Аннотация

Данная работа посвящена обучению бинарных нейронных сетей, поскольку они позволяют сократить требования к вычислительной мощности и объему памяти, что особенно актуально в условиях ограниченных ресурсов. На сегодняшний день бинарные сети не обеспечивают достаточное качество, сравнимое с качеством традиционных вещественных сетей, поэтому разработка более результативных методов их обучения крайне актуальна. В данной работе предлагается вероятностная модель нейронной сети, которая может быть преобразована в бинарную сеть и рассмотрено три способа бинаризации. Экспериментальные результаты показали, что поэлементная бинаризация с последующим дообучением позволяет добиться точности распознавания 87% для задачи распознавания изображений выборки MNIST при точности бинарной модели, обученной стандартными методами составила 59%.

Содержание

Введение

Глубокие нейронные сети успешно выполняют многие задачи, начиная от распознавания видео и заканчивая генерацией текста или диалоговыми системами. Основная траектория развития глубоких нейросетевых моделей – создание новых и усложнение существующих архитектур для повышения точности и решения новых задач. Однако современные глубокие нейронные сети содержат сотни слоев и миллионы обучаемых параметров. Это означает, что во время прямого прохода сети приходится совершать миллиарды арифметических операций и хранить множество промежуточных значений. Однако, когда необходимо использовать искуственный интеллект в смартфонах и носимых устройствах, таких как наручные часы, невозможно удовлетворить эти запросы. Подобные устройства должны быть компактными, легкими и энергоэффективными. Похожая ситуация и со встраиваемыми устройствами, датчиками и камерами. Там могут быть более емкие батареи, но все еще существуют ограничения производительности процессоров. Таким образом, на маломощных устройствах использовать глубокие нейронные сети затруднительно, а небольшие модели дают заметно меньшее качество распознавания [1].

Одно из решений – дискретизация, или квантование, весов и активаций сети. Изначально веса и активации вещественные, берется дискретный набор значений и каждый вес заменяется на ближайшее к нему значение из дискретного набора. Самым радикальным вариантом дискретизации является бинаризация, в этом случае дискретный набор состоит всего из двух значений: -1 и 1. В первую очередь, при полностью бинаризованных весах и активациях можно избавиться от вычислительно-затратной операции умножения вещественных чисел и использовать лишь битовые операции [2]. Кроме того, благодаря бинаризации существенно сокращается объем используемой памяти – в 32 раза в случае вещественных весов и активаций одинарной точности, так как каждое вещественное число, занимающее 32 бита, представляется одним битом.

Основной проблемой бинарных нейронных сетей является их обучение: стандартный метод обратного распространения ошибки с последующей оптимизацией градиентными методами не подходит для обучения, поскольку сам процесс квантования недифференцируем [2]. В вещественном случае градиентный спуск работает, так как все операции дифференцируемы, а веса можно корректировать непрерывным образом. В дискретном же случае некоторые функции перестают быть дифференцируемыми, а кроме того веса могут изменяться лишь на дискретные величины. Поэтому для бинарных сетей активно развиваются специальные методы обучения.

Существующие методы обучения бинарных сетей можно разделить на две большие группы. Методы первой группы осуществляют параллельное обучение бинарной и вещественнозначной сетей, методы второй обучают вероятностную модель сети, где каждое значение равно $+1$ или -1 с некоторой вероятностью.

В работе рассматривается метод из второй группы. Методы этой группы выглядят перспективно и показывают отличные результаты по сравнению с вещественными сетями. Основной их минус – плохая теоретическая обоснованность и, иногда, вычислительная сложность. В работе предлагается модель наивной бернуллиевской сети, которая является непрерывной аппроксимацией бинарной сети. Эта модель может улучшить методы обучения бинарных сетей, при этом не делая новых теоретических допущений.

Результаты работы были представлены на 65-й Всероссийской научной кон-

ференции МФТИ [21]

1 Подходы к обучению бинарных сетей

Бинарной сетью называется такая модель нейронной сети, у которой веса равны +1 или -1, а функции активации являются функциями знака. Покажем, как в такой модели будет выглядеть полносвязный слой бинарной сети, одними из первых его предложили в работе [22]. Для такого слоя на вход поступает вектор x размера N. Затем этот вектор умножается на бинарную матрицу весов W. Эта матрица, как и входной вектор, состоит из элементов, равных 1 или -1. К произведению применяется пороговая функция активации (например, sign), чаще всего это функция знака, доопределенная единицей в нуле.

$$x_{out} = \text{sign}(x_{in}W) \tag{1}$$

В результате итоговый вектор бинаризуется и получается бинарный вектор, который обозначен z^b. Схема слоя показана на 1

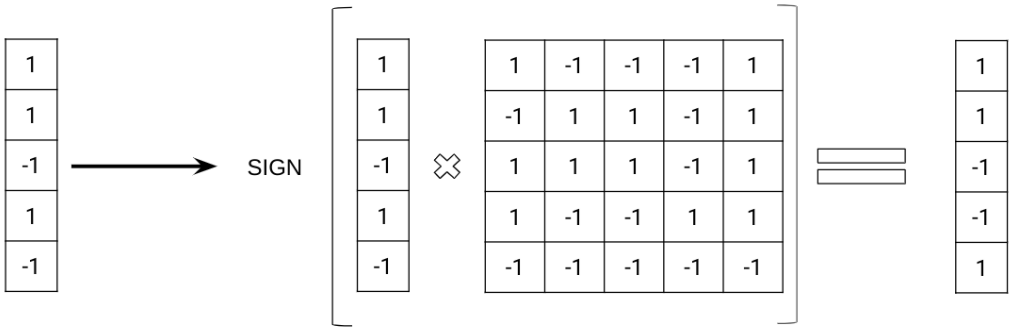

Рис. 1: Классический полносвязный бинарный слой

Благодаря бинаризации для вычислений внутри сети можно использовать битовые операции, которые дают существенную экономию в затратах на вычисления, по сравнению с вещественными. К тому же бинарные веса занимают на порядок меньше памяти. Таким образом бинарная архитектура дает ощутимую экономию в ресурсах. Однако есть и проблемы – классические подходы к обучению с градиентным спуском тут не работают: ведь при дискретизации исчезает дифференцируемость функций, а также бинарные веса невозможно изменять непрерывным образом. Для обучения бинарных сетей (BNN) существует множество методов, среди которых можно выделить две большие группы. Методы первой группы осуществляют параллельное обучение бинарной и вещественнозначной сетей. В этих методах на основании классической вещественной сети строится аналогичная сеть, но с уже бинаризованными весами и активациями. Затем по выходному результату осуществляется корректировка вещественных весов. Методы второй группы обучают вероятностную модель сети и на её основе строят бинарную. В этом случае сеть имеет вещественные веса между 0 и 1, где каждый вес соответствует вероятности того, что в бинарной модели этой же сети находится значение +1.

1.1 Обучение бинарных весов параллельно с вещественными

Рассмотрим семейство алгоритмов обучения BNN, которые позволяют параллельно обучать вещественные и бинарные веса. Этот подход был предложен

Hubara et al. в [3], а одна из его модификаций описана в статье [4].

Для прямого прохода авторы предлагают квантовать веса и активации. А в обратном проходе вычисляются полноразрядные градиенты, используя градиенты бинарных весов и активаций. Затем веса и параметры обновляются.

Darabi et al. [4] рассмотрели три улучшения этого подхода. Во-первых, вместо использования функции знака в качестве функции активации, они предлагают аппроксимировать её с помощью сигмоидальной функции и дополнительного гиперпараметра. Это позволяет настраивать наилучшую аппроксимацию функции знака, которая при этом является дифференцируемой.

Во-вторых, они вводят дополнительный параметр в функцию регуляризации, который также можно обучать. Этот параметр вычисляется динамически в процессе обучения через статистики весов. В классическом случае цель регуляризации заключается в предотвращении переобучения, например, через уменьшение модулей весов. Однако в этом случае регуляризация используется для приближения вещественных весов к бинарным значениям -1 или +1 и штрафу за удаление от них.

В-третьих, с помощью регуляризации модифицируют функцию потерь для бинарного случая и делают её обучаемой. К классической функции потерь авторы прибавляют сумму функций регуляризации по слоям, домноженную на параметр, который также можно настраивать и обучать для повышения точности распознавания.

Эти улучшения были протестированы на выборках CIFAR-10 [5] и ImageNet [6], и бинарные сети, построенные на основе архитектур VGG [7] и AlexNet [8], показали сопоставимые результаты с вещественнозначными сетями. Сеть на основе VGG показала точность в 91%, а сеть на основе AlexNet – 86%. Это свидетельствует о том, что BNN-сети могут успешно использоваться в задачах визуальной классификации.

В [9] представлена интересная модификация алгоритма обучения бинарной нейронной сети параллельно с вещественной. Основная идея заключается в том, чтобы представить матрицы весов вещественных и бинарных нейронов в виде векторов, которые задают некоторые подпространства. Затем, путем нахождения параметрического подпространства, базис которого задает переход между вещественными и бинарными подпространствами, авторы обучают бинарную нейронную сеть.

Для построения базиса использовались два подхода. Первый – случайная инициализация, заключающаяся в использовании случайных величин из стандартного нормального распределения. Однако этот способ оказался вычислительно затратным и не дал значительного улучшения качества работы сети. Второй подход – ортогональная инициализация – основывался на идее того, что похожие веса отвечают за похожие представления. При использовании ортогональной матрицы сохраняется зависимость и независимость между векторами подпространств, что улучшает работу нейронной сети. Процесс обучения начинается с инициализации случайной ортогональной матрицы, затем модель обучается в течение нескольких эпох, после чего обучается базис перехода градиентными шагами.

В работе рассматривается вычисление базиса для отдельного сверточного слоя независимо от остальной сети. Каждый фильтр перед бинаризацией представляется в виде линейной комбинации базисных векторов. В статье эмпирически выводится такой вид базиса, который помогает свести к минимуму и кросс-энтропию, и ошибки квантования.

Эксперименты проводились на наборах данных CIFAR-10, CIFAR-100 и ImageNet с использованием моделей ResNet-18 [10] и WRN-22 [11]. Лучший результат, достигнутый авторами, составил 93% при использовании WRN-22 на датасете CIFAR-10. Эта работа дает новое видение в использовании бинарных нейронных сетей и предоставляет новые методы для их обучения.

Кроме того, в работе проводится анализ результатов для разных типов инициализации базисных векторов и различных параметров обучения. Было установлено, что обучение базиса перехода положительно влияет на итоговую точность модели и на процесс сходимости.

В целом, результаты данного исследования подтверждают, что использование бинарных сетей может значительно уменьшить вычислительные затраты, при этом сохраняя высокую точность предсказаний.

В докладе Wei Tang и его коллег [12] была выдвинута гипотеза о том, что бинарные нейронные сети показывают низкое качество распознавания изображений по сравнению с вещественными нейронными сетями, потому что их учат тем же способом, что и обычные сети. Это приводит к тому, что некоторые параметры, такие как скорость обучения, которые подходят для вещественных сетей, не работают хорошо для бинарных сетей.

В своей работе исследователи проанализировали процессы обучения бинарных нейронных сетей на больших выборках и выявили ошибки, которые могут приводить к низкому качеству распознавания. Они предложили несколько способов решения этих проблем, включая понижение скорости обучения для бинарных сетей, использование адаптивного масштабирования и новое выражение для регуляризации.

Кроме того, авторы доклада предложили два варианта обучения бинарных сетей: с высоким сжатием, при котором уделяется основное внимание экономии памяти, и с высокой точностью, при котором целью является улучшение качества распознавания. Исследование показало, что использование предложенных авторами методов позволило повысить качество распознавания изображений на выборке AlexNet более чем на 20% и повысить степень сжатия в 3 раза по сравнению с другими методами, такими как BNN [3], XNOR-net [13] и DoReFa [14]. Таким образом, исследование Wei Tang и его коллег внесло важный вклад в развитие бинарных нейронных сетей.

У рассмотренных методов имеется ряд преимуществ: в большинстве случаев они просты для реализации и достигают своей цели – существенной оптимизации ресурсов необходимых для использования сети. Однако есть и недостатки: методы данной группы все еще нуждаются в основательной адаптации к каждой конкретной задаче с подбором рабочих стратегий и параметров.

1.2 Вероятностные подходы

Давайте рассмотрим методы обучения нейронных сетей, основанные на вероятностном подходе. В таких методах веса и активации принимают значения +1 или -1 с некоторой вероятностью, и распределения пересчитываются на каждом шаге. Однако проблемой такого подхода является необходимость рассчитывать распределение параметров на следующем слое, что требует большого объема вычислений.

Чтобы решить эту задачу, часто используют аппроксимацию суммы, представляющей вероятность. Для этого могут применяться различные методы в зависимости от размера слоя сети. Например, в [15] для малоразмерных слоев

используется CP-декомпозиция тензора вероятностей: тензор $\mathcal{A} \in \mathbb{F}^{I_0 \times I_1 \times \ldots \times I_C}$ представляет собой совокупность многомерных наблюдений, организованных в M-размерный массив, где $M = C + 1$, $\mathbb{F}$ – пространство тензоров, I_k – k-ая размерность тензора, C – количество размерностей тензора. Каждый тензор может быть представлен с подходящим размером R в виде линейной комбинации r тензоров ранга 1:

$$\mathcal{A} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_{0,r} \otimes \mathbf{a}_{1,r} \otimes \mathbf{a}_{2,r} \cdots \otimes \mathbf{a}_{c,r} \otimes \cdots \otimes \mathbf{a}_{C,r}, \tag{2}$$

где $\lambda_r \in \mathbb{R}$ и $\mathbf{a}_{m,r} \in \mathbb{F}^{I_m}$, где $1 \leq m \leq M$.

Для средних слоев – аппроксимация через быстрое преобразование Фурье, а для больших слоев можно использовать центральную предельную теорему (ЦПТ) и считать, что параметры следующего слоя будут распределены нормально.

Тем не менее, использование ЦПТ на скрытых слоях сети может казаться сомнительным. Однако экспериментальная проверка показала, что это возможно [15]. В других исследованиях для решения данной проблемы применяются репараметризация и функция Гамбеля [16], [17], которые вводят небольшие смещения в распределения весов. После применения этих операций можно сгенерировать конкретные значения из выходных распределений и использовать их в качестве входных данных для следующего слоя.

Таким образом, методы обучения бинарных сетей второй группы, основанные на вероятностном подходе, могут быть эффективными, но требуют использования различных методов аппроксимации, в зависимости от размера слоя сети. При этом можно применять как ЦПТ, так и репараметризацию и функцию Гамбеля, чтобы решить проблему расчета распределения параметров на следующем слое.

В [18] авторы предлагают новый метод обучения моделей с бинарными весами и активациями, который использует локальную репараметризацию. Суть метода заключается в применении функции активации с предыдущего слоя к матрице случайных бинарных весов. Это позволяет получить нормальное распределение в соответствии с ЦПТ. Затем на основе этих распределений генерируются конкретные значения, к которым применяется нормализация. Таким образом, используется дискретное распределение весов и входного вектора для получения распределений, а затем находится распределение по активациям с помощью ЦПТ. Полученные значения затем подаются на выход слоя.

Этот метод может быть полезен для обучения бинарных нейронных сетей, так как он позволяет получить более точные результаты в сравнении с другими рассмотренными методами. Результаты исследования могут быть полезными для разработки новых алгоритмов машинного обучения и улучшения существующих.

Другой подход, рассмотренный в [19], основан на вероятностном описании бинарных весов и активаций. Для этого используется априорное и апостериорное распределения весов, а функция потерь минимизирует кросс-энтропию. Однако, для решения этой задачи оптимизации, распределения весов должны принадлежать к экспоненциальному семейству одного вида. Чтобы справиться с этой проблемой, авторы начинают с семейства распределений Бернулли, которое является минимальным экспоненциальным семейством, и затем обучают апостериорные вероятности весов с помощью обратного распространения и функции Гамбеля.

Однако, обучение бинарных нейронных сетей с помощью вероятностных методов может быть достаточно сложным и требовательным к вычислениям. Для решения этой проблемы, в [20], предлагаются аппроксимации реальных распределений вероятностей. Авторы используют два подхода.

В первом подходе, они вычисляют рекуррентную формулу для реальной вероятности, затем аппроксимируют ее средними значениями и упрощают полученную предельную вероятность. Этот метод основан на анализе распределения с использованием метода Монте-Карло и является довольно эффективным. Суть этого метода заключается в том, чтобы провести большое количество случайных выборок из распределения и вычислить их среднее значение. Таким образом, реальная вероятность аппроксимируется средними значениями выборок, что позволяет получить упрощенное выражение для оптимизации вычислений.

Во втором подходе, авторы рассматривают упрощение суммирования в допущении, что число слагаемых достаточно велико. Этот метод называется «асимптотическим» и позволяет получить аналитическое выражение для вероятности, которое можно использовать для оптимизации. Таким образом, этот подход позволяет аналитически решить задачу.

Кроме того, авторы работы [20] предлагают альтернативный подход к обучению бинарных нейронных сетей, который основывается на комбинации аппроксимации и обратного распространения ошибки. Они показывают, что алгоритм можно использовать для обеих видов нейронных сетей: «бинарных» и «непрерывных». Бинарные веса обновляются с помощью аппроксимации, а непрерывные веса обновляются стандартным образом с помощью обратного распространения ошибки.

Методы этой группы выглядят перспективно и показывают неплохие результаты по сравнению с вещественными сетями. Однако их основной минус заключается в плохой теоретической обоснованности и иногда в вычислительной сложности. Тем не менее, эти методы продолжают привлекать внимание исследователей в области машинного обучения.

В целом, методы обучения бинарных нейронных сетей являются активно развивающейся областью исследований.

1.3 Постановка задачи

Цель данной работы состоит в разработке нового метода обучения бинарных нейронных сетей на основе непрерывной аппроксимации вероятностной модели сети, который таким образом будет иметь лучшую теоретическую обоснованность. Это поможет сделать шаг вперед в развитии способов обучения бинарных нейронных сетей и в перспективе позволит использовать сложные модели на пользовательских устройствах.

Для достижения цели выполнены следующие задачи:

- создать модель, которая будет непрерывной аппроксимацией бинарной нейронной сети и метод её обучения;
- предложить и исследовать различные способы бинаризации данной модели.

2 Обучение бинарных нейросетевых моделей с помощью непрерывных аппроксимаций

2.1 Наивная бернуллиевская модель

В этой работе предлагается модель, аппроксимирующая бинарную сеть, и метод её обучения. Сначала обучается вещественная сеть, веса которой отражают вероятности того, что соответствующие веса в бинарной сети равны $+1$.

При прямом проходе каждому слою на вход приходит вектор вероятностей, при этом предполагается, что элементы входного вектора – независимые случайные величины. Для каждого значения выходного вектора вычисляется свое распределение и в итоге получается матрица распределений. Затем осуществляется переход к следующему слою, в котором производятся аналогичные операции. При обратном распространении ошибки вычисляется градиент и изменяются вероятности. Поэтому модель получила название наивной бернуллиевской сверточной сети.

Предложенная модель, во-первых, использует непрерывные веса и дифференцируемую функцию активации. Во-вторых, предлагаемая версия сети является аппроксимацией бинарной сети, так что её можно привести к бинарному виду непрерывным переходом. Таким образом, рассматриваемая модель является промежуточным шагом для обучения бинарных нейронных сетей.

2.1.1 Полносвязный слой

В предлагаемой модели полносвязного слоя на вход поступает вектор вероятностей $\tilde{X}$ размера N, каждый элемент которого соответствует вероятности получить 1 в бинаризованном векторе X^b. Компоненты бинарного вектора можно получить из компонент вещественного по формуле

$$X_i^b = \begin{cases} 1, \text{ если } X_i \geq 0, \\ -1, \text{ если } X_i < 0. \end{cases} \tag{3}$$

Во внутренних слоях сети вычисления происходят без дополнительных операций, однако для первого слоя необходимо произвести некоторую «подготовку», так как на вход сети поступает классический вектор. Сначала он бинаризуется к виду X_i^b, а затем строится вектор вероятностей по следующей формуле:

$$\tilde{X}_i = \begin{cases} 1, \text{ если } X_i^b = 1, \\ 0, \text{ если } X_i^b = -1. \end{cases} \tag{4}$$

Матрицу весов размера $M \times N$ обозначим как W. Ей соответствует матрица вероятностей $\tilde{W}$, где каждый вес отражает вероятность того, что соответсвующее значение в исходной марице равно $+1$. Инициализируем матрицу вероятностей случайным шумом, распределенным нормальным образом вокруг значения 0.5 с дисперсией 0.5, обрезая значения больше 1 или меньше 0.

Далее нужно рассчитать распределение выходного вектора слоя. В классическом случае выходной вектор Z вычисляется по входному вектору X путем умножения на матрицу весов W_{ij} по формуле:

$$
\begin{aligned}
Z_i &= \sum_{j=1}^{N} X_j \cdot W_{ij}, \\
V_{ij} &= X_i W_{ji}, \\
Z_i &= \sum_{j=1}^{N} V_{ji}.
\end{aligned}
\tag{5}
$$

В предложенной модели для этого нужно несколько шагов. Первый шаг заключается в вычислении трехмерной матрицы $\tilde{V}$ по формуле:

$$
\tilde{V}_{ij} = \tilde{X}_i \cdot \tilde{W}_{ji} + (1 - \tilde{X}_i) \cdot (1 - \tilde{W}_{ji}). \tag{6}
$$

Данная формула отражает вероятность получить $+1$ при перемножении двух элементов входного вектора и матрицы весов: это либо $+1$ умножить на $+1$ или -1 умножить на -1. Схема первого шага отражена на рисунке 2.

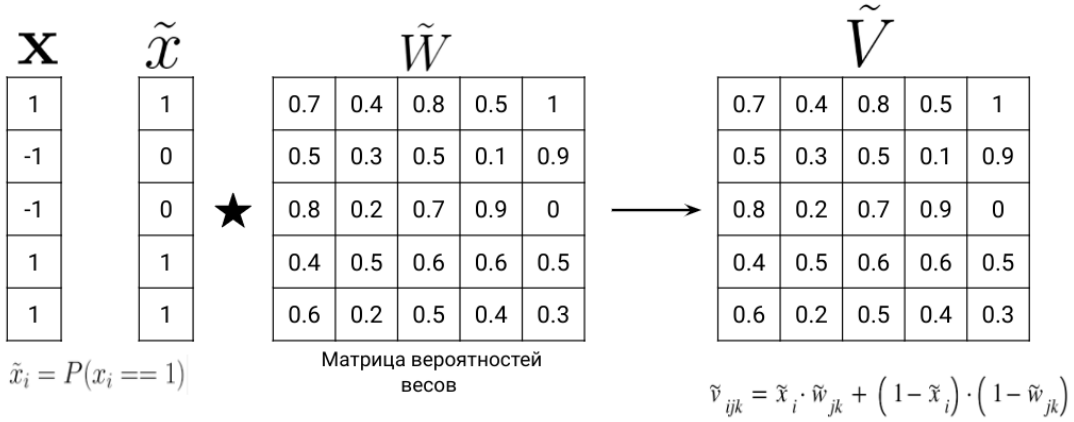

Рис. 2: Первый шаг полносвязного бернуллиевского слоя

Затем вычисляется матрица распределений выходного вектора $\tilde{Z}$, которая определена следующим образом:

$$
\tilde{Z}_{ij} = P(Z_i = 2j - N - 2), \;\; j = 1, .., N + 1. \tag{7}
$$

Это все “значимые” элементы распределения, поскольку

$$
P(Z_i = 2j - N - 1) = 0, \;\; j = 1, .., N + 1, \tag{8}
$$

в силу четности Z, которая меняется при прибавлении следующего бинарного элемента (см. Рис. 3).

Вычислим ее с использованием итеративного подхода. Все элементы $\tilde{V}_{ik}$ задают бернуллиевские распределения. Чтобы получить $\tilde{Z}$ эти распределения суммируются итеративно по индексу k. Для суммирования введем промежуточную матрицу $\tilde{Y}$, вычисление которой будет проходить итеративно для $i = 1, ..., N$, $j = 1, ..., k + 1$, $k = 2, ..., N$:

$$
\begin{aligned}
\tilde{Y}_{i,1} &= 1 - \tilde{V}_{1i} \\
\tilde{Y}_{i,2} &= \tilde{V}_{1i}
\end{aligned}
\tag{9}
$$

Далее рассмотрим процесс вычисления распределений для элементов выходного вектора.

На первом шаге значение будет равно 1 с вероятностью $\tilde{V}_{i1}$, а 0 с вероятностью 0 и -1 с вероятностью $1 - \tilde{V}_{i1}$. Далее значения с вероятностью 0 будут опускаться для компактности выкладок. Возможные значений этой суммы по шагам представлены на Рис. 3.

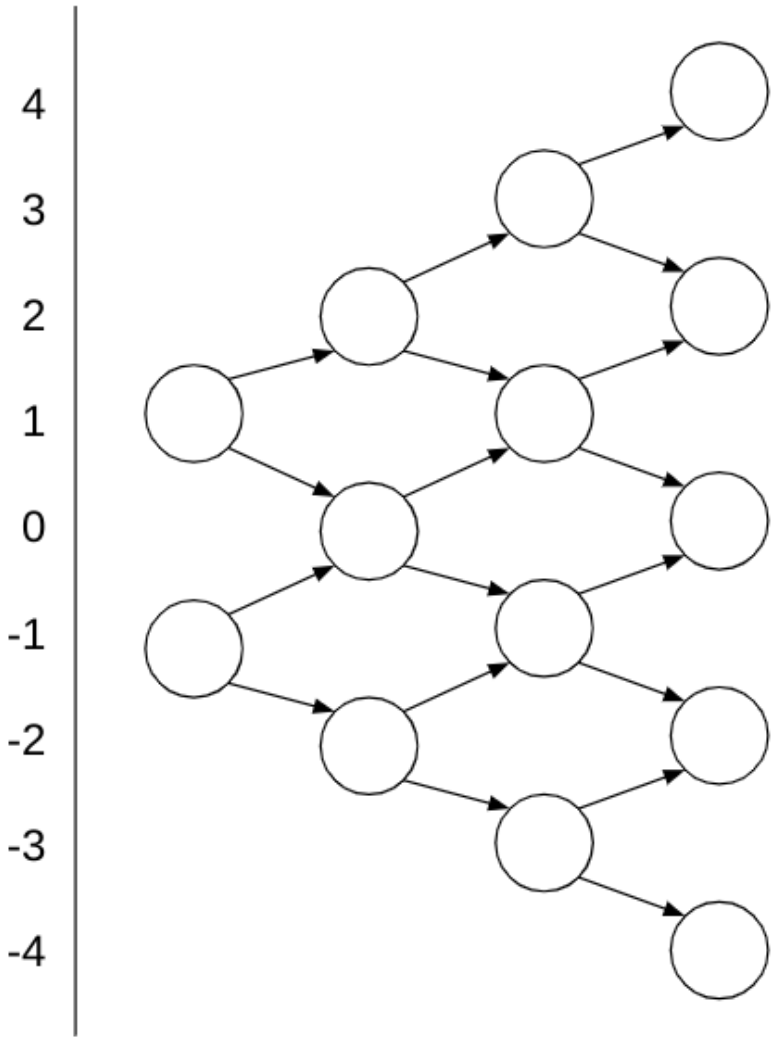

Рис. 3: Схема суммирования бинарных вероятностей ($\tilde{V}_{ij}$) для получения распределения вектора Z_i.

$$\begin{aligned} \tilde{Y}_{i,1,k} &= (1 - \tilde{V}_{ik}) Y_{i,1,k-1} \\ \tilde{Y}_{i,j,k} &= (1 - \tilde{V}_{ik}) Y_{i,j,k-1} + \tilde{V}_{i,k} Y_{i,j-1,k-1} \\ \tilde{Y}_{i,k+1,k} &= \tilde{V}_{ik} Y_{i,k,k-1} \end{aligned} \tag{10}$$

$$\tilde{Z}_{i,j} = \tilde{Y}_{i,j,N} \tag{11}$$

Таким образом, матрица распределений выходного вектора теперь отражает вероятность получить некоторое значение в результате суммирования соответствующего числа 1 или -1. Этот процесс проиллюстрирован на Рис. 4.

Полученная матрица $\tilde{Z}$ имеет размер $M \times (N+1)$ и каждая ее строка задает распределение элемента выходного вектора слоя. Обозначим бинаризованный

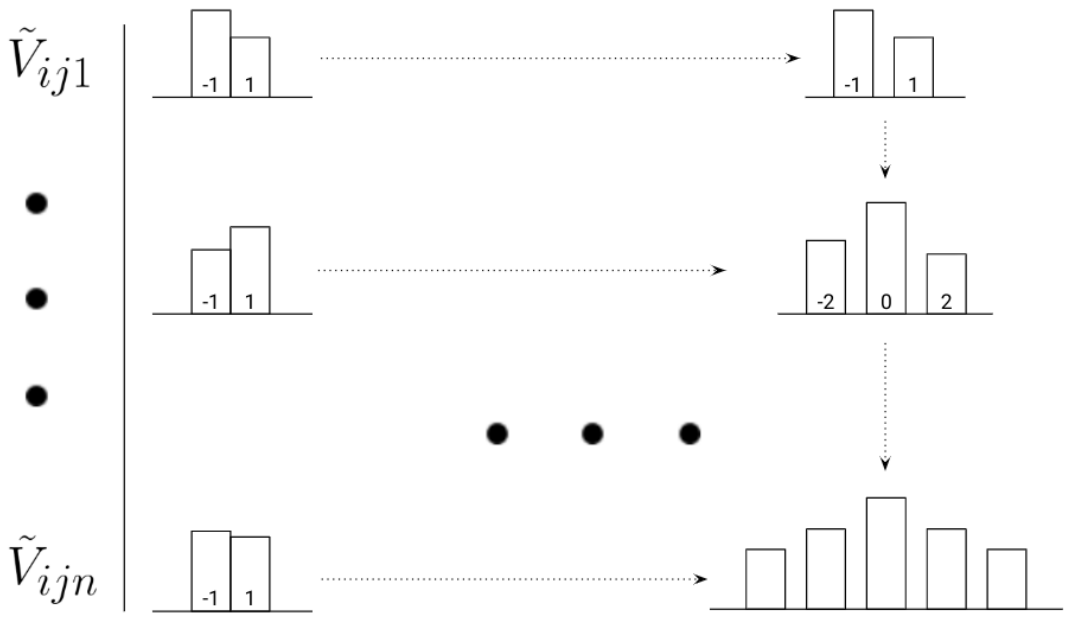

Рис. 4: Вычисление матрицы распределений $\tilde{Z}$.

вектор как $Z^b = sign(Z)$, его распределение

$$\tilde{Z}_i^b = \sum_{j=N/2}^{N+1} \tilde{Z}_{ij}, \tag{12}$$

потому что необходимо получить вероятность того, что в итоговом векторе после бинаризации стоит $+1$, соответственно нужно просуммировать все вероятности, отвечающие неотрицательным значениям.

2.1.2 Полносвязный слой с математическим ожиданием

Если слой последний, то на выходе нужно получить вектор с вещественными, а не бинарными значениями, то есть вектор не вероятностей, а чисел. Со входным вектором производятся все те же операции, что и в полносвязном слое, но вместо суммирования на последнем этапе берется математическое ожидание выходного вектора до бинаризации:

$$y_i = \mathbb{E}Z_i = \sum_j \left(j - \frac{N}{2}\right) \cdot \tilde{Z}_i j \tag{13}$$

Это даст на выходе предсказание модели, основанное на вероятностном распределении результатов.

2.2 Обратное распространение ошибки

Обратное распространение ошибки – это алгоритм обучения нейронных сетей, который используется для оптимизации весов между нейронами [24]. Алгоритм работает по следующей схеме:

1. Прямое распространение – для заданного входного значения нейронной сети вычисляются значения на выходе каждого нейрона. Эти значения

затем передаются в следующий слой, где они снова обрабатываются нейронами, и так далее, до тех пор пока выходные значения всей сети не будут получены.

2. Вычисление функции потерь – после получения выходных значений нейронной сети вычисляется разница между полученными значениями и ожидаемыми значениями. Эта разница определяет, насколько хорошо сеть выполняет задачу, которая ей была поставлена. Функция потерь обычно используется для определения того, как изменение весов между нейронами влияет на результат работы сети.

3. Обратное распространение – на основе вычисленной функции потерь вычисляется градиент функции потерь по всем весам в сети. Этот градиент затем используется для обновления весов между нейронами с целью уменьшения функции потерь. Обновление весов происходит в направлении, противоположном градиенту, чтобы минимизировать функцию потерь.

Некоторые из наиболее распространенных функций потерь [25]:

- Среднеквадратическая ошибка (MSE) – наиболее часто используется для задач регрессии. Она вычисляет среднее значение квадрата разницы между ожидаемыми и предсказанными значениями.

$$\text{MSE}(y, y^{real}) = 1/n \sum_{i}^{n} (y_i - y_i^{real})^2, \tag{14}$$

  y – вектор выходных значений, y^{real} – вектор эталонных выходных значений.

- Средняя абсолютная ошибка (MAE)– также используется для задач регрессии. Она вычисляет среднее значение абсолютной разницы между ожидаемыми и предсказанными значениями.

$$\text{MAE}(y, y^{real}) = 1/n \sum_{i}^{n} |y_i - y_i^{real}|, \tag{15}$$

  y – вектор выходных значений, y^{real} – вектор эталонных выходных значений.

- Категориальная кросс-энтропия (CCE) – наиболее часто используется для задач классификации. Она отражает расстояние между ожидаемым и предсказанным распределением вероятностей классов.

$$\text{CCE}(y, y^{real}) = -\sum_{i}^{n} y_i^{real} \cdot \log(y_i), \tag{16}$$

  y – вектор выходных значений, y^{real} – вектор эталонных выходных значений.

- Бинарная кросс-энтропия (BCE) – также используется для задач классификации, но в случае, когда есть только два класса. Она отражает расстояние между ожидаемым и предсказанным распределением вероятностей двух классов.

$$\text{BCE}(y, y^{real}) = -(y \cdot \log(y^{real}) + (1 - y) \cdot \log(1 - y^{real})), \tag{17}$$

y – вектор выходных значений, y^{real} – вектор эталонных выходных значений.

Вернемся к рассмотрению обратного распространения ошибки в наивной бернуллиевской сети. Вероятностный выходной вектор слоя $\tilde{Z}^b$ представляет собой вектор вероятностей:

$$\tilde{Z}_i^b = P(X_i = 1) \tag{18}$$

Теперь пусть L – некоторая функция потерь. Пусть из обратного распространения ошибки известно, что

$$\frac{\partial L}{\partial \tilde{Z}_i^b} = g_i \tag{19}$$

Тогда из (12)

$$\frac{\partial L}{\partial \tilde{Z}_{ij}} = \begin{cases} g_i, \text{ если } j \geq N/2, \\ 0, \text{ иначе.} \end{cases} \tag{20}$$

Заметим, что

$$\frac{\partial L}{\partial \tilde{V}_{ij}} = \frac{\partial L}{\partial \tilde{Z}_{ij}} \cdot \frac{\partial \tilde{Z}_{ij}}{\partial \tilde{V}_{ij}}. \tag{21}$$

Тогда, если обозначим $x_{i,j} = Y_{i,j,N-1}$, то из (10) следует, что

$$\frac{\partial L}{\partial \tilde{V}_{iN}} = g_i \cdot \left(\sum_{k=1}^{N/2} x_{i,k} - \sum_{k-1}^{N/2} x_{i,k-1} \right) = g_i \cdot x_{i,N/2} \tag{22}$$

Поскольку итоговые распределения $(\tilde{Z}_i)$ не зависят от порядка суммирования (порядка итераций в формуле 10) при их вычислении, формула производной общая для всех элементов. Значение $x_{i,N/2}$ свое для каждого элемента $\tilde{V}$, где вектор x – распределения элемента выходного вектора на предфинальном этапе суммирования распределения матрицы $\tilde{Z}$.

Тогда можно записать общую формулу для производной:

$$\frac{\partial L}{\partial \tilde{W}_{ij}} = \frac{\partial L}{\partial \tilde{V}_{ij}} \cdot \frac{\partial \tilde{V}_{ij}}{\partial \tilde{W}_{ij}} = g_i \cdot x_{i,N/2} \cdot \sum_{k}^{N} (x_k + 1 - x_k) = g_i \cdot x_{i,N/2} \cdot N \tag{23}$$

Чтобы найти $x_{i,N/2}$ нужно решить систему

$$A'x = y, \tag{24}$$

где $y = \tilde{Z}_i^T$, где A – матрица $(N+1) \times N$ вида

$$A = \begin{pmatrix} p & 0 & 0 & \dots & 0 \\ q & p & 0 & \dots & 0 \\ 0 & q & p & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & q \end{pmatrix} \tag{25}$$

Тут введены обозначения

$$p = \tilde{V}_{ik} \tag{26}$$

$$q = 1 - p \tag{27}$$

В этой системе $N+1$ уравнение, можно выбрать N из них, но любая такая система будет плохообусловленной, если p близко к нулю или единице [26]. Можно, конечно, выбирать в одном случае первые N уравнений, а в другом – последние, но при таком подходе нельзя будет использовать векторные операции, что существенно замедлит все вычисления. Поэтому модифицируем систему к виду $A^T Ax = A^T y$.

$$B = A^T \cdot A = \begin{pmatrix} p^2+q^2 & pq & 0 & \dots & 0 \\ pq & p^2+q^2 & pq & \dots & 0 \\ 0 & pq & p^2+q^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & p^2+q^2 \end{pmatrix}$$

Теперь систему уравнений можно будет решить методом прогонки за $O(N)$ [27]. Метод прогонки (или метод простых итераций) -– это эффективный численный метод для решения трехдиагональных систем линейных уравнений. Трехдиагональная система определяется следующим образом:

$$\begin{cases} b_1 x_1 + c_1 x_2 = d_1 \\ a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, \quad i = 2, 3, \dots, n-1, \\ a_n x_{n-1} + b_n x_n = d_n, \end{cases} \tag{28}$$

где a_i, b_i, c_i, d_i – известные коэффициенты и правые части уравнений.

Для решения этой системы методом прогонки необходимо выполнить следующие шаги.

Прямой ход: вычисление коэффициентов α_i и β_i для $i = 1, 2, \dots, n$, где

$$\alpha_1 = -\frac{c_1}{b_1}, \quad \beta_1 = \frac{d_1}{b_1} \tag{29}$$

$$\alpha_i = -\frac{c_i}{a_i \alpha_{i-1} + b_i}, \quad \beta_i = \frac{d_i - a_i \beta_{i-1}}{a_i \alpha_{i-1} + b_i}, \quad i = 2, 3, \dots, n-1 \tag{30}$$

$$\alpha_n = 0, \quad \beta_n = \frac{d_n - a_n \beta_{n-1}}{a_n \alpha_{n-1} + b_n} \tag{31}$$

Обратный ход: вычисление решения x_i для $i = n, n-1, \dots, 1$, где

$$x_n = \beta_n \tag{32}$$

$$x_i = \alpha_i x_{i+1} + \beta_i, \quad i = n-1, n-2, \dots, 1 \tag{33}$$

Таким образом метод прогонки позволяет решить трехдиагональную систему линейных уравнений за $O(N)$ операций.

В рассматриваемой задаче необходимо решить систему уравнений вида:

$$Bx = A^T y \tag{34}$$

Вычислим вектор $A^T y$:

$$A^T y = \begin{pmatrix} p \\ 0 \\ 0 \\ \dots \\ 0 \end{pmatrix} y_1 + \begin{pmatrix} q \\ p \\ 0 \\ \dots \\ 0 \end{pmatrix} y_2 + \dots + \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \\ q \end{pmatrix} y_N \tag{35}$$

где y_i – элементы вектора y.

Вычислим коэффициенты α_i и β_i для прямого хода:

$$\alpha_1 = -\frac{pq}{p^2+q^2}, \quad \beta_1 = \frac{py_1}{p^2+q^2} \tag{36}$$

$$\alpha_i = -\frac{pq}{p^2+2q^2-pq\alpha_{i-1}}, \quad \beta_i = \frac{qy_{i-1}+py_i-pq\beta_{i-1}}{p^2+2q^2-pq\alpha_{i-1}}, \quad i = 2,3,\dots,N-1 \tag{37}$$

$$\alpha_N = 0, \quad \beta_N = \frac{qy_N - pq\beta_{N-1}}{p^2+q^2-pq\alpha_{N-1}} \tag{38}$$

Теперь вычислим решение x с помощью обратного хода:

$$x_N = \beta_N \tag{39}$$

Для $i = N-1, N-2, \dots, 1$:

$$x_i = \alpha_{i+1}x_{i+1} + \beta_i \tag{40}$$

Таким образом, решение системы

$$A^T Ax = A^T y \tag{41}$$

методом прогонки будет иметь вид:

$$x_N = \frac{py_N - pq\beta_{N-1}}{p^2+q^2-pq\alpha_{N-1}} \tag{42}$$

$$x_i = \alpha_{i+1}x_{i+1} + \beta_i, \quad i = N-1, N-2, \dots, 1 \tag{43}$$

Важно отметить, что в общем случае метод прогонки не является устойчивым в отношении возмущений матрицы. Критерием его устойчивости является диагональное преобладание матрицы B. В данном случае

$$p^2+q^2 \geq 2pq, \tag{44}$$

то есть решение системы всегда будет устойчивым.

Также вычислить $x_{N/2}$ можно приближенно с помощью аппроксимации системы (34) по трем центральным уравнениям. То есть

$$B_1 x = t, \tag{45}$$

где

$$B_1 = \begin{pmatrix} p^2+q^2 & pq & 0 \\ pq & p^2+q^2 & pq \\ 0 & pq & p^2+q^2 \end{pmatrix} \tag{46}$$

и вектор t имеет вид

$$t = \begin{pmatrix} y'_{N/2-1} \\ y'_{N/2} \\ y'_{N/2+1} \end{pmatrix}, \tag{47}$$

где $y' = A^T y$.

Аналогично можно рассмотреть более точное приближение по 5 центральным уравнениям.

2.3 Сверточный слой

Сверточный слой – это основной блок в сверточных нейронных сетях [28], которые применяются для обработки изображений, звуковых файлов и других типов данных, важно лишь, чтобы эти данные могли быть представлены в виде 2D-или 3D-тензоров.

Основная идея сверточного слоя заключается в том, чтобы выполнять локальные операции свертки на входном тензоре с помощью набора фильтров (ядра свертки), которые обычно представляются небольшими матрицами весов. Каждый фильтр содержит набор параметров, которые подбираются в процессе обучения сети, чтобы определять, какие признаки входных данных следует обнаруживать.

В процессе свертки фильтр «скользит» по входному тензору и вычисляет скалярное произведение между своими весами и каждой локальной областью входного тензора. Этот процесс повторяется для каждого фильтра на входном тензоре, что приводит к формированию нового тензора, называемого картой признаков.

Операция свертки для ядра и входной карты признаков выражается формулой

$$g(x, y) = w \cdot f(x, y, c) = \sum_{c=1}^{C} \sum_{\Delta x=-a}^{a} \sum_{\Delta y=-b}^{b} w(\Delta x + a, \Delta y + b, c) f(x - \Delta x, y - \Delta y, c), \tag{48}$$

где $g(x, y)$ – результат применения свёртки, w – ядро свёртки, $f(x, y, c)$ – входной тензор, a, b – произвольные границы свёртки, а C – число каналов входной карты признаков.

Сверточный слой позволяет нейронной сети эффективно изучать локальные особенности входных данных и использовать их для создания более сложных и абстрактных представлений в более глубоких слоях сети.

Операцию свертки можно представить как операцию матричного умножения [28]. Можно по аналогии применять все то, что применялось в нашей работе для полносвязного слоя, представленного в разделе 2.1.1.

2.4 Подходы к бинаризации сети

2.4.1 Полная одновременная бинаризация

Первый и самый очевидный способ бинаризовать сеть – взять наиболее вероятное значение каждого элемента матрицы весов. Одновременно для всех весов и слоев:

$$W_{ij} = \begin{cases} 1, \tilde{W}_{ij} > 0.5, \\ -1, \tilde{W}_{ij} \leq 0.5. \end{cases} \tag{49}$$

2.4.2 Послойная бинаризация с дообучением

Второй путь – бинаризовать сеть по слоям. То есть, сначала бинаризовать матрицу весов очередного слоя тем же способом, что и в полной бинаризации, представленной в разделе 2.4.1, а затем дообучить сеть, зафиксировав веса бинаризованного слоя. Для наглядности процесс бинаризации отражен на Рис. 5.

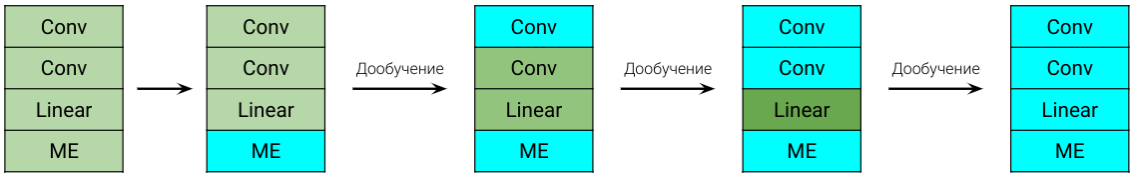

Рис. 5: Послойная бинаризация, зеленым обозначены наивные бернуллиевские слои, синим – бинаризованные слои, conv – сверточный слой, linear – полносвязный слой, ME – полносвязный слой с математическим ожиданием. Предполагается, что сначала бинаризуется полносвязный слой с математическим ожиданием, а затем слои от первого к последнему.

2.4.3 Бинаризация по частям внутри слоя

Третий путь – расширение второго. Теперь бинаризуется не вся матрица весов слоя, а лишь некоторая часть весов, а затем дообучается вся сеть. Этот способ дал гораздо лучшие результаты, как будет показано экспериментально в разделе 3. Он проиллюстрирован на Рис. 6.

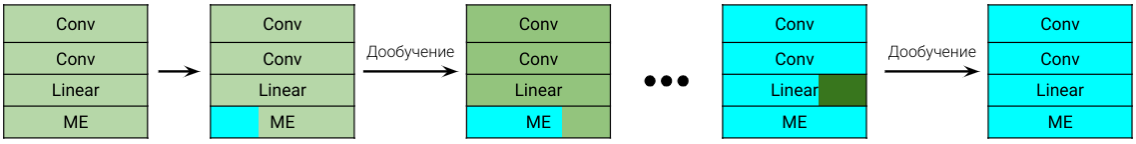

Рис. 6: Поэлементная бинаризация, зеленым обозначены наивные бернуллиевские слои, синим – бинаризованные слои, conv – сверточный слой, linear – полносвязный слой, ME – полносвязный слой с математическим ожиданием. Предполагается, что сначала бинаризуется полносвязный слой с математическим ожиданием, а затем слои от первого к последнему.

2.5 Регуляризация

Регуляризация в сверточных нейронных сетях – это методы, которые применяются для предотвращения переобучения модели, то есть ситуации, когда модель хорошо работает на обучающих данных, но плохо на тестовых [30].

Существуют несколько методов регуляризации в CNN:

1. Dropout: этот метод заключается в том, чтобы случайным образом «выключать» (устанавливать в ноль) некоторые веса нейронов в процессе обучения. Это позволяет сети избежать переобучения и обучаться более устойчиво.

2. L_1 и L_2 регуляризация: эти методы добавляют штрафы на веса модели в функцию потерь, что позволяет уменьшить размер весов и предотвратить переобучение.

3. Аугментация данных: это метод заключается в том, чтобы создавать новые образцы данных на основе существующих. Например, можно поворачивать изображения, изменять их размер или цвета. Это позволяет увеличить количество данных для обучения и сделать сеть более устойчивой к различным вариантам данных.

4. Ранняя остановка: это метод заключается в том, чтобы остановить обучение модели, когда ее производительность на тестовых данных перестает улучшаться. Это предотвращает переобучение и помогает получить модель с лучшей обобщающей способностью.

Все эти методы могут применяться как отдельно, так и в комбинации друг с другом, чтобы получить наилучший результат.

В рассматриваемом случае нужно бинаризовать модель, то есть взять наиболее вероятные значения элементов матриц весов. А значит, бинаризация пройдет тем лучше, чем более уверенна будет сеть в своих весах. Поэтому в модель добавлена регуляризация:

$$L_{reg} = \mathbb{E}(\sigma(\tilde{W}) \cdot (I - \sigma(\tilde{W}))), \tag{50}$$

где I – матрица, состоящая из единиц. Таким образом вероятности весов приблизятся к 0 и 1.

3 Экспериментальные результаты

Эксперименты проводились на языке python с помощью библиотеки PyTorch, позволяющей производить вычисления как на центральном, так и на графическом процессоре. Перед началом экспериментов был загружен набор данных MNIST (Modified National Institute of Standards and Technology) [31] – один из наиболее известных и широко используемых наборов данных для задач классификации изображений. Он состоит из набора рукописных цифр (от 0 до 9), представленных в виде серых изображений размером 28x28 пикселей. Кроме того, использовался оптимизатор Adam [33] для обучения модели.

Затем полученные изображения были бинаризованы с помощью функции:

$$X_i^b = \begin{cases} 1, \text{ если } X_i \geq 0.5, \\ 0, \text{ иначе.} \end{cases} \tag{51}$$

Из стандартных обучающих и тестовых выборок были сформированы подвыборки размером 100 примеры в каждой, по которым далее проводилать оптимизация. Во всех экспериментах использовались функции train и test, которые каждую подвыборку из обучающей выборки подавали на вход сети, затем вычисляли функцию потерь и запускали оптимизацию параметров сети с помощью оптимизатора. Кроме того, функции вычисляли и выводили точность предсказаний модели.

3.1 Аппроксимации градиентов

Точное вычисление градиента является вычислительно-затратным и увеличивает время обучения сети. Поэтому были рассмотрены несколько более простых аппроксимаций градиента. Были заведены четыре модели сети, состоящие из одного полносвязного слоя. В качестве градиента для обратного распространения ошибки в этих сетях использованы:

1. Точное вычисление градиента через решение системы (34).
2. Аппроксимация градиента по трем центральным уравнениям системы (45).
3. Аппроксимация градиента по пяти центральным уравнениям.

В качестве задачи для моделей была сгенерирована случайная матрица, которая должна была быть предсказана сетями. В качестве входа сетям подавались случайные вектора, а в качестве выхода —- результаты умножения этих векторов на сгенерированную матрицу. Таким образом, исследовалось насколько быстро матрица весов сети совпадет с изначально сгененрированной матрицей.

Ниже приведены результаты обучения сетей для предсказания случайно сгенерированной матрицы. На Рис. 7 отражены изменения функци потерь на 100 итерациях.

В Таблице 1 отражено количество итераций, которое потребовалось соответствующим сетям для выучивания сгенерированной матрицы. Можно видеть, что аппроксимации по 3 и 5 центральным уравнениям не оказывают значительного влияния на скорость схождения к решению. При этом при обучении предложенной модели аппроксимация по трем центральным уравнениям дает меньшее время обучения по сравнению с другими способами, как показано в

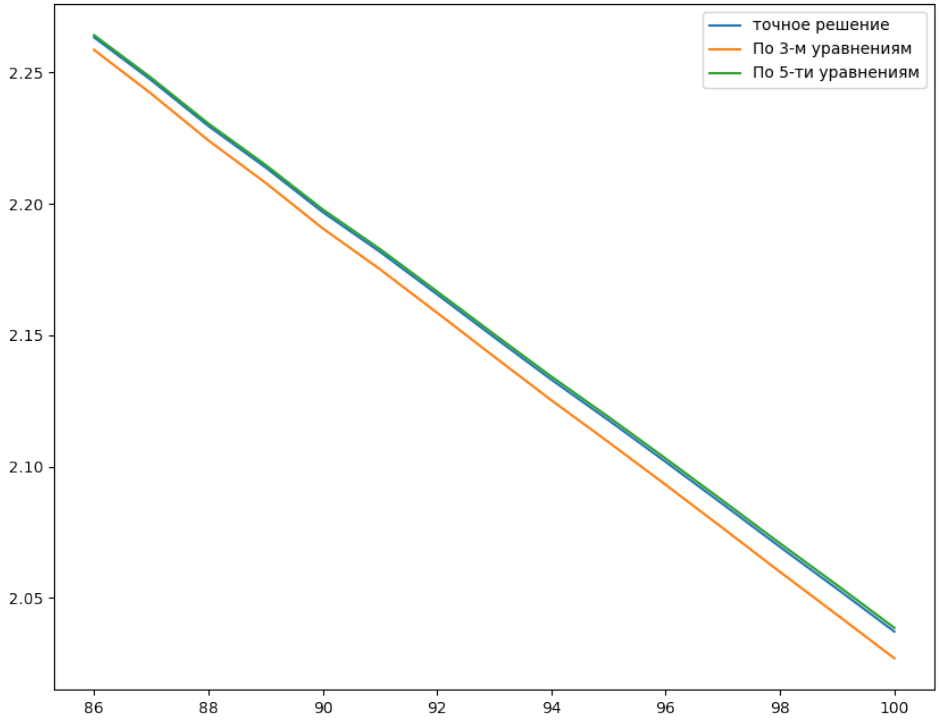

Рис. 7: Зависимость функции потерь (Loss) от числа итераций N для различных аппроксимаций градиента.

Таблица 1: Среднее количество итераций до схождения в зависимости от аппроксимации градиента.

Градиент	**Итераций до схождения**
Точное вычисление	57.7
Аппроксимация по 3-м	63.4
Аппроксимация по 5-ти	59.2

Таблица 2: Среднее время выполнения итерации обучения модели.

Градиент	**Итераций в секунду**
Точное вычисление	0.23
Аппроксимация по 3-м	6.08
Аппроксимация по 5-ти	5.90

Таблице 2. Поэтому в дальнейших экспериментах использовалась аппроксимация по трем центральным уравнениям.

По совокупности параметров для дальнейших исследований была выбрана аппроксимация градиента по 3-м центральным уравнениям. Данный метод существенно сокращает время обучения модели и при этом не сильно теряет в точности относительно точного решения или аппроксимации по 5-ти уравнениям.

3.2 Сверточная наивная бернуллиевская сеть

Для проведения экспериментов использовалась базовая LeNet-подобная [32] архитектура, потому что она является стандартом для начального исследования сверточных моделей. В процессе экспериментов были подобраны наилучшие параметры для архитектуры и результат показан на Рис. 8. Параметры обучения, которые помогли достичь лучших результатов приведены в Таблице 3.

Наивная бернуллиевская модель с вещественными весами при использовании этой архитектуры достигла точности распознавания 97.3%.

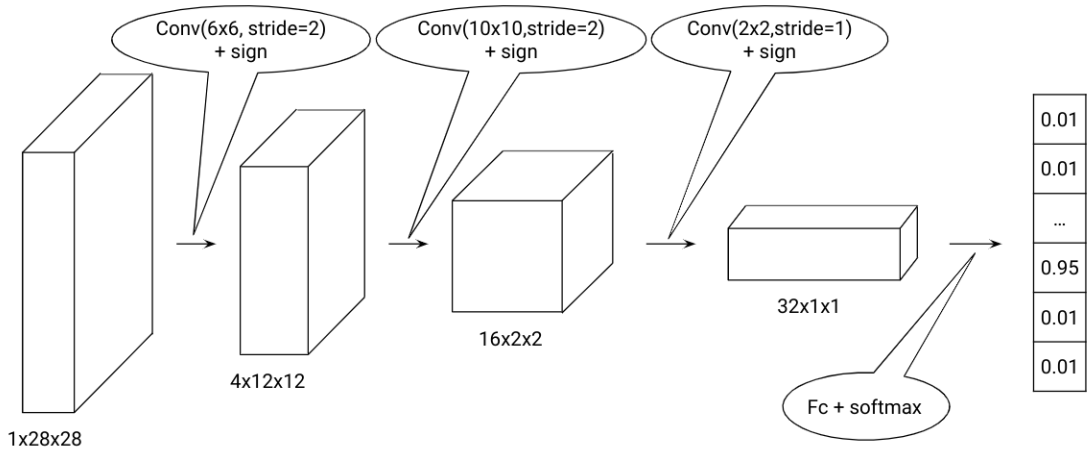

Рис. 8: Предложенная сверточная сеть с 17,984 параметрами. Conv($n \times n$, stride=k) + sign – сверточный слой с ядром $n \times n$, сдвигом k и пороговой функцией активации; Fc + softmax – слой математического ожидания (полносвязный) с функцией активации.

Таблица 3: Параметры обучения сверточной сети.

Параметр	Значение
Оптимизатор	AdamW
Скорость обучения	$1e^{-3}$
Функция потерь	Кросс-энтропия
Число эпох	20

Для контроля качества также была обучена сеть той же архитектуры, но с использованием классических слоев. Параметры обучения были те же. Точность предложенной сети оказалось сравнимым с качеством вещественной сети, которая достигла 98%.

3.3 Обучение референсной модели

В качестве ориентира использована бинарная сеть такой же архитектуры, но с использованием стандартного подхода пробрасывания градиента (STE) [34].

STE – это метод обучения нейронных сетей с помощью обратного распространения ошибки, который позволяет применять функции активации с недифференцируемыми точками.

Такие функции активации, как пороговая функция, могут быть использованы для квантования моделей, но они не являются дифференцируемыми, что затрудняет их использование в обычных методах обучения нейронных сетей.

STE позволяет решить эту проблему, применяя аппроксимацию градиента для недифференцируемых функций. В этом методе при обратном распространении ошибки используется производная функции активации только для прямого прохода по сети, а затем для обновления весов используется «выбранный путь», то есть при проходе обратно по сети градиент просто передается дальше без изменения.

Например, для пороговой функции STE может использовать ее производную только на участках, где она определена, а в остальных случаях просто передавать градиенты без изменения.

Этот подход позволяет использовать недифференцируемые функции активации и создавать более простые модели, сохраняя возможность использования метода обратного распространения ошибки для обновления весов.

Для реализации предложенной модели в сверточном слое использована стандартная функция свертки, после которой к выходному вектору применялась функция бинаризации. Эта функция при прямом проходе применяла к каждому элементу функцию.

$$f(x) = \begin{cases} 1, \text{ если } x > 0, \\ 0, \text{ иначе.} \end{cases} \tag{52}$$

При обратном проходе – просто пробрасывала градиент, не изменяя его.

В последнем слое математического ожидания – аналогично.

Параметры модели показаны в Таблице 4.

Таблица 4: Параметры обучения с помощью STE.

Параметр	Значение
Оптимизатор	Adam
Скорость обучения	$5e^{-5}$
Функция потерь	Кросс-энтропия
Число эпох	10

Её результат – точность распознавания 58%.

3.4 Бинаризация модели

Для каждого вида бинаризации, рассмотренного в разделе 2.4, были построены модифицированные архитектуры и проведены эксперименты. Всего получилось три разных подхода:

- полная одновременная бинаризация всей сети;
- послойная бинаризация с дообучением;
- поэлементная бинаризация, также с дообучением.

3.4.1 Полная бинаризация

При полной бинаризации матрицы при инициализации слоя матрица весов бинаризовалась к значениям 1 и -1. При прямом проходе входной вектор просто умножался на бинаризованную матрицу.

Полная бинаризация сети означает одновременную замену всех весов обученной сети на соответствубщие бинарные. Затем модель проверялась на тестовой выборке. При таком подходе качество распознавания наивной бернуллиевской сети упало с 97% до 10%. Поскольку в MNIST всего 10 вариантов вывода, то 10%, что соответствует случайному угадыванию результата.

Кроме того, проведены эксперименты, где исследовалось зависимость между векторами по мере прохождения ими слоев исходной и бинаризованной сети. Оказалось, что с каждым слоем корреляция уменьшается от полной зависимости до практически полной независимости, как показано в Таблице 5.

Таблица 5: Корреляция между векторами по мере прохождения ими слоев исходной и бинаризованной сети.

Этап работы сети	Корреляция
Входные вектора	1.00
После 1-го сверточного	0.63
После 2-го сверточного	-0.03
После полносвязного	0.09
Выходные вектора	-0.07

Данный эксперимент показывает, что при полной бинаризации обученность сети сходит на нет и такой способ бинаризации не может быть использован.

3.4.2 Послойная бинаризация

В этом случае к обученному слою сети применялась бинаризация, выключался подсчет градиента – таким образом слой «замораживался», а затем дообучались остальные слои сети.

Эмпирически выснилось, что гораздо эффективнее бинаризовать сначала последний слой сети, а затем уже все остальные по порядку с первого сверточного. При дообучении использовались параметры обучения сети, которые показаны в Таблице 6

Таблица 6: Параметры послойной бинаризации

Параметр	Значение
Оптимизатор	Adam
Скорость обучения	$5e^{-4}$
Функция потерь	Кросс-энтропия
Число эпох	5

При использовании этих параметров с послойной бинаризацией с дообучением удалось достигнуть 47% точности распознавания. К сожалению, это оказалось все еще ниже результатов референсной сети.

3.4.3 Поэлементная бинаризация

При поэлементной бинаризации были использованы дополненные версии слоев наивной бернуллиевской сети. Дополнительно к матрице W у этих слоев есть

матрица M такого же размера, которая отражает долю бинаризованных весов в матрице W данного слоя. Элементы матрицы M принимают следующие значения:

$$M_{ij} = \begin{cases} 1, \text{ если соответствующий вес } W_{ij} \text{ должен быть бинаризован,} \\ 0, \text{ иначе.} \end{cases} \tag{53}$$

Представим, что необходимо бинаризовать $t\%$ весов.

Сначала в случайном порядке заменим $t\%$ элементов матрицы M с нуля на единицу. После этого при прямом проходе используется следующая формула для вычисления текущей матрицы весов:

$$\tilde{W} = (1 - M) \cdot \sigma(W) + M \cdot (W > 0) \tag{54}$$

, где σ – сигмоида, используемая в качестве функции активации. Таким образом веса, которым соответствует 0 в матрице M, не изменяются, а те, которым соответствует 1 заменяются на свою бинарную версию.

При обратном распространении ошибки используется следующая формула для вычисления градиента:

$$grad_{next} = grad_{prev} \cdot (1 - M) \tag{55}$$

То есть, градиенты небинаризованных весов пробрасываются, как и должны, а градиенты бинаризованных – зануляются, поскольку не нужно дообучать уже бинаризованные веса.

Для бинаризации и дообучения сети использованы параметры из Таблицы 7.

Таблица 7: Параметры поэлементной бинаризации

Параметр	Значение
Оптимизатор	AdamW
Скорость обучения	$5e^{-3}$
Функция потерь	Кросс-энтропия
Процент бинаризации весов за этап	5

В каждом слое сначала бинаризовался определенный процент весов, случайно выбранных среди небинаризованных. Затем происходило дообучение небинаризованных весов, после чего процесс повторялся до полной бинаризации всех весов слоя.

При поэлементной бинаризации лучший результат составил 87.5%, что превысило результат референтной модели.

3.5 Результаты экспериментов

Общий итог всех экспериментов проиллюстрирован на столбчатой диаграмме Рис. 9. Классическая вещественная сеть достигла результата в 98% точности, чуть хуже показала себя предложенная наивная бернуллиевская модель – 97%. Референтная модель обучилась до качества в 59%. При бинаризации заметен существенный спад качества: 10% при полной бинаризации и 41% при послойной. И, наконец, поэлементная бинаризация превысила референтную модель и достигла точности в 86%.

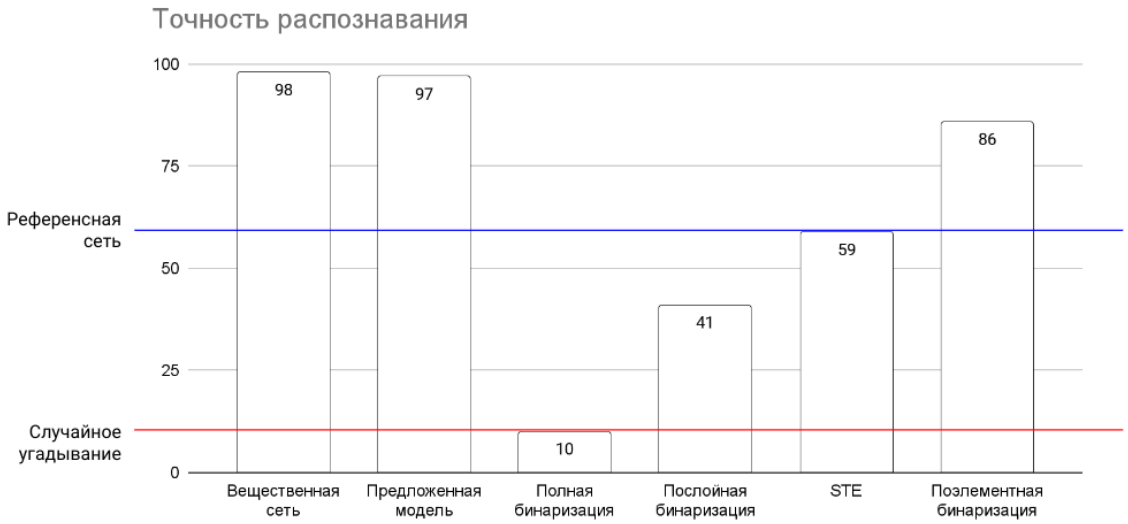

Рис. 9: Точность распознавания на MNIST различными способами.

Заключение

В данной работе исследована проблема обучения бинарных нейронных сетей. Цель состояла в разработке нового метода обучения бинарных нейронных сетей на основе непрерывной аппроксимации бинарной сети с помощью вероятностной модели, который таким образом будет иметь лучшую теоретическую обоснованность.

В ходе исследования были предложены полносвязные и сверточные слои, построена нейросетевая модель и разработаны методы бинаризации. Результаты показывают, что использование слоев наивной бернуллиевской сети может значительно улучшить качество обучения бинарной модели: при использовании классического подхода STE было получена точность распознавания в 59%, а при использовании предложенной модели лучший результат составил 87.5%.

Также было проведено экспериментальное исследование различных способов бинаризации. Обнаружено, что методы поэлементной бинаризации привести к более точным и стабильным результатам, в то время как методы полной и послойной бинаризации могут вызывать потерю информации и снижение точности.

В целом, эта работа вносит вклад в область бинарных нейронных сетей, предлагая новые подходы к обучению с помощью непрерывной аппроксимации и способы перехода к полностью бинарной модели. Дальнейшие исследования могут сосредоточиться на проверке методов на других нейросетевых архитектурах и расширении области их применения для решения разнообразных задач машинного обучения.

Список литературы

[1] Xu Y. et al. Learning frequency domain approximation for binary neural networks //Advances in Neural Information Processing Systems. – 2021. – Т. 34. – С. 25553-25565.

[2] Qin H. et al. Binary neural networks: A survey //Pattern Recognition. – 2020. – Т. 105. – С. 107281.

[3] Courbariaux M., I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. «Binarized neural networks: Training deep neural networks with weights and activations constrained to + 1 or-1.» arXiv preprint arXiv:1602.02830 (2016).

[4] Darabi, Sajad, M. Belbahri, M. Courbariaux, and V. Partovi Nia. «Bnn+: Improved binary network training.» (2018).

[5] Krizhevsky A. et al. Learning multiple layers of features from tiny images. – 2009.

[6] Deng J. et al. Imagenet: A large-scale hierarchical image database //2009 IEEE conference on computer vision and pattern recognition. – Ieee, 2009. – С. 248-255.

[7] Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition //arXiv preprint arXiv:1409.1556. – 2014.

[8] Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks //Communications of the ACM. – 2017. – Т. 60. – №. 6. – С. 84-90.

[9] He, Xiangyu, Z. Mo, K. Cheng, W. Xu, Q. Hu, P. Wang, Q. Liu, and J. Cheng. «Proxybnn: Learning binarized neural networks via proxy matrices.» In European Conference on Computer Vision, pp. 223-241. Springer, Cham, 2020.

[10] He K. et al. Deep residual learning for image recognition //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2016. – С. 770-778.

[11] Zagoruyko S., Komodakis N. Wide residual networks //arXiv preprint arXiv:1605.07146. – 2016.

[12] T. Wei, G. Hua, and L. Wang. «How to train a compact binary neural network with high accuracy?.» In Thirty-First AAAI conference on artificial intelligence. 2017.

[13] Jang E., Shixiang Gu, and Ben Poole. «Categorical reparameterization with gumbel-softmax.» arXiv preprint arXiv:1611.01144 (2016).

[14] Rastegari M., V. Ordonez, J. Redmon, and A. Farhadi. «Xnor-net: Imagenet classification using binary convolutional neural networks.» In European conference on computer vision, pp. 525-542. Springer, Cham, 2016.

[15] Su J., M. Cvitkovic, and F. Huang. «Sampling-free learning of Bayesian quantized neural networks.» arXiv preprint arXiv:1912.02992 (2019).

[16] R. Wolfgang, G. Schindler, H. Fröning, and F. Pernkopf. «Training discrete-valued neural networks with sign activations using weight distributions.» In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 382-398. Springer, Cham, 2019.

[17] Z. Shuchang, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. «Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients.» arXiv preprint arXiv:1606.06160 (2016).

[18] Peters J. WT and M. Welling. «Probabilistic binary neural networks.» arXiv preprint arXiv:1809.03368 (2018).

[19] Meng X., R. Bachmann, and M. Emtiyaz Khan. «Training binary neural networks using the bayesian learning rule.» In International conference on machine learning, pp. 6852-6861. PMLR, 2020.

[20] Soudry D., I. Hubara, and R. Meir. «Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights.» Advances in neural information processing systems 27 (2014).

[21] Д. Павлюченков, Е. Лимонова, А. Трусов «Обучение бинарных нейросетевых моделей с помощью непрерывных аппроксимаций» Сборник тезисов к 65-й Всероссийской научной Конференции МФТИ (в печати, 2023)

[22] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In Advances in neural information processing systems, pp. 3123–3131, 2015

[23] S. Elfwing, E. Uchibe, and K. Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. Neural Networks, 2018.

[24] Yu X., Efe M. O., Kaynak O. A general backpropagation algorithm for feedforward neural networks learning //IEEE transactions on neural networks. – 2002. – Т. 13. – №. 1. – С. 251-254.

[25] Janocha K., Czarnecki W. M. On loss functions for deep neural networks in classification //arXiv preprint arXiv:1702.05659. – 2017.

[26] Цей Р., Шумафов М. М. Число обусловленности матрицы как показатель устойчивости при решении прикладных задач //Труды ФОРА. – 2011. – №. 16.

[27] Lee W. T. Tridiagonal matrices: Thomas algorithm //MS6021, Scientific Computation, University of Limerick. – 2011.

[28] O'Shea K., Nash R. An introduction to convolutional neural networks //arXiv preprint arXiv:1511.08458. – 2015.

[29] K. Chellapilla, S. Puri and P. Simard, «High performance convolutional neural networks for document processing», Proc. 10th Int. Workshop Frontiers Handwriting Recognit., 2006.

[30] Nusrat I., Jang S. B. A comparison of regularization techniques in deep neural networks //Symmetry. – 2018. – Т. 10. – №. 11. – С. 648.

[31] LeCun Y. et al. Gradient-based learning applied to document recognition //Proceedings of the IEEE. – 1998. – T. 86. – №. 11. – C. 2278-2324.

[32] LeCun Y. et al. Gradient-based learning applied to document recognition //Proceedings of the IEEE. – 1998. – T. 86. – №. 11. – C. 2278-2324.

[33] Lydia A., Francis S. Adagrad – an optimizer for stochastic gradient descent //Int. J. Inf. Comput. Sci. – 2019. – T. 6. – №. 5. – C. 566-568.

[34] Yin P. et al. Understanding straight-through estimator in training activation quantized neural nets //arXiv preprint arXiv:1903.05662. – 2019.