**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«РОССИЙСКИЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Г.В. ПЛЕХАНОВА»**

Факультет математической экономики, статистики и информатики

Кафедра Автоматизированных систем обработки информации и управления

|  |
| --- |
| «Допустить к защите» |
| Заведующий кафедрой  АСОиУ  Микрюков Андрей Александрович  «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_2016 г. |

Выпускная квалификационная работа

Направление 02.03.03 «Математическое обеспечение и администрирование информационных систем»

Профиль «Разработка бизнес-ориентированных информационных систем»

ТЕМА: «Разработка программного обеспечения для распознавания изображений с использованием нейросетевых технологий»

Выполнил студент Новиков Роман Сергеевич

|  |
| --- |
| Научный руководитель выпускной квалификационной работы  Мазуров Михаил Ефимович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Автор\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Оглавление

[ВВЕДЕНИЕ 3](#_Toc454917052)

[1. ТЕХНИКО-ЭКОНОМИЧЕСКАЯ ХАРАКТЕРИСТИКА СРЕДСТВА ДЛЯ РЕШЕНИЯ ЗАДАЧИ РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ 5](#_Toc454917053)

[1.1. О классификации и приложении искусственных нейронных сетей 5](#_Toc454917054)

[1.1.1. Классификация искусственных нейронных сетей по структуре 6](#_Toc454917055)

[1.1.2. Обучение нейронных сетей 7](#_Toc454917056)

[1.1.3. Прикладные задачи, решаемые нейронными сетями 9](#_Toc454917057)

[1.2. Обзор существующих методов решения задачи распознавания изображений. 15](#_Toc454917058)

[1.2.1. Искусственный нейро́н - нейрон Маккалока - Питтса 15](#_Toc454917059)

[1.2.2. Сравнительный анализ известного метода распознавания сложных объектов и рассматриваемого в данной работе. 17](#_Toc454917060)

[1.3. Постановка задачи квалификационной работы 20](#_Toc454917061)

[Вывод по главе 1 20](#_Toc454917062)

[2. ИСПОЛЬЗОВАНИЕ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ 22](#_Toc454917063)

[2.1. Математическая модель нейрона МакКаллока-Питтса 22](#_Toc454917064)

[2.2. Избирательный нейрон 22](#_Toc454917065)

[2.3. Структурные схемы нейронных сетей типа перцептронов. Нейронные сети типа однослойного и многослойного перцептронов 25](#_Toc454917066)

[2.3. Математическая теория избирательного нейрона 26](#_Toc454917067)

[2.4. Ввод данных на вход нейронной сети 28](#_Toc454917068)

[2.5. Методы получения бинарных строк 29](#_Toc454917069)

[Вывод по главе 2. 34](#_Toc454917070)

[3. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПО РАСПОЗНАВАНИЮ ИЗОБРАЖЕНИЙ. 35](#_Toc454917071)

[3.1. Обоснование выбора аппаратных и программных средств разработки 35](#_Toc454917072)

[3.1.1. Выбор аппаратной части 35](#_Toc454917073)

[3.1.2.Выбор операционной системы 36](#_Toc454917074)

[3.1.3.Выбор среды разработки и языка программирования 37](#_Toc454917075)

[3.2. Информационное обеспечение задачи 37](#_Toc454917076)

[3.3. Работа с эталонным полноцветным изображением. 38](#_Toc454917077)

[3.4. Работа с тестовым полноцветным изображением. 42](#_Toc454917078)

[3.5. Состав и структура программного обеспечения. 43](#_Toc454917079)

[3.6. Описание программных модулей. 46](#_Toc454917080)

[3.7. Руководство пользователя программного обеспечения по распознаванию изображения. 65](#_Toc454917081)

[Вывод по главе 3. 69](#_Toc454917082)

[4. ЭКСПЕРИМЕНТАЛЬНАЯ ПРОВЕРКА И ОЦЕНКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ НА ИЗБИРАТЕЛЬНЫХ НЕЙРОНАХ. 70](#_Toc454917083)

[4.1. Описание результатов испытаний 70](#_Toc454917084)

[4.2. Оценивание надежности программного обеспечения 71](#_Toc454917085)

[4.3. Обоснование экономической эффективности 73](#_Toc454917086)

[4.3.1. Методика расчета экономической эффективности 73](#_Toc454917087)

[4.3.2. Расчет показателей экономической эффективности 77](#_Toc454917088)

[Вывод по главе 4 83](#_Toc454917089)

[ЗАКЛЮЧЕНИЕ 84](#_Toc454917090)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 85](#_Toc454917091)

[ПРИЛОЖЕНИЯ 88](#_Toc454917092)

[Приложение 1. Метод ближайшего соседа. 89](#_Toc454917093)

[Приложение 2. Блок-схема решение задачи. 90](#_Toc454917094)

[Приложение 3. Листинги программного обеспечения. 91](#_Toc454917095)

# ВВЕДЕНИЕ

Задача распознавания изображений в настоящее время является актуальной при распознавании изображений со спутников, самолетов, беспилотных аппаратов; решении задач управления автоматического регулирования, принятия решений. Также задача распознавания является важнейшей информационной задачей для биологических систем.

Широкий круг задач, решаемый нейронными сетями, не позволяет в настоящее время создавать универсальные, мощные сети, вынуждая разрабатывать специализированные сети, функционирующие по различным алгоритмам. Тем не менее, тенденции развития нейронных сетей растут.

Целью данной выпускной квалфикационной работы являетсяразработка методов распознавания черно-белых и полноцветных изображений на основе нейросетевых технологий для широкого круга прикладных задач.

Объектом исследования являются существующие подходы к распознаванию черно-белых и полноцветных изображений, основанные на нейросетевых технологиий. Предметом исследования методы получения черно-белых и полноцветных изображений, позволяющие повысить эффективность распознавания по сравнению с нейросетевыми технологиями, использующих искусственные нейроны Маккалока-Питса.

В соответствии со сформулированной целью в выпускной квалификационной работе были поставлены следующие задачи:

1. Обзор существующих методов распознавания изображений, использующих нейроны Маккаллока-Питса;
2. Разработка нейросетевой технологии распознавания изображений, основанной на использовании избирательных нейронов;
3. Распознавание черно-белых и полноцветных изображений;
4. Расчет экономической эффективности и надежности ПО.

# ТЕХНИКО-ЭКОНОМИЧЕСКАЯ ХАРАКТЕРИСТИКА СРЕДСТВА ДЛЯ РЕШЕНИЯ ЗАДАЧИ РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ

## О классификации и приложении искусственных нейронных сетей

Современные вычислительные машины во много раз превосходят человека в вопросах вычислительного характера, однако, человеческий мозг способен, например, хранить до 10 миллиардов бит информации в столь маленьком объеме, при этом обеспечивая столь быстрый доступ к хранящейся информации. Также человек превосходит компьютер в вопросе преобразования той лавины частот, которая приходит от различных органов чувств (частоты света, звуковые частоты и так далее), в конкретное представление о мире, то есть в вопросах распознавания объектов. Основной причиной в столь сильном различии в производительности компьютера и человека является различие архитектур построения нейронной системы человеческого мозга и логической бинарной системы современных компьютеров. В частности архитектура вычислительной машины оптимизирована для решения дискретных вычислительных задач, человеческий мозг же оптимизирован для решения нелинейных аналоговых задач, например, решения проблем восприятия.

Таким образом, проблемы интеграции между компьютером и человеком могут решаться с помощью создания искусственных нейронных сетей, архитектура которых с одной стороны опирается на архитектуру реальных нейронных сетей, а с другой стороны использует вычислительные возможности современных вычислительных машин.

Подобно реальной нейронной системе нейронные сети является вычислительной системой с огромным числом параллельно функционирующих простых процессоров с множеством связей [Анохин П.К., 1973;, Виноградова, 2000; Дунин-Барковский, Терехин, 1990; Мак-Каллок, Питтс, 1956; Розенблат, 1965; Уоссермен, 1992; Хайкин, 2006]. Модели нейронных сетей в некоторой степени воспроизводят «организационные» принципы, свойственные мозгу человека. Моделирование структуры мозга с использованием нейронных сетей может также способствовать лучшему пониманию биологических функций. С другой стороны, искусственные нейронные сети также стимулируют эти дисциплины, обеспечивая их новыми инструментами и представлениями. Этот симбиоз жизненно необходим для исследований по нейронным сетям.

### Классификация искусственных нейронных сетей по структуре

Вычислительные системы, основанные на искусственных нейронных сетях, обладают рядом качеств, которые отсутствуют в машинах с архитектурой фон Неймана но присущи мозгу человека:

1. Массовый параллелизм;

2. Распределённое представление информации и вычисления;

3. Способность к обучению и обобщению;

4. Адаптивность;

5. Свойство контекстуальной обработки информации;

6. Толерантность к ошибкам.

С точки зрения структуры или архитектурыискусственная нейронная сеть (ИНС) может рассматриваться как направленный граф с взвешенными связями, в котором искусственные нейроны являются узлами. Связи между нейронами характеризуются весами, отражающими степень влияния передающих нейронов на принимающие нейроны.

По архитектуре связей искусственные нейронные сети могут быть сгруппированы в два класса, как показано на рисунке 1.

Можно выделить сети прямого распространения, в которых графы не имеют петель, и рекуррентные сети, или сети с обратными связями. В наиболее распространенном семействе сетей первого класса, называемых многослойным перцептроном, нейроны расположены слоями и имеют однонаправленные связи между слоями. На рисунке 1 представлены типовые сети каждого класса.

Сети прямого распространения являются статическими в том смысле, что на заданный вход они вырабатывают одну совокупность выходных значений, не зависящих от предыдущего состояния сети. Рекуррентные сети являются динамическими, так как в силу обратных связей в них модифицируются входы нейронов, что приводит к изменению состояния сети. Рассмотрим классификация нейронных сетей по архитектуре.

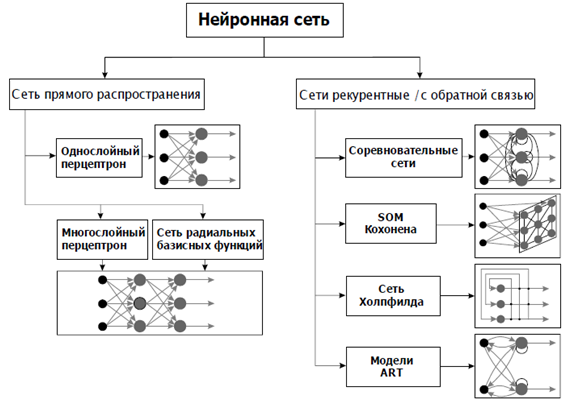
****

Рис.1. Классификация нейронных сетей по архитектуре

### Обучение нейронных сетей

В контексте искусственной нейронной сети процесс обучения имеет следующее определение. Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения - одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Обучение нейронной сети- настройка архитектуры сети и весов связей для эффективного выполнения специальной задачи. Обычно нейронная сеть должна настроить веса связей по имеющейся обучающей выборке. Функционирование сети улучшается по мере итеративной настройки весовых коэффициентов. Свойство сети обучаться на примерах делает их более привлекательными по сравнению с системами, которые следуют определенной системе правил функционирования, сформулированной экспертами. Для конструирования процесса обучения, прежде всего, необходимо иметь модель внешней среды, в которой функционирует нейронная сеть - знать доступную для сети информацию. Эта модель определяет парадигму обучения. Во-вторых, необходимо понять, как модифицировать весовые параметры сети, какие правила обучения управляют процессом настройки. Алгоритм обучения означает процедуру, в которой используются правила обучения для настройки весов. Процесс обучения нейронной сети с обратным распространением ошибки схематически иллюстрируется рис. 2.

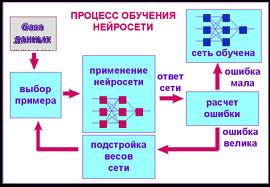


Рис. 2. Процесс обучения нейронной сети

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п. С математической точки зрения, обучение нейронных сетей — это многопараметрическая задача нелинейной оптимизации. С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники. С точки зрения развития вычислительной техники и программирования, нейронная сеть — способ решения проблемы эффективного параллелизма [Александров, Анохин и др.,2008; Дунин-Барковский, Терехин, 1990; Уоссермен, 1992; Хайкин, 2006; Цетлин, 1969; Ясницкий, 2005]. А с точки зрения искусственного интеллекта, искусственная нейронная сеть является основой философского течения коннективизма и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искаженных данных.

По характеру обучения нейронные сети подразделяются на группы:

1. Обучение с учителем — выходное пространство решений нейронной сети известно.
2. Обучение без учителя — нейронная сеть формирует выходное пространство решений только на основе входных воздействий. Такие сети называют самоорганизующимися. Обучение с подкреплением — система назначения штрафов и поощрений от среды.

### Прикладные задачи, решаемые нейронными сетями

Рассмотрим некоторые задачи, решаемые искусственными нейронными сетями, с дополнением кратких пояснений.

1. Распознавание образов;
2. Кластер-анализ и классификация;
3. Ассоциативная память;
4. Сжатие данных;

5) Аппроксимация функций;

6) Задачи управления;

7) Управление движением транспортного средства и т. д.;

8) Прогнозирование;

9) Принятие решений;

10) Задачи оптимизации;

Приведем краткую классификацию искусственных нейронных сетей, не претендующую на полноту и основанную на учете приложений искусственных нейронных сетей, сведенную в таблицу 1.

**Таблица 1.**

**Искусственные нейронные сети**

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Автор | Год создания | Применение |
| Однослойный перцептрон (Single layer perceptron) | R. Rosenblatt | 1959 | Распознавание образов/категоризация |
| Обратное распространение ошибки (Back Propagation) | R. Rosenblatt,  M.Minsky, S. Papert | 1960-е | Распознавание образов, классификация, прогнозирование |
| Встречное распространение (Counter Propagation) | R. Hecht-Neilsen | 1986 | Распознавание образов, восстановление образов (ассоциативная память), сжатие данных |

*Продолжение таблицы 1*

|  |  |  |  |
| --- | --- | --- | --- |
| Входная звезда (Instar Network) | S. Groosenberg | 1974 | Распознавание образов |
| Выходная звезда (Outstar Network) | S. Groosenberg | 1974 | Распознавание образов |
| Искусственный резонанс-1 (ART-1 Network) | S. Grossenberg, G. Carpenter | 1986 | Распознавание образов, кластерный анализ |
| Сеть Хопфилда (Hopfield Network) | J. J. Hopfield | 1982 | Поиск и восстановление данных по их фрагментам |
| Сеть Хэмминга (Hamming Network) | R. W. Hamming | 1987 | Распознавание образов, классификация, ассоциативная память, надежная передача символов в условиях помех |
| Сеть Кохонена (Kohonen Network) | T. Kohonen | 1984 | Кластерный анализ, распознавание образов, классификация |
| Сеть поиска максимума (MAXNET) | R. P. Lippman | 1987 | Совместно с сетью Хэмминга, в составе НС систем распознавания образов |
| Сеть поиска максимума с прямыми связями (Feed-Forward MAXNET) | R. P. Lippman | 1987 | Совместно с сетью Хэмминга, в составе НС систем распознавания образов |

*Продолжение таблицы 1*

|  |  |  |  |
| --- | --- | --- | --- |
| Двунаправленна ассоциативная память (BAM Network) | B. Kosko | Вторая половина 80-х | Ассоциативная память, распознавание образов |
| Обучение Больцмана (Boltzman machine) | J. Hinton,  T. Sejnovsky, H. Szu | 1985 | Распознавание изображений, сигналов радара, радара |
| Классификатор Гаусса (Neural Gaussian Classifier) | R. P. Lippman | 1987 | Распознавание, образов, классификация |
| Генетические алгоритмы (Genetic training algorithm) | J. Holland, D. Goldberg | 1975 1988 | Обучение НС, распознавание сигналов сонаров |

Дадим краткие пояснения прикладным задачам, решаемым с помощью искусственной нейронной сети.

1. Классификация образов.

Задача состоит в указании принадлежности входного образа (например, речевого сигнала или рукописного символа), представленного вектором признаков, одному или нескольким предварительно определенным классам. К известным приложениям относятся распознавание букв, распознавание речи, классификация сигнала электрокардиограммы, классификация клеток крови. В робототехнике одним из основных приложений является распознавание объектов из видеоинформации, полученной от системы технического зрения.

1. Кластер-анализ и классификация (без учителя).

К задачам распознавания относятся также кластеризация. При решении задачи кластеризации, которая известна также как классификация образов "без учителя", отсутствует обучающая выборка с метками классов. Кластеризация - метод, применяющийся для анализа больших наборов данных заключающийся в разбиении всего множества на группы близкородственных элементов (кластеры). Кластеризация может быть использована для решения таких задач как обработка изображений, классификация, тематический анализ коллекций документов, построение репрезентативной выборки.

1. Ассоциативная память*.*

В модели вычислений фон Неймана, которая используется в современных вычислительных машинах, обращение к памяти доступно только посредством адреса, который не зависит от содержания памяти. Более того, если допущена ошибка в вычислении адреса, то может быть найдена совершенно иная информация. Ассоциативная память, или память, адресуемая по содержанию, доступна по указанию заданного содержания. Содержимое памяти может быть вызвано даже по частичному входу или искаженному содержанию. Ассоциативная память чрезвычайно желательна при создании мультимедийных информационных баз данных.

1. Сжатие данных.

Способность нейросетей к выявлению взаимосвязей между различными параметрами дает возможность выразить данные большой размерности более компактно, если данные тесно взаимосвязаны друг с другом. Обратный процесс — восстановление исходного набора данных из части информации — называется автоассоциативной памятью. Автоассоциативная память позволяет также восстанавливать исходный сигнал/образ из зашумленных/поврежденных входных данных. Решение задачи гетероассоциативной памяти позволяет реализовать память, адресуемую по содержимому.

1. Аппроксимация функций*.*

Предположим, что имеется обучающая выборка  (пары данных вход-выход), которая генерируется неизвестной функцией, искаженной шумом. Задача аппроксимации состоит в нахождении оценки неизвестной функции . Аппроксимация функций необходима при решении многочисленных инженерных и научных задач моделирования.

1. Задачи управления.

Рассмотрим динамическую систему, заданную совокупностью , где  является входным управляющим воздействием, а  - выходом системы в момент времени . В системах управления с эталонной моделью целью управления является расчет такого входного воздействия , при котором система следует по желаемой траектории, диктуемой эталонной моделью. Примером является оптимальное управление приводами роботов.

1. Управление движением транспортного средства и т. д.

Задачи управления движением в настоящее время являются актуальными и активно развиваются.

8) Прогнозирование**.**

Пусть заданы n дискретных отсчетов в последовательные моменты времени  Задача состоит в предсказании значения в некоторый будущий момент времени, что можно выразить в виде формулы . Предсказание/прогноз имеют значительное влияние на принятие решений в бизнесе, науке и технике. Предсказание цен на фондовой бирже и прогноз погоды являются типичными приложениями техники предсказания/прогноза.

1. Принятие решений.

Эта задача близка к задаче классификации. Классификации подлежат ситуации, характеристики которых поступают на вход нейронной сети. На выходе сети при этом должен появиться признак решения, которое она приняла. При этом в качестве входных сигналов используются различные критерии описания состояния управляемой системы.

10) Оптимизация.

Многочисленные проблемы в математике, статистике, технике, науке, медицине и экономике могут рассматриваться как проблемы оптимизации. Задачей алгоритма оптимизации является нахождение такого решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию. Задача коммивояжера, является классическим примером задачи оптимизации.

## Обзор существующих методов решения задачи распознавания изображений.

Распознавание изображений — это отнесение исходных данных к определенному классу с помощью выделения существенных признаков, характеризующих эти данные, из общей массы несущественных данных.

Классическая постановка задачи распознавания изображений: Дано множество изображений. Относительно них необходимо провести классификацию. Множество представлено подмножествами, которые называются классами. Заданы: информация о классах, описание всего множества и описание информации об изображении, принадлежность которого к определенному классу неизвестна. Требуется по имеющейся информации о классах и описании изображения установить - к какому классу относится это изображение.

Представим один из методов решения данной задачи.

### Искусственный нейро́н - нейрон Маккалока - Питтса

Математически искусственный нейрон обычно представляют как некоторую нелинейную функцию от единственного аргумента – линейной комбинации всех входных сигналов. Эту функцию называют функцией активации, функцией срабатывания или передаточной функцией нейрона. Полученный результат отправляется на единственный выход нейрона. Математическая модель искусственного нейрона представлена на рис. 2.

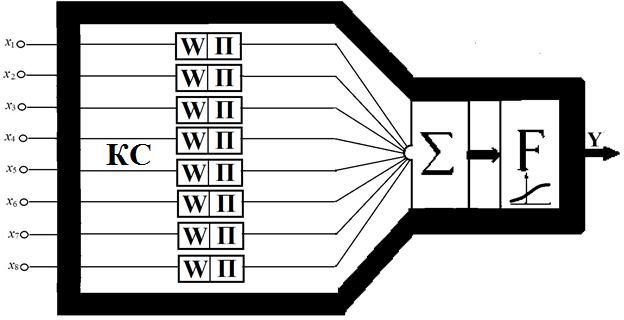


Рис. 3. Структура искусственного нейрона МакКаллока – Питтса

Линейная комбинация входных сигналов с определенными весами подается на вход нейрона. Кроме того, иногда ко входу нейрона специально добавляют некоторую случайную величину, которая называется смещением. Смещение можно рассматривать как сигнал на дополнительном, всегда нагруженном синапсе.

Результирующую реакцию формального нейрона можно представить следующим образом. После сумматора получаем ответ S по формуле

|  |  |
| --- | --- |
| , | (1.1) |

где – сигналы на входах нейрона,

- веса входов нейрона,

– скалярное произведение ,

N – число входов нейрона.

После прохождения порогового устройства и преобразования пороговой функцией получаем выходной ответ , равный

|  |  |
| --- | --- |
| . | (1.2) |

Взвешенную сумму иногда представляют в виде , где - порог нейронного элемента, характеризующий сдвиг функции активации по оси абсцисс.

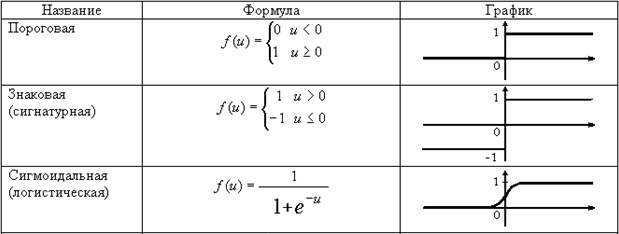
Нейрон содержит следующие блоки:

1. Умножители. Количество умножителей должно быть равно количеству входов нейрона.
2. Сумматор - сглаживающий фильтр. Сумматор один, и с увеличением количества входов его сложность не меняется.
3. Функция активации. Функция активации определяет зависимость сигнала на выходе нейрона от взвешенной суммы сигналов на его входах.

Различные виды функций активации приведены в таблице 2.

**Таблица 2.**

**Виды функций активации**



### Сравнительный анализ известного метода распознавания сложных объектов и рассматриваемого в данной работе.

Решение задачи распознавания 100 изображений размером 100 х 100, когда каждое изображение содержит 10000 пикселей. Бинарная строка содержит 10000 элементов при распознавании контурных изображений, при распознавании полутоновых изображений содержит 30000 элементов и более, при распознавании полноцветных изображений содержит 90000 элементов и более.

При наличии 100 распознаваемых изображений общее количество весовых коэффициентов при распознавании однослойного перцептрона на основе нейронов Маккаллока-Питса будет равнятся 10000 х 100 =1 миллиону. При распознавании цветных изображений количество весовых коэффициентов будет равно 9 миллионов и более.

Вычисления весовых коэффициентов производится по следующим неравенствам:

|  |  |
| --- | --- |
|  | (1.3) |

где - характеристика схожести i-го изображения с j-м изображением,

– сигнал от i-го входного нейрона к j-му выходному нейрону, ,

– весовой коэффициент сигнала от i-го входного нейрона к j-му выходному нейрону,

– пороговое значение,

n – количество ячеек бинарной строки (входных нейронов),

m – количество изображений (выходных нейронов).

Всего имеется условий для нахождения m\*n весовых коэффициентов . Если, например, n=10000, m=100, то у необходимых условий для нахождения m\*n коэффициентов в 100 раз меньше.

Ввиду такого положения, задача вычисления весовых коэффициентов является неопределенным, допускающим бесчисленное количество решений, если n>m.

Сама вычислительная процедура для нахождения 10000\*100=1000000 коэффициентов является весьма трудоемкой, требует значительного объема вычислений.

Следует отметить, что большое количество весовых коэффициентов приводит к ненадежной и неустойчивой работе перцептрона в целом. Добавление нового объекта распознавания требует пересчета всей системы весовых коэффициентов. То есть все обучение требуется проводить заново.

Избирательный нейрон не требует использования весовых коэффициентов, поскольку определение входящих в кластер нейрона каналов связи реализуется по кодовой комбинации входного сигнала непосредственно без всяких расчетов, что обеспечивает высокую эффективность использования избирательных нейронов. Избирательный нейрон не обладает недостатками с ненадежностью и неустойчивостью больных нейронных сетей.

Избирательный перцептрон допускает добавление очередного объекта распознавания без каких-либо принципиальных трудностей, что будет показано ниже.

В связи со сказанным выше в данной работе для решения сложной задачи распознавания черно-белых и полноцветных изображений была предложена и исследована система распознания черно-белых и полноцветных изображений была предложена система распознавания, основанная на использовании избирательных нейронов и нейронных сетей.

## Постановка задачи квалификационной работы

Целью квалификационной работы является разработка программного обеспечения для распознавания изображений по введенному пользователем тестовому изображению. Имеется 50 эталонных портретов, 50 картин и 10 электрокардиограмм. Данные изображения представляют собой jpg-файлы формата N\*N. Портреты, картины и электрокардиограммы находятся в своих директориях. Также имеются файлы с описанием соответствующих изображений, расположенных в своих директориях.

По введенному тестовому изображению необходимо определить изображение из хранящихся в базе данных и вывести его на экран с описанием.

Входными данными для программного средства будут являться:

1. Базы данных эталонных портретов и соответствующих описаний.
2. Базы данных эталонных картин и соответствующих описаний.
3. Базы данных эталонных электрокардиограмм и соответствующих описаний.
4. Введенное пользователем изображение для распознавания.

Результатом работы программы будет вывод на экран наиболее похожего эталонного изображения с его описанием, либо вывод на экран сообщения о том, что искомое изображение отсутствует в базе.

## Вывод по главе 1

В первой главе выполнен обзор и классификация искусственных нейронных сетей, рассмотрены варианты их применения на практике.

Также были рассмотрены некоторые существующие методы решения задачи распознавания изображений.

Затем была поставлена задача выпускной квалификационной работы, заключающаяся в создании нового программного обеспечения. Разрабатываемое программное обеспечение должно будет распознавать тестовые изображения с возможностью вывода наиболее похожего эталонного изображения портретов, картин или электрокардиограмм с соответствующим описанием.

# 2. ИСПОЛЬЗОВАНИЕ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ

## 2.1. Математическая модель нейрона МакКаллока-Питтса

Первой формальной моделью нейронных сетей была модель МакКаллока-Питтса. Суть её заключается в следующем.

Имеется n входных величин импульсов  в бинарном формате, описывающих объект . Попадая в нейрон, импульсы сталкиваются с весами . Если вес положительный, то соответствующий синапс возбуждающий, если отрицательный, то тормозящий. После импульсы суммируются и преобразуются и на выходе мы получаем 0 либо 1.

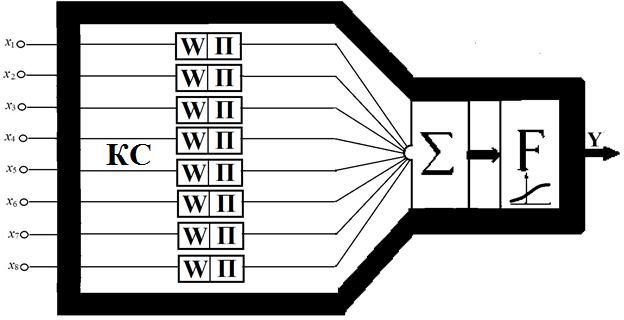


Рис. 4. Схема нейрона МакКаллока – Питтса

## 2.2. Избирательный нейрон

Избирательный нейрон относится ко второй волне построения нейронных сетей. Именно он используется в данной работе в качестве основы нейронной сети. Эта модель нейрона более приближена к модели биологического нейрона, чем нейрон МакКаллока-Питтса, за счёт того, что отражение входной информации в ней отражается не в весовых коэффициентах, а в изменении количества и качества дендритов.

Может ли система дендритов на входе нейрона быть столь пластичной и лабильной, чтобы адекватным образом отражать входную информацию за счет изменения числа возбуждающих и тормозных дендритов? Ранее этот вопрос был изучен недостаточно. Однако в последнее время были сделаны ряд удивительных открытий, показавших наличие у дендритов совершенно необычных возможностей их роста, изменения количества и качества. Установлено также наличие у дендритов активных свойств, похожих на свойства аксонов. Все это вместе взятое доказывает возможность адаптации к отражению любой входной информации не за счет весовых функций дендритов, а за счет изменения их количества и качества дендритов. Под различным качеством дендритов понимаются их свойства: дендриты возбуждающие и тормозные, различные по диаметру и электрической мощности и т.д.; возможность роста и гибели отдельных дендритов. В связи с этим было предложено устройство — избирательный нейрон, выполняющее функции нейрона, структурная схема которого показана на рис. 12.

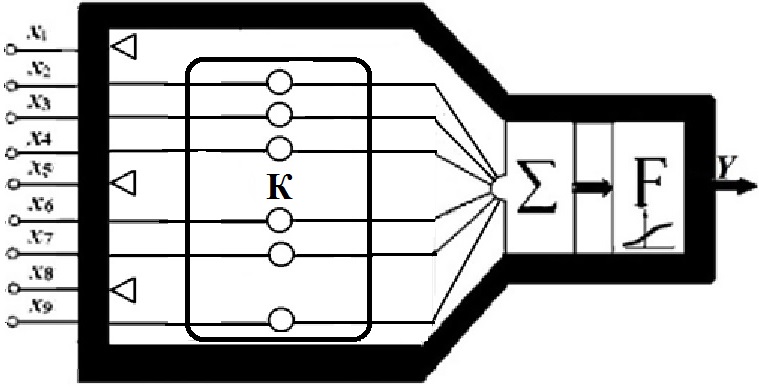


Рис. 5. Схема избирательного нейрона. На рисунке приняты следующие обозначения: – входные сигналы; К – формируемые кластеры дендритов; ∑ - сумматор; F - пороговый нелинейный преобразователь.

Рассмотрим кратко его устройство и отличие от обычного искусственного нейрона МакКаллока – Питтса. Треугольниками отмечены заблокированные каналы связи из числа входных, не входящие в состав характеристических векторов для объекта на входе перцептрона. Схематически устройство, формирующее кластеры дендритов К показано на рисунке с левой стороны от блок-схемы.

Нормирующее устройство производит выравнивание максимального числа активных входов при использовании нейрона в качестве элемента нейронной сети. Нейронной сетью является, например, однослойный перцептрон с несколькими нейронами в слое.

Математическая модель избирательного нейрона в большей степени учитывает реальные свойства биологического нейрона. Решение основной задачи нейрона — сжатие сенсорной или промежуточной информации, распознавание образов — достигается за счет изменения числа дендритов на входе нейрона, возбуждающих и тормозных. Надобность в регулировании весовых функций дендритов отпадает.

Принципиальным отличием предлагаемого избирательного нейрона от искусственного нейрона МакКаллока—Питтса является полное отсутствие весовых коэффициентов для входных воздействий и всех недостатков нейронных сетей, связанных с наличием весового суммирования.

Для обработки информации используются нейронные сети. Основой этих сетей – элементарным элементом – является однослойный перцептрон. Более подробно нейронные сети перцептронного типа описаны ниже.

## 2.3. Структурные схемы нейронных сетей типа перцептронов. Нейронные сети типа однослойного и многослойного перцептронов

Рассмотрим основные структурные схемы нейронных сетей. Однослойные нейронные сети для обработки импульсных потоков от рецепторов и промежуточных нейронов показаны на рис. 5.

Для распознавания возможно использование многослойных перцептронов с различным количеством скрытых слоев. Перцептрон, включающий 2 слоя нейронов, показан на рис. 6.

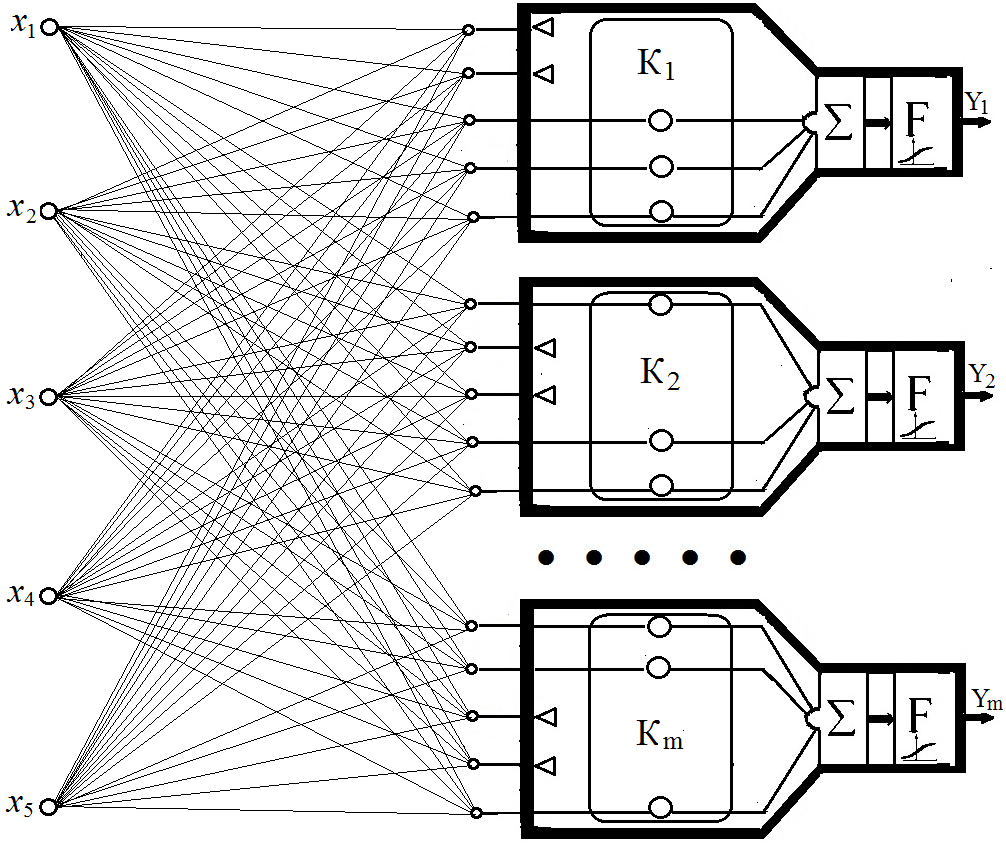


Рис. 6 Блок-схема избирательного однослойного перцептрона на избирательных нейронах

В данной работе используется избирательный однослойный перцептрон, показанный на рис. 5.

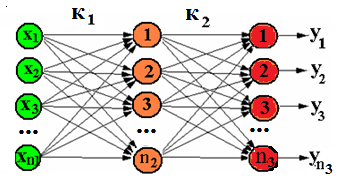


Рис. 7. Перцептрон, включающий 2 слоя нейронов. K1, K2 – системы каналов.

Перцептрон, включающий 3 слоя нейронов, показан на рис. 7.

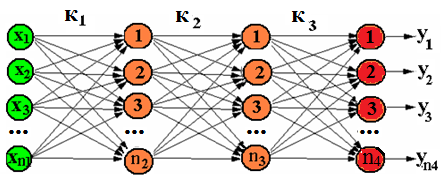


Рис. 8. Перцептрон, включающий 3 слоя нейронов. K1, K2, K3 – системы каналов.

## 2.3. Математическая теория избирательного нейрона

Рассмотрим математическую теорию рассматриваемого избирательного нейрона. Обозначим возможные характеристические кодовые комбинации объектов на входе нейрона в виде векторов ; … ; , где n - число элементов кодовой комбинации; m - число объектов. Все возможные кодовые комбинации входных объектов образуют матрицу A, которую можно представить в виде

|  |  |
| --- | --- |
|  | (2.1) |

Пусть конкретный нейрон содержит кластер связей, характеризуемый кодовой комбинацией . При подаче на вход нейрона с номером  кодовой комбинации входного объекта получим

|  |  |
| --- | --- |
| . | (2.2) |

Значения сумм  равны элементам матрицы *B*, равной

|  |  |
| --- | --- |
| , | (2.3) |

где - транспонированная матрица к *А*. Всего мы получим х сумм . Наибольшей будет сумма

|  |  |
| --- | --- |
| , | (2.4) |

где - число единиц в кодовой комбинации . Свойство сумм в том, что , используется для распознавания входных объектов.

Можно дать красивую интерпретацию избирательных свойств однослойного перцептрона, представив значения матрицы  в виде графика в трехмерном пространстве. С физической точки зрения значения  - это значения матрицы , численно равные сумме после прихода входного сигнала  через кластер каналов связи, соединяющий входы нейрона с номером i на выходе перцептрона с его пороговой нелинейной системой. Для построения графика в трехмерном пространстве воспользуемся программой в Matlab-7. По осям Ох и Оу отложены значения i и j .

В качестве примера было реализовано избирательное распознавание для 30 картин. Графическая интерпретация избирательных свойств системы иллюстрируется рисунком, приведенным ниже.

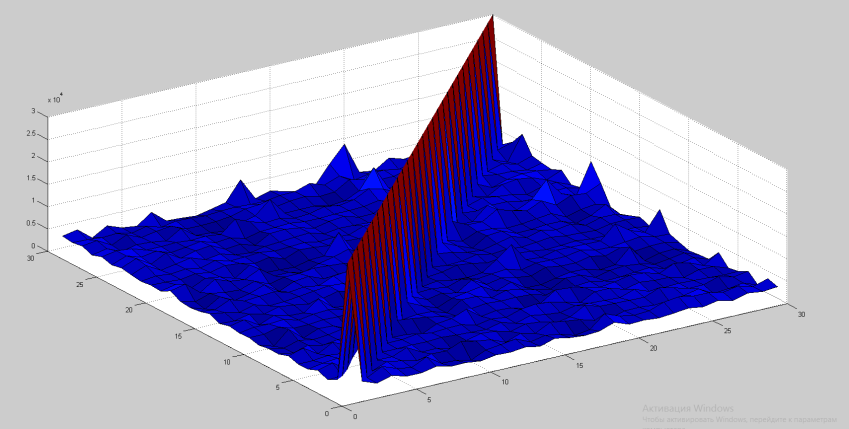
****

Рис. 9. Характеристики информационных фильтров входных объектов

На приведенном рисунке показан график, характеризующий избирательные свойства системы. Избирательность характеризуется диагональной частью графика и выражена достаточно эффективно.

## 2.4. Ввод данных на вход нейронной сети

Ввод данных на вход нейронной сети, в нашем случае однослойного перцептрона, может быть осуществлен с помощью выборки изображений из нужной директории с их предварительным преобразованием в бинарную кодовую строку или последовательность. Способ ввода изображений иллюстрируется рисунком 15.

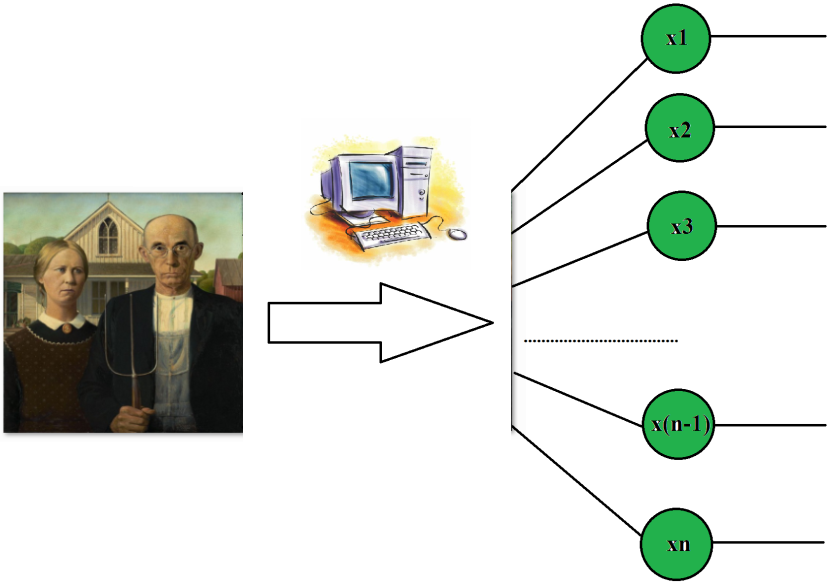


Рис. 10. Ввод изображения из директории с последующим получением бинарных кодовых комбинаций с помощью компьютера

## 2.5. Методы получения бинарных строк

Метод 1. Кодирование интенсивности за счет движения единицы по длине элемента.

* 1. Представление пикселя в виде 0 или 1.

Дано бинарное изображение в виде матрицы, где цвет каждого пикселя может быть либо черным, либо белым. Для вычислений преобразуем матрицу в вектор-строку и рассмотрим каждый пиксель. Преобразуем каждое значение вектора в число по формуле:

|  |  |
| --- | --- |
| *,* | (2.5) |

где Code – число, характеризующее пиксель,

White – признак, является ли пиксель белым.

После данных преобразований получается бинарная комбинация из нулей и единиц, пригодная для использования в процессе распознавания изображения.

Достоинства:

1) Самый маленький размер бинарной комбинации, получаемой на выходе, при использовании данного метода по сравнению с остальными.

Недостатки:

1) Данный метод работает только с бинарными изображениями.

2) Входящие изображения должны быть нормализованы. Соотношение черных и белых пикселей должно быть примерно равно для всех изображений.

* 1. Представление пикселя полутонового изображения в виде бинарной комбинации.

Дано полутоновое изображение в виде матрицы, где интенсивность серого цвета каждого пикселя описано числом от 0 до 255 (0 – черный цвет, 255 - белый). Для вычислений преобразуем матрицу в вектор-строку и рассмотрим каждый пиксель. Пусть пиксели изображения могут иметь всего один из 3 оттенков серого. Преобразуем каждый значение вектора в двоичный код по формуле:

|  |  |
| --- | --- |
| *,* | (2.6) |

Где Code – двоичный код, характеризующий пиксель,

Grey – интенсивность серого цвета.

Достоинства:

1. Лучше всего подходит при работе с полутоновыми изображениями;
2. Входящие изображения не обязаны проходит процесс нормализации, так как количество единиц в итоговых бинарных комбинациях для полутоновых изображений с одинаковым разрешением будет одинаковым вне зависимости от самих пикселей.

Недостатки:

1. Данный метод не работает с полноцветными изображениями.
   1. Представление пикселя полноцветного изображения в виде бинарной комбинации.

Дано полноцветное изображение в виде трех матриц, каждая из которых отвечает за интенсивность красного, зеленого и синего цвета пикселя соответственно. Интенсивность любого из трех цветов описано числом от 0 до 255 включительно.

Для вычислений преобразуем матрицы в вектор-строки и рассмотрим каждый пиксель. Пусть пиксели изображения могут иметь всего один из 4 оттенков красного, зеленого и синего цвета. Преобразуем каждый значение вектора в двоичный код по формуле:

|  |  |  |
| --- | --- | --- |
|  | (2.7) | |
| *,* | | (2.8) |
| *,* | (2.9) | |
|  | (2.10) | |

где CodeRed (CodeGreen, CodeBlue) – двоичный код, характеризующий красный (зеленый, синий) цвет пикселя;

Red (Green, Blue) – интенсивность красного (зеленого, синего) цвета;  
TotalCode – двоичный код, характеризующий пиксель;

«+» – операция конкатенации.



Рис. 11. Пустой бинарный вектор при 4 оттенках красного, зеленого и синего цвета.



Рис. 12. Изменение бинарной строки при 4 оттенках красного, зеленого и синего цвета.



Рис. 13. Пример итоговой бинарной строки при 4 оттенках красного, зеленого и синего цвета.

Достоинства:

1. Лучше всего подходит при работе с полноцветными изображениями;
2. Входящие изображения не обязаны проходит процесс нормализации, так как количество единиц в итоговых бинарных комбинациях для полноцветных изображений с одинаковым разрешением будет одинаковым вне зависимости от самих пикселей.

Недостатки:

1. Неоптимален с точки зрения памяти, если на вход подаются полутоновые или бинарные изображения.

Метод 2. Двоичное кодирование интенсивности.

Дано полноцветное изображение в виде трех матриц, каждая из которых отвечает за интенсивность красного, зеленого и синего цвета пикселя соответственно. Интенсивность любого из трех цветов описано числом от 0 до 255 включительно.

Для вычислений преобразуем матрицы в вектор-строки и рассмотрим каждый пиксель. Пусть пиксели изображения могут иметь всего один из 4 оттенков красного, зеленого и синего цвета. Преобразуем каждый значение вектора в двоичный код по формуле:

|  |  |
| --- | --- |
| *,* | (2.11) |
| *,* | (2.12) |
| *,* | (2.13) |
| , | (2.14) |

где CodeRed (CodeGreen, CodeBlue) – двоичный код, характеризующий красный (зеленый, синий) цвет пикселя;

Red (Green, Blue) – интенсивность красного (зеленого, синего) цвета;  
TotalCode – двоичный код, характеризующий пиксель;

«+» – операция конкатенации.

Данный метод не подходит для решения задачи представления пикселя в виде бинарной комбинации. С одной стороны, количество единиц в бинарной комбинации становится нефиксированным, поэтому входные изображения нуждаются в нормализации. С другой, представляется слишком сложным, а также нецелесообразным при наличии доступных альтернатив, найти способ провести эту нормализацию.

Метод 3. Цифровое кодирование.

Дано полноцветное изображение в виде трех матриц, каждая из которых отвечает за интенсивность красного, зеленого и синего цвета пикселя соответственно. Интенсивность любого из трех цветов описано числом от 0 до 255 включительно.

Для вычислений преобразуем матрицы в вектор-строки и рассмотрим каждый пиксель. Пусть пиксели изображения могут иметь всего один из 4 оттенков красного, зеленого и синего цвета. Преобразуем каждый значение вектора в число по формуле:

|  |  |
| --- | --- |
| *,* | (2.15) |
| *,* | (2.16) |
| *,* | (2.17) |
|  | (2.18) |

Данный метод не подходит для решения задачи представления пикселя в виде бинарной комбинации, так как пикслель по данному методу не представляется в итоге как бинарная комбинация.

## Вывод по главе 2.

Вначале была представлена математическая модель нейрона МакКаллока-Питтса и избирательного нейрона. Были показаны отличия в устройстве нейронов. Ключевым отличием стало отсутствие надобности вычисления весовых коэффициентов у избирательного нейрона.

Затем было представлено, каким образом изображения преобразовываются в бинарные строки. Также были показаны некоторые методы вычисления бинарных строк, где лучше всего себя показал метод кодирование интенсивности цвета пикселя за счет движения единицы по длине элемента.

# 3. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПО РАСПОЗНАВАНИЮ ИЗОБРАЖЕНИЙ.

## 3.1. Обоснование выбора аппаратных и программных средств разработки

### 3.1.1. Выбор аппаратной части

Минимальные системные требования для работы программы основываются:

1. На требованиях к аппаратному и программному обеспечению, предоставляемых пакетом Matlab R2011b (указаны в таблице 3).
2. На требованиях к аппаратному обеспечению, необходимых для работы операционной системы Windows 8 (указаны в таблице 4).

**Таблица 3.**

**Системные требования Matlab R2011b**

|  |  |
| --- | --- |
| Операционная система | Windows XP Service Pack 3, Window Vista Service Pack 2, Windows 7, Windows 8 |
| Процессор | Любой процессор Intel или AMD, поддерживающий набор инструкций SSE2 |
| Оперативная память (ОЗУ) | 1024 МБ (рекомендуется 2048 МБ) |
| Жесткий диск | Не менее 1 ГБ свободного места (рекомендуется 3-4 ГБ) |

**Таблица 4.**

**Системные требования Windows 8**

|  |  |
| --- | --- |
| Процессор | 32-битный процессор (х86) с частотой 1 ГГц |
| ОЗУ | 1 ГБ |
| Жесткий диск | Около 16 ГБ свободного места |
| Дополнительно | графическое устройство Microsoft DirectX 9 с драйвером WDDM |

На основе вышеперечисленных требований определена конфигурация компьютера, необходимая для запуска разработанного программного обеспечения (указана в таблице 5).

**Таблица 5.**

**Итоговые системные требования**

|  |  |
| --- | --- |
| Процессор | 32-битный процессор (х86) с частотой 1 ГГц или быстрее |
| Оперативная память | 1024 МБ (рекомендуется 2048 МБ) |
| Жесткий диск | Не менее 17 ГБ свободного места (рекомендуется 20 ГБ или больше) |
| Дополнительно | Видеокарта с поддержкой DirectX 9 |

### 3.1.2.Выбор операционной системы

Выбор операционной системы обоснован такими условиями:

1. Удовлетворение минимальных требований пакета Matlab R2011b.
2. Популярность среди потенциальных пользователей.
3. Низкие по современным меркам требования к аппаратной конфигурации компьютеров.

Под необходимые условия лучше всего подходит операционная система Windows 8. Во-первых, она позволяет установить пакет Matlab R2011b. Во-вторых, распространена более широко, чем большинство других систем. В-третьих, самые популярные офисные пакеты поддерживают эту операционную систему. И в-четвертых, у нее относительно невысокие системные требования.

### 

### 3.1.3.Выбор среды разработки и языка программирования

Основными требованиями к среде разработки являются

1. Максимальная встроенная поддержка математических операций, применяющихся в решении задачи.
2. Высокая скорость вычислений.

Этим требованиям лучше всего удовлетворяет программный пакет Matlab R2011b с его встроенным языком программирования m-файлов.

Matlab (сокращение от англ.  «MatrixLaboratory») — пакет прикладных программ для решения задач технических вычислений и одноимённый язык программирования, используемый в этом пакете. Он обладает множеством специализированных функций и может быть использован для автоматизации практически любых вычислений. Его используют более 1 000 000 инженерных и научных работников, он работает на большинстве современных операционных систем.

Все объекты в Matlab представляются в виде массивов без явного указания размерности. Благодаря этому скорость выполнения вычислений значительно превышает возможности подобных систем.

## 3.2. Информационное обеспечение задачи

Задача программного средства состоит в распознавании изображений

В папке с программой лежит подпапки «Портреты», содержащая 50 jpg-файлов портретов; «Картины», содержащая 50 jpg-файлов картин; «ЭКГ», содержащая 10 jpg-файлов электрокардиограмм; «Портреты (описание)», содержащая txt-файлы с описаниями портретов; «Картины (описание)», содержащая txt-файлы с описаниями картины; «ЭКГ (описание)», содержащая txt-файлы с описаниями электрокардиограммы. Введение каждого изображения производится следующим образом. В этих jpg-файлах хранятся эталонные изображения. Каждое изображение и каждый файл с описанием имеет название в виде порядкового номера. Изображения имеют квадратную форму. Тексты сохранены в кодировке Windows-1251. Каждый текст состоит из одной строки.

Пример файла эталонного изображения:



Рис. 14. Пример эталонного изображения.

Пример текстового файла с описанием:

М. И. Цветаева

Успешным результатом работы программы является распознанное тестовое изображение. Изображение выводится в окне вместе с описанием. Если изображение распознать не удалось, сообщение об этом также выводится окне.

## 3.3. Работа с эталонным полноцветным изображением.

Производится необходимая работа с каждым эталонным изображением для его приведения к бинарной комбинации.

1. Читаем из jpg-файла изображение и помещаем в три различные матрицы, где каждая матрица хранит значения интенсивностей пикселей определенного цвета.

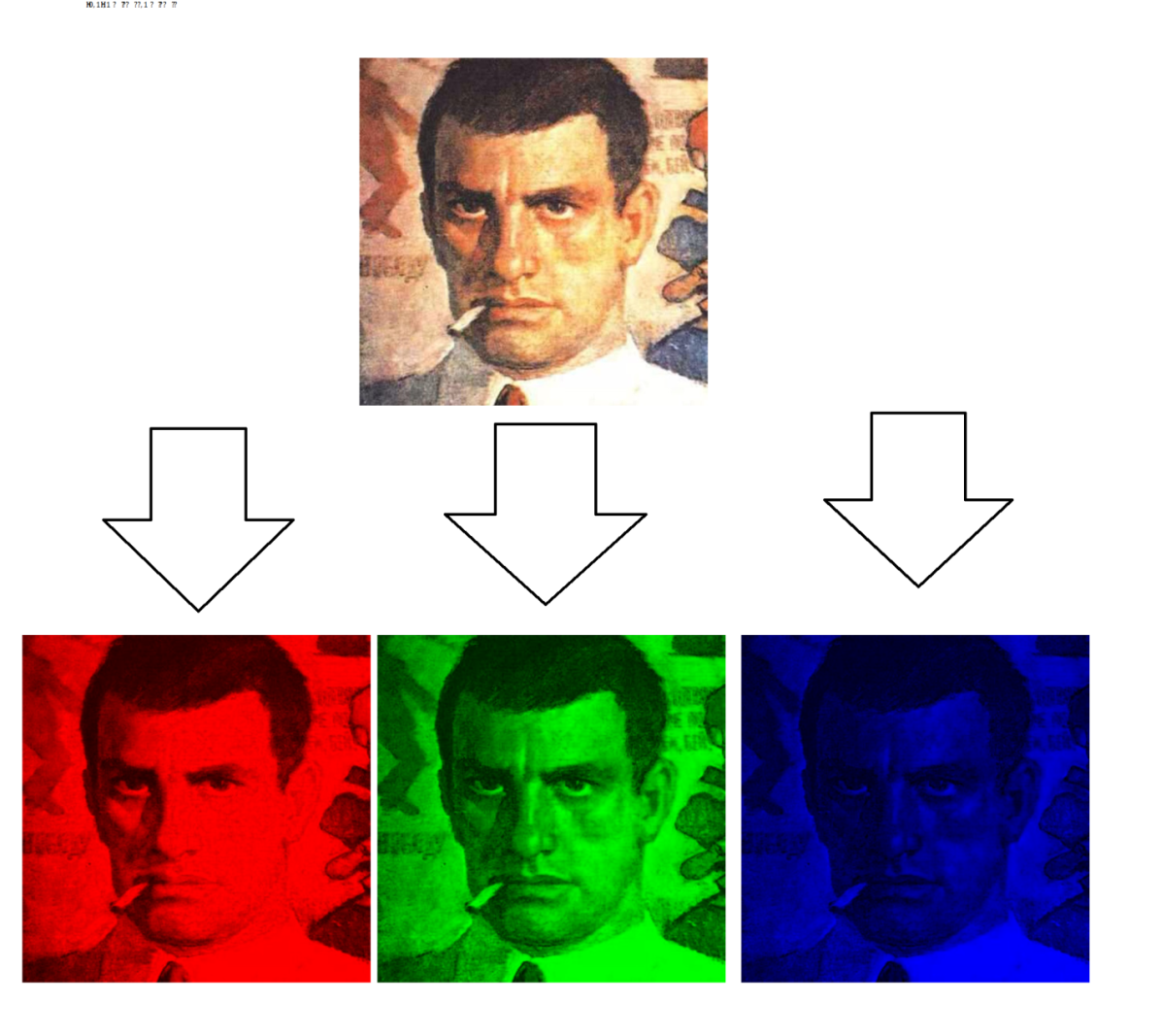


Рис. 15. Деление цветного изображение на «красную», «зеленую» и «синюю» матрицы

1. Приведение эталонных изображений к стандартной форме размера 100х100 с помощью метода ближайшего соседа (см. Приложение 1).

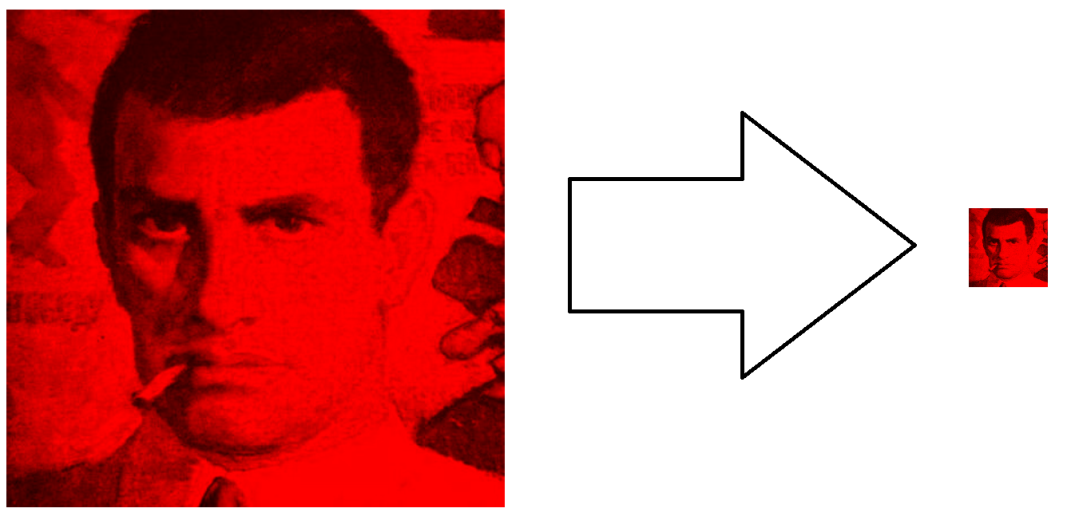


Рис.16. Приведение матрицы к размеру 100х100.

1. Приводим матрицы к вектор-строкам длиной 10000.

4) Приводим каждую вектор-строку к бинарной вектор-строке, где каждому элементу массива (пикселю) ставится в соответствие своя бинарная комбинация.

Рассмотрим метод создания бинарной последовательности на примере одного пикселя:

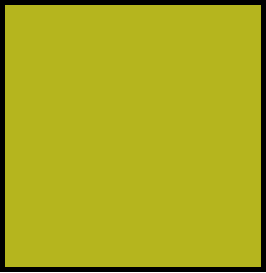


Рис. 17. Пример пикселя

Интенсивность красного цвета – 181, зеленого -181, синего – 30. Пусть предлагается разбить каждый из цветов на 4 градации. Для данного пикселя создается бинарный вектор из 12 элементов.



Рис. 18. Пустой бинарный вектор

Вначале все значения массива являются нулевыми.

Затем изменяем значение на одном из «красных» индексов (от 1 до 4) на единицу по следующей формуле.

|  |  |
| --- | --- |
|  | (3.1) |

где Index – индекс бинарного вектора, где значение меняется на единицу;

Red – интенсивность красного цвета у пикселя.

Промежуточный итог виден на рис.194:



Рис. 19. Изменение бинарной строки

Аналогичным образом изменяем значения на одном из «зеленых» (от 5 до 8) и на одном из «синих» (от 9 до 12) индексов на единицу по следующим формулам:

|  |  |
| --- | --- |
| , | (3.2) |
| , | (3.3) |

Где Green - интенсивность зеленого цвета у пикселя,

Blue - интенсивность синего цвета у пикселя.

В результате получается вектор-бинарная строка или код (рис. 20).



Рис. 20. Итоговая бинарная строка

На этом обработка эталонных изображений заканчивается.

Преимуществом данного способа получения бинарной строки является возможность получения нормированной диагонали матрицы A \* AT. На диагонали должно получаться 30 000.

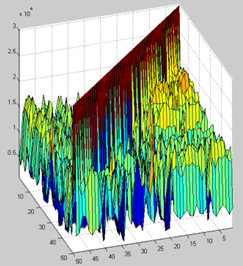


Рис.21. График матрицы A \* AT.

Из структуры фильтра видно, что пороговое значение равняется

где – максимальное значение элементов матрицы A \* AT (),

- диагональные элементы A \* AT , которые равны между собой.

Значения переменных на осях имеют следующий физический смысл: Одна ось имеет значение номера изображения, которое сравнивют; вторая ось имеет значение номера изображения, с которым сравнивают.

## 3.4. Работа с тестовым полноцветным изображением.

* 1. Запрашиваем у пользователя посредством диалогового окна тестовое изображение формата jpg для его распознавания.
  2. Производится необходимая работа с тестовым изображением для его приведения к бинарной комбинации (аналогично пункту 3.3).
  3. На основе избирательного нейрона производится сравнение тестового изображения со всеми эталонными. На основе данных сравнений находится самое похожее эталонное изображение, и происходит его вывод на экран с кратким описанием, либо выводится в окне сообщение о неудачном распознании (если порог распознания не превышен).

Пример сравнения тестового изображения с наиболее похожим на него эталонным:

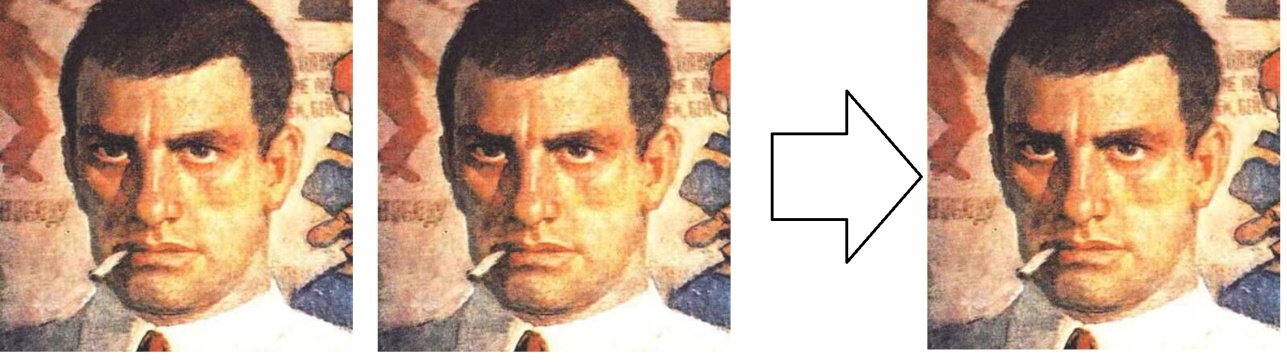


Рис. 22. Сравнение тестового изображения с наиболее похожим на него эталонным.

Пример сравнения тестового изображения с непохожим на него эталонным:

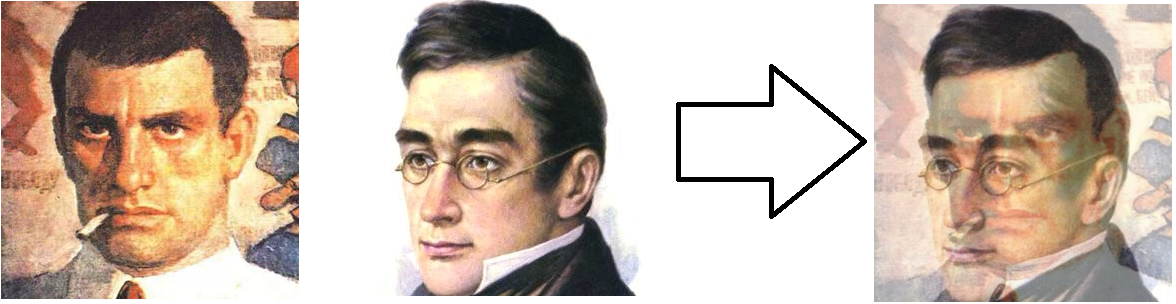


Рис. 23. Сравнение тестового изображения с непохожим на него эталонным.

## 3.5. Состав и структура программного обеспечения.

Название программных модулей, используемых в программе, показано на рисунке 29.

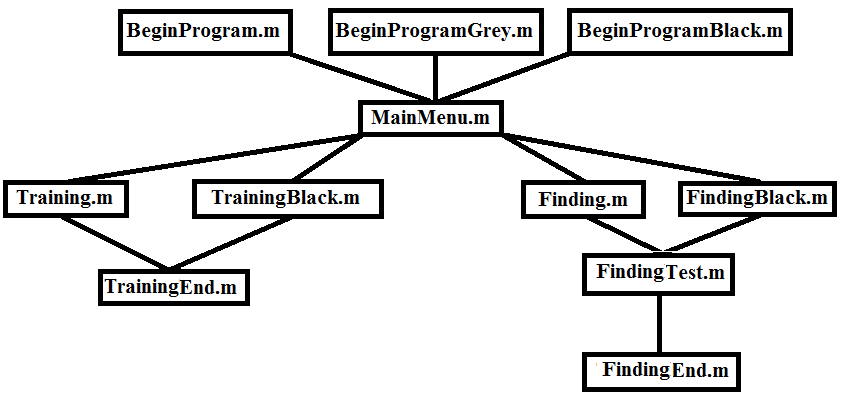
****

Рис. 24. Структура программного обеспечения.

1. Основное приложение распознавания полноцветных изображений с помощью избирательного нейрона запускается из модуля BeginProgram. Для вывода главного меню на экран из модуля BeginProgram вызывается модуль MainMenu.

1.1) Для вывода на экран меню ввода эталонного полноцветного изображения для обучения программы из модуля MainMenu вызывается модуль Training. Он запрашивает у пользователя путь к изображению, которому должна быть обучена программа, описание изображения и базу, в которую запишется само изображение с описанием. Далее из модуля Training вызывается модуль TrainingEnd. Он выводит на экран сообщение об успешном сохранении изображения с описанием в определенной базе. После из модуля TrainingEnd вызывается модуль MainMenu для возвращения в главное меню.

* 1. Для вывода на экран меню ввода тестового полноцветного изображения для его распознавания программой из модуля MainMenu вызывается модуль Finding. Он запрашивает у пользователя путь к изображению, которое необходимо распознать программой, и базу эталонных изображений, в которой предлагается найти изображение, наиболее похожее на тестовое. Далее из модуля Finding вызывается модуль FindingTest. Он выводит на экран изображение, которое необходимо распознать. Затем из модуля FindingTest вызывается модуль FindingEnd для вывода эталонного изображения из выбранной базы, наиболее похожее на тестовое. После из модуля FindingEnd вызывается модуль MainMenu для возвращения в главное меню.

1. Основное приложение распознавания полутоновых изображений с помощью избирательного нейрона запускается из модуля BeginProgramGrey. Для вывода главного меню на экран из модуля BeginProgramGrey вызывается модуль MainMenu.
   1. Для вывода на экран меню ввода эталонного полутонового изображения для обучения программы из модуля MainMenu вызывается модуль Training. Он запрашивает у пользователя путь к изображению, которому должна быть обучена программа, описание изображения и базу, в которую запишется само изображение с описанием. Далее из модуля Training вызывается модуль TrainingEnd. Он выводит на экран сообщение об успешном сохранении изображения с описанием в определенной базе. После из модуля TrainingEnd вызывается модуль MainMenu для возвращения в главное меню.
   2. Для вывода на экран меню ввода тестового полутонового изображения для его распознавания программой из модуля MainMenu вызывается модуль Finding. Он запрашивает у пользователя путь к изображению, которое необходимо распознать программой, и базу эталонных изображений, в которой предлагается найти изображение, наиболее похожее на тестовое. Далее из модуля Finding вызывается модуль FindingTest. Он выводит на экран изображение, которое необходимо распознать. Затем из модуля FindingTest вызывается модуль FindingEnd для вывода эталонного изображения из выбранной базы, наиболее похожее на тестовое. После из модуля FindingEnd вызывается модуль MainMenu для возвращения в главное меню.
2. Основное приложение распознавания контурных изображений с помощью избирательного нейрона запускается из модуля BeginProgramBlack. Для вывода главного меню на экран из модуля BeginProgramBlack вызывается модуль MainMenu.

3.1) Для вывода на экран меню ввода эталонного контурного изображения для обучения программы из модуля MainMenu вызывается модуль TrainingBlack. Он запрашивает у пользователя путь к изображению, которому должна быть обучена программа, описание изображения и базу, в которую запишется само изображение с описанием. Далее из модуля Training вызывается модуль TrainingEnd. Он выводит на экран сообщение об успешном сохранении изображения с описанием в определенной базе. После из модуля TrainingEnd вызывается модуль MainMenu для возвращения в главное меню.

3.2) Для вывода на экран меню ввода тестового полноцветного изображения для его распознавания программой из модуля MainMenu вызывается модуль Finding. Он запрашивает у пользователя путь к изображению, которое необходимо распознать программой, и базу эталонных изображений, в которой предлагается найти изображение, наиболее похожее на тестовое. Далее из модуля Finding вызывается модуль FindingTest. Он выводит на экран изображение, которое необходимо распознать. Затем из модуля FindingTest вызывается модуль FindingEnd для вывода эталонного изображения из выбранной базы, наиболее похожее на тестовое. После из модуля FindingEnd вызывается модуль MainMenu для возвращения в главное меню.

## 3.6. Описание программных модулей.

1. Модуль BeginProgram.m

Функция BeginProgram реализует работу избирательного нейрона путем обучения с помощью заранее подготовленных полноцветных изображений и соответствующих описаний к ним.

Вначале создается глобальная переменная strategy со значением 1, необходимая для других модулей, чтобы они работали в режиме работы с полноцветными изображениями.

После создается глобальная переменная palitra, обозначающая количество оттенков красного, зеленого и синего цвета, которые будут участвовать в палитре.

Затем определяются глобальные двумерные массивы, хранящие бинарные комбинации эталонных контурных изображений: portraitimages, pictureimages. Каждый из массивов хранит бинарные комбинации портретов и картин соответственно. У каждого из данных массивов первый индекс означает номер эталонного изображения, второй означает номер бинарной строки, а значение представляет собой «1» или «0».

Потом определяются глобальные четырехмерные массивы, хранящие информацию о цвете пикселей контурных эталонных изображений, готовые к отображению на экран в качестве ответа: notpreparedportraitimages, notpreparedpictureimages. Каждый из массивов хранит изображения портретов и картин соответственно. У каждого из данных массивов первый индекс означает номер эталонного изображения, второй и третий означают координаты пикселя изображения, четвертый – номер цвета (1 - красный, 2 - зеленый, 3 - синий), а значение представляет собой степень интенсивности определенного цвета пикселя (от 0 до целой части 256/palitra).

После определяются глобальные одномерные массивы, хранящие описание соответствующего изображения в виде строки: portraitanswers, pictureanswers. Каждый из массивов хранит описания портретов и картин соответственно. У каждого из данных массивов индекс означает номер эталонного изображения.

Затем вызывается функция tobk для принятия на вход полноцветных изображений и их описаний для внесения самих изображений в notpreparedportraitimages, notpreparedpictureimages; их бинарных комбинаций в portraitimages, pictureimages; их описаний в portraitanswers, pictureanswers.

[portraitimages, notpreparedportraitimages]=tobk(palitra, portraitimages,… notpreparedportraitimages, pictureanswers, 'Портреты/',…

'Портреты (описание)/');

[pictureimages, notpreparedpictureimages, pictureanswers]=tobk(palitra,… pictureimages, notpreparedpictureimages, pictureanswers, 'Картины/',…

'Картины (описание)/');

Затем происходит вызов модуля MainMenu.m.

MainMenu;

* 1. Функция tobk.

function [images, notpreparedimages, answers]=tobk(palitra, images,…

notpreparedimages,answers, imagesPath, answersPath)

На вход функции подается количество оттенков красного, зеленого и синего цвета palitra, неинициализированные массивы изображений notpreparedimages, бинарных строк images, описаний к изображениям answers и пути к директориям с jpg-файлами изображений и txt-файлами описаний imagesPath и answersPath. По завершению функции возвращаются готовые к работе массивы изображений notpreparedimages, бинарных строк images, описаний к изображениям answers.

Вначале происходит поиск jpg-изображений внутри директории imagesPath и вычисления количества найденных подходящих файлов.

MyJpgFiles=dir([imagesPath, '\*.jpg']);

MyJpgFilesQuantity=length(MyJpgFiles);

Затем происходит заполнение массива images и notpreparedimages нулями.

images = zeros(MyJpgFilesQuantity, 10000\*palitra\*3);

notpreparedimages = zeros(MyJpgFilesQuantity, 100, 100);

После происходит работа с каждым найденным эталонным изображением:

* + 1. Получение набора из трех двумерных массивов, взятых из jpg-файла. Каждый массив отвечает за интенсивность красного, зеленого и синего цвета каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

MToDisplay=imresize(imread([imagesPath, MyJpgFiles(i).name]), [100 100]);

* + 1. Перевод массивов в вектор-строки и последующий перевод в бинарную вектор-строку. Кодирование интенсивности происходит за счет движения единицы по длине элемента.

for j = 1:3

for k=1:10000

images(i,palitra\*3\*(k-1) + fix(V(1, k, j)/(256/palitra)) + palitra\*(j-1) + 1) = true;

end

end

* + 1. Сохранение изображения массива для вывода (если тестовое изображение будет распознано)

for j = 1:3

for k=1:100

for l=1:100

notpreparedimages(i,k,l,j)=fix(MToDisplay(k, l, j)/(256/palitra))\*fix(256/palitra);

end

end

end

Затем происходит поиск txt-файлов в директории answersPath и вычисление кол-ва найденных файлов.

MyTxtFiles=dir([answersPath, '\*.txt']);

MyTxtFilesQuantity=length(MyTxtFiles);

В конце происходит заполнение массива answers первыми строками из найденных файлов.

for i = 1 : MyTxtFilesQuantity

fileID = fopen([answersPath, MyTxtFiles(i).name]);

buf = textscan(fileID, '%s','delimiter','\n');

answers{i} = buf{1};

fclose(fileID);

end

1. Модуль BeginProgramGrey.m

Функция BeginProgramGrey реализует работу избирательного нейрона путем обучения с помощью заранее подготовленных полутоновых изображений и соответствующих описаний к ним.

Вначале создается глобальная переменная strategy со значением 2, необходимая для других модулей, чтобы они работали в режиме работы с полутоновыми изображениями.

После создается глобальная переменная palitra, обозначающая количество оттенков серого цвета, которые будут участвовать в палитре.

Затем определяются глобальные двумерные массивы, хранящие бинарные комбинации эталонных контурных изображений: portraitimages, pictureimages. Каждый из массивов хранит бинарные комбинации портретов и картин соответственно. У каждого из данных массивов первый индекс означает номер эталонного изображения, второй означает номер бинарной строки, а значение представляет собой «1» или «0».

Потом определяются глобальные трехмерные массивы, хранящие информацию о цвете пикселей контурных эталонных изображений, готовые к отображению на экран в качестве ответа: notpreparedportraitimages, notpreparedpictureimages. Каждый из массивов хранит изображения портретов и картин соответственно. У каждого из данных массивов первый индекс означает номер эталонного изображения, второй и третий означают координаты пикселя изображения, а значение представляет собой степень интенсивности серого цвета пикселя (от 0 до целой части 256/palitra).

После определяются глобальные одномерные массивы, хранящие описание соответствующего изображения в виде строки: portraitanswers, pictureanswers. Каждый из массивов хранит описания портретов и картин соответственно. У каждого из данных массивов индекс означает номер эталонного изображения.

Затем вызывается функция tobk для принятия на вход полутоновых изображений и их описаний для внесения самих изображений в notpreparedportraitimages, notpreparedpictureimages; их бинарных комбинаций в portraitimages, pictureimages; их описаний в portraitanswers, pictureanswers.

[portraitimages, notpreparedportraitimages]=tobk(palitra, portraitimages,… notpreparedportraitimages, pictureanswers, 'Портреты/', 'Портреты (описание)/');

[pictureimages, notpreparedpictureimages, pictureanswers]=tobk(palitra, pictureimages, notpreparedpictureimages, pictureanswers, 'Картины/','Картины (описание)/');

Затем происходит вызов модуля MainMenu.m.

MainMenu;

* 1. Функция tobk.

function [images, notpreparedimages, answers]=tobk(palitra, images,notpreparedimages,answers, imagesPath, answersPath)

На вход функции подается количество оттенков серого цвета palitra, неинициализированные массивы изображений notpreparedimages, бинарных строк images, описаний к изображениям answers и пути к директориям с jpg-файлами изображений и txt-файлами описаний imagesPath и answersPath. По завершению функции возвращаются готовые к работе массивы изображений notpreparedimages, бинарных строк images, описаний к изображениям answers.

Вначале происходит поиск jpg-изображений внутри директории imagesPath и вычисления количества найденных подходящих файлов.

MyJpgFiles=dir([imagesPath, '\*.jpg']);

MyJpgFilesQuantity=length(MyJpgFiles);

Затем происходит заполнение массива images и notpreparedimages нулями.

images = zeros(MyJpgFilesQuantity, 10000\*palitra);

notpreparedimages = zeros(MyJpgFilesQuantity, 100, 100);

После происходит работа с каждым найденным эталонным изображением:

2.1.1)Получение двумерного массива, взятого из jpg-файла. Массив отвечает за интенсивность серого цвета каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

MToDisplay= rgb2gray(imresize(imread([imagesPath, MyJpgFiles(i).name]), [100 100]));

2.1.2)Перевод массивов в вектор-строки и последующий перевод в бинарную вектор-строку. Кодирование интенсивности происходит за счет движения единицы по длине элемента.

for j = 1:3

images(i, (m-1)\*palitra + fix(V(1, m, 1)/(256/palitra)) + 1)=true;

end

2.1.3)Сохранение изображения массива для вывода (если тестовое изображение будет распознано)

for k=1:100

for l=1:100

notpreparedimages(i,k,l)=fix(MToDisplay(k, l, 1)/(256/palitra))\*fix(256/palitra);

end

end

Затем происходит поиск txt-файлов в директории answersPath и вычисление кол-ва найденных файлов.

MyTxtFiles=dir([answersPath, '\*.txt']);

MyTxtFilesQuantity=length(MyTxtFiles);

В конце происходит заполнение массива answers первыми строками из найденных файлов.

for i = 1 : MyTxtFilesQuantity

%Взятие первой строки из файла и копирование в соответствующую ячейку массива

fileID = fopen([answersPath, MyTxtFiles(i).name]);

buf = textscan(fileID, '%s','delimiter','\n');

answers{i} = buf{1};

fclose(fileID);

end

1. Модуль BeginProgramBlack.m

Функция BeginProgramBlack реализует работу избирательного нейрона путем обучения с помощью заранее подготовленных контурных изображений и соответствующих описаний к ним.

Вначале создается глобальная переменная strategy со значением 3, необходимая для других модулей, чтобы они работали в режиме работы с контурными изображениями.

Затем определяются глобальные двумерные массивы, хранящие бинарные комбинации эталонных контурных изображений: portraitimages, pictureimages, ECGimages. Каждый из массивов хранит бинарные комбинации портретов, картин и электрокардиограмм соответственно. У каждого из данных массивов первый индекс означает номер эталонного изображения, второй означает номер бинарной строки, а значение представляет собой «1» или «0».

Потом определяются глобальные трехмерные массивы, хранящие информацию о цвете пикселей контурных эталонных изображений, готовые к отображению на экран в качестве ответа: notpreparedportraitimages, notpreparedpictureimages, notpreparedECGimages. Каждый из массивов хранит изображения портретов, картин и электрокардиограмм соответственно. У каждого из данных массивов первый индекс означает номер эталонного изображения, второй и третий означают координаты пикселя изображения, а значение представляет собой цвет («1» или «0»).

После определяются глобальные одномерные массивы, хранящие описание соответствующего изображения в виде строки: portraitanswers, pictureanswers, ECGanswers. Каждый из массивов хранит описания портретов, картин и электрокардиограмм соответственно. У каждого из данных массивов индекс означает номер эталонного изображения.

Затем вызывается функция tobk для принятия на вход контурных изображений и их описаний для внесения самих изображений в notpreparedportraitimages, notpreparedpictureimages, notpreparedECGimages; их бинарных комбинаций в portraitimages, pictureimages, ECGimages; их описаний в portraitanswers, pictureanswers, ECGanswers.

[portraitimages, notpreparedportraitimages, portraitanswers] = tobk(portraitimages, notpreparedportraitimages, portraitanswers, 'Портреты/', 'Портреты (описание)/');

[pictureimages, notpreparedpictureimages, pictureanswers] = tobk(pictureimages, notpreparedpictureimages, pictureanswers, 'Картины/', 'Картины(описание)/');

[ECGimages, notpreparedECGimages, ECGanswers] = tobk(ECGimages, notpreparedECGimages, ECGanswers, 'ЭКГ/', 'ЭКГ (описание)/');

Затем происходит вызов модуля MainMenu.m.

MainMenu;

3.1)Функция tobk.

function [images, notpreparedimages, answers]=tobk(images, notpreparedimages,answers, imagesPath, answersPath)

На вход функции подается неинициализированные массивы изображений notpreparedimages, бинарных строк images, описаний к изображениям answers и пути к директориям с jpg-файлами изображений и txt-файлами описаний imagesPath и answersPath. По завершению функции возвращаются готовые к работе массивы изображений notpreparedimages, бинарных строк images, описаний к изображениям answers.

Вначале происходит поиск jpg-изображений внутри директории imagesPath и вычисления количества найденных подходящих файлов.

MyJpgFiles=dir([imagesPath, '\*.jpg']);

MyJpgFilesQuantity=length(MyJpgFiles);

Затем происходит заполнение массива images и notpreparedimages нулями.

notpreparedimages = zeros(MyJpgFilesQuantity, 100, 100);

После происходит работа с каждым найденным эталонным изображением:

3.1.1) Получение двумерных массивов, взятых из jpg-файла. Каждый массив отвечает за наличие/отсутствие белого цвета каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

MToDisplay=im2bw(imresize(imread([imagesPath,MyJpgFiles(i).name]), [100 100]), 0.5);

3.1.2) Перевод массива в вектор-строку. Так как она уже бинарная, то перевод в бинарную строку не требуется. Но, так как важная информация, например, в ЭКГ черного цвета, а фон – белого, то происходит инвертирование бинарной строки (черному пикселю теперь соответствует «1», белому – «0»).

for m=1:1:10000

images(i, m)=~V(1,m);

end

3.1.3) Сохранение изображения массива для вывода (если тестовое изображение будет распознано)

notpreparedimages(i, :, :)=MToDisplay;

Затем происходит поиск txt-файлов в директории answersPath и вычисление кол-ва найденных файлов.

MyTxtFiles=dir([answersPath, '\*.txt']);

MyTxtFilesQuantity=length(MyTxtFiles);

В конце происходит заполнение массива answers первыми строками из найденных файлов.

for i = 1 : MyTxtFilesQuantity

fileID = fopen([answersPath, MyTxtFiles(i).name]);

buf = textscan(fileID, '%s','delimiter','\n');

answers{i} = buf{1};

fclose(fileID);

end

1. Модуль MainMenu.m

Модуль выводит на экран меню с двумя кнопками: «Добавить изображение для обучения» и «Распознавать картинку».

4.1) При нажатии на кнопку «Добавить изображение для обучения» запускается модуль Training.m (при работе с полноцветными и полутоновыми изображениями) или TrainingBlack.m (при работе с контурными изображениями).

4.2) При нажатии на кнопку «Распознавать картинку» запускается модуль Finding.m (при работе с полноцветными и полутоновыми изображениями) или FindingBlack.m (при работе с контурными изображениями).

1. Модуль Training.m

Модуль Training.m реализует дополнительное обучение нейронной сети для работы с полноцветными и полутоновыми изображениями. Модуль запрашивает у пользователя путь к контурному изображению для обучения нейронной сети, описания изображения и базу данных, в которую следует добавить данное изображение с описанием. После получения необходимых данных происходит запуск функции addimage, которая на их основе добавляет изображение в нейронную сеть.

function addimage(images, notpreparedimages, answers, imagepath, descriptionpath, handles)

Функция addimage получает на вход массив бинарных строку images, массив изображений notpreparedimages, массив описаний answers, путь к директории, где хранятся изображения imagepath и путь к директории, где хранятся описания descriptionpath.

Если идет работа с полутоновыми изображениями, то вначале происходит получение двумерного массива, взятого из jpg-файла. Если идет работа с полноцветными изображениями, то вначале происходит получение набора из трех двумерных массивов, взятых из jpg-файла. Массив отвечает за интенсивность цветов каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

switch strategy

case 1

MToDisplay=imresize(imread(get(handles.PathEdit, 'String')), [100 100]);

case 2

MToDisplay=rgb2gray(imresize(imread(get(handles.PathEdit, 'String')), [100 100]));

end

Затем идет перевод массива в вектор-строку, а полученная вектор-строка переводится в бинарную вектор-строку. Кодирование интенсивности происходит за счет движения единицы по длине элемента.

Потом происходит добавление бинарной строки в общую матрицу бинарных строк images, изображения в общую матрицу изображений notpreparedimages, описания в общую матрицу описаний answers и исходных jpg- и txt-файлов в соответствующие директории.

В конце происходит переход к модулю TrainingEnd.m

1. Модуль TrainingBlack.m

Модуль TrainingBlack.m реализует дополнительное обучение нейронной сети для работы с контурными изображениями. Модуль запрашивает у пользователя путь к контурному изображению для обучения нейронной сети, описания изображения и базу данных, в которую следует добавить данное изображение с описанием. После получения необходимых данных происходит запуск функции addimage, которая на их основе добавляет изображение в нейронную сеть.

function addimage(images, notpreparedimages, answers, imagepath, descriptionpath, handles)

Функция addimage получает на вход массив бинарных строку images, массив изображений notpreparedimages, массив описаний answers, путь к директории, где хранятся изображения imagepath и путь к директории, где хранятся описания descriptionpath.

Вначале происходит получение двумерного массива, взятых из jpg-файла. Массив отвечает за наличие/отсутствие белого цвета каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

MToDisplay=im2bw(imresize(imread(get(handles.PathEdit, 'String')), [100 100]), 0.5);

Затем идет перевод массива в вектор-строку. Так как она уже бинарная, то перевод в бинарную строку не требуется. Но, так как важная информация в ЭКГ черного цвета, а фон – белого, то происходит инвертирование бинарной строки (черному пикселю теперь соответствует «1», белому – «0»).

BufBinary = zeros(1, 10000);

for m=1:1:10000

BufBinary (i, m)=~V(1,m);

end

Потом происходит добавление бинарной строки в общую матрицу бинарных строк images, изображения в общую матрицу изображений notpreparedimages, описания в общую матрицу описаний answers и исходных jpg- и txt-файлов в соответствующие директории.

images(size(images, 1)+1, :) = BufBinary(1, :);

notpreparedimages(size(notpreparedimages, 1)+1, :, :)=MToDisplay;

answers(length(answers)+1)=get(handles.DescriptionEdit, 'String');

imwrite(MToDisplay, [imagepath, num2str(size(notpreparedimages, 1)), '.jpg']);

fileID = fopen([descriptionpath, num2str(length(answers)), '.txt'], 'a');

cell=answers(length(answers));

fprintf(fileID, '%s' , cell{1});

fclose(fileID);

В конце происходит переход к модулю TrainingEnd.m

1. Модуль TrainingEnd.m.

Модуль выводит на экран окно с уведомлением «Изображение сохранено» и вызывает модуль MainMenu.m для перехода в главное меню.

1. Модуль Finding.m.

Модуль Finding.m реализует распознавание тестового полноцветного или полутонового изображения. Модуль запрашивает у пользователя путь к полноцветному или полутоновому изображению для распознавания, и базу эталонных изображений, среди которых следует искать данное изображение. После получения необходимых данных происходит следующее:

8.1) Получение двумерных массивов, взятых из jpg-файла. Каждый массив отвечает за интенсивность своего цвета каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

switch strategy

case 1

notpreparedtest=imresize(imread(get(handles.PathEdit, 'String')), [100 100]);

case 2

notpreparedtest=imresize(rgb2gray(imread(get(handles.PathEdit, 'String'))), [100 100]);

end

8.2) Перевод массива в вектор-строку. Кодирование интенсивности происходит за счет движения единицы по длине элемента.

8.3) Выбор базы изображений, среди которых будет идти поиск наиболее подходящего эталонного изображения и перемножение матрицы эталонных бинарных строк и тестовой бинарной строки.

if get(handles.PortraitRadio, 'Value') == 1

res = portraitimages \* testimage';

imagekind='portrait';

elseif

res = pictureimages \* testimage';

imagekind='picture';

end

Результатом будет вектор-столбец, где каждое значение будет хранить количество пикселей тествого изображения, совпадающих с соответствующими пикселями эталонных изображений.

8.4) Поиск максимального элемента в полученном векторе, т. е. наиболее схожего эталонного изображения с тестовым.

max = 0;

imax = 0;

for i = 1 : length(res)

if res(i) > max

max = res(i);

imax = i;

end;

end

8.5) Сохранение номера наиболее похожего эталонного изображения, если порог превышен. Иначе сохраняем нулевой номер (отсутствие похожего эталонного изображения).

global ImageToShow;

if max > threshold

ImageToShow=imax;

else

ImageToShow=0;

end

В конце происходит вызов модуля FindingTest.m.

1. Модуль FindingBlack.m.

Модуль FindingBlack.m реализует распознавание тестового контурного изображения. Модуль запрашивает у пользователя путь к контурному изображению для распознания, и базу эталонных изображений, среди которых следует искать данное изображение. После получения необходимых данных происходит следующее:

9.1) Получение двумерного массива, взятого из jpg-файла. Массив отвечает за наличие/отсутствие белого цвета каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

notpreparedtest=imresize(im2bw(imread(get(handles.PathEdit, 'String')), 0.5), [100 100]);

9.2) Перевод массива в вектор-строку. Так как она уже бинарная, то перевод в бинарную строку не требуется. Но, так как важная информация в ЭКГ черного цвета, а фон – белого, то происходит инвертирование бинарной строки (черному пикселю теперь соответствует «1», белому – «0»).

V(:,:)=reshape(notpreparedtest, 1, 10000);

testimage = zeros(1, 10000);

for m=1:1:10000

testimage(1, m)=~V(1,m);

end

9.3) Выбор базы изображений, среди которых будет идти поиск наиболее подходящего эталонного изображения и перемножение матрицы эталонных бинарных строк и тестовой бинарной строки.

if get(handles.PortraitRadio, 'Value') == 1

res = portraitimages \* testimage';

imagekind='portrait';

elseif get(handles.PictureRadio, 'Value') == 1

res = pictureimages \* testimage';

imagekind='picture';

else

res = ECGimages \* testimage';

imagekind='ECG';

end

Результатом будет вектор-столбец, где каждое значение будет хранить количество пикселей тествого изображения, совпадающих с соответствующими пикселями эталонных изображений.

9.4) Поиск максимального элемента в полученном векторе, т. е. наиболее схожего эталонного изображения с тестовым.

max = 0;

imax = 0;

for i = 1 : length(res)

if res(i) > max

max = res(i);

imax = i;

end;

end

9.5) Сохранение номера наиболее похожего эталонного изображения, если порог превышен. Иначе сохраняем нулевой номер (отсутствие похожего эталонного изображения).

global ImageToShow;

if max > threshold

ImageToShow=imax;

else

ImageToShow=0;

end

В конце происходит вызов модуля FindingTest.m.

1. Модуль FindingTest.m.

Модуль выводит на экран окно с тестовым изображением и уведомлением «Test Picture» и переходит к модулю FindingEnd.m.

1. Модуль FindingEnd.m.

Модуль выводит на экран окно с наиболее похожим эталонным изображением с соответствующим описанием и переходит к модулю MainMenu.m. для возврата к главному меню.

Вначале в зависимости от того, какая база изображений использовалась для распознавания тестового, берутся соответствующие массивы бинарных строк images и массивы изображений notpreparedimages.

switch imagekind

case 'portrait'

notpreparedimages=notpreparedportraitimages;

answers=portraitanswers;

case 'picture'

notpreparedimages=notpreparedpictureimages;

answers=pictureanswers;

case 'ECG'

notpreparedimages=notpreparedECGimages;

answers=ECGanswers;

end

Если тестовое изображение распознано, то в зависимости от того, с какой цветностью изображений работает программа выводится либо полноцветное, либо полутоновое, либо контурное эталонное изображение с описанием. Иначе выводится сообщение 'Image not found'.

## Руководство пользователя программного обеспечения по распознаванию изображения.

1. Запуск программы.

1.1)Запустить MATLAB R2011b;

* 1. Настроить директорию, в которой находится данное ПО;

Директория

Рис. 30. Директория, в которой расположено ПО.

* 1. Ввести в консоли BeginProgram и нажать Enter;

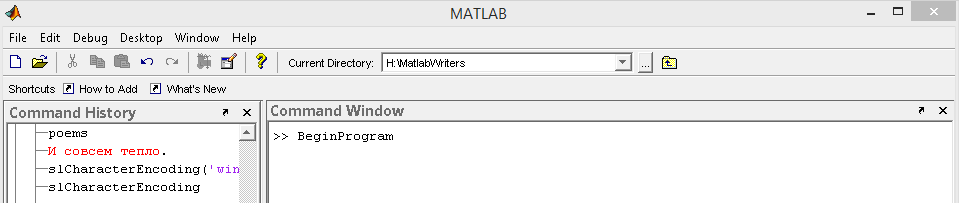


Рис. 25. Ввод в консоли команды запуска.

* 1. Программа запущена;

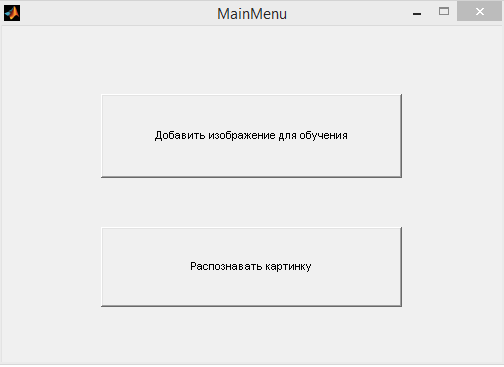


Рис. 26. Окно главного меню.

2) Добавление изображения для обучения.

* 1. Нажать кнопку «Добавить изображение для обучения»;

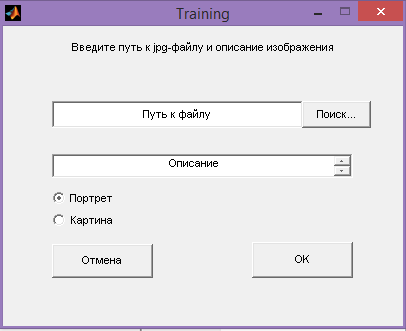


Рис. 27. Окно ввода изображения для обучения.

* 1. C помощью кнопки «Поиск…» найти нужное изображение для обучения программы. После успешного поиска путь к файлу отобразится в поле «Путь к файлу»;

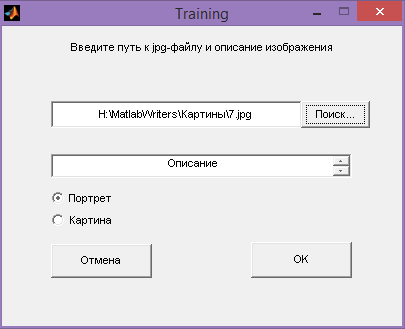


Рис. 28. Окно ввода изображения для обучения с введенным путем.

* 1. В поле «Описание» ввести описание изображения для запоминания. Выбрать, в какую базу изображений сохранить данное изображения посредством выбора одного из вариантов «Портрет» и «Картина» и нажать «ОК»;
  2. Выведется окно, подтверждающее, что данное изображение сохранено.

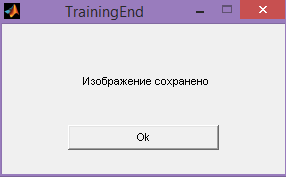


Рис. 29. Окно для подтверждения сохранения изображения в определенной базе.

3) Распознавание тестового изображения.

* 1. Нажать кнопку «Распознавать картинку»;
  2. C помощью кнопки «Поиск…» найти нужное изображение для распознавания программой. После успешного поиска путь к файлу отобразится в поле «Путь к файлу»;

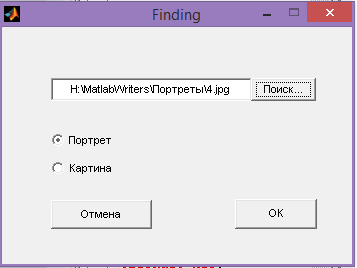


Рис. 30. Окно ввода изображения для его распознания.

* 1. Выбрать, среди эталонных изображений какой базы программа будет искать тестовое изображение посредством выбора одного из вариантов «Портрет» и «Картина» и нажать «ОК»;
  2. Выведется окно с изображением, которое требуется распознать. Нажать «ОК».

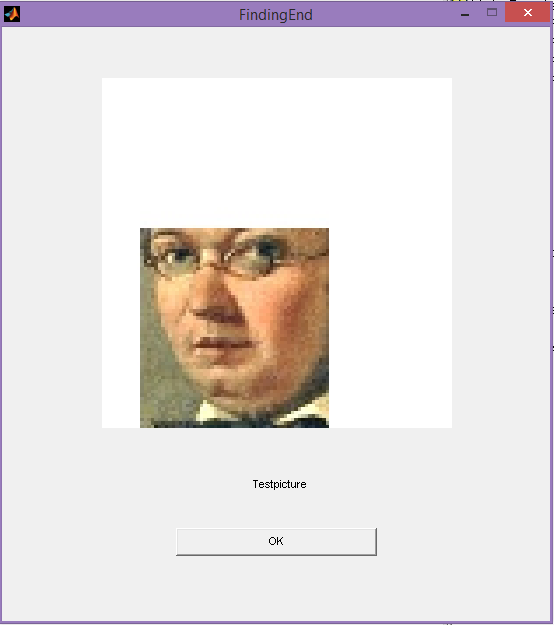


Рис. 31. Окно с изображением, которое нужно распознать.

* 1. Выведется окно, подтверждающее, что данное изображение распознано программой в виде изображения из базы, на которое тестовое изображение больше всего похоже, с соответствующим описанием.

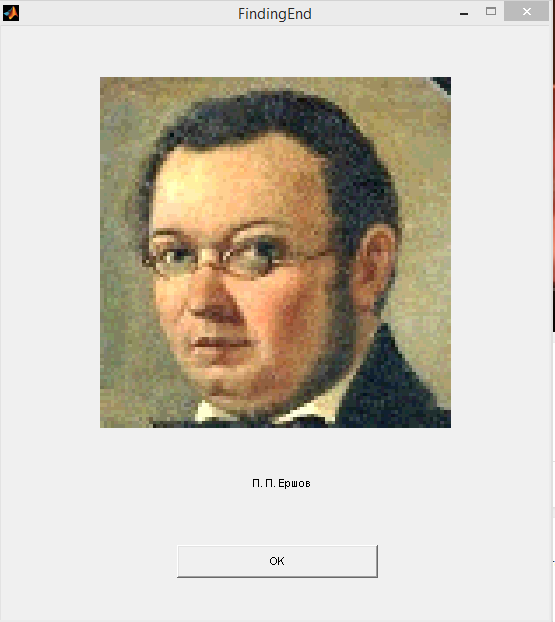


Рис. 32. Окно с изображением, наиболее похожим на тестовое, с соответствующим описанием.

## Вывод по главе 3.

В начале 3 главы был указан и обоснован выбор Windows 8 в качестве операционной системы для разработки ПО для распознавания полноцветных, полутоновых и контурных изображений. Также для решения данной задачи был указан и обоснован выбор Matlab R2011b в качестве среды разработки и Matlab в качестве языка программирования. На этой основе была определена конфигурация компьютера, необходимая для запуска разработанного программного обеспечения.

Затем было описано информационное обеспечение задачи, где было указано, в каком виде хранится база изображений портретов, картин и электрокардиограмм вместе с описанием к данным изображениям.

Потом были указаны и описаны алгоритмы, которые использовались при разработке программного обеспечения:

1) алгоритм работы с эталонными изображениями квадратной формы формата jpg (обучение);

2) алгоритм работы с тестовым изображением квадратной формы формата jpg (распознавание);

3) метод ближайшего соседа (масштабирование изображений).

Затем были описаны состав и структура программного обеспечения, где было проведено подробное описание поведения отдельных модулей и взаимодействия данных модулей между собой.

В конце 3 главы была представлена инструкция пользователя ПО для его запуска, добавления изображения в определенную базу изображений и распознавания тестового изображения.

# 4. ЭКСПЕРИМЕНТАЛЬНАЯ ПРОВЕРКА И ОЦЕНКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ НА ИЗБИРАТЕЛЬНЫХ НЕЙРОНАХ.

## 4.1. Описание результатов испытаний

Испытания проводятся для того, чтобы убедиться, что разработанное программное обеспечение удовлетворяет требованиям поставленной задачи.

При испытании были проведены следующие эксперименты.

1. Распознавание писателя по изображению.
2. Распознавание картины по ее изображению.
3. Распознавание электрокардиограммы по ее изображению
4. Распознавание писателя по изображению, отсутствующего в базе.

При проведении перечисленных экспериментов были получены следующие результаты.

1. Распознавание писателя по изображению происходит верно.
2. Распознавание картины по ее изображению происходит верно.
3. Распознавание электрокардиограммы по ее изображению происходит верно.
4. Программа выдает сообщение о том, что введенное изображение отсутствует в базе.

Во время проведения испытаний программа работала без сбоев.

Результаты проведенных испытаний говорят о том, что разработанное программное обеспечение удовлетворяет поставленным требованиям. Программа успешно обработала некорректно введенные данные и работала стабильно, без аварийных завершений и другого непредвиденного поведения. Обозначенные в постановке задачи на проектирование требования удовлетворены в полном объеме.

## 4.2. Оценивание надежности программного обеспечения

Надежность программного обеспечения – это его способность с достаточно большой вероятностью безотказно выполнять определенные функции в течение заданного периода времени при заданных условиях. Считается, что в программе произошел отказ, если она не выполнила функции, предусмотренные в техническом задании. Однако наличие ошибок в программе не всегда приводит к отказу. Программа считается надежной, если при ее практическом использовании в требуемых условиях отказы происходят достаточно редко. Убедиться в надежности программы можно при проведении тестирования, а впоследствии и при использовании в практических целях.

Для подсчета количественных показателей надежности в данной квалификационной работе будем использовать модель Коркорэна. Она относится к статическим моделям надежности ПО. Модель рассчитывается на основе N испытаний, в которых произошло Ni ошибок i-го типа, и не связывает возникновение отказов со временем. Для различных типов ошибок модель учитывает изменяющиеся вероятности отказов. Модель Коркорэна выгодно отличается от других моделей тем, что ее относительно легко рассчитать.

В модели Коркорэна для оценки вероятности безотказного выполнения программы используется следующая формула:

|  |  |
| --- | --- |
|  | (4.1) |

где R – показатель уровня надежности,

N0 –число безотказных выполнений программы,

N – общее число выполнений,

K – известное заранее число типов ошибок,

Ni – количество ошибок i-го типа,

ai – вероятность появления ошибки i-го типа.

В модели Коркорэна вероятность ai определяется на основе заранее известных результатов работы схожих по функциональности программных средств.

При тестировании программы было выполнено 85 прогонов. Число ошибок каждого типа и вероятности их появления указаны в таблице 3. 73 испытание из 80 было произведено успешно. Всего во время проведения испытаний произошло 12 отказов программы, вызванных одной ошибкой вычисления, одной логической ошибкой, четыренмя ошибками ввода-вывода, тремя ошибками манипулирования данными и тремя ошибками в базе данных.

**Таблица 3.**

**Выявленные ошибки**

|  |  |  |
| --- | --- | --- |
| Тип ошибки | Вероятность появления (ai) | Количество выявленных ошибок (Ni) |
| Ошибки вычисления | 0.09 | 1 |
| Логические ошибки | 0.26 | 1 |
| Ошибки ввода-вывода | 0.16 | 4 |
| Ошибки манипулирования данными | 0.18 | 3 |
| Ошибки сопряжения | 0.17 | 0 |
| Ошибки определения данных | 0.08 | 0 |
| Ошибки в БД | 0.06 | 3 |

Вычислим показатель надежности программы, используя вышеприведенную формулу. В нашем случае N0 = 73, N = 85, K = 7. Отсюда получаем вероятность безотказной работы программы

Таким образом, вероятность того, что программа будет работать без сбоев, равна 87,01%. Такое высокое значение говорит о том, что программа для распознавания текстов с помощью нейронных сетей на избирательных нейронах полностью готова к практическому использованию. Программное обеспечение показало высокую степень надежности.

## 4.3. Обоснование экономической эффективности

### 4.3.1. Методика расчета экономической эффективности

Каждый проект внедрения программного обеспечения обработки информации должен быть обоснован расчетом его ожидаемой экономической эффективности.

Для расчета экономической эффективности внедрения данного программного обеспечения сравним затраты при двух вариантах решения задачи:

1) Базовый вариант - работник компании решает задачу собственными силами без использования специально разработанного для этих целей программного средства.

2) Проектируемый вариант - для решения задачи используется программное обеспечение из данной квалификационной работы.

Экономическая эффективность подразделяется на две составляющие: косвенный эффект и прямой эффект.

Косвенный эффект сложно подсчитать, так как он по большей части характеризуется улучшением качественных показателей, например, снижением затрат на сырье и материалы, уменьшением процента брака, привлечением большего числа клиентов, повышением прибыли, уменьшением сумм штрафов. Из-за этого осложнено выявление доли косвенного эффекта от программной обработки данных в общей экономической эффективности, тесно связанной с комплексом мер по повышению производительности компании.

Прямой эффект, в свою очередь, легко формализуется, потому что он отражает экономию материально-трудовых ресурсов и денежных средств, полученную в результате автоматизации решения задач.

К трудовым относятся следующие показатели.

1) Абсолютный показатель снижения трудовых затрат - разность между годовыми трудовыми затратами базового и проектируемого вариантов обработки данных.

|  |  |
| --- | --- |
|  | (4.2) |

где Tб – годовая трудоемкость обработки информации по базовому варианту,

Tп – годовая трудоемкость обработки информации по проектируемому варианту.

2) Коэффициент снижения трудовых затрат отображает значение относительного снижения трудовых затрат.

|  |  |
| --- | --- |
| . | (4.3) |

3) Индекс снижения трудовых затрат отображает повышение производительности труда по причине внедрения автоматизированной обработки данных.

|  |  |
| --- | --- |
|  | (4.4) |

К стоимостным показателям относятся:

1) Экономия финансовых затрат за счет автоматизации решения задачи рассчитывается как разница между затратами по базовому и проектируемому вариантами.

|  |  |
| --- | --- |
|  | (4.5) |

где Сб – стоимостные затраты на решение задачи по базовому варианту,

Сп – стоимостные затраты на решение задачи по второму варианту.

Стоимостные затраты на обработку информации по проектируемому варианту рассчитываются по формуле

|  |  |
| --- | --- |
|  | (4.6) |

где Цмч – стоимость машинного часа,

Зч – средняя зарплата оператора за один час,

Кнр – коэффициент накладных расходов.

1) Коэффициент относительного снижения стоимостных затрат отображает, на сколько процентов снизятся затраты при переходе на предлагаемый вариант решения задачи.

|  |  |
| --- | --- |
|  | (4.7) |

2) Индекс снижения затрат показывает, во сколько раз снизятся затраты на решение задачи после внедрения проектируемого варианта.

|  |  |
| --- | --- |
|  | (4.8) |

Кроме показателей, перечисленных выше, будет также полезно подсчитать и другие.

1) Годовой экономический эффект равен сумме прямого и косвенного эффектов.

|  |  |
| --- | --- |
|  | (4.9) |

где Эп – прямой экономический эффект,

Эк – косвенный экономический эффект.

Прямая составляющая экономической эффективности рассчитывается по формуле

|  |  |
| --- | --- |
|  | (4.9) |

где Ен – нормативный коэффициент эффективности вложений капитала,

Кб – затраты капитала при базовом варианте обработки информации,

Кп – затраты капитала при проектируемом варианте обработки информации,

∆К – дополнительные затраты капитала при миграции с базового варианта на проектируемый.

2) Расчетный коэффициент эффективности равен отношению абсолютного снижения стоимостных затрат к дополнительным капитальным вложениям на переходный период.

|  |  |
| --- | --- |
|  | (4.10) |

3) Время окупаемости вложений на внедрение проекта автоматизированной обработки информации обратно пропорционально расчетному коэффициенту эффективности.

|  |  |
| --- | --- |
|  | (4.11) |

### 4.3.2. Расчет показателей экономической эффективности

Используем информацию, приведенную выше, для подсчета показателей экономической эффективности программного обеспечения, разработанного в рамках данной выпускной квалификационной работы. В первую очередь определим трудовые ресурсы и денежные средства, затрачиваемые на распознавание электрокардиограмм при базовом и предлагаемом вариантах. Рассчитанные экономические показатели представим в виде таблиц и графиков для удобства восприятия.

В год сотруднику требуется распознать около тысячи электрокардиограмм. До внедрения предлагаемого программного средства на каждое распознавание он тратил в среднем 10 минут. Таким образом, годовая трудоемкость решения задачи по базовому варианту равна

Зарплата сотрудника составляет 45000 рублей в месяц или приблизительно 1125 рублей в час. Отсюда получаем годовые стоимостные затраты на решение задачи по базовому варианту

C использованием разработанного в рамках данного квалификационной работы программного обеспечения для распознавания изображений затраты материально-трудовых ресурсов и денежных средств снизятся.

Трудовые затраты сотрудника, решающего задачу по проектируемому варианту включают в себя две составляющие. Во-первых, время на ознакомление с программным средством, изучение способов работы с ним составит 80 минут. Во-вторых, время, затрачиваемое на распознавание одного художественного произведения, составит в среднем 1 минуту. Отсюда получаем годовую трудоемкость решения по проектируемому варианту

Теперь посчитаем абсолютные показатель снижения трудовых затрат.

Коэффициент относительного снижения трудовых затрат равен

И наконец индекс снижения трудовых затрат получаем равным приблизительно

Теперь рассчитаем стоимостные экономические показатели. Нам уже известно, что сотрудник при работе по предлагаемому варианту в год на решение задачи затрачивает 18 часов. Допустим, что при переходе на автоматизированную обработку данных его зарплата не изменится. Стоимость машинного часа примем равной 10 рублям и возьмем стандартное значение коэффициента накладных расходов в 0,1. Стоимостные затраты на решение задачи по проектируемому варианту будут равны

Теперь рассчитаем абсолютное снижение стоимостных затрат.

Коэффициент относительного снижения стоимостных затрат равен

В конце получим индекс снижения стоимостных затрат

В таблице 4 представлены все вычисленные трудовые и стоимостные показатели экономической эффективности перехода на предлагаемый вариант решения задачи с использованием программного обеспечения для распознавания портретов.

Следующим шагом рассчитаем оставшиеся показатели экономической эффективности всего проекта. Сначала вычислим годовой экономический эффект. Для этого определим себестоимость разработанного в рамках данной квалификационной работы программного обеспечения. Она складывается из капитальных вложений на покупку, поддержку и эксплуатацию программно-аппаратного комплекса, использованного при разработке, и расходов на заработную плату разработчиков. Затраты на заработную плату разработчикам R вычислим по формуле

|  |  |
| --- | --- |
|  | (4.12) |

где n – число разработчиков, участвовавших в проекте,

C – заработная плата специалиста (в час),

T – количество человеко-часов, потребовавшихся для разработки данной квалификационной работы.

Программа разрабатывалась в течение трех месяцев с участием одного разработчика, заработная плата которого составляла 20000 рублей в месяц или примерно 500 рублей в час. Трудоемкость разработки составила 504 человека-часа. Расходы на заработную плату разработчиков за весь период равны

Единовременные капитальные вложения считаем равными K = 8000 рублей. Отсюда получаем суммарную себестоимость данного программного средства:

**Таблица 5.**

**Показатели экономической эффективности.**

|  |  |  |
| --- | --- | --- |
| Тип показателя | Трудовые показатели | Стоимостные показатели |
| Затраты при базовом варианте решения задачи | 167 часов | 187875 руб. |
| Затраты при проектируемом варианте решения задачи | 18 часов | 22455 руб. |
| Абсолютное снижение затрат | 149 часа | 141170 руб. |
| Относительное снижение затрат | 89 % | 75,1% |
| Индекс снижения затрат | 9,3 раз | 8,4 раз |

Себестоимость использования базового варианта считаем равным нулю. Тогда дополнительные затраты капитала при миграции с базового варианта на проектируемый равны

Нормативный коэффициент эффективности считаем равным Ен = 0,15. Теперь можем рассчитать прямой экономический эффект. Он равен

Косвенный экономический эффект от миграции на автоматизированный вариант обработки информации проявляется, например, из-за сокращения количества ошибок в расчетах. В итоге мы получили, что годовой экономический эффект равен

Определим теперь расчетный коэффициент эффективности.

Время окупаемости вложений составит

Так как годовой экономический эффект получился достаточно невысоким, требуется принимать во внимание снижение трудовых затрат сотрудников, занятых решением задачи, а также учитывать косвенный экономический эффект, который непросто формализовать и выразить в цифрах, как уже было сказано выше. По итогам общей оценки рассчитанных значений экономических показателей можно сделать вывод о том, что с экономической точки зрения переход на использование разработанного программного средства эффективен.

Диаграммы снижения трудовых и стоимостных затрат на распознавание портретов при базовом и предлагаемом вариантах приведены на рисунках 33 и 34.

Рис. 33. Диаграмма снижения трудовых затрат

Рис. 34. Диаграммы снижения стоимостных затрат

Из этих диаграмм хорошо видно, что относительные снижения трудовых и стоимостных затрат примерно равны между собой. Так происходит по причине того, что основная часть затрат на решение задачи представляет собой расходы на заработную плату сотрудников.

## Вывод по главе 4

Четвертая глава выпускной квалификационной работы посвящена экспериментальному тестированию и оценке экономической эффективности разработанного программного обеспечения для распознавания изображений с помощью нейронных сетей на избирательных нейронах.

В ходе проведения тестирования не было зафиксировано сбоев в работе программы. Программа работала корректно, выдавала правильные результаты, на введенные пользователем данные отвечала согласно ожиданиям.

Была произведена оценка надежности программного обеспечения по модели Коркорэна. После всех вычислений коэффициент надежности программы получился равным примерно 87,01%. Такой относительно высокий результат говорит о готовности программы к внедрению в практическую эксплуатацию.

В последнем параграфе были вычислены основные показатели экономической эффективности программного обеспечения. Эти вычисления заключались, главным образом, в сравнении между собой затрат при двух вариантах решения задачи:

1) Базовый вариант - работник компании решает задачу собственными силами без использования специально разработанного для этих целей программного средства.

2) Проектируемый вариант - для решения задачи используется программное обеспечение из данной выпускной квалификационной работы.

В результате проведенных исследований было вынесено решение о том, что разработанное программное обеспечение для распознавания изображений с помощью нейронных сетей на избирательных нейронах имеет высокую экономическую эффективность.

# ЗАКЛЮЧЕНИЕ

Итогом написания данной выпускной квалификационной работы является решение задачи разработки программного обеспечения для распознавания изображений при помощи нейронных сетей на избирательных нейронах. Разработанная программа удовлетворяет всем условиям, поставленным в момент постановки задачи. Сформулированная цель данной ВКР полностью достигнута.

Исследования возможностей существующего программного обеспечения, которое базируется на нейроне Маккалока – Питтса, показало необходимость разработки новой программы, которая будет обладать рядом преимуществ и не будет иметь недостатков предшественников.

Разработанная программа была протестирована и отлажена, а также была подвергнута оценке надежности и экономической эффективности.

В результате вычислений установлена высокая вероятность того, что разработанная программа будет работать без сбоев. Сделан вывод, что программа для распознавания изображений с помощью нейронных сетей на избирательных нейронах готова к практической эксплуатации.

На основании результатов вычисления основных экономических показателей вынесено заключение о высокой экономической эффективности разработанного в рамках данной ВКР программного обеспечения.

Программное обеспечение распознает портреты, картины и электрокардиограммы из числа сохраненных в ее базе. При этом для успешного распознавания достаточно ввести тестовое изображение и указать, в какой базе изображений стоит проводить поиск наиболее похожих эталонов.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Александров Ю.И., Анохин К.В., Соколов Е.Н., Греченко Т.Н. и др. Нейрон. Обработка сигналов. Пластичность. Моделирование. Фундаментальное руководство //Изд-во Тюменского государственного университета. 2008 . 548 с.
2. Анохин П.К. Принципиальные вопросы общей теории функциональных систем // Принципы системной организации функций. – М.: Наука, 1973.
3. Беркинблит М.Б. Нейронные сети – М.: Издательство МИРОС, 1993.
4. Вальцев В.Б, Лавров В.В. Целесообразное фрагментирование информации на входе в мозг // "Информационные технологии", №2, 2006, с.22-30.
5. Веденов А. А. Моделирование элементов мышления. М.: Наука, 1988. 159 с.
6. Виноградова О.С. Нейронаука конца второго тысячелетия: смена парадигм // Журнал высш. нервн. деятел. 2000. Т. 50. С. 743-774.
7. Вороновский Г. К., Махотило К. В., Петрашев С. Н., Сергеев С. А. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности. — Харьков: Основа, 1997. — 112 с. — ISBN 5-7768-0293-8.
8. Голубев Ю. Ф. Нейросетевые методы в мехатронике. — М.: Изд-во Моск. унта, 2007. — 157 с. — ISBN 978-5-211-05434-9.
9. Горбань А. Н., Дунин-Барковский В. Л. и др. Нейроинформатика. — Новосибирск: Наука, 1998.
10. Дунин-Барковский В. Л., Терехин А. Т. Нейронные сети и нейрокомпьютеры: тенденции развития исследований и разработок // Микропроцеcсорные средства и системы. 1990. N 2. C. 12-14.
11. Еремин Д. М., Гарцеев И. Б. Искусственные нейронные сети в интеллектуальных системах управления. — М.: МИРЭА, 2004. — 75 с. — ISBN 5-7339-0423-2.
12. Клини С. Математическая логика// М.: Мир, 1973. 480 с.
13. Комарцова Л. Г., Максимов А. В. Нейрокомпьютеры. — 1-е. — Изд-во МГТУ им. Н.Э. Баумана, 2002. — С. 320. — ISBN 5-7038-1908-3
14. Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория и практика. — М.: Горячая линия - Телеком, 2001. — 382 с. — ISBN 5-93517-031-0.
15. Мак-Каллок У., Питтс В. Логическое исчисление идей, относящихся к нервной активности // Нейрокомпьютер. –1992. - №3/4. – Р.40-50.
16. Минский М., Пейперт С. Перцептроны / пер. с англ. М.: Мир, 1971. 261 с. (Англ. оригинал: Minsky M., Papert S. Perceptrons: An Introduction to Computational Geometry. Cambridge, MA: MIT Press, 1969.).
17. Миркес Е. М. Нейрокомпьютер. Проект стандарта. — Новосибирск: Наука, 1999. — 337 с. — ISBN ISBN 5-02-031409-9.
18. Николлс Дж., Мартин Р., Валлас Б., Фукс П. От нейрона к мозгу – М.: Издательство Эдиториал УРСС, 2003.
19. Розенблат Ф. Принципы нейродинамики. Перцептроны и теория механизмов мозга / пер. с англ. М.: Мир, 1965. 175 с. (Англ. оригинал: Rosenblatt F. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Washington, D.C.: Spartan Books, 1962.)
20. Савельев А. В. На пути к общей теории нейросетей. К вопросу о сложности // Нейрокомпьютеры: разработка, применение. — 2006. — № 4—5. — С. 4—14.
21. Тадеусевич Рышард, Боровик Барбара, Гончаж Томаш, Леппер Бартош. Элементарное введение в технологию нейронных сетей с примерами программ / Перевод И. Д. Рудинского. — М.: Горячая линия — Телеком, 2011. — 408 с. — ISBN 978-5-9912-0163-6..
22. Терехин А.Т., Будилова Е.В. Сетевые механизмы биологической регуляции // Успехи физиологических наук. 1995. Т. 26. N 4. С.75-97.
23. Терехов В. А., Ефимов Д. В., Тюкин И. Ю. Нейросетевые системы управления. — М.: Высшая школа, 2002. — 184 с. — ISBN 5-06-004094-1.
24. Уоссермен Ф. Нейрокомпьютерная техника. Теория и практика // М.: Мир, 1992. 240 с.
25. Фриман Дж.У., Динамика мозга в восприятии и сознании: творческая роль хаоса // В сб. «Синергетика и психология». Вып.3. "Когнитивные процессы", Издательство «Когито-Центр", 2004.
26. Хайкин С. Нейронные сети: полный курс // Neural Networks: A Comprehensive Foundation.- 2-е. М.: «Вильямс», 2006. -1104 с.
27. Цетлин М.Л. Исследования по теории автоматов и моделированию биологических систем. — М.: Наука, 1969. 316с.
28. Швырков В.Б. Теория функциональной системы как методологическая основа нейрофизиологии поведения // Успехи физиологических наук. 1978. Т. 9. №1. С.81-105.
29. Ясницкий Л. Н. Введение в искусственный интеллект. — М.: Издат. центр «Академия», 2005. — 176 с. — ISBN 5-7695-1958-4.
30. Bechtel W, Abrahamsen A. Connectionism and the Mind: Parallel Processing, Dynamics, and Evolution in Networks, 2002.
31. Grossberg Stephen. 1988. Neural Networks and Natural Intelligence. MIT Press, Cambridge, MA.
32. Hebb, D. O. The organization of behavior: a neuropsychological theory. New York (2002) (Оригинальное издание — 1949)
33. Lipman R. An introdaction to computing with neural nets // IEEE Acoustic,Speech and Signal Processing Magazine,1987,no 2,p 4-22.
34. Rosenblatt F. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Washington, DC: Spartan Books (1962).

# ПРИЛОЖЕНИЯ

## Приложение 1. Метод ближайшего соседа.

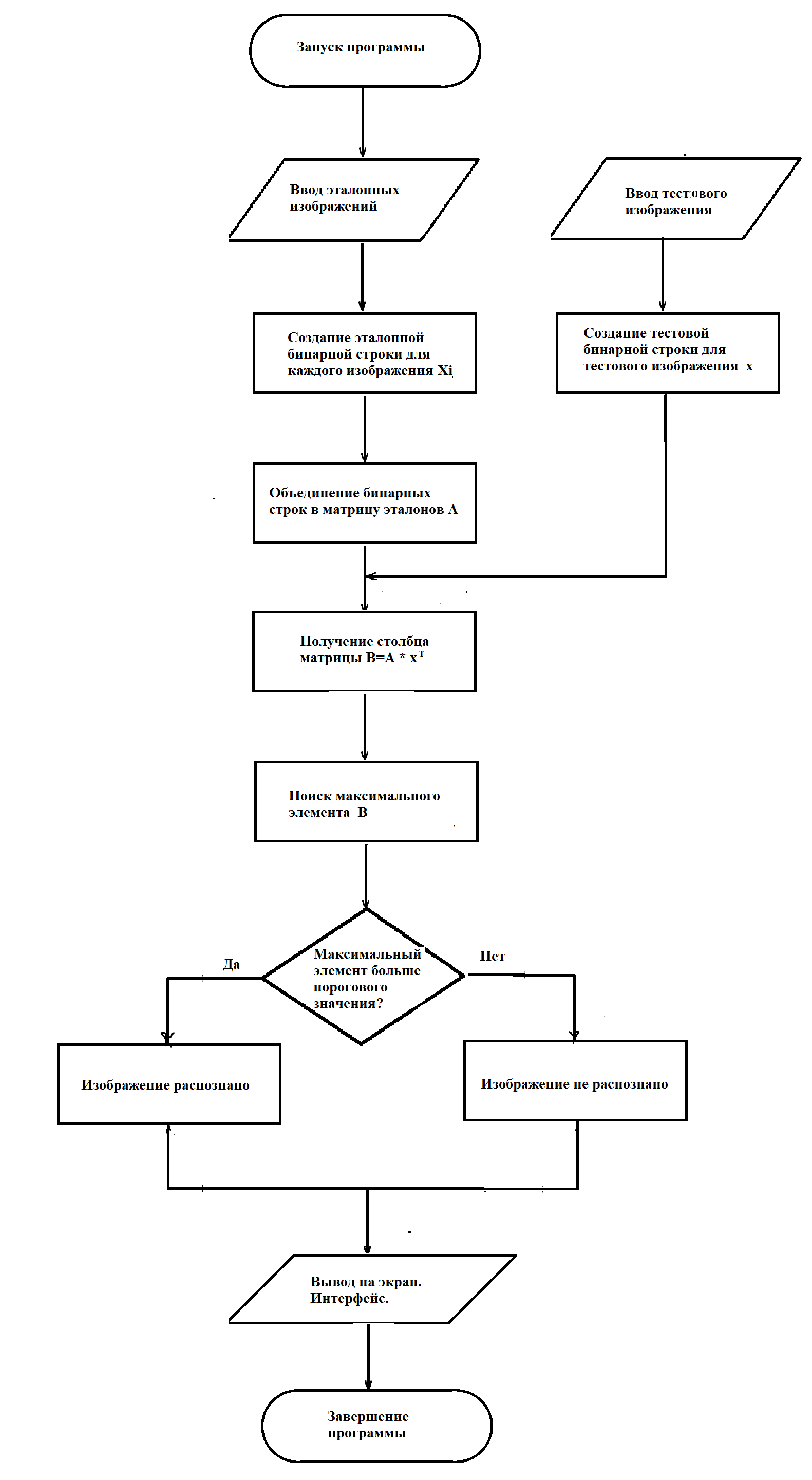
Метод ближайшего соседа заключается в следующем. Исходное изображение имеет большое количество пикселей (большее 10000). Изображение подготовлено в виде квадратной формы. Распознавание расчитано на исходные объекты 100х100=10000 пикселей. Поэтому необходимо приведение исходного изображения к стандартной форме. Это преобразование осуществляется с помощью метода ближайшего соседа. В Matlab R2011b метод ближайшего соседа реализуется с помощью функции imresize. Идея метода ближайшего соседа заключается в следующем:

Группа близлежащих пикселей заменяется одним пикселем, наиболее близким к этой группе. Близость оценивается с помощью критерия близости.

Один из типовых подходов заключается в разбиении данного отображения прямоугольной сеткой на квадраты, содержащие больше 1 пикселя. Например, сетку 100х100=10000 пкиселей разбиваем на прямоугольники 70х70=4900 пикселей. При этом в каждый прямоугольник попадает от 1 до 3 пикселей. Простейшим способом является взятие среднего арифметического интенсивностей пикселей, попавших внутрь элемента более крупной сетки. Другие способы описаны в программах реализации данного метода.

Для каждого пикселя конечного изображения выбирается один пиксель исходного, наиболее близкий к его положению с учетом масштабирования.

## Приложение 2. Блок-схема решение задачи.



## Приложение 3. Листинги программного обеспечения.

**Модуль BeginProgram.m**

function BeginProgram()

%Очистка памяти перед выполнением программы

clear;

%Стратегия - полноцветные

global strategy;

strategy=1;

%palitra - количество оттенков красного, зеленого и синего цвета,

%которые будут участвовать в палитре

global palitra;

palitra=16;

%Создание пустых массивов

%images - массив эталонных полноветных изображений, переведенных в бинарную комбинацию.

%Первый индекс - номер изображения.

%Второй индекс - индекс бинарной комбинации, характеризующей данное изображение

%Значение массива - значение элемента данной бинарной комбинации, характеризующей данное изображение

global portraitimages;

global pictureimages;

%notpreparedimages - массив эталонных полноветных изображений, готовых к отображению на экране в качестве ответа

%Первая координата - номер изображения

%Вторая и третья координаты - координаты пикселя данного изображения

%Четвертая - номер цвета (1 - красный, 2 - зеленый, 3 - синий)

%Значение массива - цвет данного пикселя данного изображения

global notpreparedportraitimages;

global notpreparedpictureimages;

%V - массив, хранящий изображение. Необходим для заполнения

%массива portraitimages

%Второй индекс V - номер элемента бинарной комбинации.

%Третий индекс V - цвет изображения

%Значение V - значение элемента бинарной комбинации

V = zeros(1, 10000, 3);

global portraitanswers;

global pictureanswers;

[portraitimages, notpreparedportraitimages, pictureanswers]=tobk(palitra, portraitimages, notpreparedportraitimages, pictureanswers, 'Портреты/', 'Портреты (описание)/');

[pictureimages, notpreparedpictureimages, pictureanswers]=tobk(palitra, pictureimages, notpreparedpictureimages, pictureanswers, 'Картины/', 'Картины (описание)/');

transposedimages=pictureimages.';

mat=pictureimages\*transposedimages;

[x,y]=meshgrid(1:1:30, 1:1:30);

surf(x,y,mat);

%MainMenu;

end

function [images, notpreparedimages, answers]=tobk(palitra, images, notpreparedimages, answers, imagesPath, answersPath)

% поиск jpg-файлов в директории imagesPath

% и вычисление кол-ва найденных файлов

MyJpgFiles=dir([imagesPath, '\*.jpg']);

MyJpgFilesQuantity=length(MyJpgFiles);

%Заполнение массивов images и notpreparedimages нулями

images = zeros(MyJpgFilesQuantity, 10000\*palitra\*3);

notpreparedimages = zeros(MyJpgFilesQuantity, 100, 100, 3);

%Работа с каждым эталонным изображением

for i = 1 : MyJpgFilesQuantity

%Получение набора из трех двумерных массивов, взятых из jpg-файла. Каждый массив отвечает за

%интенсивность красного, зеленого и синего цвета каждого пикселя соответственно

%После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа

MToDisplay=imresize(imread([imagesPath, MyJpgFiles(i).name]), [100 100]);

%Перевод значений массива из типа uint8 в тип double

MToDisplay=double(MToDisplay);

%Перевод массива в вектор-строку

for x = 1 : 3

V(:,:,x)=reshape(MToDisplay(:, :, x), 1, 10000);

end

%Перевод вектора-строки в бинарную вектор-строку

for j = 1:3

for k=1:10000

images(i,palitra\*3\*(k-1) + fix(V(1, k, j)/(256/palitra)) + palitra\*(j-1) + 1) = true;

end

end

%Сохранение изображения массива для вывода (если тестовое изображение будет распознано)

for j = 1:3

for k=1:100

for l=1:100

notpreparedimages(i,k,l,j)=fix(MToDisplay(k, l, j)/(256/palitra))\*fix(256/palitra);

end

end

end

end

images=double(images);

answers=[];

% поиск txt-файлов в директории answersPath

% и вычисление кол-ва найденных файлов

MyTxtFiles=dir([answersPath, '\*.txt']);

MyTxtFilesQuantity=length(MyTxtFiles);

%Работа с каждым описанием эталонных изображений

for i = 1 : MyTxtFilesQuantity

%Взятие первой строки из файла и копирование в соответствующую ячейку массива

fileID = fopen([answersPath, MyTxtFiles(i).name]);

buf = textscan(fileID, '%s','delimiter','\n');

answers{i} = buf{1};

fclose(fileID);

end

end

**Модуль BeginProgramGrey.m**

function BegimProgramGrey()

%Очистка памяти перед выполнением программы

clear;

%Стратегия - полутоновые

global strategy;

strategy=2;

%Ввод пользователем количеств оттенков серого цвета,

%которые будут участвовать в палитре

global palitra;

palitra=16;

%Создание пустых массивов

%images - массив эталонных полутоновых изображений, переведенных в бинарную комбинацию.

%Первый индекс - номер изображения.

%Второй индекс - индекс бинарной комбинации, характеризующей данное изображение

%Значение массива - значение элемента данной бинарной комбинации, характеризующей данное изображение

global portraitimages;

global pictureimages;

%notpreparedimages - массив эталонных полутоновых изображений, готовых к отображению на экране в качестве ответа

%Первая координата - номер изображения

%Вторая и третья координаты - координаты пикселя данного изображения

%Значение массива - цвет данного пикселя данного изображения

global notpreparedportraitimages;

global notpreparedpictureimages;

%V - массив, хранящий изображение. Необходим для заполнения

%массива images

%Второй индекс V - номер элемента бинарной комбинации.

%Значение V - значение элемента бинарной комбинации

V = zeros(1, 10000);

% Описание эталонных портретов писателей и картин

global portraitanswers;

global pictureanswers;

[portraitimages, notpreparedportraitimages, pictureanswers]=tobk(palitra, portraitimages, notpreparedportraitimages, pictureanswers, 'Портреты/', 'Портреты (описание)/');

[pictureimages, notpreparedpictureimages, pictureanswers]=tobk(palitra, pictureimages, notpreparedpictureimages, pictureanswers, 'Картины/', 'Картины (описание)/');

MainMenu;

end

function [images, notpreparedimages, answers]=tobk(palitra, images, notpreparedimages, answers, imagesPath, answersPath)

% поиск jpg-файлов в директории imagesPath

% и вычисление кол-ва найденных файлов

MyJpgFiles=dir([imagesPath, '\*.jpg']);

MyJpgFilesQuantity=length(MyJpgFiles);

%Заполнение массивов images и notpreparedimages нулями

images = zeros(MyJpgFilesQuantity, 10000\*palitra);

notpreparedimages = zeros(MyJpgFilesQuantity, 100, 100);

%Работа с каждым эталонным изображением

for i = 1 : MyJpgFilesQuantity

%Получение двумерных массивов, взятых из jpg-файла. Массив отвечает за

%интенсивность серого цвета каждого пикселя соответственно

%После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа

MToDisplay=rgb2gray(imresize(imread([imagesPath, MyJpgFiles(i).name]), [100 100]));

%Перевод значений массива из типа uint8 в тип double

MToDisplay=double(MToDisplay);

%Перевод массива в вектор-строку

V(:,:)=reshape(MToDisplay, 1, 10000);

%Перевод вектора-строки в бинарную вектор-строку

for m=1:1:10000

images(i, (m-1)\*palitra + fix(V(1, m, 1)/(256/palitra)) + 1)=true;

end

%Сохранение изображения массива для вывода (если тестовое изображение будет распознано)

for k=1:100

for l=1:100

notpreparedimages(i,k,l)=fix(MToDisplay(k, l, 1)/(256/palitra))\*fix(256/palitra);

end

end

images=double(images);

answers=[];

% поиск txt-файлов в директории answersPath

% и вычисление кол-ва найденных файлов

MyTxtFiles=dir([answersPath, '\*.txt']);

MyTxtFilesQuantity=length(MyTxtFiles);

%Работа с каждым описанием эталонных изображений

for i = 1 : MyTxtFilesQuantity

%Взятие первой строки из файла и копирование в соответствующую ячейку массива

fileID = fopen([answersPath, MyTxtFiles(i).name]);

buf = textscan(fileID, '%s','delimiter','\n');

answers{i} = buf{1};

fclose(fileID);

end

end

end

**Модуль BeginProgramBlack.m**

function BeginProgramBlack()

%Очистка памяти перед выполнением программы

clear;

%Стратегия - полутоновые

global strategy;

strategy=3;

%V - массив, хранящий изображение. Необходим для заполнения

%массива images

%Второй индекс V - номер элемента бинарной комбинации.

%Третий индекс V - цвет изображения

%Значение V - значение элемента бинарной комбинации

V = zeros(1, 10000);

%Создание пустых массивов

%images - массив эталонных изображений, переведенных в бинарную комбинацию.

%Первый индекс - номер изображения.

%Второй индекс - индекс бинарной комбинации, характеризующей данное изображение

%Значение массива - значение элемента данной бинарной комбинации, характеризующей данное изображение

global portraitimages;

global pictureimages;

global ECGimages;

%notpreparedimages - массив палитровых изображений, готовых к отображению на экране в качестве ответа

%Первая координата - номер изображения

%Вторая и третья координаты - координаты пикселя данного изображения

%Значение массива - цвет данного пикселя данного изображения

global notpreparedportraitimages;

global notpreparedpictureimages;

global notpreparedECGimages;

%answers - массив, хранящий описание соответствующего изображения в виде строки.

%индекс - номер эталонного изображения

%значение - строка описания изображения

global portraitanswers;

global pictureanswers;

global ECGanswers;

%Заполнение массивов images, notpreparedimages и portraitanswers

[portraitimages, notpreparedportraitimages, portraitanswers] = tobk(portraitimages, notpreparedportraitimages, portraitanswers, 'Портреты/', 'Портреты (описание)/');

[pictureimages, notpreparedpictureimages, pictureanswers] = tobk(pictureimages, notpreparedpictureimages, pictureanswers, 'Картины/', 'Картины (описание)/');

[ECGimages, notpreparedECGimages, ECGanswers] = tobk(ECGimages, notpreparedECGimages, ECGanswers, 'ЭКГ/', 'ЭКГ (описание)/');

MainMenu;

end

function [images, notpreparedimages, answers]=tobk(images, notpreparedimages, answers, imagesPath, answersPath)

% поиск jpg-файлов в директории imagesPath

% и вычисление кол-ва найденных файлов

MyJpgFiles=dir([imagesPath, '\*.jpg']);

MyJpgFilesQuantity=length(MyJpgFiles);

%Заполнение массивов images и notpreparedimages нулями

images = zeros(MyJpgFilesQuantity, 10000);

notpreparedimages = zeros(MyJpgFilesQuantity, 100, 100);

%Работа с каждым эталонным изображением

for i = 1 : MyJpgFilesQuantity

%Получение двумерных массивов, взятых из jpg-файла. Каждый массив отвечает за

%наличие/отсутствие белого цвета каждого пикселя соответственно.

%После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

MToDisplay=im2bw(imresize(imread([imagesPath, MyJpgFiles(i).name]), [100 100]), 0.5);

%Перевод массива в вектор-строку

V(:,:)=reshape(MToDisplay, 1, 10000);

%Перевод вектора-строки в бинарную вектор-строку

for m=1:1:10000

images(i, m)=~V(1,m);

end

%Сохранение изображения массива для вывода (если тестовое изображение будет распознано)

notpreparedimages(i, :, :)=MToDisplay;

end

notpreparedimages=logical(notpreparedimages);

images=double(images);

answers=[];

% поиск txt-файлов в директории answersPath

% и вычисление кол-ва найденных файлов

MyTxtFiles=dir([answersPath, '\*.txt']);

MyTxtFilesQuantity=length(MyTxtFiles);

%Работа с каждым описанием эталонных изображений

for i = 1 : MyTxtFilesQuantity

%Взятие первой строки из файла и копирование в соответствующую ячейку массива

fileID = fopen([answersPath, MyTxtFiles(i).name]);

buf = textscan(fileID, '%s','delimiter','\n');

answers{i} = buf{1};

fclose(fileID);

end

end

**Модуль MainMenu.m**

function varargout = MainMenu(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @MainMenu\_OpeningFcn, ...

'gui\_OutputFcn', @MainMenu\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

function MainMenu\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = MainMenu\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function trainmethod\_Callback(hObject, eventdata, handles)

global strategy;

switch strategy

case 1

Training;

case 2

Training;

case 3

TrainingBlack;

end

hf=findobj('Name','MainMenu');

close(hf);

function findmethod\_Callback(hObject, eventdata, handles)

global strategy;

switch strategy

case 1

Finding;

case 2

Finding;

case 3

FindingBlack;

end

hf=findobj('Name','MainMenu');

close(hf);

**Модуль Training.m**

function varargout = Training(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @Training\_OpeningFcn, ...

'gui\_OutputFcn', @Training\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

function Training\_OpeningFcn(hObject, eventdata, handles, varargin)

set(handles.PortraitRadio, 'Value', 1);

handles.output = hObject;

guidata(hObject, handles);

function varargout = Training\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function PathEdit\_Callback(hObject, eventdata, handles)

function PathEdit\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

%Добавление изображения в базу данных и масссив images

function addimage(images, notpreparedimages, answers, imagepath, descriptionpath, handles)

global palitra;

global strategy;

switch strategy

case 1

V = zeros(1, 10000, 3);

BufBinary = zeros(1, 10000\*palitra\*3);

case 2

V = zeros(1, 10000);

BufBinary = zeros(1, 10000\*palitra);

end

%Получение двумерного массива(набора из трех двумерных массиво), взятого из jpg-файла. Массив отвечает за

%интенсивность цветов каждого пикселя соответственно.

%После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

switch strategy

case 1

MToDisplay=imresize(imread(get(handles.PathEdit, 'String')), [100 100]);

case 2

MToDisplay=rgb2gray(imresize(imread(get(handles.PathEdit, 'String')), [100 100]));

end

%Перевод массива в вектор-строку

switch strategy

case 1

for x = 1 : 3

V(:,:,x)=reshape(MToDisplay(:, :, x), 1, 10000);

end

case 2

V(:,:)=reshape(MToDisplay, 1, 10000);

end

%Перевод вектор-строки в бинарную вектор-строку

switch strategy

case 1

for j = 1:3

for k=1:10000

BufBinary(1, palitra\*3\*(k-1) + fix(V(1, k, j)/(256/palitra)) + palitra\*(j-1) + 1) = true;

end

end

case 2

for m=1:1:10000

BufBinary(1, (m-1)\*palitra + fix(V(1, m)/(256/palitra)) + 1)=true;

end

end

%Добавление изображения в нейронную сеть

images(size(images, 1)+1, :) = BufBinary(1, :);

switch strategy

case 1

BufRawImage=zeros(100, 100, 3);

for j = 1:3

for k=1:100

for l=1:100

BufRawImage(k,l,j)=fix(MToDisplay(k, l, j)/(256/palitra))\*fix(256/palitra);

end

end

end

case 2

BufRawImage=zeros(100, 100, 1);

for k=1:100

for l=1:100

BufRawImage(k,l,1)=fix(MToDisplay(k, l)/(256/palitra))\*fix(256/palitra);

end

end

end

notpreparedimages(size(notpreparedimages, 1)+1, :, :, :)=BufRawImage;

answers(length(answers)+1)=get(handles.DescriptionEdit, 'String');

imwrite(MToDisplay, [imagepath, num2str(size(notpreparedimages, 1)), '.jpg']);

fileID = fopen([descriptionpath, num2str(length(answers)), '.txt'], 'a');

cell=answers(length(answers));

fprintf(fileID, '%s' , cell{1});

fclose(fileID);

%Запуск добавления изображения в нейронную сеть

function ForwardButton\_Callback(hObject, eventdata, handles)

global palitra;

global potrtaitimages;

global pictureimages;

global notpreparedportraitimages;

global notpreparedpictureimages;

global portraitanswers;

global pictureanswers;

global strategy;

if get(handles.PortraitRadio, 'Value') == 1

addimage(potrtaitimages, notpreparedportraitimages, portraitanswers, 'Портреты\', 'Портреты (описание)\', handles);

else

addimage(pictureimages, notpreparedpictureimages, pictureanswers, 'Картины\', 'Картины (описание)\', handles);

end

TrainingEnd;

hf=findobj('Name','Training');

close(hf);

%Возвращение в главное меню

function BackButton\_Callback(hObject, eventdata, handles)

MainMenu;

hf=findobj('Name','Training');

close(hf);

function DescriptionEdit\_Callback(hObject, eventdata, handles)

function DescriptionEdit\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

%Нахождение пути к эталонному изображению

function BrowseButton\_Callback(hObject, eventdata, handles)

[FileName,PathName] = uigetfile('\*.jpg','Select jpg-image');

Path = get(handles.PathEdit, 'String');

Path = [PathName, FileName];

set(handles.PathEdit, 'String', Path);

%Выбор базы портретов

function PortraitRadio\_Callback(hObject, eventdata, handles)

if get(hObject,'Value')==0

set(hObject,'Value', 1);

else

set(handles.PictureRadio, 'Value', 0);

end

%Выбор базы картин

function PictureRadio\_Callback(hObject, eventdata, handles)

if get(hObject,'Value')==0

set(hObject,'Value', 1);

else

set(handles.PortraitRadio, 'Value', 0);

end

**Модуль TrainingBlack.m**

function varargout = TrainingBlack(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @TrainingBlack\_OpeningFcn, ...

'gui\_OutputFcn', @TrainingBlack\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

function TrainingBlack\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = TrainingBlack\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function PathEdit\_Callback(hObject, eventdata, handles)

function PathEdit\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

%Добавление изображения в базу данных и масссив images

function addimage(images, notpreparedimages, answers, imagepath, descriptionpath, handles)

V = zeros(1, 10000);

BufBinary = zeros(1, 10000);

%Получение двумерного массива, взятого из jpg-файла. Массив отвечает за

%наличие/отсутствие белого цвета каждого пикселя соответственно.

%После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

MToDisplay=im2bw(imresize(imread(get(handles.PathEdit, 'String')), [100 100]), 0.5);

%Перевод массива в бинарную вектор-строку и последующее инвертирование

V(:,:)=reshape(MToDisplay, 1, 10000);

for m=1:1:10000

BufBinary(1, m)=~V(1,m);

end

%Добавление изображения в нейронную сеть

images(size(images, 1)+1, :) = BufBinary(1, :);

notpreparedimages(size(notpreparedimages, 1)+1, :, :)=MToDisplay;

answers(length(answers)+1)=get(handles.DescriptionEdit, 'String');

imwrite(MToDisplay, [imagepath, num2str(size(notpreparedimages, 1)), '.jpg']);

fileID = fopen([descriptionpath, num2str(length(answers)), '.txt'], 'a');

cell=answers(length(answers));

fprintf(fileID, '%s' , cell{1});

fclose(fileID);

%Нахождение пути к эталонному изображению

function BrowseButton\_Callback(hObject, eventdata, handles)

[FileName,PathName] = uigetfile('\*.jpg','Select jpg-image');

Path = get(handles.PathEdit, 'String');

Path = [PathName, FileName];

set(handles.PathEdit, 'String', Path);

function DescriptionEdit\_Callback(hObject, eventdata, handles)

function DescriptionEdit\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

%Выбор базы портретов

function PortraitRadio\_Callback(hObject, eventdata, handles)

if get(hObject,'Value')==0

set(hObject,'Value', 1);

else

set(handles.PictureRadio, 'Value', 0);

set(handles.ECGRadio, 'Value', 0);

end

%Выбор базы картин

function PictureRadio\_Callback(hObject, eventdata, handles)

if get(hObject,'Value')==0

set(hObject,'Value', 1);

else

set(handles.PortraitRadio, 'Value', 0);

set(handles.ECGRadio, 'Value', 0);

end

%Выбор базы ЭКГ

function ECGRadio\_Callback(hObject, eventdata, handles)

if get(hObject,'Value')==0

set(hObject,'Value', 1);

else

set(handles.PortraitRadio, 'Value', 0);

set(handles.PictureRadio, 'Value', 0);

end

%Возвращение в главное меню

function BackButton\_Callback(hObject, eventdata, handles)

MainMenu;

hf=findobj('Name','TrainingBlack');

close(hf);

%Запуск добавления изображения в нейронную сеть

function OKButton\_Callback(hObject, eventdata, handles)

global potrtaitimages;

global pictureimages;

global ECGimages;

global notpreparedportraitimages;

global notpreparedpictureimages;

global notpreparedECGimages;

global portraitanswers;

global pictureanswers;

global ECGanswers;

if get(handles.PortraitRadio, 'Value') == 1

addimage(potrtaitimages, notpreparedportraitimages, portraitanswers, ‘Портреты’, handles);

else if get(handles.PictureRadio, 'Value') == 1

addimage(pictureimages, notpreparedpictureimages, pictureanswers,’Картины', handles);

else

addimage(ECGimages, notpreparedECGimages, ECGanswers, 'ЭКГ', handles);

end

end

TrainingEnd;

hf=findobj('Name','TrainingBlack');

close(hf);

**Модуль TrainingEnd.m**

function varargout = TrainingEnd(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @TrainingEnd\_OpeningFcn, ...

'gui\_OutputFcn', @TrainingEnd\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

%Вывод окна с подтверждением сохранения изображения

function TrainingEnd\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = TrainingEnd\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

%Возврат к главному меню

function pushbutton1\_Callback(hObject, eventdata, handles)

MainMenu;

hf=findobj('Name','TrainingEnd');

close(hf);

**Модуль Finding.m**

function varargout = Finding(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @Finding\_OpeningFcn, ...

'gui\_OutputFcn', @Finding\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

function Finding\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = Finding\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function PathEdit\_Callback(hObject, eventdata, handles)

function PathEdit\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

%Нахождение пути к тестовому изображению

function BrowseButton\_Callback(hObject, eventdata, handles)

[FileName,PathName] = uigetfile('\*.jpg','Select jpg-image');

Path=get(handles.PathEdit, 'String');

Path = [PathName, FileName];

set(handles.PathEdit, 'String', Path);

%Возврат к главному меню

function BackButton\_Callback(hObject, eventdata, handles)

MainMenu;

hf=findobj('Name','Finding');

close(hf);

function ForwardButton\_Callback(hObject, eventdata, handles)

global strategy;

global notpreparedtest;

global palitra;

global portraitimages;

global pictureimages;

switch strategy

case 1

V = zeros(1, 10000, 3);

case 2

V = zeros(1, 10000);

end

%Работа с тестовым изображением

%Получение двумерных массивов, взятых из jpg-файла. Каждый массив отвечает за

%интенсивность своего цвета каждого пикселя соответственно

%После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа

switch strategy

case 1

notpreparedtest=imresize(imread(get(handles.PathEdit, 'String')), [100 100]);

case 2

notpreparedtest=imresize(rgb2gray(imread(get(handles.PathEdit, 'String'))), [100 100]);

end

%Перевод массива в вектор-строку

switch strategy

case 1

for i = 1 : 3

V(:,:,i)=reshape(notpreparedtest(:, :, i), 1, 10000);

end

case 2

V(:,:)=reshape(notpreparedtest, 1, 10000);

end

%Перевод значений массива из типа int в тип double

notpreparedtest=double(notpreparedtest);

%Создание вектора-строки testimage, который будет хранить бинарную комбинацию тестового изображения

switch strategy

case 1

testimage = zeros(1, 10000\*palitra\*3);

case 2

testimage = zeros(1, 10000\*palitra);

end

%Заполнение вектора-столбца testimage

switch strategy

case 1

for j = 1:3

for k=1:10000

testimage(1, palitra\*3\*(k-1) + fix(V(1, k, j)/(256/palitra)) + palitra\*(j-1) + 1)=true;

end

end

case 2

for m=1:1:10000

testimage(1, (m-1)\*palitra + fix(V(1, m)/(256/palitra)) + 1)=true;

end

end

testimage=double(testimage);

% Перемножаем эталонные и тестовую БК

% Результатом будет вектор-столбец, где каждое значение будет хранить

% количество пикселей тествого изображения, совпадающих с соответствующими

% пикселями эталонных изображений

global imagekind;

if get(handles.PortraitRadio, 'Value') == 1

res = portraitimages \* testimage';

imagekind='portrait';

else

res = pictureimages \* testimage';

imagekind='picture';

end

% Ищем максимаьный элемент в полученном векторе,

% т. е. наиболее схожее эталонное изображение с

% тестовым

max = 0;

imax = 0;

for i = 1 : length(res)

if res(i) > max

max = res(i);

imax = i;

end;

end

%Назначаем порог схожести

switch strategy

case 1

threshold=5000;

case 2

threshold=1500;

end

global ImageToShow;

if max > threshold

ImageToShow=imax;

%Выводим изображение на экран, если порог превышен

else

ImageToShow=0;

%Иначе выводим в консоли изображение не распознано

end

FindingTest;

hf=findobj('Name','Finding');

close(hf);

%Выбор базы портретов

function PortraitRadio\_Callback(hObject, eventdata, handles)

if get(hObject,'Value')==0

set(hObject,'Value', 1);

else

set(handles.PictureRadio, 'Value', 0);

end

%Выбор базы картин

function PictureRadio\_Callback(hObject, eventdata, handles)

if get(hObject,'Value')==0

set(hObject,'Value', 1);

else

set(handles.PortraitRadio, 'Value', 0);

end

**Модуль FindingBlack.m**

function varargout = FindingBlack(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @FindingBlack\_OpeningFcn, ...

'gui\_OutputFcn', @FindingBlack\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

function FindingBlack\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = FindingBlack\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function PathEdit\_Callback(hObject, eventdata, handles)

function PathEdit\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

%Нахождение пути к тестовому изображению

function BrowseButton\_Callback(hObject, eventdata, handles)

[FileName,PathName] = uigetfile('\*.jpg','Select jpg-image');

Path=get(handles.PathEdit, 'String');

Path = [PathName, FileName];

set(handles.PathEdit, 'String', Path);

%Выбор базы портретов

function PortraitRadio\_Callback(hObject, eventdata, handles)

if get(hObject,'Value')==0

set(hObject,'Value', 1);

else

set(handles.ECGRadio, 'Value', 0);

set(handles.PictureRadio, 'Value', 0);

end

%Выбор базы картин

function PictureRadio\_Callback(hObject, eventdata, handles)

if get(hObject,'Value')==0

set(hObject,'Value', 1);

else

set(handles.PortraitRadio, 'Value', 0);

set(handles.ECGRadio, 'Value', 0);

end

%Выбор базы ЭКГ

function ECGRadio\_Callback(hObject, eventdata, handles)

if get(hObject,'Value')==0

set(hObject,'Value', 1);

else

set(handles.PortraitRadio, 'Value', 0);

set(handles.PictureRadio, 'Value', 0);

end

%Возврат к главному меню

function BackButton\_Callback(hObject, eventdata, handles)

MainMenu;

hf=findobj('Name','FindingBlack');

close(hf);

function OKButton\_Callback(hObject, eventdata, handles)

global notpreparedtest;

global portraitimages;

global pictureimages;

global ECGimages;

V = zeros(1, 10000);

%Получение двумерного массива, взятого из jpg-файла. Массив отвечает за

%наличие/отсутствие белого цвета каждого пикселя соответственно.

%После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

notpreparedtest=imresize(im2bw(imread(get(handles.PathEdit, 'String')), 0.5), [100 100]);

%Перевод массива в вектор-строку

V(:,:)=reshape(notpreparedtest, 1, 10000);

%Создание вектора-строки testimage, который будет хранить бинарную комбинацию тестового изображения

testimage = zeros(1, 10000);

%Заполнение вектора-столбца testimage

for m=1:1:10000

testimage(1, m)=~V(1,m);

end

% Выбор базы изображений, среди которых будет идти поиск наиболее подходящего эталонного изображения

% Перемножаем эталонные и тестовую БК

% Результатом будет вектор-столбец, где каждое значение будет хранить

% количество пикселей тествого изображения, совпадающих с соответствующими

% пикселями эталонных изображений

global imagekind;

if get(handles.PortraitRadio, 'Value') == 1

res = portraitimages \* testimage';

imagekind='portrait';

elseif get(handles.PictureRadio, 'Value') == 1

res = pictureimages \* testimage';

imagekind='picture';

else

res = ECGimages \* testimage';

imagekind='ECG';

end

% Ищем максимальный элемент в полученном векторе,

% т. е. наиболее схожее эталонное изображение с

% тестовым

max = 0;

imax = 0;

for i = 1 : length(res)

if res(i) > max

max = res(i);

imax = i;

end;

end

%Назначаем порог схожести

threshold=205;

global ImageToShow;

if max > threshold

%Сохранение номера наиболее похожего эталонного изображения, если порог превышен

ImageToShow=imax;

else

%Иначе сохраняем нулевой номер (отсутствие похожего эталонного изображения)

ImageToShow=0;

end

%Вызов модуля FindingTest.m

FindingTest;

hf=findobj('Name','FindingBlack');

close(hf);

**Модуль FindingTest.m**

function varargout = FindingTest(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @FindingTest\_OpeningFcn, ...

'gui\_OutputFcn', @FindingTest\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

function FindingTest\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

global notpreparedtest;

global ImageToShow;

global notpreparedportraitimages;

global notpreparedpictureimages;

global portraitanswers;

global pictureanswers;

global strategy;

global imagekind;

set(handles.ResultImage, 'visible', 'on');

switch strategy

case 1

notpreparedtest=uint8(notpreparedtest);

case 2

notpreparedtest=uint8(notpreparedtest);

case 3

notpreparedtest=double(notpreparedtest);

end

%Вывод тестового изображения

axes(handles.ResultImage);

imshow(notpreparedtest);

set(handles.Description, 'String', 'Testpicture');

guidata(hObject, handles);

function varargout = FindingTest\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

%Переход к модулю FindingEnd.m

function OKButton\_Callback(hObject, eventdata, handles)

FindingEnd;

hf=findobj('Name','FindingTest');

close(hf);

function pushbutton2\_Callback(hObject, eventdata, handles)

FindingEnd;

hf=findobj('Name','FindingTest');

close(hf);

**Модуль FindingEnd.m**

function varargout = FindingEnd(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @FindingEnd\_OpeningFcn, ...

'gui\_OutputFcn', @FindingEnd\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

function FindingEnd\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

global ImageToShow;

global notpreparedportraitimages;

global notpreparedpictureimages;

global notpreparedECGimages;

global portraitanswers;

global pictureanswers;

global ECGanswers;

global strategy;

global imagekind;

set(handles.ResultImage, 'visible', 'on');

%В зависимости от того, какая база изображений использовалась для распознавания тестового

% берутся соответствующие массивы бинарных строк и массивы изображений

switch imagekind

case 'portrait'

notpreparedimages=notpreparedportraitimages;

answers=portraitanswers;

case 'picture'

notpreparedimages=notpreparedpictureimages;

answers=pictureanswers;

case 'ECG'

notpreparedimages=notpreparedECGimages;

answers=ECGanswers;

end

%Если тестовое изображение распознано

if ImageToShow > 0

% В зависимости от того, с какой цветностью изображений работает программа

switch strategy

case 1

%Вывести полноцветное эталонное изображение

I = zeros(100, 100, 3);

I(:, :, :) = notpreparedimages(ImageToShow, :, :, :);

I=uint8(I);

axes(handles.ResultImage);

case 2

%Вывести полутоновое эталонное изображение

I = zeros(100, 100);

I(:, :) = notpreparedimages(ImageToShow, :, :);

I=uint8(I);

axes(handles.ResultImage);

case 3

%Вывести контурное эталонное изображение

I = zeros(100, 100);

I(:, :) = notpreparedimages(ImageToShow, :, :);

I=logical(I);

axes(handles.ResultImage);

end

imshow(I);

Path = answers{ImageToShow};

%Вывести описание эталонного изображения

set(handles.Description, 'String', Path);

%Если тестовое изображение НЕ распознано

else

%Вывести 'Image not found'

set(handles.ResultImage, 'visible', 'off');

set(handles.Description, 'String', 'Image not found');

end

guidata(hObject, handles);

function varargout = FindingEnd\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

%Возврат в главное меню

function OKButton\_Callback(hObject, eventdata, handles)

MainMenu;

hf=findobj('Name','FindingEnd');

close(hf);