ПРОГРАММИРОВАНИЕ, 2024, № 6, с. 3–11

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

УДК 004.056.55

ВЫСОКОСКОРОСТНОЙ АЛГОРИТМ СКАЛЯРНОГО УМНОЖЕНИЯ ДЛЯ ПРОЕКТИРОВАНИЯ НЕЙРОННЫХ СЕТЕЙ, СОХРАНЯЮЩИХ КОНФИДЕНЦИАЛЬНОСТЬ

© 2024 г. М. А. Лапина[a], *, Е. М. Ширяев[a], **, М. Г. Бабенко[a], *, И. Истамов[b], ******

[a] *Северо-Кавказский центр математических исследований, Северо-Кавказский федеральный университет*
355017 Ставрополь, ул. Пушкина, 1, Россия
[b] *Самаркандский государственный университет имени Шарофа Рашидова*
140104 Самарканд, Университетский бульвар, 15, Узбекистан
**E-mail: mlapina@ncfu.ru*
***E-mail: eshiriaev@ncfu.ru*
****E-mail: mgbabenko@ncfu.ru*
*****E-mail: istamovismoilzoda@gmail.com*

Поступила в редакцию 10.07.2024 г.
После доработки 16.07.2024 г.
Принята к публикации 26.07.20204 г.

В силу юридических ограничений либо ограничений, связанных с внутренней информационной политикой компаний, зачастую бизнес не доверяет конфиденциальную информацию публичным облачным провайдерам. Одним из механизмов, позволяющих обеспечить безопасность конфиденциальных данных в облаках, является гомоморфное шифрование. Для проектирования решений, использующих нейронные сети, в данных условиях используются нейронные сети, сохраняющие конфиденциальность. Они эксплуатируют механизм гомоморфного шифрования, позволяя таким образом обеспечить безопасность коммерческой информации в облаке. Основным сдерживающим фактором использования нейронных сетей, сохраняющих конфиденциальность, является большая вычислительная и пространственная сложность алгоритма скалярного умножения, который является базовым для вычисления математической свертки. В работе предлагается алгоритм скалярного умножения, который позволяет уменьшить пространственную сложность с квадратичной до линейной, а также уменьшить время вычисления скалярного умножения в 1.38 раза.

Ключевые слова: матричные операции; искусственные нейронные сети; полностью гомоморфное шифрование; CKKS; TenSEAL; нейронные сети, сохраняющие конфиденциальность

DOI: 10.31857/S0132347424060012, **EDN:** DZISPQ

1. ВВЕДЕНИЕ

Методы искусственного интеллекта (ИИ) [1] набирают все большую популярность в последние годы. Если в 2000-х гг. ИИ зачастую интересовались только исследовательские круги, то в последние десятилетия ИИ набирает популярность во всех областях человеческой деятельности. Если проанализировать научно-технические достижения, то можно отметить следующее. На рост популярности методов ИИ в большей степени повлияло развитие децентрализованных вычислительных архитектур, в том числе облачных вычислений, аппаратных ускорителей, а также общая тенденция увеличения вычислительной мощности устройств. Методы ИИ сегодня применяются в производстве для повышения эффективности автоматизации, в медицине и финансовых структурах для анализа больших данных и т. д. С ростом популярности языковых моделей и запуском GPT с открытым исходным кодом методы ИИ охватили еще большее количество сфер человеческой деятельности [2].

Однако, как и в конце XX в., когда появился Интернет/Всемирная паутина, методы ИИ стали предметом различных дискуссий [3], как с точки зрения закона и законотворчества, так и с точки зрения безопасности данных. Задачи, решаемые ИИ, как правило, сопряжены с обработкой больших объемов данных, а в случае с теми же языковыми моделями – очень больших объемов данных. Большие данные [4] могут содержать информацию с ограниченным доступом, например, если ИИ используется компаниями для

обработки данных пользователей, в медицинской организации – персональные данные пациентов, муниципальной/государственной – данные граждан, внутренние документы, в финансовых организациях – данные клиентов, информация о счетах, биржевые котировки и т. д. Все вышеперечисленные данные зачастую являются конфиденциальными и охраняются законом, например, в Российской Федерации это Федеральный закон от 27 июля 2006 г. № 152-ФЗ "О персональных данных", который направлен на усиление контроля за обработкой и распространением личной информации граждан [5]. Если методы ИИ используются внутри закрытой сети, вопросы безопасности можно решить стандартными методами, но создание и поддержка высокопроизводительной закрытой сети требует больших вычислительных ресурсов и финансирования. Поэтому зачастую эффективнее обратиться к поставщику услуг облачных вычислений, что и делает большинство компаний. При аренде вычислительных мощностей сеть становится общедоступной, что создает риск компрометации конфиденциальных данных. Хотя поставщики облачных услуг гарантируют безопасность данных, хранящихся в облаке, гарантировать безопасность вычислений на данный момент практически невозможно, поскольку можно хранить данные в зашифрованном виде, но не обрабатывать.

Таким образом, возникает проблема обработки конфиденциальных данных ИИ в публичных сетях. В качестве решения проблемы можно рассмотреть такой криптографический примитив, как полностью гомоморфное шифрование (ПГШ) [6], он позволяет выполнять гомоморфные операции сложения и умножения над зашифрованными данными. Этого достаточно, например, для работы нейронных сетей (НС) [7], когда нам нужно обрабатывать входные данные с помощью уже обученных нейронов. В этой области также существуют нерешенные проблемы, например, с аппроксимацией некоторых функций активации. Кроме того, хотя для матриц операции можно реализовать на основе сложения и умножения (вычитание реализуется как сложение положительных и отрицательных чисел), из-за особенностей схем ПГШ возникает большая избыточность данных, что приводит к неэффективной работе конфиденциальных НС. Цель данной работы – исследовать матричное умножение в контексте гомоморфного шифрования для повышения эффективности использования памяти, разработать новый алгоритм и протестировать его эффективность для конфиденциальных НС.

Работа организована следующим образом: в разделе 2 рассматриваются конфиденциальные НС и методы их организации, в разделе 3 представлено исследование конфиденциального умножения матриц, в разделе 4 анализируются полученные результаты на основе экспериментального исследования, в разделе 5 подводятся итоги проделанной работы и описываются будущие работы.

2. НЕЙРОННЫЕ СЕТИ, СОХРАНЯЮЩИЕ КОНФИДЕНЦИАЛЬНОСТЬ

2.1. Искусственная нейронная сеть

Математическая модель искусственных нейронных сетей представляет собой линейную композицию взаимосвязанных нейронов с нелинейными функциями активации (рис. 1).

Входной слой обрабатывает входные данные, скрытый слой выполняет вычисления, а выходной слой отвечает за выходную информацию. Искусственная НС, как и биологическая НС, работает посредством активации нейронов, т. е., когда значение внутри нейрона достигает функции активации, следующему нейрону передаются ее значение. Каждый нейрон имеет свое базовое значение (вес). Тогда состояние нейрона S можно описать как

$$S = \sum_{i=0}^{n} x_i w_i,$$

где x_i – значение i-го входа нейрона; w_i – вес i-го входа; n – количество входов нейрона. Передача значений следующему нейрону осуществляется посредством функции активации:

$$Y = f(S),$$

где $f(S)$ – функция активации. Например, сигмоидальная функция активации [8], определяется как

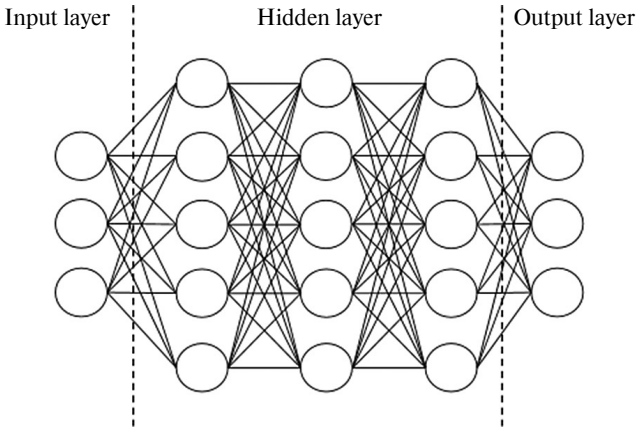


Рис. 1. Модель искусственной нейронной сети.

$$f(S) = \frac{1}{1 + e^{-aS}}. \quad (1)$$

С математической точки зрения, НС – это многопараметрическая задача нелинейной оптимизации, где веса нейронов скрытого слоя представляют собой параметры, а нейроны выходного слоя – ограничения. Чтобы нейронная сеть работала правильно, ее нужно обучить. Обучение происходит путем изменения внутренних значений весов нейронов. Существует несколько способов обучения, как с учителем, так и без него. Самый популярный способ обучения с учителем – минимизация ошибок. На основе этого типа обучения строятся сети с обратным распространением, которые используются для поиска закономерностей, прогнозирования и качественного анализа. Анализируя рис. 1, можно заметить, что если ввести операцию умножения матриц, то можно повысить эффективность обработки данных, как это и делается в большинстве случаев. Чтобы строить НС, сохраняющие конфиденциальность, необходимо ввести понятие криптографического примитива ПГШ.

2.2. Полностью гомоморфное шифрование и схема CKKS

Полностью гомоморфное шифрование (ПГШ) – криптографический примитив, развивающий идеи гомоморфного шифрования (ГШ). ГШ позволяет выполнять гомоморфное сложение или гомоморфное умножение над зашифрованным текстом. Примерами ГШ являются асимметричные шифры, такие как RSA [9], ElGamal [10] и др. Криптографы еще в 1980-х гг. предполагали, что полностью гомоморфное шифрование, когда шифр позволяет выполнять и гомоморфное сложение, и гомоморфное умножение, возможно. Первая схема ПГШ была представлена Джентри в его работе 2009 г. [6]. Однако эта схема не была эффективной и обрабатывала двоичные биты с помощью логических операций довольно долго (по сравнению с современными схемами), кроме того, схема имела большие ограничения на количество допустимых операций (количество операций, после которых сообщение может быть восстановлено). За следующие 15 лет сам Джентри [11–13] и его последователи [14–17] разработали новые схемы ПГШ, которые работают с целыми числами, выполняются быстрее и ослабляют ограничения на вид и количество операций.

Следующим шагом в истории ПГШ стала схема CKKS (первоначально HEaaN), которая позволяет обрабатывать рациональные числа [18]. CKKS – это система гомоморфного шифрования, предназначенная для эффективного выполнения приближенных арифметических операций над зашифрованными данными. Она идеально подходит для вычислений с вещественными или комплексными числами над полем $C^{N/2}$. Пространство открытых текстов и пространство шифротекстов имеют одну и ту же область

$$Z_Q[X]/(X^N + 1),$$

где N – чаще всего степень двойки.

Пакетное кодирование $C^{\frac{N}{2}} \leftrightarrow Z_Q[X]/(X^N + 1)$ отображает массив комплексных чисел в многочлен со свойством: $decode(encode(m_1) \otimes encode(m_2)) \approx m_1 \odot m_2$, где $\otimes$ – покомпонентное умножение, а $\odot$ – негациклическая свертка.

Схема CKKS работает по стандарту [19], который содержит рекомендуемые параметры для 128-битных ключей ГШ троичной формы $s \in_u \{-1, 0, 1\}^N$. Шифрование в CKKS осуществляется путем вычисления полиномов Лагранжа в поле комплексных чисел.

Схема использует приближенную арифметику для построения шифротекстов. Рассмотрим заданную арифметику. В начале мы фиксируем основание $p > 0$ и модуль q_0, причем $q = pq_0$ при $0 < l \le L$. Целое число p будет использоваться в качестве основы для масштабирования в приблизительных расчетах. В качестве параметра безопасности λ выбирается такой параметр, что $M = M(\lambda, q_L)$ для полиномиального кольца. При границах $< l \le L$ уровня шифротекста l определяется вектор в $\mathcal{R}_{q\ell}^{k}$ для фиксированного целого числа k.

1. Генерация ключей: процесс шифрования начинается с генерации ключей: открытого ключа pk и закрытого ключа sk. Закрытый ключ используется для дешифровки данных, а открытый – для их шифрования.
2. Шифрование: чтобы зашифровать вектор открытого текста x, выполняются следующие действия:
   - Дополнение: вектор $m(x)$ дополнен нулями, длина вектора равна заданной степени двойки N;
   - Кодирование: вектор открытого текста x кодируется в полином открытого текста $m(x)$, который является полиномиальным представлением сообщения;
   - Гомоморфное шифрование: полином $m(x)$ шифруется с помощью pk для получения

полинома $c(x)$ шифротекста, при этом контролируется уровень шума шифротекста – количество специально вносимых ошибок e, удовлетворяющее $| e |_{\infty}^{can} \le e_{Max}$, для выражения $c, sk = m + e_{Max} \left(mod\, q_L \right)$.

3. Десшифрование: для дешифровки полинома $c(x)$ шифротекста выполняются следующие действия:
   - Гомоморфное дешифрование: полином $c(x)$ дешифруется с помощью секретного ключа для получения полинома $m(x) \leftarrow c, sk \left(mod\, q_l \right)$ в пространстве открытых текстов;
   - Декодирование: для получения исходного текстового вектора x текстовый полином $m(x)$ снова преобразуется из полинома в полином сообщений.
4. Гомоморфные операции: CKKS поддерживает несколько приближенных арифметических операций над зашифрованными данными, включая сложение и умножение. Гомоморфное сложение и умножение можно выполнять в пространстве шифротекстов без необходимости их дешифровки:
   - Гомоморфное сложение: при получении двух шифротекстов $c_1(x)$ и $c_2(x)$, представляющих зашифрованные значения $m_1(x)$ и $m_2(x)$ соответственно, выполняется гомоморфное сложение путем сложения соответствующих коэффициентов по модулю: $c(x) = c_1(x) + c_2(x)$, при этом ошибки e_1 и e_2 также суммируются;
   - Гомоморфное умножение: при наличии двух шифротекстов $c_1(x)$ и $c_2(x)$, представляющих зашифрованные значения $m_1(x)$ и $m_2(x)$ соответственно, гомоморфное умножение выполняется путем преобразования зашифрованных полиномиальных текстов для последующего покомпонентного умножения по модулю исходного модульного текста и обратного преобразования: $c(x) = c_1(x) + c_2(x)$, для умножения выделяются собственные границы ошибок $e_{mult} \in \mathcal{R}$ с $\left| e_{mult} \right|_{\infty}^{can} e_{multMax}(l)$, где $e_{multMax}(l)$ заданная константа.

Как сложение, так и умножение приводят к увеличению ошибки аппроксимации e, схема CKKS позволяет дешифровать данные, если ошибка находится в определенных пределах. При использовании схемы CKKS важно контролировать рост ошибки, который зависит от количества операций и их порядка. Учитывая особенности арифметики, умножение вносит большую погрешность. Различные программные реализации схемы CKKS предлагают разные способы контроля уровня ошибки.

2.3. Нейронные сети, сохраняющие конфиденциальность

Интерес к нейронным сетям, сохраняющим конфиденциальность (НССК), возник несколько лет назад. В своем обзоре [20] авторы исследуют эту концепцию с теоретической точки зрения, рассматривая основные задачи и проблемы, с которыми сталкиваются исследователи при построении НССК на основе ПГШ. В основе НССК лежит концепция Machine Learning as a Service (MLaaS) [21], которая схожа с концепциями облачных вычислений [22–24]. В статье даны определения операций ПГШ, включая проблемные операции. Помимо умножения матриц, также упоминается бутстраппинг [25], который используется для увеличения количества допустимых операций умножения. Также описаны инструменты для работы с ПГШ [26–28], включая используемые в НССК [29]. Проблема ускорения НССК, работающих в ПГШ, упоминается в статье отдельно. Эта тема также популярна среди исследователей, например, в [30] изучается ускорение операции умножения матриц путем модификации метода Хавели [31]. Однако операции умножения, основанные на этом методе, по-прежнему достаточно ресурсоемки. В следующем разделе подробно рассмотрена операция умножения матриц и способы повышения скорости обработки данных.

3. ОПЕРАЦИЯ УМНОЖЕНИЯ МАТРИЦ В ПРИБЛИЖЕННОЙ СХЕМЕ ГОМОМОРФНОГО ШИФРОВАНИЯ

Умножение матриц – базовая операция для многих систем, в том числе и для НС. Рассмотрим ее алгоритм, основанный на умножении квадратных матриц a и b размера $n \times n$:

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j},$$

где $i, j, k \in \overline{1,n}$, c – результат умножения. В открытом виде этот алгоритм довольно прост. Однако в ПГШ его выполнение невозможно, так как мы не можем отдельно обратиться к элементу каждого внутреннего вектора. Рассмотрим подробно метод Хавели [31]. Для выполнения матричного умножения в ПГШ матрицы должны быть закодированы в диагональном представлении. Затем для выполнения умножения необходимо

выполнить несколько операций вращения со вспомогательными матрицами. Рассмотрим пример.

Пусть матрица A размера $m \times m$ представлена в зашифрованном виде $y_0, \ldots, y_{m-1}$, где $y_i = (A_{0,i}, A_{1,i+1}, \ldots, A_{m-1,m+1})$. Сначала, $y_i[j] = A_{j,j+1}$. Далее, кусочное произведение между вектором весов и матрицей $w = vA$, где $v = \{v_0, v_1, \ldots, v_{n-1}\}$ – входной вектор, может быть вычислено как:

$$v_0 = \{x_0, x_1, \ldots, x_{n-1}\} \odot \{y_0, y_1, \ldots, y_{n-1}\} = \{x_0 y_0, x_1 y_1, \ldots, x_{n-1} y_{n-1}\},$$

$$v_1 = \{x_{n-1}, x_0, \ldots, x_{n-2}\} \odot \{y_{n-1}, y_0, \ldots, y_{n-2}\} = \{x_{n-1} y_{n-1}, x_0 y_0, \ldots, x_{n-2} y_{n-2}\},$$

$$\ldots,$$

$$v_{n-1} = \{x_1, x_2, \ldots, x_{n-1}, x_0\} \odot \{y_1, y_2, \ldots, y_{n-1}, y_0\} = \{x_1 y_1, x_2 y_2, \ldots, x_{n-1} y_{n-1}, x_0 y_0\},$$

где $\odot$ – покомпонентное произведение векторов.

Однако есть и другой способ, который подходит для НССК. Возникает вопрос, настолько ли необходим подобный уровень конфиденциальности.

Учитывая тот факт, что в настоящее время невозможно обучить НС в ПГШ за приемлемое время, НС обучается в открытом виде. Справедливо заметить, что значения весов являются открытыми и могут быть общедоступными и существует возможность их компрометации. Тогда нет смысла их шифровать, и мы можем применять их в открытом виде. Тогда можно использовать модифицированный алгоритм, основанный на предыдущем, где входная матрица кодируется в диагональном представлении, а веса представляются в виде вектора. Такое умножение рассматривается как умножение матрицы на скаляр, которое реализуется посредством операций умножения и вращения. Рассмотрим пример.

Пусть матрица A размера $n \times n$ представлена в зашифрованном виде $a_0, \ldots, a_{m-1}$, где $a_i = (A_{0,i}, A_{1,i+1}, \ldots, A_{n-1,n+1})$. Сначала, $a_i[j] = A_{j,j+1}$. Следующее произведение $w = vA$, где $v = \{v_0, v_1, \ldots, v_{n-1}\}$ – входной вектор, может быть вычислено как

$$v_0 = x_0 \{a_0, a_1, \ldots, a_{n-1}\},$$

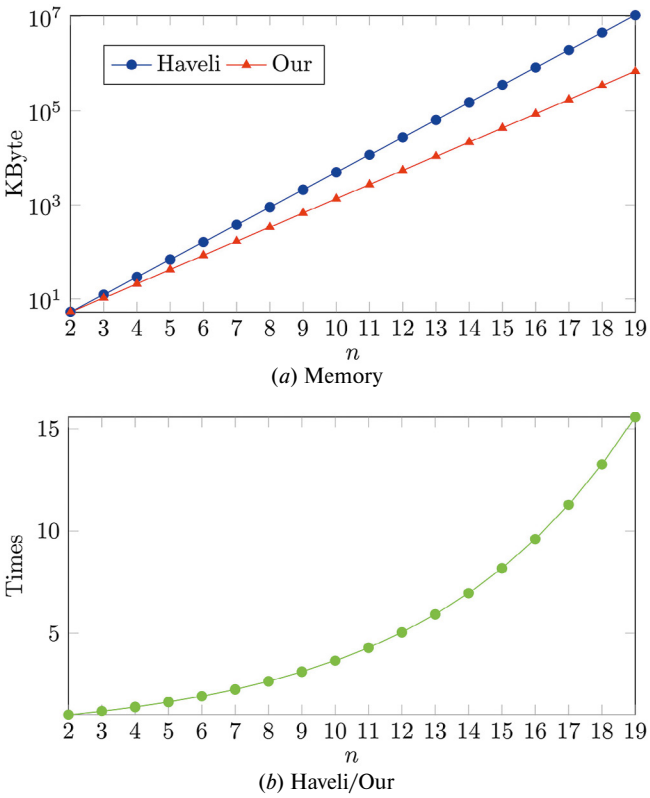
$$v_1 = x_1 \{a_{n-1}, a_0, \ldots, a_{n-2}\},$$

$$\ldots,$$

$$v_{n-1} = x_{n-1} \{a_1, a_2, \ldots, a_{n-1}, a_0\}.$$

Этот метод требует m операций вращения, умножения и сложения. Кроме того, вспомогательные матрицы w и v в зашифрованном виде занимают много памяти, как и промежуточные зашифрованные матрицы до получения результата. Из формул видно, что, используя открытые значения, мы не только сокращаем количество операций, но и расход памяти. Для подтверждения выводов было проведено экспериментальное исследование (рис. 2).

Из данных, представленных на рис. 2*a*, можно сделать вывод, что предложенный алгоритм позволяет сократить объем памяти в среднем в 7.89 раза. Линия тренда для данных, представленных на рис. 2*b*, равна $0.0656n^2 - 0.4452n + 2.0912$ с коэффициентом детерминации $R^2 = 0.9925$. Таким образом, можно сделать вывод, что пространственная сложность уменьшилась с $O(n^4)$ до $O(n^2)$ для произведения матриц размера $n \times n$. Учитывая, что для произведения квадратных матриц размера $n \times n$ необходимо вычислить n^2 скалярных умножений и $2n^2$ циклических сдвигов векторов, пространственная сложность алгоритма скалярного умножения с зашифрованными данными снижается с $O(n^2)$ до $O(n)$.



(*a*) Memory

(*b*) Haveli/Our

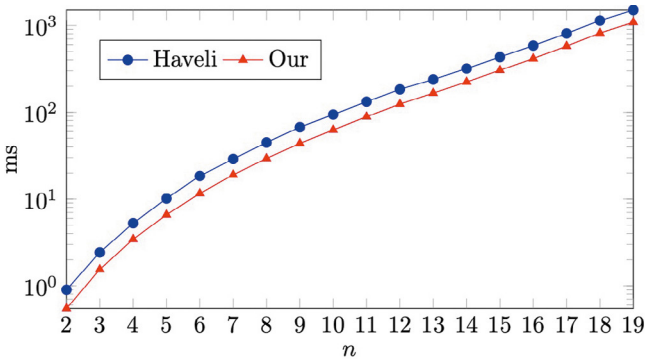
Рис. 2. Исследование потребления памяти предлагаемым методом.

Как видно на рис. 2*b*, потребление памяти сократилось с квадратичного закона до линейного. Это дает преимущество в эффективности при работе с HCCK. Однако, учитывая специфику схемы CKKS, необходимо проверить, не повлияли ли внесенные изменения на точность результата. Для этого необходимо построить нейронную сеть и провести исследование. Далее рассмотрим скорость, с которой выполняются вычисления. Стоит отметить, что на рис. 3*a* показана общая скорость выполнения операций, включая шифрование и дешифрование.

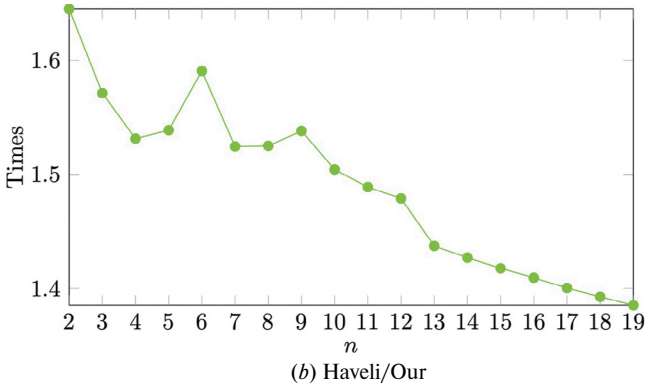
Анализируя рис. 3*a*, можно сказать, что предложенный метод выполняет умножение быстрее. Этот эффект достигается как за счет нового подхода к умножению, так и за счет того, что шифрование и дешифрование упрощаются, поскольку умножение выполняется на открытом векторе весов HCCK. Кроме того, мы предлагаем проанализировать соотношение скоростей методов (рис. 3*b*).

Линии тренда зависимости времени от n для алгоритма Хавели $0.0634n^4 - 1.8561n^3 + 20.897n^2 - 88.794n + 118.56$, для предложенного алгоритма $0.0461n^4 - 1.3405n^3 + 14.955n^2 - 63.491n + 84.81$ с коэффициентом детерминации для обеих линий равным $R^2 = 0.9995$ (рис. 3*a*). Асимптотически выигрыш во времени при увеличении n равен

$$\lim_{n\to\infty} \frac{0.0634n^4 - 1.8561n^3 + 20.897n^2 - 88.794n + 118.56}{0.0461n^4 - 1.3405n^3 + 14.955n^2 - 63.491n + 84.81} \approx$$
$$\approx 1.38.$$



(*a*) Time



(*b*) Haveli/Our

Рис. 3. Исследование времени вычисления операции умножения матриц.

Время работы алгоритма для произведения квадратных матриц $n \times n$ уменьшилось в среднем в 1.49 раза (рис. 3*b*). С увеличением размера график становится более линейным, что можно объяснить увеличением избыточности, которая зависит от длины вектора, в то время как при малых размерах она зависит как от конструкции вектора, так и от вспомогательных матриц, необходимых для вращения.

В целом можно сказать, что предложенный метод эффективен как с точки зрения потребления памяти, так и скорости вычислений. Стоит отметить, что этот результат достигается за счет снижения конфиденциальности, а именно конфиденциальности весов HCCK, при условии, что веса являются общеизвестными при обучении HCCK в открытом виде.

4. ИССЛЕДОВАНИЕ ТОЧНОСТИ

В рамках исследования были проведены эксперименты по обучению и тестированию нейронной сети, а также ее зашифрованной версии на наборе данных MNIST. Целью эксперимента было оценить производительность модели в обычном и зашифрованном режимах, а также изучить влияние гомоморфного шифрования на производительность и точность модели. Аппаратная конфигурация состоит из процессора Intel(R) Xeon(R) CPU E5–2696 v3 с тактовой частотой 2.30 ГГц, 32 ГБ оперативной памяти DDR4 с частотой 2133 МГц и твердотельного накопителя объемом 1 ТБ. Среднее время измерялось путем запуска алгоритмов на платформе 10000 раз. В ходе эксперимента собирались данные о потерях при обучении и тестировании, а также о точности классификации для каждого класса. Для этого была построена НС на основе следующей математической модели.

Рассмотрим сверточную нейронную сеть (СНС) со следующими слоями и параметрами:

- Входное изображение: I, одноканальное изображение.
- Первый сверточный слой (C_1): применяет 4 фильтра с размером ядра 7×7 с шагом 3 и размером 0.
- Первый слой – полносвязный (F_1): преобразует упрощенные карты признаков с использованием H-нейронов.

- Второй полносвязный слой (F_2): сопоставляет скрытый слой с выходным слоем с O-нейронами.

Математические операции, выполняемые СНС, выглядят следующим образом:

1. Работа первого сверточного слоя может быть определена как
$$C_1(I) = Conv2d(I, K_1, S_1, P_1),$$
где $K_i = 7 \times 7$ – размер ядра, с шагом $S_1 = 3$ и размером $P_1 = 0$.
2. Выходной сигнал C_1 проходит через функцию активации и, возможно, другие операции, такие как объединение или нормализация, после чего сглаживается и поступает в первый слой с полным подключением.
3. Работа первого полносвязанного слоя может быть определена как
$$F_1(X) = XW_{F_1} + b_{F_1},$$
где X – входной вектор для F_1; W_{F_1} и b_{F_1} представляют собой веса и смещения F_1 соответственно.
4. Работа второго полносвязанного слоя аналогично определяется как
$$F_2(Y) = YW_{F_2} + b_{F_2},$$
где Y – входной вектор F_2, полученный из выхода F_1; W_{F_2} и b_{F_2} – веса и смещения F_2 соответственно. Эта модель описывает структуру СНС, подчеркивая последовательность от обработки на сверточном слое до генерации конечного вывода через полносвязанные слои.

СНС была выбрана в качестве модели потому, что ПГШ обладает свойствами как гомоморфизма, так и автоморфизма, благодаря которым вращение зашифрованных матриц для реализации матричного умножения реализовать достаточного просто. Кроме того, эти свойства позволяют достаточно эффективно выполнять математическую операцию свертки [32]. Далее рассмотрим

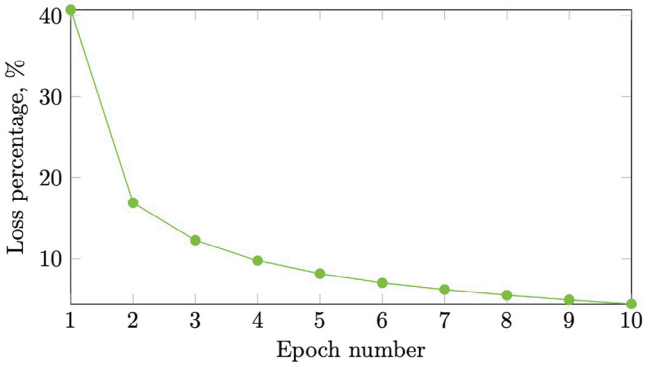


Рис. 4. Исследование функции потерь для PPNN.

результаты работы полученной модели. А именно, данные о потерях (рис. 4).

На рис. 4 показана динамика потерь нейронной сети в процессе обучения для 10 эпох. По оси абсцисс откладывается номер теста, равный количеству эпох (от 1 до 10), а по оси ординат – количество потерь. На графике видно уменьшение потерь с каждой последующей эпохой, что свидетельствует об адекватном поведении модели при использовании нового алгоритма обучения.

На рис. 5 показана точность классификации модели на тестовых данных для каждого из 10 классов. По оси абсцисс представлены классы (от 0 до 9), а по оси ординат – процент точности для каждого класса. График помогает наглядно представить, как модель справляется с классификацией различных категорий, выявляя классы, в которых модель работает лучше или хуже.

Результаты экспериментов показывают, что нейронная сеть демонстрирует высокую точность как в обычном, так и в зашифрованном режимах, причем в зашифрованном режиме общая точность несколько повышается. Это говорит о том, что применение гомоморфного шифрования не оказывает существенного негативного влияния на способность модели к классификации. Кроме того, тот факт, что в некоторых классах зашифрованная СНС показывает более

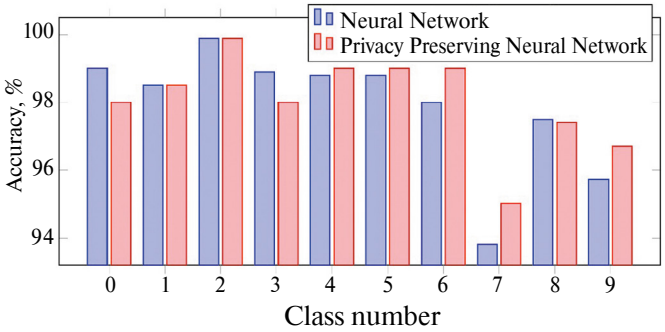


Рис. 5. Исследование точности PPNN для различных классов.

точные результаты, требует дополнительных исследований.

5. ЗАКЛЮЧЕНИЕ

Результатом исследования по адаптации НССК для работы с ПГШ стали результаты, которые подчеркивают потенциал и ограничения данного подхода. Исследование показывает, что, изменив метод умножения зашифрованных матриц, можно успешно использовать НССК при сохранении эффективности обработки данных и потребления памяти.

Анализ результатов тестирования НССК показал, что модель демонстрирует улучшение производительности с каждой эпохой, что видно по снижению потерь при тестировании. Это говорит о том, что НССК адекватно адаптируется к зашифрованным данным и эффективно их обрабатывает. Результаты тестирования модели показали высокую точность классификации как для каждого из классов, так и в целом, что подтверждает эффективность модели в задачах классификации. Интересно отметить, что зашифрованная версия модели показала сопоставимую, а в некоторых случаях даже более высокую точность, что говорит о том, что применение ПГШ не оказывает существенного негативного влияния на способность модели к классификации. Тем не менее следует отметить, что использование приближенной ПГШ требует дальнейших исследований для оптимизации баланса между безопасностью, конфиденциальностью и производительностью модели. Важно изучить влияние различных типов аппроксимации функции активации на точность и общую производительность модели, а также разработать методы улучшения производительности НССК.

В статье предложен алгоритм скалярного умножения, позволяющий уменьшить пространственную сложность с $O(n^2)$ до $O(n)$ и сократить время вычисления скалярного умножения в 1.38 раза.

Результаты данного исследования открывают новые перспективы для разработки безопасных НС, особенно в тех областях, где требуется обработка конфиденциальных данных, а также подчеркивают важность продолжения исследований в этой области для достижения оптимального сочетания безопасности, конфиденциальности и эффективности в НССК. В будущем планируется исследовать и разработать методы выполнения других операции в НССК для повышения эффективности вычислений, потребления памяти, точности и конфиденциальности.

6. ИСТОЧНИК ФИНАНСИРОВАНИЯ

Исследование выполнено за счет гранта Российского научного фонда № 19-71-10033, https://rscf.ru/project/19-71-10033/.

СПИСОК ЛИТЕРАТУРЫ

1. *Hunt E.B.* Artificial intelligence. Academic Press, 2014.
2. *Radford A. et al.* Improving language understanding by generative pre-training. OpenAI, 2018.
3. *Wamser F. et al.* Traffic characterization of a residential wireless Internet access // Telecommunication Systems. Springer, 2011. V. 48. P. 5–17.
4. *Sagiroglu S., Sinanc D.* Big data: A review // 2013 international conference on collaboration technologies and systems (CTS). IEEE, 2013. P. 42–47.
5. *О персональных данных* [Electronic resource]. http://pravo.gov.ru/proxy/ips/?docbody&nd=102108261 (accessed: 16.06.2024)
6. *Gentry C.* A fully homomorphic encryption scheme. Stanford university, 2009.
7. *Yegnanarayana B.* Artificial neural networks. PHI Learning Pvt. Ltd., 2009.
8. *Pratiwi H. et al.* Sigmoid activation function in selecting the best model of artificial neural networks // Journal of Physics: Conference Series. IOP Publishing, 2020. V. 1471. № 1. P. 012010.
9. *Rivest R.L., Shamir A., Adleman L.* A method for obtaining digital signatures and public-key cryptosystems // Commun. ACM. 1978. V. 21. № 2. P. 120–126.
10. *ElGamal T.* A public key cryptosystem and a signature scheme based on discrete logarithms // IEEE transactions on information theory. IEEE, 1985. V. 31. № 4. P. 469–472.
11. *Gentry C.* Fully homomorphic encryption using ideal lattices // Proceedings of the forty-first annual ACM symposium on Theory of computing. Bethesda MD USA: ACM, 2009. P. 169–178.
12. *Van Dijk M. et al.* Fully Homomorphic Encryption over the Integers // Advances in Cryptology – EUROCRYPT 2010 / ed. Gilbert H. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. V. 6110. P. 24–43.
13. *Gentry C., Halevi S.* Implementing gentry's fully-homomorphic encryption scheme // Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15–19, 2011. Proceedings 30. Springer, 2011. P. 129–148.
14. *Brakerski Z.* Fully homomorphic encryption without modulus switching from classical GapSVP // Annual Cryptology Conference. Springer, 2012. P. 868–886.
15. *Brakerski Z., Vaikuntanathan V.* Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages // Advances in Cryptology – CRYPTO 2011 / ed. Rogaway P. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. V. 6841. P. 505–524.

16. *Brakerski Z., Gentry C., Vaikuntanathan V.* (Leveled) Fully Homomorphic Encryption without Bootstrapping // ACM Trans. Comput. Theory. 2014. V. 6. № 3. P. 1–36.
17. *Dijk M. van et al.* Fully homomorphic encryption over the integers // Annual international conference on the theory and applications of cryptographic techniques. Springer, 2010. P. 24–43.
18. *Cheon J.H. et al.* Homomorphic encryption for arithmetic of approximate numbers // International conference on the theory and application of cryptology and information security. Springer, 2017. P. 409–437.
19. *Homomorphic Encryption Standardization* – An Open Industry / Government / Academic Consortium to Advance Secure Computation [Electronic resource]. https://homomorphicencryption.org/ (accessed: 10.12.2022)
20. *Pulido-Gaytan B. et al.* Privacy-preserving neural networks with Homomorphic encryption: Challenges and opportunities // Peer-to-Peer Netw. Appl. 2021. V. 14. № 3. P. 1666–1691.
21. *Ribeiro M., Grolinger K., Capretz M.A.* Mlaas: Machine learning as a service // 2015 IEEE14th international conference on machine learning and applications (ICMLA). IEEE, 2015. P. 896–902.
22. *Manvi S.S., Shyam G.K.* Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey // Journal of network and computer applications. Elsevier, 2014. V. 41. P. 424–440.
23. *Rodero-Merino L. et al.* Building safe PaaS clouds: A survey on security in multitenant software platforms // computers & security. Elsevier, 2012. V. 31. № 1. P. 96–108.
24. *Cusumano M.* Cloud computing and SaaS as new computing platforms // Commun. ACM. 2010. V. 53. № 4. P. 27–29.
25. *Chen H., Chillotti I., Song Y.* Improved bootstrapping for approximate homomorphic encryption // Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part II. Springer, 2019. P. 34–54.
26. Microsoft SEAL: C++. Microsoft, 2023.
27. *OpenFHE.org* – OpenFHE – Open-Source Fully Homomorphic Encryption Library [Electronic resource]. https://www.openfhe.org/ (accessed: 01.04.2024)
28. *Dai W., Sunar B.* cuHE: A homomorphic encryption accelerator library // International Conference on Cryptography and Information Security in the Balkans. Springer, 2015. P. 169–186.
29. *Benaissa A. et al.* TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption: arXiv:2104.03152. arXiv, 2021.
30. *Lee J.-W. et al.* Privacy-preserving machine learning with fully homomorphic encryption for deep neural network // IEEE Access. IEEE, 2022. V. 10. P. 30039–30054.
31. *Halevi S., Shoup V.* Algorithms in helib // Annual Cryptology Conference. Springer, 2014. P. 554–571.
32. *Özerk Ö. et al.* Efficient number theoretic transform implementation on GPU for homomorphic encryption // The Journal of Supercomputing. Springer, 2022. V. 78. № 2. P. 2840–2872.

HIGH-SPEED CONVOLUTION CORE ARCHITECTURE FOR PRIVACY-PRESERVING NEURAL NETWORKS

© 2024 M. A. Lapina[a], E. M. Shiriaev[a], M. G. Babenko[a], I. Istamov[b]

[a] *North Caucasian Center for Mathematical Research, North Caucasus Federal University*
1, Pushkina st., Stavropol, 355017 Russia
[b] *Samarkand State University named after Sharof Rashidov*
15, University blv. Samarkand, 140104 Uzbekistan

Due to legal restrictions or restrictions related to companies' internal information policies, businesses often do not trust sensitive information to public cloud providers. One of the mechanisms to ensure the security of sensitive data in clouds is homomorphic encryption. Privacy-preserving neural networks are used to design solutions that utilize neural networks under these conditions. They exploit the homomorphic encryption mechanism, thus enabling the security of commercial information in the cloud. The main deterrent to the use of privacy-preserving neural networks is the large computational and spatial complexity of the scalar multiplication algorithm, which is the basic algorithm for computing mathematical convolution. In this paper, we propose a scalar multiplication algorithm that reduces the spatial complexity from quadratic to linear, and reduces the computation time of scalar multiplication by a factor of 1.38.

Keywords: matrix operations; artificial neural networks; fully homomorphic encryption; CKKS; TenSEAL, privacy-preserving neural networks