Московский государственный университет имени М.В. Ломоносова Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Сидоров Леонид Станиславович

Информационная модель временного ряда как его представление

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
к.ф-м.н., доцент
А.И.Майсурадзе

Москва, 2022

Аннотация

Область графовых моделей в машинном обучении обрела новую жизнь после нескольких крупных публикаций. Спрос на это направление ожидаем, ведь большое количество данных изначально имеет графовую структуру, и эту дополнительную информацию необходимо как-то учитывать.

Данная работа продемонстрирует пример подобного подхода. Её основной фокус направлен на данные электроэнцефалограммы мозга, полученные с помощью нейрокомпьютерного интерфейса. Этот интерфейс снимает электрические импульсы с головы человека при помощи электродов, расположенных по её периметру. Мы предлагаем использовать дополнительную информацию о физическом расположении электродов, чтобы перейти от стандартной задачи анализа временного ряда к задаче анализа во времени графа, который получен на основе физического расположения электродов.

Такой подход позволил нам превзойти по точности распознавания уже существующие алгоритмы на модельном наборе данных и кратно снизить количество обучаемых параметров по сравнению с предыдущим фаворитом.

Содержание

1 Введение

Огромное количество различных структур может быть представлено в виде графа: компьютерные сети, молекулы, схемы финансовых транзакций, планы предприятий и не только. Подобные структуры могут достигать впечатляющих размеров и иметь большую сложность. Поэтому существует необходимость в особом классе алгоритмов, способных напрямую работать с графовыми представлениями объектов.

Имея некоторую постановку задачи, мы можем искать её решение среди классических моделей нейронных сетей, таких как свёрточная нейронная сеть (convolution neural network, CNN), многослойный персептрон (multy layer perceptron, MLP) или рекуррентная нейронная сеть (recurrent neural network, RNN). Но есть несколько фундаментальных различий между представлением данных в виде графа и обычно используемыми структурами, такими как изображения или временные ряды. Из-за них нам необходимо искать новые подходы к представлению данных.

Классические модели глубокого обучения предназначены для решения задач с линейными последовательностями данных: текст, звук или изображения. Все эти типы данных по своей структуре представляют решётки, которая в свою очередь является частным случай графа.

Итак, давайте обсудим основные отличия графически структурированных данных.

- **Размер и форма**. Когда появится новая вершина, мы не сможем поддерживать предыдущую структуру без понимания природы данных. В случае изображений это не является проблемой, потому что там определены простые операции обрезки и масштабирования.

- **Изоморфизм** (инвариантность к перестановкам вершин). На графах в целом нет стандартных шаблонов ориентации, таких как "влево, вправо, сверху или снизу". Это сразу делает невозможным применение наивного подхода, использующего матрицу смежности в качестве входных данных для MLP: перестановка двух строк влечет за собой автоматическую перестановку столбцов. Это приводит к путанице функций.

- **Не евклидова структура**. Требование изоморфизма приводит к отсутствию заданного положения вершины в пространстве, из-за чего невозможно установить систему координат и задать метрику для графа.

Таким образом, использование более общей чем решётка структуры представления данных влечёт за собой необходимость поиска обобщений уже знакомых нам операций, таких как свёртка или агрегация данных.

2 Анализ мозговых импульсов

Многие люди с тяжелыми двигательными нарушениями используют дополнительные коммуникационные устройства. Но те, кто полностью парализован, не могут использовать обычные технологии усиления, которые требуют определенного мышечного контроля. В последние десятилетия активно исследовалась возможность обеспечения дополнительных коммуникаций при помощи сигналов головного мозга, записанных с кожи головы или изнутри черепа человека. Эта технология не требует мышечного контроля, подробный обзор можно найти в [17].

Системы BCI улавливают особенности мозговой активности и преобразуют их в сигналы управления произвольными устройствами. Признаки, используемые в исследованиях на сегодняшний день, включают в себя медленные кортикальные потенциалы, вызванные потенциалы P300, сенсомоторные ритмы, записанные с кожи головы, потенциалы, связанные с событиями, записанные в коре головного мозга, и потенциалы действия нейронов.

Основной задачей в данной работе является распознавание целевых символов, на которых испытуемый фокусирует своё внимание, при этом не подавая никаких других сигналов о своём выборе. Основным источником информации в такой постановке является наличие потенциала P300.

2.1 Вызванный потенциал P300

Волна P300 (P3) — это вызванный потенциал (ВП), возникающий в процессе принятия решений. Считается, что P300 отражает процессы, связанные с оценкой или катего-

ризацией стимулов. Это эндогенный потенциал, так как его возникновение связано не с физическими атрибутами стимулов, а с реакцией человека на них.

P300 используется для построения современных нейро-компьютерных интерфейсов (brain computer interface, BCI), устройств, позволяющих связать мозг человека с компьютером. К примеру, BCI позволяет людям с инвалидностью вести полноценный образ жизни, предоставляя им возможность управлять протезами и коммуницировать с внешним миром без слов.

На электроэнцефалограмме (ЭЭГ) P300 проявляется как положительное отклонение напряжения с задержкой (задержка между стимулом и реакцией) от 250 до 500 мс. В научной литературе часто проводится различие в P300, которое делится по времени: раннее окно (300-400 мс) и позднее окно (380-440 мс).

Сигнал обычно наиболее выражен в теменной доле мозга. Наличие, величина, топография и время появления этого сигнала часто используются в качестве показателей когнитивной функции в процессах принятия решений. В то время как предпосылки проявления P300 все еще остаются туманными, воспроизводимость и повсеместность этого сигнала делают его типичным выбором для психологических тестов как в клинике, так и в лаборатории.

2.2 Предыдущие исследования

Таким образом, алгоритм, решающий задачу распознавания символа по активности головного мозга, так или иначе использует информацию о волне P300. Можно сказать, что распознавание символа эквивалентно распознаванию электрического импульса специального вида и что предлагаемый алгоритм должен решать именно такую задачу. Подобные исследования уже проводились в нескольких работах.

В [1] впервые было предложено использовать свёрточные нейронные сети для распознавания импульса P300. Там также был предложен переход от распознавания целевых символов к распознаванию промежуточных вспышек, формирующих вывод каждого символа. Подробнее этот приём будет рассмотрен в разделе описания наборов данных. В [10] было предложено развитие исходной идеи и использование метода главных компонент в связке с более параметризованными вариациями свёрточных нейронных сетей.

Также стоит отметить пару классических подходов к решению задачи распознавания импульса P300. Эти алгоритмы не полагаются на нейросетевые модели и базируются на стандартных практиках обработки сигналов и классических моделях машинного обучения.

Решение, предложенное в [11], заключается в применении ансамбля, основанного на методе опорных векторов (Support Vector Machine, SVM). Входной сигнал фильтруется и прореживается с целью получения более компактного представления. Обучающая выборка делится на группы, каждая из которых связана с написанием пяти символов. На каждой такой группе обучается метод опорных векторов без ядра, после чего выполняется процедура выбора канала. Алгоритм выбора канала представляет собой рекурсивное исключение каналов на основе критериев качества алгоритма. Распознавание символов достигается за счёт суммирования прогнозов каждого алгоритма из ансамбля.

В [7] используется своя вариация метода градиентного бустинга. Также как и в нашем случае авторы переходят от задачи распознавания целевых символов к задаче распознавания импульса P300.

Однако стоит отметить, что ни одна из предыдущих работ не рассматривала проблему с точки зрения задач на графах. Все рассмотренные подходы опирались на стандартные операции для работы с изображениями, временными рядами и сигналами. В разделе 5.3 мы сравним их с предложенными архитектурами.

3 Графовые нейронные сети

Для обсуждения конкретной графовой архитектуры необходимо ввести базовые концепции для этой области машинного обучения. Для начала определимся с базовыми математическими понятиями.

Определение 1. *Простой граф $G(V, E)$ есть совокупность двух множеств — непустого множества V и множества E неупорядоченных пар различных элементов множества V. Множество V называется множеством вершин, а множество E называется множеством рёбер*

$$G(V, E) = \langle V, E \rangle, \quad V \neq \varnothing, \quad E \subseteq V \times V, \quad \{v, v\} \notin E, \quad v \in V,$$

то есть множество E состоит из 2-элементных подмножеств множества V.

Введенное определение графа включает в себя набор пар вершин, который обобщается на более знакомую для прикладной математики сущность — матрицу.

Определение 2. *Матрица смежности графа G с конечным числом вершин n (пронумерованных числами от 1 до n) — это квадратная целочисленная матрица $\mathbf{A}$ размера $n \times n$, в которой значение элемента $a_{i,j}$ равно числу рёбер из i-й вершины графа в j-ю вершину.*

Иногда, особенно в случае неориентированного графа, петля (ребро из i-й вершины в саму себя) считается за два ребра, то есть значение диагонального элемента $a_{i,i}$ в этом случае равно удвоенному числу петель вокруг i-й вершины.

Матрица смежности простого графа (не содержащего петель и кратных рёбер) является бинарной матрицей и содержит нули на главной диагонали.

Иногда степень связности вершин может варьироваться, например, пропускная способность сети может варьироваться в зависимости от географического расположения сервера. В этом случае каждому ребру может быть присвоен некоторый вес w_{ij}. Эта структура называется *взвешенным графом*. В случае графа с одинаково взвешенными ребрами весами можно просто пренебречь. Их относительные значения для нас не важны, поэтому такой график называется *невзвешенным*.

3.1 Паттерны проектирования графовых нейронных сетей

В этом разделе мы рассмотрим общие паттерны проектирования графовых нейронных сетей (graph neural network, GNN) для базовых постановок задач и типов графов. Как правило, процесс проектирования состоит из четырёх этапов: (1) поиск структуры графа, (2) задание типа графа и масштаба, (3) задание функции потерь и (4) непосредственное построение модели с использованием вычислительных модулей.

Существуют общие практики для каждого этапа этого процесса. В этом разделе мы рассмотрим основную логику, лежащую в основе каждого шага. Более подробный обзор существующих практик можно найти в [22].

Но сначала мы должны выявить структуру графа. Обычно существует два сценария: структурные и неструктурные. В **структурных** сценариях структура графа задаётся явно, например, в задачах для молекул, физических систем, графов знаний и так далее. В **неструктурных** сценариях структура графа заранее неизвестна, то есть мы должны по-

строить его из имеющихся данных, например, построить граф слов для текста или граф сцены для изображения.

После того, как мы получим сам граф, последующие этапы проектирования помогут нам найти оптимальную модель для его анализа.

3.1.1 Задание типа и масштаба

После выявления структуры мы можем перейти к этапу задания типа и масштаба графа. Графы со сложными типами связей могли бы предоставить больше информации об узлах и их соединениях. Графы обычно классифицируются как:

- **Направленные / Неориентированные** графы. Все ребра в ориентированных графах направлены от одной вершины к другой, что предоставляет больше информации, чем неориентированные графы. Каждое ребро в неориентированных графах также можно рассматривать как два направленных ребра.
- **Гомогенные / Гетерогенные** графы. Узлы и ребра в гомогенных графах имеют одинаковые типы, и разные в гетерогенных. Типы узлов и ребер играют важную роль в гетерогенных графах и должны быть приняты во внимание.
- **Статические / Динамические** графы. Когда топология графа меняется со временем, он рассматривается как динамический граф. Информация о времени должна быть тщательно рассмотрена в динамических графах.

Эти категории ортогональны, что означает, что типы графов могут быть объединены. Например, можно иметь дело с динамическим направленным гетерогенным графом. Главная идея состоит в том, чтобы рассмотреть дополнительную информацию, предоставляемую этими типами.

Что касается масштаба, то на сегодняшний день нет четкого критерия классификации для “малых” и “больших” графов. С развитием вычислительных устройств исследователи могут пересматривать свои представления о масштабе.

3.1.2 Дизайн функции потерь

В глубоком обучении самой модели недостаточно для решения задачи. После проектирования сети мы должны рассмотреть процесс обучения, который соответствовал бы

решаемой задаче. Обучение модели опирается на функцию потерь и оптимизатор. Для оптимизатора всегда есть несколько предопределенных шаблонов для каждой ситуации. В то время как проектирование функции потерь часто является нетривиальной задачей, которая учитывает специфику рассматриваемой модели. На этом шаге мы должны спроектировать функцию потерь на основе нашей задачи и особенностей процесса обучения.

Для задач на графах обычно существует три типа постановок:

- **Задачи на уровне вершины** включают в себя классификацию, регрессию и кластеризацию. Классификация вершин делит их на несколько множеств, а регрессия предсказывает непрерывное значение для каждой вершины. Кластеризация направлена на разделение вершин на несколько непересекающихся групп, где похожие вершины должны находиться в одной группе.

- **Задачи на уровне ребер** — это классификация и прогнозирование связей. Прогнозирование связей заключается в предсказании наличия или отсутствия ребра между двумя заданными вершинами.

- **Задачи на уровне графов** включают в себя классификацию, регрессию и сопоставление графов. Для задач этого типа требуется модель для получения представлений на уровне графов.

Мы также можем классифицировать задачи обучения на графах по трем различным типам обучения:

- **Обучение с учителем** предоставляет размеченные данные для обучения.

- **Обучение без учителя** предлагает только не размеченные данные для поиска шаблонов. Кластеризация узлов — это типичная задача обучения без учителя.

- **Смешанный тип обучения** дает малое количество размеченных вершин и большое количество не размеченных вершин для обучения. Большинство задач классификации вершин и ребер относятся к смешанному типу.

С помощью типа задачи и обучения мы можем спроектировать определённую функцию потерь. Например, для задачи классификации смешанного типа на уровне вершин кросс энтропия может использоваться для размеченных вершин в обучающей выборке.

3.1.3 Построение модели из вычислительных модулей

Наконец, мы можем приступить к построению модели с использованием вычислительных модулей. Вот некоторые типы часто используемых модулей:

- **Модуль распространения (Propagation Module)**. Модуль распространения используется для передачи информации между вершинами, чтобы агрегирование могло захватывать пространственную информацию. В оператор свертки и рекуррентный оператор обычно используются для агрегирования информации от соседей, в то время как операция соединения (skip connection) используется для сбора информации из исторических представлений вершин и борьбы с проблемой чрезмерного сглаживания.

- **Модуль выборки (Sampling Module)**. Если графы имеют слишком много вершин и все их сложно хранить в памяти, то требуются модули выборки. Модуль выборки обычно объединяется с модулем распространения.

- **Модуль "пулинга"(Pooling Module)**. Когда нам нужны высокоуровневые представления подграфов, пулинг необходим для извлечения информации из вершин.

Типичная модель GNN обычно строится путем объединения этих компонентов. Эти модули могут повторяться для достижения лучшего эффекта (так, двойное применение пулига позволяет получить более высокоуровневые представления исходного графа).

3.2 Модули графовых нейронных сетей

В этом разделе мы рассмотрим различные вычислительные модули GNN, используемые в нашем исследовании, и их интуицию.

3.2.1 Графовая свёртка

Рассмотрим алгоритмическую основу модуля графовой свёртки, а именно этап передачи сообщений (message passing). На этапе k передачи сообщения текущая вершина получает информацию от вершин, находящихся на расстоянии k от нее. Фиксируя некоторое число T_i для каждой вершины v_i, мы можем построить вычислительный граф на

основе локального соседства, а затем пройти по нему и получить окончательное описание объекта рассматриваемой вершины.

Если этап передачи сообщений на каждом шаге от 1 до T моделируется некоторой нейронной сетью, например, однослойным перцептроном для каждой вершины, то вычислительный граф может быть промоделирован с помощью полносвязной нейронной сети, что означает, что мы работаем с моделью MLP. В будущем мы будем рассматривать этот подход как **базовый**.

Каждый слой нейронной сети, который моделирует этап передачи сообщений, называется слоем передачи сообщений (message passing layer). На первом слое каждая вершина v содержит начальное описание объекта, а на слое k — векторами, полученными после прохода через k слоёв нейронной сети.

Итоговая формула для эмбеддинга $h_v^{(k)}$ вершины v на k-м шаге приведена ниже:

$$h_v^{(k)} = AGG^{(k)}(MSG^{(k)}(h_v^{(k-1)}), \quad \{MSG^{(k)}(h_u^{(k-1)}), \quad \forall u \in \mathcal{N}(v)\}),$$

где $\mathcal{N}(v)$ — множество соседей вершины v.

Заметим, что с определив функций $MSG^{(k)}$ (сообщение) и $AGG^{(k)}$ (агрегация), мы можем получить любой существующий вид графовой свёртки.

Функция сообщения должна быть дифференцируемой на рассматриваемом множестве. Интуиция заключается в том, что каждая вершина создает сообщение, которое отправляется всем соседним вершинам. Примером функции сообщения может служить линейное преобразование $m_v^{(k)} = MSG^{(k)}(h_v^{(k-1)}) = W^{(k)}h_v^{(k-1)}$.

Функция агрегации должна быть дифференцируемой и симметричной по всем аргументам, т.е. она должна быть инвариантной к порядку последовательного агрегирования вершин. $h_v^{(k)} = AGG^{(k)}(m_v^{(k)}, \quad \{m_u^{(k)}, \quad \forall u \in \mathcal{N}(v)\})$ Например, суммирование с весами, среднее арифметическое, максимум для каждого аргумента.

Сообщение с предыдущего шага позволяет нам сохранить накопленную информацию. Обычно этот аргумент представляет собой преобразование вектора из предыдущего слоя. Функция агрегирования может быть нетривиальной, например, можно объединить агрегированные сообщения соседей с сообщением текущей вершины [6].

Помимо этого подхода, называемого *пространственным*, существует также *спектральный*, эксплуатирующий спектральное разложение графов. Однако было доказано, что он

плохо работает с не разреженными графами, состоящими из одной компоненты связности и имеющими множество связей [22].

3.2.2 Графовая агрегация

В области компьютерного зрения за свёрточным слоем обычно следует агрегирующий слой или "пулинг"для получения более общих характеристик. Сложные и крупномасштабные графы обычно содержат богатые иерархические структуры, которые имеют большое значение для задач классификации на уровне вершин и графов. В то время, как операция пулинга для изображений хорошо нам известна и легко рассчитывается, пулинг на графах является нетривиальной задачей, поскольку в этом случае мы отбрасываем вершины графа со сколько угодно сложными представлениями.

Именно поэтому написано большое количество работ, предлагающих свои вариации графового пулинга. В этом разделе мы представляем два вида модулей объединения: модули прямого и иерархического пулинга.

Модули прямого агрегирования изучают представления на уровне графа непосредственно из вершин с различными стратегиями выбора узлов.

- **Простое объединение вершин**. Простые методы объединения вершин используются несколькими моделями. В этих моделях операции максимума, среднего, суммирования или внимания применяются к вершинам для получения глобального графового представления.

- **Set2set**. Set2set [16] предназначен для работы с неупорядоченным множеством и использует метод на основе LSTM для создания представления, инвариантного к порядку, после заранее определенного количества шагов.

- **SortPooling**. SortPooling [21] сначала сортирует вложения вершин в соответствии с их структурными ролями, а затем отсортированные вложения передаются в CNN для получения представления.

Методы, упомянутые ранее, непосредственно изучают представления графа из вершин, они не исследуют иерархические свойства структуры графа. Далее мы поговорим о методах, которые следуют иерархическому шаблону объединения и изучают представления графов по слоям.

- **Graph Coarsening**. Ранние методы обычно основаны на алгоритмах укрупнения графа. Сперва использовались алгоритмы спектральной кластеризации, но они были неэффективны из-за этапа собственной декомпозиции. Graclus [3] обеспечивает более быстрый способ кластеризации узлов и применяется в качестве модуля объединения.

- **ECC**. Свертка с учетом границ Edge-Conditioned Convolution (ECC) [13] проектирует свой модуль пулинга с рекурсивной операцией понижающей дискретизации. Метод понижающей дискретизации основан на разбиении графа на две составляющие по знаку наибольшего собственного вектора лапласиана.

- **DiffPool**. DiffPool [19] использует обучаемый модуль иерархической кластеризации путем обучения матрицы присваивания S_t на каждом слое. Эта матрица маскирует укрупнённую матрицу смежности и обозначает вероятности того, что вершина на слое t может быть сопоставлена вершине на слое $t + 1$.

- **gPool**. gPool [4] использует вектор проекта для получения оценок проекции для каждого узла и выбора узлов с наивысшими оценками k. По сравнению с DiffPool, он использует вектор вместо матрицы на каждом уровне, что снижает сложность хранения. Но процедура проекции не учитывает структуру графа.

4 Предлагаемая модель

Используя полученные знания, зададим архитектуру тестовой модели. Однако сперва нам стоит предложить постановку решаемой задачи и понять, откуда именно в ней появляется граф. Для начала зададим граф G, опишем способ его построения и предложим формализацию исходной задачи распознавания импульса P300 в терминах графовых моделей. После этого мы сможем приступить к обсуждению технических деталей и заданию конкретной архитектуры нейронной сети.

4.1 Построение графа по доступным данным

Все нейроинтерфейсы, для которых проводились эксперименты имели один и тот же тип монтажа, это международный стандарт 10-10 (не считая электроды Nz, F9, F10, FT9,

FT10, A1, A2, TP9, TP10, P9, и P10), состоящий из 64 электродов. Данные были получены с помощью системы BCI2000 [17].

Обозначим за G граф, представляющий рассматриваемый стандарт 10-10. Вершинами этого графа являются электроды, каждая вершина кодируется вектором, который содержит значения потенциалов, замеренных в каждый момент времени. То есть каждой вершине графа сопоставляется временной ряд, записанный за время распознавания импульса P300.

В данной работе мы получим граф G двумя способами, при помощи триангуляции Делоне и метода k ближайших соседей, и сравним результаты (Рис. 1).

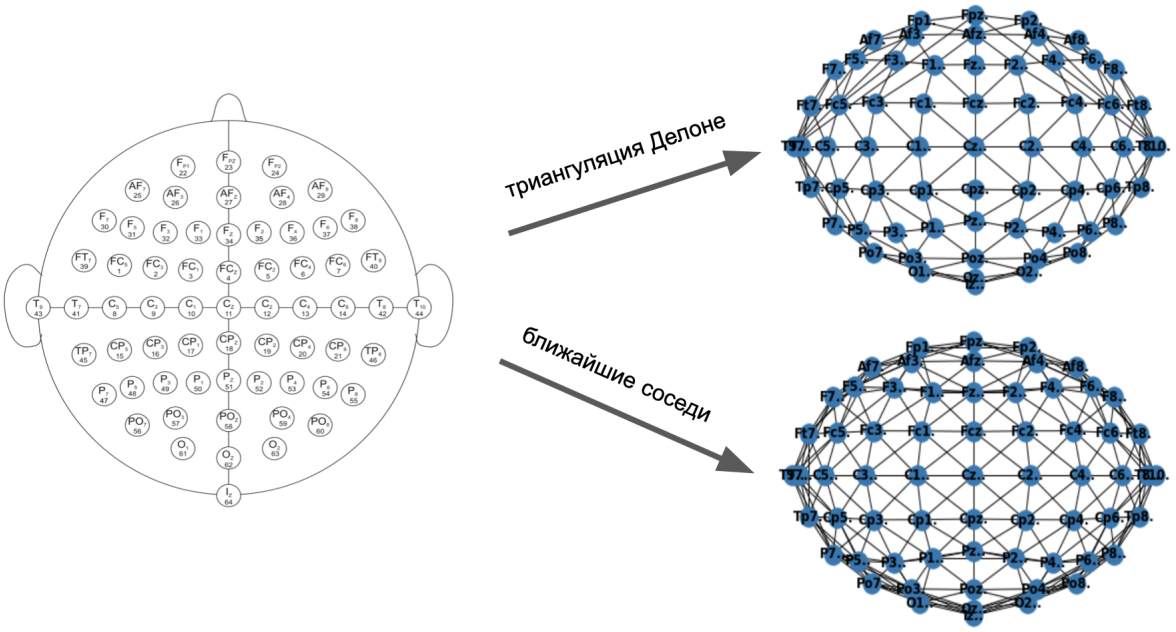

Рис. 1: Схема монтажа электродов 10-10 и предложенные структуры графа.

Для начала стоит отметить, что медицинские данные как правило имеют достаточно подробные спецификации. Например, в нашем случае вместе с данными находились статья, описывающая процедуру их сбора, и файл с координатами каждого электрода. Именно эту пространственную информацию мы и хотим включить в нейросетевую модель, чтобы улучшить качетсво получаемых прогнозов.

Таким образом, по пространственным координатам мы должны получить граф G, характеризующий структуру используемого в экспериментах устройства. Подчеркнём, что граф строится по трёхмерному объекту, принимая во внимание форм-фактор нейроинтерфейса, по двухмерному многообразию.

Мы хотим построить граф по отношению соседства, то есть между двумя вершинами есть ребро тогда и только тогда, когда они являются соседями с точки зрения координат. Теперь нам остаётся лишь формализовать понятие "соседа".

К счастью, человечество уже умеет решать подобную задачу, и нами было рассмотрено два способа решения данной проблемы.

- Первым подходом является **триангуляция Делоне** [2]. Этот алгоритм позволяет построить планарный граф, состоящий только из треугольников. Причём вокруг каждого треугольника можно описать окружность, в которую не будут входить никакие другие точки триангуляция, кроме вершин этого треугольника. То есть полученный граф будет разрежен.

- Второй подход основывается на классическом алгоритме k **ближайших соседей**, который вычисляет расстояния до всех объектов выборки и ищет среди них k ближайших. Поскольку мы работаем с реальным физическим объектом, то было выбрано евклидово расстояние. Экспериментальным путём было подобрано значение параметра k, равное 9.

Теперь, имея граф G, мы можем приступить к формальной постановке задачи распознавания образов.

4.2 Постановка задачи

Имея в распоряжении все ранее введённые термины, переведём изначальную задачу распознавания импульсов P300 с помощью свёрточных нейронных сетей на язык графовых обобщений.

Согласно разделу 3.1, граф G имеет чётко заданную **структуру**, а также является **неориентированным, гомогенным и статичным** во времени.

Имея граф G, каждой вершине которого сопоставлен одномерный временной ряд, мы хотим отнести его к одному из двух классов, где 1 — наличие воздействия, а 0 — его отсутствие. То есть решается **задача на уровне графа с учителем**. Для такой постановки подойдёт классическая функция потерь для многоклассовой классификации, кросс энтропия.

В нашей терминологии объектом выборки является граф. Все объекты имеют структуру G, однако разные временные ряды, соответствующие каждой вершине. Таким образом, как и в задаче для свёрточной сети, каждое наблюдение задаётся матрицей размера $N \times T$, где N — количество каналов/электродов, а T — количество промежутков времени, в которые были осуществлены снятия показаний с прибора.

Теперь мы можем приступить к следующему этапу проектирования, построению модели из вычислительных модулей.

4.3 Предлагаемая архитектура

Основной целью данной работы является сравнение возможностей классических и графовых свёрток, поэтому мы постараемся использовать одну архитектуру для обоих типов моделей. На эту роль прекрасно подходит классическая модель VGG (Visual Geometry Group, группа авторов) [14], предложенная в 2014 году и породившая с тех пор множество переосмыслений и модификаций. Мы сравним классическую модель и её графовое обобщение.

Кроме того, мы применим архитектуру, специально разработанную для анализа временных рядов, использующих графовую структуру STGCN (Spatio-Temporal Graph Convolutional Networks, пространственно-временная графовая нейронная сеть). Она была предложена в 2017 году [20] и является классическим подходом для анализа подобных структур. Таким образом мы оценим потенциал всей графовой парадигмы для задач подобного рода.

4.3.1 Архитектура на основе VGG

Исходная модель состояла из трёх типов модулей: свёртки, пулинга и линейного слоя (умножение на матрицу обучаемых параметров). Как уже обсуждалось выше, первые два типа имеют различные обобщения на графы. Последний же модуль идентичен для обоих подходов и заключается в умножении вектора-объекта на матрицу обучаемых параметров. Такая операция определена лишь для векторов, однако она будет применяться на последнем этапе преобразования объекта, когда и граф в нашей задаче, и матрица в исходной будут приведены к одномерному вектору-эмбедингу.

Также исходная модель могла масштабироваться для задач разной сложности. Логически её можно было разделить на две части: блок-энкодер, состоящий из модулей свёртки и агрегации, он приводит входной тензор к одномерному вектору гораздо меньшей размерности, и блок-предиктор, являющийся многослойным перцептроном, то есть состоящий только из линейных модулей. Оба блока адаптируются под сложность задачи.

Энкодер имеет на входе один модуль свёртки, после чего в нём встречаются смешанные модули, эквивалентные последовательному применению свёртки и пулинга. Количество этих смешанных модулей характеризует сложность энкодера. Также каждую свёртку сопровождает нелинейная функция ReLU [9].

Предиктором является MLP оптимальной глубины для решения в нашем случае задачи классификации на векторах, полученных после применения энкодера.

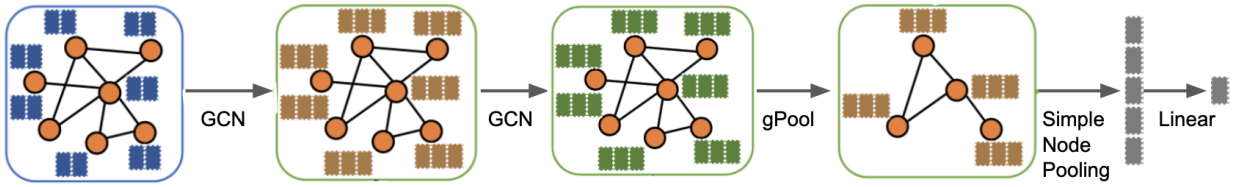

Рис. 2: Архитектура тестовой модели на основе сети VGG.

Заранее подчеркнём, что в текущей постановке задачи объекты выборки имеют относительно небольшую размерность, а именно $N = 64, T = 42$. Более подробное описание входных данных будет приведено в разделе 5.1.

В таком случае итоговая сеть будет относительно неглубокой. Блок-энкодер будет состоять всего из одного смешанного модуля, а блок-предиктор из одного линейного слоя (Рис. 2). На выходе сети мы получим одно число — бинарную метку класса.

4.3.2 Архитектура на основе STGCN

Как и предполагает название, эта архитектура основана на чередовании двух типов свёрток: временных и пространственных. Пространственная свёртка является свёрткой на графе, конкретный вид которой мы обсудим ниже.

Временная свёртка (Temporal Conv) является суперпозицией стандартных одномерных свёрточных слоёв с применением сквозных связей (Рис. 3). Так, в нашей модифика-

ции после применения операции свёртки результат складывается с входными данными и пропускается через функцию нелинейности (ReLU [9]).

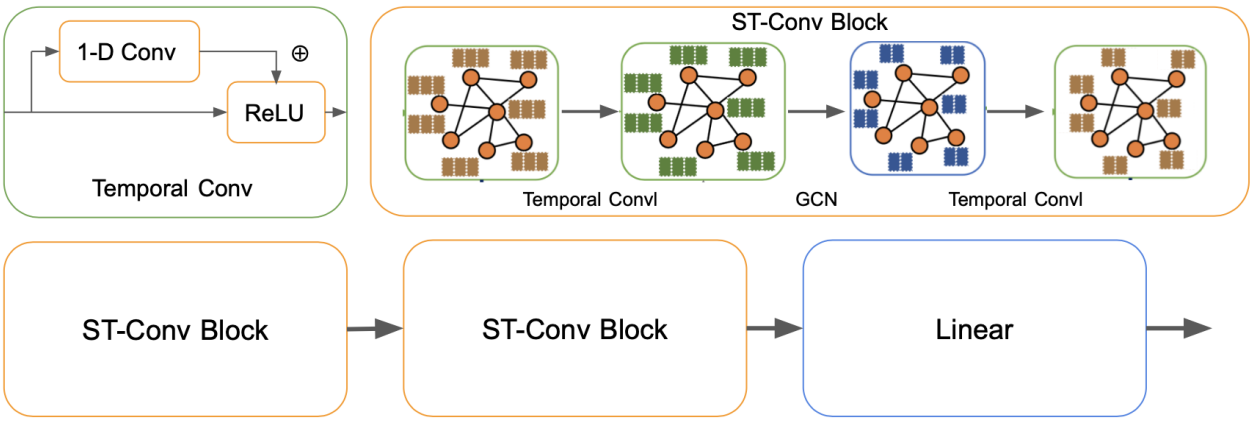

Рис. 3: Архитектура сети STGCN. Сверху изображена структура вычислительных блоков, снизу общий концепт модели.

Сама сеть состоит из последовательного применения пространственно-временных блоков (ST-Conv Block). Каждый такой блок является последовательностью: временная свёртка, пространственная свёртка, временная свёртка (Рис. 3). Таким образом модель учит пространственные и временные фильтры. Графовая свёртка агрегирует информацию с электронов, а временная находит оптимальные фильтры для поиска ключевых признаков. Наша модификация уменьшает размерность по количеству измерений ($T = 42$) в два раза после каждого применения пространственно-временного блока.

Итоговая модель состоит из двух пространственно-временных блоков, которые сопровождает линейный слой (Рис. 3). Он был добавлен для приведения полученных эмбедингов к размерности задачи, в нашем случае это одно число — бинарная метка класса.

Однако мы всё ещё не определили конкретный вид графовых свёртки и пулинга для модели. В следующих разделах мы сравним основные варианты этих модулей и зафиксируем финальный вид графовой модели.

4.4 Выбор свёртки

Нами были протестированы две наиболее популярные вариации пространственной графовой свёртки: GIN (Graph Isomorphism Network) [18] и GraphSAGE (Graph SAmple

and aggreGatE) [6]. Остальные варианты, такие как Graph Attention Networks [15] или LGCN (Learnable Graph Convolutional Network) [5], были отвергнуты из-за недостаточной мощности преобразования или большого количества обучаемых параметров.

GraphSAGE представляет собой парадигму, которая генерирует векторы путем выборки и агрегирования объектов из локальной окрестности вершины (отсюда и название) (Рис. 4):

$$h_{\mathcal{N}(v)}^{(k)} = AGG^{(k)}(\{h_u^{(k-1)}, \quad \forall u \in \mathcal{N}(v)\}),$$
$$h_v^{(k)} = \sigma(W^{(k)} \cdot [h_v^{(k-1)} || h_{\mathcal{N}(v)}^{(k)}]),$$

где $||$ означает конкатенацию тензоров.

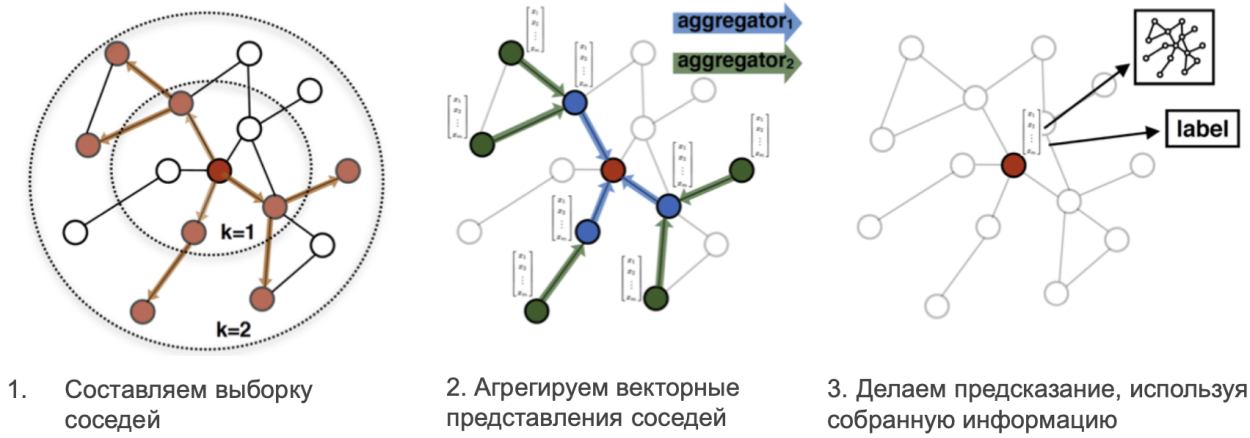

Рис. 4: Принцип работы свёртки GraphSAGE.

Вместо использования полного набора соседей GraphSAGE равномерно выбирает набор соседей фиксированного размера для агрегирования информации. $AGG^{(k)}$ — это функция объединения, GraphSAGE предлагает три агрегатора: агрегатор средних значений, агрегатор LSTM и агрегатор пулинга.

GIN же был предложен позднее и имеет одну важную особенность, он имеет наибольшие возможности в плане решения задач на графах, то есть обладает наибольшей мощностью.

Это значит что, эта свёртка отображает вершины с различными графами вычислений в различные векторы, то есть она обладает наибольшей описательной способностью

$$h_v^{(k)} = MLP^{(k)}((1 + \epsilon^{(k)})h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)})h_u^{(k-1)},$$

она достигается за счёт применении в качестве функции агрегации поэлементного суммирования вместо усреднения и поэлементной максимазиции.

Основным наблюдением в этой задаче является тот факт, что наличие отличий в графах вычислений хотя бы на одном уровне позволяет посредством рекурсии различать их полностью. Так как на каждом слое GNN единственным механизмом, позволяющим нам отличать различные совокупности вершин на одинаковых уровнях графов друг от друга, является функция агрегации, нам необходимо найти такую функцию, которая бы имела способность отображать различные объекты в различные элементы. Такие функции называются инъективными.

Также справедлива следующая теорема. Её подробное доказательство приведено в [18].

Теорема 1. *Фуикция агрегации GIN является ииъективиой, то есть отображеиие различиых вершии происходит в различиые элемеиты миожества эмбеддиигов.*

GIN является не единственной свёрткой, обладающей этим свойством, однако самой "легковесной"из таковых. Поэтому при выборе свёртки я остановился именно на ней. Кроме того, преимущество GraphSAGE в виде использования контекста в графе не релевантно в нашей задаче, фокусирующейся на графах с большим количеством связей.

Отметим, что в отличие от классической операции свёртки в GIN нет "бегающего фильтра и поэтому количество обучаемых параметров падает кратно (мы обучаем двухмерный тензор, а не трёхмерный). Также мы перешли от свёртки по каналам N к свёртке по моментам времени T, что в нашем случае более выгодно с вычислительной точки зрения ($N = 64, T = 42$). Таким образом, конкретно при решении рассматриваемой задачи удаётся добиться экономии обучаемых параметров более чем в 7 раз. В этом случае итоговая нейросетевая модель занимает в 7 раз меньше места в памяти вычислительного устройства и быстрее обучается.

4.5 Выбор агрегации

Наш намеренный уход от тяжёлых операций и блоков с большим количеством параметров, а также отказ от спектрального подхода оставляют нам лишь две операции пулинга для рассмотрения. Это gPool и простое объединение вершин.

Простое объединение вершин не имеет обучаемых параметров и ограничено по характеру взаимодействия, однако позволяет значительно сократить количество параметров, например, при переходе от графа к линейному слою нейронной сети. В дальнейшем мы будем использовать эту технику как вспомогательную.

gPool [4] же отвечает всем нашим требованиям по области применения и сложности итоговой модели, он обладает достаточной гибкостью и не слишком сильно параметризован.

В этом слое используется обучаемый проекционный вектор p. Проецируя все объекты узла в одномерное пространство, мы можем сравнить полученные проекции и выбрать k вершин для следующих слоёв. Поскольку выбор основан на одномерной проекции каждой вершины, связность в новом графе сохраняется. Учитывая вершину i с её вектором признаков x_i, скалярная проекция x_i на p равна $y_i = x_i p/||p||$. Здесь y_i измеряет, сколько информации о вершине i может быть сохранено при проецировании на направление p. Выбирая вершины, мы хотим сохранить как можно больше информации из исходного графа. Чтобы достичь этого, мы выбираем вершины с наибольшими значениями скалярной проекции на p, чтобы сформировать новый граф.

Предположим, что в графе G есть N вершин, каждая из которых содержит C объектов. Граф может быть представлен двумя матрицами: матрицей смежности $A^l \in \mathbb{R}^{N\times N}$ и матрицей признаков $X^l \in \mathbb{R}^{N\times C}$. Каждая ненулевая запись в матрице смежности A представляет ребро между двумя узлами графа. Каждая строка x_i^l в матрице объектов X^l обозначает вектор объектов вершины i на графе. Правило послойного распространения определяется как:

$$\boldsymbol{y} = X^l \boldsymbol{p}^l / ||\boldsymbol{p}^l||, \quad idx = rank(\boldsymbol{y}, k), \quad \tilde{\boldsymbol{y}} = \sigma(\boldsymbol{y}[idx]),$$
$$\tilde{X^l} = X^l[idx, :], \quad A^{l+1} = A^l[idx, idx], \quad X^{l+1} = X^l \odot (\tilde{\boldsymbol{y}}\, \mathbf{1}_C^T),$$

где k — количество узлов, выбранных в новом графе. $rank(\boldsymbol{y}, k)$ — это операция ранжирования узлов, которая возвращает индексы k наибольших значений в $\boldsymbol{y}$. idx, возвращаемый из $rank(\boldsymbol{y}, k)$, содержит индексы узлов, выбранных для нового графа. $A^l[idx, idx]$ и $X^l[idx, :]$ выполняют извлечение строк и/или столбцов для формирования матрицы смежности и матрицы объектов для нового графа. $\boldsymbol{y}[idx]$ извлекает значения $\boldsymbol{y}$. $\mathbf{1}_C^T \in \mathbb{R}^C$ — это вектор размера C, все компоненты которого равны 1, а $\odot$ представляет собой поэлементное умножение матриц.

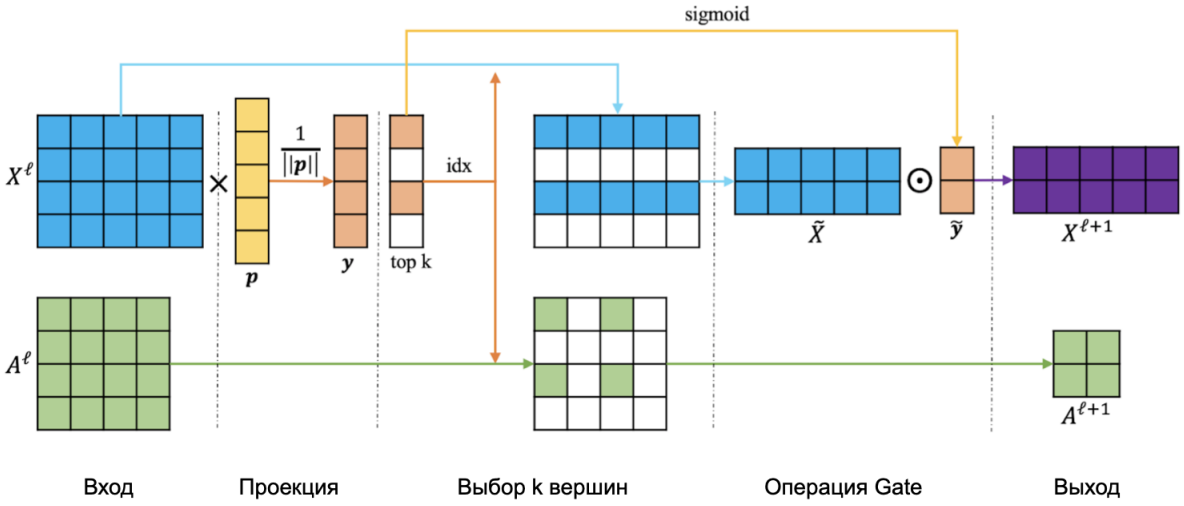

Рис. 5: Принцип работы gPool.

Примечательно, что операция gate делает вектор проекции p обучаемым путем обратного распространения. На Рис. 5 представлена иллюстрация работы данного слоя. По сравнению с обычными операциями пулинга, этот слой использует дополнительные обучающие параметры в проекционном векторе p. Их количество значительно меньше, чем в свёрточном слое, однако согласно [4] такая модификация позволяет добиться значительного улучшения обобщающей способности.

5 Эксперименты

Стоит отметить, что фактически мы будем сравнивать две модели: классическую свёрточную архитектуру, которая применяет одномерную свёртку по каналам и пулинг по промежуткам времени, и графовую модификацию, в которой свёртка воздействует на векторы временных рядов, а пулинг прореживает сам граф (то есть воздействует на каналы). Конечно, такое сравнение далеко от идеального, ведь фактически свёртка и пулинг в нём меняются ролями, однако такой дизайн эксперимента максимально приближает нас к сравнению именно постановок задачи и выявлению вклада графовой архитектуры в финальное решение.

Однако перед описанием вычислительного эксперимента и его результатов нам сначала стоит изучить входные данные для решаемой задачи и редуцировать её до бинар-

ной классификации. После описания исходного набора данных и способа его получения мы сможем настроить нейросетевую модель на его обработку.

Медицинские данные весьма специфичны по сравнению с остальными предметными областями и требуют особого подхода, в дальнейшем мы убедимся в этом и постараемся адаптироваться к их особенностям.

5.1 Описание данных

Данные были получены в соответствии с парадигмой из [12]. Пациенты для сбора данных(A и B) были выбраны случайно из пяти здоровых людей. Для сбора импульсов P300 была задействована парадигма странностей (ориг. oddball paradigm), которую мы опишем ниже.

Графический пользовательский интерфейс для проведения эксперимента представлен на рисунке выше. Он представляет собой матрицу символов 6×6 (Рис. 6). В начале эксперимента каждая из 12 строк и столбцов мигает случайным образом. Мигание длится 100 мс, а интервал между двумя миганиями составляет 75 мс. Испытуемому было предложено сосредоточиться на целевом символе и молча считать количество мигающих повторений строки и столбца, содержащих целевой символ.

Рис. 6: Экран вывода для сбора данных.

Столбцы и строки подсвечивались в случайном порядке блоками по 12 повторений, то есть каждые строка и столбец подсвечивались ровно один раз за блок. Каждая эпоха состоит из 15 таких блоков и соответствует одному целевому символу.

В парадигме странностей символ определяется парой строки и столбца. Предполагается, что символ соответствует пересечению накопления нескольких волн P300. Наи-

лучшее накопление волн P300 для вертикальных мигающих огней определяет столбец желаемого символа. То же справедливо и для горизонтальных мигающих огней и строк.

Набор данных состоит из одного обучающего (85 символов) и одного тестового (100 символов) наборов для каждого из двух подопытных A и B. Все сигналы ЭЭГ были собраны с помощью 64-электродного скальпа, отфильтрованы в диапазоне от 0.1 до 60 Гц и оцифрованы с частотой 240 Гц.

Предполагается, что каждая символьная эпоха содержит два сигнала P300, один для подсветки строки и один для столбца. Следовательно, обучающая выборка содержит $85 \times 2 \times 15$ импульсов.

Однако мы немного изменим изначальную постановку, чтобы реализовать на максимум потенциал глубоких нейросетевых моделей. Так, объединим исходные тестовую и обучающую выборки в хронологическом порядке и выделим для валидации 20% от полученного набора. Глубокие модели модели машинного обучения довольно требовательны к данным, поэтому иметь валидационную выборку больше обучающей довольно непрактично.

Для рассмотренного набора данных существует два типа задач классификации:

- Обнаружение сигнала P300, задача бинарной классификации: один класс представляет сигналы, соответствующие волне P300, второй класс - противоположный. В этом случае создание разметки может быть довольно сложной задачей. Хотя мы знаем, когда стоит ожидать сигнал P300, его появление зависит от испытуемого, причём сам испытуемый тоже не контролирует этот процесс. Производство волны P300 не является феноменом сознания, оно возникает из-за внешних стимулов (мерцание строк и столбцов).

- Распознавание целевых символов. Результаты предыдущей задачи комбинируются для распознавания целевых классов (буквы, символы, действия и т.д.). В то время как разметка первого этапа классификации остается неопределенной, разметка для распознавания символов может быть легко получена, поскольку целевой символ четко задан испытуемому.

Мы остановимся на втором, потому что он предоставляет нам большое количество наблюдений и легко редуцируется до первого. В разделе 5.3 будет проведено сравнение качества для обоих вариантов.

5.2 Вычислительный эксперимент

Для сравнения результатов мы воспользуемся предоставленным разделением выборки на тестовую и обучающую. Так мы сможем сравнивать свои алгоритмы с другими подходами, в том числе и с теми, которые были разработаны за время соответствующего соревнования в 2004 году.

Как уже было отмечено, мы имеем обучающую выборку из 85 символов и тестовую выборку из 100 символов. При переходе ко второму типу классификации из предыдущего раздела мы получаем соответственно обучающую выборку из 15300 наблюдений и тестовую из 18000 наблюдений.

В обучающей выборке очень важен порядок поступления данных, ведь он обусловлен постановкой эксперимента для записи потенциала P300. Поэтому выборки не перемешиваются, иначе градиентные методы не смогут обучаться.

Также стоит отметить, что нейросетевые модели обучаются при помощи метода градиентного спуска, а именно его модификации Adam [8]. Он основан на принципах раздельного обучения градиентов и учёта их накопленного импульса. Важным параметром в этом случае является размер батча, подвыборки данных относительно небольшого размера для обучения на ней стохастических градиентов. Мы опирались на исходную природу данных и положили его равным 180, именно столько наблюдений описывает один целевой символ из исходной задачи классификации.

Исходя из парадигмы странностей, которая была применена для получения данных, на каждые 12 объектов приходится 2 наблюдения положительного класса и 10 негативного. Отсюда в задаче присутствует значимый дисбаланс классов (1 к 6). Кроме того, в каждом батче содержится ровно 30 объектов положительного класса и 150 отрицательного, что гарантированно из постановки эксперимента.

Чтобы бороться с дисбалансом классов, мы стали учитывать градиенты объектов положительного класса с дополнительным весом, то есть увеличили вклад положительных

наблюдений в финальную функцию ошибки. Экспериментальным путём было подобрано значение веса, равное 4.

Теперь, когда была определена процедура обучения нейросетевых моделей, мы можем перейти непосредственно к сравнению этих моделей на предоставленном наборе данных.

5.3 Результаты

Сначала мы сравним качество предложенных алгоритмов для задачи бинарной классификации. Учитывая два вида предложенной графовой структуры и два варианта архитектуры нейросети, получим 4 тестовых графовых алгоритма.

Мы сравним их с существующими классическими подходами для анализа мозговых импульсов, которые были рассмотрены в разделе 2.2, а именно градиентным бустингом и ансамблем на основе метода опорных векторов. Однако стоит отметить, что поскольку ансамбль предсказывает целевые символы напрямую, то мы не сможем включить его в сравнение для бинарной классификации.

Также интересно будет обучить достаточно простой алгоритм логистической регрессии и посмотреть, насколько хорошо он справится с поставленной задачей.

Кроме того, в сравнение войдёт тестовая свёрточная архитектура (1D Convolution NN), сравнив её с аналогичной моделью из раздела 4.3.1, мы сможем оценить обоснованность перехода от стандартной постановки задачи к графовой.

Модель	**Испытуемый A**	**Испытуемый B**
Логистическая регрессия	65.9%	68.4%
Градиентный бустинг	67.6%	69.0%
1D Convolution NN	72.4%	75.7%
Graph Convolution NN (Делоне)	73.5%	77.1%
Graph Convolution NN (Соседи)	73.2%	77.2%
STGCN (Делоне)	**74.3**%	77.6%
STGCN (Соседи)	74.1%	**77.8**%

Таблица 1. Результаты эксперимента для бинарной классификации (метрика accuracy).

Как мы видим из Таблицы 1, все 4 графовых алгоритма превосходят аналоги в качестве классификации. Среди них выделяется сеть STGCN, которая была разработана специально для задач подобного рода.

Также эксперименты показали, что выбор графовой структуры незначительно влияет на качество финальной модели. Алгоритмы из одного семейства, обученные на разных типах графов, имеют почти идентичное качество.

Теперь перейдём к задаче распознавания целевых символов. Как уже упоминалось ранее, она эквивалентна рассмотренной ранее задаче бинарной классификации. Разница заключается лишь в том, что каждый целевой символ записывался 15 раз, то есть человек наблюдал за одной и той же точкой, пока её строка и столбец не загорится по 15 раз. Такое количество повторений необходимо, чтобы повысить надёжность итогового решения.

Таким образом, мы можем наблюдать прогрессию метрики точности с течением времени (Таблица 2). Чем больше раз подсвечивается тот или иной символ, тем более устойчивые оценки мы получим, то есть наблюдаемая метрика монотонно возрастает с ростом числа повторений.

Для перехода к указанной задаче достаточно лишь соотнести предсказания бинарной классификации с данными об очерёдности показываемых строк и столбцов, которые также были нам предоставлены. Имея номера строк и столбцов, на которых испытуемый сфокусировал своё внимание, мы можем без труда восстановить целевые символы.

Модель	**Испытуемый A**			**Испытуемый B**		
	5	10	15	5	10	15
Логистическая регрессия	52%	74%	87%	54%	75%	85%
Градиентный бустинг	53%	76%	89%	56%	79%	87%
Ансамбль SVM	**72%**	83%	97%	75%	91%	**96%**
1D Convolution NN	61%	85%	97%	79%	90%	92%
Graph Convolution NN (Делоне)	63%	87%	98%	80%	91%	92%
Graph Convolution NN (Соседи)	64%	86%	98%	80%	**92%**	92%
STGCN (Делоне)	65%	87%	**99%**	**82%**	91%	92%
STGCN (Соседи)	65%	**88%**	98%	81%	91%	93%

Таблица 2. Результаты эксперимента для распознавания целевых символов (метрика accuracy).

Действительно, с ростом количества повторений точность всех рассмотренных алгоритмов постепенно повышается. Поэтому большинство предложенных решений имеют качество выше 90% после 15 повторений. Однако даже так алгоритмы из графового семейства демонстрируют лучшие точность и скорость распознавания.

Исключением является ансамбль SVM, который демонстрирует лучшее качество для первых пяти повторений у пациента A и всех пятнадцати повторениях у пациента B. Мы видим, что этот алгоритм является самым "быстрым"в терминах распознавания целевых символов для испытуемого A, а также обходит все остальные подходы на испытуемом B. Однако он проигрывает предложенным подходам во всех остальных тестах, в том числе и для пациента B.

Это может быть вызвано тем, что данные пациента B имели недостаточно сложную структуру или были зашумлены из-за проблем с оборудованием. Ведь у этого испытуемого распознавание идёт в среднем быстрее, однако пиковая точность для него меньше. Вероятно, именно поэтому SVM демонстрирует на нём лучший результат.

6 Заключение

В итоге данного исследования была предложена новая постановка классической задачи распознавания импульса p300. Мы применили к исходной задаче графовые алгоритмы и добились лучшего качества для обоих пациентов из выборки и в 7 раз сократили количество обучаемых параметров по сравнению с предыдущим лучшим решением, основанном на стандартной операции свёртки.

Тем самым было доказано, что использование графовой структуры, заложенной в природу данных, может значительно сократить количество параметров модели и улучшить её обобщающую способность.

Таким образом, в данной работе было реализовано следующее:

- исходная задача обработки временных рядов была модифицирована для решения задач на графах;
- было предложено два варианта структурного описания нейроинтерфейса в виде графа;

- была разработана нейросетевая архитектура с гораздо меньшим числом параметров, чем предшествующие "глубокие"аналоги;
- предложенная архитектура имеет лучшую обобщающую способность на изучаемом наборе данных, чем более ранние альтернативы.

В качестве дальнейшего развития данной проблемы возможно провести исследование влияния структуры исходного графа или способа его построения на обобщающую способность итоговой модели. Также представляет интерес применение предложенной формализации к наборам данных другой природы, например, показателям с датчиков на промышленных конвейерах или банковским транзакциям. Для подобных задач рассмотренный подход может быть доработан и расширен.

Список литературы

[1] Hubert Cecotti and Axel Graser. Convolutional neural networks for p300 detection with application to brain-computer interfaces. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):433–445, 2010.

[2] Boris Delaunay et al. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934.

[3] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11):1944–1957, 2007.

[4] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pages 2083–2092. PMLR, 2019.

[5] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1416–1424, 2018.

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[7] Ulrich Hoffmann, Gary Garcia, J-M Vesin, Karin Diserens, and Touradj Ebrahimi. A boosting approach to p300 detection with application to brain-computer interfaces. In *Conference Proceedings. 2nd International IEEE EMBS Conference on Neural Engineering, 2005.*, pages 97–100. IEEE, 2005.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[10] Feng Li, Xiaoyu Li, Fei Wang, Dengyong Zhang, Yi Xia, and Fan He. A novel p300 classification algorithm based on a principal component analysis-convolutional neural network. *Applied sciences*, 10(4):1546, 2020.

[11] Alain Rakotomamonjy and Vincent Guigue. Bci competition iii: dataset ii-ensemble of svms for bci p300 speller. *IEEE transactions on biomedical engineering*, 55(3):1147–1154, 2008.

[12] Gerwin Schalk, Dennis J McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R Wolpaw. Bci2000: a general-purpose brain-computer interface (bci) system. *IEEE Transactions on biomedical engineering*, 51(6):1034–1043, 2004.

[13] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[16] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

[17] Jonathan R Wolpaw, Niels Birbaumer, Dennis J McFarland, Gert Pfurtscheller, and Theresa M Vaughan. Brain–computer interfaces for communication and control. *Clinical neurophysiology*, 113(6):767–791, 2002.

[18] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[19] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

[20] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

[21] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[22] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.