Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Алтайский государственный технический университет им. И.И. Ползунова»

Факультет (институт) информационных технологий

Кафедра «Информатика, вычислительная техника

и информационная безопасность»

Направление 09.03.01 Информатика и вычислительная техника

Допустить к защите в ГЭК

УДК 004.852\_\_\_\_\_\_\_ Заведующий кафедрой ИВТиИБ

А.Г. Якунин

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_2016 г.

**БАКАЛАВРСКАЯ РАБОТА**

**БР 09.03.01.17.000 ПЗ**

Разработка программного комплекса анализа нейронной сетью реографических и кардиографических сигналов

*тема бакалаврской работы*

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

Студент группы ИВТ-22 Смирнова Валерия Вячеславовна

*фамилия, имя, отчество*

Руководитель работы к.т.н., доцент\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Н. Тушев

*должность, ученая степень инициалы,фамилия*

БАРНАУЛ 2016

еферат

**Реферат**

Бакалаврская работа посвящена разработке программного комплекса для выполнения обработки и анализа кардиосигналов и реосигналов с использованием искусственной нейронной сети.

В бакалаврской работе был разработан алгоритм, по которому выполняется предварительная обработка кардиосигналов и последующий их анализ. Для реализации данного алгоритма было разработано четыре программных модуля, каждый из которых выполняет определенную задачу при обработке поступающих сигналов. При последовательной обработке входных данных на выходе системы получается наименование заболевания, соответствующего поступившему сигналу.

Бакалаврская работа содержит 74 страницы, 31 рисунок, 3 приложения.

**The abstract**

Bachelor's work is devoted to the development of software for performing the processing and analysis of cardio and reosignalov using artificial neural network.The algorithm was developed bachelor work on which is performed a preliminary processing of cardio and subsequent analysis. To implement this algorithm was developed four program modules, each of which performs a specific task in the processing of incoming signals. When sequential processing of input output system turns the disease name corresponding to the signal entered.

The work contains 74 pages, 31 drawings, 3 appositions.

*Разраб.*

*№ документа*

*Изм.*

*Подпись*

*Дата*

*Разработка программного комплекса анализа нейронной сетью реографических и кардиографических сигналов*

*БР 090301.17.000 ПЗ*

*Лит*

*Лист*

*Листов*

*АлтГТУ, ФИТ*

*гр. ИВТ-22*

*Утв.*

*Пров.*

*Н.контр.*

*Смирнова В.В.*

*Тушев А.Н.*

*Горячих А.И.*

*Якунин А.Г.*

У

*2*

74

Бакалаврская работа посвящена разработке программного обеспечения для обработки периодических данных мониторинга с применением геометрического паттерна.

В бакалаврской работе разработано программное обеспечение, в котором реализовано формирование простых и сложных геометрических паттернов на основе данных измерений, полученных из БД. Реализована проверка соответствия данных этим паттернам.

Ключевые слова – периодические данные, геометрический паттерн.

Бакалаврская работа содержит 110 страниц, 35 рисунков, 1 приложение.

The abstract

Bachelor's work is devoted to the development of software for processing periodic monitoring data using the geometric pattern.

The bachelor work developed software, which implements the formation of simple and complex geometric patterns on the basis of measurement data obtained from the database. Implemented verification of compliance with data these patterns.

Keywords – periodic data, geometric pattern.

The work consists 110 of pages, contains 35 drawings, 1 apposition.

**Содержание**

[Введение 5](#_Toc454494868)

[1 Описание предметной области 6](#_Toc454494869)

[1.1 Область применения 6](#_Toc454494870)

[1.2 Обзор аналогичных разработок 11](#_Toc454494871)

[1.3 Постановка задачи 13](#_Toc454494872)

[2 Разработка программного комплекса 14](#_Toc454494873)

[2.1 Описание модулей 14](#_Toc454494874)

[2.1.1 Модуль преобразования данных 14](#_Toc454494875)

[2.1.2 Модуль выполнения фильтрации 14](#_Toc454494876)

[2.1.3 Модуль выделения параметров ЭКГ 16](#_Toc454494877)

[2.1.4 Модуль работы с нейронной сетью 17](#_Toc454494878)

[2.2 Алгоритмы реализации программного комплекса 19](#_Toc454494879)

[2.2.1 Модуль преобразования данных 19](#_Toc454494880)

[2.2.2 Модуль выполнения фильтрации 21](#_Toc454494881)

[2.2.3 Модуль выделения параметров ЭКГ 24](#_Toc454494882)

[2.2.4 Модуль работы с нейронной сетью 27](#_Toc454494883)

[3 Обзор результатов работы 34](#_Toc454494884)

[3.1 Разработанное программное обеспечение 34](#_Toc454494885)

[3.1.1 Модуль преобразования данных 34](#_Toc454494886)

[3.1.2 Модуль выполнения фильтрации 38](#_Toc454494887)

[3.1.3 Модуль выделения параметров ЭКГ 41](#_Toc454494888)

[3.1.4 Модуль работы с нейронной сетью 43](#_Toc454494889)

[3.2 Выбор среды разработки 48](#_Toc454494890)

[Заключение 49](#_Toc454494891)

[Список использованных источников 50](#_Toc454494892)

[Приложение А Задание на выполнение бакалаврской работы 52](#_Toc454494893)

[Приложение Б Руководство программиста по установке разработанного ПО 54](#_Toc454494894)

[Приложение В Листинг разработанного программного обеспечения 56](#_Toc454494895)

Введение

В настоящее время сердечно-сосудистые заболевания являются самой распространенной причиной смерти во всем мире. В связи с этим они нуждаются в качественной диагностике. Одним из самых информативных методов обследования состояния сердца является изучение его электрической деятельности. Данный метод предполагает регистрацию электрических потенциалов работы сердца, при помощи специального аппарата (электрокардиографа), выполняя отображение полученных данных, в виде графика, на бумаге или на экране монитора.

Электрографический метод диагностики заболеваний требует высокой точности анализа полученных результатов. Но, т.к. зачастую, достаточно серьезные сердечные заболевания отражаются на электрокардиограмме (ЭКГ) лишь незначительным изменением амплитуды и формы пиков, традиционные методы анализа ЭКГ не всегда позволяют достигнуть необходимой точности. Очень часто на правильность поставленного диагноза влияет опыт и уровень квалификации врача. Чтобы исключить «человеческий фактор», желательно автоматизировать анализ ЭКГ, и найти такой метод, который был бы способен распознавать наиболее характерные изменения на ЭКГ при тех или иных заболеваниях.

Данная работа осуществляет решение поставленной задачи посредством использования искусственных нейронных сетей (ИНС). Нейронные сети способны выделять полезную составляющую при не совсем точной или неопределенной информации. Таким образом, использование интеллектуальных систем на основе ИНС позволяет выявлять даже самые незначительные отклонения во входном сигнале, которые могут играть важную роль в постановке диагноза.

1 Описание предметной области

**1.1 Область применения**

Для выявления диагностических признаков сердечно-сосудистых заболеваний существует большое множество инструментальных методов. Но все методы значительно отличаются друг от друга по диагностической информативности, доступности и популярности. Все три параметра достаточно трудно совместить в каком-то конкретном методе, так, например, существуют методы информативные и доступные, но не пользующиеся популярностью в практической диагностике. К таким методам относится векторкардиография. Так же существуют методы, сочетающие в себе популярность и информативность, но имеющие высокую стоимость. Примером этой группы может служить метод ультразвуковых измерений (УЗИ).

К одним из наиболее популярных методов диагностики сердечно-сосудистых заболеваний относятся реоэнцефалография (РЭГ) и электрокардиография (ЭКГ) [1]. Эти методы отличаются наличием четко определенных процедур анализа регистрируемых кривых и правил построения диагностических заключений.

Рассмотрим основные характеристики данных инструментальных методов.

Реография представляет собой неинвазивный метод исследования динамики пульсового кровенаполнения как органов и тканей, так и отдельных участков тела. Данный метод основан на регистрации изменения электрического сопротивления тканей, которое происходит вследствие изменения кровенаполнения. Сопротивление снижается при увеличении кровенаполнения и увеличивается, при его уменьшении. Эти колебания регистрируются специальным прибором - реографом в виде кривой -реограммы.

Электрическое сопротивление живых тканей, органов или участков тела определяется путем пропускания через эти ткани переменного электрического тока высокой частоты, но слабого по силе. Ткань при этом рассматривается, как электрический проводник, имеющий ионную проводимость. При прохождении электрического тока в живой ткани происходит воздействие различных факторов, таких как мембранный потенциал, наличие токов действия и покоя, медленные колебания потенциала, но основным определяющим фактором является сопротивление.

Весьма существенным преимуществом метода реографии является возможность одновременного изучения кровенаполнения различных сосудистых областей - мозга, легких, печени, почек, конечностей и других областей, трудно доступных для исследования другими методами. Реография дает возможность оценить состояние кровообращения в исследуемых областях, судить о состоянии тонуса сосудов, их проводимости.

Электрокардиография представляет собой информативный и наиболее распространенный метод исследования состояния сердечно-сосудистой системы. Это неинвазивный метод регистрации разности потенциалов электрического поля сердца, возникающего при его деятельности. Регистрация осуществляется при помощи специального аппарата – электрокардиографа. Результат отображается на листе бумаги или на экране монитора в виде кривой, которая называется электрокардиограммой [1].

Для получения полной картины электрической активности сердца необходимо проводить измерения разности потенциалов. Данные измерения можно проводить практически в любых двух точках тела человека с различными потенциалами. Но рекомендуется делать измерения в определенных точках. Расположение двух электродов в этих точках называется ЭКГ-отведением. Отведения имеют полярность, которая указываю на то, в какую сторону протекает ток по оси отведения. Полюс, который имеет больший потенциал, называется положительным, его подключают к положительному полюсу гальванометра электрокардиографа. На ЭКГ возникает положительный зубец, если суммарный вектор электрического поля сердца направлен к положительному полюсу отведения. Если от него, то возникает отрицательный зубец [2].

В настоящее время различают 12 отведений ЭКГ, запись которых необходимо при любом исследовании сердечно-сосудистой системы пациента. Различают три стандартных отведения – I, II, III; три усиленных однополюсных отведения от конечностей – aVL, aVR, aVF и шесть грудных отведений – V1, V2, V3, V4, V5, V6. Все представленные отведения отражаются на пленке электрокардиограммы и позволяют проанализировать работу того или иного участка сердца [2, 3].

Для записи стандартных отведений электрические потенциалы отводятся от конечностей, представив при этом обе руки и левую ногу как углы равностороннего треугольника. Эти отведения фиксируют разность потенциалов между двумя точками электрического поля, удаленными от сердца и расположенными на конечностях.

I отведение представляет собой разность потенциалов между левой (+) и правой (-) рукой. Оно отражает потенциалы передней и боковой стенок левого желудочка.

II отведение – разность потенциалов между правой рукой (-) и левой ногой (+). Это отведение является промежуточным относительно I и III.

II отведение – разность потенциалов между левой ногой (+) и левой рукой (-).Данное отведение показывает потенциалы задней и нижней стенки левого желудочка.

Существенным недостатком стандартных отведений является то, что они отражают изменения потенциалов электрического поля сердца во фронтальном сечении, не затрагивая при этом переднезаднего направления.

При записи усиленных отведений потенциал отводится только от одной конечности человеческого тела с помощью активного электрода. Вторая точка, на которую накладывается неактивный электрод, должна иметь нулевой потенциал, но на человеческом теле такой точки нет, поэтому необходимо ее создать. Для этого соединяют три электрода, которые накладываются на три конечности, сумма которых практически равна нулю. При выполнении данных измерений на практике отключают тот электрод, на конечность которого наложен активный.

aVR – отведение от правой руки.

aVL – отведение от левой руки.

aVF – отведение от левой ноги.

Данные отведения также расположены во фронтальной плоскости, поэтому при их помощи получают только дополнительную информацию к уже имеющейся. Так, отведение aVR направлено противоположно оси I и II стандартных отведений, поэтому его ЭКГ напоминает перевернутую ЭКГ в этих отведениях. aVL дополняет I отведение, aVF – II и III.

Грудные однополюсные отведения регистрируют разность потенциалов между активным положительным электродом, установленным в определенных точках на поверхности грудной клетки, и электродом, который образуется при соединении трех конечностей (правой и левой руки, левой ноги), потенциал которых близок к нулю.

Активный электрод накладывают на следующие точки:

V1 – четвертое межреберье по правому краю грудины.

V2 – четвертое межреберье по левому краю грудины.

V3 – посредине расстояния между точками V1 и V4, примерно на уровне IV ребра по левой парастернальной линии.

V4 – у верхушки сердца или пятое межреберье чуть внутрь от среднеключичной линии.

V5 – по передней подмышечной линии на уровне V4.

V6 – по средней подмышечной линии на уровне V4.

Отличие грудных отведений в том, что они регистрируют сигналы преимущественно в горизонтальной плоскости, т.е. переднезаднем направлении.

ЭКГ отражает последовательный охват возбуждением отделов миокарда. На графике данные процессы отражаются в виде зубцов. Зубцы могут быть положительными и отрицательными. Их амплитуда измеряется в милливольтах (мВ), а длительность в секундах (с).

Первый зубец, который появляется на ЭКГ – это зубец P. Он отражает деполяризацию и возбуждение предсердий.

После зубца P на ЭКГ регистрируется QRS-комплекс. Он отражает деполяризацию и возбуждение желудочков. Комплекс состоит из трех зубцов: Q, R и S. Зубец R положительный и является главным в этом комплексе. Зубцы Q и К расположены до и после R-зубца и являются отрицательными.

Следом за QRS-комплексом, спустя небольшой промежуток времени следует T-зубец, который чаще всего является положительным. Он отражает процесс реполяризации желудочков, т.е. переход их из состояния возбуждения в состояние покоя.

Иногда после R-зубца может быть отмечен зубец U, который связан с реполяризацией проводящей системы.

Между зубцами можно выделить интервалы и сегменты.

Интервал P-Q отражает предсердно-желудочковую проводимость (АВ-проводимость). Его измеряют от начала зубца P до конца зубца Q.

Сегмент ST отражает медленную фазу реполяризации желудочков. В норме он располагается практически на изолинии и имеет слегка восходящее направление.

Интервал Q-T измеряется от начала QRS-комплекса до конца зубца T. Его продолжительность зависит от частоты сердечных сокращений (ЧСС) – чем выше ЧСС, тем меньше его длительность.

Интервал R–R отражает длительность сердечного цикла. При синусовом регулярном ритме, он постоянен (допустима погрешность в 10%) [2, 3].

Применение метода электрокардиографии позволяет своевременно диагностировать многие сердечно-сосудистые заболевания – инфаркт миокарда, гипертоническую болезнь, нарушения ритма и проводимости и т.д., а также позволяет оценить эффективность медикаментозного или хирургического лечения болезней сердца [2].

**1.2 Обзор аналогичных разработок**

В связи с тем, что данная проблема имеет глобальный масштаб, в мире разрабатываются различные комплексы для анализа электрокардиограмм.

В частности, 1993 году в центре «Медицина и электроника» при Алтайском государственном техническом университете был разработан медицинский диагностический комплекс ЭФКР-4, в котором был реализован ряд популярных, доступных и информативных методов функциональной диагностики. Данный комплекс многофункционален и позволяет проводить широкий спектр кардиографических, реографических, фонокардиографических и спирометрических исследований организма человека. Особенностью ЭФКР-4 является цифровой способ регистрации сигналов с их передачей через последовательный порт в компьютер. Ведь именно компьютерный анализ сигналов позволяет на современном уровне выполнять диагностические процедуры [1].

В разработке данного комплекса участвовали Якунин А.Г. и Тушев А.Н. Разработкой программного обеспечения для комплекса также занималась Гарколь Н.С.

Данная работа выполняет разработку математической модели реографических кривых с целью построения обучающей выборки искусственной нейронной сети, а также алгоритм вычисления параметров реограммы. По полученным данным выполняется построение диагностических заключений нарушений кровотока с применением ИНС. Также работа предполагает выявление диагностических признаков сердечно-сосудистых заболеваний по параметрам II отведения ЭКГ путем применения метода потенциальных функций и локализацию поражений сердца по параметрам 12 отведений на основе обучения ИНС [4].

Анализ реограмм с использованием нейронной сети используется в ряде лечебных учреждений, где используется прибор ЭФКР-4.

Разработанные алгоритмы успешно используются в практическом здравоохранении.

Помимо этого комплекса существуют и другие. Так, при институте физиологии им. И.П. Павлова РАН в Санкт-Петербурге был разработан комплекс по анализу ЭКГ. Этот программно-аппаратный комплекс позволяет выполнять снятие электрокардиограммы не только у человека, но и у животных, что позволяет проводить тестирование новых лекарств на их безопасность.

Комплекс состоит из нескольких независимых модулей. Основной из них выполняет непосредственно снятие ЭКГ. Осуществляет обработку сигнала, расчет частоты сердечных сокращений, а также вывод данных на экран и в файл. Запись идет в два файла: с ЧСС и с необработанной электрокардиограммой.

Второй модуль данного комплекса выполняет работу с полученной электрокардиограммой. Здесь замеряется период и амплитуда выбранного пика, а также выполняется счетчик аномалий ритма.

Также в качестве примера анализатора ЭКГ можно привести программу, разработанный в Запорожском Национальном Техническом университете на Украине. Данная разработка выполняет загрузку ЭКГ сигнала, его делинеацию при помощи вейвлет-преобразования и определение класса заболевания на основе обученной нейронной сети. В результате выполнения на выход поступает заключение о наличии сердечно-сосудистого заболевания и его наименовании или его отсутствии.

**1.3 Постановка задачи**

Целью данной работы является разработка программного комплекса для выявления сердечно-сосудистых заболеваний путем анализа ЭКГ с использованием искусственной нейронной сети. Особенностью данной работы является существенное увеличение количества выявляемых заболеваний, использование характерных параметров всех 12 отведений ЭКГ (стандартных, усиленных и грудных), и кроме этого, используется альтернативный подход построения обучающей выборки по самим дискретным отчетам ЭКГ без предварительного оценивания параметров.

Для достижения поставленной цели необходимо выполнить ряд подзадач:

а) Построение алгоритма работы комплекса и определение необходимых модулей для его правильной реализации.

б) Разработка современного графического интерфейса для создаваемого программного обеспечения с использованием современного языка программирования.

3) Создание модуля для перевода графических изображений кардиосигнала в его дискретные отсчеты.

4) Создание модуля фильтрации входящих сигналов ЭКГ с целью избавления от помех.

5) Создание модуля выделения QRS-комплекса и вычисления характеристических параметров ЭКГ.

6) Создание модуля для осуществления работы с нейронной сетью.

Отличительной особенностью данной работы является возможность подачи на вход нейронной сети не только числовых значений параметров, но и дискретных отсчетов всех двенадцати отведений кардиосигнала.

2 Разработка программного комплекса

**2.1 Описание модулей**

## 2.1.1 Модуль преобразования данных

Для достижения поставленной цели было разработано четыре программных модуля. Первый модуль осуществляет преобразование графических изображений кардиосигналов в их числовую форму.

В настоящее время стало недостаточно использовать набор кардиограмм, имеющихся в различных медицинских учреждениях, так как требуется достаточно большая выборка, с увеличенным количеством распознаваемых заболеваний.

Перед написанием данного модуля необходимо произвести отбор примеров электрокардиограмм для разнообразных заболеваний по различным атласам ЭКГ. Для этого нужно отобрать графические изображения кардиосигналов того или иного заболевания и разбить его на 12 более мелких изображений, отображающих кардиосигналы от каждого отведения.

Поскольку созданный набор данных представлен в виде графических изображений кардиосигналов, необходимо разработать программный модуль, который смог бы выполнять преобразование графических сигналов в их дискретные отсчеты.

Первый модуль осуществляет данную процедуру, создавая тем самым набор дискретных отсчетов всех двенадцати отведений кардиосигнала. Данные полученные на этом этапе используются при работе последующих модулей.

## 2.1.2 Модуль выполнения фильтрации

Перед выполнением анализа кардиосигналов необходимо выполнить их цифровую фильтрацию. Это связано с тем, что любой сигнал, в том числе и кардиосигнал, помимо полезной составляющей содержит высокочастотные или низкочастотные помехи, которые необходимо удалить для устранения ошибок при вычислении параметрических характеристик сигнала.

Кардиосигналы зачастую представляют собой довольно сложные сигналы с циклически повторяющимися информативными участками, в виде разнополярных импульсов (зубцов). Параметрические характеристики значений именно этих зубцов, а также интервалов и сегментов между ними и определяют состояние сердца. Зубцы зачастую имеют довольно непредсказуемые формы и параметры, и наличие помех привело бы к тому, что корректный анализ сигнала был бы просто невозможен.

В данной работе для цифровой обработки сигналов используется преобразование Фурье. Так как в данной работе сигнал имеет достаточно небольшую продолжительность (1 - 2 секунды) и обычно частота дискретизации равна 360 Гц, для фильтрации достаточно использовать дискретное преобразование Фурье с удалением частот, считающимися помехами в каждом конкретном случае. Так, для кардиосигналов помехой считаются частоты свыше 35 Гц, дополнительно проверяется обнуление сигнала на частоте 50 Гц (удаление сетевой наводки). Помехой для реосигнала являются все составляющие с частотой свыше 22 Гц.

Для выполнения фильтрации поступающих отсчетов кардиосигналов был разработан модуль, осуществляющий выделение полезной составляющей сигнала путем удаления частот, являющихся помехами. Несмотря на то, что для фильтрации сигналов в данной работе достаточно использовать дискретное преобразование Фурье (ДПФ), также было реализовано и быстрое преобразование Фурье (БПФ) на тот случай, если на вход будет подан более продолжительный сигнал.

## 2.1.3 Модуль выделения параметров ЭКГ

Для параметрического подхода построения обучающей выборки необходимо иметь алгоритмы вычисления параметров кардиосигнала. К выделению параметров можно приступать после выполнения фильтрации кардиосигнала.

К таким параметрам относится частота сердечных сокращений (ЧСС), амплитуда и продолжительность зубцов P и Т, амплитуды зубцов QRS-комплекса, длительность QRS-комплекса, а так же длительность интервала QT и сегментов ST, PQ. Помимо этого важным показателем является смещение сегмента ST относительно изоэлектрической линии. Выделение необходимых параметров необходимо провести для каждого из 12 отведений электрокардиограммы.

Расчет характеристических параметров выполняется для того, чтобы по их значениям определить, входят они в категорию нормальных показателей или же при работе сердца выявлены какие-то нарушения. Определение класса заболевания либо его отсутствия, к которому относится результат анализа деятельности сердца, выполняется путем применения обученной нейронной сети, на вход которой подается файл обучающей выборки, построенный по значениям рассчитанных параметров.

Рассмотрим нормальные значения для параметров ЭКГ:

а) зубец P:

1) амплитуда: не более 0,5 (0,05 мВ);

2) продолжительность: 0,07 – 0,1 с;

б) зубец Q:

1) амплитуда: не более 2 мм (0,2 мВ);

2) продолжительность: не более 0,03 с;

в) зубец R:

1) амплитуда: не более 25 мм (2,5 мВ);

г) зубец S:

1) амплитуда: не более 2 мм (0,2мВ);

д) зубец T:

1) амплитуда: не более 1,6 мм (0,16 мВ);

2) продолжительность: не более 0,2 с;

е) интервал P-Q:

1) продолжительность: 0,12 – 0,2 с;

ж) сегмент ST:

1) смещение: -0,5 мм (0,05 мВ) – 1 мм (0,1 мВ);

з) интервал Q-T:

1) продолжительность: 0,35 – 0,44 с;

и) комплекс QRS:

1) продолжительность: 0,06 – 0,1 с [3].

## 2.1.4 Модуль работы с нейронной сетью

Четвертый модуль осуществляет работу с нейронной сетью. Данный модуль состоит из трех подразделов. Первый из них позволяет просматривать данные заболеваний в виде таблиц параметров ЭКГ либо в виде графиков, построенных по дискретным отсчетам сигналов. Каждый раз выводятся данные только одного отведения и, нажимая на кнопку, можно просмотреть данные последующих отведений.

Второй раздел осуществляет обучение нейронной сети. На вход сети подается обучающий файл выборки, по которому происходит обучение. В данной работе существует два таких файла. Один из них содержит данные заболеваний в виде дискретных отсчетов кардиосигнала, а второй – в виде значений основных параметров ЭКГ.

Каждый обучающий файл содержит набор образов, которые соответствуют тому или иному заболеванию.

Образ представляет собой массив числовых значений. Для файла с дискретными отсчетами – это числовые значения сигнала, считываемые в течение двух минут, т.е. при частоте дискретизации, равной 360 Гц, их количество равно 720. Данные значения записываются для каждого отведения, т.е. двенадцать строк. Таким образом формируется входной массив, состоящий из 8640 значений.

Файл с характеристическими параметрами также содержит числовые массивы данных для 12 отведений. Количество параметров в данной работе равно 12, следовательно размерность каждого вектора-образа равна 144.

Также первая строчка файла должна содержать такие данные как размерность каждого образа, т.е. количество строк и столбцов, количество классов, между которыми распределены эти образы и общее количество образов, содержащихся в файле.

После каждого описанного образа необходимо указать класс, в виде числового значения, который соответствует тому или иному заболеванию. В данной работе количество выявляемых заболеваний равно десяти. Т.е. каждый класс будет обозначаться числовым значением от 1 до 10.

Фрагмент файла обучающей выборки по дискретным отсчетам кардиосигнала представлен на рисунке 2.1

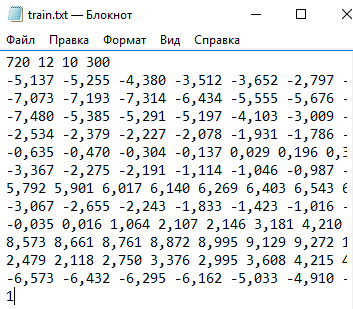


Рисунок 2.1 – Фрагмент обучающего файла

На рисунке 2.2 представлен фрагмент обучающего файла по характеристическим параметрам ЭКГ.

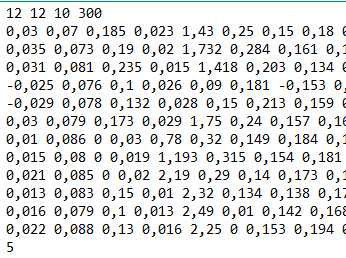


Рисунок 2.2 – Фрагмент обучающего файла

Также четвертый модуль позволяет выполнять проверку правильности определения диагноза после выполнения обучения нейронной сети. Для этого на вход также подается файл, но уже не обучающий, а тестовый, который содержит образы заболеваний. При указании номера вектора, который необходимо распознать, программа должна выдать правильный диагноз заболевания.

**2.2 Алгоритмы реализации программного комплекса**

## 2.2.1 Модуль преобразования данных

Для создания обучающей выборки был отобран набор графических изображений кардиосигналов двенадцати отведений для различных сердечно-сосудистых заболеваний [5]. Но для подачи этих данных на вход нейронной сети необходимо перевести имеющиеся графические данные в дискретные отсчеты этих сигналов.

Для выполнения этой цели был разработан первый модуль. При выполнении описанных ниже действий программа выполняет считывание данных с загруженного изображения и заносит их в виде числового массива в текстовый файл.

Для сохранения файлов с числовыми данными создается новый каталог, куда заносятся файлы при их записи.

В программе реализована работа взаимодействия с файлами при нажатии на кнопки. Первая кнопка позволяет выбрать графический файл для его загрузки в элемент pictureBox1, в котором пользователь и сможет работать с кардиосигналом, а вторая кнопка производит запись уже считанных с изображения данных в файл.

Считывание изображения производится следующим образом: пользователь, после выбора необходимого графического файла, нажимает на изображение в той точке, откуда необходимо начать запись. При этом устанавливается флаг начала записи, который говорит, что последующее движение курсора мыши по элементу будет приводить к записи координат его положения в два числовых массива X и Y соответственно. Для остановки записи, необходимо, чтобы пользователь кликнул мышкой в точке остановки. Тогда флаг записи сбросится в нуль и считывание прекратится.

Считывание координат положения курсора мыши реализовано с помощью события pictureBox1\_MouseMove, которое действует на изменение положения курсора. Флаг записи устанавливается или сбрасывается при срабатывании события pictureBox1\_MouseClick, которое работает при нажатии клавиши мыши по элементу.

После прекращения считывания необходимо нажать кнопку «Записать» и массив со значениями Y-координат будет записан в текстовый файл с названием, которое соответствует отведению кардиосигнала, считывание которого производилось.

Название папки, в которую сохраняются данные и название заболевания записываются в соответствующие текстовые поля на форме. А наименование отведения выбирается с помощью радиокнопки.

Так же данный модуль позволяет выводить считанные данные в виде графиков на форме. Для этого нужно просто перейти в раздел просмотра, выбрать необходимый каталог, все файлы которого необходимо считать и нажать кнопку «Ок». На экране отобразятся графики сигналов кардиограммы для всех 12 отведений по порядку, а также в текстовой метке слева отобразится название заболевания, которое соответствует данным. Для вывода графиков сигналов используется элемент управления chart.

## 2.2.2 Модуль выполнения фильтрации

Для анализа кардиосигналов, необходимо предварительно провести их фильтрацию, т.е. удалить помехи для выделения его полезной составляющей.

Каждый негармонический сигнал во временной области имеет составляющие его гармонические сигналы. Каждый гармонический сигнал имеет свою частоту, амплитуду и фазу. Это говорит о том, что каждый сложный сигнал можно преобразовать из временной области в его эквивалент в частотной области. В данной работе данное преобразование выполняется с помощью преобразования Фурье [6].

Дискретное преобразование Фурье подразумевает сопоставление отчетам сигнала комплексных спектральных отчетов , где . Таким образом для вычисления одного спектрального отчета требуется операций комплексного сложения и умножения [6].

Так как в работе используются сигналы, временной интервал которых короткий (1 - 2 секунды), можно использовать дискретное преобразование Фурье (ДПФ).

Формула (2.1) используется для выполнения ДПФ.

, (2.1)

где – частотный выход ДПФ в -ой точке спектра, ;

– количество отсчетов сигнала;

;

– -ый отсчет во временной области, .

Для представления данных после преобразования используется амплитудный спектр сигнала, который рассчитывается по формуле (2.2).

, (2.2)

где  – квадрат косинусного коэффициента ряда Фурье;

 – квадрат синусного коэффициента ряда Фурье.

Данный график отражает распределение амплитуд сигнала по их частотам.

После этого выполняется обнуление частот, которые являются помехами для того или иного сигнала. Затем осуществляется обратное дискретное преобразование Фурье, в результате чего отображается исходный сигнал, но уже без помех. Формула (2.3) используется для осуществления обратного преобразования Фурье.

, (2.3)

Для более длительных сигналов применяется быстрое преобразование Фурье. Но оно осуществляется для сигналов, количество дискретных отчетов которых является степенью двойки. Иначе сигнал необходимо либо дополнять нулевыми значениями, либо удалять лишние отчеты.

Быстрое преобразование Фурье позволяет ускорить обработку данных путем разделения количества отчетов пополам. Полученное количество отчетов можно делить на 2 далее, пока не останется только 2 отчета. Полученные группы подвергаются по-отдельности дискретному преобразованию Фурье. После выполнения преобразования осуществляется объединение обработанных данных.

Деление можно выполнить используя один из двух алгоритмов: деление по времени или деление по частоте.

Деление по времени заключается в том, что в одну группу объединяются отчеты сигнала с четными номерами, а во вторую – с нечетными.

Деление по частоте определяет в одну группу первую половину отчетов, а во вторую группу – вторую половину.

В данной работе при реализации БПФ используется разделение сигнала по времени. Получившиеся группы подвергаются ДПФ, а затем проходят обратную операцию объединения. Объединение рассчитанных данных осуществляется при помощи формул (2.4 – 2.5).

, (2.4)

. (2.5)

Процесс объединения получил название «бабочка» из-за его формы графа (рисунок 2.3).

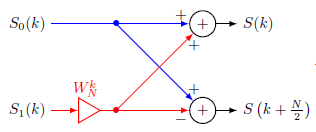


Рисунок 2.3 – Граф «бабочка»

Перед началом выполнения преобразования необходимо выполнить двоично-инверсную перестановку отчетов, это выполняется для того, чтобы выходные данные имели прямой порядок [7, 8, 9].

## 2.2.3 Модуль выделения параметров ЭКГ

Модуль выделения параметров выполняет выделение основных характеристических параметров электрокардиограммы из поступающего на вход программы массива отфильтрованных дискретных отсчетов кардиосигнала. К основным параметрам электрокардиосигнала относится ширина и амплитуда зубцов P, Q и T, высота зубцов P и S. А также продолжительность интервалов сегментов QT и PQ, комплекса QRS и смещение от изолинии сегмента ST [10, 11].

Для реализации модуля был использован алгоритм выделения параметров, описанный ниже.

1) Рассчитываем и .

2) Рассчитываем порог .

3)

4)

5) Если , то 5), иначе 15)

6)

7)

8)

9)

10) Если , то 11), иначе 12)

11) Если , то 6), иначе 9)

12) Если , то и

13)

14) Если , то 5), иначе 17)

15)

16) Переход на 5)

17)

18) Если , то 4), иначе 19)

19)

20)

21)

22)

23)

24)

25)

26)

27) Если , то 28), иначе 29)

28) Если , то 25), иначе 26)

29)

30)

31)

32)

33)

34)

35)

36)

37) Если , то 34), иначе 38)

38)

39)

40) // номер зубца

41) Если , то 42), иначе 44)

42)

43)

44)

45)

46)

47)

48) Если , то 49), иначе 51)

49) Если , то

50) Переход на 47)

51)

52) Если , то и , иначе, если , то и , иначе, если , то 53), иначе, если , то и , иначе, если , то и .

53) Переход 57)

54) и

55) Если , то , ,

56) Переход 52)

57) Расчет интервалов и сегментов

Здесь,

– минимальное значение амплитуды на всем числовом множестве;

– минимальное значение амплитуды на всем числовом множестве;

– среднее значение амплитуды на всем числовом множестве;

– пороговое значение амплитуды;

– значения амплитуды соответствующего зубца;

– значение положения максимума зубца на временной оси;

*–* минимальное значение, которое может быть между зубцами R-R.

– максимальное значение, которое может быть между зубцами R-R.

– максимальное возможное значение между зубцом Q и T.

– максимальное возможное значение половины комплекса QRS.

Обобщенный алгоритм выделения параметров представлен на рисунке 2.4.

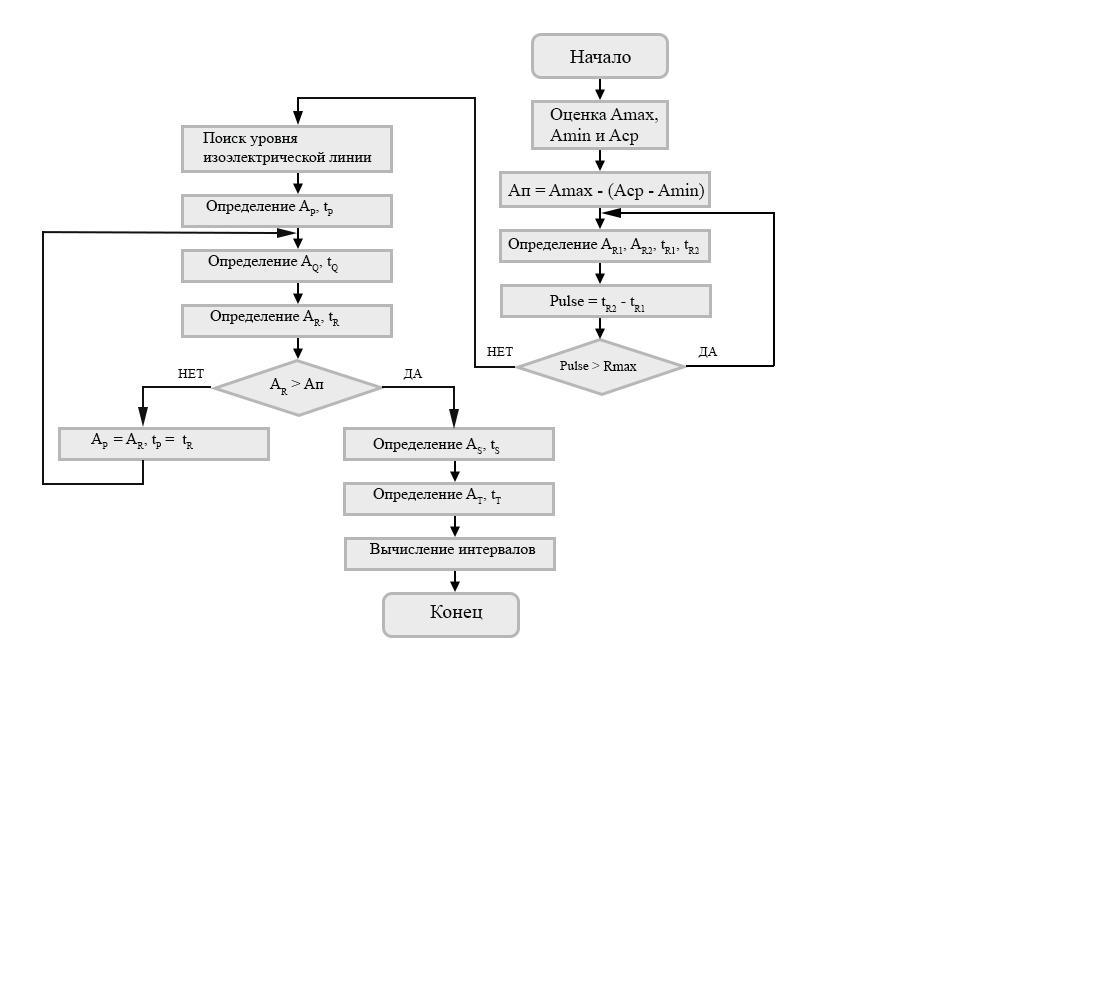


Рисунок 2.4 - Алгоритм выделения параметров

## 2.2.4 Модуль работы с нейронной сетью

Нейронная сеть представляет собой модель биологических нейронных сетей мозга. Нейрон в данной сети - это простой, однотипный элемент.

Для представления нейронной сети можно рассматривать граф, вершинами которого являются искусственные нейроны, а дугами – синаптические связи.

Модель нейронной сети можно выполнить как в программной, так и в аппаратной реализации. В данной работе используется первый способ.

Нейронная сеть состоит из слоев, каждый из которых представляет совокупность нейронов. Слои содержат произвольное количество нейронов, которые не взаимодействуют на уровне одного слоя, но могут взаимодействовать с нейронами предыдущего или последующего слоев.

Основной функцией искусственного нейрона является формирование выходного сигнала, в зависимости от того, какой сигнал был получен на вход.

Входные сигналы каждого нейрона обрабатываются сумматором, выходное значение которого преобразуется активационной функцией, в результате чего получается выходной сигнал нейрона [12].

Внешний вид искусственного нейрона представлен на рисунке 2.5.

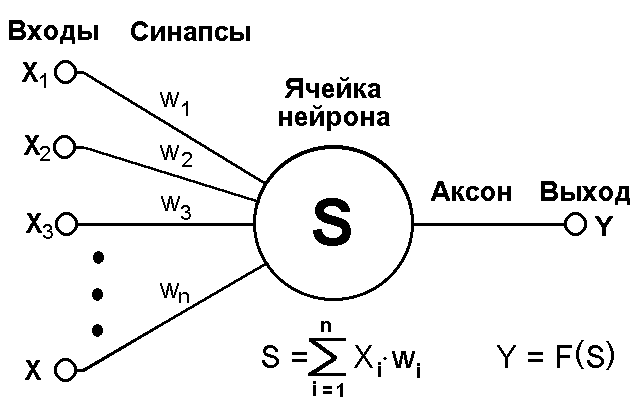


Рисунок 2.5 - Искусственный нейрон

Основными характеристиками нейрона являются его состояние и группа синапсов, которая связывает его с выходами нейронов предыдущего слоя.

Каждый синапс имеет числовое значение синапсической связи, которое называется ее весом .

Текущее состояние нейрона определяется по формуле 2.6.

(2.6)

На выходе нейрона получается значение активационной функции, примененной к полученному состоянию .

В качестве активационной функции зачастую используется сигмоидальная функция. Это обусловлено тем, что она всюду дифференцируема.

Нейронные сети бывают однослойные и многослойные. Однослойные сети имеют только один слой нейронов, которые получают массив входных данных, а выводят уже готовые выходные данные (рисунок 2.6).

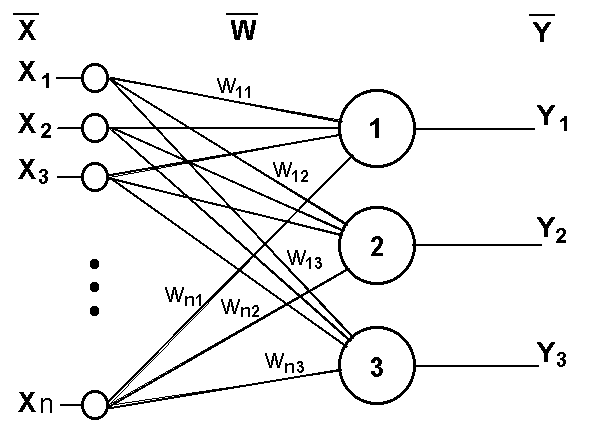


Рисунок 2.6 – Однослойный персептрон

Многослойные сети имеют входной и выходной слой и могут иметь скрытые слои. Количество нейронов на слоях устанавливается произвольным образом. Между слоями установлена связь типа «многие ко многим», т.е. каждый нейрон получает на вход все выходные значения нейронов предыдущего слоя, а его выход передается на все входы нейронов последующего слоя (рисунок 2.7).

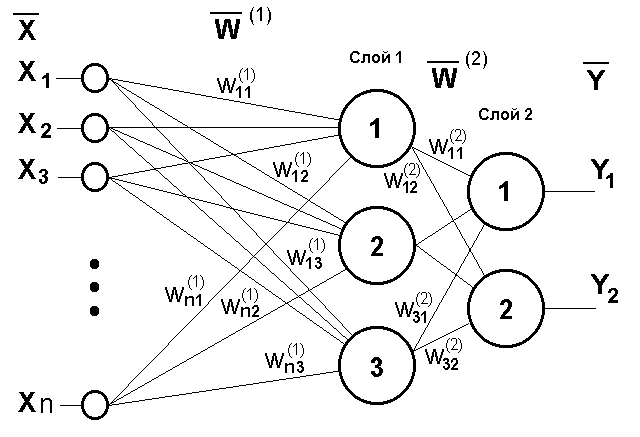


Рисунок 2.7 – Двухслойный персептрон

Перед тем, как использовать нейронную сеть для распознавания образов, ее необходимо обучить, т.е. подстроить ее внутренние параметры под конкретную задачу. Алгоритм обучения нейронной сети представляет собой циклический процесс. Каждый его цикл называется эпохой обучения. Одна эпоха выполняет предъявление всех образов из обучающего файла.

Обучение нейронной сети может быть двух видов: «с учителем» или «без учителя». В данной работе использовался метод обучения «с учителем». Это означает, что обучающая выборка содержит в себе пары: вектор входа и соответствующее ему значение на выходе. Данный вид обучения называется управляемым. В процессе обучения происходит подстраивание весов синаптических связей с целью уменьшения ошибки [13].

Обучение «без учителя» является неуправляемым. В результате такого обучения образы распределяются поклассам, сами классы при этом, а также их количество могут быть заранее не известны. Выходы при таком обучении формируются самостоятельно, а веса вычисляются по определенному алгоритму, который учитывает только входные сигналы, а также производные от них.

При выполнении каждой итерации обучения сети находится ошибка обучения, которая вычисляется путем вычитания из реального значения выхода полученного.

При обучении нейронной сети может возникнуть проблема переобучения, которая возникает при излишне точном соответствии нейронной сети конкретному набору обучающих примеров, в результате чего сеть теряет способность к обобщению.

Причинами переобучения являются слишком долгое обучение, недостаточное количество обучающих примеров, а также очень сложная структура нейронной сети.

Для того, чтобы избежать переобучения используется разделение файлов обучающей и тестовой выборки.

Для обучения сети в данной работе использовался метод обратного распространения ошибки. Данный метод предполагает наличие прямого и обратного прохода по всем слоям нейронной сети. Прямой проход заключается в том, что на входной слой подается сигнал, который путем некоторых вычислений распространяется на последующие слои, пока не получится выходной сигнал на последнем слое. При прямом проходе все синаптические веса нейронов фиксированы, а при обратном – настраиваются по правилу коррекции ошибок, т.е. вычисляется функция ошибки, которая затем распространяется по сети в обратном направлении. Синаптические веса настраиваются таким образом, чтобы уменьшить размер ошибки.

В качестве активационной функции используется, как уже говорилось выше, сигмоидальная функция (2.7).

(2.7)

где – это параметр наклона сигмоидальной функции. Он позволяет строить функции с различной крутизной.

Рассмотрим алгоритм обучения нейронной сети методом обратного распространения ошибки:

а) Проинициализировать синаптические веса случайными значениями.

б) На вход сети подать очередной вектор входного сигнала.

в) Применяя необходимые формулы, вычислить выходное значение сети.

г) Вычислить ошибку функции, отняв от требуемого значения выхода полученное.

д) Выполнить корректировку весов сети для уменьшения ошибки.

е) Повторять шаги б) – д) для каждого обучающего вектора, пока ошибка не достигнет приемлемого уровня или не выполнится указанное число итераций.

Инициализация весов на первом этапе происходит случайными значениями из интервала .

Шаги б) и в) алгоритма реализуют прямой проход, т.е. выполняют вычисления выходного сигнала путем прямого прохода от слоя к слою.

Шаги г) и д) выполняют обратный проход, в ходе чего корректируют веса и уменьшают ошибку.

Рассмотрим подробнее процесс корректировки ошибки.

Для начала приведем формулы для вычисления значений на выходном слое. Введем величину ,

где – это истинное значение выхода нейронной сети;

– это обозначение выходного слоя.

Тогда, применяя это значение, получаем формулу (2.8) для вычисления новых весов.

(2.8)

где – номер текущей итерации обучения;

– значение синаптического веса, который соединяет нейрон слоя с нейроном слоя ;

– коэффициент «скорости обучения». Используется для того, чтобы управлять средней величиной изменения весов;

– выходное значение предыдущего слоя;

– это обозначение предыдущего слоя.

Теперь рассмотрим формулы для корректировки весов на скрытых слоях. Также введем величину ,

где – это номер нейрона в слое.

Веса вычисляются по формуле (2.9)

(2.9)

Данные вычисления производятся до тех пор, пока ошибка не достигнет необходимой точности или не будет выполнено нужное количество итераций [12, 13].

3 Обзор результатов работы

**3.1 Разработанное программное обеспечение**

## 3.1.1 Модуль преобразования данных

Ниже представлена работа программы для перевода графических данных в дискретные отсчеты сигналов.

Данная программа выполняет перевод графических данных в дискретные отсчеты сигналов, а также позволяет просмотреть полученные результаты.

При запуске данного модуля, появляется окно, представленное на рисунке 3.1.

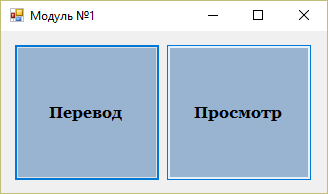


Рисунок 3.1 - Начальное окно программы

Здесь можно перейти в раздел перевода данных либо в раздел просмотра, полученных результатов.

При переходе в раздел перевода данных откроется окно, представленное на рисунке 3.2.

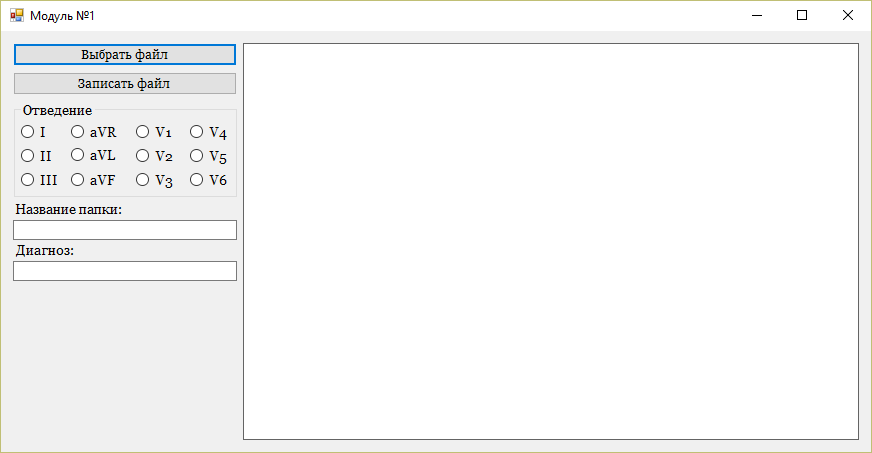


Рисунок 3.2 - Окно перевода данных

Для того, чтобы перевести данные, необходимо выбрать графический файл, нажав на кнопку «Выбрать файл» (рисунок 3.3). Указать, какое отведение изображено на рисунке, поставив необходимый флажок. Это значение используется для названия текстового файла, в который записываются данные. Также, в первом текстовом поле необходимо указать название папки, куда будут сохраняться полученные данные, а во втором тестовом поле указать наименование заболевания (рисунок 3.4). Далее на выбранном изображении необходимо плавно вести курсором мыши по линии сигнала, и получаемые координаты X и Y, будут сохраняться в два соответствующих массива. После выполнения всех описанных действий необходимо нажать кнопку «Записать файл», после чего будет создан текстовый файл с записанными в него значениями координаты Y.

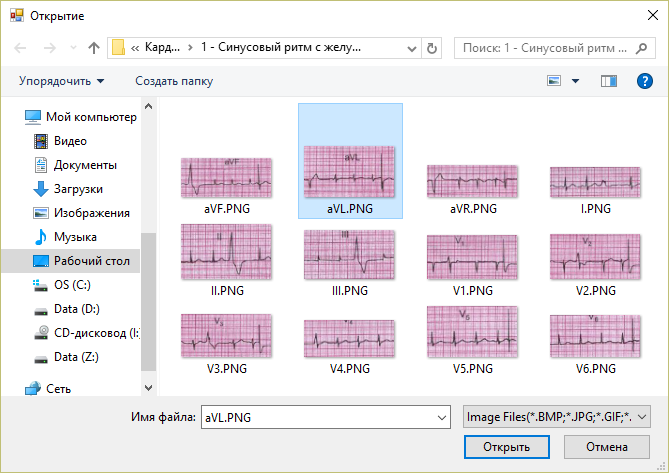


Рисунок 3.3 - Выбор графического изображения

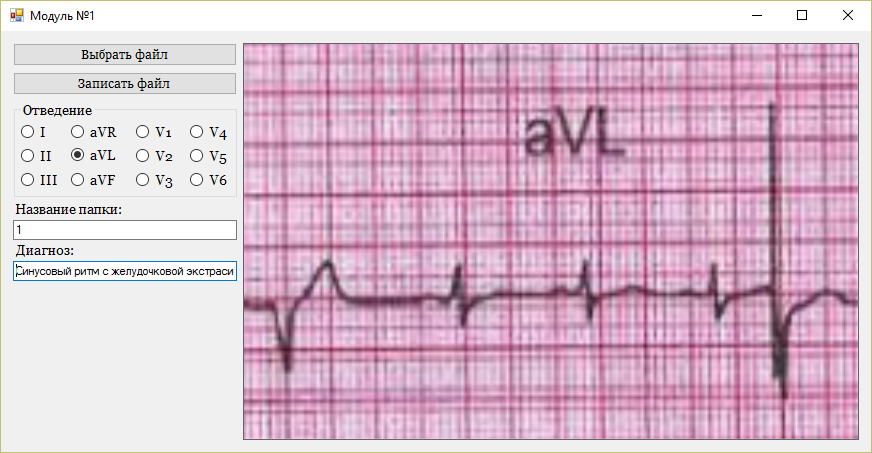


Рисунок 3.4 - Выбранный файл

Пример полученного файла представлен на рисунке 3.5.

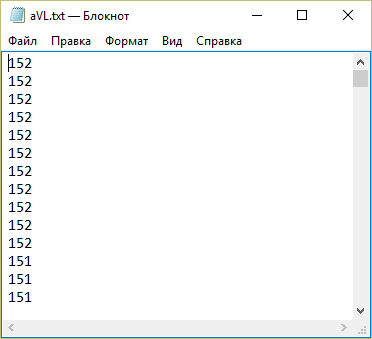


Рисунок 3.5 - Фрагмент полученных данных для отведения aVL

При переходе в раздел просмотра данных откроется окно, представленное на рисунке 3.6.

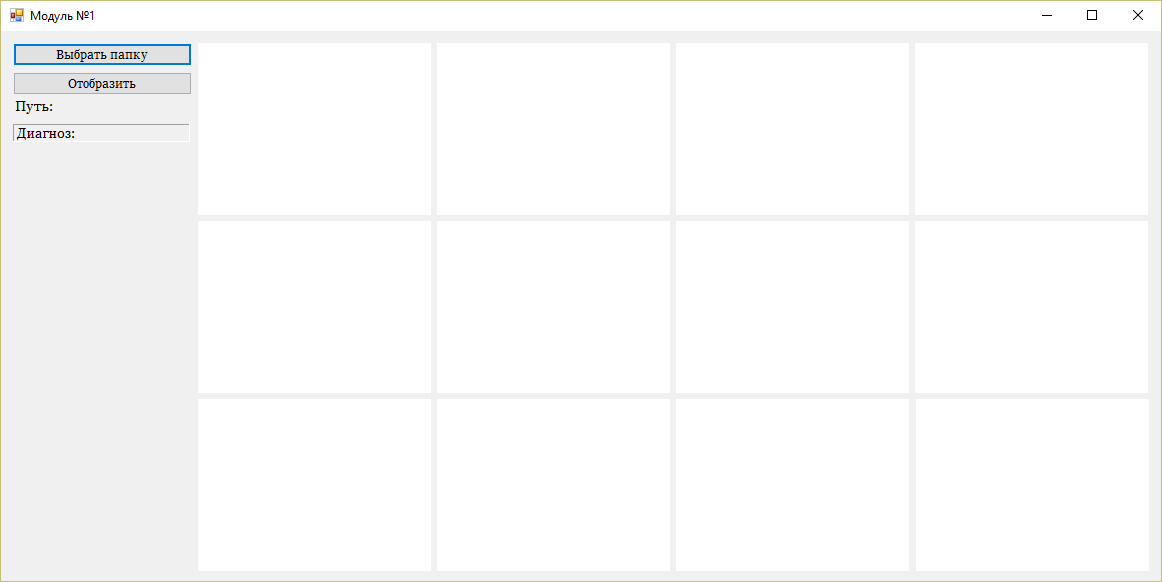


Рисунок 3.6 - Окно просмотра данных

Чтобы просмотреть данные, необходимо выбрать папку с полученными файлами и нажать кнопку «Отобразить». Окно выбора папки представлено на рисунке 3.7.

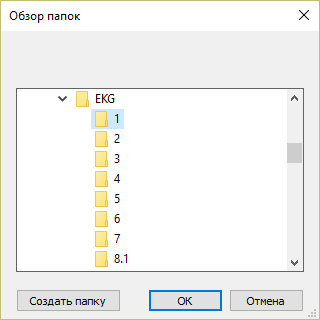


Рисунок 3.7 - Окно выбора файла

Результат отображения представлен на рисунке 3.8.

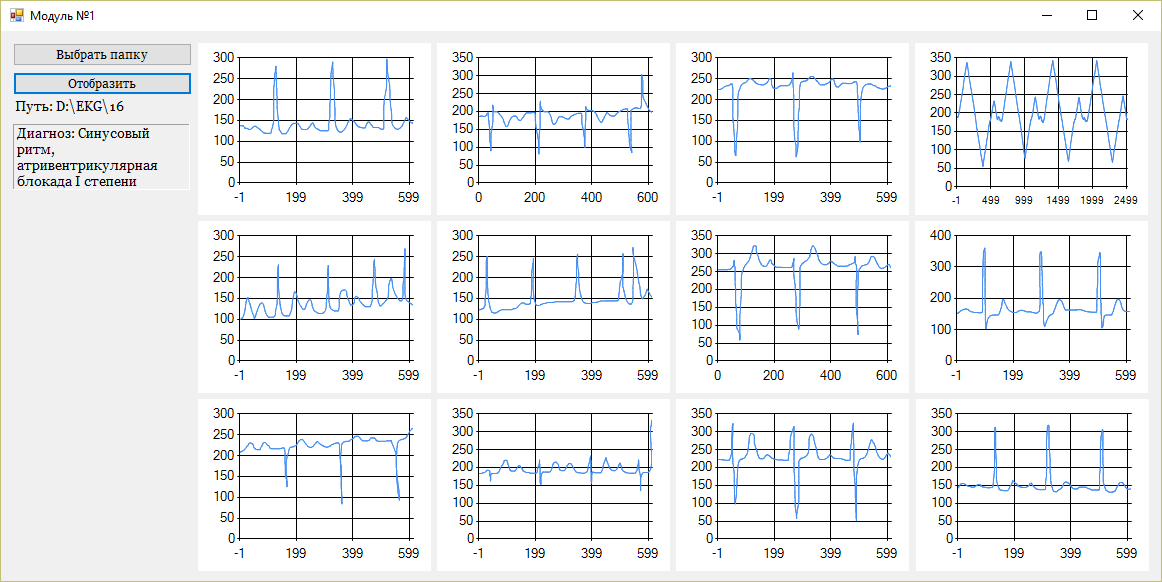


Рисунок 3.8 - Отображение данных

## 3.1.2 Модуль выполнения фильтрации

Работа программы для выполнения фильтрации представлена ниже.

При запуске программы, открывается начальное окно, представленное на рисунке 3.9.

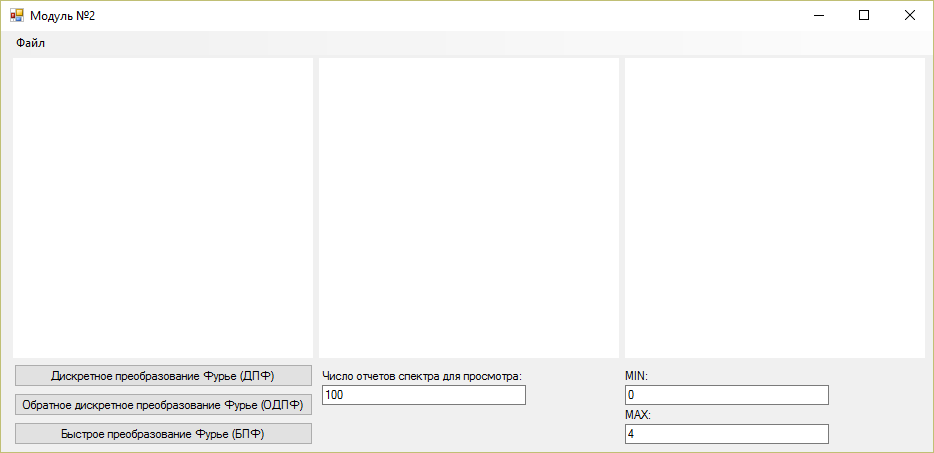


Рисунок 3.9 - Начальное окно фильтрующей программы

Для начала фильтрации необходимо выбрать файл с данными, которые необходимо обработать (рисунок 3.10). Для этого надо нажать «Файл» → «Открыть» и, перейдя в нужный каталог, выбрать файл.

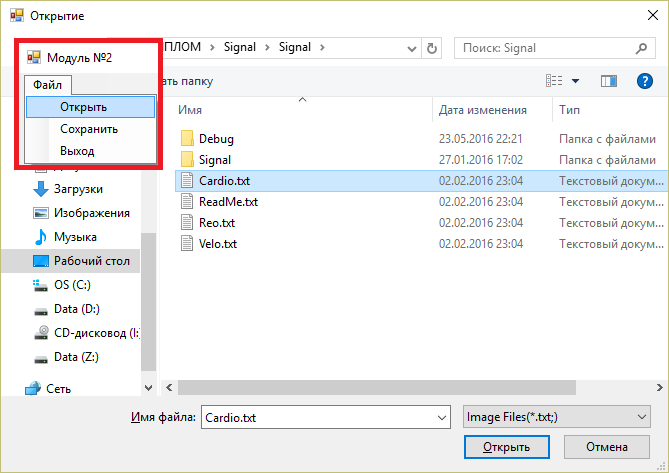


Рисунок 3.10 - Выбор файла для фильтрации

После проделанных операций в элементе chart1, в виде графика, будут отображены данные, содержащиеся в этом файле. Вертикальная ось отображает значение каждой конкретной точки отсчета, а горизонтальная - время, в течение которого регистрировался данный сигнал (рисунок 3.11).



Рисунок 3.11 - Отображение данных

После нажатия на кнопку «Дискретное преобразование Фурье (ДПФ)», будет выполнено прямое дискретное преобразование Фурье. Полученные данные будут отображены на графике chart2. Здесь вертикальная ось содержит значение амплитудного спектра, который рассчитывается по формуле (2.2), а горизонтальная - значения частот (рисунок 3.12).

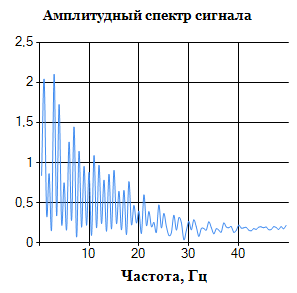


Рисунок 3.12 - Результат дискретного преобразования Фурье

Далее, для выполнения фильтрации, необходимо выполнить обратное дискретное преобразование Фурье, указав диапазон частот, которые являются помехами, подвергнув их удалению. Так, например, для кардиосигналов, помехами являются частоты свыше 35 Гц, а для реосигналов - свыше 22 Гц.

Отфильтрованный кардиосигнал представлен на рисунке 3.13.



Рисунок 3.13 - Кардиосигнал после фильтрации

После выполнения фильтрации полученный сигнал можно подавать на вход модуля распознавания параметров ЭКГ.

## 3.1.3 Модуль выделения параметров ЭКГ

Модуль выделения параметров выполняет по заданному алгоритму выделение и расчет всех основных параметров ЭКГ. Работа данного модуля описана ниже.

При запуске программы появляется окно (рисунок 3.14).

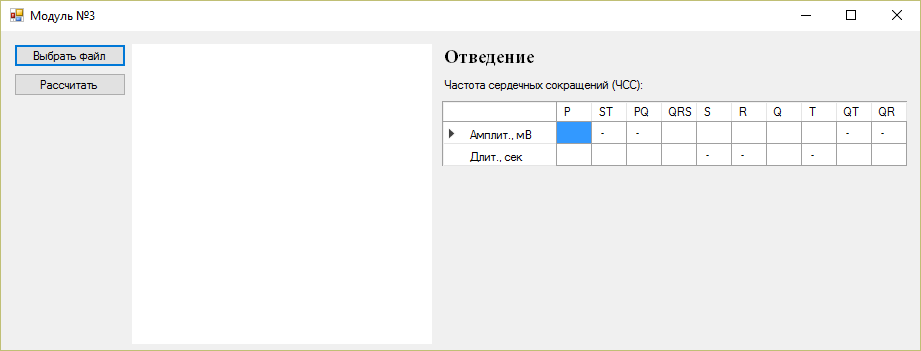


Рисунок 3.14 - Начальное окно программы выделения параметров ЭКГ

Для того, чтобы выбрать сигнал, параметры которого необходимо рассчитать необходимо нажать кнопку «Выбрать файл». Откроется окно, представленное на рисунке 3.15.

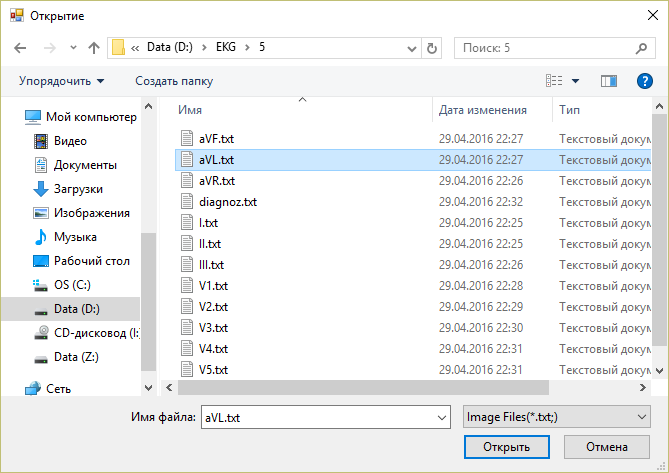


Рисунок 3.15 - Окно выбора файла

После выбора файла и нажатия на кнопку «Рассчитать» программа выполняет все необходимы расчеты и выводит результат в виде отображения QRS-комплекса на графике, а также в виде числовых параметров в таблицу.

Также будет выведено название отведения, комплекс которого был рассчитан, и частота сердечных сокращений. Результат работы программы представлен на рисунке 3.16.

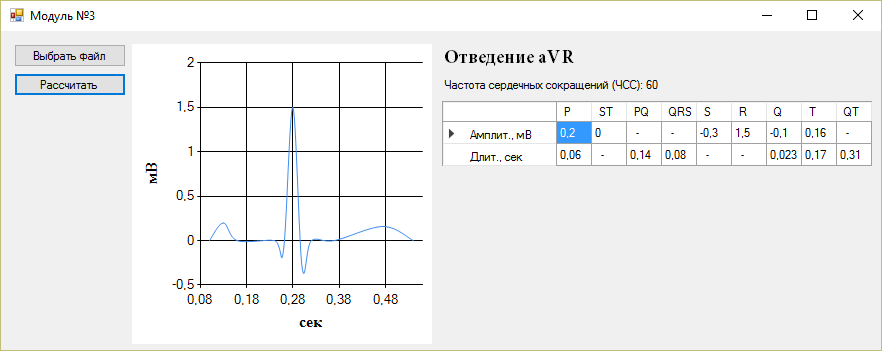


Рисунок 3.16 - Вывод параметров ЭКГ

## 3.1.4 Модуль работы с нейронной сетью

При запуске четвертого модуля программного комплекса открывается окно выбора режима работы (рисунок 3.17). Здесь можно выбрать режим просмотра данных, который позволяет просмотреть данные сигнала каждого отведения, перелистывая их путем нажатия кнопки, либо параметры, которые выводятся в виде таблицы для каждого отведения.

Изначально необходимо выбрать файл, вектора которого будем обрабатывать. А затем выполнять одно из 3 возможных действий.

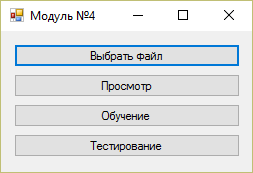


Рисунок 3.17 - Главное окно модуля нейронной сети

При просмотре данных можно выбрать режим отображения сигналов в виде графика или режим параметров в виде таблицы (рисунок 3.18).

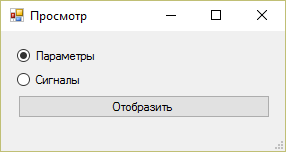


Рисунок 3.18 - Выбор режима отображения

Результат просмотра данных заболевания в виде графика одного из отведений представлен на рисунке 3.19.

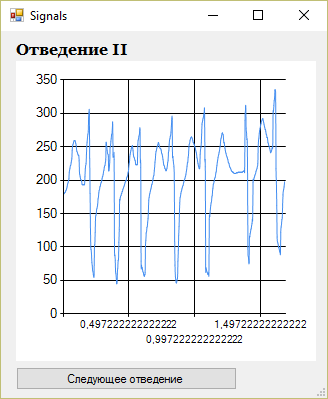
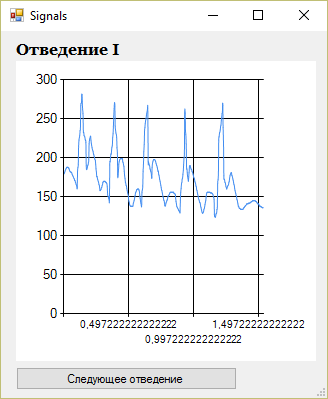


Рисунок 3.19 - Отсчеты кардиосигнала I и II стандартных отведений

Результат просмотра данных заболевания в таблицы параметров одного из отведений представлен на рисунке 3.20.

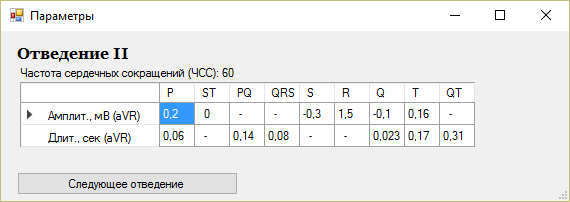


Рисунок 3.20 – Отображение таблицы параметров отведения I

Режим обучения нейронной сети предполагает обучение сети на основе входного файла обучающей выборки.

Процесс обучения сети по дискретным отсчетам кардиосигнала представлен на рисунке 3.21.

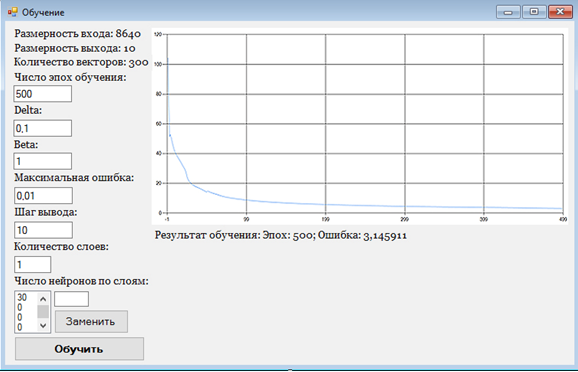


Рисунок 3.21 – Обучение сети по дискретным отсчетам сигнала

Процесс обучения сети по характеристическим параметрам кардиосигнала представлен на рисунке 3.22.

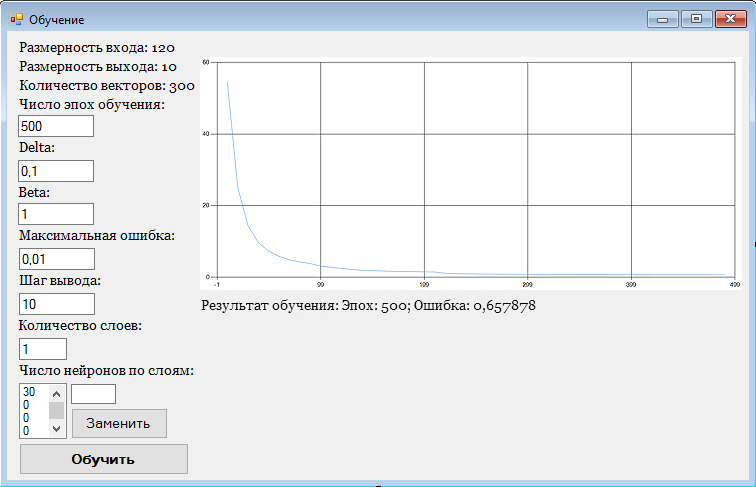


Рисунок 3.22 – Обучение сети по параметрам ЭКГ

После выполнения обучения сети можно выполнить тестирование работы распознавания. Для этого на вход нейронной сети необходимо подать файл, который содержит тестовую выборку. Указать номер вектора, который необходимо распознать и нажать на кнопку. В зависимости от того, содержит файл дискретные отсчеты или параметры, будет выведен результат в виде графиков или таблицы параметров для всех отведений. Результат распознавания представлен на рисунках 3.23 - 3.24.

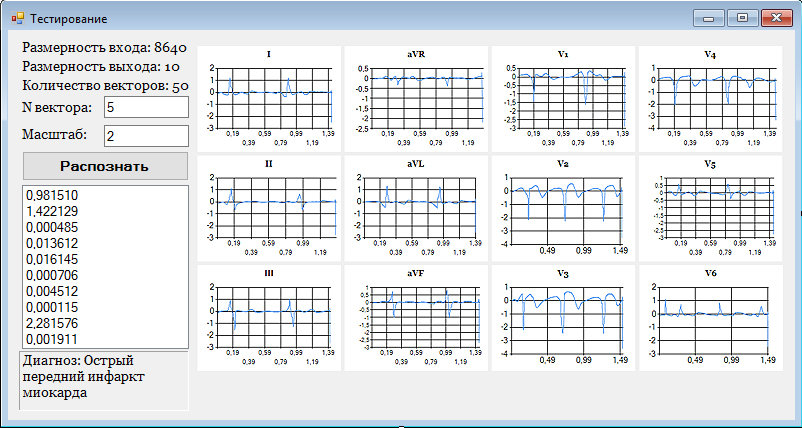


Рисунок 3.23 – Результат распознавания заболевания

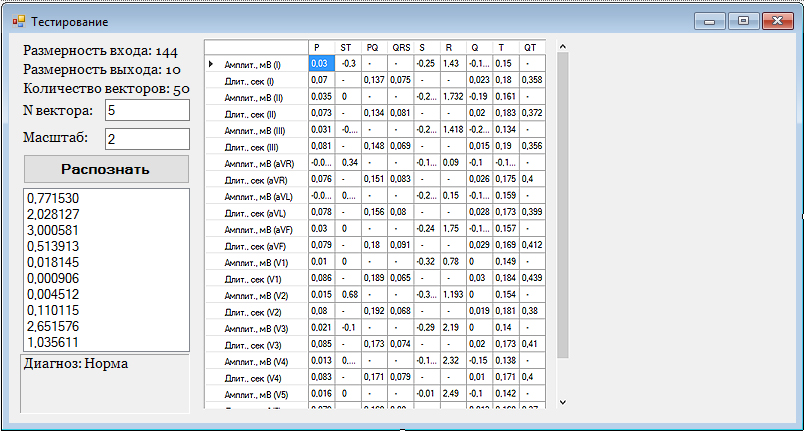


Рисунок 3.24 – Результат распознавания заболевания

**3.2 Выбор среды разработки**

Для разработки программного комплекса была выбрана интегрированная среда разработки Visual Studio 2013.

Visual Studio 2013 позволяет разработчикам работать в удобной среде и с легкостью создавать приложения под те нужды, которые потребуются заказчику. Главные преимущества Visual Studio 2013 - это возможность создавать приложения для любого вида платформ и типа устройств, а также современные инструменты и средства для разработки.

Для написания самой программы был использован объектно-ориентированный язык C# [14, 15].

Заключение

В ходе выполнения работы было выполнено расширение базы выявляемых заболеваний, разработан программный комплекс, состоящий из четырех модулей, каждый из которых выполняет определенную функцию, а также реализована возможность анализа ЭКГ не только по параметрам кардиосигнала, но и по его дискретным отсчетам 12 отведений.

В дальнейшем данная работа предполагает использование метода глубокого обучения нейронной сети, а также добавление модуля анализа реограмм.

В перспективах данной разработки лежит добавление модуля анализа реограмм и внедрение метода глубоко обучения нейронной сети.

Объем программного комплекса составляет примерно 64 Мб.

Список использованных источников

1) Воробьева, Е.Н. Применение нейронных сетей для выявления диагностических признаков сердечно-сосудистых заболеваний [Текст] / Е.Н. Воробьева, Н.С. Гарколь, А.Н. Тушев. – Барнаул: Изд-во Алт. ун-та, 2000. – 96 с.: ил.

2) Мухин, Н.А. Пропедевтика внутренних болезней: Учебник [Текст] / Н.А. Мухин, В.С. Монсеев. – М.:ГЭОТАР-МЕД, 2004. – 768 с.: ил.

3) Зутбинов, Ю.И. Азбука ЭКГ [Текст] / Ю.И. Зутбинов. –3-е изд., Ростов-на-Дону: изд-во «Феникс», 2003. – 160с.

4) Де Луна, А.Б. Руководство по клинической ЭКГ [Текст] / А.Б. Де Луна. – М.: Медицина, 1993. – 704 с.: ил.

5) Гарколь, Н.С. Применение искусственных нейронных сетей для диагностики заболеваний сердечно-сосудистой системы [Текст]: дис. на соиск. учен. степ. канд. техн. наук (05.11.17) / Наталья Станиславовна Гарколь; Алт. гос. техн. ун. – Барнаул, 2002. – 176 с.: ил.

6) Хэмптон, Д.Р. Атлас ЭКГ. 150 клинических ситуаций [Текст] / Д.Р. Хэмптон. – М.: Изд-во Медицинская литература, 2008. – 300 с.

7) Сергиенко, А.Б. Цифровая обработка сигналов [Текст] / А.Б. Сергиенко. – СПб.: Питер, 2002. – 606 с.: ил.

8) Александров, В.А. Преобразование Фурье: Учеб. пособие [Текст] / В.А. Александров. – Н.: НГУ, 2002. – 62 с.

9) Кардиомониторы. Аппаратура непрерывного контроля ЭКГ [Текст]: Учеб. пособие для вузов / ред. А.Л. Барановский, А.П. Немирко. – М.: Радио и связь, 1993. –248 с.

10) Витрук, С.К. Пособие по функциональным методам исследования сердечно-сосудистой системы [Текст] / С.К. Витрук. – К.: Здоровье, 1990. –220 с.

11) Нагин, В.А. Выделение QRS-комплексов в компьютерных ЭКГ-системах [Текст] / В.А. Нагин, И.В. Потапов, С.В. Селищев. – М., 2000. – Ч. 1. – 121 с.

12) Хайкин, С. Нейронный сети: полный курс [Текст] / С. Хайкин. –, 2-е изд., пер. с англ. – М.: Вильямс, 2006. – 1104 с.: ил.

13) Круглов, В.В Искусственные нейронные сети. Теория и практика [Текст] / В.В. Круглов, В. В. Борисов. – 2-е изд. – М.: Горячая линия-Телеком, 2002. – 382 с.

14) Лабор, В.В. Си Шарп: Создание приложений для Windows [Текст] / В.В. Лабор. – Мн.: Харвест, 2003. – 384 с.

15) Шилдт, Герберт Полный справочник по C# [Текст] / Герберт Шилдт. – М.: Изд. дом «Вильямс», 2004. – 752 с.: ил.

# Приложение А

**Задание на выполнение бакалаврской работы**

Министерство образования и науки Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Алтайский государственный технический университет им. И.И. Ползунова»

**УТВЕРЖДАЮ**

Заведующий кафедрой ИВТиИБ

А.Г. Якунин

(подпись) (и.о. фамилия)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_2016 г.

**ЗАДАНИЕ № 17**

**НА ВЫПОЛНЕНИЕ БАКАЛАВРСКОЙ РАБОТЫ**

|  |  |  |
| --- | --- | --- |
| По направлению | | *09.03.01 «Информатика и вычислительная техника»* |
| Студенке группы | | *ИВТ-22 Смирновой Валерии Вячеславовне* |
|  | |  |
| Тема: | *Разработка программного комплекса анализа нейронной сетью* | |
|  | *реографических и кардиографических сигналов* | |

|  |  |
| --- | --- |
| Утверждено приказом ректора от | *30.03.2016 № Л - 902* |
| Срок исполнения проекта |  |
| Задание приняла к исполнению | *В.В. Смирнова* |

*подпись и.о. фамилия*

БАРНАУЛ 2016

1 Исходные Данные

Задание на выполнение, ресурсы Internet, справочная информация, нормативные документы, ГОСТы.

2 Содержание разделов БР

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование разделов работы и их содержание | Трудоемкость, % | Срок  выполнения | Консультант  (Ф.И.О., подпись) |
| 1 Описание предметной области | 25 | 07.06.2016 | Тушев А.Н. |
| 2 Проектирование системы и выбор средств реализации | 25 | 15.06.2016 | Тушев А.Н. |
| 3 Описание программного обеспечения | 50 | 24.06.2016 | Тушев А.Н. |

3 Научно – библиографический поиск

3.1 По справочным материалам о языке программирования C#, книгам по обработке цифровых сигналов, а также по медицинской литературе об анализе ЭКГ, провести анализ существующих систем и разработать алгоритмы решения для создания собственного программного комплекса.

3.2 По нормативной литературе просмотреть указатели государственных и отраслевых стандартов за последний год.

Руководитель работы Тушев А.Н.

подпись фамилия, имя, отчество

# Приложение Б

**Руководство программиста по установке разработанного ПО**

Для работы с комплексом по обработке и анализу электрокардиограмм необходимо иметь приложения всех четырех модулей, с каждым из которых осуществлять работу в следующем порядке:

1. выполнить преобразование графических данных в числовой формат:
2. выбрать изображение кардиосигнала;
3. указать номер отведения;
4. указать папку для сохранения файла;
5. указать диагноз заболевания;
6. выполнить считывание координат положения курсора мыши при движении его по контуру сигнала;
7. нажать кнопку записи файла;
8. выполнить фильтрацию сигналов:
9. выбрать файл для фильтрации;
10. выполнить дискретное или быстрое преобразование Фурье, нажав на соответствующую кнопку;
11. выполнить обратное дискретное преобразование Фурье путем нажатия на кнопку. В результате этого действия обработанные данные запишутся в соответствующий файл в каталоге отфильтрованных сигналов;
12. выполнить выделение параметров электрокардиограммы из набора дискретных отсчетов:
13. Выбрать файл, содержащий дискретный набор данных;
14. Нажать на кнопку «Рассчитать», в результате чего числовые параметры сохранятся в текстовый файл, а также отобразятся в виде таблицы и графика в окне приложения;
15. выполнить обработку данных с использованием нейронной сети:
16. выбрать файл обучающей выборки;
17. указать необходимые параметры для обучения;
18. выполнить обучение сети, нажав на соответствующую кнопку;
19. перейти в раздел тестирования работы обученной сети;
20. выбрать файл тестовой выборки;
21. указать номер вектора для распознавания;
22. нажать на кнопку «Распознать».

# Приложение В

**Листинг разработанного программного обеспечения**

Код программы модуля распознавания данных:

Form1:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace pril1{

public partial class Form1 : Form {

int flag = 0;

Image image; //Bitmap для открываемого изображения

static int R = 1; // размерность массива

int[] X = new int[R]; // координаты щелчка мыши

double[] Y = new double[R];

List<int> temp; // список для Х

List<double> temp1; // список для Y

int i = 0; // счетчик

int N = 0; // количество отсчетов

public Form1() {

InitializeComponent();

}

// загрузка изображения в форму

private void button1\_Click(object sender, EventArgs e) {

i = 0;

R = 1;

// очищение массива Х

temp = X.ToList();

for (int k = 0; k < X.Length; k++ )

temp.RemoveAt(0);

X = temp.ToArray();

// очищение массива Y

temp1 = Y.ToList();

for (int k = 0; k < Y.Length; k++)

temp1.RemoveAt(0);

Y = temp1.ToArray();

// выбор файла

OpenFileDialog open\_dialog = new OpenFileDialog();

open\_dialog.Filter =

"Image Files(\*.BMP;\*.JPG;\*.GIF;\*.PNG)|\*.BMP;\*.JPG;\*.GIF;\*.PNG|All files (\*.\*)|\*.\*";

if (open\_dialog.ShowDialog() == DialogResult.OK) //если в окне была нажата кнопка "ОК"

{

try {

Bitmap image1;

image = new Bitmap(open\_dialog.FileName); // файл для загрузки

Graphics Gr1 = Graphics.FromImage(image); // создаем новый объект из рисунка

image1 = new Bitmap(pictureBox1.Width, pictureBox1.Height, Gr1);

Graphics Gr2 = Graphics.FromImage(image1);

Rectangle compressionRectangle = new Rectangle(0, 0, pictureBox1.Width, pictureBox1.Height);

Gr2.DrawImage(image, compressionRectangle);

pictureBox1.Image = image1; //назначили второй рисунок компоненте

pictureBox1.Size = image1.Size; //поставили размер компоненты по размерам нового рисунка

pictureBox1.Invalidate(); }

catch {

DialogResult rezult = MessageBox.Show("Невозможно открыть выбранный файл",

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

} } }

// нажатие мыши на элементе

private void pictureBox1\_MouseClick(object sender, MouseEventArgs e) {

if (flag == 0)

flag = 1;

else if (flag == 1) flag = 0; }

// записать файл

private void button2\_Click(object sender, EventArgs e) {

string fileName = ""; //имя файла для сохранения параметров

string fileName1 = "diagnoz.txt"; // имя файла для сохранения диагноза

string folderName = @"D:\EKG"; // путь к папке верхнего уровня

string pathString = System.IO.Path.Combine(folderName, textBox1.Text); // путь к подпапке

string pathString1 = System.IO.Path.Combine(folderName, textBox1.Text); // путь к подпапке

System.IO.Directory.CreateDirectory(pathString); // создание указанного пути

System.IO.Directory.CreateDirectory(pathString1); // создание указанного пути

if (radioButton1.Checked) {

fileName = "I.txt";

}

if (radioButton2.Checked) {

fileName = "II.txt";

}

if (radioButton3.Checked) {

fileName = "III.txt";

}

if (radioButton4.Checked) {

fileName = "aVR.txt";

}

if (radioButton5.Checked) {

fileName = "aVL.txt";

}

if (radioButton6.Checked) {

fileName = "aVF.txt";

}

if (radioButton7.Checked) {

fileName = "V1.txt";

}

if (radioButton8.Checked) {

fileName = "V2.txt";

}

if (radioButton9.Checked) {

fileName = "V3.txt";

}

if (radioButton10.Checked) {

fileName = "V4.txt";

}

if (radioButton11.Checked) {

fileName = "V5.txt";

}

if (radioButton12.Checked) {

fileName = "V6.txt";

}

pathString = System.IO.Path.Combine(pathString, fileName);

pathString1 = System.IO.Path.Combine(pathString1, fileName1);

// если файл существует,

if (System.IO.File.Exists(pathString))

System.IO.File.Delete(pathString);

//{

System.IO.StreamWriter textFile = new System.IO.StreamWriter(pathString);

System.IO.StreamWriter textFile1 = new System.IO.StreamWriter(pathString1);

//}

// вывод массивов в файл

N = 1;

textFile.WriteLine(Convert.ToString(/\*X[0] + " " + \*/Y[0])); // запись в файл значения

for (int i = 1; i < X.Length; i++) {

if (X[i] > X[i - 1]) {

textFile.WriteLine(Convert.ToString(/\*X[i] + " " + \*/Y[i])); // запись в файл значения

N++; } }

textFile.Close();

// запись файла с диагнозом

textFile1.WriteLine(textBox2.Text);

textFile1.Close(); }

// Движение мыши по элементу

private void pictureBox1\_MouseMove(object sender, MouseEventArgs e) {

// если флаг равен 1, то запись идет, если 0, то запись останавливается

if (flag == 1) {

int x = e.X; // считывание координаты X

double y = e.Y; // считывание координаты Y

Array.Resize<int>(ref X, R); // увеличение размера массива

Array.Resize<double>(ref Y, R);

Console.WriteLine(Convert.ToString(x) + " " + Convert.ToString(y));

X[i] = x;

Y[i] = pictureBox1.Height - y;

i++;

R++; } } }}

Form2:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace pril1{

public partial class Form2 : Form {

String fileName = "";

int[] X = new int[10000]; // координаты щелчка мыши

double[,] Y = new double[13,100000];

double[] T = new double[10000];

string[] files;

//string lineFile = "";

int []sizeFile = new int[13]; // размерность файла

public Form2() {

InitializeComponent(); }

private void button1\_Click(object sender, EventArgs e) {

FolderBrowserDialog FBD = new FolderBrowserDialog();

if (FBD.ShowDialog() == DialogResult.OK) {

fileName = FBD.SelectedPath; }

files = Directory.GetFiles(fileName, "\*.txt"); //получение всех вложенных файлов

label1.Text = "Путь: " + fileName; }

private void button2\_Click(object sender, EventArgs e) {

chart1.Series[0].Points.Clear(); // очищение графика

chart2.Series[0].Points.Clear();

chart3.Series[0].Points.Clear();

chart4.Series[0].Points.Clear();

chart5.Series[0].Points.Clear();

chart6.Series[0].Points.Clear();

chart7.Series[0].Points.Clear();

chart8.Series[0].Points.Clear();

chart9.Series[0].Points.Clear();

chart10.Series[0].Points.Clear();

chart11.Series[0].Points.Clear();

chart12.Series[0].Points.Clear();

// считывание всех файлов по порядку

for (int i = 0; i < files.Length; i++) {

StreamReader fs = new StreamReader(files[i]);

string line = "";

int k = 0;

while (true) {

// Читаем строку из файла во временную переменную.

line = fs.ReadLine(); // считывание строки

if (line == null) break;

if (files[i] != fileName + "\\diagnoz.txt") {

// string[] words = line.Split(' '); // загоняем строку в массив

// X[k] = Convert.ToInt32(words[0]);

Y[i,k] = Convert.ToDouble(line); }

if (files[i] == fileName + "\\diagnoz.txt") {

label2.Text = "Диагноз: " + line; }

k++; }

sizeFile[i] = k; // размерность файла равна количеству записанных значений

}

for (int i = 0; i < files.Length; i++) {

int Ch = 360; // частота дискретизации

double x, dx;

x = 0;

dx = 1.0 / Ch; // вычисляем шаг в зависимости от частоты (переводим в секунды)

for (int j = 0; j < sizeFile[i]; j++) {

T[j] = x;

T[j] = Math.Round(T[j], 2);

x += dx; }

int k = 0;

// Если достигнут конец файла, прерываем считывание

if (files[i] == fileName + "\\I.txt") {

while (k < sizeFile[i]) {

chart1.Series[0].Points.AddXY(T[k], Y[i, k]);

k++; } }

if (files[i] == fileName + "\\aVR.txt") {

while (k < sizeFile[i]){

chart2.Series[0].Points.AddXY(T[k], Y[i, k]);

k++; } }

if (files[i] == fileName + "\\V1.txt"){

while (k < sizeFile[i]) {

chart3.Series[0].Points.AddXY(T[k], Y[i, k]);

k++; } }

if (files[i] == fileName + "\\V4.txt") {

while (k < sizeFile[i]) {

chart4.Series[0].Points.AddXY(T[k], Y[i, k]);

k++; } }

if (files[i] == fileName + "\\II.txt") {

while (k < sizeFile[i]) {

chart5.Series[0].Points.AddXY(T[k], Y[i, k]);

k++; } }

if (files[i] == fileName + "\\aVL.txt") {

while (k < sizeFile[i]) {

chart6.Series[0].Points.AddXY(T[k], Y[i, k]);

k++; } }

if (files[i] == fileName + "\\V2.txt") {

while (k < sizeFile[i]) {

chart7.Series[0].Points.AddXY(T[k], Y[i, k]);

k++; } }

if (files[i] == fileName + "\\V5.txt") {

while (k < sizeFile[i]) {

chart8.Series[0].Points.AddXY(T[k], Y[i, k]);

k++; } }

if (files[i] == fileName + "\\III.txt"){

while (k < sizeFile[i]){

chart9.Series[0].Points.AddXY(T[k], Y[i, k]);

k++; }}

if (files[i] == fileName + "\\aVF.txt") {

while (k < sizeFile[i]) {

chart10.Series[0].Points.AddXY(T[k], Y[i, k]);

k++; }}

if (files[i] == fileName + "\\V3.txt") {

while (k < sizeFile[i]) {

chart11.Series[0].Points.AddXY(T[k], Y[i, k]);

k++; } }

if (files[i] == fileName + "\\V6.txt"){

while (k < sizeFile[i]) {

chart12.Series[0].Points.AddXY(T[k], Y[i, k]);

k++; } } } }}}

Form3:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace pril1{

public partial class Form3 : Form {

public Form3() {

InitializeComponent(); }

private void button1\_Click(object sender, EventArgs e) {

Form1 fm = new Form1();

fm.Show();}

private void button2\_Click(object sender, EventArgs e) {

Form2 fm = new Form2();

fm.Show(); } }}

Модуль фильтрации:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace pril2{

public partial class Form1 : Form {

public string MyFile = "";

public double[] Data = new double[50000]; // значения кардиограммы из файла

public double[] A = new double[50000];

public double[] B = new double[50000];

public double[] Amp = new double[50000];

public double[] T = new double[50000];

int N = 0;

public Form1() {

InitializeComponent();

chart1.ChartAreas[0].AxisY.Title = "Амплитуда, мм";

chart1.ChartAreas[0].AxisX.Title = "Время, сек";

chart1.ChartAreas[0].AxisY.TitleFont = new Font("Times New Roman", 12, FontStyle.Bold);

chart1.ChartAreas[0].AxisX.TitleFont = new Font("Times New Roman", 12, FontStyle.Bold);

//chart2.ChartAreas[0].AxisY.Title = "у";

chart2.ChartAreas[0].AxisX.Title = "Частота, Гц";

//chart2.ChartAreas[0].AxisY.TitleFont = new Font("Times New Roman", 12, FontStyle.Bold);

chart2.ChartAreas[0].AxisX.TitleFont = new Font("Times New Roman", 12, FontStyle.Bold);

chart3.ChartAreas[0].AxisY.Title = "Ампл., мВ";

chart3.ChartAreas[0].AxisX.Title = "Время, сек";

chart3.ChartAreas[0].AxisY.TitleFont = new Font("Times New Roman", 12, FontStyle.Bold);

chart3.ChartAreas[0].AxisX.TitleFont = new Font("Times New Roman", 12, FontStyle.Bold); }

// пункт меню "Выход"

private void выходToolStripMenuItem\_Click(object sender, EventArgs e) {

Close(); }

// пункт меню "Открыть"

private void открытьToolStripMenuItem\_Click(object sender, EventArgs e) {

string fileName = ""; // новая строка

OpenFileDialog open\_dialog = new OpenFileDialog();

open\_dialog.Filter =

"Image Files(\*.txt;)|\*.txt;|All files (\*.\*)|\*.\*"; //формат загружаемого файла

if (open\_dialog.ShowDialog() == DialogResult.OK) //если в окне была нажата кнопка "ОК" {

try {

fileName = open\_dialog.FileName; // имя выбранного файла

StreamReader fs = new StreamReader(fileName);

string lines = "";

int i = 0;

while (true) {

N++; // количество отчетов

// Читаем строку из файла во временную переменную

lines = fs.ReadLine();

// Если достигнут конец файла, прерываем считывание

if (lines == null) break;

// Пишем считанную строку в итоговую переменную

Data[i] = Double.Parse(lines); // заполняем массив Y значениями из выбранного файла

int Ch = 360; // частота дискретизации

double x, dx;

x = 0;

dx = 1.0 / Ch; // вычисляем шаг в зависимости от частоты (переводим в секунды)

for (int j = 0; j < N; j++) {

T[i] = x;

x += dx; }

// прорисовываем график время от считанных данных

chart1.Series[0].Points.AddXY(T[i], Data[i]);

i++; } }

catch {

DialogResult rezult = MessageBox.Show("Невозможно открыть выбранный файл",

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error); } }

MyFile = fileName; }

// ДИСКРЕТНОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ

private void button1\_Click(object sender, EventArgs e) {

int i;

double x,h;

x=0.0; // Значение по оси X

h=1.0/(N/360.0); // Шаг по частоте

string s = ""; // Строка для считывания данных

s=textBox3.Text; // Число отчетов спектра для просмотра

DPF(0); // Дискретное преобразование Фурье

chart2.Series[0].Points.Clear(); // Очищение графика перед перерисовкой

int k; // Переменная для записи значения

k = Convert.ToInt32(s); // Число отчетов спектра для просмотра

// Пересчет в Гц, 1 сек = 360 отчетов

for(i = 0; i < N; i++) {

T[i] = x;

x += h; }

for(i = 1; i < k; i++) {

chart2.Series[0].Points.AddXY(T[i], Amp[i]); // Прорисовка данных (по X - Герцы, по Y - Амплитудный спектр) } }

// КНОПКА "БПФ"

private void button3\_Click(object sender, EventArgs e) {

int i;

double x, h;

string s = "";

s = textBox3.Text;

BPF(0);

chart2.Series[0].Points.Clear();

int k;

k = Convert.ToInt32(s);

x = 0.0;

h = 1.0 / (N / 360.0);

// пересчет в Гц 1 сек = 360 Гц

for (i = 0; i < N; i++) {

T[i] = x;

x += h; }

for (i = 0; i < k; i++){

chart2.Series[0].Points.AddXY(T[i], Amp[i]); }}

// БЫСТРОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ

public void BPF(int t) {

int i, M, k, j, N2, s, m;

double Sa;

double WmR, WmI;

double WR, WI;

double tr, ti, ur, ui;

double x, y;

M = 1;

N2 = N;

while (N2 > 1) {

N2 = N2 / 2; // деление на группы

if (N2 > 1)

M++; // количество делений

}

if (t == 0) {

for (i = 0; i < N; i++) {

A[i] = Data[i];

B[i] = 0; } }

for (i = 0; i < N; i++) {

k = reverse(i, M); // номер отчета, котрый является обратным по битам

// если он стоит после текущего отчета, то меняем их местами

if (i < k) {

Sa = A[i];

A[i] = A[k];

A[k] = Sa;

Sa = B[i];

B[i] = B[k];

B[k] = Sa; } }

// иттерационный алгоритм

for (s = 1; s <= M; s++) {

m = 1;

for (j = 1; j <= s; j++) {

m = m \* 2; }

WmR = Math.Cos(2 \* 3.1415926 / m);

WmI = Math.Sin(2 \* 3.1415926 / m);

for (k = 0; k < N; k += m) {

WR = 1;

WI = 0;

for (j = 0; j <= m / 2 - 1; j++) {

tr = A[k + j + m / 2] \* WR - B[k + j + m / 2] \* WI;

ti = A[k + j + m / 2] \* WI + B[k + j + m / 2] \* WR;

ur = A[k + j];

ui = B[k + j];

// бабочка

A[k + j] = tr + ur;

B[k + j] = ti + ui;

A[k + j + m / 2] = ur - tr;

B[k + j + m / 2] = ui - ti;

x = WR \* WmR - WI \* WmI;

y = WR \* WmI + WI \* WmR;

WR = x;

WI = y; } }}

// Делим на N

if (t == 0) {

for (k = 0; k < N; k++) {

A[k] = A[k] / N;

B[k] = B[k] / N; } }

// Рассчитываем амлитудный спектр

if (t == 0) {

for (i = 0; i < N; i++) {

Amp[i] = Math.Sqrt(A[i] \* A[i] + B[i] \* B[i]); } }}

// ДИСКРЕТНОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ

public void DPF(int t) {

int i, k;

double P = 3.1415926; // Число ПИ

for (k = 0; k < N; k++) {

if (t == 0) {

A[k] = 0; // Начальные значения равны 0

B[k] = 0; }

if (t == 1) {

Data[k] = 0; }

for (i = 0; i < N; i++) {

if (t == 0) {

A[k] += Data[i] \* Math.Cos(2.0 \* P \* k \* i / N);

B[k] += Data[i] \* Math.Sin(2.0 \* P \* k \* i / N); }

if (t == 1) {

Data[k] += A[i] \* Math.Cos(2.0 \* P \* k \* i / N) + B[i] \* Math.Sin(2.0 \* P \* k \* i / N);

} }

if (t == 0) {

A[k] = A[k] / N;

B[k] = B[k] / N; } }

if (t == 0) {

for (i = 0; i < N; i++) {

Amp[i] = Math.Sqrt(A[i] \* A[i] + B[i] \* B[i]); } } }

public int reverse(int I, int T) {

int R;

int Shift;

int k;

R = 0;

Shift = T - 1;

int Low, High;

Low = 1;

High = 1 << Shift;

while (Shift >= 0) {

k = ((I & Low) << Shift) | ((I & High) >> Shift);

R = R | k;

Shift -= 2;

Low = Low << 1;

High = High >> 1; }

return R; }

// Обратное преобразование Фурье

private void button2\_Click(object sender, EventArgs e) {

int i,m1,m2;

string s = "";

chart3.Series[0].Points.Clear(); // Очищение графика

s=textBox1.Text; // Минимальная частота

m1 = Convert.ToInt32(s);

s=textBox2.Text; // Максимальная частота

m2 = Convert.ToInt32(s);

// Если минимальная частота меньше максимальной, то зануляем данные на этих частотах

if(m1 < m2) {

for(i = m1; i < m2; i++) {

A[i]=0;

B[i]=0; }

for(i = N-m2; i < N-m1; i++) {

A[i]=0;

B[i]=0; } }

DPF(1);

// Запись в файл

string fileName1 = ""; //имя файла для сохранения параметров

string folderName = @"D:\filtr"; // путь к папке верхнего уровня

string pathString = System.IO.Path.Combine(folderName, "4"); // путь к подпапке

System.IO.Directory.CreateDirectory(pathString); // создание указанного пути

if (MyFile == "D:\\EKG\\4\\I.txt")

fileName1 = "I.txt";

if (MyFile == "D:\\EKG\\4\\II.txt")

fileName1 = "II.txt";

if (MyFile == "D:\\EKG\\4\\III.txt")

fileName1 = "III.txt";

if (MyFile == "D:\\EKG\\4\\aVR.txt")

fileName1 = "aVR.txt";

if (MyFile == "D:\\EKG\\4\\aVL.txt")

fileName1 = "aVL.txt";

if (MyFile == "D:\\EKG\\4\\aVF.txt")

fileName1 = "aVF.txt";

if (MyFile == "D:\\EKG\\4\\V1.txt")

fileName1 = "V1.txt";

if (MyFile == "D:\\EKG\\4\\V2.txt")

fileName1 = "V2.txt";

if (MyFile == "D:\\EKG\\4\\V3.txt")

fileName1 = "V3.txt";

if (MyFile == "D:\\EKG\\4\\V4.txt")

fileName1 = "V4.txt";

if (MyFile == "D:\\EKG\\4\\V5.txt")

fileName1 = "V5.txt";

if (MyFile == "D:\\EKG\\4\\V6.txt")

fileName1 = "V6.txt";

pathString = System.IO.Path.Combine(pathString, fileName1);

System.IO.StreamWriter textFile = new System.IO.StreamWriter(pathString);

for (int p = 0; p < N; p++) {

textFile.WriteLine(Convert.ToString(Data[p])); // запись в файл значения

}

textFile.Close();

// Пересчет в секундах

double x,h;

x=0;

h=1/360.0;

for(i = 0 ; i < N; i++) {

T[i]=x;

x+=h; }

for(i=0; i < N; i++) {

chart3.Series[0].Points.AddXY(T[i],Data[i]/10); }}}}

Модуль реализации нейронной сети:

Ann.h

#pragma once

#define MaxNumber 60000

#define MaxOutput 50

#define MaxInputX 120

#define MaxInputY 100

#define MaxLayers 5

#define MaxNeurons 100

class Ann{

public:

Ann(void);

~Ann(void);

char fName[300];

void Load();

int NinputX,NinputY,Noutput,Number,Ninput;

unsigned char \*Data;

int Classes[MaxNumber];

int Type; // 0 - bit, 1 - grey

void Init();

int NumberLayers;

double W1[MaxInputX\*MaxInputY][MaxNeurons];// input -> first

double W2[MaxNeurons][MaxOutput]; // last -> output

double W[MaxLayers][MaxNeurons][MaxNeurons]; // internal

double B1[MaxNeurons]; // смещение на первом слое

double B2[MaxOutput]; // смещение на последнем слое

double B[MaxLayers][MaxNeurons]; // смещения на промежуточных слоях

double EB1[MaxNeurons]; // смещение на первом слое

double EB2[MaxOutput]; // смещение на последнем слое

double EB[MaxLayers][MaxNeurons]; // смещения на промежуточных слоях

int Epoch;

int Step;

double MaxErr;

double Er; // суммарная ошибка по выборке

double Eri; // ошибка по i элементу выборки

int neurons[MaxLayers]; // нейронов по слоям

double Delta;

double Beta;

bool Train();

double Inputs[MaxLayers][MaxNeurons];

double Outputs[MaxLayers][MaxNeurons];

double CurrentVector[ MaxInputX\* MaxInputY];

double Activate(double x);

double DifActivate(double x);

double CurrentOutput[MaxOutput];

double EoutX[MaxOutput]; //ошибка выходного слоя

double EoutY[MaxOutput];

double EW2[MaxNeurons][MaxOutput];

double EW[MaxLayers][MaxNeurons][MaxNeurons];

double EOutputs[MaxLayers][MaxNeurons];

double EInputs[MaxLayers][MaxNeurons];

double EW1[MaxInputX\*MaxInputY][MaxOutput];

void Forward();};

Ann.cpp

#include "StdAfx.h"

#include "Ann.h"

#include <fstream>

#include <stdio.h>

#include <ctime>

using namespace std;

Ann::Ann(void){}

Ann::~Ann(void){}

void Ann::Load(){

int i,j,y;

FILE \*fin;

int z,n;

unsigned char \*c;

fin=fopen(fName,"r");

fscanf(fin,"%d", &Type); // считывание первой переменной Type

if(Type<2) // если она меньше 2 {

fscanf(fin,"%d %d %d %d",&NinputX,&NinputY,&Noutput,&Number); Ninput=NinputX\*NinputY; // размерность матрицы

Data=new unsigned char[Number\*NinputX\*NinputY];

c=Data; // присваиваем указатели

n=0;

// проход по всем образам

for(i=0;i<Number;i++) {

// проход по всем элементам матрицы

for(j=0;j<Ninput;j++) {

y= fscanf(fin,"%d ",&z); // считывание элемента массива

\*c=z; // запись его кода

c++; }

fscanf(fin,"%d ",&z); // считывание наименования класса

Classes[i]=z; }

fclose(fin); }}

void Ann::Init(){

int i,j,k;

int m,m1,m2;

double u,uu;

neurons[NumberLayers+1]=Noutput;

srand((unsigned)time(0));

uu=-0.8;

for(i=0;i<Ninput;i++){

m1=neurons[1];

for(j=0;j<m1;j++){

u= (double)rand() / (RAND\_MAX + 1)\*2-1;

W1[i][j]=u; }}

for(i=1;i<NumberLayers;i++){

m1=neurons[i];

m2=neurons[i+1];

for(j=0;j<m1;j++){

for(k=0;k<m2;k++){

u= (double)rand() / (RAND\_MAX + 1)\*2-1;

W[i][j][k]=u; }}}

m1=neurons[NumberLayers];

m2=Noutput;

for(i=0;i<MaxNeurons;i++){

for(j=0;j<Noutput;j++){

u= (double)rand() / (RAND\_MAX + 1)\*2-1;

W2[i][j]=u; }}

m1=neurons[1];

for(j=0;j<m1;j++){

u= (double)rand() / (RAND\_MAX + 1)\*2-1;

B1[j]=u; }

for(i=2;i<=NumberLayers;i++)

{ m1=neurons[i];

for(k=0;k<m1;k++){

u= (double)rand() / (RAND\_MAX + 1)\*2-1;

B[i][k]=u; }}

m1=Noutput;

for(j=0;j<m1;j++){

u= (double)rand() / (RAND\_MAX + 1)\*2-1;

B2[j]=u; }}

double Ann::Activate(double x) {

// sigmoid

return 1.0/(1+exp(-Beta\*x)); }

double Ann::DifActivate(double x) {

return Beta\*x\*(1-x); }

bool Ann::Train(){

int k,i,j,m,n,q,p,z,d;

// std::ofstream outfile;

unsigned char cc;

int ns;

q=0;

ns=0;

while(ns<Step) {

Er=0.0;

for(m=0;m<Number;m++) {

if(Type<2) {

k=0;

for(q=0;q<NinputY;q++){

for(p=0;p<NinputX;p++) {

z=q\*NinputX+p;

z=z+m\*NinputX\*NinputY;

cc=Data[z];

CurrentVector[k]=(double)cc;

k++;} } }

// forward

Forward();

// Error

Eri=0.0;

for(k=0;k<Noutput;k++){

if(Classes[m]-1==k) {

Eri += (CurrentOutput[k]-1)\*(CurrentOutput[k]-1)/2; }

else{

Eri += (CurrentOutput[k])\*(CurrentOutput[k])/2; } }

Er+=Eri;

// backward

// EoutY

for(k=0;k<Noutput;k++) {

if(Classes[m]-1==k) {

EoutY[k]=CurrentOutput[k]-1; }

else {

EoutY[k]=CurrentOutput[k]; }}

// EoutX

for(k=0;k<Noutput;k++) {

EoutX[k]=EoutY[k]\*DifActivate(CurrentOutput[k]); }

// Ew посл. - выход

for(n=0;n<neurons[NumberLayers];n++){

for(k=0;k<Noutput;k++) {

EW2[n][k]=Outputs[NumberLayers][n]\*EoutX[k]; }}

for(k=0;k<Noutput;k++) {

EB2[k]=EoutX[k]; }

for(n=0;n<neurons[NumberLayers];n++){

EOutputs[NumberLayers][n]=0.0;

for(k=0;k<Noutput;k++) {

EOutputs[NumberLayers][n] += W2[n][k]\*EoutX[k]; }}

for(k=NumberLayers;k>1;k--){

for(n=0;n<neurons[k];n++) {

EInputs[k][n]=EOutputs[k][n]\*DifActivate(Outputs[k][n]); }

for(n=0;n<neurons[k];n++) {

for(d=0;d<neurons[k-1];d++){

EW[k-1][d][n]=EInputs[k][n]\*Outputs[k-1][d]; }}

for(n=0;n<neurons[k];n++) {

EB[k-1][n]=EInputs[k][n]; }

for(n=0;n<neurons[k-1];n++) {

EOutputs[k-1][n]=0.0;

for(d=0;d<neurons[k];d++){

EOutputs[k-1][n] += W[k-1][n][d]\*EInputs[k][d]; } } }

for(n=0;n<neurons[1];n++) {

EInputs[1][n]=EOutputs[1][n]\*DifActivate(Outputs[1][n]); }

for(n=0;n<neurons[1];n++) {

for(d=0;d<Ninput;d++) {

EW1[d][n]=EInputs[1][n]\*CurrentVector[d]; }}

for(n=0;n<neurons[1];n++) {

EB1[n]=EInputs[1][n]; }

// correct W

for(i=0;i<Ninput;i++){

for(j=0;j<neurons[1];j++) {

W1[i][j] -= Delta\*EW1[i][j]; }}

for(j=0;j<neurons[1];j++) {

B1[j] -= Delta\*EB1[j]; }

for(k=1;k<NumberLayers;k++){

for(i=0;i<neurons[k];i++){

for(j=0;j<neurons[k+1];j++){

W[k][i][j] -= Delta\*EW[k][i][j];

}}}

for(k=1;k<NumberLayers;k++){

for(j=0;j<neurons[k+1];j++){

B[k][j] -= Delta\*EB[k][j]; }}

for(i=0;i<Noutput;i++){

for(j=0;j<neurons[NumberLayers];j++){

W2[j][i] -= Delta\*EW2[j][i]; }}

for(j=0;j<neurons[NumberLayers];j++){

B2[i] -= Delta\*EB2[i]; }}

ns++; }

if(Er<MaxErr)

return true;

return false;}

void Ann::Forward(){

int m,n,k,d;

for(n=0;n<neurons[1];n++) // input -> 1 internal{

Inputs[1][n]=0.0;

for(d=0;d<Ninput;d++) {

Inputs[1][n] += W1[d][n]\*CurrentVector[d]; }

Inputs[1][n] += B1[n]; }

// output 1

for(n=0;n<neurons[1];n++){

Outputs[1][n]=Activate(Inputs[1][n]); }

for(k=2;k<=NumberLayers;k++){

for(n=0;n<neurons[k];n++) // k-1 -> k internal{

Inputs[k][n]=0.0;

for(d=0;d<neurons[k-1];d++) {

Inputs[k][n] += W[k-1][d][n]\*Outputs[k-1][d]; }

Inputs[k][n] += B[k][n]; }

for(n=0;n<neurons[k];n++){

Outputs[k][n]=Activate(Inputs[k][n]); }}

for(n=0;n<Noutput;n++){

CurrentOutput[n]=0;

for(d=0;d<neurons[NumberLayers];d++) {

CurrentOutput[n] += W2[d][n]\*Outputs[NumberLayers][d]; }

CurrentOutput[n] += B2[n]; }

for(n=0;n<Noutput;n++) {

CurrentOutput[n]=Activate(CurrentOutput[n]); }}

DataLook.h

#pragma once

#include "Ann.h"

namespace NeuralNets {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

public ref class DataLook : public System::Windows::Forms::Form{

public:

Ann \*ann;

Bitmap ^ bmp;

private: System::Windows::Forms::PictureBox^ pictureBox1;

public:

Graphics ^g;

int BegImage;

int NimagesX,NimagesY;

private: System::Windows::Forms::Label^ label1;

public:

private: System::Windows::Forms::TextBox^ textBox1;

public:

private: System::Windows::Forms::Button^ button1;

public:

DataLook(void) {

InitializeComponent();

ann=new Ann();}

protected:

~DataLook(){

if (components) {

delete components; }}

private:

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

void InitializeComponent(void) {

this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());

this->button1 = (gcnew System::Windows::Forms::Button());

this->label1 = (gcnew System::Windows::Forms::Label());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^ >(this->pictureBox1))->BeginInit();

this->SuspendLayout();

this->pictureBox1->Location = System::Drawing::Point(49, 26);

this->pictureBox1->Name = L"pictureBox1";

this->pictureBox1->Size = System::Drawing::Size(100, 50);

this->pictureBox1->TabIndex = 0;

this->pictureBox1->TabStop = false;

this->button1->Location = System::Drawing::Point(49, 191);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(75, 23);

this->button1->TabIndex = 1;

this->button1->Text = L"Next";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &DataLook::button1\_Click);

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(160, 191);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(134, 17);

this->label1->TabIndex = 2;

this->label1->Text = L"Ќачальный образ: ";

this->textBox1->Location = System::Drawing::Point(318, 191);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(106, 22);

this->textBox1->TabIndex = 3;

this->textBox1->Text = L"0";

this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(804, 255);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->label1);

this->Controls->Add(this->button1);

this->Controls->Add(this->pictureBox1);

this->Name = L"DataLook";

this->Text = L"DataLook";

this->Load += gcnew System::EventHandler(this, &DataLook::DataLook\_Load);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^ >(this->pictureBox1))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();}

#pragma endregion

private: System::Void DataLook\_Load(System::Object^ sender, System::EventArgs^ e)

{

int h,w;

int wp,hp,hl;

System::Drawing::Rectangle workingRectangle = Screen::PrimaryScreen->WorkingArea;

this->Size = System::Drawing::Size( workingRectangle.Width - 10, workingRectangle.Height - 10 );

this->Left = 0;

this->Top = 0;

h=workingRectangle.Height - 10 ;

w=workingRectangle.Width - 10 ;

wp=w-20;

hp=h-80;

hl=20;

// pictureBox1

pictureBox1->Top = 10;

pictureBox1->Left=0;

pictureBox1->Width = wp;

pictureBox1->Height = hp;

bmp = gcnew Bitmap(wp, hp);

pictureBox1->Image = bmp;

System::Drawing::SolidBrush^ br =

gcnew System::Drawing::SolidBrush(System::Drawing::Color::White);

g = Graphics::FromImage( bmp );

g->FillRectangle(br, 0, 0, wp, hp);

BegImage=0;

NimagesX=wp/ann->NinputX-3;

NimagesY=hp/(ann->NinputY+hl);

// button

button1->Top=hp+5;

button1->Left=10;

label1->Top=button1->Top;

label1->Left=button1->Left+button1->Width+10;

textBox1->Top=button1->Top;

textBox1->Left=label1->Left+label1->Width+10;

DrawImages(); }

void DrawImages(){

int i,j,m,z;

int x,y,hx,hy;

int p,q;

Color col;

unsigned char cc;

cc=0;

int hl=20;

hx=ann->NinputX; // количество элементов по горизонтали

hy=ann->NinputY; // количество элементов по вертикали

System::Drawing::SolidBrush^ br =

gcnew System::Drawing::SolidBrush(System::Drawing::Color::White);

g = Graphics::FromImage( bmp );

g->FillRectangle(br, 0, 0, NimagesX\*hx, NimagesY\*(hy+hl));

m=BegImage;

x=0;

y=0;

for(j=0;j<NimagesY;j++) {

for(i=0;i<NimagesX;i++) {

x=0;

y=0;

if(m<ann->Number) {

for(q=0;q<ann->NinputY;q++) {

for(p=0;p<ann->NinputX;p++){

z=q\*ann->NinputX+p;

z=z+m\*ann->NinputX\*ann->NinputY;

cc=ann->Data[z];

if(ann->Type==0) {

if(cc==0)

cc=255;

else

cc=0; col=Color::FromArgb(cc,cc,cc);

bmp->SetPixel(x+i\*ann->NinputX,y+j\*(ann->NinputY+hl),col); } if(ann->Type==1) {

col=Color::FromArgb(cc,cc,cc); bmp->SetPixel(x+i\*ann->NinputX,y+j\*(ann->NinputY+hl),col);} x++;}x=0; y++;} }m++;} }

pictureBox1->Image = bmp; }

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

String ^ s =gcnew String("");

s=textBox1->Text;

int c = System::Convert::ToInt32(s);

BegImage=c;

DrawImages();}};}

NetTrain.h

#pragma once

#include "Ann.h"

namespace NeuralNets {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::Threading;

public ref class NetTrain : public System::Windows::Forms::Form

{

public:

Ann \*ann;

NetTrain(void) {

InitializeComponent();

//

//TODO: добавьте код конструктора

//

ann=new Ann(); }

protected:

~NetTrain(){

if (components) {

delete components; }}

#pragma endregion

private: System::Void NetTrain\_Load(System::Object^ sender, System::EventArgs^ e) {

int h,w;

int wp,hp,hl;

System::Drawing::Rectangle workingRectangle = Screen::PrimaryScreen->WorkingArea;

this->Size = System::Drawing::Size( workingRectangle.Width - 10, workingRectangle.Height - 10 );

this->Left = 0;

this->Top = 0;

h=workingRectangle.Height - 10 ;

w=workingRectangle.Width - 10 ;

wp=w-20;

hp=h-80;

hl=20;

// chart1

chart1->Left=30;

chart1->Top=listBox1->Top+listBox1->Height+20;

chart1->Width=wp-100;

chart1->Height=hp-50-chart1->Top;

String ^ s =gcnew String("");

if(ann->Type==0 || ann->Type==1) {

s = (ann->NinputX\*ann->NinputY).ToString();

s="Размерность входа: "+s;

label1->Text=s;

s = (ann->Noutput).ToString();

s="Размерность выхода: "+s;

label2->Text=s;

s = (ann->Number).ToString();

s="Количество векторов: "+s;

label3->Text=s; }

label11->Top=chart1->Top+chart1->Height+20; }

private: System::Void listBox1\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

int k;

k=listBox1->SelectedIndex;

if(k>=0) {

textBox7->Text= listBox1->Items[k]->ToString(); // устанавливаем в текстовое поле выбранный элемент из списка }}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

int k;

k=listBox1->SelectedIndex; // индекс выбранного элемента

listBox1->Items[k]=textBox7->Text; // элемент в списке заменяем на новый введенный }

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

int m,ne,i,j,k;

bool pr;

double x;

String ^ s =gcnew String("");

s=textBox6->Text;

k = System::Convert::ToInt32(s);

ann->NumberLayers=k; // количество слоев

s=textBox1->Text;

k = System::Convert::ToInt32(s);

ann->Epoch=k; // количество эпох обучения

s=textBox5->Text;

k = System::Convert::ToInt32(s);

ann->Step=k; // шаг вывода

s=textBox4->Text;

x = System::Convert::ToDouble(s);

ann->MaxErr=x; // максимальная ошибка

s=textBox2->Text;

x = System::Convert::ToDouble(s);

ann->Delta=x; // Дельта

s=textBox3->Text;

x = System::Convert::ToDouble(s);

ann->Beta=x; // Бета

s=textBox6->Text;

k = System::Convert::ToInt32(s);

ann->NumberLayers=k;

ann->neurons[0]=ann->NinputX\*ann->NinputY; // нейроннов в 0-ом слое

// устанавливаем количество нейроннов по слоям (к = количество слоев)

for(i=1;i<=k;i++){

s=listBox1->Items[i-1]->ToString();

ann->neurons[i]=System::Convert::ToInt32(s); }

chart1->Series[0]->Points->Clear(); // очищаем график

ann->Init(); // инициализация

StepTrain();}

void StepTrain() {

int ne,m;

bool pr;

ne=0;

m=0;

while (1) {

pr=ann->Train();

if(pr) {

break; }

ne += ann->Step;

if(ne>=ann->Epoch) {

break; }

chart1->Series[0]->Points->AddXY(m,ann->Er);

chart1->Refresh();

m++;}

String ^ s1 =gcnew String("");

String ^ s2 =gcnew String("");

s1="Результат";

s2 = ne.ToString();

s1=s1+" Эпох:";

s1=s1+s2;

s2=ann->Er.ToString();

s1=s1+" Ошибка: ";

s1=s1+s2;

label11->Text=s1; }};}

NetTest.h

#pragma once

#include "Ann.h"

namespace NeuralNets {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

private: System::Void NetTest\_Load(System::Object^ sender, System::EventArgs^ e) {

int h,w;

int wp,hp,hl;

String ^s;

System::Drawing::Rectangle workingRectangle = Screen::PrimaryScreen->WorkingArea;

this->Size = System::Drawing::Size( workingRectangle.Width - 10, workingRectangle.Height - 10);

this->Left = 0;

this->Top = 0;

h=workingRectangle.Height - 10 ;

w=workingRectangle.Width - 10 ;

wp=w-20;

hp=h-80;

hl=20;

pictureBox1->Left=350;

pictureBox1->Top=20;

listBox1->Left=500;

listBox1->Top=20;

listBox1->Height=h-100;

if(ann->Type==0 || ann->Type==1) {

s = (ann->NinputX\*ann->NinputY).ToString();

s="Размерность входа: "+s;

label1->Text=s;

s = (ann->Noutput).ToString();

s="Размерность выхода: "+s;

label2->Text=s;

s = (ann->Number).ToString();

s="Количество векторов: "+s;

label3->Text=s; } }

private: System::Void button1\_Click(System::Object^ senderSystem::EventArgs^ e) {

int k,w,h;

int m,x,y,i,j,p,q,z;

int Sc;

Color col;

unsigned char cc;

cc=0;

String ^s;

s=textBox2->Text;

k = System::Convert::ToInt32(s);

Sc=k;

w=ann->NinputX\*k;

h=ann->NinputY\*k;

pictureBox1->Width=w;

pictureBox1->Height=h;

bmp = gcnew Bitmap(w,h);

pictureBox1->Image = bmp;

g = Graphics::FromImage(bmp);

// listbox

listBox1->Left=pictureBox1->Left+pictureBox1->Width+30;

SolidBrush ^br;

Color c1;

c1 = System::Drawing::Color::FromArgb(255,255,255);

br=gcnew SolidBrush(c1);

g->FillRectangle(br,0,0,w,h);

s=textBox1->Text;

m = System::Convert::ToInt32(s);

x=0;

y=0;

for(q=0;q<ann->NinputY;q++) {

for(p=0;p<ann->NinputX;p++){

z=q\*ann->NinputX+p;

z=z+m\*ann->NinputX\*ann->NinputY;

cc=ann->Data[z];

if(ann->Type==0) {

if(cc==0)

cc=255;

else

cc=0;

col=Color::FromArgb(cc,cc,cc);

for(i=0;i<Sc;i++) {

for(j=0;j<Sc;j++) {

bmp->SetPixel(x+i,y+j,col); }} }

if(ann->Type==1) {

col=Color::FromArgb(cc,cc,cc);

for(i=0;i<Sc;i++){

for(j=0;j<Sc;j++) {

bmp->SetPixel(x+i,y+j,col); } } }

x+=Sc;}

x=0;

y+=Sc; }

if(ann->Type<2) {

k=0;

for(q=0;q<ann->NinputY;q++){

for(p=0;p<ann->NinputX;p++){

z=q\*ann->NinputX+p;

z=z+m\*ann->NinputX\*ann->NinputY;

cc=ann->Data[z];

ann->CurrentVector[k]=(double)cc;

k++; }} }

ann->Forward();

listBox1->Items->Clear();

for(i=0;i<ann->Noutput;i++) {

s=ann->CurrentOutput[i].ToString();

listBox1->Items->Add(s); }}};}