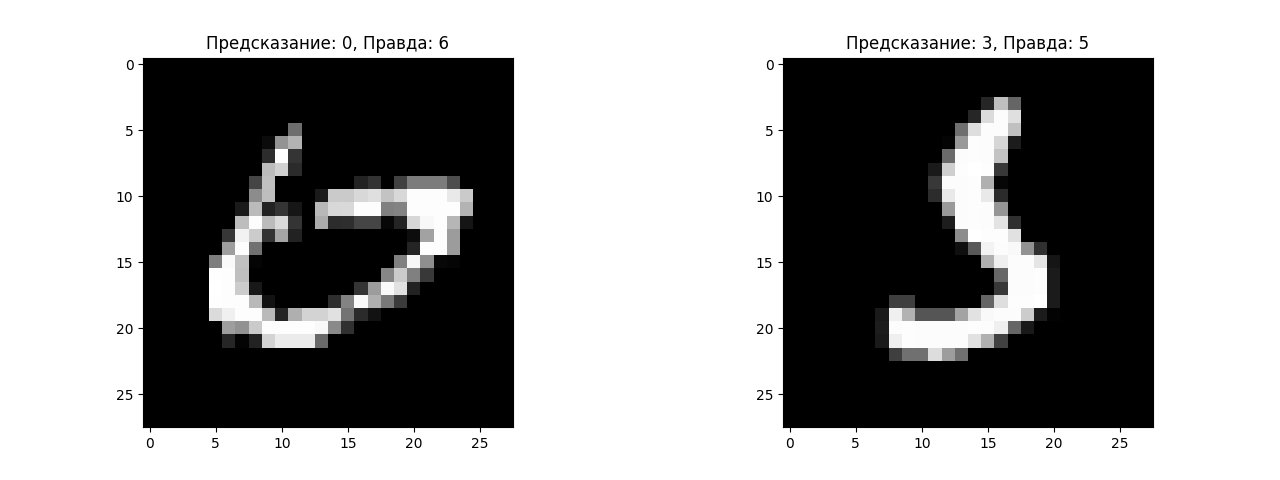
**ПРИЛОЖЕНИЕ 4**

**Рис. 1** Графики обучения для большого перцептрона

**Рис. 2** Графики обучения для оптимальных параметров

**Рис. 3** Визуализация весов двух слоев многослойного перцептрона

**ПРИЛОЖЕНИЕ 5**

**Рис. 1** Примеры ошибки нейросети

Ссылка на проект в GitHab:

<https://github.com/SamoninEvgeniy/neural_network>