Исследование и разработка алгоритма формальной верификации и метрики оценки качества на основе методов понижения размерности ИНС

Р.М. Селевенко, Е.Н. Строева

***Аннотация*—Проверка качества работы искусственных нейронных сетей (ИНС) при помощи алгоритмов формальной верификации является наиболее гарантированным методом среди прочих, но существует проблема вычислительной сложности и очень больших затрат по времени. Приоритетным направлением исследований является уменьшение времени работы алгоритмов, что можно сделать либо изменяя непосредственно сам алгоритм формальной верификации, либо проводя какие-то действия над проверяемой архитектурой искусственной нейронной сети.**

В данной работе представлено исследование методов формальной верификации исскуственных нейронных сетей, основанных на методах понижения ИНС, описаны проведенные эксперименты с целью проверки свойств у выбранных архитектур нейронных сетей и сделан анализ этих методов.

В качестве алгоритма формальной верификации был взят алгоритм Planet, проверялась работа архитектуры с функцией активации ReLU, поэтому была выбрана LeNet, использовались такие методы понижения размерности, как дропаут, прунинг и квантизация, проверялось свойство достижимости, качество определялось при помощи метрики RNE. Результаты экспериментов показали, что применение прунинга дает выигрыш по времени, но наиболее эффективным оказался дропаут-метод, в то время как применение квантизации для выбраннного алгоритма Planet оказалось невозможным.

***Ключевые слова*—формальная верификация, машинное обучение, dropout, дистилляция, pruning**

I. Введение

Применение и распространение исскуственных нейронных сетей в важных для общества областях таких, как беспилотное управление, медицина, юриспруденция, промышленность, приводит к появлению потребностей в системах проверки свойств исскуственных нейронных сетей для доказательства корректности работы и защиты от состязательных атак.

Применяемое на данный момент тестирование нейронных сетей несовершенно, т.к. опирается на предположение о конечности пространства входных данных, которое зачастую бесконечно.

Также тестирование предполагает прогон нейронных сетей на каждом из выделенных пользователем случаев.

Одна из альтернатив тестированию — формальная верификация. Формальная верификация — это процесс математического доказательства соответствия программы или системы заданным спецификациям. В случае нейронных сетей формальная верификация означает доказательство того, что поведение сети соответствует определенным ожиданиям и безопасным границам.

Формальная верификация нейронных сетей привносит важные аспекты в область обеспечения надежности и безопасности искусственного интеллекта. Посредством строгих математических методов и логических доказательств, данная методология предоставляет надежные гарантии корректности работы нейронных сетей. Рассмотрим ключевые преимущества данного подхода.

Разработчики, имеющие целью провести верификацию, определяют свойства и ограничения для входных данных и искусственной нейронной сети. Например, в задачах автопилотирования самолётов при условии, что угол поворота должен находиться в диапазоне от 0 до 20 градусов, можно назначить, что выходные значения искусственной нейронной сети должны быть в пределах этого диапазона. Это свойство можно поставить как обязательное к исполнению для искусственной нейронной сети.

В данной работе исследование будет сфоркусировано на одном алгоритме формальной верификации Planet, разработанном в 2017 году. Planet – алгоритм, основанный на представлении узлов нейронной сети в качестве линейных условий и сочетании линейного решателя и SMT-решателя. [1].

В данной работе на примере Planet мы решаем проблему высокой сложности алгоритмов формальной верификации и предоставляем исследование методов формальной верификации нейронных сетей и способов по уменьшению количества нелинейных узлов в нейронных сетях.

Проведены эксперименты с целью проверки различных свойств у нейронной сети LeNet и проведен анализ этих методов.

Выявлены основные проблемы методов верификации.

Проведены эксперименты для методов упрощения архитектуры искусственных нейронных сетей, а именно — сжатия, дистиляции, прунинга, квантизации, дропаут-метода и других.

II. Основные определения

Под нейронной сетью будем понимать функцию f, которая переводит множество X в множество Y [2].

Глубокие искусственные нейронные сети (ГНС) состоят из входного слоя, выходного слоя и нескольких

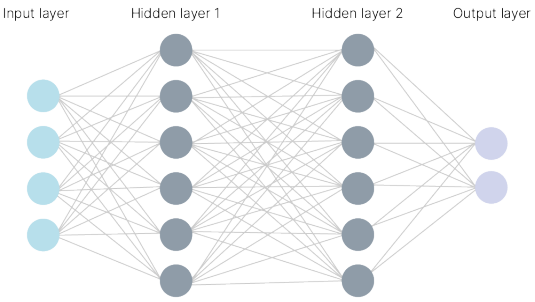

Рис. 1. Пример глубокой нейронной сети

скрытых слоев между ними. Слой состоит из нескольких узлов, каждый из которых связан с узлами предыдущего слоя при помощи заранее заданных весов (рис. 1). Выбор весов критически важен и выполняется во время этапа обучения. Путем назначения значений входам и последующей передачи их через сеть, значения каждого слоя могут быть вычислены из значений предыдущего слоя, что в конечном итоге приводит к значениям выходов.

Значение каждого скрытого узла в сети определяется путем вычисления линейной комбинации значений узлов из предыдущего слоя, а затем применения нелинейной функции активации. В данной работе предполагается, что нелинейная функция активации — это функция активации ReLU, при применении которой к значениям предыдущих узлов её значение вычисляется как максимум из линейной комбинации узлов предыдущего слоя и 0 ($ReLU(x) = max(0, x)$).

Свойство нейронной сети — это утверждение, говорящее о том, что для любого входного значения x из заданной области X выход нейронной сети y будет принадлежать заданной области Y [2]:

$$\forall x \in X \quad \exists f(x) = y : y \in Y.$$

Инженеры, имеющие целью провести верификацию, определяют свойства и ограничения для входных данных и искусственной нейронной сети. Например, в упомянутой выше задачах автопилотирования самолётов обязательное свойство для выходных значений угла поворота может быть строго сформулировано как $\forall x \in X y(x) = \phi in[0; 20]$.

Любой LP-решатель (Linear programming solver) основан на линейном программировании.

Формальная постановка задачи: найти значение x, такое, что:

$$c^T x \to max;$$

$$Ax \leq b;$$

$$x \geq 0.$$

Линейное программирование (Linear Programming, LP) является математическим методом для решения оптимизационных задач, в которых требуется найти наилучшее решение среди возможных, в рамках линейных ограничений. Оно широко используется для моделирования и оптимизации различных процессов и ресурсов. Линейная целевая функция является линейной комбинацией переменных, которые нужно оптимизировать. Цель может состоять в максимизации прибыли, минимизации затрат и т.д.

Линейные ограничения представляют собой линейные неравенства или равенства, которые связывают переменные между собой. Они ограничивают допустимые значения переменных и определяют область поиска оптимального решения. Решение, удовлетворяющее всем линейным ограничениям, называется допустимым решением. Оптимальное решение является допустимым решением с наибольшим или наименьшим значением целевой функции, в зависимости от постановки задачи. Множество всех допустимых решений образует полиэдральную область в многомерном пространстве. Полиэдральная область представляет собой выпуклый многогранник, ограниченный гиперплоскостями, соответствующими линейным ограничениям.

Самым распространненым методом LP программирования является симплекс-метод [3] [4] [5].

Процедура симплекс является конечным итеративным методом, который решает задачи с линейными неравенствами, аналогично решению систем линейных уравнений или матричной инверсии методом Гаусса. [6]

Симплекс-метод [3] является стандартным и высокоэффективным для определения удовлетворяемости TR-связок линейных атомов.

Теория — это пара $T = (\sigma, I)$, где σ — это сигнатура, а I — это класс σ-интерпретаций, моделей T.

Атом — неравенство вида $\sum_{x_i \in X} c_i x_i \{<, >, =\} d$.

III. Основные идеи формальной верификации искусственных нейронных сетей

Задача формальной верификации — гарантировать, что искусственная нейронная сеть выдаёт правильные результаты на заданном множестве данных.

Условие «выдаёт правильные результаты на заданном множестве данных» формулируется в терминах свойств. В общем случае любой алгоритм формальной верификации проверяет заранее сформулированные свойства для уже обученной нейронной сети и, по сути, является бинарным классификатором, выходы которого «свойство выполнено» или «найден контрпример (свойство не выполнено)», блок-схема представлена на рисунке 2.

Рис. 2. Блок-схема принципа формальной верификации нейронной сети

Проверяемые свойства разделяют на два типа:

1. робастности (robustness) — входным значениям, лежащим в определённом диапазоне, соответствует один и тот же класс в выходном множестве. В основном формулируются в задаче поиска контрпримеров.
2. достижимости (reachability) — входным значениям, лежащим в определённом диапазоне, соответствует один и тот же интервал в выходном множестве. Формулируются в задачах с постусловием, когда

необходимо наложение ограничений на выходные данные.

Отметим, что основной способ проверки работоспособности обученной глубокой нейронной сети — тестирование с использованием размеченного набора данных. Но тестирования недостаточно для обеспечения гарантированного результата, поскольку тестирование, по сути, является методом, работающим на дискретном множестве данных, а входные данные для нейронной сети могут принадлежать какому-либо промежутку $[l, u]$, то есть необходимо перейти к методам анализа для непрерывных множеств.

Наглядно эту идею можно проиллюстрировать на примере функции активации ReLU, представленной на рисунке 3.

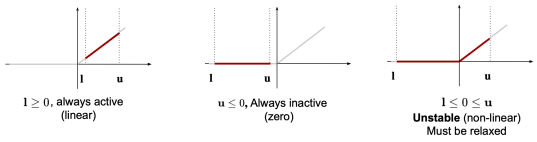

Рис. 3. Задача — проанализировать выходы функции ReLU (Deep Sparse Rectifier Neural Networks)

Если на вход функции ReLU приходит интервал, полностью лежащий в отрицательной полуплоскости, то выходом функции ReLU будет одна точка — ноль, если входной интервал лежит целиком в положительной полуплоскости, то на выходе будет тот же самый интервал. Но в ситуации, когда на вход приходит интервал $[l, u]$, содержащий ноль (или, другими словами, имеющий отрицательную левую границу $l < 0$ и положительную правую границу $u > 0$), непонятно, какой будет результат после применения функции ReLU.

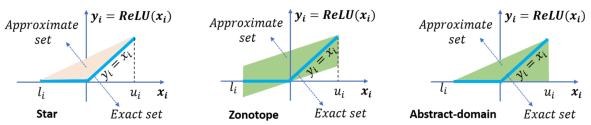

Рис. 4. Возможные варианты приближения функции активации ReLU плоским множеством

Ключевой идеей решения поставленной задачи стал переход к плоскому множеству, внутри которого лежат все возможные варианты выхода для функции ReLU для всего входного интервала (рис. 4). Таким образом множество выходных данных будет заведомо содержать верное выходное значение для проверяемого параметра, который лежит в интервале. То есть неизвестно, в какой именно точке интервала лежит значение этого параметра, но мы точно можем сказать, что в выходном множестве будет содержаться верное выходное значение функции ReLU для любого значения параметра, лежащего в указанном интервале.

Основным отличием всех существующих алгоритмов формальной верификации искусственных нейронных сетей является способы представления плоского множества, которое приближает различные функции активации, а также способ вычислений — посредством уже существующих SAT/SMT-решателей или методов, основанных на переходе к задаче линейного программирования.

Отметим, что общепринятые принципы работы с интервалами, заданные в интервальной арифметике, на каждой операции сложения или умножения существенно расширяют результирующий выходной интервал. Поэтому вторым важным этапом в алгоритмах формальной верификации искусственных нейронных сетей стал переход к символьному представлению интервалов, при котором входной интервал представляется в виде одной переменной, «проходящей» через всю нейронную сеть слой за слоем, нейрон за нейроном. На выходе каждого нейрона получается линейная комбинация, содержащая введённую переменную, после чего происходит возврат к интервальной арифметике и подсчёт выходного интервала с последующей передачей его на следующий слой. При таком подходе выходной интервал становится существенно меньше.

Ещё раз подчеркнём, что алгоритмы формальной верификации очень тщательно анализирует всю нейронную сеть слой за слоем, нейрон за нейроном, предоставляя математические гарантии наличия правильных выходных данных в результирующие выходном множестве. Все возможные выходы (точки) для всех точек из входного интервала $[l, u]$ на каждом нейроне переводятся в плоское множество с минимально возможной площадью, либо с минимальным количеством ограничений, чтобы уменьшить количество вычислений и сократить расширение возможного выходного интервала, полученного вследствие работы с интервальной арифметикой.

Итак, формальная верификация — процесс математического доказательства соответствия программы или системы заданным спецификациям. В случае искусственных нейронных сетей формальная верификация означает доказательство того, что поведение сети соответствует определенным ожиданиям и безопасным границам.

Несмотря на впечатляющие достижения нейронных сетей, они остаются в значительной степени «черными ящиками». Это означает, что сложно предсказать, как сеть будет вести себя вне пределов данных, на которых она была обучена. Ошибки в работе нейронных сетей могут иметь серьезные последствия, особенно в критических областях, таких как медицина или автономная навигация.

Формальная верификация может помочь выявить и устранить потенциальные уязвимости и ошибки в поведении нейронных сетей, позволяя создать более надежные и предсказуемые системы искусственного интеллекта.

Формальная верификация глубокой нейронной сети может осуществляться разными способами.

Дана функция нейронной сети $f_\theta(x)$ (с параметрами θ) с входной областью $D_x \subset R^{k_i}$ и выходной областью $D_y \subset R^{k_o}$, где k_i — это число входных узлов и k_o – это число выходных узлов, решение задачи формальной верификации требует показать, что свойство в текущей форме сохраняется:

$$x \in X \to y = f_\theta(x) \in Y,$$

где $X \subset D_x$ и $Y \in D_y$.

Перечислим основные методы проверки корректности работы искусственной нейронной сети:

1. спецификация: задание требований к поведению искусственной нейронной сети. К примеру, спецификация свойств достижимости, робастности или правильного распознавания объектов в темноте;
2. математическое моделирование: создание математической модели искусственной нейронной сети и ее поведения. Это включает в себя описание архитектуры сети, функций активации, весов и других параметров;
3. формальные методы проверки: применение формальных методов, таких как символьные вычисления и верификация моделей, для доказательства соответствия поведения ИНС заданным спецификациям.

Поскольку современные искусственные нейронные сети содержат в себе миллиарды параметров, алгоритмы верификации искусственных нейронных сетей обладают экспоненциальной сложностью, важно исследовать новые алгоритмы формальной верификации, а также методы упрощения нейронных сетей и методы увличения скорости обучения и верификации [7] [8].

Внедрение формальной верификации может значительно повысить доверие к искусственным нейронным сетям и расширить их применение в критических областях. Алгоритмы формальной верификации могут помочь предсказать и предотвратить нежелательное поведение сетей, обеспечивая безопасность как для людей, так и для систем, которые зависят от работы искусственных нейронных сетей.

Линейные решатели, используемые во многих алгоритмах формальной верификации, основаны на симплекс-алгоритме.

IV. Алгоритм формальной верификации Planet

Рассмотрим алгоритм, основанный на представлении узлов нейронной сети в качестве линейных условий и сочетании линейного решателя и SMT-решателя [1].

Допустим, есть сеть $G = (V, E, W, B, T)$, представляющая функцию $f : R_n \to R_m$. Необходимо создать систему линейных ограничений с использованием V в качестве переменных, которая приближает f, т.е. каждая функция присвоения значения узла является правильным решением системы линейных ограничений, а ограничения как можно более точными.

В данном алгоритме ReLU приближается методом треугольника.

Пусть, c — это взвешенная сумма всех входов с прошлого слоя в текущий узел. Выход узла обозначим как d. Если у нас есть верхние и нижние границы $[l, u]$ для c, то мы можем приблизительно описать связь между c и d с помощью ограничений $d \geq 0, d \geq c$ и $d \leq \frac{u(c-l)}{u-l}$, которые являются линейными уравнениями для u и l.

Для случая узлов MaxPool можно аппроксимировать поведение узлов линейно, аналогично случаю ReLU, за исключением того, что верхние ограничения для значений узлов не нужны.

Пусть $c_1, \ldots, c_k$ — значения узлов с ребрами, ведущими к узлу MaxPool, $l_1, \ldots, l_k$ — их нижние границы, а d — значение выхода узла. Мы задаем следующие линейные ограничения:

$$\wedge_{i\in\{1,\ldots,k\}}(d \geq c_i) \wedge (c_1 + \cdots + c_k) \geq \geq d + \sum_{i\in\{1,\ldots,k\}} l_i - max_{i\in\{1,\ldots,k\}}).$$

После того, как построена линейная программа (для LP-решателя), приближающая поведение всей сети, можно использовать ее, чтобы сделать все будущие приближения еще более точными.

Для этого добавляется спецификация проблемы ψ в виде ограничений и принимается решение для каждой переменной $v \in V$, результирующую линейную программу, минимизируя сначала для целевых функций $1 * v$, а затем для целевой функции $-1 * v$.

Это дает новые, более точные нижние и верхние границы $[l, u]$ для каждого узла (если в сети есть узлы ReLU), которые можно использовать для получения более точной линейной программы.

Включение спецификации в процесс позволяет получать более точные границы, чем можно было бы получить без нее. Весь процесс можно повторять несколько раз: когда получены новые верхние и нижние границы, их можно использовать для построения более точного приближения линейной сети, что в свою очередь позволяет получить новые, более точные верхние и нижние границы.

Далее сформулируем ограничения для каждого нелинейного узла, чтобы сформировать конъюнктивно нормальную форму для решения с использованием SMT-решателя:

1) для каждой ≤ 0 фазы для узла ReLU v, добавим два вида ограничений, первое — $v = 0$ и второе — $\sum_{(v',v)\in E} W((v', v)) * v' + B(v) \leq 0$;
2) для каждой ≥ 0 фазы для узла ReLU v, добавим ограничения $v \geq \sum_{(v',v)\in E} W((v', v)) * v' + B(v)$;
3) для каждой (v', v) фазы для узла MaxPool v, добавим ограничения $v = v'$.

Если результирующая линейная программа недопустима, то можно отбросить все улучшения оценки из рассмотрения в процессе поиска. Это делается путем добавления конфликтного конъюнкта, который исключает булево кодирование этого выбора фазы, так что даже после перезапуска решателя сохраняется причина недопустимости.

Следующий шаг — выведение фазы узла при помощи уже сформированной конъюнктивно-нормальной формы.

На этом шаге также используется линейный решатель. Если набор узлов выполним, то при помощи линейного решателя минимизируется ошибка ReLU путем минимизации разницы между функцией присвоения $a(v)$ и $max(\sum_{v'\in V,(v',v)\in E} W((v', v)) * a(v') + B(v), 0)$ для каждого присваивания a, вычисленного в линейном приближении поведения сети, и каждого узла ReLU v. Соответственно, оценка a может быть использована для вывода дополнительного конъюнкта для SAT-решателя.

Также с помощью приема вывода фаз из фаз предыдущего слоя можно существенно сократить конъюнктивно нормальную форму и упростить задачу по времени.

Возьмем, например, участок сети из рисунка 5.

В ней есть два узла ReLU, названных r_1 и r_2, и один узел MaxPool. Предположим, что во время первоначального анализа сети было определено, что значение узла r_1

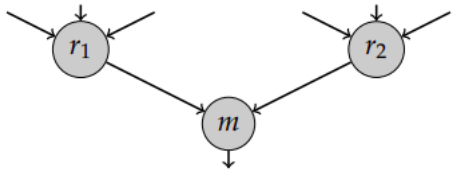

Рис. 5. Пример нейронной сети для вычисления выходного интервала.

находится между 0,0 и 1,5, а значение узла r_2 — между 0,1 и 2,0.

Во-первых, SAT решатель может определить, что узел r_2 находится в фазе ≥ 0.

Затем, если в какой-то момент SAT решатель решает, что узел r_1 должен находиться в фазе ≤ 0, то фиксирует значение r_1 на 0. Поскольку поток из r_2 имеет нижнюю границу > 0, мы можем заключить, что фазу m следует установить на (r_2, m).

Algorithm 1 Top-level view onto the neural network verification algorithm.

```
1:  function VerifyNN(V, E, T, B, W)
2:      (min→, max→) ← ComputeInitialBounds(V, E, T, B, W)        ▷ Section 3.1
3:      (min→, max→) ← RefineBounds(V, E, T, B, W, min→, max→)    ▷ Section 3.1
4:      p ← ∅, extra ← ∅
5:      ψ ← ⋀_{v∈V,T(v)=MaxPool} ⋁_{v'∈V,(v',v)∈E} x_{v,(v',v)}
6:      ψ ← ψ ∧ ⋀_{v∈V,T(v)=MaxPool,v',v''∈V,v'≠v'',(v',v)∈E,(v'',v)∈E} (¬x_{v,(v',v)} ∨ ¬x_{v,(v'',v)})
7:      ψ ← ψ ∧ ⋀_{v∈V,T(v)=ReLU} (x_{v,≤0} ∨ x_{v,≥0}) ∧ (¬x_{v,≤0} ∨ ¬x_{v,≥0})
8:      while ψ has a satisfying assignment do
9:          while extra is non-empty do
10:             Perform unit propagation, conflict detection, backtracking, and clause
11:             learning for p on ψ, while moving the clauses from extra to ψ one-by-one.
12:         extra ← InferNodePhases(V, E, T, B, W, p, min→, max→)   ▷ Section 3.4
13:         if extra = ∅ then
14:             extra ← CheckForFeasibility(V, E, T, B, W, p, min→, max→)   ▷ Section 3.2-3.3
15:             if p ⊨ c for all clauses c ∈ extra then
16:                 if p is a complete assignment to all variables then
17:                     return Satisfiable
18:                 Add a new variable assignment b↦ true to p for some variable b in ψ.
19:                 if p cannot be extended to a satisfying valuation to ψ then
20:                     p = p \ {b ↦ true} ∪ {b ↦ false}
21:     return Unsatisfiable
```

Рис. 6. Псевдокод функции VerifyNN.

Алгоритм 1 на рисунке 6 показывает общий подход алгоритма Planet.

На первом шаге вычисляются верхние и нижние границы значений для всех узлов. Затем решатель готовит пустую частичную оценку для переменных SAT и пустой список extra, в котором хранятся дополнительные дизъюнкты, созданные шагами анализа экземпляра LP, предложенными в данном разделе. Экземпляр SAT инициализируется дизъюнктами, которые обеспечивают, чтобы каждый узел ReLU и каждый узел MaxPool выбрали ровно одну фазу.

В основном цикле алгоритма первым шагом выполняются основные действия по решению SAT, такие как распространение значений, обнаружение и анализ конфликтов и другие.

Дополнительные условия из extra добавляются к экземпляру SAT ψ пошагово, поскольку эти дополнительные условия могут вызвать дополнительное распространение значений и даже конфликты, которые нужно немедленно решать.

После того, как все условия из extra добавлены в ψ, и частичная оценка была расширена следствиями литералов, которые мы вывели в строке 12, применяется вывод из нескольких фаз.

Если он возвращает новые следствия (в виде дополнительных условий), они обрабатываются следующими шагами решения SAT в строке 11. Это происходит потому, что уже имеющиеся условия в ψ могут привести к распространению значений на вновь выведенные литералы. Только когда все фазы узлов были выведены, проверяется выполнимость в линейном приближении (строка 14).

После проверки есть две возможные ситуации: если LP проблема невыполнима (infeasible), то создается новый конфликтный дизъюнкт, и условие на строке 15 не выполняется. В этом случае алгоритм продолжает выполнение снова с линии 11. В противном случае выполняется шаг ветвления SAT-решателя. Если p уже является полной оценкой, мы знаем, что экземпляр удовлетворим, так как функция CheckForFeasibility, которая только что была выполнена, работала с LP-проблемой, которая не является приближенной, а точно описывает поведение сети. В противном случае, p расширяется решением установки переменной b в true (для некоторой переменной, выбранной эвристиками выбора переменных SAT-решателя). Когда это происходит, используется обычный SAT-решатель для проверки того, может ли частичная оценка быть расширена до оценки, удовлетворяющей ψ.

V. Методы понижения ИНС

Поскольку скорость работы алгоритма формальной верификации зависит от количества нелинейных узлов в ИНС, а также количества линейных приближений, имеет смысл уменьшать количество узлов в ИНС, а также количество уникальных значений в матрице весов ИНС.

A. Прунинг

Учитывая набор входных данных $M = \{(X, Y)\}$, где каждый вход представляет собой последовательность слов $X = x_1, ..., x_t, ..., x_{N_x}$, а N_x обозначает длину входных данных, основная цель состоит в том, чтобы предсказать метку(и) для X, обозначенную(ые) как Y.

В стандартной многослойной искусственной нейронной сети входной слой сначала отображает каждый входной токен x_t в векторное представление $h_0^t \in R^{D\times 1}$, где D обозначает размерность. Поверх входного слоя модель накладывает L промежуточных нейронных слоев. Пусть $h_l^t \in R^{D\times 1}$ обозначает представление токена x_t на l-м уровне. $H_l \in R^{D\times N}$ — это объединение представлений на l-м уровне для всех токенов на входе X. Каждый уровень сети включает в себя несколько операций, таких как полносвязные операции, ReLU, операции внимания или остаточные соединения. Группа всех операций внутри слоя l обозначается F_l, что отображает H_l в H_{l+1}:

$$H^{l+1} = F_l(H^l).$$

Выходные данные последнего слоя h_L^t передаются на последний линейный слой для прогнозирования. Чтобы сократить модель нейронной сети, пусть $m^l \in \{0,1\}^{D\times 1}$ обозначает маску для измерений представления на слое l. Количество единиц в m^l — это заранее определенный гиперпараметр, обозначаемый K, контролирующий разреженность сети. $M^l \in \{0,1\}^{D\times N}$ создает N копий m^l, делая размерность маски такой же, как и размерность представлений слоев для X. Пусть u^l обозначает набор

индексов для сохраненных измерений, где $m^l[j] = 1$ для j в u_l.

Ключевым моментом послойного прунинга является построение корреляций между измерениями в двух последовательных слоях $l-1$ и l. Затем, основываясь на корреляциях, мы можем сократить сеть: мы выбираем K верхних коррелированных измерений в L-м слое на основе меры корреляции, обнуляя остальные. Пусть $I(A, B)$ обозначает корреляцию между двумя наборами измерений:

$$u^L = argmax(I(u, u^{linear})).$$

Далее мы переходим к $(L-1)$-му слою, сохраняя в $(L-1)$-м слое измерения, которые наиболее коррелируют с сохраняемыми размерами в L-м слое.

$$u^{L-1} = argmax(I(u, u^L)).$$

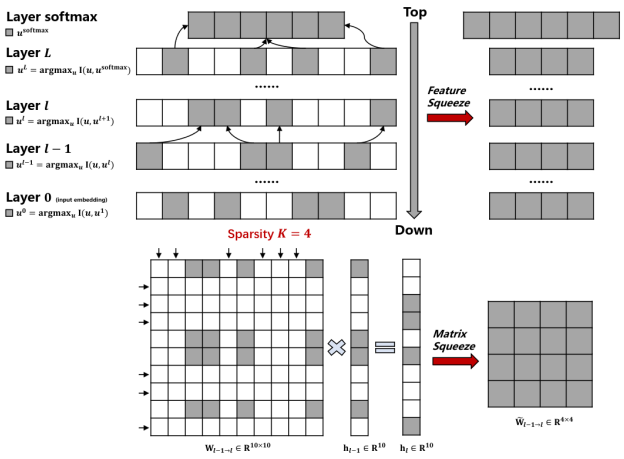

Рис. 7. Обзор предлагаемого метода послойного прунинга. Верхняя часть показывает сокращение на уровне объекта, а нижняя часть показывает сокращение на уровне весовой матрицы. Послойное сокращение сначала выбирает размеры объектов в каждом слое с учетом некоторого критерия корреляции $I(\cdot, \cdot)$, а затем сокращает строки и столбцы матрицы в соответствии с выбранными измерениями на последовательных слоях, после чего можно сжимать как объекты, так и матрицы.

Этот процесс продолжается до нижнего входного эмбеддинг слоя. Иллюстрация предложенного метода послойного прунинга показана на рисунке 7.

B. Дропаут

Dropout — это метод, похожий на прунинг, который решает проблему сложной архитектуры ИНС методом уменьшения количества узлов в ИНС. Термин «дропаут» относится к выпадающим узлам (скрытым и видимым) в искусственной нейронной сети. Под удалением узла подразумевается временное удаление его из сети вместе со всеми его входящими и исходящими соединениями, как показано на рисунке 8. Выбор того, какие устройства удалять, является случайным. В простейшем случае каждый узел сохраняется с фиксированной вероятностью p, независимой от других единиц, причем p можно выбрать с помощью набора проверки или просто установить на уровне $0,5$, что кажется близким к оптимальному для широкого круга сетей и задания. Однако для входных

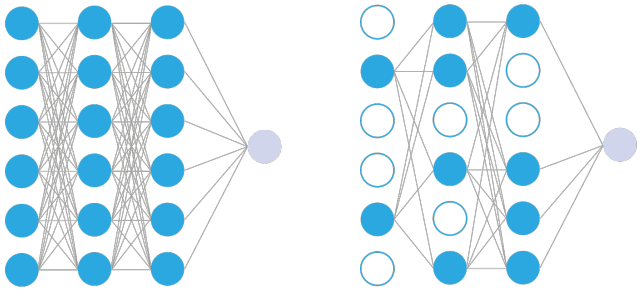

Рис. 8. Модель нейронной сети с дропаутом. Слева: стандартная искусственная нейронная сеть с двумя скрытыми слоями. Справа: пример сети, полученной путем применения дропаута к сети слева. Перекрещенные юниты были отброшены.

единиц оптимальная вероятность удержания обычно ближе к 1, чем к $0,5$.

Скорость алгоритма формальной верификации зависит от количества приближений нелинейных узлов. Соответственно для уменьшения работы ИНС и алгоритма формальной верификации на этой ИНС, следует уменьшить количество нелинейных узлов, таким образом, новый алгоритм включает в себя добавления дропаут-слоя после каждого слоя активации.

C. Квантизация

Предположим, что ИНС имеет L слоев с обучаемыми параметрами, обозначенными как $\{W_1, W_2, ..., W_L\}$, где θ обозначает комбинацию всех таких параметров. Без потери общности сосредоточимся на задаче обучения с учителем, номинальной целью которой является оптимизация следующей эмпирической функции минимизации риска:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(x_i; y_i; \theta),$$

где (x, y) — входные данные и соответствующая метка, $l(x, y; \theta)$ — функция потерь (например, среднеквадратическая ошибка или потеря перекрестной энтропии), а N — общее количество точек данных.

Обозначим также входные скрытые активации i-го слоя как h_i, а соответствующую выходную скрытую активацию как a_i. Предполагается, что у есть параметры обученной модели θ, хранящиеся с точностью float64. Целью квантизации является снижение точности как параметров (θ), так и промежуточных карт активации (т. е. h_i, a_i) до низкой точности с минимальным влиянием на мощность/точность обобщения модели. Для этого нам нужно определить оператор квантизации, который отображает значение с плавающей запятой в квантизированное значение, что описано ниже.

Сначала необходимо определить функцию, которая может квантовать веса и активации ИНС до конечного набора значений. Эта функция принимает реальные значения с плавающей запятой и отображает их в диапазон более низкой точности. Популярным выбором функции квантования является следующий:

$$Q(r) = Int(r/S) - Z,$$

где Q — оператор квантования, r — действительный входной сигнал (активация или вес), S — действительный масштабный коэффициент, а Z — целочисленная нулевая точка. Кроме того, функция Int сопоставляет вещественное значение с целочисленным посредством операции округления (например, округления до ближайшего значения). По сути, эта функция представляет собой отображение действительных значений r в некоторые целочисленные значения. Этот метод квантования также известен как равномерное квантование, поскольку результирующие квантованные значения (так называемые уровни квантования) расположены равномерно (рис. 8, слева).

Существуют также методы неравномерного квантования, квантованные значения которых не обязательно расположены равномерно. Можно восстановить действительные значения r из квантованных значений $Q(r)$ с помощью операции, которую часто называют деквантованием:

$$r = S(Q(r) + Z).$$

Одним из важных факторов равномерной квантизации является выбор масштабного коэффициента S в уравнении $Q(r) = Int(r/S) - Z$. Этот масштабный коэффициент по существу делит заданный диапазон действительных значений r на несколько разделов:

$$S = \frac{\beta - \alpha}{2^b - 1},$$

где $[\alpha, \beta]$ обозначает диапазон отсечения, ограниченный диапазон, с помощью которого отсекаются реальные значения, а b — разрядность квантования. Следовательно, чтобы определить коэффициент масштабирования, сначала следует определить диапазон отсечения $[\alpha, \beta]$. Процесс выбора диапазона отсечения часто называют калибровкой.

Простой выбор — использование минимального/максимального сигнала для диапазона ограничения, т.е. $\alpha = r_{min}$ и $\beta = r_{max}$. Этот подход представляет собой схему асимметричного квантования, поскольку диапазон ограничения не обязательно симметричен относительно начала координат, т. е. $-\alpha = \beta$. Также возможно использовать схему симметричного квантования, выбрав симметричный диапазон ограничения $\alpha = -\beta$.

Популярный выбор — выбирать их на основе минимальных/максимальных значений сигнала: $-\alpha = \beta = max(|r_{max}|, |r_{min}|)$. Асимметричное квантование часто приводит к более узкому диапазону ограничения по сравнению с симметричным квантованием. Это особенно важно, когда целевые веса или активации несбалансированы, например, активация после ReLU, которая всегда имеет неотрицательные значения. Однако использование симметричного квантования упрощает функцию квантования, заменив нулевую точку на $Z = 0$:

$$Q(r) = Int\left(\frac{r}{S}\right).$$

Здесь есть два варианта масштабного коэффициента.

В «полном диапазоне» симметричное квантование S выбирается как

$$\frac{2max(|r|)}{2^n - 1}$$

(с режимом округления нижнего предела), чтобы использовать полный диапазон $INT8$ $[-128, 127]$. Однако в «ограниченном диапазоне» S выбирается как

$$\frac{max(|r|)}{2^{n-1} - 1},$$

который использует только диапазон $[-127, 127]$.

Как и ожидалось, метод полного диапазона является более точным. Симметричное квантование широко применяется на практике для квантования весов, поскольку обнуление нулевой точки может привести к снижению вычислительных затрат во время вывода $[255]$, а также делает реализацию более простой.

Симметричная квантизация разделяет отсечение, используя симметричный диапазон. Это упрощает реализацию, поскольку приводит к $Z = 0$ в уравнении. Однако это неоптимально для случаев, когда диапазон может быть перекошенным и несимметричным. В таких случаях предпочтительнее асимметричная квантизация.

До сих пор рассматривались различные методы калибровки для определения диапазона отсечения $[\alpha, \beta]$. Еще одним важным отличием методов квантизации является определение диапазона ограничения. Этот диапазон можно вычислить статически для весов, поскольку в большинстве случаев параметры фиксируются во время вывода. Однако карты активации различаются для каждого входного образца. Таким образом, существует два подхода к квантованию активаций: динамическая квантизация и статическая квантизация.

При динамической квантизации этот диапазон динамически рассчитывается для каждой карты активации во время выполнения. Этот подход требует вычисления статистики сигнала в реальном времени (минимум, максимум, процентиль и т. д.), что может иметь очень высокие накладные расходы. Однако динамическая квантизация часто приводит к более высокой точности, поскольку диапазон сигнала точно рассчитывается для каждого входа.

Другой подход к квантизации — статическая квантизация, при которой диапазон отсечения заранее рассчитывается и статичен во время вывода. Этот подход не добавляет никаких вычислительных затрат, но обычно приводит к более низкой точности по сравнению с динамической квантизацией. Одним из популярных методов предварительного расчета является выполнение шести серий калибровочных входных данных для расчета типичного диапазона активаций.

Для поиска наилучшего диапазона было предложено несколько различных метрик, включая минимизацию среднеквадратической ошибки (MSE) между исходным неквантованным распределением веса и соответствующими квантованными значениями. Можно также рассмотреть возможность использования других показателей, таких как энтропия, хотя MSE является наиболее распространенным методом. Другой подход заключается в изучении/наложении этого диапазона отсечения во время обучения искусственной нейронной сети. Заметными работами здесь являются LQNets, PACT , LSQ и LSQ+ [9], которые совместно оптимизируют диапазон отсечения и веса в ИНС во время обучения.

Динамическая квантизация динамически вычисляет диапазон ограничения каждой активации и часто дости-

гает высочайшей точности. Однако динамический расчет диапазона сигнала очень дорог с точки зрения времени, и поэтому практики чаще всего используют статическое квантование, при котором диапазон ограничения фиксирован для всех входов.

VI. Практическая реализация

A. Программное и аппаратное обеспечение

Алгоритм Planet написан на C++ и использует линейное программирование через пакет GLPK 4.611 и SAT-решатель Minisat 2.2.0 [10].

Все численные вычисления выполняются с точностью double. Все временные эксперименты, указанные далее, были получены на компьютере с процессором Intel® Core™ i7-8750H CPU @ 2.20GHz × 12 и 15,5 ГБ оперативной памяти под управлением x64 версии GNU/Linux. Использование памяти всегда было менее 1 ГБ. Все инструменты запускались с одним вычислительным потоком. Процесс прерываля вручную, если время его работы превышало 25 000 секунд.

B. Набор данных и архитектура верифицируемой нейронной сети

Набор данных MNIST представляет собой изображения представленнные в оттенках серого и имеют размер 28×28 пикселей. Упрощенная сеть, для которой проводились эксперименты, имеет следующую архитектуру:

1) один входной слой с 28×28 узлами;
2) один сверточный слой с $3 \times 13 \times 13$ узлами, где каждый узел имеет 16 входных ребер;
3) один слой пулинга с $3 \times 4 \times 4$ узлами, где каждый узел имеет 16 входных ребер;
4) один слой ReLU с 8 узлами;
5) один выходной слой ReLU с 10 узлами.

Слои ReLU полностью связаны. В целом сеть имеет 522 узла, пространство поиска для фаз узлов имеет размер $163 * 4 * 4 * 2^8 * 2^{10} = 2^{162}$, а в сети имеется 4672 ребра.

Эта архитектура была использована для обучения сети на 100 000 обучающих изображениях из набора данных, и полученная сеть показывает точность $95,05\%$ на отдельном тестовом наборе данных.

VII. Эксперименты, направленные на уменьшение времени работы алгоритмов

Первая экспериментальная задача была сформулирована как повторение результатов статьи [3], а именно — проверить, сможет ли алгоритм Planet проверифицировать свойства достижимости и робастности на архитектуре LeNet $f : R^n \to R^m$ [11].

A. Эксперимент проверки времени работы алгоритма Planet для свойств достижимости

Определим свойство достижимости как свойство «картинка классифицируется строго как класс 2». То есть, вероятность классификации изображения как 2 сильно больше, чем вероятность его классификации любым другим классом (в нашем случае это любые другие цифры).

Формально: необходимо получить входное изображение $(x_{1,1}, \ldots, x_{28,28})$, для которого сеть выводит вектор $(y_0, \ldots, y_9)$, удовлетворяющий условию $y_2 \geq y_i + \delta$ для всех $i \in \{0, 1, 3, 4, 5, 6, 7, 8, 9\}$ при большом значении δ.

Мы верифицируем это свойство используя δ в диапазоне $[0, 30]$. Проверку свойств для больших δ произвести не получилось из-за длительного времени работы программы.

Результаты экспериментов для свойств достижимости представлены на графике 9. Ось x на графике — это значение δ, а ось y — это значение времени затраченного на запуск программы. Здесь можно увидеть зависимость времени работы программы формальной верификации от значения δ в свойствах достижимости.

Такой характер функции объявняется тем, что с увеличением требований для разрыва между интересующим нас классом 2 и всеми остальными классами, нам сложнее искать картинку удовлетворяющую таким требованиям.

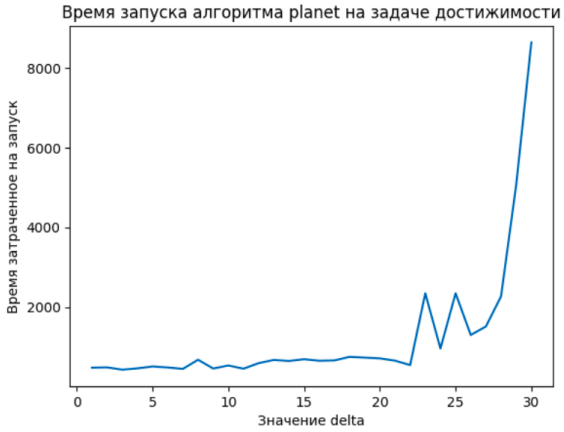

Рис. 9. Время работы верификации planet для свойств достижимости. Ось x — значение δ, а ось y — значение времени затраченного на запуск программы.

B. Эксперимент проверки свойств робастности для алгоритма Planet

Интересно проверить, сколько шума можно добавить к изображениям, прежде чем они перестанут быть правильно классифицированными.

Дано изображение, представленное на рисунке 10.

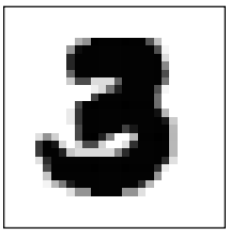

Рис. 10. Картинка для тестирования свойства робастности.

Необходимо проверить, существует ли другое изображение, которое будет классифицировано как 4, но при этом каждый пиксель будет иметь значения, которые находятся в абсолютном диапазоне $\pm\epsilon\%$ от интенсивности цвета пикселей оригинального изображения, при этом мы сохраняем пиксели, которые находятся не более чем

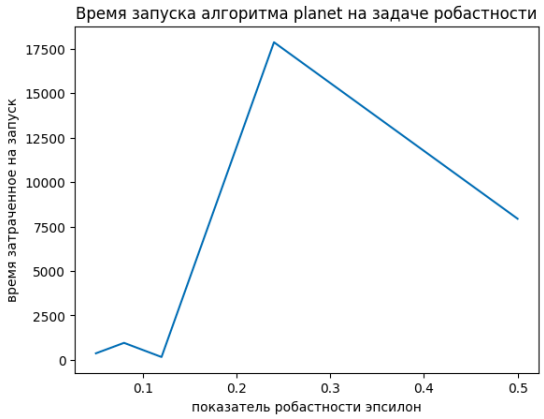

Рис. 11. Время работы верификации planet для свойств робастности. По оси x — значения ϵ, по оси y — значение времени затраченного на запуск программы.

в трех пикселях от границ (чтобы вносить шум только в цифру, а не в белый фон).

Результаты данного эксперимента показаны на рисунке 11 для значений $\epsilon \in \{0.08, 0.12, 0.24, 0.5\}$. Здесь по оси x — значения ϵ, по оси y — значение времени, затраченного на запуск программы. Можно увидеть отсутствие паттерна увеличения времени с увеличением параметра шума. Это можно объяснить тем, что в данном эксперименте количество картинок, которое необходимо проверить на выполнение заданного свойства, не увеличивается с увеличением параметра. Шум ϵ всегда добавляется к фиксированному количеству пикселей. Соответственно, и при увеличении параметра ϵ, время работы программы не увеличивается (таблица I).

Таблица I
Время в секундах, затраченное на запуск всех тестов указанного свойства робастности.

Время \ Алгоритм верификации	planet	alpha-beta-crown
0.08	958	60
0.12	162	65
0.24	17880	3240
0.5	7944	35

C. Вывод

Современные алгоритмы формальной верификации могут использоваться для проверки различных свойств небольших искусственных нейронных сетей. Однако, эти методы могут столкнуться с проблемами при проверке больших искусственных нейронных сетей и свойств, время проверки которых зависит от параметров этих свойств, например, свойства достижимости, упомянутого выше. В основном это связано с высоким временем выполнения аглоритмов формальной верификации. Такое большое время работы зависит от количества параметров нейронной сети, поданной на вход, и структурой свойств, которые нужно проверить. Как видно из второго эксперимента, алгоритмы формальной верификации отлично подходят для верификации свойства робастности, т.к. время затраченное на работу алгоритма формальной верификации не так сильно зависит от величины параметра шума ϵ. Это видно на рисунке 11.

Кроме того, стоит отметить, что большинство современных алгоритмов формальной верификации не поддерживают аппроксимацию или представление других нелинейных слоев нейронных сетей, кроме слоя ReLU.

Таким образом, потребность в уменьшении времени работы алгоритмов формальной верификации действительно существует.

VIII. Эксперименты с методами понижения ИНС

В работе проводились эксперименты с алгоритмом Planet, т.к. он является модификацией стандартного алгоритма ReluPlex, выполняющегося одним потоком, а значит вычисления на нём будут наиболее репрезентативны.

Цель эксперимента — найти такой метод понижения размерности ИНС, что при применении его к оригинальной ИНС, итоговая ИНС имеет такое же поведение, что и оригинальная ИНС, но с меньшим временем.

Проверифицируем свойства достижимости у архитектуры нейронной сети LeNet: $R^n \to R^m$.

Определим свойство достижимости как свойство: «картинка классифицируется строго как класс 2». То есть, вероятность классификации изображения как 2 сильно больше, чем вероятность его классификации любым другим классом (в нашем случае это любые другие цифры). Формально, мы хотели получить входное изображение $(x_{1,1} \dots x_{28,28})$, для которого сеть выводит вектор $(y_0, \dots, y_9)$, удовлетворяющий условию $y_2 \geq y_i + \delta$ для всех $i \in 0, 1, 3, 4, 5, 6, 7, 8, 9$ при большом значении δ. Мы верифицируем это свойство (рис. 12) используя δ в диапазоне [5, 18].

Algorithm 9.1: Свойство достижимости.

Parameter digit: int = 2, delta: float

:

```
1 Network(N)
2 for x do
3     Implies(0 ⩽ x ⩽ 1) Or( for i = 0 to 9 do
4         if digit ≠ i then
5             (N(x)[0][digit] < N(x)[0,i] + delta)
6     )
```

Рис. 12. Псевдокод свойства достижимости

Далее мы будем изменять δ в описанном диапазоне и смотреть на поведение моделей на нём.

A. Эксперименты с добавлением прунинга

Для начала был проведён прунинг оригинальной модели. Прунинг удаляет первые n процентов узлов в весовых слоях, которые наиболее близки к 0, а значит, вносят наименьшие изменения. По нашему предположению время работы алгоритма формальной верификации должно уменьшаться вместе с точностью модели на валидации. Прунинг проводился с шагом 10%: 10%, 20%, 30%. Прунинг с большим параметром оказался нецелесообразным из-за слишком большого количества занулённых узлов и маленькой точности. Время в таблицах приведено в секундах.

В таблице II и на графике 13 можно видеть зависимость времени работы алгоритма формальной верификации от параметра δ по горизонтали и параметра процента

Таблица II
Изменение времени работы формальной верификации в зависимости от процента отключенных узлов в нейронной сети.

delta	6	7	8	9	10	11	12	13	14	15	RNE
эталон	95.4020	117.9088	137.2168	1433.9259	>3600	>3600	>3600	>3600	80.9785	70.0088	_
10%	78.7034	79.8195	85.7101	68.6157	>3600	>3600	>3600	>3600	14.9456	16.3949	**74.35**
20%	>3600	11.5040	2.7505	2.7423	>3600	>3600	>3600	>3600	3.2105	2.8661	-28.56
30%	27.5538	29.3362	30.2423	2.5213	>3600	>3600	>3600	>3600	2.4174	2.7054	1.12
	SAT	SAT	SAT	SAT	SAT	SAT	SAT	SAT	UNSAT	UNSAT	
	Ответы эталона										

Таблица III
Изменение времени работы формального верификатора в зависимости от процента отключенных активационных узлов в искусственной нейронной сети.

delta	6	7	8	9	10	11	12	13	14	15	16	RNE
эталон	95.4020	117.9088	137.2168	1433.9259	>3600	>3600	>3600	>3600	80.9785	70.0088	60.004	_
10%	63.9976	63.8171	61.0345	77.7890	197.4022	2731.8460	>3600	>3600	>3600	>3600	11.2205	**264.322**
20%	58.0651	67.9593	64.4821	152.1101	645.7324	>3600	>3600	>3600	>3600	>3600	131.3130	122.796
40%	59.1461	66.0422	107.7267	71.7780	>3600	>3600	>3600	>3600	83.8742	16.7394	15.3227	8.115
50%	67.7802	70.8767	69.3697	73.0192	>3600	>3600	>3600	14.6314	14.5573	16.7816	15.8879	11.203
	SAT	SAT	SAT	SAT	SAT	SAT	SAT	SAT	UNSAT	UNSAT	UNSAT	
	Ответы эталона											

Таблица IV
Изменение точности ИНС в зависимости от процента отключенных узлов в нейронной сети.

эталон	10%	20%	30%
95.03	92.89	84.11	58.58

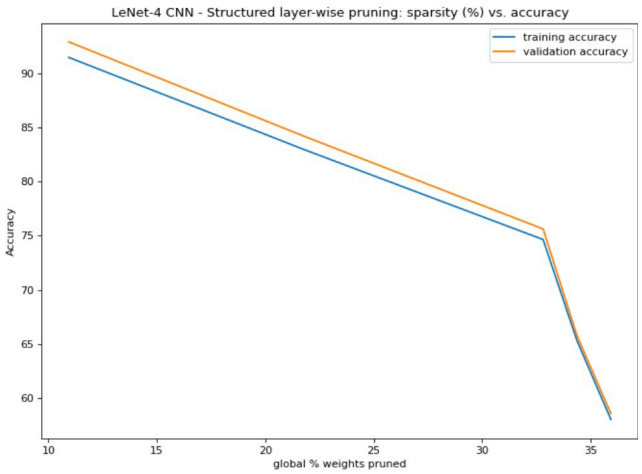

Рис. 13. На графике можно видеть зависимость падения точности от процента удалённых узлов в методе прунинга.

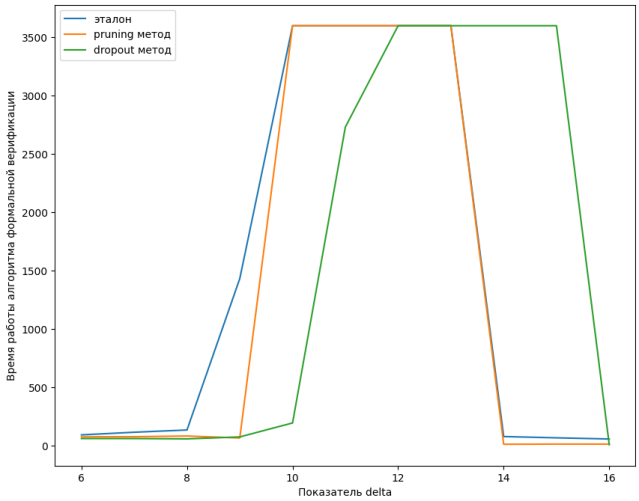

Рис. 14. График времени работы алгоритма формальной верификации Planet для разных ИНС и свойства достижимости при различных delta.

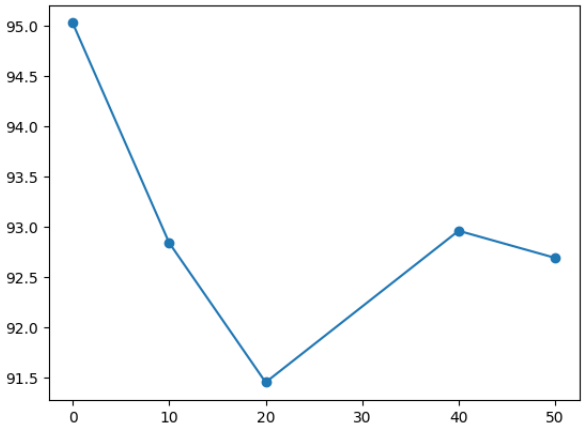

Рис. 15. График изменения точности в зависимости от процента отключенных узлов в нейросети в методе DropOut.

занулённых узлов во вертикали. В таблице III и на графике 14 можно видеть зависимость падения точности от процента удалённых узлов в методе прунинга.

Как можно видеть из результатов эксперимента, график времени работы алгоритма формальной верификации представляет собой обратную параболу, где слева от пика эксперименты выдают ответ SAT, а справа UNSAT. Это происходит потому, что подобрать вариант с большей вероятностью классификации при добавлении меньшего δ легче, чем при больших δ. Но при этом при очень больших δ сразу видно, что таких показателей ни в одном классе не присутствует.

Можно видеть, что время работы действительно понизилось. При этом важно заметить, что в случаях с двадцатью и тридцатью процентами занулённых узлов поведение алгоритма изменилось, точность снизилась. Счита-

ем, что наибольшее уменьшение времени при меньшем падении эффективности достигается на 10%.

B. Эксперименты с добавлением дропаута

Дропаут-слой добавляется после каждого слоя активации, таким образом обнуляя n случайных узлов. В таблице IV можно увидеть зависимость времени работы алгоритма формальной верификации от параметра δ по горизонтали и параметра процента занулённых узлов во вертикали для дропаут-метода.

Метод дропаута также оказался успешным, причем изначальное удаление 10% узлов на всех слоях активации даёт наиболее эффективное улучшение алгоритма (таблица V).

Последующее увеличение процента удалённых узлов смещает график времени работы формальной верификации, при отсутствии сохранения поведения на этих экспериментах. При удалении 10% узлов на слоях активации поведение тоже не сохраняется, но только лишь из-за увеличения робастности. Это можно видеть из смещенного графика времени работы алгоритма вправо (рис. 15).

Таблица V
Изменение точности ИНС в зависимости от процента отключенных узлов в нейросети для метода DropOut.

эталон	10%	20%	40%	50%
95.03	92.84	91.45	92.96	92.69

C. Эксперименты с добавлением квантизации

Для реализации метода квантизации требуется изменение алгоритма формальной верификации Planet. Новую матрицу весов нельзя оценивать с помощью ограничений треугольника $d \leq \frac{u(c-l)}{u-l}$, и данный метод потребует разработки нового метода ограничений. Сами приближения ограничений также будут некорректны в оригинальном алгоритме, т.к. изменение матрицы весов ИНС требует и изменения ограничений, наложенных на эту матрицу. Соответственно, данный метод не позволяет изменить представленную модель без последствий для алгоритма и требует дальнейших исследований.

D. Вывод

Было исследовано влияние алгоритмов понижения размерности ИНС на алгоритмы формальной верификации. В результате исследования было выявлено, что десятипроцентное понижение количества узлов методом дропаута даёт понижение времени работы в 2 раза на SAT части и на 30% на всех экспериментах. Также достигается увеличение робастности с показателя 13 до показателя 15.

Также было исследовано влияние прунинга на время работы алгороитма формальной верификации. С помощью уменьшения количества нелинейных узлов удалось снизить время работы алгоритма формальной верификации в 2 раза на SAT части. Согласно метрике RNE оптимальный параметр процента удалённых узлов - 10%. При дальнейшем удалении узлов модель уменьшает метрику точности.

Был разработан метод сравнения алгоритмов формальной верификации с помощью метрики RNE. Согласно этому методу наилучшая модель имеет наименьшее отклонение по точности и наибольшее по времени от оригинальной модели на меньшем количестве экспериментов.

Исходя из результатов экспериментов и показателей метрики RNE, можно заключить, что наиболее успешным алгоритмом формальной верификации является алгоритм с применением метода дропаута. Это происходит потому, что дропаут метод удаляет непосредственно слои активации, а также способствует повышению робастности модели, что даёт сильное уменьшение времени работы алгоритма формальной верификации на одних и тех же свойствах. Дропаут-метод также даёт возможность понизить время работы алгоритма формальной верификации при небольшом изменении точности модели.

Современные алгоритмы формальной верификации могут использоваться для проверки различных свойств небольших искусственных нейронных сетей. Однако, эти методы могут столкнуться с проблемами при проверке больших искусственных нейронных сетей и свойств, время проверки которых зависит от параметров этих свойств, например, свойства достижимости, упомянутого выше. В основном это связано с высоким временем выполнения алгоритмов формальной верификации. Такое большое время работы зависит от количества параметров нейронной сети, поданной на вход, и структурой свойств, которые нужно проверить. Как видно из второго эксперимента, алгоритмы формальной верификации отлично подходят для верификации свойства робастности, т.к. оно не зависит от величины параметра шума ϵ.

Кроме того, стоит отметить, что большинство современных алгоритмов формальной верификации не поддерживают аппроксимацию или представление других нелинейных слоев нейронных сетей, кроме слоя ReLU.

IX. Заключение

Задача понижения сложности алгоримтов и уменьшения времени работы алгоритмов формальной верификации сложная, но в ней существует потребность, подход с добавлением дропаута показал неплохие результаты. В дальнейшем следует разработать алгоритм оптимального подбора параметров для методов понижения ИНС, чтобы автоматизировать процесс выбора наилучшего метода. Также следует рассматривать алгоритмы, которые не только рассматривают алгоритм формальной верификации ИНС как черный ящик, структуру которого мы не можем менять, но и изменют структуру алгоритмов верификации. Например, можно изменить алгоритм формальной верификации Planet для возможности поддержки им квантизированных слоёв. Для этого следует учесть, что значения квантизированных слоёв являются целочисленными.

X. Благодарности

Мы благодарны сотрудникам кафедры Информационной безопасности факультета ВМК МГУ имени М.В. Ломоносова за ценные обсуждения данной работы. Исследование выполнено при поддержке Междисциплинарной

научно-образовательной школы Московского университета «Мозг, когнитивные системы, искусственный интеллект». Статья является продолжением серии публикаций, посвященных устойчивым моделям машинного обучения и кибербезопасности систем искусственного интеллекта. Она подготовлена в рамках проекта кафедры Информационной безопасности факультета ВМК МГУ имени М.В. Ломоносова по созданию и развитию магистерской программы «Искусственный интеллект в кибербезопасности».

Список литературы

[1] Ehlers. R. Formal verification of piece-wise linear feed-forward neural networks. // 15th International Symposium on Automated Technology for Verification and Analysis. — 2017.
[2] Ekaterina Stroeva Aleksey Tonkikh. Methods for formal verification of artificial neural networks: A review of existing approaches // International Journal of Open Information Technologies. — 2022. — Vol. 10. — P. 21–28.
[3] Dantzig G. Linear programming and extensions // Princeton university press. — 1959.
[4] Dantzig G. Programming in a linear structure. // Technical report, U.S. Air Force Comptroller, USAF, Washington, D.C. — 1948.
[5] Dantzig G. Maximization of a linear function of variables subject to linear inequalities. // Activity Analysis of Production and Allocation,. — 1951. — no. 13. — P. 339–347.
[6] G. Dantzig A. Orden P. Wilfe. The generalized simplex method for minimizing a linear form under linear inequality restraints. — Pacific Journal of Mathematics, 1955. — Vol. 5.
[7] Brandon Paulsen Jingbo Wang Chao Wang. Reludiff: Differential verification of deep neural networks // ICSE '20: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. — 2020. — no. 42. — P. 714–726.
[8] Huang Yanping, Yonglong Cheng, Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. — 2018.
[9] Zhang Dongqing, Yang Jiaolong, Ye Dongqiangzi, Hua Gang. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. — 2018. — 1807.10029.
[10] Niklas Een Niklas Sorensson. An extensible sat-solver. // 6th International Conference on Theory and Applications of Satisfiability Testing, (SAT). Selected Revised Papers. — 2003. — P. 502–518.
[11] Yann LeCun Leon Bottou Yoshua Bengio. Gradientbased learning applied to document recognition // Proceedings of the IEEE. — 1998. — P. 2278–2324.

Р.М. Селевенко - МГУ имени М.В. Ломоносова (email: rselevenko@gmail.com)

Е.Н. Строева - МГУ имени М.В. Ломоносова (email: katestroeva@gmail.com)

Research and development of a formal verification algorithm and quality assessment metrics based on ANN dimensionality reduction methods

Roman Selevenko, Ekaterina Stroeva

***Abstract*—Checking the quality of operation of artificial neural networks (ANN) using formal verification algorithms is the most guaranteed method among others, but there is a problem of computational complexity and very high time costs. The priority direction of research is to reduce the operating time of algorithms, which can be done either by directly changing the formal verification algorithm itself, or by carrying out some actions on the artificial neural network architecture being verified.**

This paper presents a study of methods for formal verification of artificial neural networks based on ANN reduction methods, describes experiments conducted to test the properties of selected neural network architectures, and analyzes these methods.

The Planet algorithm was taken as a formal verification algorithm, the operation of the architecture with the ReLU activation function was checked, so LeNet was chosen, dimensionality reduction methods such as dropout, pruning and quantization were used, the reachability property was checked, and the quality was determined using the RNE metric. The experimental results showed that the use of pruning gives a time gain, but the dropout method turned out to be the most effective, while the use of quantization for the selected Planet algorithm turned out to be impossible.

***Keywords*—formal verification algorithms, machine learning, dropout, pruning**

Список литературы

[1] Ehlers. R. Formal verification of piece-wise linear feed-forward neural networks. // 15th International Symposium on Automated Technology for Verification and Analysis. — 2017.

[2] Ekaterina Stroeva Aleksey Tonkikh. Methods for formal verification of artificial neural networks: A review of existing approaches // International Journal of Open Information Technologies. — 2022. — Vol. 10. — P. 21–28.

[3] Dantzig G. Linear programming and extensions // Princeton university press. — 1959.

[4] Dantzig G. Programming in a linear structure. // Technical report, U.S. Air Force Comptroller, USAF, Washington, D.C. — 1948.

[5] Dantzig G. Maximization of a linear function of variables subject to linear inequalities. // Activity Analysis of Production and Allocation,. — 1951. — no. 13. — P. 339–347.

[6] G. Dantzig A. Orden P. Wilfe. The generalized simplex method for minimizing a linear form under linear inequality restraints. — Pacific Journal of Mathematics, 1955. — Vol. 5.

[7] Brandon Paulsen Jingbo Wang Chao Wang. Reludiff: Differential verification of deep neural networks // ICSE '20: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. — 2020. — no. 42. — P. 714–726.

[8] Huang Yanping, Yonglong Cheng, Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. — 2018.

[9] Zhang Dongqing, Yang Jiaolong, Ye Dongqiangzi, Hua Gang. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. — 2018. — 1807.10029.

[10] Niklas Een Niklas Sorensson. An extensible sat-solver. // 6th International Conference on Theory and Applications of Satisfiability Testing, (SAT). Selected Revised Papers. — 2003. — P. 502–518.

[11] Yann LeCun Leon Bottou Yoshua Bengio. Gradientbased learning applied to document recognition // Proceedings of the IEEE. — 1998. — P. 2278–2324.