Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Физико-механический институт
Высшая школа теоретической механики и математической физики

Работа допущена к защите
Директор ВШТМиМФ,
д.ф.-м.н., чл.-корр. РАН
_____ А.М. Кривцов
«___» _____ 2023 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Управление робототехническим устройством на основе классификации биоэлектрических сигналов активности мозга

по направлению подготовки
01.03.03 «Механика и математическое моделирование»
направленность
01.03.03_02 Биомеханика и медицинская инженерия

Выполнил
студент гр. 5030103/90201 Д.А. Беркман

Научный руководитель
доцент ВШАиР, к.т.н. Л.А. Станкевич

Консультант
Старший научный
сотрудник ЦНИИ РТК А.М. Корсаков

Санкт-Петербург
2023

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО
Физико-механический институт
Высшая школа теоретической механики и математической физики

УТВЕРЖДАЮ

Директор ВШТМиМФ

А.М. Кривцов

«__»_____20__г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студенту Беркман Даниеле Александровне, гр. 5030103/90201

1. Тема работы: Управление робототехническим устройством на основе классификации биоэлектрических сигналов активности мозга.

2. Срок сдачи студентом законченной работы: 05.06.2023.

3. Исходные данные по работе: научные публикации по теме работы, справочная литература.

4. Содержание работы (перечень подлежащих разработке вопросов): анализ биоэлектрических сигналов активности мозга, написание классификатора, написание блока управления робототехническим устройством, применение биоэлектрических сигналов активности мозга к управлению робототехническим устройством.

5. Перечень графического материала (с указанием обязательных чертежей): не предусмотрено

6. Консультанты по работе: Корсаков Антон Михайлович, старший научный сотрудник ЦНИИ РТК

7. Дата выдачи задания 27.02.2023

Руководитель ВКР: Станкевич Л. А., доцент ВШ автоматизации и робототехники, к.т.н.

(подпись) инициалы, фамилия

Задание принял к исполнению: 27.02.2023

Студент: Беркман Д. А.

(подпись) инициалы, фамилия

РЕФЕРАТ

На 66 с., 20 рисунков, 7 таблиц, 2 приложения

КЛЮЧЕВЫЕ СЛОВА:

ИМК; ЭЭГ; P300; ЗАДАЧА КЛАССИФИКАЦИИ; СПАЙКОВЫЕ НЕЙРОННЫЕ СЕТИ; МОДЕЛЬ НЕЙРОНА ИЖИКЕВИЧА; OMEGABOT;

Тема выпускной квалификационной работы: «Управление робототехническим устройством на основе классификации биоэлектрических сигналов активности мозга».

Данная работа посвящена разработке программы управления робототехническим агентом с использованием интерфейса мозг-компьютер, основанного на классификации сигналов P300. В процессе решения был написан код классификатора на языке программирования Python и код управления роботом Arduino. Был предложен и исследован метод классификации необходимой волны P300 на электроэнцефалографической картине с помощью спайковой нейронной сети. Предложенный метод позволил исключить трудности, связанные с обучением спайковой нейронной сети на нейронах Ижикевича, что является актуальной задачей. Особенностью данного метода является обучение спайкового классификатора с использованием однослойного персептрона с дальнейшей заменой в уже обученной методом обратного распространения ошибки искусственной нейронной сети формальных нейронов на математические модели био-инспинированных нейронов Ижикевича. С целью качественного переноса процессов, протекающих в персептроне на спайковую сеть из моделей нейронов Ижикевича, модель Ижикевича была модифицирована. Точность полученного бинарного классификатора для нахождения необходимой волны P300 составляла 87±5%, а также в большинстве экспериментов полученная сеть совпадала по результатам классификации с искусственной нейронной сетью типа персептрона, на которой она обучалась.

ABSTRACT

66 pages, 20 pictures, 7 tables, 2 appendixes

KEYWORDS:

BCI; EEG; P300; CLASSIFICATION PROBLEM; SPIKING NEURAL NETWORKS; MODEL OF IZHIKEVICH NEURON; OMEGABOT;

The subject of the graduate qualification work is «Control of a robotic device based on the classification of bioelectrical signals of brain activity».

The given work is devoted to the development of a program for controlling a robotic agent using a brain-computer interface based on the P300 signal classification. In the process of solving, the classifier code was written in the Python programming language and the Arduino robot control code. A method for classifying the required P300 wave in an electroencephalographic pattern using a spiked neural network was proposed and investigated. The proposed method made it possible to eliminate the difficulties associated with training a spiking neural network on Izhikevich neurons, which is an urgent task. A feature of this method is the training of a spiking classifier using a single-layer perceptron with further replacement in the artificial neural network of formal neurons already trained by the backpropagation method on mathematical models of Izhikevich bio-inspired neurons. In order to qualitatively transfer the processes occurring in the perceptron to the spiking network of Izhikevich neuron models, the Izhikevich model was modified. The accuracy of the resulting binary classifier for finding the required wave P300 was 87±5%, and in most experiments, the resulting network coincided with the classification results with the artificial neural network of the perceptron type on which it was trained.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Работа посвящена разработке программы управления мобильными объектами с помощью биоэлектрических сигналов активности мозга. В настоящее время всё ещё является актуальной проблема мобильности людей с ограниченными возможностями, например, для парализованных людей, не имеющих повреждений мозга.

Наиболее распространенным решением данной проблемы в настоящее время являются ИМК – интерфейсы мозг-компьютер. Они используют снятые с поверхности головного мозга сигналы электроэнцефалографии (ЭЭГ) и далее, после их обработки, классифицируют их соответствующим заданной задаче образом. Нейроинтерфейсы мозг-компьютер пользуются популярностью в задачах управления ассистивными и вспомогательными устройствами: инвалидными колясками, протезами, ортезами, экзоскелетами, электростимуляторами мышц, орфографическими системами, манипуляторами и т. д. Нейроинтерфейсы чаще стали применяться для нейрореабилитации пациентов, а также для работы с техническими устройствами в экстремальных условиях.

Для классификации снимаемых сигналов необходимо применить к ним сначала предобработку данных и далее алгоритмы машинного обучения. Важным фактором при решении задачи классификации биоэлектрических сигналов является не только точность и быстрота, но также и энергоэффективность алгоритмов. В силу этого в настоящее время все больше внимания уделяется изучению и использованию классификаторов на спайковых нейронных сетях.

Целью данной работы является разработка программы управления робототехническим устройством на основе классификатора биоэлектрических сигналов активности мозга. Для ее достижения решаются следующие задачи:

- аналитический обзор типов сигналов, снимаемых с электроэнцефалографической картины, нейроинтерфейсов, сделанных работ по теме ИМК,
- разработка классификатора на искусственных и спайковых нейронных сетях,
- реализация управления объектом на основе появления целевых значений на экране и синхронизации времени их появления с электроэнцефалографической картиной.

ГЛАВА 1. ОБЗОР ЗАДАЧИ

1.1 Исследования на тему управления с помощью ЭЭГ-сигналов

Разработка вспомогательных средств, улучшающих качество жизни людей, в настоящее время собирает большой интерес научных групп. Чаще всего в реализации управления с помощью ЭЭГ-сигналов используются неинвазивные нейроинтерфейсы мозг-компьютер (ИМК), основанные на обработке и классификации сигналов. Например, авторы работ [11, 22] используют эти интерфейсы для связи с помощью орфографических систем, для движения и увеличения возможностей человека, а также для других применений в жизни как обычного человека, так и людей с ограниченными возможностями.

В последние годы исследований в области использования нейроинтерфейсов в аппаратных реализациях стало больше, они стали популярнее в научной сфере. Интерфейсы компьютер-мозг могут дать возможность людям с ограниченными возможностями общаться с другими людьми. Большинство коммуникационных функций могут быть реализованы посредством компьютера. За последние 10 лет в лабораториях было доказано, что даже страдающие серьёзными когнитивными нарушениями люди могут взаимодействовать с компьютерами, используя только свой мозг.

Технология ИМК может обеспечивать поддержку мобильности либо через управляемые мозгом инвалидные коляски [16], либо посредством мысленного управления мобильным роботом дистанционного присутствия [20], оснащённого датчиками обнаружения препятствий, а также средствами коммуникации для того, чтобы присоединяться удалённо к родственникам, друзьям или коллегам. На данный момент созданы и функционируют коммерческие платформы для такого взаимодействия, например, peoplebot (Mobile Robots Inc., Amherst, США), iRobot (iRobot Corp., Бедфорд, США).

Для управления инвалидным креслом можно использовать данные ЭЭГ, а именно, отслеживать P300 [13] – потенциал, вызванный ожидаемым

нечастым стимулом. Вызвать P300 можно следующим образом: в случайном порядке высвечиваются опции, потенциально интересные субъекту в данной ситуации; опция, при высвечивании которой P300 будет самым сильным, – и есть выбор субъекта. Процесс выбора одной из предоставленных опций занимает небольшое время – порядка 10 с.

Альтернативный вариант – использовать для управления робототехническим устройством движения, доступные субъекту. Например, в исследовании [6] роботизированная коляска с манипулятором (роботизированной рукой) может управляться как с помощью вызванных потенциалов, так и с помощью регистрирования движений шеи и головы.

1.2 Типы нейроинтерфейсов

Нейрокомпьютерные интерфейсы могут быть инвазивными и неинвазивными. Неинвазивными ИМК называются интерфейсы, не предусматривающие вживление электроники. Например, неинвазивными являются интерфейсы, использующие данные электроэнцефалограммы. Инвазивные же ИМК регистрируют активность нейронов через микроэлектроды, вживляемых в мозг. Существуют также полуинвазивные ИМК — такие интерфейсы получают сигнал с коры головного мозга; электроды при этом находятся под черепом, но неглубоко. Как правило, неинвазивные ИМК не обеспечивают такой точности, как инвазивные, но являются более безопасными [19].

1.3 Импульсные нейронные сети и модели нейронов

В настоящее время наиболее распространённым подходом при решении задачи классификации является машинное обучение – направление искусственного интеллекта, связанное с разработкой и построением аналитических моделей, которые автоматически способны обнаруживать в данных скрытые закономерности, а также самостоятельно приобретать

свойства, необходимые для распознавания этих закономерностей. В настоящее время в машинном обучении широко применяются искусственные нейронные сети (ИНС) на формальных моделях нейронов. Эти сети работают с непрерывно меняющимися значениями. ИНС на формальных моделях нейронов второго поколения (глубокое обучение, свёрточные сети) позволили осуществить прорыв во многих областях.

Однако с точки зрения своего практического применения ИНС второго поколения обладают существенным недостатком – такие сети не являются энергоэффективным решением. Занятые разработками в данной области крупнейшие на настоящий момент IT компании мира (OpenAI, Google, Meta) и России (Сбербанк, Yandex, Лаборатория Касперского) вынуждены тратить колоссальные средства на обучение и дальнейшее функционирование таких сетей. Так компания OpenAI в своём проекте ChatGPT использует более 10.000 параллельно работающих видеокарт, стоимость обучения на которых составила порядка 600.000.000 $.

ИНС третьего поколения – спайковые (импульсные) нейронные сети (СНС). Спайковая сеть вместо непрерывно меняющихся во времени значений оперирует дискретными событиями, происходящими в определённые моменты времени. Сеть получает на входы серию импульсов и выдаёт импульсы на выходе, что является намного более энергоэффективным решением по сравнению с ИНС второго поколения в случае аппаратной реализации. Одной из наиболее распространённых моделей спайкового нейрона является модель Ижикевича, которая была использована в рассматриваемой работе. Однако следует отметить, что к настоящему времени нет общепринятых способов решения задачи классификации как СНС вообще, так и модели нейрона Ижекевича в частности, что обуславливает актуальность решаемой в настоящей работе задачи.

Рассмотрим также и другие известные модели нейронов (рис.1.1). Например, наиболее популярная биологически-правдоподобная моделью нейрона Ходжкина-Хаксли [12]. Эта модель была впервые введена в 1952 году

и представляет собой относительно сложную модель нейрона с четырехмерными нелинейными дифференциальными уравнениями, описывающими поведение нейрона с точки зрения переноса ионов в нейрон и из него. Из-за своей биологической правдоподобности модели Ходжкина-Хаксли были очень популярны в нейроморфных реализациях, которые пытаются точно моделировать биологические нейронные системы [10].

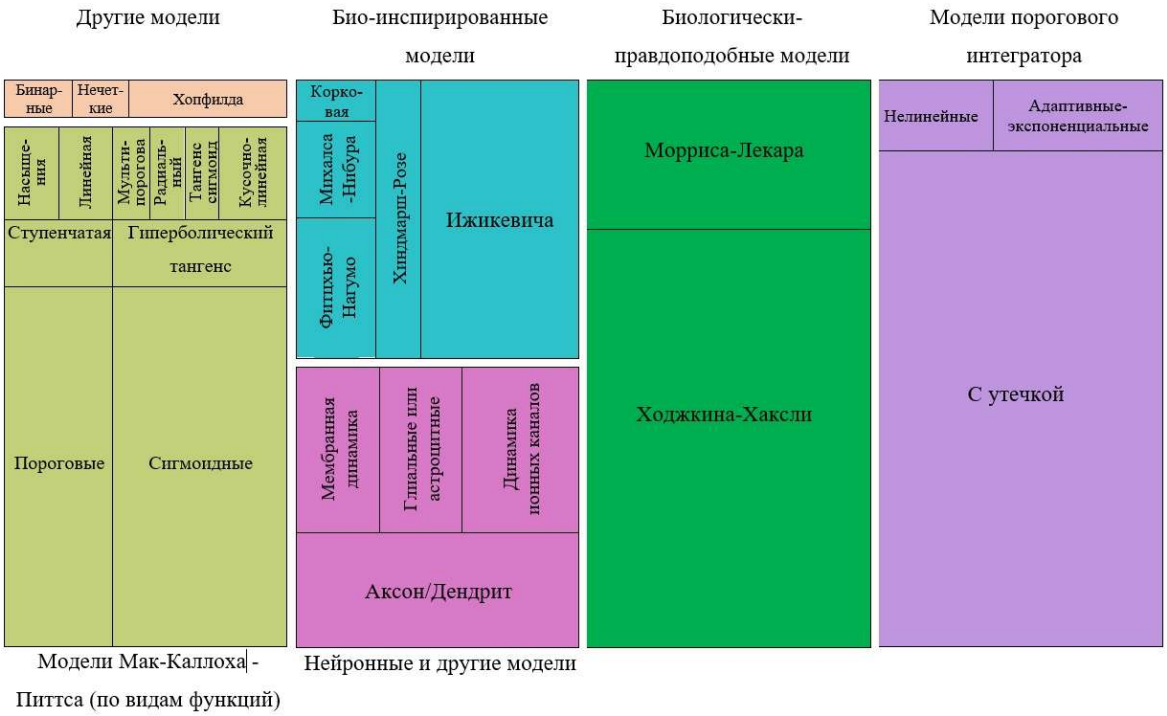


Рис. 1.1: Сравнительное количество моделей нейронов, реализованных аппаратно

Существует множество нейронных моделей, которые являются упрощенными версиями модели Ходжкина-Хаксли, которые были реализованы в аппаратном обеспечении, включая используемую в работе модель нейрона Ижикевича [14]. Эти модели имеют тенденцию быть как более простыми в вычислительном отношении, так и более простыми с точки зрения количества параметров, но они становятся более биологически-вдохновленными (биологически-инспирированными), чем биологически-правдоподобными, потому что они пытаются моделировать поведение, а не подражать физической активности в биологических системах. С точки зрения нейроморфного вычислительного оборудования, более простые вычисления

могут привести к более простым реализациям, которые более эффективны и могут быть реализованы с меньшими затратами.

Модель Ижикевича была разработана для описания поведения нейронов, подобно модели Ходжкина-Хаксли, но со значительно меньшим объемом вычислений [14]. Модель Ижикевича была очень популярна в нейроморфной литературе из-за ее одновременной простоты и способности воспроизводить биологически точное поведение. Другие (биологически-инспирированные, пороговые и т. д.) модели также распространены.

ГЛАВА 2. АНАЛИЗ БИОЭЛЕКТРИЧЕСКИХ СИГНАЛОВ АКТИВНОСТИ МОЗГА

2.1 Введение в типы сигналов

Биоэлектрические сигналы активности мозга записываются в виде информации с помощью электроэнцефалограмм (ЭЭГ). Электрические сигналы активности мозга представляют собой суммарный электрический потенциал от разных нейронов.

Реакции в мозге можно разделить на экзогенные и эндогенные, то есть, вызванные внешними воздействиями и возникающие в связи с принятием решений соответственно. В данной работе будут рассматриваться эндогенные реакции.

Сложным моментом в анализе ЭЭГ является её чувствительность к изменению физического и ментального состояния человека в разное время снятия данных: усталость, тревожность или перевозбуждённость. Также, ЭЭГ-картина сильно чувствительна к внешним раздражителям, шуму и другим экзогенным факторам.

Существуют ИМК, основанные на классификации разных типов сигналов:

- на основе классификации типов ментальной деятельности:

  Они включают в себя воображение моторики и пространственного или вербального воображения. В качестве пространственно-временного ЭЭГ исследуется пространственное распределение амплитуд различных ритмов ЭЭГ по поверхности головы. Их значение и соотношение отражает преобладание конкретных когнитивных процессов.

- на основе компонента воображаемых движений:

  При классификации паттернов ЭЭГ, предназначенных для распознавания компоненты воображаемых движений, используются параметры вызванных потенциалов (ВП) и показатели вызванной

синхронизации/десинхронизации ЭЭГ в частотных диапазонах мю ритма и бета ритма ЭЭГ [18].

Актуальные на данный момент исследования по применению ИМК на основе классификации воображаемых движений нацелены в основном на крупные части тела, например, руки и ноги [3]. На основе этого подхода можно проводить реабилитацию пациентов, перенесших инсульт [17] Она осуществляется посредством формирования новых септических связей во время воображения движений.

- на основе компонента воображаемых движений мелкой моторики:

Такие ИМК являются развитием ИМК на основе компонента воображаемых движений: на практике оказалось, что методы определения паттернов крупных моторных команд не приносят хорошего результата и что существующие на тот момент методы классификации не подходят, следовательно, требуется разработка новых методов классификации. Мотивации к разработке таких ИМК служит их функциональность, такой интерфейс способен предоставить большее количество степеней свободы для систем управления по средствам ИМК.

Также данный метод может найти применение в области восстановления подвижности пальцев рук после инсульта, за счёт формирования новых септических связей во время воображения движений. Основной проблемой метода является низкое пространственное разрешение мозга.

- на основе зрительных вызванных потенциалов (ВЗП):

Мозг реагирует на внешние раздражители. Особенно хорошо проявляется реакция на громкий звук или яркий свет. Изменения в ЭЭГ, возникающие в результате такой реакцией, называют вызванными потенциалами (ВП). Амплитуда ВП обычно на порядок меньше амплитуды фонового шума, поэтому для их обнаружения требуется

накопление и усреднение данных от некоторого количества измерений, проведённых в одинаковых условиях.

Форма возникающих сигналов индивидуальна для каждого испытуемого. Более того, она может даже сильно меняться от раза к разу у одного и того же испытуемого. По этой причине создать шаблонный способ обнаружения ВП невозможно.

Рисунок 2.1 иллюстрирует эту сильную зависимость картины ЭЭГ от испытуемого при одинаковых условиях измерения. Также на этом рисунке показано отличие ЭЭГ одного испытуемого при разных измерениях.

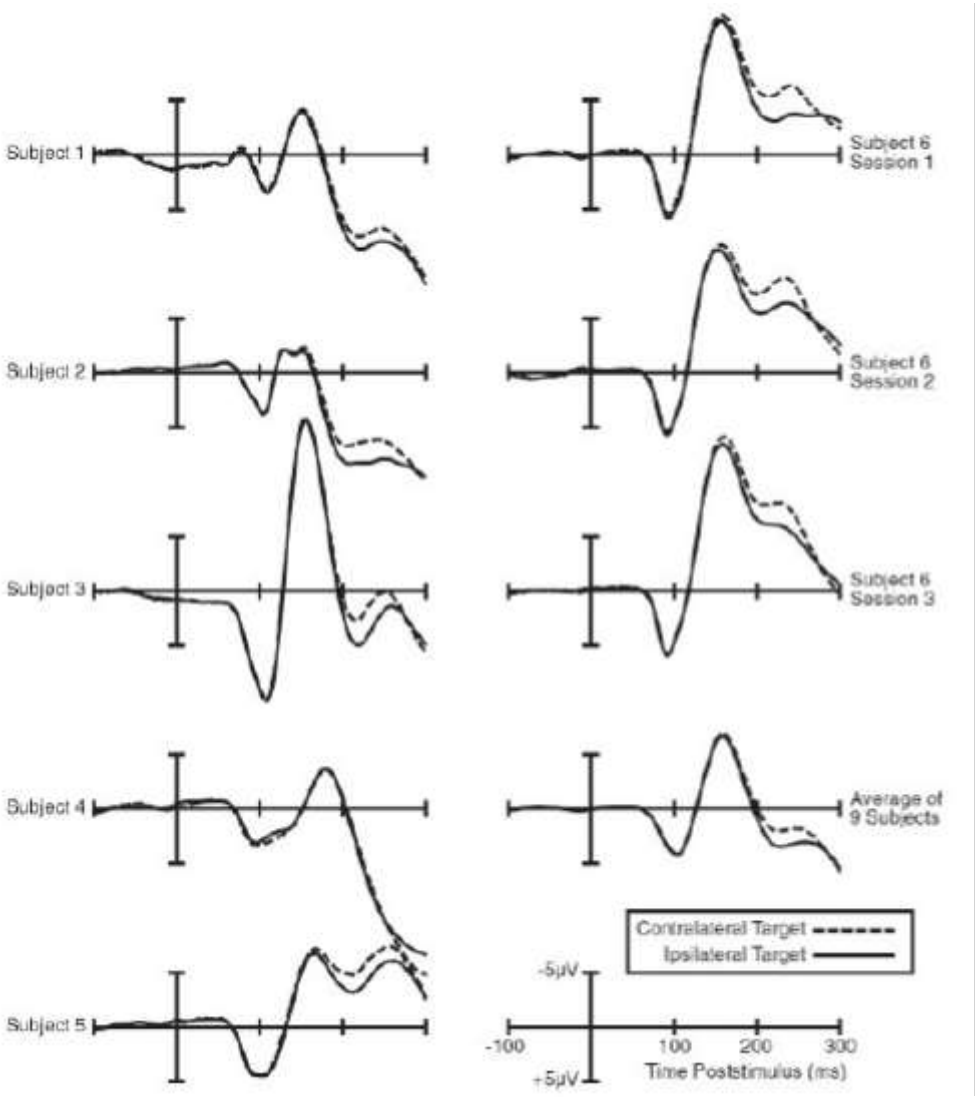


Рис. 2.1. Графики реакции различных испытуемых на стимулы.

- на основе P300:

На рисунке 2.1 изображены разновидности сигналов P300, которые будут рассмотрены в пункте 2.2. Также, видно, что везде наблюдается увеличение амплитуд сигнатуры P300 примерно в тех диапазонах, где он и должен возникать (примерно спустя 300 мс). Подход с нахождением этой волны основан на ЗВП, однако несёт в себе иной смысл.

2.2 Сигналы P300

В задаче рассматривалось решение на основе связанного с событием потенциала P300. В названии этого сигнала содержится некоторая информация о нём: буква «P» означает положительность сигнала, а число 300 говорит о том, что он регистрируется спустя 300мс после предъявления ожидаемого стимула.

Сигнал представляет собой реакцию мозга на предъявление целевого стимула. Скорость управления при таком подходе ограничена, так как нужно не слишком быстро переключать стимулы, чтобы была возможность определить, к какому из них относится зарегистрированный сигнал.

Помимо указанного недостатка, у метода есть и достоинства. Основным из них является отсутствие необходимости тренировки оператора, ведь сигнал возникает естественным образом.

Для обнаружения ВП и потенциала, связанного с событием (ПСС) требуется накопление и усреднение результатов нескольких измерений, проведённых в одной и той же среде. Это подтверждает рисунок 2.2 — на нём хорошо видно, что на одиночных записях ВП неразличим, но после усреднения результатов 64-х измерений шумы, попадая под усреднение, исчезают, а интересующие нас потенциалы становятся различимы.

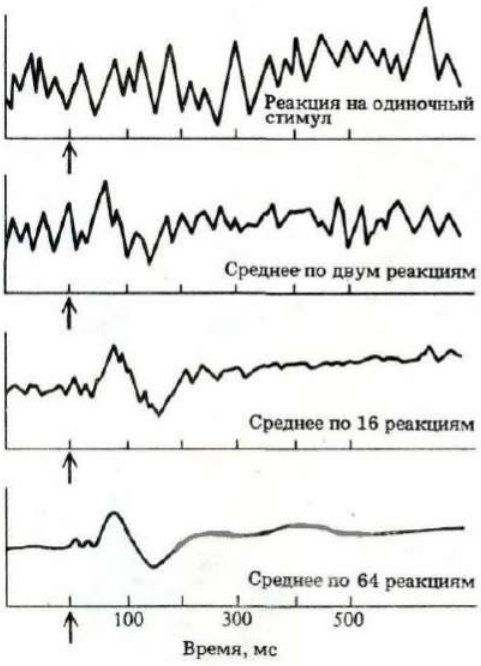


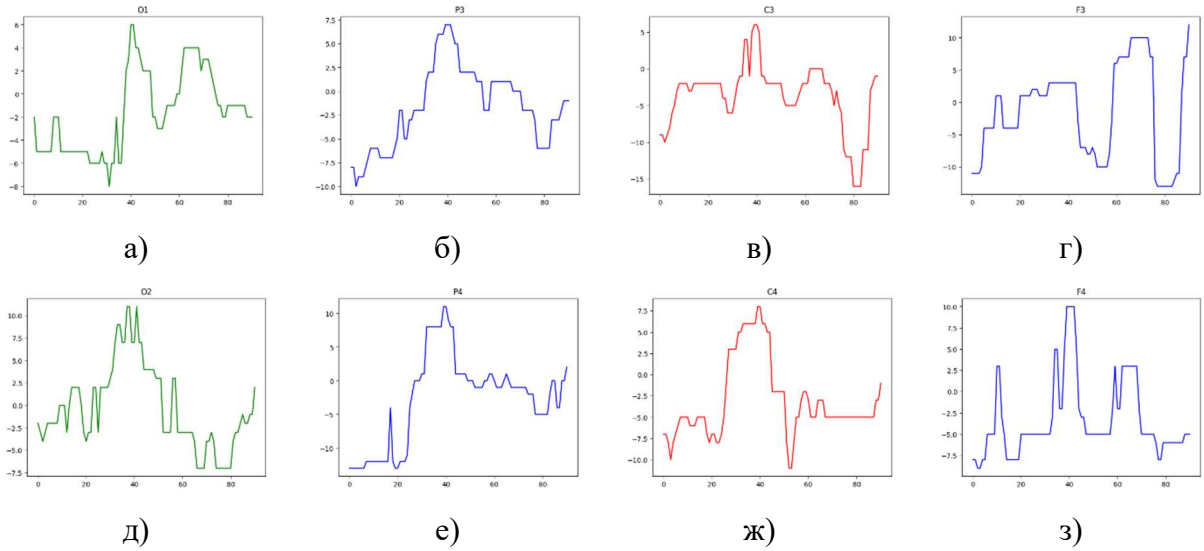
Рис. 2.2. Усреднение сигнала ЭЭГ

Для взятия ЭЭГ-данных использовалось два устройства: NeuroPlay8-cup (рис. 2.3 б)) и Muse (рис. 2.3 а)). При обработке применялись временной высокочастотный фильтр, пространственный фильтр, усреднение попыток, простая фильтрация сигналов в полосе частот 1-40 Гц.

а) б)

Рис. 2.3. Используемые в работе нейроинтерфейсы: Muse (а), NeuroPlay8-cup (б)

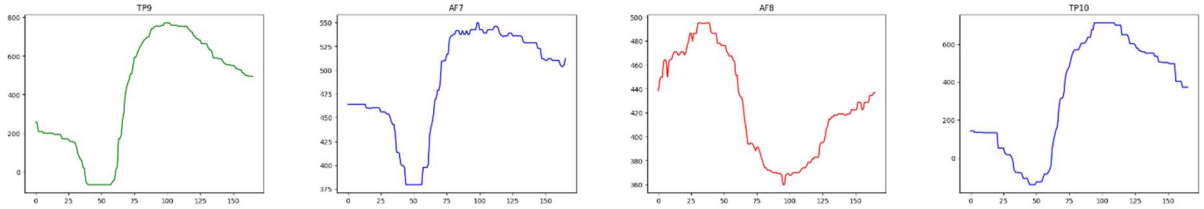
После сложения проб на каждом электроде получились результаты, изображенные на рисунке 2.4 и 2.5.



а) б) в) г)

д) е) ж) з)

Рис. 2.4. Сложение проб на электродах интерфейса NeuroPlay8-cup (а) – O1, б) – P3, в) – C3, г) – F3, л) – O2, е) – P4, ж) – C4, з) – F4)



а) б) в) г)

Рис. 2.5. Сложение проб на электродах интерфейса Muse (а) – TP9, б) – AF7, в) – AF8, г) – TP10)

При просмотре получившихся паттернов на рисунках 2.4. и 2.5. можно сделать вывод, что, хоть и на NeuroPlay8-cup на некоторых электродах можно заметить похожую по характеру волну, на Muse волну P300 обнаружить легче за счет меньшего фонового шума. Исходя из этого, дальнейшие эксперименты будут проводиться на данных, снятых на Muse. Также, надо заметить, что волна, похожая по характеру на P300 находится в теменной зоне головного мозга на обоих приборах для снятия ЭЭГ, как и ожидалось (TP9, TP10 – на Muse, P3, P4 – на NeuroPlay8-cup). Поэтому, дальнейшие эксперименты будут проводиться на данных, снятых на Muse с электродов TP9 и TP10.

ГЛАВА 3. НЕЙРОМОРФНАЯ КЛАССИФИКАЦИЯ ЭЭГ-СИГНАЛОВ

3.1. Вводная информация про спайковые сети

Одним из наиболее распространённых подходов при решении задачи классификации является машинное обучение – направление искусственного интеллекта, связанное с разработкой и построением аналитических моделей, которые способны автоматически обнаруживать в данных скрытые закономерности, а также самостоятельно приобретать свойства, необходимые для распознавания этих закономерностей. В настоящее время в машинном обучении широко применяются искусственные нейронные сети (ИНС) на формальных моделях нейронов. Эти сети обычно работают с непрерывно изменяющимися значениями. Хотя такие ИНС позволили осуществить прорыв во многих областях, в биологическом отношении они не вполне соответствуют структуре реальных нейронов и механизмам обработки информации в человеческом мозге.

Появление спайковых (импульсных) нейронных сетей (СНС) [4,15] позволило уменьшить разрыв между нейронаукой и машинным обучением, поскольку в них используются более реалистичные биоподобные модели нейронов. Спайковая сеть вместо непрерывно меняющихся во времени значений оперирует дискретными событиями, происходящими в определенные моменты времени. Сеть получает на входы серию импульсов и выдаёт импульсы на выходе, что является намного более энергоэффективным решением по сравнению с ИНС второго поколения в случае аппаратной реализации [8]. Одной из наиболее распространённых моделей спайкового нейрона является Модель Ижикевича [14], которая была использована в настоящей работе.

Задачи классификации традиционно решаются с использованием средств с машинным обучением, в том числе, и ИНС [1]. Однако следует отметить, что к настоящему времени нет общепринятых способов решения задачи классификации с использованием как СНС вообще, так и модели

нейрона Ижикевича в частности, что обуславливает актуальность решаемой в настоящей работе задачи.

3.2. Постановка задачи классификации

Пусть задано множество объектов X, множество допустимых классов Y и существует целевая функция $y^* : X \to Y$, значения которой $y_i = y^*(x_i)$ известны только на конечном подмножестве объектов $X^l = \{(x_1, y_1), \dots, (x_l, y_l)\}$, называемом обучающей выборкой. Задача классификации состоит в построении решающей функции (или алгоритма) $a : X \to Y$, которая приближала бы целевую функцию $y^*(x)$, причём не только на объектах обучающей выборки, но и на всём множестве X. Пары «объект-ответ» (x_i, y_i) называются прецендентами или примерами. Объект $x \in X$ описывается с помощью признаков $(f_1(x), \dots, f_n(x))$ – результатов измерения некоторых характеристик объекта [5].

Решающий алгоритм a реализуется с помощью модели, обладающей конечным множеством параметров $\theta = (\theta_1, \dots, \theta_m)$. Процесс построения решающего алгоритма a состоит в подборе значений параметров модели θ по обучающей выборке X^l, что называется обучением.

Задача классификации сводится к нахождению таких значений параметров θ, чтобы алгоритм a давал минимальное значение эмпирического риска на заданной обучающей выборке.

В настоящей работе на первом этапе классификатор на ИНС типа персептрона с одним скрытым слоем необходимо было обучить методом обратного распространения ошибки [2]. На втором этапе нужно решить ту же задачу классификации с использованием СНС, имеющей структуру и веса обученного персептрона, но при замене модели формального нейрона на модель нейрона Ижикевича [15]. После такой замены подобрать оптимальный способ кодирования информации, а также параметры модели нейрона Ижикевича, чтобы обеспечить качественный перенос процессов,

протекающих в персептроне, на СНС с нейронами, построенными на моделях Ижикевича.

3.3. Написание нейроморфного классификатора и проверка на синтетических данных

Для проверки работоспособности сети в начале был проведён эксперимент на синтетических данных. В качестве этих модельных входных данных для СНС были выбраны два линейно разделимых класса, содержащие по два обучающих примера каждый. Тестовая выборка включала в себя по десять примеров из каждого класса.

3.3.1 Обучение сети: персептрон на формальных нейронах

При разработке СНС сначала был реализован персептрон с одним скрытым слоем (см. рис. 3.1). Его обучение производилось с использованием метода обратного распространения ошибки. В результате обучения были сформированы две матрицы весов: W_1[2x4] между входным и скрытым слоями и W_2[4x1] между скрытым и выходным слоями.

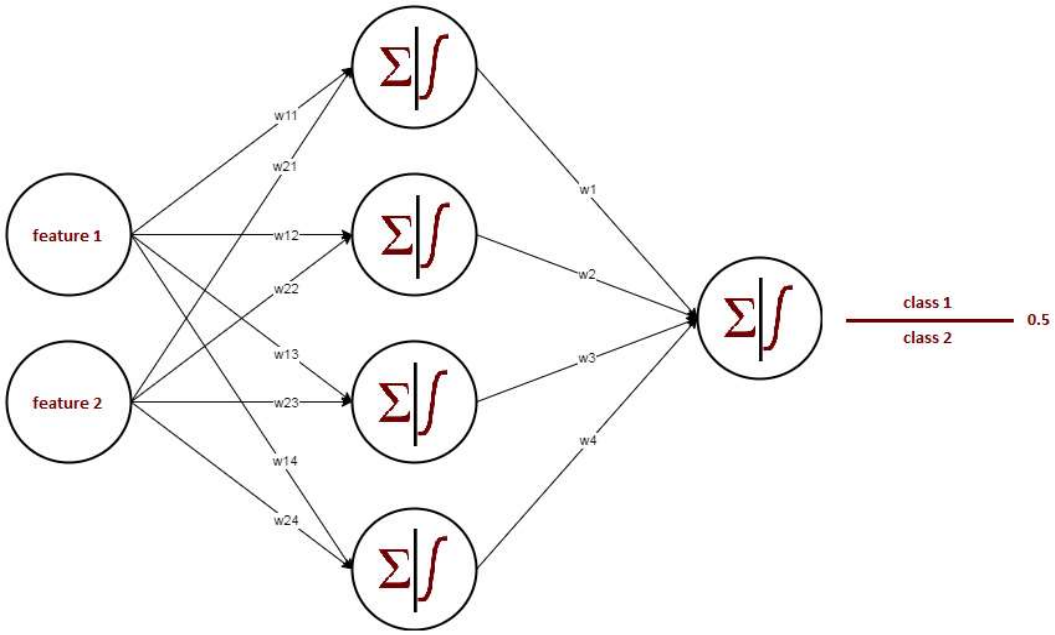


Рис. 3.1. Общая схема обучающего персептрона.

На рис. 3.1 изображена общая схема персептрона с одним скрытым слоем, содержащим четыре нейрона, с двумя входными нейронами и одним выходным. В формальном нейроне происходит суммирование данных с нейронов предыдущего слоя, умноженных на соответствующие им веса.

В формальных нейронах скрытого слоя:

$$q_j = \sum_{i=1}^{2} w_{1_{ij}} p_i \,, \tag{3.1}$$

где $w_{1_{ij}}$, – веса персептрона, соответствующие связям между i-м нейроном входного слоя и j-м нейроном скрытого слоя ($j = \overline{1,4}$);

p_i – данные на i-том входном нейроне;

q_j – данные на j-том скрытом нейроне.

Далее к выходным сигналам q_j применялась сигмоидальная функция активации:

$$\sigma_{hid\quad j} = \frac{1}{1 + e^{-q_j}} \,, \tag{3.2}$$

где σ_{hidden_j} – выход j-го скрытого нейрона.

В формальном нейроне выходного слоя:

$$r = \sum_{j=1}^{4} w_{2_j} \sigma_{hidden_j} \,, \tag{3.3}$$

где w_{2_j}, – веса персептрона, соответствующие связям между нейроном выходного слоя и j-м нейроном скрытого слоя;

r – значение на выходном нейроне.

Далее к выходному сигналу r применялась сигмоидальная функция активации:

$$\sigma_{out} = \frac{1}{1 + e^{-r}} \,, \tag{3.4}$$

где σ_{out} – выход выходного нейрона.

Полученные при обучении веса $w_{1_{ij}}$, w_{2_j} использовались в дальнейшем при построении классификатора на сети из нейронов Ижикевича.

Для решения задачи классификации на персептроне на вход необходимо подать два входных условных признака feature 1 и feature 2. На выходе персептрона классификация происходит в соответствии с правилом:

$$\begin{gathered} if\ out \geq 0.5: class\ 1 \\ else: class\ 2\ , \end{gathered} \tag{3.5}$$

где $out\ = \sigma_{out}$ – численное значение на выходном нейроне.

3.3.2 Модификация модели нейрона Ижикевича

Классическая модель Ижикевича с параметрами, полученными путём подгонки динамики инициации спайка, описывается следующими уравнениями [14]:

$$\begin{cases} v' = 0.04v^2 + 5v + 140 - u + I_{syn} \\ u' = a(bv - u) \\ if\ v \geq v_{peak}: \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \end{cases} \tag{3.6}$$

Параметр a определяет время восстановления переменной u, параметр b – чувствительность переменной восстановления u к подпороговым колебаниям мембранного потенциала v, параметр c – значение сброса мембранного потенциала v после спайка, параметр d – сброс переменной восстановления u [14].

В настоящей работе с целью увеличения времени восстановления мембранного потенциала v после входного воздействия в стандартную формулу классической модели добавлялась мембранная ёмкость C [1]:

$$\begin{cases} v' = \dfrac{0.04v^2 + 5v + 140 - u + I_{syn}}{C} \\ u' = a(bv - u) \\ if\ v \geq v_{peak}: \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \end{cases} \tag{3.7}$$

С целью обеспечения качественного переноса процессов, протекающих в персептроне, на сеть из нейронов Ижикевича в настоящей работе рассматривались два отдельных типа нейрона – возбуждающие (RS) и тормозящие (TS) (см. рис. 3.2, 3.4). Это обусловлено необходимостью

учитывать возможность некоторых входных данных и весов принимать отрицательные значения. Таким образом, модифицированная модель нейрона Ижикевича получила возможность динамически менять свой тип нейрона с возбуждающего на тормозящий или наоборот при изменении знака синаптического тока, подаваемого на него в конкретный момент времени. Необходимо отметить, что возбуждающие и тормозящие нейроны модифицированной модели Ижикевича отличались между собой только применявшимися параметрами модели (рис. 3.3).

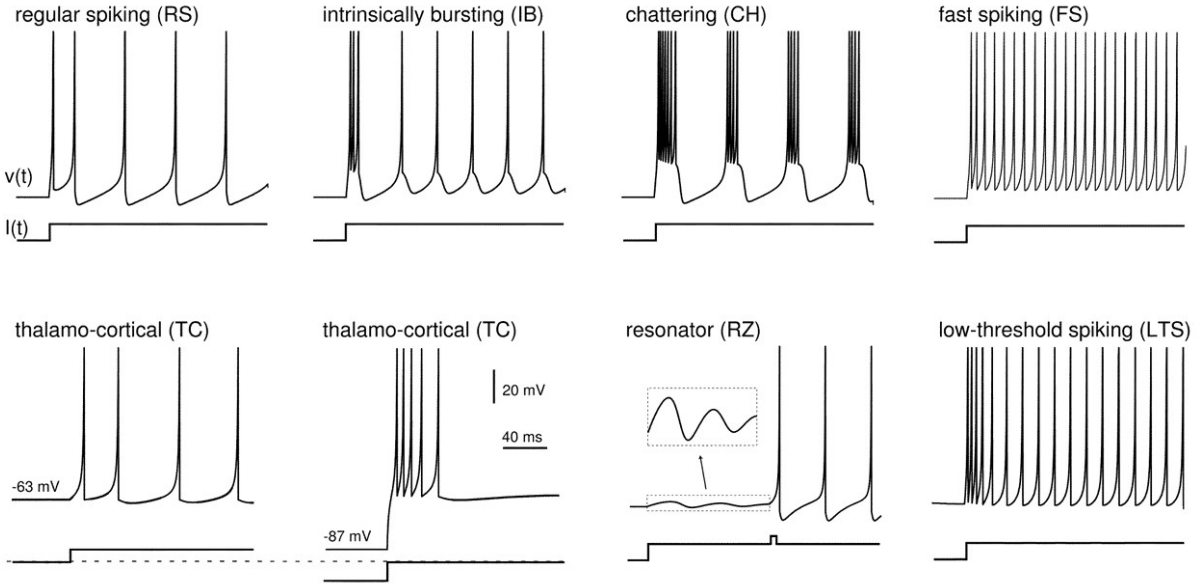


Рис. 3.2. Типы возбуждающих и тормозных нейронов [14]

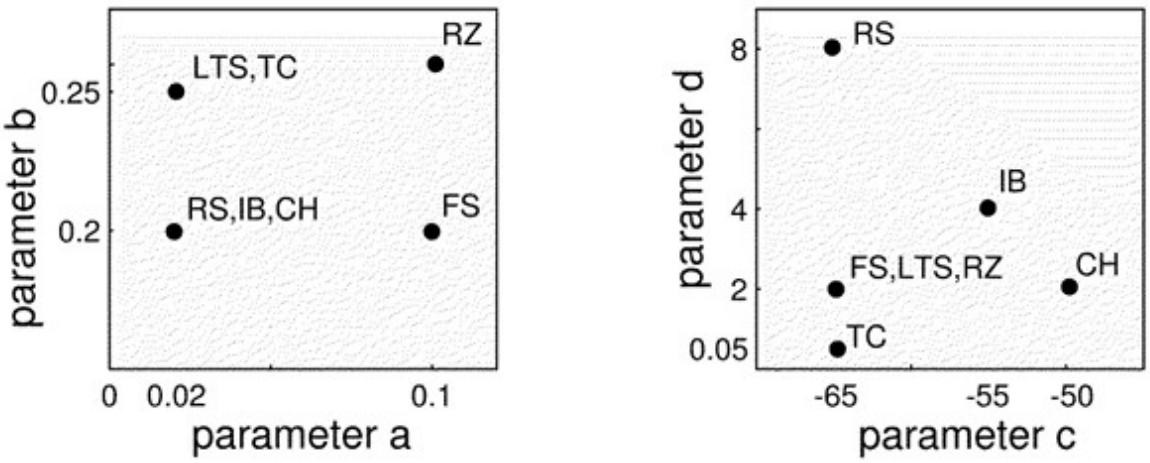


Рис. 3.3. Взаимная зависимость параметров для различных моделей типов нейрона [14].

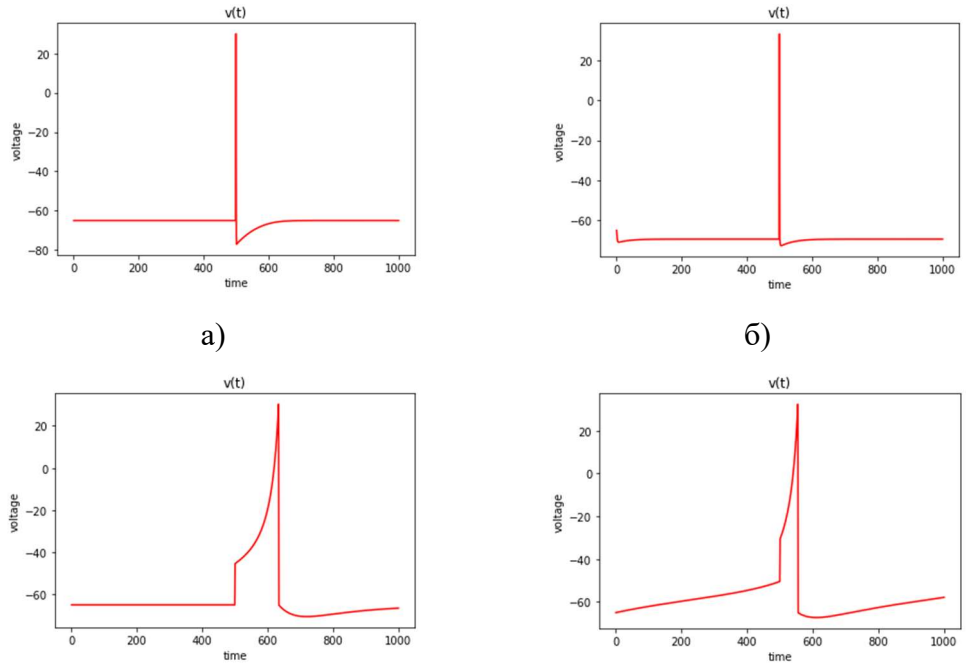


Рис. 3.4. Изменение мембранного потенциала нейрона Ижикевича при единичном воздействии (а – классическая модель, возбуждающий нейрон, б – классическая модель, тормозящий нейрон, в – возбуждающий нейрон с учётом мембранной ёмкости, г – тормозящий нейрон с учётом мембранной ёмкости)

3.3.3 Классификатор на спайковых нейронных сетях

При построении классификатора на СНС структура персептрона сохранялась, но формальные нейроны заменялись на модифицированные нейроны Ижикевича.

На рисунке 3.5 изображена структура с нейронами Ижикевича. СНС, так же, как и персептрон, имеет один скрытый слой, содержащий четыре нейрона, два входных нейрона и один выходной.

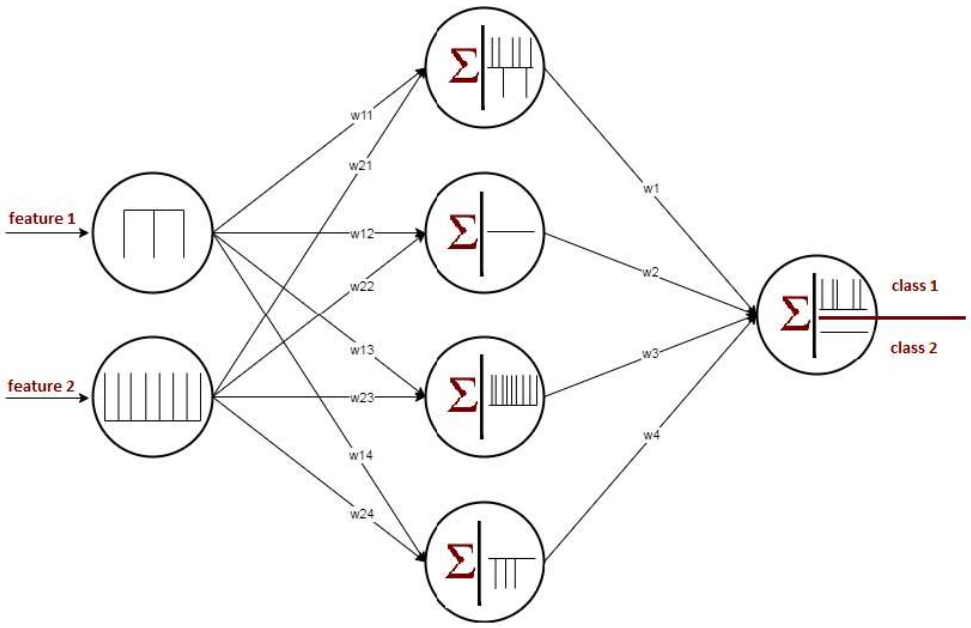


Рис. 3.5. Общая схема сети из модифицированных нейронов Ижикевича на базе персептрона с одним скрытым слоем, четыре нейрона, с двумя входными нейронами и одним выходным.

Для преобразования непрерывных входных данных feature 1 и feature 2 в частоту использовалось соотношение:

$$y_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} * F. \tag{3.8}$$

Отрицательная частота интерпретировалась как последовательность отрицательных входных спайков, поступающих с частотой, равной своему модулю.

Таким образом, в процессе решения задачи классификации на вход каждого j-ого скрытого нейрона подавался синаптический ток:

$$I_{sy\ \ _j}(t) = \sum_{i=1}^{N_{inp}} w_{ij} I_{inp_i}(t)\ , \tag{3.9}$$

где w_{ij} – веса обученного персептрона, соответствующие связям между i-м входным нейроном и j-м скрытым;

N_{inp} – количество входных нейронов;

$I_{inp_i}(t)$ – массив из значений входного тока I_0 при отсутствии входного воздействия, где I_0 определено как:

$$I_0 = -(0.04c^2 + 5c + 140 - bc), \tag{3.10}$$

либо I_1 при его наличии, где I_1 определено как:

$$I_1 = \left(v_{peak} + |c| + I_0\right) * C * red. \quad (3.11)$$

Был введён понижающий коэффициент $red < 1$, применяющийся к значению входного тока, соответствующему срабатыванию нейрона с целью уменьшения вклада отдельных входных воздействий на мембранный потенциал. Этот коэффициент позволяет подавать ток на мембрану маленькими порциями, делая модель менее восприимчивой к единичным входным воздействиям. На рисунке 3.4 проиллюстрирован эффект от появления новых постоянных C и red при единичном входном воздействии.

На выходном нейроне, по аналогии со скрытыми нейронами, на вход подавался синаптический ток:

$$I_{syn}(t) = \sum_{i=1}^{N_h} w_i I_{inp_i}(t)\ , \quad (3.12)$$

где $w_i -$ веса обученного персептрона, соответствующие связям между i-м скрытым нейроном и выходным;

$N_h -$ количество скрытых нейронов;

$I_{inp_i}(t) -$ массив из значений входного тока I_0 при отсутствии входного воздействия (3.10), либо I_1 при его наличии (3.11).

Отрицательному спайку соответствовала ситуация, когда синаптический ток, подаваемый на мембрану нейрона, принимал отрицательное значение [21]. На отрицательный спайк реагировал нейрон с параметрами тормозящего (рис. 3.2, 3.3). Положительному спайку соответствовала ситуация, когда синаптический ток, подаваемый на мембрану нейрона, принимал положительное значение. На положительный спайк реагировал нейрон с параметрами возбуждающего (рис. 3.2, 3.3).

На выходе модели происходила классификация по срабатыванию (class 1) или не срабатыванию (class 2) выходного нейрона.

3.3.4 Экспериментальные исследования

Для проведения эксперимента были взяты параметры модели: $v_{peak} = 30,\ C = 100,\ red = 0.2,\ F = 25$. В качестве возбуждающего нейрона был выбран RS (рис. 3.2, 3.3, 3.4 а), в)) с параметрами $a = 0.02,\ b = 0.2,\ c = -65,\ d = 8$, в качестве тормозного – LTS (рис. 3.2, 3.3, 3.4 б), г)) с параметрами $a = 0.02,\ b = 0.25,\ c = -65,\ d = 2$ [14].

Для проверки корректности работы СНС, для обучающей и тестовой выборки были подобраны синтетические данные.

В таблице 3.1 приведена линейно-разделимая обучающая выборка, состоящая из четырёх примеров, имеющих два признака. Каждый класс обучался на двух примерах. На рисунке 4а точками обозначены обучающие примеры. Цветом и формой определена принадлежность к одному из классов (красные точки – class 1, синие ромбы – class 2). В таблицах и на рисунке 3.6 приняты новые обозначения: class 1 – 1, class 2 – 0.

Таблица 3.1

Обучающая выборка

№	***1***	***2***	***3***	***4***
Answer	1	0	1	0
feature 1	-2	25	-15	17
feature 2	-1	6	-6	4

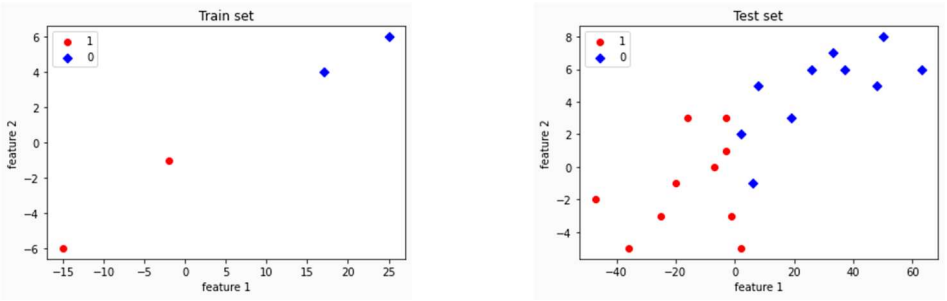
Для тестовой выборки подобраны синтетические данные, которую обученный персептрон распознаёт полностью верно.

В таблице 3.2 приведена линейно-разделимая тестовая выборка, состоящая из двадцати примеров, имеющих два признака. Каждый класс тестировался на десяти примерах. На рисунке 3.6 б) точками обозначены тестовые примеры. Цветом и формой определена принадлежность к одному из классов (красные точки – class 1, синие ромбы – class 2).

Таблица 3.2

Тестовая выборка

№	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***16***	***17***	***18***	***19***	***20***
Answer	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
feature 1	-47	2	-36	19	-25	6	-16	26	-20	33	2	37	-7	48	-3	63	-3	50	-1	8
feature 2	-2	2	-5	3	-3	-1	3	6	-1	7	-5	6	0	5	1	6	3	8	-3	5



а) б)

Рис. 3.6. Обучающая (а) и тестовая (б) выборка.

В таблице 3.3 и 3.4 приведены результаты работы двух классификаторов. В персептроне классификация происходила по формуле (3.5). В СНС с нейронами Ижикевича классификация происходила по критерию срабатывания или не срабатывания выходного нейрона. Если нейрон сработал (не важно, какое количество раз) – пример относился к class 1, если не сработал ни разу – к class 2.

Таблица 3.3

Сравнение результатов классификации с помощью персептрона и классификации с помощью сети с нейронами Ижикевича (худший результат)

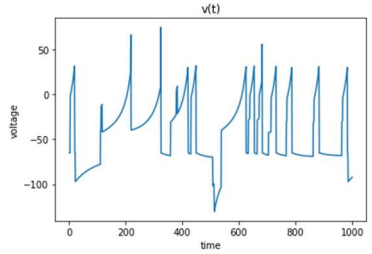
DATA	№	Perseptron		Net with Izhikevich neurons	
		out	Answer	Number of spikes	Answer
Learn	1	0.96158084	1	4	1
	2	0.03713539	0	0	0
	3	0.96167912	1	18	1
	4	0.03713539	0	0	0
Test	1	0.96167912	1	13	1
	2	0.0371504	0	0	0
	3	0.96167912	1	16	1
	4	0.03713539	0	0	0
	5	0.96167912	1	11	1
	6	0.03713687	0	12	1
	7	0.96167912	1	0	0
	8	0.03713539	0	0	0
	9	0.96167912	1	10	1
	10	0.03713539	0	0	0
	11	0.83354584	1	10	1
	12	0.03713539	0	0	0
	13	0.96167911	1	3	1
	14	0.03713539	0	0	0
	15	0.95974236	1	0	0
	16	0.03713539	0	0	0
	17	0.6000625	1	0	0
	18	0.03713539	0	0	0
	19	0.96149093	1	9	1
	20	0.03713539	0	0	0

Так как классификация бинарная, из полученных данных можно вывести оценки точности классификации, считая class 1 за positive, class 2 – за negative. Тогда: TP = 7, TN = 9, FP = 1, FN = 3. Следовательно, доля правильных классификаций:

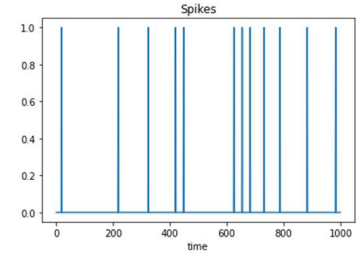
$$Acc = \frac{TP + TN}{TP + TN + FP + FN} = 0{,}8 \qquad (3.13)$$

По полученным результатам, можно заключить, что СНС с модифицированными нейронами Ижикевича с весами персептрона, имеющего аналогичную структуру и обученного с помощью метода обратного распространения ошибки, правильно классифицирует верно минимум 80% тестовой выборки и чаще ошибается в определении class 1.

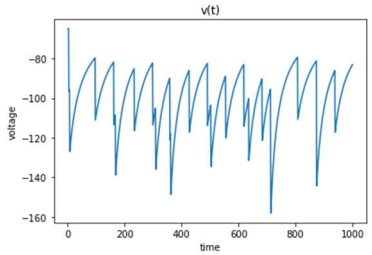
На рисунке 3.7 графически изображены примеры правильного и неправильного срабатывания и не срабатывания выходного нейрона в виде напряжения на мембране (а, в, д, ж) и в виде спайков на нём (б, г, е, з).



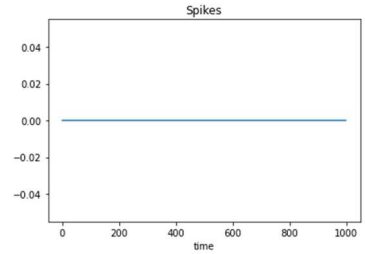
а)



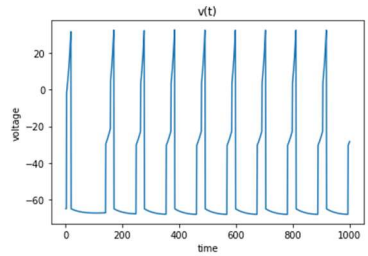
б)



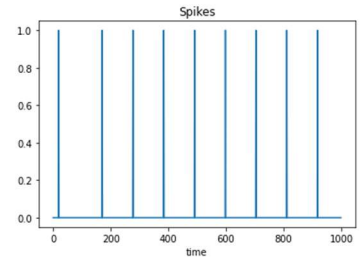
в)



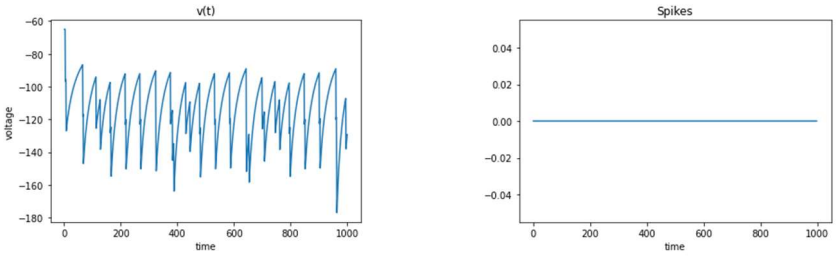
г)



д)



е)



ж) з)

Рис. 3.7. Примеры выходных нейронов. (а, б – пример ложного срабатывания для class 2 (пример FP), выход 6 тестового примера, в, г – пример правильного несрабатывания для class 2 (пример TN), выход 20 тестового примера, д, е – пример правильного срабатывания для class 1 (пример TP), выход 19 тестового примера, ж, з – пример ложного несрабатывания для class 1 (пример FN), выход 15 тестового примера)

Таблица 3.4

Сравнение результатов классификации с помощью персептрона и классификации с помощью сети с нейронами Ижикевича (лучший результат)

DATA	№	Perseptron		Net with Izhikevich neurons	
		out	Answer	Number of spikes	Answer
Learn	1	0.96170205	1	2	1
	2	0.03818424	0	0	0
	3	0.96201049	1	12	1
	4	0.03818424	0	0	0
Test	1	0.96201049	1	36	1
	2	0.03829734	0	0	0
	3	0.96201049	1	28	1
	4	0.03818424	0	0	0
	5	0.96201049	1	19	1
	6	0.03818444	0	0	0
	7	0.96201049	1	13	1
	8	0.03818424	0	0	0
	9	0.96201049	1	16	1
	10	0.03818424	0	0	0
	11	0.7917755	1	0	0
	12	0.03818424	0	0	0
	13	0.96201049	1	6	1
	14	0.03818424	0	0	0
	15	0.96152355	1	3	1
	16	0.03818424	0	0	0
	17	0.93439722	1	1	1
	18	0.03818424	0	0	0
	19	0.96107358	1	1	1
	20	0.03818424	0	0	0

Теперь, при меньшем MSE при обучении персептрона, метрика показывает большую точность классификатора: TP = 9, TN = 10, FP = 0, FN = 1. Следовательно, доля правильных классификаций:

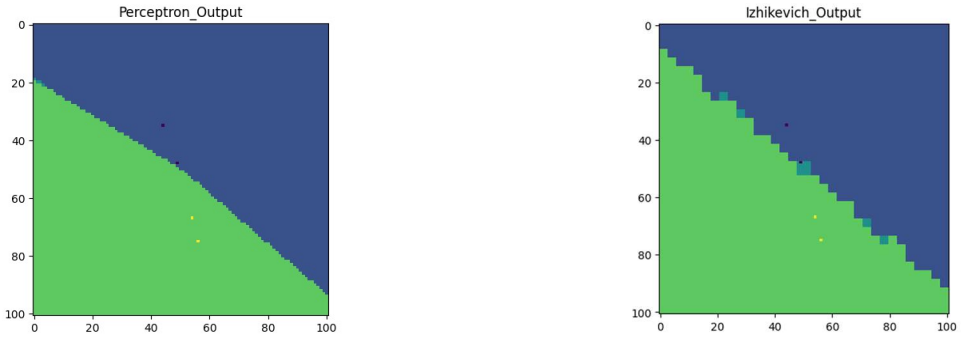
$$Acc = \frac{TP + TN}{TP + TN + FP + FN} = 0{,}95 \qquad (3.14)$$

3.3.5 Выводы о создании нейроморфного классификатора

Эксперимент проводился в предположении, что персептрон идеально классифицирует тестовую выборку. Тестирование показало, что предложенная сеть правильно классифицирует 80-95% тестовой выборки. Учитывая, что в предложенном варианте классификатора внутри сети передаются не непрерывные, а импульсные сигналы, достигается большая энергоэффективность. Данный подход может найти применение в аппаратно-программных решениях, для которых данная характеристика является важной.

Эксперименты, описанные в 3.3.4 были проведёны на синтетических данных. На первом этапе проводилась классификация синтетических данных на двух линейно разделимых классах. В результате получено практически полное соответствие работы персептрона и сети на модели Ижикевича. На втором этапе на персептрон и модель Ижикевича подавались данные из диапазона [-50, 50] с шагом 1. Результат разделения двух классов показан на рис. 3.8.

С одной стороны, качественная оценка результатов на рис. 3.8 показывает отсутствие полной эквивалентности персептрона и модели Ижикевича (хотя результаты близки). С другой стороны, с задачей линейного разделения двух классов сеть, построенная на модели нейрона Ижикевича справилась даже лучше.



а) б)

Рис. 3.8. Разделение классов персептроном (а) и моделью Ижикевича (б)

3.4. Проверка нейроморфного классификатора на реальных данных

Для проведения эксперимента были взяты параметры модели: $v_{peak} = 30,\ C = 100,\ red = 0.2,\ F = 25,\ x_{min} = 0,\ x_{max} = 120.$ В качестве возбуждающего нейрона был выбран RS (рис. 3.2, 3.3, 3.9 а),в)) с параметрами $a = 0.02,\ b = 0.2,\ c = -65,\ d = 8$, в качестве тормозного – LTS (рис. 3.2, 3.3, 3.9 б),г)) с параметрами $a = 0.02,\ b = 0.25,\ c = -65,\ d = 2$.

Для проверки корректности работы СНС, для обучающей и тестовой выборки были выбраны данные с двух электродов (TP9 и TP10) интерфейса muse. В данных для каждого электрода вручную выбраны окна размером 200 мс, разделяющиеся на 10 отрезков, в каждом из которых вычислялось медианное значение, использовавшееся в дальнейшем в качестве признака. Таким образом были вычислены 20 признаков для одного примера. Обучающая выборка состоит из 10 примеров (по 5 примеров на класс), имеющих 20 признаков. На рисунке 3.9 и в таблице 3.5 показаны примеры обучающих примеров.

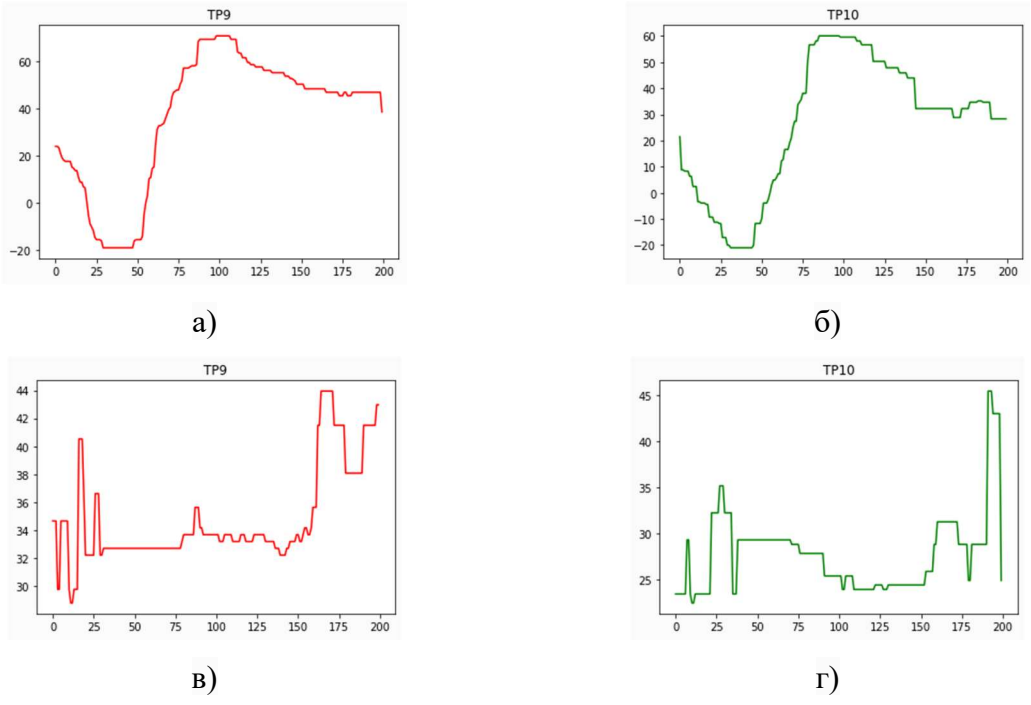


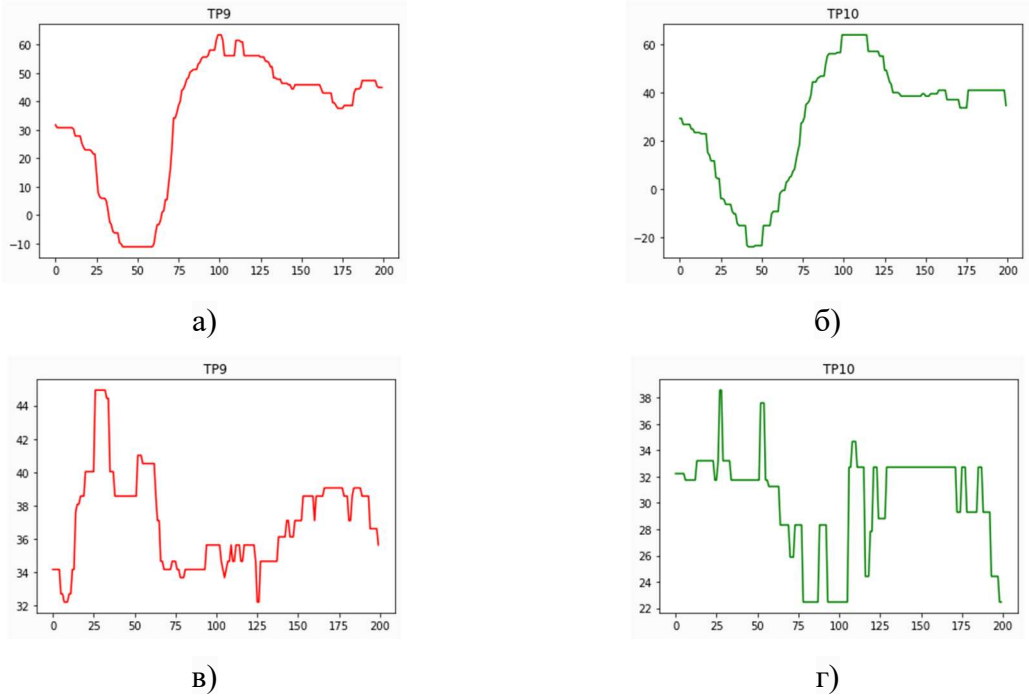
Рис. 3.9. Примеры обучающих примеров. (а, б – обучающий пример №4 (есть сигнал P300), в, г – обучающий пример №5 (нет сигнала P300))

Таблица 3.5

Примеры признаков обучающих примеров

feature №	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
learn 4	14,9659	-0,5859	-3,9144	7,1656	18,8135	26,7294	30,9640	33,3985	34,8959	36,0449
learn 5	33,5937	33,3497	33,1381	33,0385	33,2374	33,2723	33,2694	33,2855	34,2231	34,7949
feature №	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
learn 4	1,9043	-7,7148	-8,0973	0,6043	12,3438	19,8528	23,8596	25,1984	25,9251	26,4893
learn 5	23,9262	27,0512	27,7998	28,0703	27,803	27,2423	26,8208	26,6207	27,0184	27,7614

Персептрон распознаёт тестовую выборку на 80-95%. Сравним с результатами классификации на сети Ижикевича. На рисунке 3.10 и в таблице 3.6 показаны примеры тестовых примеров.



а) б)

в) г)

Рис. 3.10. Примеры тестовых примеров. (а, б – тестовый пример №8 (есть сигнал P300), в, г – тестовый пример №7 (нет сигнала P300))

Таблица 3.6

Примеры признаков тестовых примеров

feature №	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
test 6	28,8087	17,4927	7,9347	10,4066	19,2628	25,7935	29,5899	31,6162	32,6389	33,8989
test 7	34,8147	38,4279	38,7614	38,0005	37,3194	36,9427	36,6317	36,7188	36,9494	37,0459
feature №	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
test 6	23,4619	9,1064	-0,0164	3,2104	12,8369	21,0734	24,5326	26,3611	27,6123	28,9209
test 7	32,3975	32,7758	32,6985	31,5123	29,995	29,7363	29,9874	30,3284	30,4986	30,2393

В таблице 3.7 приведены результаты работы двух классификаторов. В персептроне классификация происходила по формуле (3.5).

В СНС с нейронами Ижикевича классификация происходила по критерию срабатывания или не срабатывания выходного нейрона. Если нейрон сработал (количество раз не важно) – пример относился к классу 1 (есть сигнал P300), если не сработал ни разу – к классу 0 (нет сигнала P300).

Таблица 3.7

Сравнение результатов классификации с помощью персептрона и классификации с помощью сети с нейронами Ижикевича

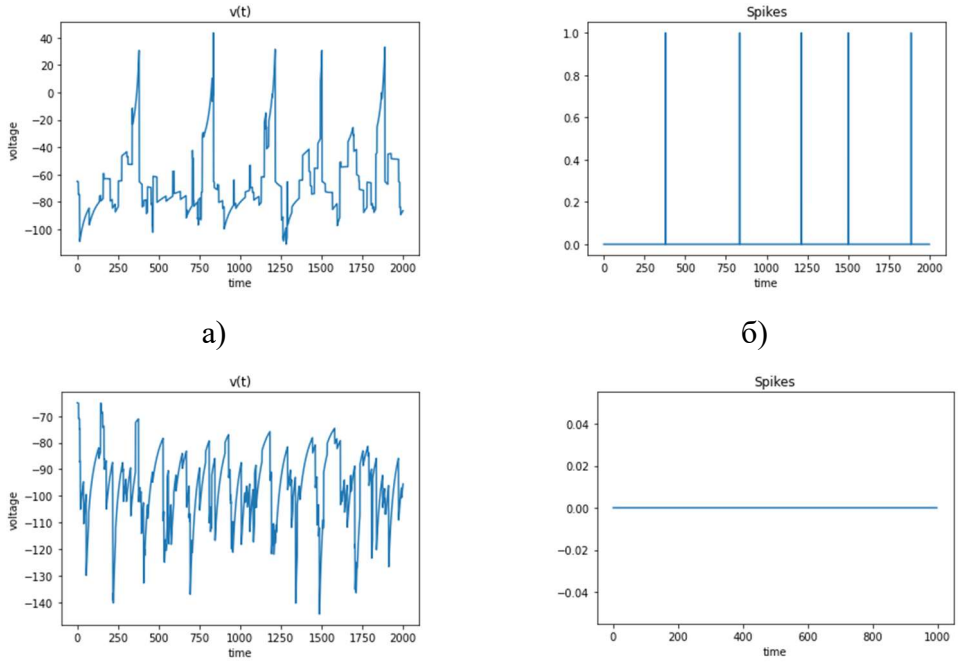
DATA	**№**	**Perseptron**		**Net with Izhikevich neurons**	
		out	Answer	Number of spikes	Answer
Learn	1	0.01939494	0	0	0
	2	0.98083412	1	7	1
	3	0.01939499	0	0	0
	4	0.98083442	1	10	1
	5	0.01939495	0	0	0
	6	0.98083455	1	10	1
	7	0.01939497	0	0	0
	8	0.98083433	1	9	1
	9	0.01939495	0	0	0
	10	0.98083332	1	8	1
Test	1	0.01939494	0	0	0
	2	0.8875294	1	5	1
	3	0.01939494	0	0	0
	4	0.82574809	1	6	1
	5	0.01939495	0	0	0
	6	0.37015266	0	0	0
	7	0.01939507	0	0	0
	8	0.93870441	1	4	1

Так как классификация бинарная, из полученных данных можно вывести оценки точности классификации, считая class 1 за positive, class 2 – за negative. Тогда: TP = 3, TN = 4, FP = 0, FN = 1. Следовательно, доля правильных классификаций:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} = 0,875 \quad (3.15)$$

Проведено сравнение результатов классификации с помощью персептрона и нейроморфной сети с весами, полученными при обучении этого персептрона, которое показало их 100%-е соответствие при данных параметрах модели и весах, полученных при обучении. Таким образом, в результате эксперимента было показано, что нейроморфный классификатор может быть обучен на модели персептрона с аналогичной структурой.

На рисунке 3.11 показаны примеры сигналов на выходах нейроморфного классификатора при правильном (есть волна P300) и неправильном (нет волны P300) срабатывании.



а) б)

в) г)

Рис. 3.11. Примеры выходных нейронов. (а, б – пример правильного срабатывания, выход 2 тестового примера, в, г – пример правильного несрабатывания, выход 5 тестового примера, а,в – графики напряжения на мембране, б,г – визуализация спайков)

3.5 Описание кода классификатора сигналов, снятых на Muse

Коды программ обучения и классификации реализованы с помощью языка Python использовалась библиотека предоставляющая общие математические и числовые операции в виде пре-скомпилированных, быстрых

функций numpy, библиотека для работы с табличными данными pandas и библиотека для визуализации данных matplotlib.pyplot.

В описании будет рассматриваться пример кода для оффлайн классификации полученных и обработанных заранее данных.

Импортируются необходимые библиотеки и переводятся в dataframe табличные ээг-данные в формате excel-файла (см. Приложение 1, блок 1).

В следующем коде (см. Приложение 1, блок 2) происходит построение наглядных графиков различия между отфильтрованными значениями и сырыми (рис. 3.12).

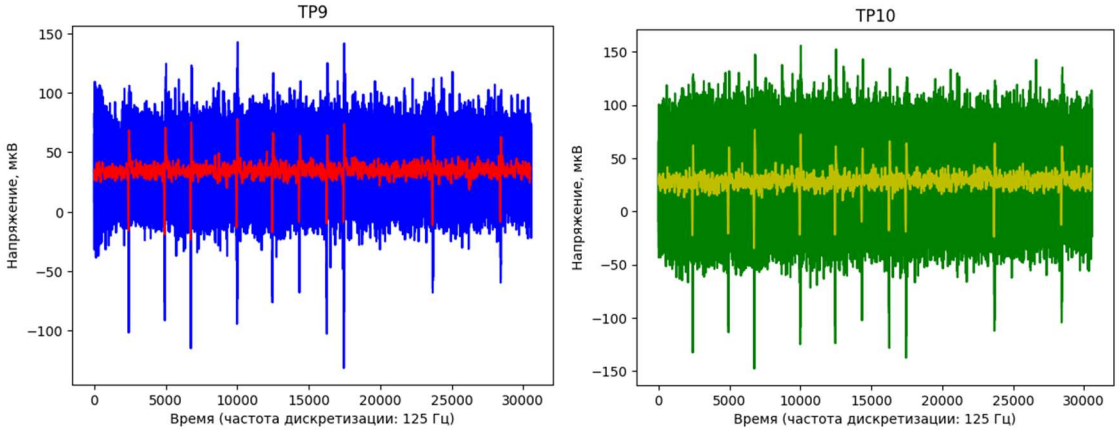


Рис. 3.12. Выведенные графики исходных данных с двух электродов в сравнении с данными с примененным к ним медианным фильтром.

После применения медианного фильтра в данных действительно стали значительно лучше разделяться целевые сигналы P300 и шумовые, что видно на красном и желтом рисунке.

Обучающие нейросеть данные были выделены вручную с помощью разметки временных точек начала P300 на ЭЭГ-картине с электродов. В блоке ниже создаётся матрица обучающих данных и определение принадлежности этих обучающих данных к классам наличия и отсутствия целевого паттерна. Визуализация некоторых обучающих примеров изображена на рисунке 3.9 (см. Приложение 1, блок 3).

Далее пишется алгоритмический вариант персептрона с одним скрытым слоем, в котором 20 входных, 25 скрытых и 1 выходной формальный нейрон.

В каждый входной нейрон направляется один признак в виде числа. Обучение реализуется методом обратного распространения ошибки с продолжительностью в 1000 эпох, со скоростью $\eta = 0{,}1$ (см. Приложение 1, блок 4).

Средняя квадратичная ошибка (MSE) в конце обучения получается в пределах от 0,001 до 0,0001. На рисунке 3.13 изображён прогресс её падения.

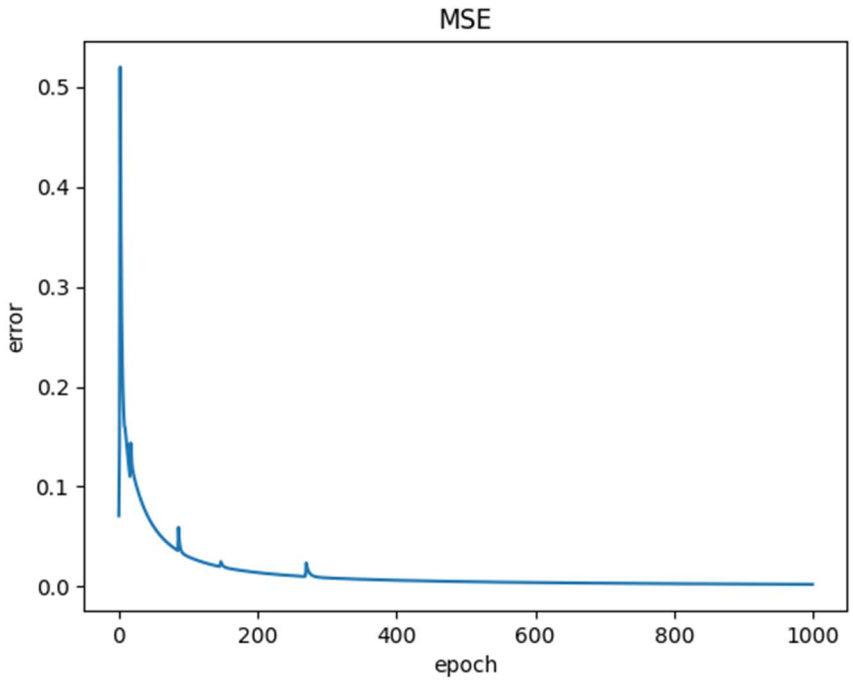


Рис. 3.13. Изменение средней квадратичной ошибки при одном из вариантов обучения данного персептрона в зависимости от эпохи обучения.

Тестовые примеры, в рассматриваемом случае, составляются по аналогии с обучающими. В реальном времени тестовые примеры образовывались каждые 10 отсчетов окнами по 200 мс и образовывалась матрица гораздо большего размера, в которой максимум тестовых примеров оказывались из класса отсутствия сигнала. Для проверки кода классификатора достаточно нескольких. Именно по этой причины тестовые примеры так же размечались вручную, а не как при онлайн-реализации. Визуализация некоторых тестовых примеров изображена на рисунке 3.10 (см. Приложение 1, блок 5).

В блок написания классификатора входит перевод численных данных в частоты спайков во входные нейроны и написание сети из моделей нейронов Ижикевича на базе обученного персептрона.

В начале реализуется с помощью частотного кодирования по формуле (3.8) перевод в спайки (см. Приложение 1, блок 6).

Далее реализуется замена формальных нейронов на модели нейронов Ижикевича, а также написание энергоэффективного классификатора. (см. Приложение 1, блок 7)

Блок работы классификатора и управления будет описан далее, в пункте 4.1.

3.6. Выводы по главе

В работе представлена спайковая нейронная сеть для решения задачи классификации, построенная на базе предварительно обученного персептрона. Сеть имеет структуру, аналогичную персептрону, но формальные нейроны в ней заменены на нейроны Ижикевича. Это позволило улучшить обучение спайковой нейронной сети.

С целью обеспечения качественного переноса процессов, протекающих в персептроне, на сеть из нейронов Ижикевича была проведена модификация стандартной спайковой модели нейрона Ижикевича. В модель дополнительно введена мембранная емкость и понижающий коэффициент для входного тока, соответствующего срабатыванию нейрона. Кроме того, использованы два отдельных типа нейрона – возбуждающие и тормозящие. Таким образом, в модифицированной модели нейрона Ижикевича стало возможным динамически менять свой тип нейрона с возбуждающего на тормозящий и наоборот при изменении знака синаптического тока, подаваемого на него в конкретный момент времени.

В качестве перспективного направления исследований рассматривается проверка применимости предложенного метода на других реальных данных, а также использование других спайковых нейронных сетей.

ГЛАВА 4. УПРАВЛЕНИЕ РОБОТОТЕХНИЧЕСКИМ УСТРОЙСТВОМ

4.1 Управление на основе целевых стимулов

Блок управления реализуется при подключении кода на python к serial-порту Bluetooth-модуля (см. Приложение 1, блок 8). С помощью синхронизации временных точек начала целевых стимулов, показываемых на отдельном экране в произвольном порядке, и временных точек найденного целевого сигнала P300 определялось, какой именно стимул ожидал испытуемый. Для определения класса движения был создан массив времен возникновения целевых движений. Была проведена синхронизация путём постановки меток на ЭЭГ записи. Таким образом получилась классификация четырёх движений на основе поиска паттернов P300 спустя 100-600 мс после возникновения ожидаемой картинки движения, что является стандартным диапазоном. Пример ожидаемых картинок движения изображен на рисунке 4.1.

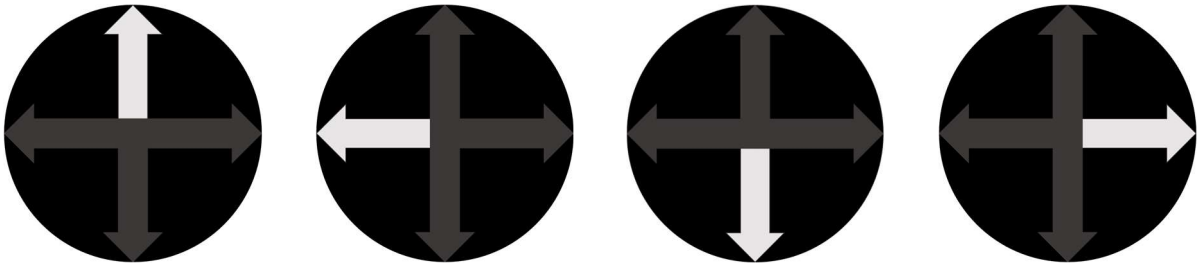


Рис. 4.1. Ожидаемые стимулы

Далее будет описан код синхронизации появления стрелочек на экране, на который смотрит испытуемый при ожидании одной из них. При появлении ожидаемого типа стрелки – на ЭЭГ-картине должна возникнуть волна P300 на выбранных исследуемых электродах спустя определённое время. Если волна P300 в данном временном участке была выявлена – движению присваивается класс, соответствующий предшествующей сигналу стрелке.

Временные метки начала возникновения каждой из стрелочек реализуются в коде приложения 1, блока 9. Работа классификатора, разделение на 4 класса движения реализуются в коде приложения 1, блока 10.

4.2 Реализация кода управления

Для проверки работоспособности программы необходимо проверить её на робототехническом устройстве. В данной работе им был выбран omegabot в связи с его доступностью. Ниже реализован простой код прошивки платформы Arduino, на которой работает omegabot.

В приложении 2, блоке 1 подключаются колеса. В приложении 2, блоке 2 вводится переменная указания класса движения, который передаётся через serial-port на Bluetooth. В приложении 2, блоке 3 ставятся ограничение работы моторов по скорости. Далее в приложении 2, блоке 4 проводится инициация последовательной связи на COM порту со скоростью 9600 бод/с. В В приложении 2, блоке 5 находится повторяющиеся действия и управление роботом по соответствующим классам.

ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе была решена задача управления устройством с помощью биоэлектрических сигналов активности мозга. Для выполнения решения была выбрана одна из самых надёжных реализаций технологий интерфейсов мозг-компьютер – управление с помощью предъявления символов на экране и выявления наличия необходимого связанного с событием потенциала P300. Были проведены необходимые эксперименты и анализ полученных данных.

Также, в процессе выполнения работы был удачно исследован, разработан и применен новый метод обучения классификатора на спайковых нейронных сетях с помощью замены стандартных формальных нейронов обученного методом обратного распространения ошибки персептрона на модифицированные математические модели возбуждающих и тормозных био-инспрированных нейронов Ижикевича. С помощью исследованного и проверенного в работе метода станет легче обучать спайковые нейронные сети. Также, классификация за счёт дискретной (событийной) передачи информации станет энергоэффективной, что немаловажно в аппаратных реализациях.

Поставленные к работе цели были выполнены успешно. В продолжении данной работы можно реализовать алгоритм передачи информации, полученной из мозговой активности (возможно, на основе других методов), на управляемый объект в настоящем времени. Также, в целях развития нового классификатора планируются проверки и разработки большей по объему сети с использованием других биоподобных моделей нейронов.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Тадеусевич Р., пер. с польск. Рудинского И.Д. Основы нейрокибернетики, Москва, Горячая линия – Телеком, 2016. – 372 с.
2. Хайкин С. Нейронные сети: Полный курс. Пер. с англ. Куссуль Н. Н., Шелестова А. Ю., 2-е изд., испр. – М.: Издательский дом Вильямс, 2008. – 1103 с.
3. Щетинин Е.Ю. Математические методы машинного обучения: учеб. Пособие. – М.: ФГБОУ ВО «МГТУ «СТАНКИН», 2016. – 168 с.
4. Евграфов В.А, Ильюшин Е.А. Спайковые нейронные сети // IInternational Journal of Open Information Technologies – 2021. – 9. N. 7. – P. 2307 – 8162.
5. Котов С.В., Турбина Л.Г., Бобров П.Д., Фролов А.А., Павлова О.Г., Курганская М.Е., Бирюкова Е.В. Применение комплекса «интерфейс “мозг-компьютер” и экзоскелет» и техники воображения движения для реабилитации после инсульта // Альманах клинической медицины. – 2015. – № 39. – С. 15 – 21.
6. Achic F., Montero J., Penaloza C., & Cuellar F. Hybrid BCI system to operate an electric wheelchair and a robotic arm for navigation and manipulation tasks // 2016 IEEE Workshop on Advanced Robotics and Its Social Impacts (ARSO). – 2016. – N. 6. – P. 23 – 32.
7. Adov D. Y., Development of software for control of virtual objects using event-related potential p300 // Vestnik NSU. Series: Information Technologies. – 2021. – 19. – P. 5.
8. Auge D. et al. A Survey of Encoding Techniques for Signal Processing in Spiking Neural Networks // Neural Processing Letters. – 2021. – 53. – P. 4693 – 4710.
9. Basu A. Small-Signal Neural Models and Their Applications//IEEE Transactions on Biomedical Circuits and Systems. – 2012.

10. Castanos F. and Franci A. The transition between tonic spiking and bursting in a six-transistor neuromorphic device // 2015 12th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE). – 2015. – 12.
11. Duan F. Design of a Multimodal EEGbased Hybrid BCI System with Visual Servo Module // IEEE Trans. Auton. Ment. – 2015. – N 4. – P. 11.
12. Hodgkin A. L. and Huxley A. F., A quantitative description of membrane current and its application to conduction and excitation in nerve // The Journal of Physiology. – 1952. – 117. – P. 500.
13. Iturrate, I., Antelis, J., and Minguez, J. Synchronous EEG brainactuated wheelchair with automated navigation // Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe. – 2009.
14. Izhikevich Eugene M. Simple Model of Spiking Neurons // Published on IEEE transactions on neural networks. – 2003. – 14. – P. 1569 – 1572.
15. Luna-Alvarez Antonio, M´ujica-Vargas Dante, Mej´ıa-Lavalle Manuel. Convolutional Model with Classification through Izhikevich Neuron // Research in Computing Science. – 2019. – 148. N 10. – P. 65–80.
16. Millan J., Galan F., Vanhooydonck D., Lew E., Philips J., and Nuttin M., in 2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society. – 2009.
17. Mokienko O. A., Lyukmanov R. Kh., Chernikova L. A., Suponeva N. A., M. A. Piradov, and A. A. Frolov. Brain–Computer Interface: The First Experience of Clinical Use in Russia // Human Physiology. – 2016. – Vol. 42. – № 1. – P. 24-31
18. Morash V., Bai O., Furlani S. et al. Classifying EEG signals preceding right hand, left hand, tongue, and right foot movements and motor imageries // Clinical neurophysiology – 2008. – Vol. 119. – № 11. – P. 2570-2578.
19. Shurkhay V. A. and Aleksandrova E. V. and Potapov A. A. and Goryainov S. A., The current state of the brain-computer interface problem // Voprosy neirokhirurgii imeni N.N. Burdenko. – 2015. – 79. – P. 97.

20. Tonin L., Leeb R., Tavella M., Perdikis S., and Millan J. del R., The role of shared-control in BCI-based telepresence // 2010 IEEE International Conference on System. – 2010.
21. Weerasinghe M.M., J.I.E. Ramos, G.Y. Wang, D. Parry, Incorporating structural plasticity approaches in spiking neural networks for EEG modelling. – 2021. – 9. – P. 117-138.
22. Wenjia O. Electroencephelograph based brain machine interface for controlling a robotic arm // IEEE Applied Imagery Pattern Recognition Workshop. – 2013. – 2. – P. 112.

ПРИЛОЖЕНИЯ

Приложение 1. Код программы на Python

Блок 1:

```
import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import math

df = pd.read_excel('TP.xlsx') # Чтение файла

T = df['t'] # Вывод столбца мс

TP9 = df['TP9'] # Вывод столбца значений с электрода TP9

TP10 = df['TP10'] # Вывод столбца значений с электрода TP10
```

Блок 2:

```
lWindow = 50 # Размер окна усреднения

# Задание пустых массивов для отфильтрованных значений

TP9New = []

TP10New = []

# Фильтр для значений с электрода TP9

for t in range(len(T)-lWindow+1):

        a9 = df.loc[t:t+lWindow,['TP9']]

        b9 = a9.to_numpy()

        bSort9 = np.sort(b9, axis=0)

        TP9New = np.append(TP9New, bSort9[int(lWindow/2)])

# Фильтр для значений с электрода TP10
```

```
for t in range(len(T)-lWindow+1):
        a10 = df.loc[t:t+lWindow,['TP10']]
        b10 = a10.to_numpy()
        bSort10 = np.sort(b10, axis=0)
        TP10New = np.append(TP10New, bSort10[int(lWindow/2)])
# Запись отфильтрованных данных в датафрейм
df.insert(3,'TP9 new', pd.Series(TP9New), False)
df.insert(4,'TP10 new', pd.Series(TP10New), False)
# Запись датафрейма в excel
df.to_excel('TPnew.xlsx')
```

Блок 3:

```
plt.figure(1)
plt.plot(TP9, 'b')
plt.plot(TP9New, 'r')
plt.title('TP9')
plt.figure(2)
plt.plot(TP10, 'g')
plt.plot(TP10New, 'y')
plt.title('TP10')
# Вывод графиков
plt.show()
```

Блок 4:

```
NoEL = 10 # Число обучающих примеров
```

```
NoET = 8 # Число тестовых примеров

NoF = 10 # Количество признаков для 1 примера с 1 электрода

N = 200 # Длина примера

n = math.floor(N/NoF) # Длина признака

LearnFeaturesMat9 = np.zeros((NoEL,NoF))

LearnFeaturesMat10 = np.zeros((NoEL,NoF))

LearnInd = [587, 2353, 3241, 4877, 5713, 6697, 8301, 9922, 10778, 12405]

# Разделим на обучающие признаки данные с электрода TP9

for i in range(NoEL):

    for j in range(NoF):

        LearnFeaturesMat9[i,j] =
        np.mean(list(TP9New[LearnInd[i]:LearnInd[i]+(j+1)*n]))

# Разделим на обучающие признаки данные с электрода TP10

for i in range(NoEL):

    for j in range(NoF):

        LearnFeaturesMat10[i,j] =
        np.mean(list(TP10New[LearnInd[i]:LearnInd[i]+(j+1)*n])

# Создадим матрицу обучающих данных data

LearnFeaturesMat = np.concatenate((LearnFeaturesMat9, LearnFeaturesMat10),1)

print(LearnFeaturesMat)

# Графики для определения Ytrue

for i in range(NoEL):

    plt.figure(2*LearnInd[i])
```

```
        plt.plot(TP9New[LearnInd[i]:LearnInd[i]+N], 'r')
        plt.title('TP9')
        plt.figure(2*LearnInd[i]+1)
        plt.plot(TP10New[LearnInd[i]:LearnInd[i]+N], 'g')
        plt.title('TP10')
Ytrue = np.array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1])
```

Блок 5:

```
# Функция активации:
def sigmoid(x):
        return 1 / (1 + np.exp(-x))
#Подсчет потерь
def MSE(yTrue, yPred):
        return ((yTrue - yPred) ** 2).mean()
#Количество нейронов на слоях
Nh = 25
Ninp = 20
Nout = 1
#Задание произвольных начальных синаптических весов
w1 = np.random.uniform(-1, 1, (Nh,Ninp))
if Nout == 1:
        w2 = np.random.uniform(-1, 1, Nh)
else:
        w2 = np.random.uniform(-1, 1, (Nout, Nh))
```

```
b1 = np.zeros(Nh)

b2 = np.zeros(Nout)

h = np.zeros(Nh)

H = np.zeros(Nh)

o = np.zeros(Nout)

Out = np.zeros(Nout)

deltaO = np.zeros(Nout)

deltaH = np.zeros(Nh)

dL_dw_HtoO = np.zeros((Nout, Nh))

dL_dw_HtoO1 = np.zeros(Nh)

dL_dw_ItoH = np.zeros((Nh,Ninp))

ERR = []

EPOCH = []

#Обучение

# Прямая связь

def feedForward(x):

    # Для нахождения скрытых нейронов

    for i in range (Nh):

        h[i] = b1[i]

        for j in range (Ninp):

            h[i] += w1[i,j]*x[j]

            H[i] = sigmoid(h[i])
```

```
        # Для нахождения выходных нейронов
        for i in range (Nout):
            o[i] = b2[i]
            for j in range (Nh):
                if Nout == 1:
                    o[i] += w2[j]*H[j]
                else:
                    o[i] += w2[i,j]*H[j]
                Out[i] = sigmoid(o[i])
        return Out, H
def train(data, Ytrue):
        eta = 0.1
        epochs = 1000
        # backPropagation
        for epoch in range(epochs):
            for x, yTrue in zip(data, Ytrue):
                Out, H = feedForward(x)
                for i in range(Nout):
                    deltaO[i] = (Out[i]-yTrue)*Out[i]*(1-Out[i])
                    yPred = Out[i]
                    for j in range(Nh):
                        dL_dw_HtoO[i,j] = deltaO[i]*H[j]
```

```
                    dL_dw_HtoO1[j] = deltaO*H[j]
                    if Nout == 1:
                        deltaH[j] += deltaO[i]*w2[j]
                    else:
                        deltaH[j] += deltaO[i]*w2[i,j]
                    for k in range(Ninp):
                        dL_dw_ItoH[j,k] = deltaH[j]*H[j]*(1-
                    H[j])*x[k]
            #Обновление весов и порогов
            for i in range(Nh):
                for j in range(Ninp):
                    w1[i,j] -= eta*dL_dw_ItoH[i,j]
            for i in range(Nout):
                for j in range(Nh):
                    if Nout == 1:
                        w2[j] -= eta*dL_dw_HtoO1[j]
                    else:
                        w2[i,j] -= eta*dL_dw_HtoO[i,j]
        if epoch % 1 == 0:
            Error = MSE(yTrue, yPred)
            ERR.append(Error)
            EPOCH.append(epoch)
return w1,w2,ERR,EPOCH
```

```
#Обучаем нейросеть

w1, w2, ERR, EPOCH = train(LearnFeaturesMat, Ytrue)

#Вывод данных об ошибке обучения

plt.plot(EPOCH,ERR)

plt.xlabel('epoch')

plt.ylabel('error')

plt.title('MSE')

plt.show()

print(ERR[-1])
```

Блок 6:

```
TestFeaturesMat9 = np.zeros((NoET,NoF))

TestFeaturesMat10 = np.zeros((NoET,NoF))

TestInd = [13000,17386,15000,14283,20000,23609,25000,16207]

# Разделим на тестовые признаки данные с электрода TP9

for i in range(NoET):

    for j in range(NoF):

        TestFeaturesMat9[i,j] =
        np.mean(list(TP9New[TestInd[i]:TestInd[i]+(j+1)*n]))

# Разделим на тестовые признаки данные с электрода TP10

for i in range(NoET):

    for j in range(NoF):

        TestFeaturesMat10[i,j] =
        np.mean(list(TP10New[TestInd[i]:TestInd[i]+(j+1)*n]))
```

```
# Создадим матрицу тестовых данных
TestFeaturesMat = np.concatenate((TestFeaturesMat9, TestFeaturesMat10),1)
print(TestFeaturesMat)
for i in range(NoET):
        plt.figure(2*TestInd[i])
        plt.plot(TP9New[TestInd[i]:TestInd[i]+N], 'r')
        plt.title('TP9')
        plt.figure(2*TestInd[i]+1)
        plt.plot(TP10New[TestInd[i]:TestInd[i]+N], 'g')
        plt.title('TP10')
LearnZ = np.zeros((NoEL,Ninp))
TestZ = np.zeros((NoET,Ninp))
maxZ = np.zeros(Ninp)
minZ = np.zeros(Ninp)
testPosL = np.zeros((NoEL,Ninp))
testPosT = np.zeros((NoET,Ninp))
dataLT = np.concatenate((LearnFeaturesMat, TestFeaturesMat),0)
F = 25
for i in range(Ninp):
        maxZ[i] = 120
        minZ[i] = 0
        for j in range(NoEL):
```

```
            testPosL[j,i] = LearnFeaturesMat[j,i]
            if LearnFeaturesMat[j,i]<0:
                testPosL[j,i] = -LearnFeaturesMat[j,i]
            LearnZ[j,i] = (LearnFeaturesMat[j,i]-minZ[i])*F/(maxZ[i]-minZ[i])
        for k in range(NoET):
            testPosT[k,i] = TestFeaturesMat[k,i]
            if TestFeaturesMat[k,i]<0:
                testPosT[k,i] = -TestFeaturesMat[k,i]
            TestZ[k,i] = (TestFeaturesMat[k,i]-minZ[i])*F/(maxZ[i]-minZ[i])
for j in range(NoEL):
    print(j,'LearnZ:')
    print(LearnZ[j,:])
for k in range(NoET):
    print(k,'TestZ:')
    print(TestZ[k,:])
```

Блок 7:

```
Cm = 100
red = 0.2
Time = 2000
I0 = 3
I1 = 98*Cm*red
# вывод матицы спайков во времени на входных нейронах
def SpikeInp(x):
```

```
    Period = np.ones(Ninp)
    input = np.ones((Ninp,Time))*I0
    for i in range(Ninp):
        if x[i]==0:
            for t in range(Time):
                input[i,t] = I0
        if x[i]!=0:
            Period[i] = 1000/x[i]
            pa = 0
            p = []
            while pa <= Time-1:
                p.append(math.ceil(pa))
                pa += abs(Period[i])
            for ap in zip(p):
                if x[i]<0:
                    input[i,ap] = -I1
                elif x[i]>0:
                    input[i,ap] = I1
    return input
# вывод времен, когда случаются спайки в нейроне (универсальная функция для всех скрытых нейронов)
def IzhikevichHidden(x,j):
    T = []
```

```
V = []
tFired = []
vFired = []
firedT = []
kolFired = 0
izh_outH = np.zeros(Time)
Iinp = SpikeInp(x)
Isyn = np.zeros(Time)
for t in range(Time):
        for i in range(Ninp):
                Isyn[t] += w1[j,i]*Iinp[i,t]
if Isyn[t]<0:
        b = 0.25
        d = 2
else:
        b = 0.2
        d = 8
c = -65
a = 0.02
v = c*np.ones((1,1))
u = b*v
for t in range(1, Time+1):
```

```
            if v >= 30:
                if t!=Time:
                    ed.append(t)
                vFired.append(float(v[0]))
                kolFired+=1
            fired = np.where(v>=30)
            V.append(float(v[0]))
            firedT.append([t + 0 * fired[0], fired[0]])
            v[fired] = c
            u[fired] +=d
            v += 0.5*(0.04*v**2+5*v+140-u+Isyn[t-1])/Cm
            v += 0.5*(0.04*v**2+5*v+140-u+Isyn[t-1])/Cm
            u += a*(b*v-u
            if v >= 30:
                if t!=Time:
                    if Isyn[t-1]<0:
                        izh_outH[t-1] = -1
                    elif Isyn[t-1]>0:
                        izh_outH[t-1] = 1
            T.append(t)
        return V,T,kolFired,tFired,izh_outH
def IzhikevichOut(j,Iinp):
```

```
To = []
Vo = []
tFiredO = []
vFiredO = []
firedTO = []
kolFiredO = 0
izh_outO = np.zeros(Time)
if Nout==1:
    Isyn = np.zeros(Time)
else:
    Isyn = np.zeros((Nout,Time))
for t in range(Time):
    for i in range(Nh):
        if Nout==1:
            Isyn[t] += w2[i]*Iinp[i,t]
        else:
            Isyn[j,t] += w2[j,i]*Iinp[i,t]
if Isyn[t]<0:
    b = 0.25
    d = 2
else:
    b = 0.2
```

```
        d = 8
c = -65
a = 0.02
v = c*np.ones((1,1))
u = b*v
for t in range(1, Time+1):
        if v >= 30:
                tFiredO.append(t)
                vFiredO.append(float(v[0]))
                kolFiredO+=1
        fired = np.where(v>=30)
        Vo.append(float(v[0]))
        firedTO.append([t + 0 * fired[0], fired[0]])
        v[fired] = c
        u[fired] +=d
        v += 0.5*(0.04*v**2+5*v+140-u+Isyn[t-1])/Cm
        v += 0.5*(0.04*v**2+5*v+140-u+Isyn[t-1])/Cm
        u += a*(b*v-u)
        if v >= 30:
                if t!=Time:
                        if Isyn[t-1]<0:
                                izh_outO[t-1] = -1
```

```
                elif Isyn[t-1]>0:
                    izh_outO[t-1] = 1
            To.append(t)
        return Vo,To,tFiredO,vFiredO,kolFiredO,izh_outO
```

Блок 8:

```
timeAr = 1000
T = 4*timeAr
comand = 4
time_stop = T*comand*4
classMove = 0
import serial
import time
ser = serial.Serial("COM3", 9600)
time.sleep(2) #ждем 2 секунды чтобы установилась последовательная связь
print ("CAR IS OK")
```

Блок 9:

```
up = [0, 5*timeAr, 8*timeAr, 12*timeAr, 18*timeAr, 23*timeAr, 25*timeAr,
31*timeAr, 34*timeAr, 37*timeAr, 42*timeAr, 44*timeAr, 48*timeAr,
52*timeAr, 58*timeAr, 60*timeAr]

left = [timeAr, 4*timeAr, 9*timeAr, 13*timeAr, 17*timeAr, 20*timeAr,
24*timeAr, 29*timeAr, 35*timeAr, 39*timeAr, 41*timeAr, 46*timeAr,
51*timeAr, 53*timeAr, 59*timeAr, 61*timeAr]

down = [2*timeAr, 6*timeAr, 11*timeAr, 14*timeAr, 16*timeAr, 21*timeAr,
26*timeAr, 30*timeAr, 32*timeAr, 36*timeAr, 40*timeAr, 45*timeAr,
50*timeAr, 54*timeAr, 56*timeAr, 62*timeAr]

right= [3*timeAr, 7*timeAr, 10*timeAr, 15*timeAr, 19*timeAr, 22*timeAr,
27*timeAr, 28*timeAr, 33*timeAr, 38*timeAr, 43*timeAr, 47*timeAr,
49*timeAr, 55*timeAr, 57*timeAr, 63*timeAr]
```

Блок 10:

```
for i in range(NoET):
```

```
inputH = np.ones((Nh,Time+1))*I0
print('Проверка на тестовом примере №', i+1)
for j in range(Nh):
    index = i
    V,T,kolFired,tFired,izh_outH = IzhikevichHidden(TestZ[index,:],j)
    if kolFired == 0:
        for t in range(Time):
            inputH[j,t] = I0
    if kolFired != 0:
        for tf in zip(tFired):
            if tf!=Time:
                inputH[j,tf] = I1
for k in range(Nout):
    Vo,To,tFiredO,vFiredO,kolFiredO,izh_outO =
    IzhikevichOut(k,inputH)
    print('количество спайков:', kolFiredO)
    #Классификация движений робота
    if kolFiredO==0:
        print('stop')
        classMove = 0
    else:
        for i in range(16):
            for j in range(timeAr):
                if TestInd[index] == up[i]+j:
                    print('up')
                    classMove = 1
                    break
                elif TestInd[index] == left[i]+j:
                    print('left')
```

```
                    classMove = 2
                    break
                elif TestInd[index] == down[i]+j:
                    print('down')
                    classMove = 3
                    break
                elif TestInd[index] == right[i]+j:
                    print('right')
                    classMove = 4
                    break
if classMove == 0:
        ser.write(b's')
        print("car stop")
        time.sleep(1)
elif classMove == 1:
        ser.write(b'u')
        print("car go up")
        time.sleep(1)
elif classMove == 2:
        ser.write(b'l')
        print("car go left")
        time.sleep(1)
elif classMove == 3:
        ser.write(b'd')
        print("car go down")
        time.sleep(1)
elif classMove == 4:
        ser.write(b'r')
```

```
        print("car go right")
        time.sleep(1)
```

Приложение 2. Код для управления роботом на Arduino

Блок 1:

```
#define M1_DIR 4
#define M1_PWM 5
#define M2_DIR 7
#define M2_PWM 6
```

Блок 2:

```
int data;
void motorsInit() {
        pinMode(M1_DIR, OUTPUT);
        pinMode(M1_PWM, OUTPUT);
        pinMode(M2_DIR, OUTPUT);
        pinMode(M2_PWM, OUTPUT);
}
```

Блок 3:

```
void Motors(int Speed1, int Speed2) {
        if (Speed1 > 255) Speed1 = 255;
        if (Speed1 < -255) Speed1 = -255;
        if (Speed2 > 255) Speed2 = 255;
        if (Speed2 < -255) Speed2 = -255;
        if (Speed1 > 0) {
```

```
            digitalWrite(M1_DIR, 1);
            analogWrite(M1_PWM, Speed1);}
        else {
            digitalWrite(M1_DIR, 0);
            analogWrite(M1_PWM, -Speed1);}
          if (Speed2 > 0) {
            digitalWrite(M2_DIR, 1);
            analogWrite(M2_PWM, Speed2);}
        else {
            digitalWrite(M2_DIR, 0);
            analogWrite(M2_PWM, -Speed2);}
        }
void Stop() {
        Motors(0, 0);
}
void Forward (int Speed, int Time) {
        Motors(Speed, Speed);
        delay(Time);
        Stop();
}
void Right (int Speed, int Time) {
        Motors(-Speed, Speed);
```

```
        delay(Time);
        Stop();
}
void Left (int Speed, int Time) {
        Motors(Speed, -Speed);
        delay(Time);
        Stop();
}
Void Back (int Speed, int Time) {
        Motors(-Speed, -Speed);
        delay(Time);
        Stop();
}
```

Блок 4:

```
void setup() {
        Serial.begin(9600);
        motorsInit();
}
```

Блок 5:

```
void loop() {
        while (Serial.available()) {
                data = Serial.read();
        }
```

```
        if (data == 's')
                Stop();
        else if (data == 'u')
                Forward (100, 100);
        else if (data == 'l')
                Left (255, 50);
        else if (data == 'd')
                Back (100, 100);
        else if (data == 'r')
                Right (255, 50);
        Stop();
}.
```