ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ БЮДЖЕТНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**(ФИНАНСОВЫЙ УНИВЕРСИТЕТ)**

Департамент анализа данных

и машинного обучения

**Дисциплина: «Технологии анализа данных и машинного обучения»**

Направление подготовки: «Прикладная информатика»

Профиль: «ИТ-сервисы и технологии обработки данных в экономике и финансах»

Факультет информационных технологий и анализа больших данных

Форма обучения очная

Учебный 2022/2023 год, 6 семестр

**Курсовая работа на тему:**

«Эффективность самонормализующихся нейронных сетей»

Выполнил:

студент группы ПИ20-4

Куликов Г. Ж.

Научный руководитель:

профессор, д.э.н. Демин И. С.

Москва 2023

**Оглавление**

[**Постановка задачи** 4](#_Toc134447649)

[**Предметная область** 5](#_Toc134447650)

[**Описание программы** 7](#_Toc134447651)

[**1)** **Набор данных** 7](#_Toc134447652)

[**2)** **Архитектура и общая логика** 15](#_Toc134447653)

[**Многослойный перцептрон** 19](#_Toc134447654)

[**Сверточная сеть** 22](#_Toc134447655)

[**Неэффективный многослойный перцептрон** 25](#_Toc134447656)

[**Заключение** 29](#_Toc134447657)

[**Список литературы** 30](#_Toc134447658)

[**Приложение (исходный код)** 31](#_Toc134447659)

**Введение**

Современный мир охватывает большое количество данных, которые необходимо обрабатывать и анализировать для получения полезной информации. Нейронные сети являются одним из наиболее перспективных и эффективных инструментов для обработки данных. Они нашли широкое применение в таких областях, как распознавание образов, обработка естественного языка, компьютерное зрение, анализ временных рядов и т.д.

Нейронные сети развиваются быстрыми темпами, постоянно появляются новые архитектуры и методы обучения. Одним из наиболее актуальных направлений в развитии нейронных сетей является самонормализация. Самонормализующиеся нейронные сети позволяют автоматически нормализовать входные данные, что ускоряет и улучшает процесс обучения.

В данной работе мы будем изучать эффективность самонормализующихся нейронных сетей и сравним их с нейронными сетями, которые не используют эту технологию. Самонормализация в нейронных сетях является достаточно новым подходом, который пока не получил широкого распространения в индустрии и научных исследованиях. Однако, уже сейчас можно сказать, что эта технология обладает большим потенциалом и может стать одним из ключевых инструментов для создания более эффективных и точных моделей нейронных сетей.

В настоящее время самонормализующиеся нейронные сети активно исследуются и разрабатываются ведущими компаниями и университетами по всему миру. Они уже показали свою эффективность в таких областях, как обработка естественного языка, распознавание речи, компьютерное зрение и другие. В будущем, с развитием технологий, мы можем ожидать еще большего расширения областей применения самонормализующихся нейронных сетей.

Целью данной курсовой работы является изучение эффективности самонормализующихся нейронных сетей и сравнение их с обычными нейронными сетями без использования этой технологии. В работе мы проведем исследования на нескольких наборах данных, чтобы оценить эффективность и точность обеих типов нейронных сетей. Мы также рассмотрим основные принципы работы самонормализующихся нейронных сетей и проанализируем их преимущества и недостатки.

# **Постановка задачи**

Постановка задачи в курсовой работе "Эффективность самонормализующихся нейронных сетей" заключается в сравнении эффективности обучения нейронных сетей с самонормализацией и без нее. Для достижения этой цели необходимо выполнить следующие задачи:

1. Изучить основные принципы работы нейронных сетей и самонормализации.
2. Изучить существующие методы и алгоритмы обучения нейронных сетей с самонормализацией и без нее.
3. Провести исследования на нескольких наборах данных и сравнить эффективность и точность самонормализующихся и обычных нейронных сетей.
4. Оценить скорость обучения, точность и устойчивость к переобучению каждого типа нейронной сети.
5. Проанализировать результаты исследований и сделать выводы о преимуществах и недостатках самонормализующихся нейронных сетей.
6. Оценить перспективы использования самонормализующихся нейронных сетей в будущем.

Для достижения этих целей необходимо провести обширный анализ литературы, изучить и реализовать существующие алгоритмы обучения нейронных сетей с самонормализацией и без нее, а также провести серию исследований на нескольких наборах данных. Результаты экспериментов позволят сделать выводы о преимуществах и недостатках самонормализующихся нейронных сетей, а также оценить их перспективы в будущем.

# **Предметная область**

Предметной областью курсовой работы "Эффективность самонормализующихся нейронных сетей" является область искусственного интеллекта и машинного обучения. Конкретно, работа посвящена изучению эффективности самонормализующихся нейронных сетей в задачах обработки данных.

Нейронные сети – это компьютерные алгоритмы, которые используются для анализа больших объемов данных и выполнения сложных задач. Они состоят из нейронов, которые имитируют функционирование нейронов в головном мозге. Каждый нейрон принимает на вход некоторое число и возвращает выходное значение, которое затем передается на вход следующего нейрона. Таким образом, информация постепенно проходит через все нейроны сети и обрабатывается, пока не будет получен окончательный результат.

Нейронные сети без самонормализации могут столкнуться с проблемой затухания градиента, когда градиент ошибки по весам сети становится слишком маленьким для обновления весов. В таком случае, сеть перестает обучаться и не достигает своей полной эффективности. Кроме того, может возникнуть проблема взрыва градиента, когда градиент ошибки по весам становится слишком большим для корректного обновления весов, что также может привести к нестабильности обучения.

Самонормализующиеся нейронные сети (self-normalizing neural networks) представляют собой класс нейронных сетей, которые автоматически нормализуют данные на каждом слое и избегают проблем затухания и взрыва градиента. Они были разработаны для решения проблем нестабильности обучения, и показывают более высокую эффективность в задачах обработки данных.

Самонормализующиеся нейронные сети достигают этого путем использования активационной функции с нормализацией. В отличие от традиционных активационных функций, которые не могут обеспечить нормализацию данных на каждом слое, активационная функция с нормализацией обеспечивает стабильное распределение активаций на каждом слое, что позволяет сети быстрее и более стабильно обучаться.

Важно отметить, что выбор нейронной сети с самонормализацией или без нее зависит от конкретной задачи, которую необходимо решить, и от характеристик данных. Для некоторых задач самонормализующиеся сети могут оказаться неэффективными, в то время как для других они могут быть наилучшим выбором. В любом случае, выбор типа нейронной сети является важным шагом в процессе решения задачи и требует тщательного анализа данных и постановки целей.

В заключение, самонормализующиеся нейронные сети представляют собой эффективный метод решения проблем затухания и взрыва градиента в процессе обучения. Они позволяют обеспечить стабильность и предсказуемость в работе нейронных сетей, что является важным фактором в задачах обработки данных. В будущем, они могут стать еще более популярными и широко применяемыми в различных областях машинного обучения и искусственного интеллекта.

# **Описание программы**

Вся работа была произведена на языке программирования Python 3.x и реализована с помощью средств, которые предоставляются фреймворком PyTorch.

## **Набор данных**

В рамках данной предметной области значительного влияния типа набора данных на результаты исследования не будет, поэтому в качестве него подойдет любой таковой. Так как самонормализация может быть реализована в любых задачах независимо от их типа, была выбрана задача множественной классификации изображений.

Для избежания возможности фактора влияния архитектуры нейронной сети на эффективность таковой с реализованной самонормализацией были выбраны 2 типа нейронных сетей: многослойный перцептрон и сверточная сеть.

В целях уменьшения временных затрат на поиски были выбраны два набора данных, которые удовлетворяют требованиям, указанным выше, и предустановленным в программном пакете torchvision: MNIST и CIFAR10.

MNIST (Modified National Institute of Standards and Technology database) – это набор данных, состоящий из изображений символов, каждый из которых является одной из цифр от 0 до 9. MNIST является простым и широко используемым набором данных для начального обучения нейронных сетей. В задачах распознавания символов обычно используется модель машинного обучения для классификации изображений на основе представления их признаков.

Набор данных MNIST был создан в 1998 году Национальным институтом стандартов и технологии (National Institute of Standards and Technology, NIST) для разработки и оценки алгоритмов распознавания символов. В последующие годы MNIST стал широко используемым набором данных для задач машинного обучения и компьютерного зрения.

Ниже на рисунке 1 изображены примеры изображений из данного набора данных.

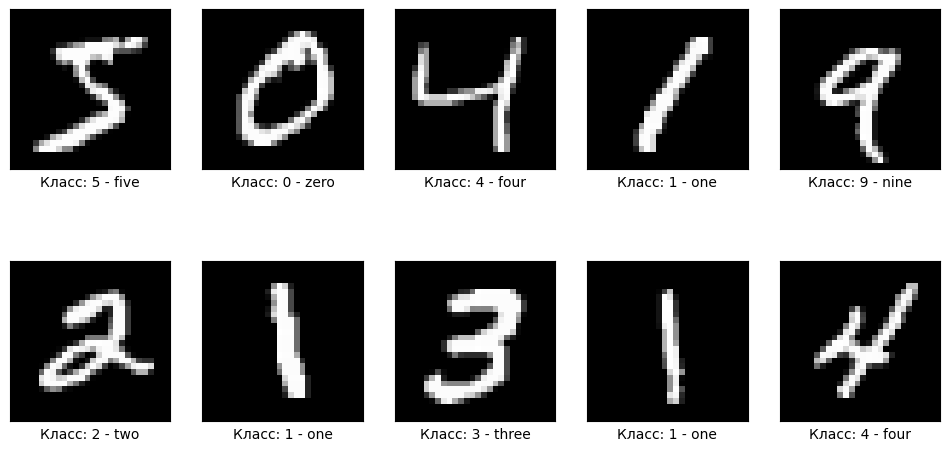


Рисунок 1 – Изображения из набора данных MNIST

Каждое изображение в датасете представляет из себя двумерную матрицу размером 28x28. Эти измерения являются разрешением изображения – т. е. 28 на 28 пикселей. Так как сами изображения являются черно-белыми они не хранят в себе компоненты трех цветов в цветовом пространстве RGB, как это обычно бывает. Вместо этого они хранят в себе компоненту белого цвета, находящейся в диапазоне от 0 до 255, где она равна 0, если цвет соответствующего пикселя является полностью черным, или 255, если цвет соответствующего пикселя является, наоборот, полностью белым.

Сам датасет состоит из 70 000 изображений. Каждому классу изображений – цифрам от 0 до 9 – соответствует в среднем по 7 000 изображений. Данное распределение можно увидеть на рисунке 2 ниже.

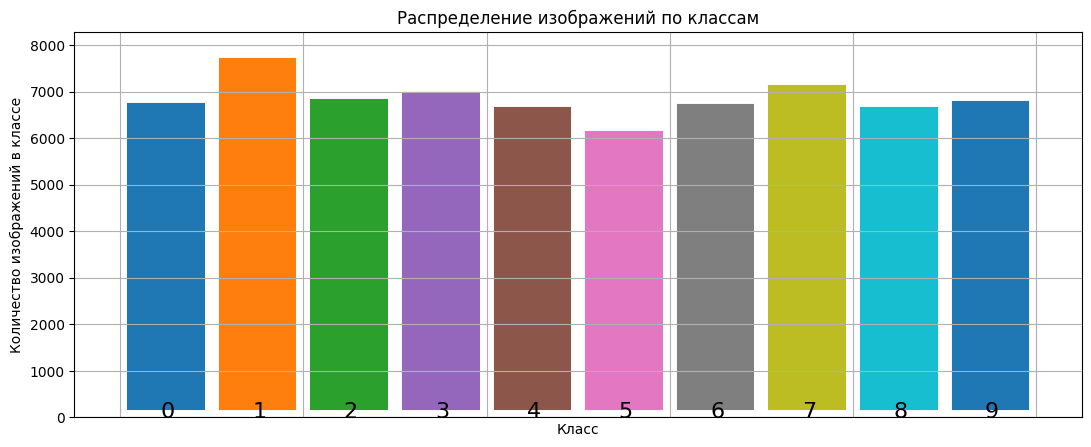


Рисунок 2 – Распределение классов в исходном датасете MNIST

Как можно увидеть на рисунке, данные распределены неравномерно, однако дисбаланс не выражен слишком сильно – стандартное отклонение составляет всего около 380 – , поэтому он не сможет критически повлиять на эффективность моделей.

Программный пакет torchvison предоставляет возможность автоматически разделить всю выборку на две части – обучающую и тестовую – однако разделение производится в соответствии с заранее жестко заданными пропорциями, настроить которые не представляется возможности.

Распределение изображений по классам в обучающей и тестовой выборках показаны ниже на рисунках 3 и 4.

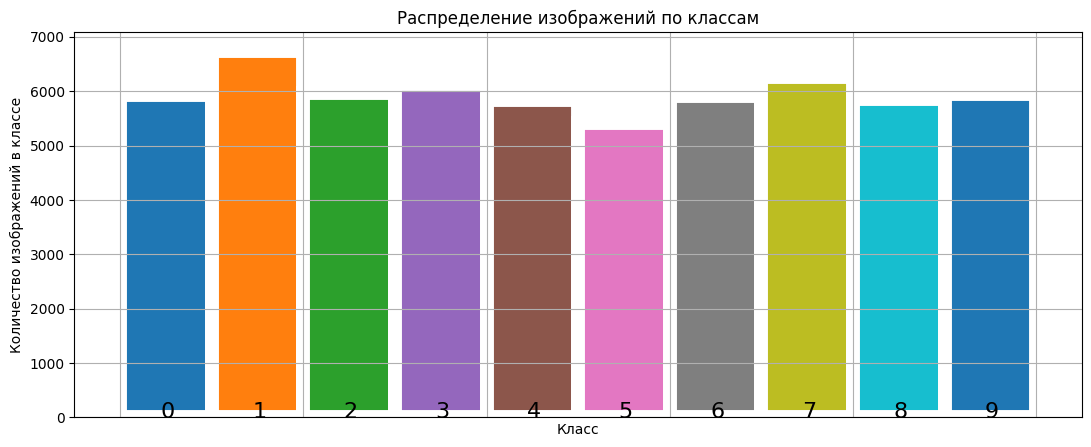


Рисунок 3 – Распределение классов в обучающем датасете MNIST

В обучающей выборке находится всего 60 000 изображений, в каждом классе в среднем по 6 000 изображений, а стандартное отклонение составляет всего около 320.

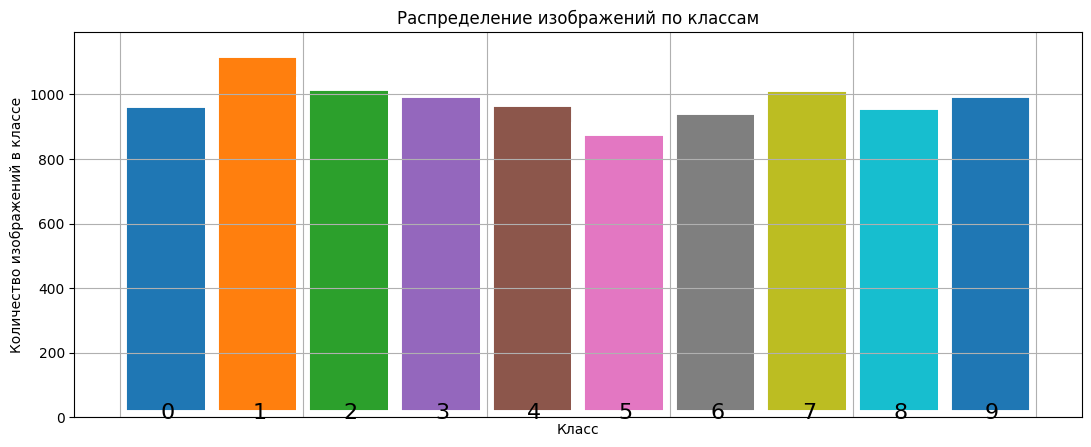


Рисунок 4 - Распределение классов в тестовом датасете MNIST

В тестовой выборке находится всего 10 000 изображений, в каждом классе в среднем по 1 000 изображений, а стандартное отклонение составляет всего около 60.

Также программный пакет torchvision предоставляет возможность произвести трансформацию данных при загрузке. Для этого используется объект типа Compose, который представляет из себя контейнер, хранящий последовательность объектов для преобразования данных.

В частности, для датасета MNIST были использованы следующие преобразования:

* ToTensor. Данный класс превращает каждое изображение в исходных данных в тензор размерностью 28x28.
* Normalize. Данный класс производит нормализацию данных по переданным значениям среднего и стандартного отклонения. В данном конкретном случае они была равны 0.1307 для среднего и 0.3081 для стандартного отклонения. Данные значения являются наиболее подходящими и часто используемыми, они получены на основе анализа обучающей части выборки и указаны в документации pytorch, что доказывает целесообразность применения их в данной модели для увеличения производительности вычислений.

CIFAR10 – это другой набор данных для задач компьютерного зрения. Данный набор состоит из изображений различных комплексных объектов, относящихся к разным, не однотипным классам.

Для каждого изображения в CIFAR10 задан один из 10 классов: "самолет", "автомобиль", "птица", "кот", "олень", "собака", "лягушка", "лошадь", "корабль", "грузовик".

CIFAR10 является сложным набором данных, поскольку объекты на изображениях имеют различную форму, размер, текстуру и цвет, а также различаются по своему классу. Изображения в CIFAR10 также содержат шум и различные искажения, что делает задачу классификации более сложной.

Набор данных CIFAR10 был создан в 2009 году Канадским институтом передовых исследований (Canadian Institute For Advanced Research) в рамках проекта CIFAR. Он был разработан для оценки и сравнения алгоритмов компьютерного зрения и глубокого обучения на сложных задачах классификации изображений.

CIFAR10 широко используется в задачах классификации изображений и детекции объектов, а также в качестве набора данных для обучения сверточных нейронных сетей. Из-за сложности задачи классификации и множества классов CIFAR10 является более сложным и интересным набором данных для исследования, чем, например, MNIST.

Еще одним отличием от датасета MNIST является то, что изображения в наборе цветный, а не черно-белые.

На рисунке 5 ниже предоставлены примеры изображений из данного набора данных.

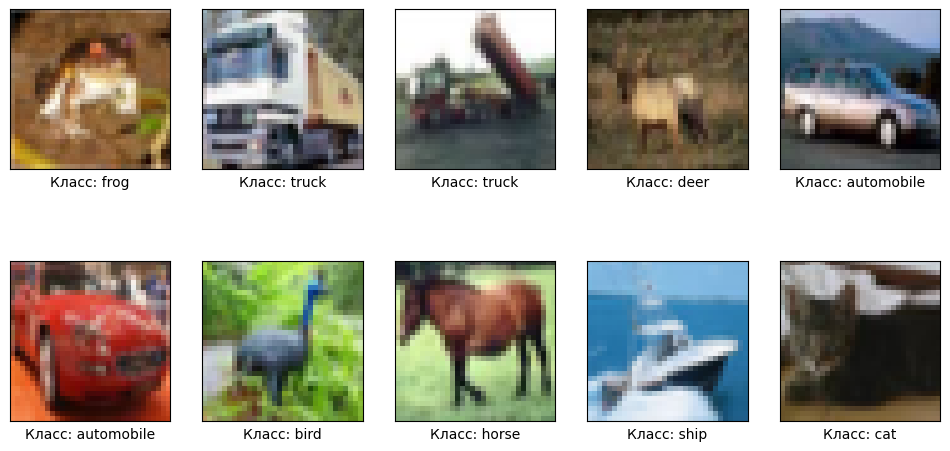


Рисунок 5 – Изображения из набора данных CIFAR10

Каждое изображение в датасете представляет из себя трехмерную матрицу размером 32x32x3. Первое и второе измерения являются разрешением изображения – т. е. 32 на 32 пикселей. Так как сами изображения являются не черно-белыми, а цветными они хранят в себе компоненты трех цветов в цветовом пространстве RGB. Каждый компонент хранит в себе значения красного, зеленого или синего цветов соответственно, которое принимает значение от 0 до 255 в зависимости от насыщенности цвета.

Сам датасет состоит из 60 000 изображений. Каждому классу изображений, которые были перечислены выше, соответствует в среднем по 6 000 изображений. Данное распределение можно увидеть на рисунке 6 ниже.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 6 – Распределение классов в исходном датасете CIFAR10

Как можно увидеть на рисунке, данные распределены абсолютно равномерно, стандартное отклонение составляет 0.

Данные были разделены на обучающую и тестовую выборки согласно заранее жестко заданным пропорциям в программном пакете torchvison.

Распределение изображений по классам в обучающей и тестовой выборках показаны ниже на рисунках 7 и 8.

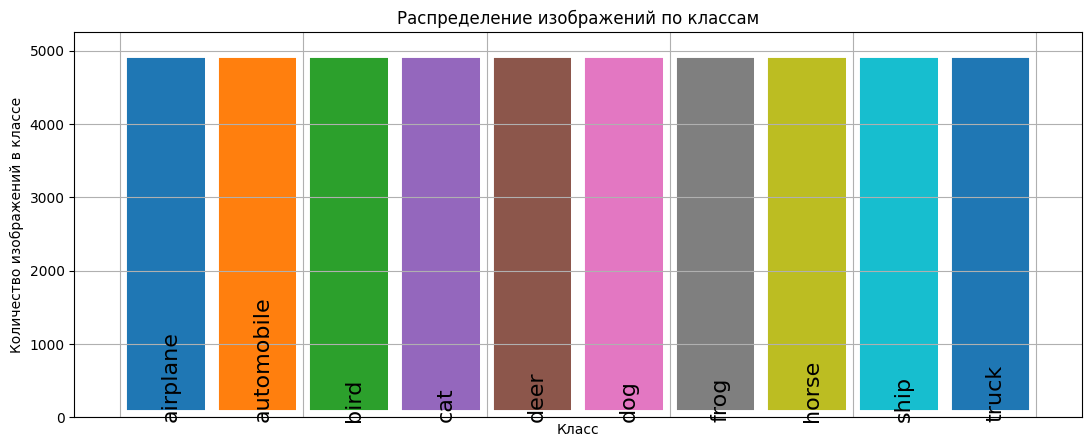


Рисунок 7 – Распределение классов в обучающем датасете CIFAR10

В обучающей выборке находится всего 50 000 изображений, в каждом классе в среднем по 5 000 изображений, а стандартное отклонение составляет 0.

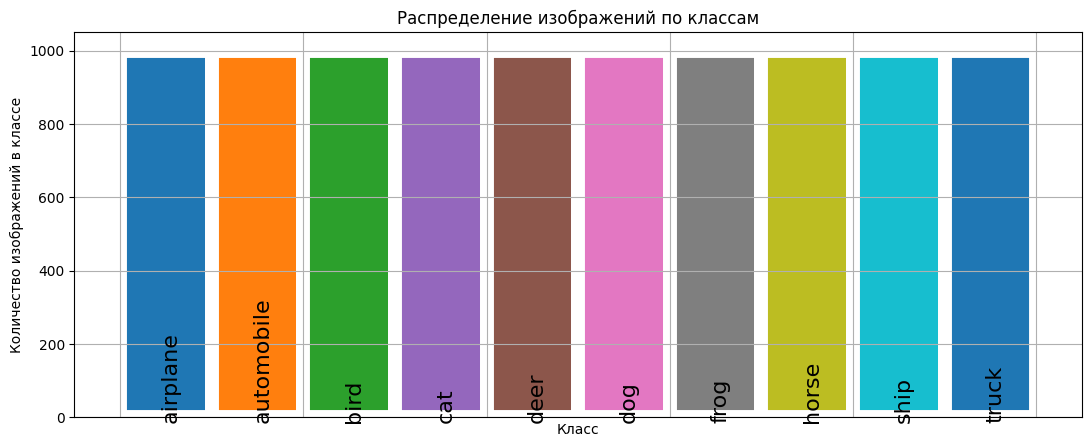


Рисунок 8 – Распределение классов в тестовом датасете CIFAR10

В тестовой выборке находится всего 10 000 изображений, в каждом классе в среднем по 1 000 изображений, а стандартное отклонение составляет 0.

Таким образом, и в обучающей, и в тестовой выборке данные сбалансированы идеально равномерно.

Для датасета CIFAR10 были использованы следующие преобразования:

* ToTensor. Данный класс превращает каждое изображение в исходных данных в тензор размерностью 28x28.
* Normalize. Данный класс производит нормализацию данных по переданным значениям среднего и стандартного отклонения. В данном конкретном случае они была равны [0.485, 0.456, 0.406] для среднего и [0.229, 0.224, 0.225] для стандартного отклонения. Данные значения являются наиболее подходящими и часто используемыми, они получены на основе анализа огромного количества изображений и указаны в документации pytorch, что доказывает целесообразность применения их в данной модели для увеличения производительности вычислений.

## **Архитектура и общая логика**

Так как самонормализация не привязана к какой-то определенному типу нейронных сетей и их архитектурам было принято решение протестировать это на нескольких типах сетей, а именно на сверточной сети и многослойном перцептроне.

В качестве инструментов проверки эффективности реализации самонормализации были выбраны следующие метрики:

* Accuracy. Accuracy (точность) - это метрика, которая используется для измерения производительности модели машинного обучения в задачах классификации. Она показывает, какой процент объектов в наборе данных был классифицирован правильно.

Для расчета accuracy, сначала необходимо построить модель машинного обучения. Затем, используя набор данных, модель применяется для классификации объектов, и затем сравнивается с истинными метками классов. Точность рассчитывается как отношение количества правильно классифицированных объектов к общему числу объектов в наборе данных:

Например, если модель правильно классифицировала 90 объектов из 100 в тестовом наборе данных, то точность равна 90%. Чем выше значение точности, тем лучше производительность модели.

Важно отметить, что точность может быть недостаточно информативной метрикой для оценки производительности модели в случаях, когда классы в наборе данных несбалансированы. Например, если в наборе данных 95% объектов относятся к одному классу, то модель может достичь высокой точности, просто всегда относя новый объект к этому классу, но в то же время она будет плохо работать на других классах. В таких случаях следует использовать другие метрики, такие как precision, recall или F1-score, которые учитывают сбалансированность классов в наборе данных.

Так как ранее было установлено, что в обоих дасетах данные сбалансированы практически идеально, вышеуказанные метрики precision, recall или F1-score не использовались.

Для получения более полной картины происходящего в работе accuracy измерялось для обеих выборок: и для обучающего набора, и для тестового.

* График функции ошибки. График функции ошибки — это график, показывающий, как значение функции ошибки изменяется в зависимости от количества итераций обучения модели. Функция ошибки в машинном обучении является мерой того, насколько хорошо модель предсказывает значения целевой переменной на обучающих данных.

На графике функции ошибки по оси X отображаются итерации обучения, а по оси Y - значение функции ошибки. Обычно график функции ошибки имеет форму "воронки", сужающейся по мере увеличения количества итераций обучения. Это связано с тем, что на начальных этапах обучения модель еще не настроена оптимально и совершает большие ошибки, а по мере улучшения параметров модели, значение функции ошибки уменьшается.

Анализ графика функции ошибки позволяет определить, насколько хорошо модель обучается на данном наборе данных, и выявить проблемы, которые могут возникнуть в процессе обучения. Если значение функции ошибки продолжает уменьшаться на тестовых данных, то это может свидетельствовать о переобучении модели. Если же значение функции ошибки не уменьшается на обучающих данных или на тестовых данных, это может указывать на недообучение модели.

График функции ошибки также может помочь выбрать оптимальные гиперпараметры для модели, такие как скорость обучения, количество эпох или количество скрытых слоев в нейронной сети. При правильном выборе гиперпараметров модели функция ошибки должна уменьшаться на обучающих и тестовых данных, что является признаком оптимальности модели.

Как и было установлено ранее, в самонормализующихся нейронных сетях используются следующие функции:

* SELU. SELU (Scaled Exponential Linear Unit) - это активационная функция в машинном обучении, которая была предложена в 2017 году в статье "Self-Normalizing Neural Networks" автором Гюнтером Клоксом и его коллегами. Эта функция активации была разработана для решения проблемы затухания градиентов в глубоких нейронных сетях.

SELU имеет следующую формулу:

где λ и α - параметры, которые могут быть подобраны при помощи оптимизации. В данной работе их значения были приняты за α=1.673… и λ=1.050…. Данные значения являются одними из наиболее оптимальных согласно множеству исследований.

SELU имеет некоторые преимущества по сравнению с другими активационными функциями, такими как ReLU и sigmoid. В частности, SELU может обеспечить самонормализацию нейронной сети, то есть каждый слой будет иметь нормализованное распределение активаций, что уменьшит вероятность затухания градиентов и ускорит обучение.

Также SELU имеет свойство сохранения нуля, что означает, что если входной сигнал имеет значение меньше нуля, то и выходное значение функции также будет меньше нуля. Это позволяет избежать проблемы "dead ReLU", когда нейроны умирают из-за большого количества отрицательных значений входных сигналов.

Однако использование SELU имеет и недостатки. Она может работать только с нейронами, использующими линейную активацию, а также требует определенного распределения входных данных и гиперпараметров для эффективной работы. В целом, SELU – это одна из возможных активационных функций, которые могут быть использованы в машинном обучении, и ее применение зависит от конкретной задачи и архитектуры модели.

* AlphaDropout. AlphaDropout - это модификация стандартной Dropout-регуляризации, которая была предложена в статье "Alpha Dropout" авторами Дропаутиано, Крисом Капуром и Юргеном Шмидхубером в 2018 году.

AlphaDropout работает аналогично Dropout-регуляризации, но вместо простого обнуления случайно выбранной доли нейронов во время обучения, она обнуляет сэмплы из случайно выбранных признаков на каждом слое сети таким образом, что среднее значений данных на текущем слое становится равным 0, а стандартное отклонение – 1.

При обнулении случайных признаков AlphaDropout генерирует случайную маску из распределения Гаусса с параметрами, которые вычисляются на основе статистических свойств признаков сети. Таким образом, AlphaDropout способна адаптироваться к данным, которые имеют разные характеристики и различную корреляцию между признаками.

AlphaDropout позволяет эффективно справляться с проблемой переобучения, снижая вероятность переобучения и улучшая обобщающую способность модели. Она может быть применена к различным архитектурам нейронных сетей и применяется в различных задачах машинного обучения, включая обработку изображений, распознавание речи и обработку естественного языка.

Однако, как и любая регуляризация, AlphaDropout имеет свои недостатки и может привести к ухудшению точности модели, если не подобрать правильные параметры. Кроме того, AlphaDropout требует дополнительных вычислений и, следовательно, может замедлять скорость обучения.

# **Многослойный перцептрон**

Первой сетью для проверки был выбран многойслойный перцептрон. Его реализации с самонормализацией и без нее принципиально не отличались. Единственным отличием были другие функции активации и дропаута.

Ниже на рисунках 9 и 10 приведены архитектуры многослойных перцептронов без самонормализации и с соответственно.

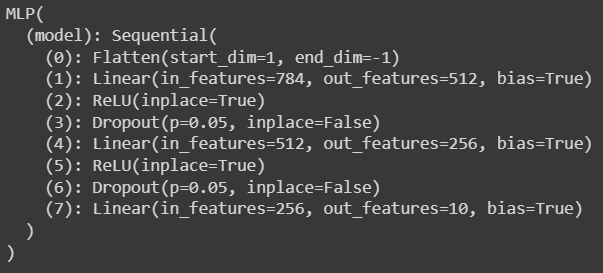


Рисунок 9 – Архитектура многослойного перцептрона без самонормализации.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Архитектура многослойного перцептрона с самонормализацией.

Как можно заметить по рисункам 9 и 10, первым слоем сети является слой Flatten, чьим предназначением является изменение входных данных таким образом, что из n-мерного тензора они становятся одномерным вектором. Данный слой необходим, так как слой Linear не умеет работать не с одномерными данными.

Следующим слоем является слой Linear из 784 признака в 512. Число 784 выбрано не случайно. Так как для многослойного перцептрона используется датасет MNIST, на вход ему подается изображение размером 28x28, которые после слоя Flatten становится одномерным тензором размером 28\*28=784.

После этого идет слой с функцией активации: ReLU для обычного многослойного перцептрона и SELU для многослойного перцептрона.

Затем следует с дропаутом. Аналогично предыдущему – Dropout для обычного многослойного перцептрона и AlphaDropout для многослойного перцептрона.

Предыдущие три слоя повторяются еще один раз. Единственным отличием являются параметры слоя Linear, которые переводит уже 512 признаков в 256.

Последним слоем является слой Linear, который переводит 256 признаков в 10, которые по сути равны вероятностям принадлежности данного сэмпла к одному из 10 классов.

После длительного подбора и анализа гиперпараметров и других параметров были выбраны следующие значения:

* Скорость обучения – 0.001
* Количество эпох – 50.
* Вероятность дропаута – 0.05.

Таким же образом были выбраны оптимизатор и функция ошибки.

В качестве первого выступает оптимизатор, реализующий алгоритм Adam. А в качестве второй – категориальная CrossEntropyLoss.

Результаты работы сетей представлены на график, изображенных на рисунке 11 ниже. Как можно заметить, эффективность работы многослойного перцептрона с самонормализацией превышает таковую у обычного примерно на 10%. Более того, судя по кривым значения функции ошибки, темпы обучения сети с самонормализацией заметно превышают таковые у обычной сети: на 15 эпохе обучения перцептрон с самонормализацией был обучен более чем в 2 раза лучше, чем перцептрон без нее.

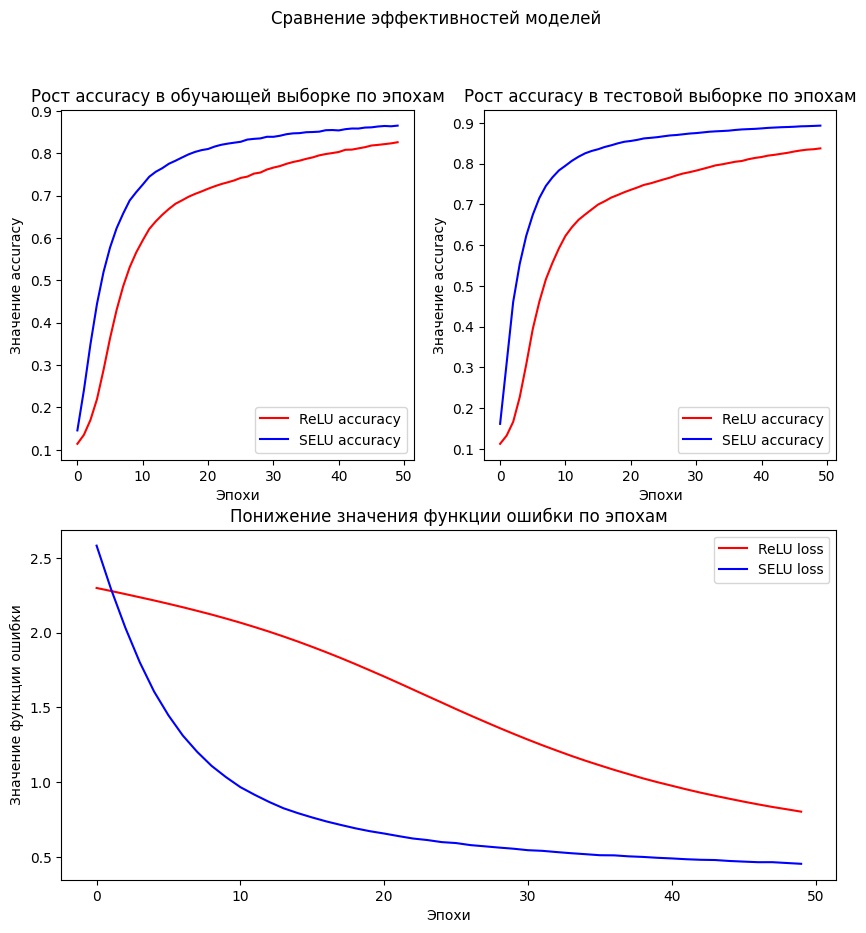


Рисунок 11 – Графики метрик работы многослойного перцептрона.

# **Сверточная сеть**

Другой сетью для тестирования утверждения эффективности внедрения самонормализации была выбрана сверточная сеть. Архитектура этих сетей представлены на рисунках 12 и 13, расположенных ниже.

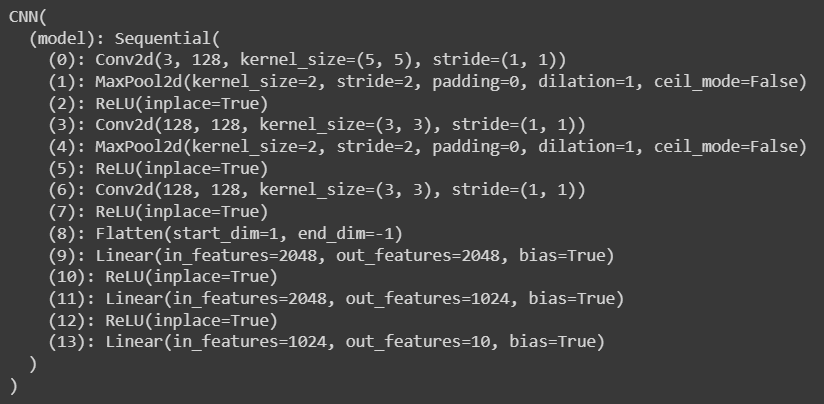


Рисунок 12 – Архитектура сверточной сети без самонормализации.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13 – Архитектура сверточной сети с нормализацией.

Аналогично многослойным перцептронам, которые были рассмотрены ранее, единственными отличиями между двумя сверточными сетями являются функции активации и дропаута.

Первым слоем сети является двумерный сверточный слой Conv2d, с размером фильтра 5x5, 3 входными признаками, 128 выходными и шагом 1.

Следующим слоем является max pooling слой с размером фильтра 2x2 и шагом 2.

После следует функция активации: ReLU или SELU.

Данные три слоя повторяются 2 раза с другими параметрами: размер фильтра сверточного слоя – 3x3, а количество входных и выходных признаков равняется 128.

После этого используется слой Flatten, для перевода двумерного сэмпла в одномерный.

Затем, подобно многослойному перцептрону, идут слои Linear и слои активации, переводящие 128\*4\*4=2048 признаков в 2048, 2048 в 1024, а 1024 в 10.

После длительного подбора и анализа гиперпараметров и других параметров были выбраны следующие значения:

* Скорость обучения – 0.001
* Количество эпох – 20.
* Вероятность дропаута – 0.1.

Таким же образом были выбраны оптимизатор и функция ошибки.

В качестве первого выступает оптимизатор, реализующий алгоритм Adam. А в качестве второй – категориальная CrossEntropyLoss.

Результаты работы сетей представлены на график, изображенных на рисунке 14 ниже. Как можно заметить, эффективность работы сверточной сети с самонормализацией стабильно превышает таковую у обычной примерно на 0.3 для accuracy на обучающей выборке. Более того, судя по кривым значения функции ошибки, темпы обучения сети с самонормализацией стабильно заметно превышают таковые у обычной сети: на всех эпохах обучения сверточаня сеть с самонормализацией была обучена более чем на 0.7 единиц лучше, чем сверточная сеть без нее.

Также можно заметить, что accuracy на тестовой выборке у сверточной сети с самонормализацией постепенно падает. Это можно объяснить тем, что модель переобучается и «запоминает» все возможные варианты из обучающей выборке и не может подстроиться к таковым из тестовой. Это можно исправить подбором других гиперпараметров или архитектуры. Тем не менее, это не было реализовано, так как задачей работы было исследовать эффективность сетей с самонормализацией относительно таких же без нее.

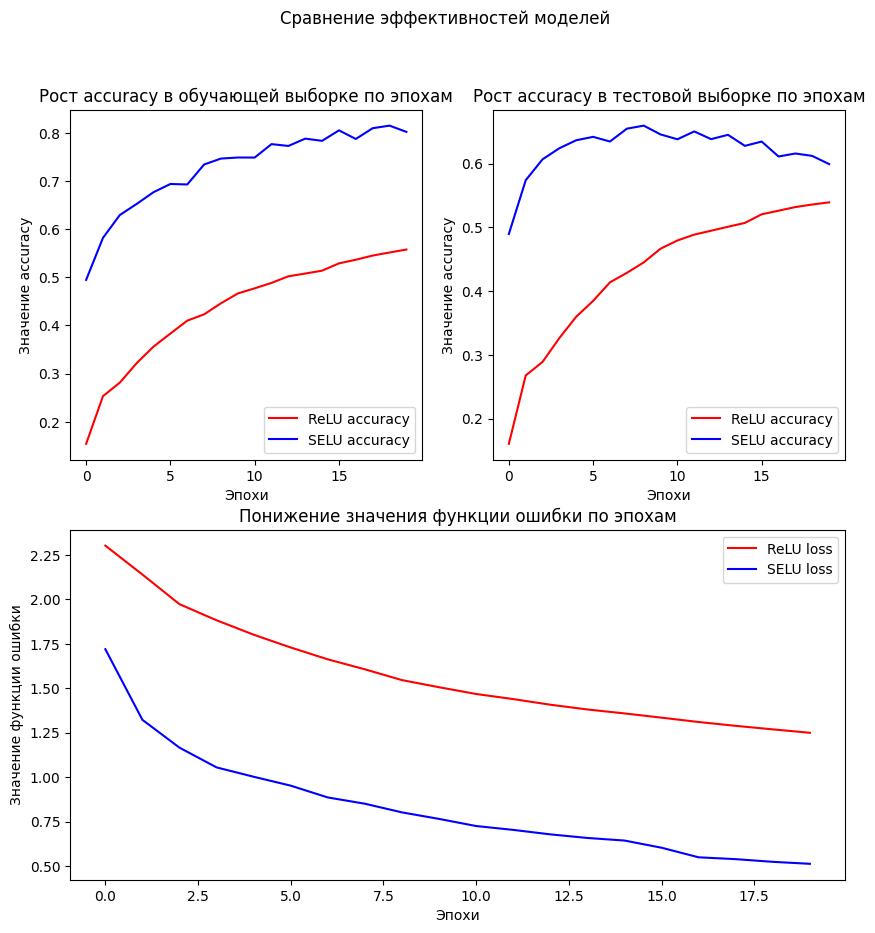


Рисунок 14 – Графики метрик работы многослойного перцептрона.

# **Неэффективный многослойный перцептрон**

Однако стоит заметить, что не всегда самонормализации значительно повышает эффективность работы нейронных сетей. При подборе определенных гиперпараметров или архитектур самонормализация может ухудшить показатели эффективности. В качестве примера была взята нейронные сети с архитектурой, указанной на рисунках 15 и 16.

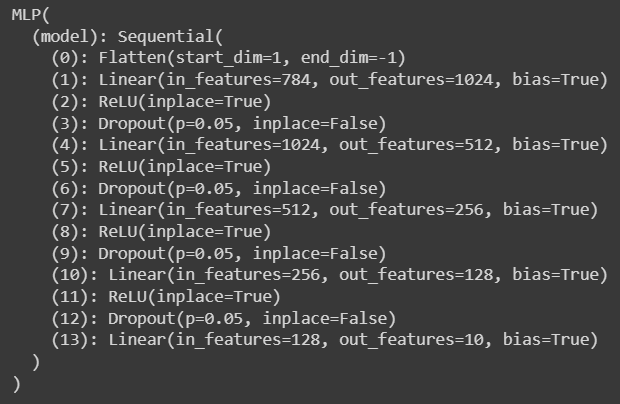


Рисунок 15 – Архитектура второго многослойного перцептрона без самонормализации.

Как можно заметить, данная архитектура является многослойным перцептроном, состоящим из повторяющихся слоев Linear, слоев активации и слоев дропаута.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 16 – Архитектура второго многослойного перцептрона с самонормализацией.

На рисунке 17 показаны результаты работы данных двух нейронных сетей. Как можно заметить, эффективность обычной сети исключительно хороша: точность почти равна 1, а значение функции ошибки – 0. Однако такая же сеть с самонормализацией имеет немного худшие параметры, и хотя она имеет точно такие же хорошие показатели, относительно обычной сети она работает хуже.

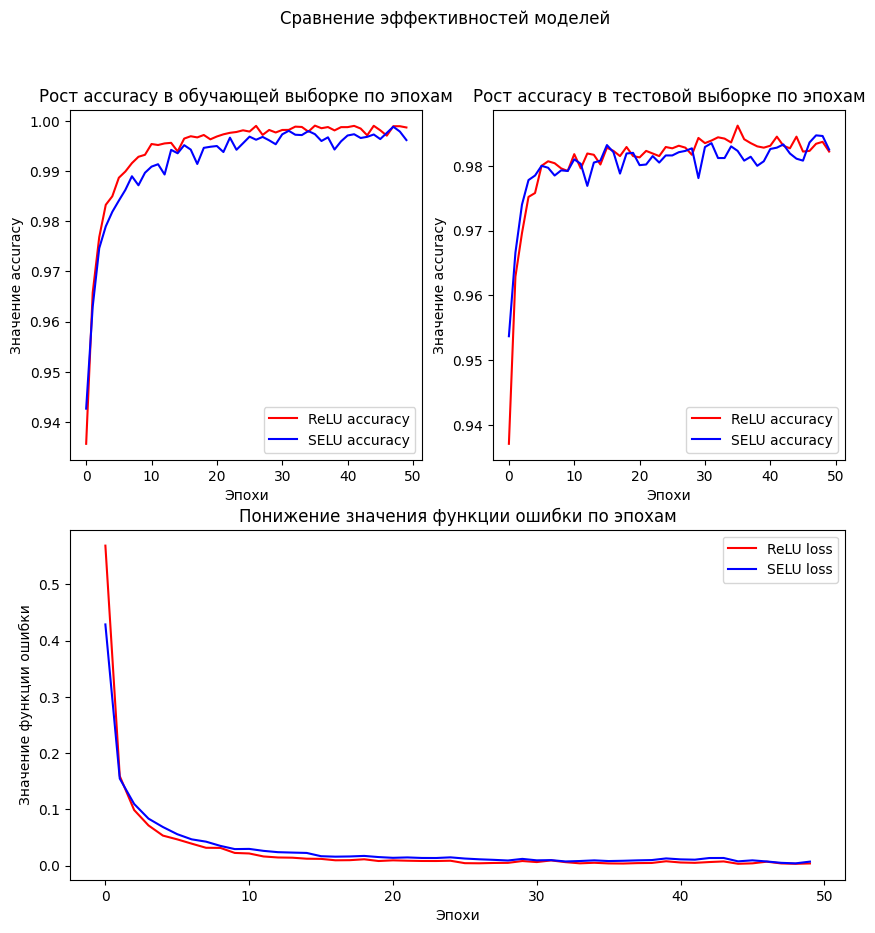


Рисунок 17 – Графики метрик работы второго многослойного перцептрона.

# **Заключение**

В заключении можно отметить, что в данной курсовой работе была рассмотрена тема эффективности самонормализующихся нейронных сетей. Была поставлена задача сравнения эффективности обучения нейронных сетей с самонормализацией и без нее. В процессе работы был проведен анализ литературных источников, а также проведено исследование на датасетах CIFAR-10 и MNIST с использованием нейронных сетей с различными типами нормализации.

Исходя из результатов исследования, можно сделать вывод, что самонормализующиеся нейронные сети показывают более высокую точность и скорость обучения по сравнению с нейронными сетями без нормализации. Это связано с тем, что самонормализация позволяет более стабильно и предсказуемо изменять веса в процессе обучения, что уменьшает проблемы затухания и взрыва градиента.

Таким образом, самонормализующиеся нейронные сети представляют собой эффективный метод решения проблем в процессе обучения и могут применяться в различных областях машинного обучения и искусственного интеллекта. Однако, выбор типа нормализации должен быть сделан исходя из конкретных задач и особенностей данных, а также требует дополнительного анализа и экспериментов.

# **Список литературы**

1. Effectiveness of Self Normalizing Neural Networks for Text Classification.

Avinash Madasu and Vijjini Anvesh Rao Samsung R&D Institute, Bangalore:

<https://arxiv.org/pdf/1905.01338.pdf>

1. Self-Normalizing Neural Networks

Günter Klambauer, Thomas Unterthiner, Andreas Mayr:

<https://arxiv.org/pdf/1706.02515.pdf>

1. SELU vs RELU activation in simple NLP models

<https://www.hardikp.com/2017/07/24/SELU-vs-RELU/>

1. SELU — Make FNNs Great Again (SNN):

<https://towardsdatascience.com/selu-make-fnns-great-again-snn-8d61526802a9>

# **Приложение (исходный код)**

1. Google colab:

https://colab.research.google.com/drive/1HwpxGPbj9kl9W7qLb26Uv4CbPqzOtNao?usp=sharing

1. GitHub:

https://github.com/GlebKulikov/Self-normalizing-neural-networks