ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Д09.03.04-ИИ.23-20/6251.КР

Кафедра прикладной математики и
искусственного интеллекта

КУРСОВАЯ РАБОТА

по дисциплине «Нейросети»

Тема: «Нейроморфный алгоритм обнаружения созвездий»

Руководители:

_____ ст. преп. О.А.Гудаев
(дата, подпись)

Нормоконтроль:

_____ асс. Н.П. Пулинец
(дата, подпись)

Исполнитель:

_____ ст. гр. ПИ-20г В.А. Пустовой
(дата, подпись)

Донецк – 2023

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОВЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет: ***Интеллектуальных систем и программирования***
Направление: ***Программная инженерия***
Профиль: ***Искусственный интеллект***
Кафедра: «***Прикладной математики и искусственного интеллекта»***

ЗАДАНИЕ НА КУРСОВОУЮ РАБОТУ

по дисциплине «Нейросети»

Студенту Пустовому Вадиму Александровичу группы ПИ-20г
(фамилия, имя, отчество)

Тема проекта: «Нейроморфный алгоритм обнаружения созвездий»

Исходные данные к проекту: Задание, выданное кафедрой, спецификация графического языка Unified Modeling Language, спецификация однослойного персептрона, спецификации многослойной нейронной сети

Перечень искомых результатов: Персептрон, разделяющий фоны от образов, нейронная сеть обнаружения графических образов диаграммы UML.

Рекомендуемая литература: Барский, А.Б. Логические нейронные сети: Учебное пособие / А.Б. Барский. - М.: Бином. ИНТУИТ.РУ, 2012. - 352 с.
Галушкин, А.И. Нейронные сети: основы теории. / А.И. Галушкин. - М.: РиС, 2015. - 496 с.
М. Прайс. C# 7 и .NET Core. / Прайс, Марк. – СПб.: Питер; 3-е изд.; 2018. - 640 с.

Дата выдачи задания 10.02.2023

Дата защиты проекта 29.05.2023

Руководители _____ ст. преп. Гудаев О. А.
(подпись) (должность, Ф.И.О.)

Разработчик _____ Пустовой В. А.
(подпись) (Ф.И.О)

РЕФЕРАТ

Пояснительная записка: страниц 66, рис. 22, источников 6, прил. 4.

Целью работы является создание однослойного персептрона для разделения изображения на фон и образ, многослойной нейронной сети для распознавания конкретного созвездия и сайта, способного определять рисунок созвездия по фотографии с веб-камеры.

Для построения диаграмм используется язык диаграмм Unified Modelling Language (UML). Для реализации работы используются языки программирования C#, HTML, JavaScript, CSS.

В ходе проектирования получены следующие результаты: математическая модель нейронной сети, структуру данных для хранения нейросети, исходный код протоколов классов на языке программирования высокого уровня C# и UML-диаграммы.

Для тестирования программ на языке высокого уровня C# была использована среда разработки «Visual Studio 2022». Для создания диаграмм были использованы программа проектирования диаграмм «Draw.io». Для написания HTML сайта использовалась среда разработки «Visual Studio Code».

НЕЙРОСЕТЬ, ПЕРСЕПТРОН, МНОГОСЛОЙНАЯ
НЕЙРОННАЯ СЕТЬ, НЕЙРОМОРФНЫЙ АЛГОРИТМ ОБНАРУЖЕНИЯ

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Нейросети представляют новую и перспективную вычислительную технологию, дающую новые подходы к исследованию сложных и трудно формализуемых задач. Первоначально они разрабатывались для решения задач распознавания образов. В настоящее время применяются в следующих областях: распознавание образов изображений, обработка зашумленных данных (фильтрация), сжатие данных, классификация и кластеризация данных, ассоциативный поиск, решение оптимизационных задач, задачи диагностики, управление сложными процессами, моделирование сложных процессов, финансовая математика (задачи прогнозирования).

Теория нейронных сетей включают широкий круг вопросов из разных областей науки: биофизики, математики, информатики, схемотехники и технологии. Поэтому понятие "нейронные сети" детально определить сложно.

Искусственные нейронные сети (ИНС) — совокупность моделей биологических нейронных сетей. Представляют собой сеть элементов — искусственных нейронов — связанных между собой синаптическими соединениями. Сеть обрабатывает входную информацию и в процессе изменения своего состояния во времени формирует совокупность выходных сигналов.

Работа сети состоит в преобразовании входных сигналов во времени, в результате чего меняется внутреннее состояние сети и формируются выходные воздействия. Обычно НС оперирует цифровыми, а не символьными величинами.

Большинство моделей НС требуют обучения. В общем случае, обучение — такой выбор параметров сети, при котором сеть лучше всего справляется с поставленной проблемой. Обучение — это задача многомерной оптимизации, и для ее решения существует множество алгоритмов.

1 АРХИТЕКТУРА НЕЙРОСЕТИ

1.1 Однослойный персептрон

Однослойный персептрон — это простейшая форма искусственной нейронной сети, состоящая из одного слоя нейронов. В каждом нейроне однослойного персептрона есть входы, каждый из которых имеет свой вес. Эти входы связаны с выходами предыдущего слоя или с внешними источниками данных. Каждый нейрон вычисляет взвешенную сумму входных сигналов, а затем пропускает эту сумму через активационную функцию, чтобы получить выходной сигнал.

Топология однослойного персептрона представляет собой прямую связь между входными и выходными нейронами. Каждый входной нейрон связан со всеми выходными нейронами, и каждая связь имеет свой вес. Нейроны в однослойном персептроне не образуют обратных связей, то есть информация перемещается только в одном направлении, от входов к выходам.

Такая топология однослойного персептрона позволяет решать простые задачи классификации, где данные могут быть линейно разделимы. Например, если у вас есть два класса объектов, и между ними можно провести прямую линию, то однослойный персептрон с активационной функцией, такой как пороговая функция или сигмоида, может классифицировать эти объекты.

Однослойный персептрон имеет ограниченные вычислительные возможности и не может решить сложные задачи, которые не могут быть линейно разделены. Для более сложных задач требуется использование многослойных нейронных сетей, таких как многослойный персептрон или глубокие нейронные сети.

1.2 Алгоритм обучения

Правило Хебба включает в себя следующие шаги:

1. Инициализация весов: изначально веса (wx, wy, w0) инициализируются случайными значениями или нулями.

2. Цикл обучения: выполняется цикл обучения, состоящий из двух фаз.

3. Проход по первой половине обучающих примеров. Для каждого примера, вычисляется взвешенная сумма S, которая представляет собой сумму произведений входов (p[i].X и p[i].Y) на соответствующие веса (wx и wy), с добавлением веса смещения w0. Затем применяется пороговая функция для определения выходного значения y (0 или 1) на основе значения S.

4. Обновление весов: Если выходное значение y не совпадает с ожидаемым значением p[i].W, то происходит коррекция весов. В случае, если y = 1, а ожидаемое значение p[i].W = 0, то происходит увеличение весов (wx, wy, w0) на небольшой шаг обучения nu. В противном случае, если y = 0, а ожидаемое значение p[i].W = 1, то происходит уменьшение весов на шаг обучения nu.

5. Проход по второй половине обучающих примеров, аналогично пункту 3.

6. Завершение обучения: после завершения цикла обучения, веса достигают оптимальных значений для данной задачи классификации.

1.3 Многослойная нейросеть

Многослойными называются нейронные сети, в которых нейроны сгруппированы в слои. При этом каждый нейрон предыдущего слоя связан со всеми нейронами следующего слоя, а внутри слоёв связи между нейронами отсутствуют.

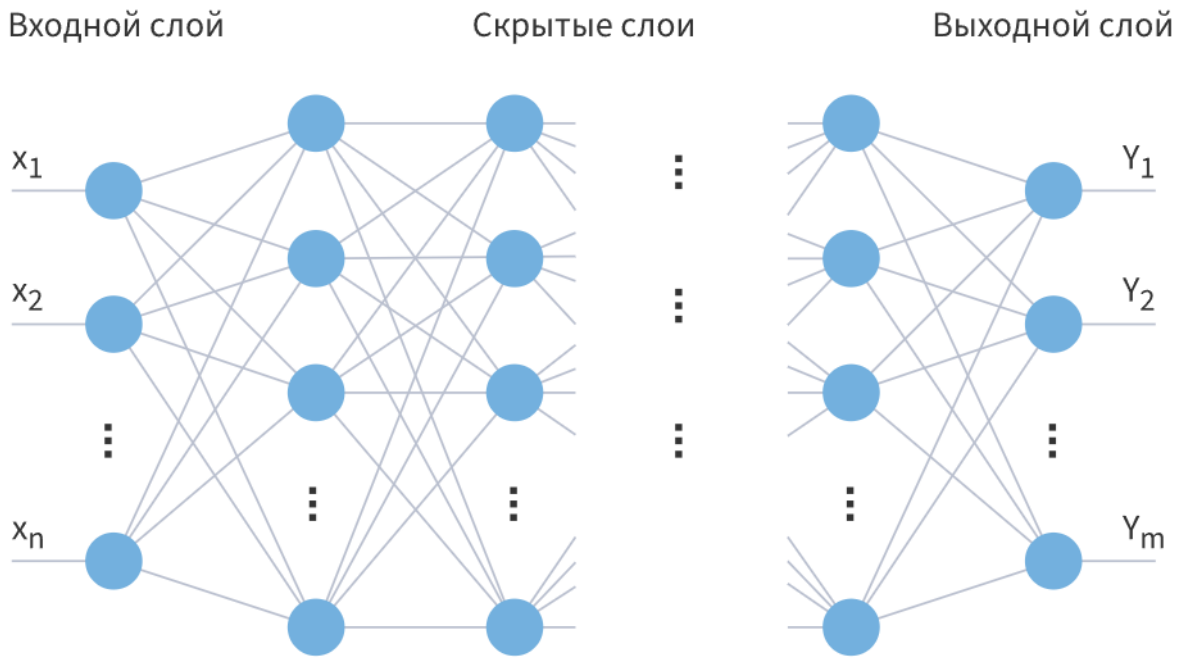


Рисунок 1.1 – Схема нейросети

Слои нумеруются слева направо. Первый слой называют входным или распределительным. Его нейроны (которые также называют входными) принимают элементы вектора признаков и распределяют их по нейронам следующего слоя. При этом обработка данных во входном слое не производится.

Последний слой называется выходным. На выходах его нейронов (они называются выходными) формируется результат работы сети — элементы выходного вектора.

Между входным и выходным слоем располагаются один или несколько промежуточных или скрытых слоёв. Скрытыми они называются по тому, что их входы и выходы неизвестны для внешних по отношению к нейронной сети программам и пользователю.

Приложение реализует НС с одним скрытым слоем.

Количество нейронов входного слоя приравняем к количеству пикселей входного изображения: $32 * 32 = 1024$.

Для оценки числа нейронов в скрытом слое используется формулы для оценки необходимого числа весов (1.1).

$$\frac{m2 \cdot n}{1+log_2 n} \leq L_w \leq m2 \cdot \left(\frac{n}{m2} + 1\right) \cdot (m + m2 + 1) + m2 \qquad (1.1),$$

где m – размерность входного сигнала (1024), $m2$ – размерность выходного сигнала (9), n – кол-во эталонов (162).

$$174{,}8 \leq L_w \leq 176\,823$$

В конечном итоге было выбрано 192 нейрона для скрытого слоя.

Всего у нас 9 классов изображений, следовательно и выходных нейронов должно быть 9.

На рисунке 1.2 изображена итоговая структура ИНС.

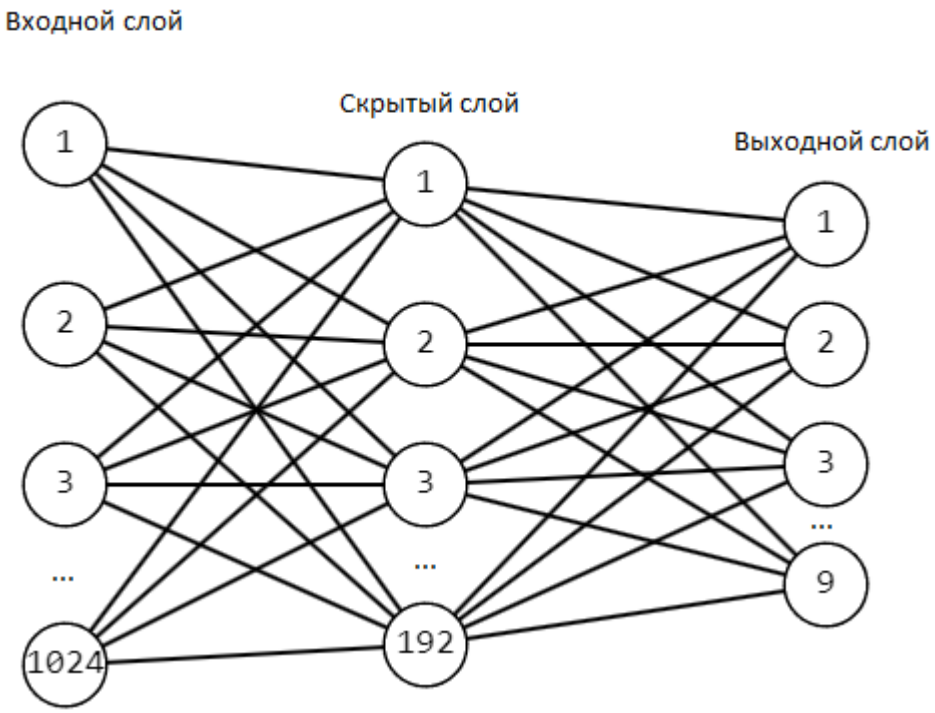


Рисунок 1.2 – Структура ИНС

1.4 Активационная функция

Сигмоидная функция активации — это нелинейная функция, которая преобразует входное значение в диапазоне от отрицательной бесконечности до положительной бесконечности в значение от 0 до 1. Эта функция активации часто используется в нейронных сетях для задач бинарной классификации.

Математически сигмоидная функция активации определяется следующим образом:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (1.2)$$

Графически сигмоидная функция активации выглядит как S-образная кривая, которая монотонно возрастает и имеет асимптоты на 0 и 1. В частности, если $x > 0$, то $f(x) > 0.5$, а если $x < 0$, то $f(x) < 0.5$. Значение 0.5 достигается при $x = 0$. Сигмоидная функция активации используется для преобразования выходного значения нейрона в вероятность, т.е. вероятность того, что входное значение относится к классу 1, если мы работаем с задачей бинарной классификации. Если значение сигмоидной функции близко к 1, то вероятность того, что входное значение относится к классу 1, высока. Если значение близко к 0, то вероятность того, что входное значение относится к классу 1, низкая.

1.5 Алгоритм обучения

Метод обратного распространения ошибки — метод вычисления градиента, который используется при обновлении весов многослойного перцептрона. Это итеративный градиентный алгоритм, который используется с целью минимизации ошибки работы многослойного перцептрона и получения желаемого выхода.

Алгоритм обучения НС

1. Считывается из файла массив тренировочных изображений, после чего сжимается до нужного размера.

2. Для инициализации весов и смещений используется методика Xavier.

3. Выбирается изображение из тренировочной выборки и совершает проход по нейросети, сохраняя выходные значения в переменные hidOutput и output.

4. Вычисляется ошибка выходных значений.

5. С помощью обратного распространения ошибки изменяются веса и смещения.

6. Шаги 3-5 повторяются для заданного числа эпох.

2 ОПИСАНИЕ UML ДИАГРАММ

UML (англ. Unified Modeling Language – унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML является языком широкого профиля, это – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода.

2.1 Диаграмма классов

Диаграммы классов приведены на рисунке 2.1. В UML диаграмма классов является типом диаграммы статической структуры. Она описывает структуру системы, показывая её классы, их атрибуты и операторы, а также взаимосвязи этих классов. Взаимосвязь – это особый тип логических отношений между сущностями, показанных на диаграммах классов и объектов.

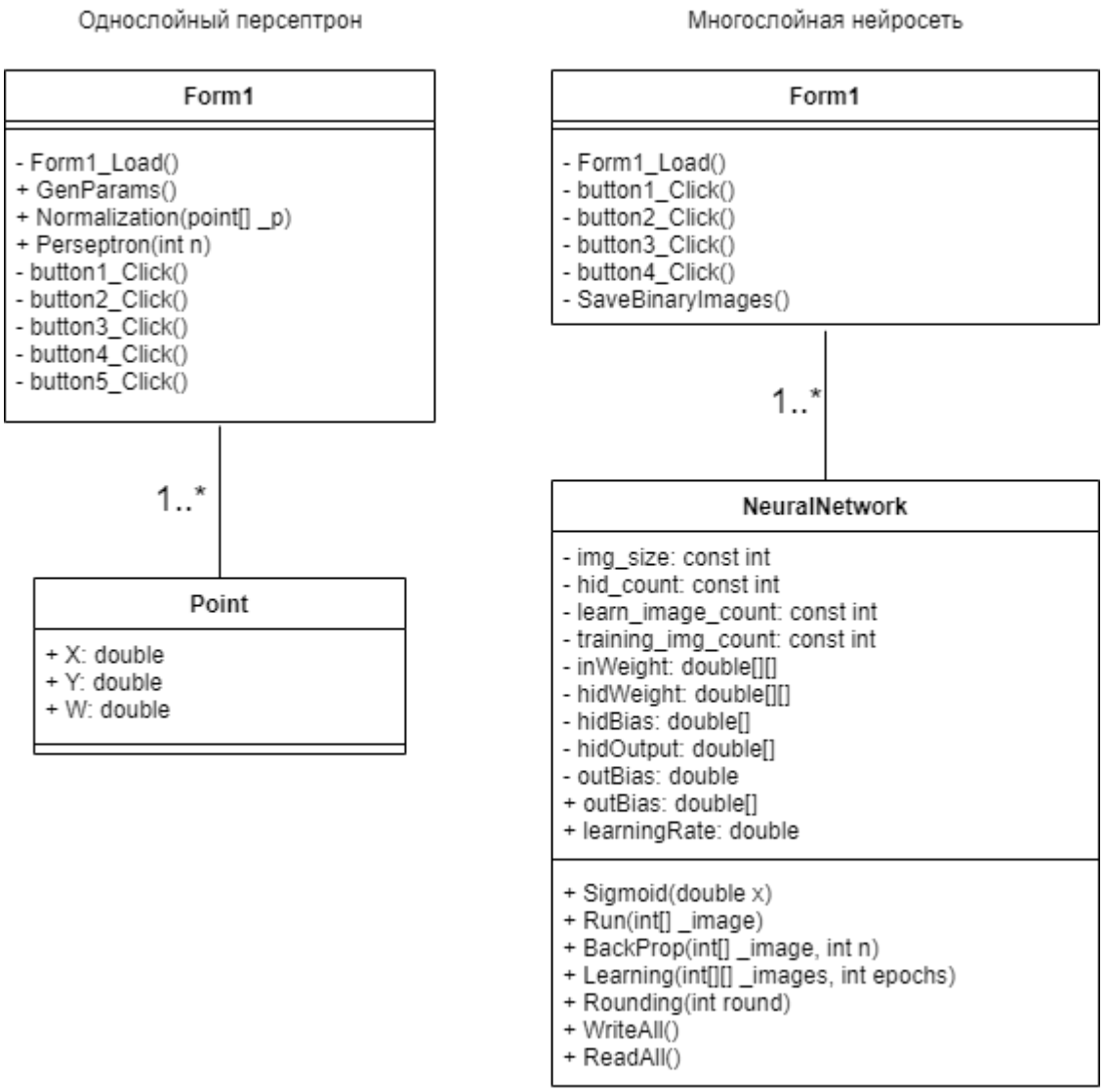


Рисунок 2.1 – Диаграммы классов

2.2 Диаграмма последовательности

Диаграмма состояний приведена на рисунке 2.2. Диаграмма последовательности отражает динамические аспекты поведения системы. По существу, эта диаграмма представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой.

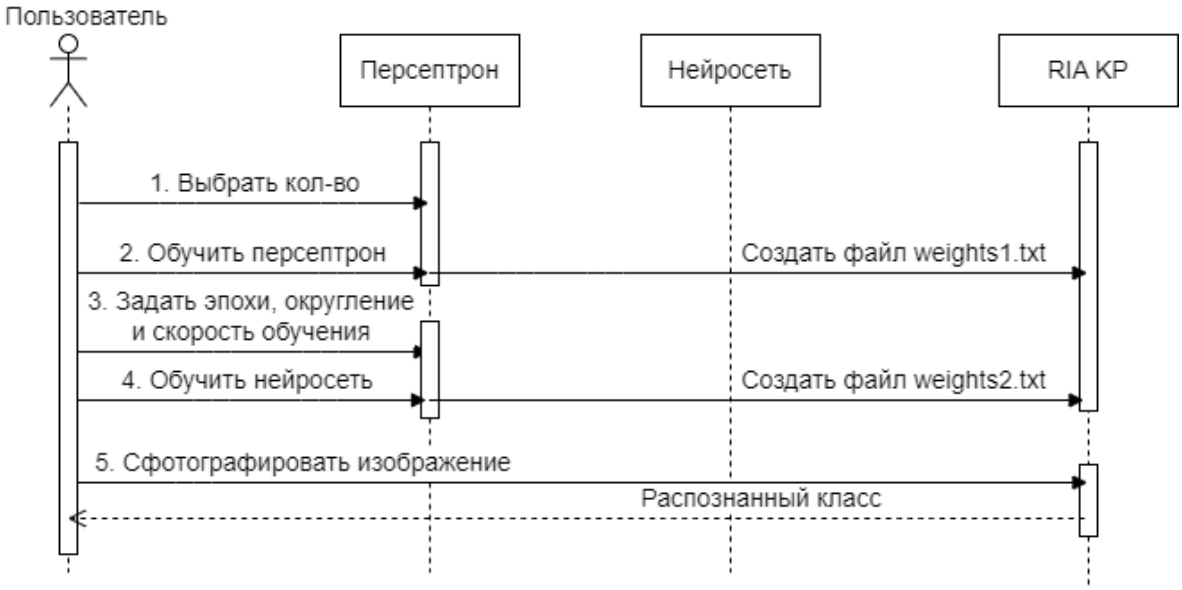


Рисунок 2.2 – Диаграмма последовательности

2.3 Диаграмма деятельности

Диаграмма деятельности приведена на рисунке 2.3. Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой, операции в предыдущем состоянии. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия, а дугами – переходы от одного состояния действия к другому.

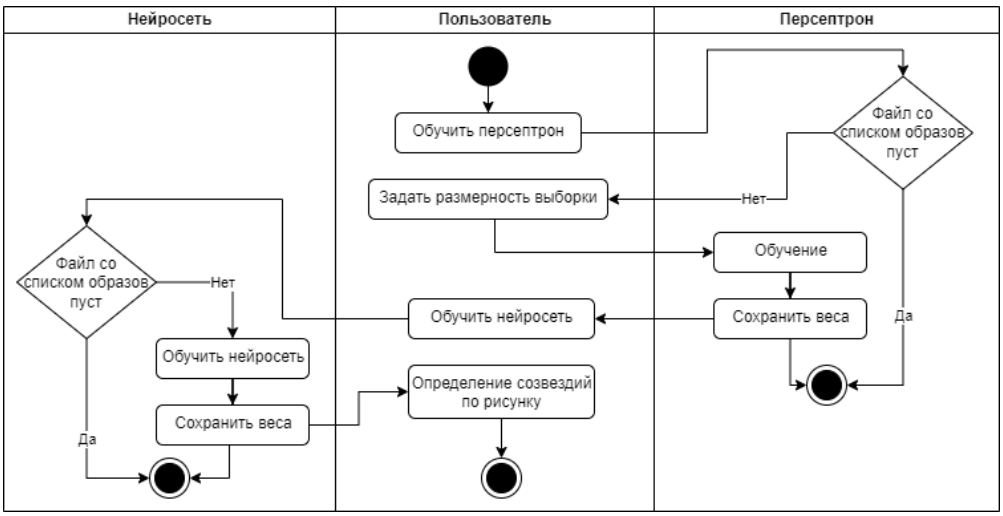


Рисунок 2.3 – Диаграмма деятельности

2.4 Диаграмма компонентов

Диаграмма компонентов приведена на рисунке 2.4.

Диаграмма компонентов описывает особенности физического представления системы. Она позволяет определить архитектуру 13 разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу.

Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

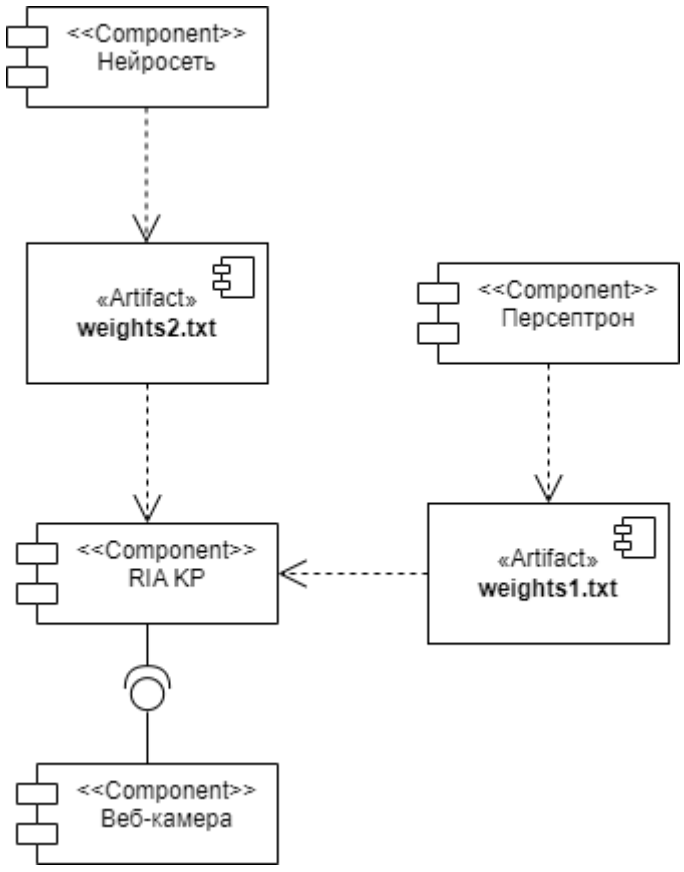


Рисунок 2.4 – Диаграмма компонентов

2.5 Диаграмма вариантов использования

Диаграмма вариантов использования приведена на рисунке 2.5.

Диаграмма вариантов использования в UML – диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Диаграмма вариантов использования рассматривается как главное средство для первичного моделирования динамики системы, используется для выяснения требований к разрабатываемой системе, фиксации этих требований в форме, которая позволит проводить дальнейшую разработку.

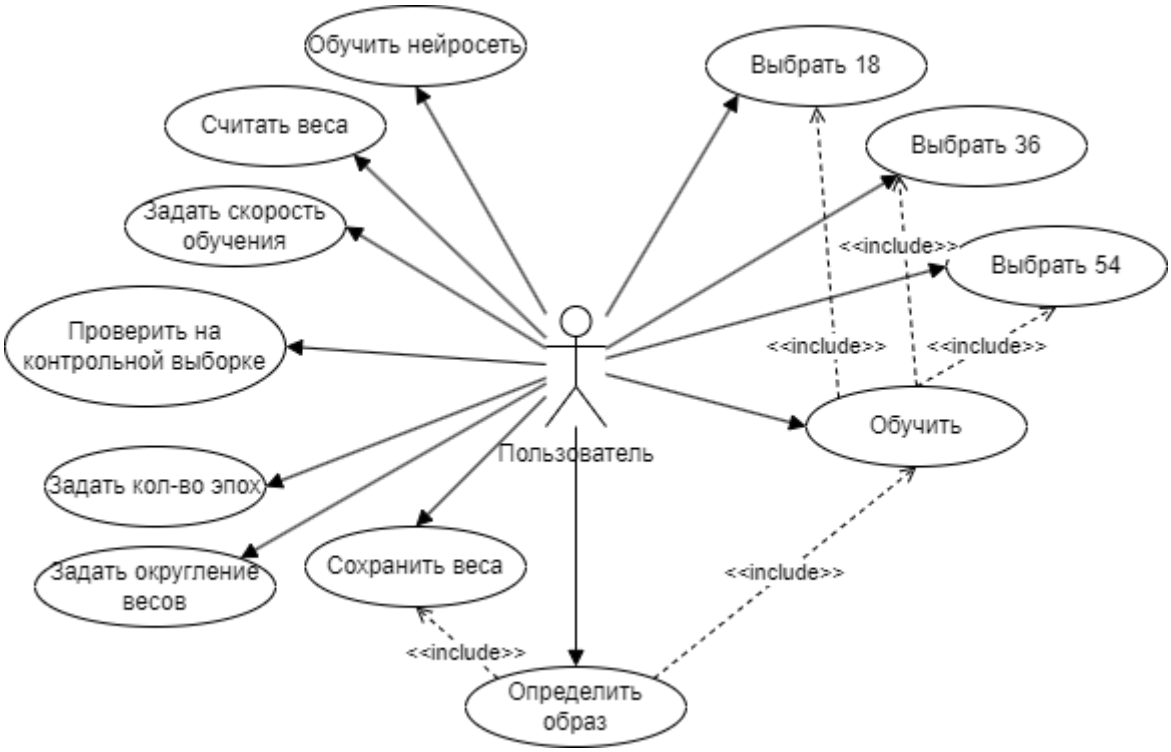


Рисунок 2.5 – Диаграмма использования

3 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1 Однослойный персептрон

На вход персептрона подаётся файл образов, данные в котором расположены в следующем порядке: номер, определитель (1 – созвездие, 0 - фон), сумма всех пикселей изображения, площадь изображения. Используемая выборка представлена в приложении Б.

На рисунке Г.1 представлена выборка из 18 образов. Первые 9 точек являются созвездиями, а другие 9 точек – фонами. При нажатии кнопки обучения линия разделяет классы. Наиболее верные веса получаются при обучении на самой большой выборке. После обучения веса автоматически сохраняются в файл weights1.txt. Пример успешного обучения на рисунке Г.2.

3.2 Многослойная нейросеть

На вход нейронной сети подаётся набор образов, состоящий из 162 картинок – по 18 штук каждого класса, а также контрольная выборка из 9 изображений для проверки. JSON-файл содержит в себе массив образов, где каждый из них имеет свой номер, определитель, высоту и ширину, сумму всех пикселей, площадь и матрицу изображения, которая состоит из нулей и единиц.

При старте программы происходит чтение файла и сжатие образов до формата 32 на 32 пикселя, с использованием класса Bitmap. На изображении Г.3 пользователь может выставить нужные ему параметры: скорость обучения, количество эпох, округление весов, а также считать уже готовые веса или записать их в файл weights2.txt. При неправильных значениях округления и скорости обучения, нейросеть не будет обучаться. На рисунке Г.4 представлены оптимальные данные, с которыми контрольная выборка всегда угадывается верно. На рисунке Г.5 представлена успешно обученная нейросеть для распознавания созвездий.

При нажатии кнопки Обучить каждый тренировочный образ будет отправлен в нейросеть, после чего сравнив выходной ответ с правильным веса будут пересчитаны.

3.3 Сайт со встроенным обнаружением

Для распознавания изображений, сайту требуются файлы weight1.txt и weight2.txt, содержащие веса персептрона и нейросети соответственно. При подключении веб-камеры на сайт выводится изображение с камеры, а также то, как это видео будет упрощено до чёрно-белой фотографии.

При нажатии кнопки сфотографировать образ будет выведен под кнопкой. Справа от кнопки будет написан тип обнаруженного изображения – фон или соответствующие созвездие. Также, если было обнаружено созвездие, внизу будет выведено увеличенное эталонное изображение из контрольной обучающей выборки.

На рисунках Г.6 – Г.16 представлены результаты определения образов.

ЗАКЛЮЧЕНИЕ

В ходе курсовой работы были разработаны программы: однослойный персептрон для разделения изображения на фон и образ, многослойная нейронная сеть для распознавания конкретного созвездия и сайт, способный определять рисунок созвездия по фотографиям с веб-камеры.

Для проектирования был использован графический язык моделирования UML, который позволил описать и сформулировать четкую структуру создаваемых приложений.

Для реализации был выбран язык программирования C# с использованием фреймворка .Net. Благодаря этому стала возможна простая необходимых классов и создание удобного и понятного интерфейса с использованием инструментов Windows Form.

Для построения диаграмм было использовано приложение Draw.io.

Работа была успешно протестирована на разнообразный пользовательский ввод и были добавлены обработчики всевозможных исключений. В результате была получена стабильная версия программы, корректно обрабатывающая любой ввод и отображающая верные данные.

ПЕРЕЧЕНЬ ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Вирт Н. Алгоритмы и структуры данных. — М.: Мир, 1989. — С. 272—286.

2. Д. Кнут. Искусство Программирования. Сортировка и Поиск. — С. 460.

3. Бенджио, Гудфеллоу, Курвилль Глубокое обучение. – С. 652.

4. Ю.Ю. Громов, О.Г. Иванова, В.В. Алексеев, М.П. Беляев, Д.П. Швец, А.И. Елисеев ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ. – С. 244.

5. Албахари, Албахари: C# 7.0. Карманный справочник. – М.: Вильямс, 2017. – 224 с.

6. Флэнаган Д. JavaScript. Полное руководство / Дэвид Флэнаган – М.: Диалектика, 2021. – 732 с.

Приложение А

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

А.1 Общие сведения

Тема курсовой работы: «Нейроморфный алгоритм обнаружения созвездий».

Курсовая работа разрабатывается студентом 3-го курса Донецкого национального технического университета, факультета ИСП, группы ПИ-20г, Пустовым В.А.

Основанием для разработки КП является задание, выданное кафедрой прикладной математики и искусственного интеллекта. Плановый срок начала работы по выполнению задания курсового проекта: 10.02.2023, срок окончания: 29.05.2023. Курсовой проект должен выполняться согласно графику, приведенному в таблице А.1

Таблица А.1 – Этапы, результаты и сроки разработки программного продукта.

№	Этап работы	Срок выполнения (№ недели)
1	Получение задания на КР	1
2	Изучение персептрона	2-3
3	Подготовка первичных данных	4
4	Разработка UML диаграмм	4
5	Разработка персептрона	5-6
6	Изучение многослойных нейронных сетей	7-8
7	Разработка нейросети	9-10
8	Отладка и тестирование нейросети	11-12
9	Разработка сайта RIA KP	13
10	Оформление пояснительной записки	14
11	Защита курсового проекта	15

А.2 Назначения и цели создания проекта

Целью создания программ является приобретение практических навыков в использовании нейросетей на примере создания обучающих моделей на языке C# и сайта со встроенной нейронной сетью для распознавания изображений.

А.3 Требования к курсовому проекту

Необходимо спроектировать программу при помощи графического языка UML, разработать программы обучения персептрона и нейронной сети, а также встроить их в сайт.

А.4 Требования к задачам и функциям программного продукта

В процессе работы необходимо обеспечить выполнение следующих функций:

– обучение персептрона;

– обучение нейросети;

– встроить проверку фон/образ и класса образа в RIA KP.html;

А.5 Требования к техническому обеспечению

К техническому обеспечению предъявляются следующие требования:

– процессор – 32-битный x86-совместимый (уровня Pentium и выше);

– объем оперативной памяти – не менее 1 Гб;

– свободное дисковое пространство – около 50 Мб. Не менее 5 Мб свободного дискового пространства для временных файлов;

– графический адаптер – VGA-совместимый;

– монитор – VGA-совместимый;

– клавиатура.

А.6 Требования к программному обеспечению

В курсовой работе должны быть соблюдены следующие требования к программному обеспечению:

– при разработке проекта должен использоваться любой объектно-ориентированный язык программирования;

– программный продукт должен обладать интуитивно понятным и легким в использовании интерфейсом.

А.7 Требования к организационному обеспечению

В программную документацию должны входить:

– пояснительная записка;

– приложения:

- техническое задание;
- первичные данные;
- листинг программных модулей;
- экранные формы.

Приложение Б

ПЕРВИЧНЫЕ ДАННЫЕ

Б.1 Эталонные изображения образов созвездий:

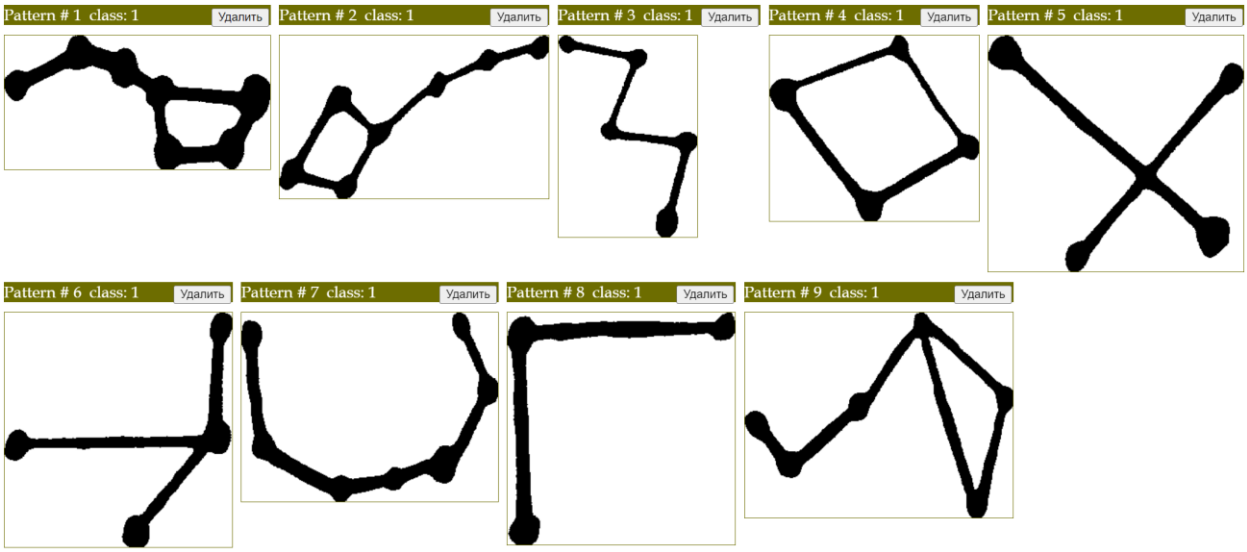
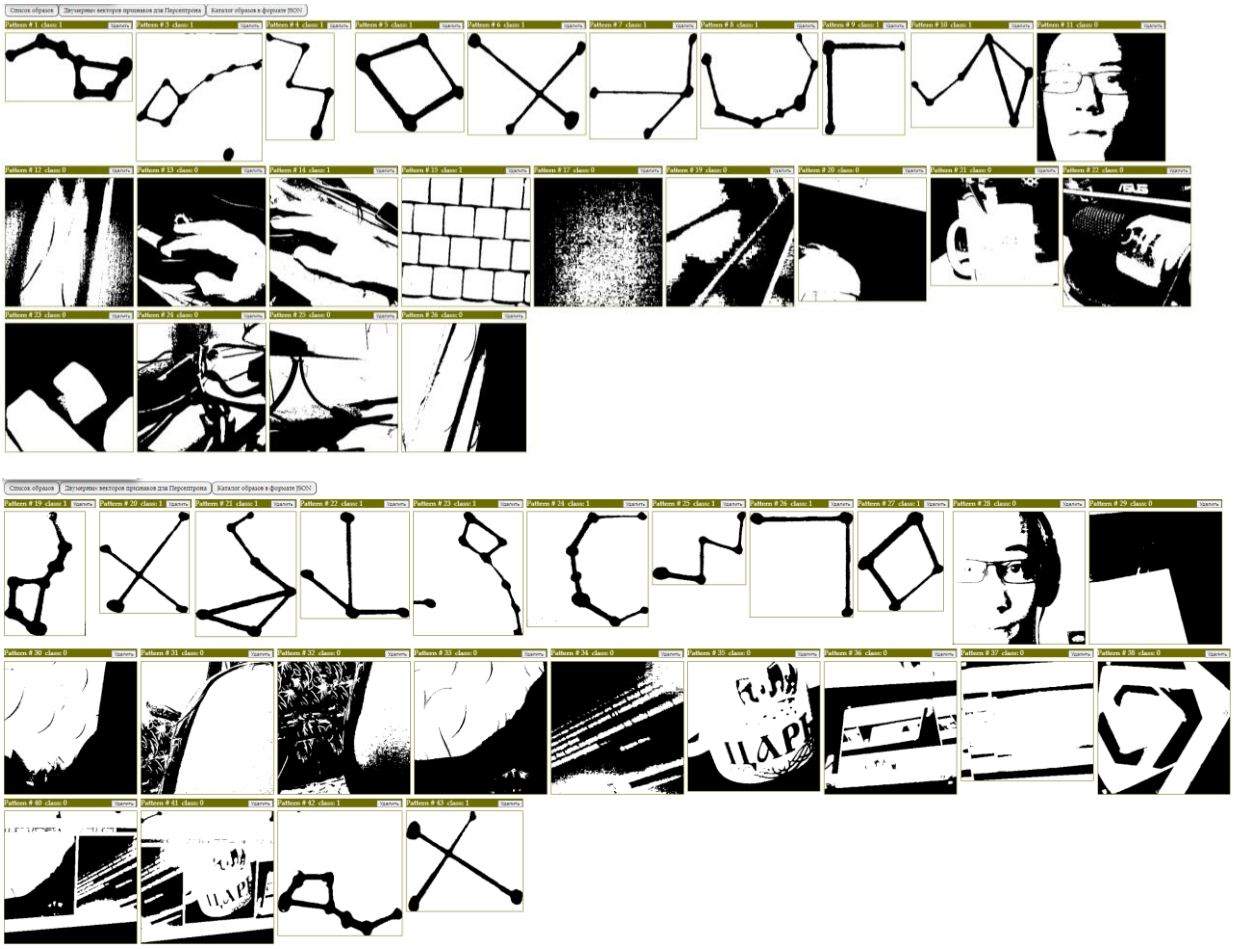


Рисунок Б.1.1 – Классы образов

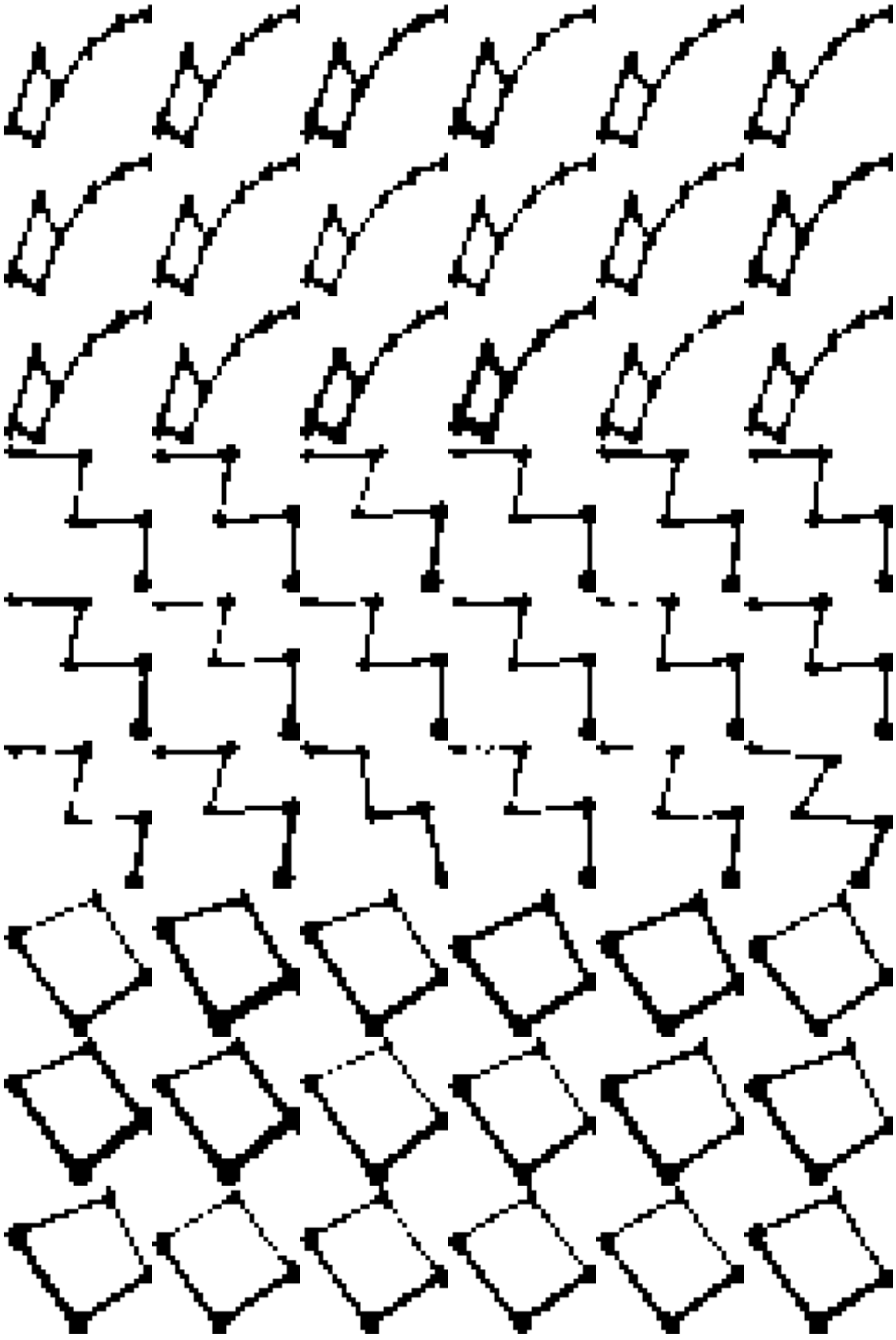
Б.2 Выборка для обучения персептрона:

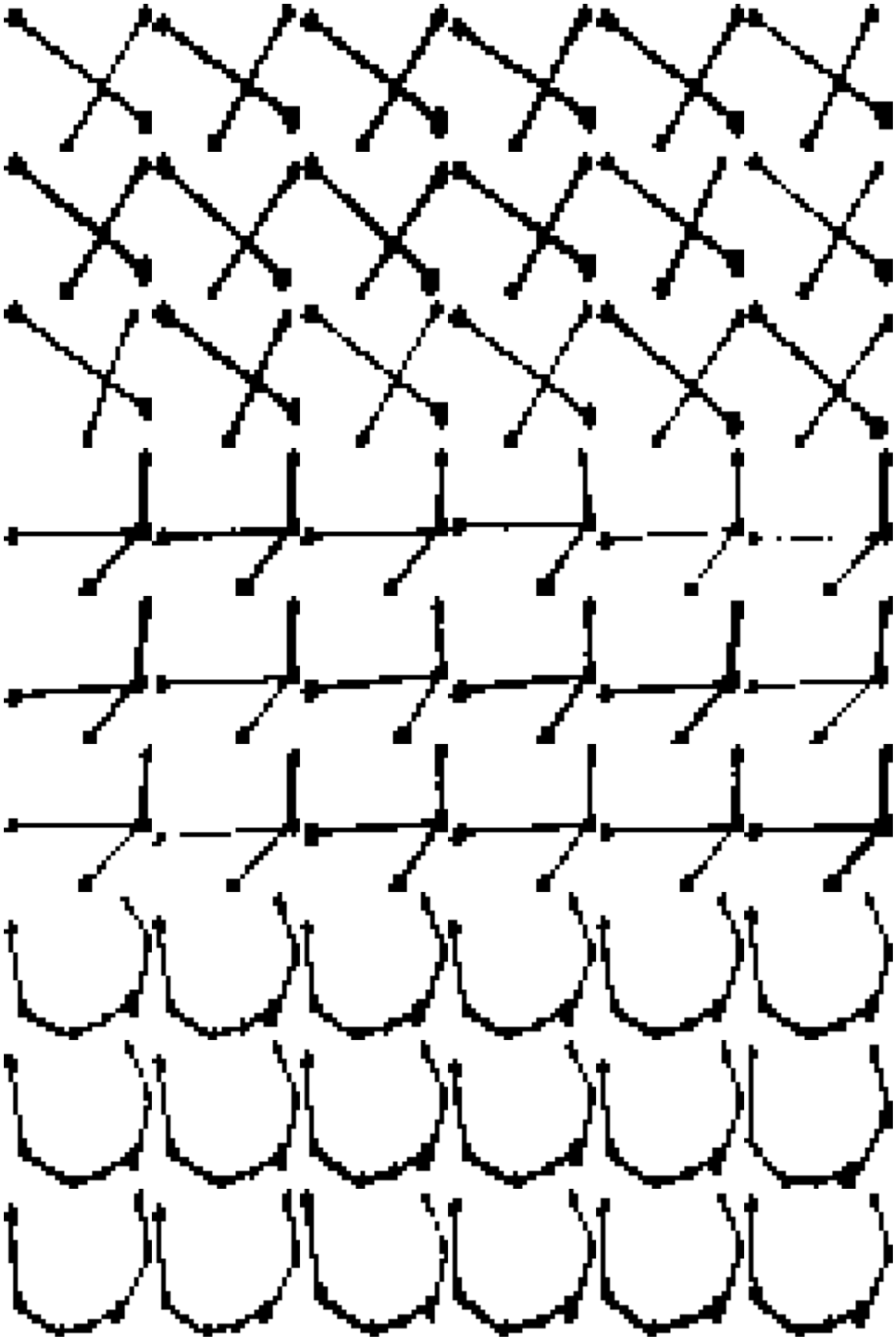
1,1,14699,64010,
2,1,6568,119707,
3,1,5258,53544,
4,1,14348,79355,
5,1,9544,88596,
6,1,5611,84096,
7,1,8505,82621,
8,1,6993,62048,
9,1,9433,84992,
10,1,11683,86580,
11,1,8685,94464,
12,1,13266,78771,
13,1,7562,119358,
14,1,9343,55590,
15,1,15680,84150,
16,1,6828,76755,
17,1,9148,75603,
18,1,4518,86387,
19,1,12107,69438,
20,1,6931,63012,
21,1,11963,87185,
22,1,7830,80647,
23,1,7089,93600,
24,1,8998,96338,
25,1,5818,47040,
26,1,7145,75621,
27,1,10389,58986,
28,0,60730,121801,
29,0,53361,121801,
30,0,80530,121801,
31,0,35058,121801,
32,0,32705,121801,
33,0,79195,121801,
34,0,25447,101559,
35,0,69845,121801,
36,0,77865,116566,
37,0,41595,101559,
38,0,86480,121801,
39,0,73789,121801,
40,0,68163,121801,
41,0,38486,121801,
42,0,53247,118311,
43,0,46963,121801,
44,0,83352,121801,
45,0,68735,121801,
46,0,82461,111680,
47,0,58364,121801,
48,0,84857,121801,
49,0,113939,121801,
50,0,60810,121801,
51,0,31249,97022,
52,0,88659,121801,
53,0,69126,121801,
54,0,52502,91709

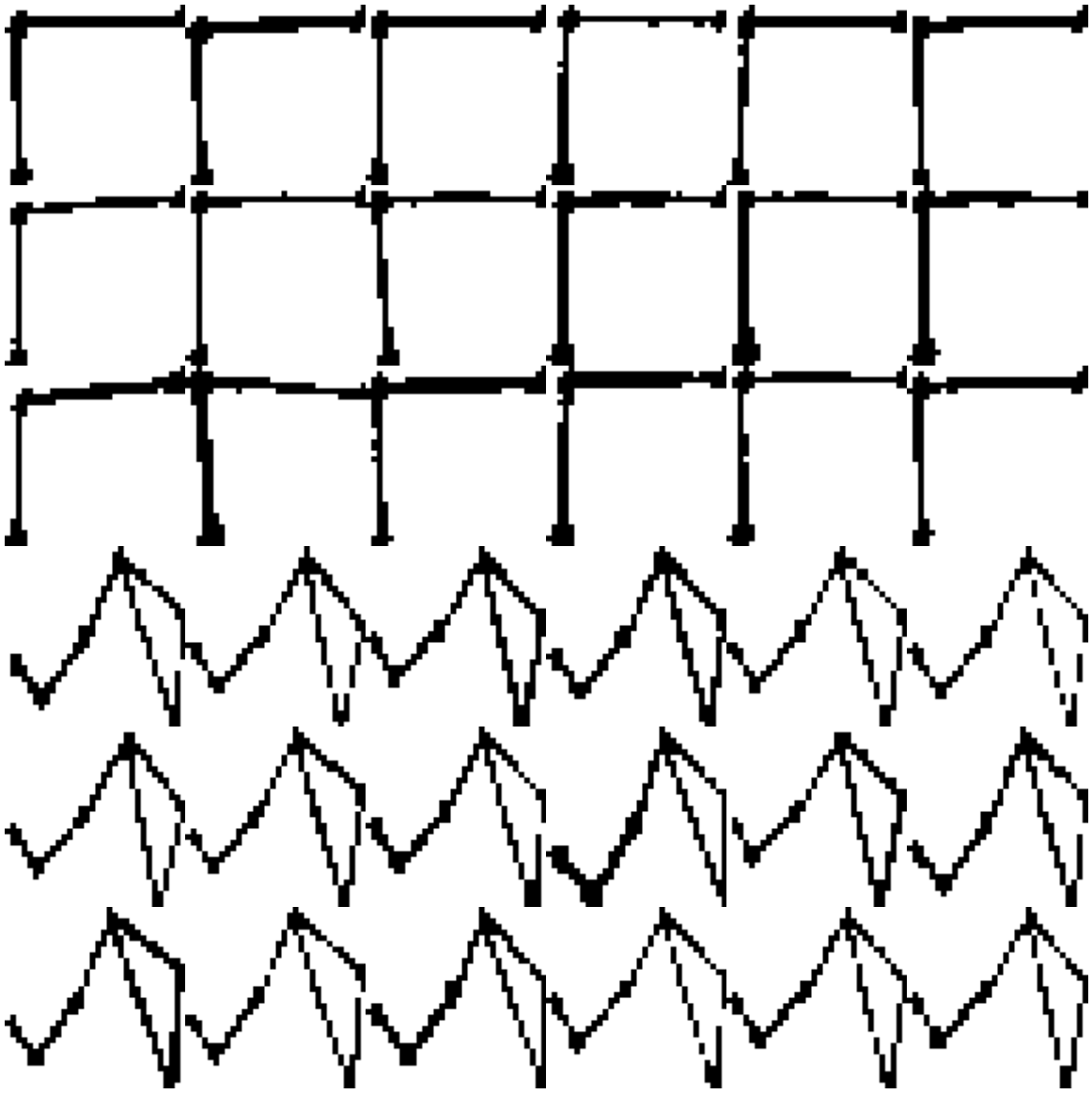
Б.3 Выборка для обучения персептрона в виде изображений:



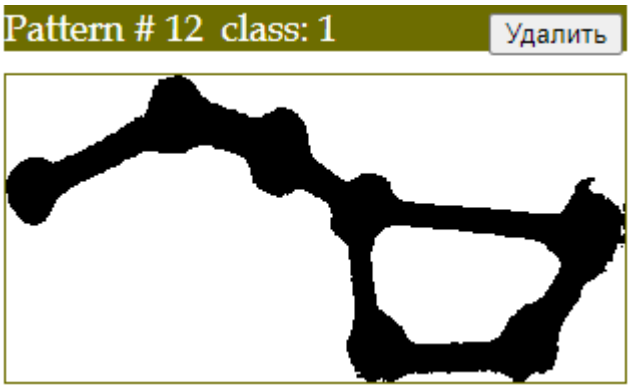
Б.4 Бинарные карты тренировочной выборки:



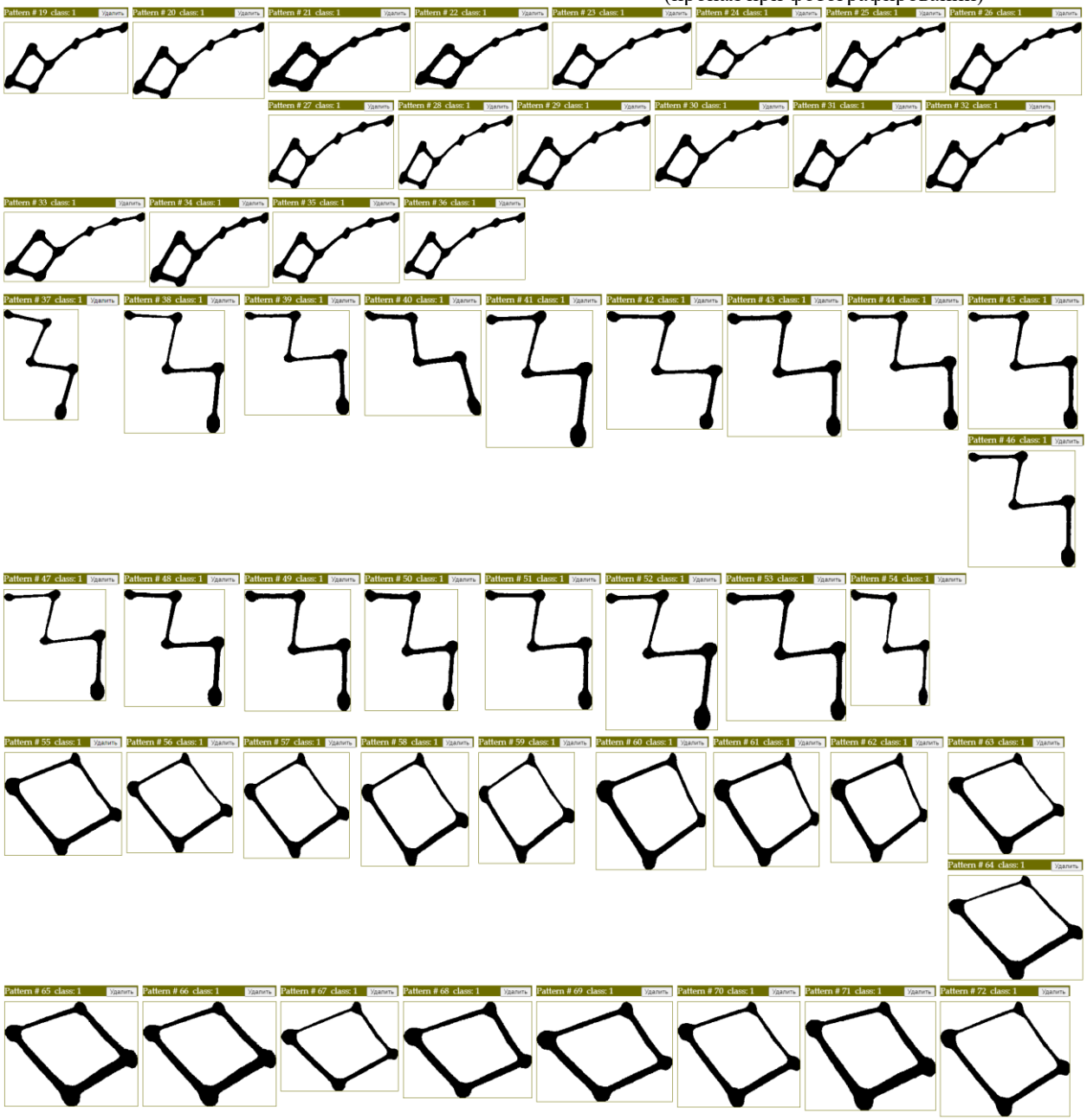


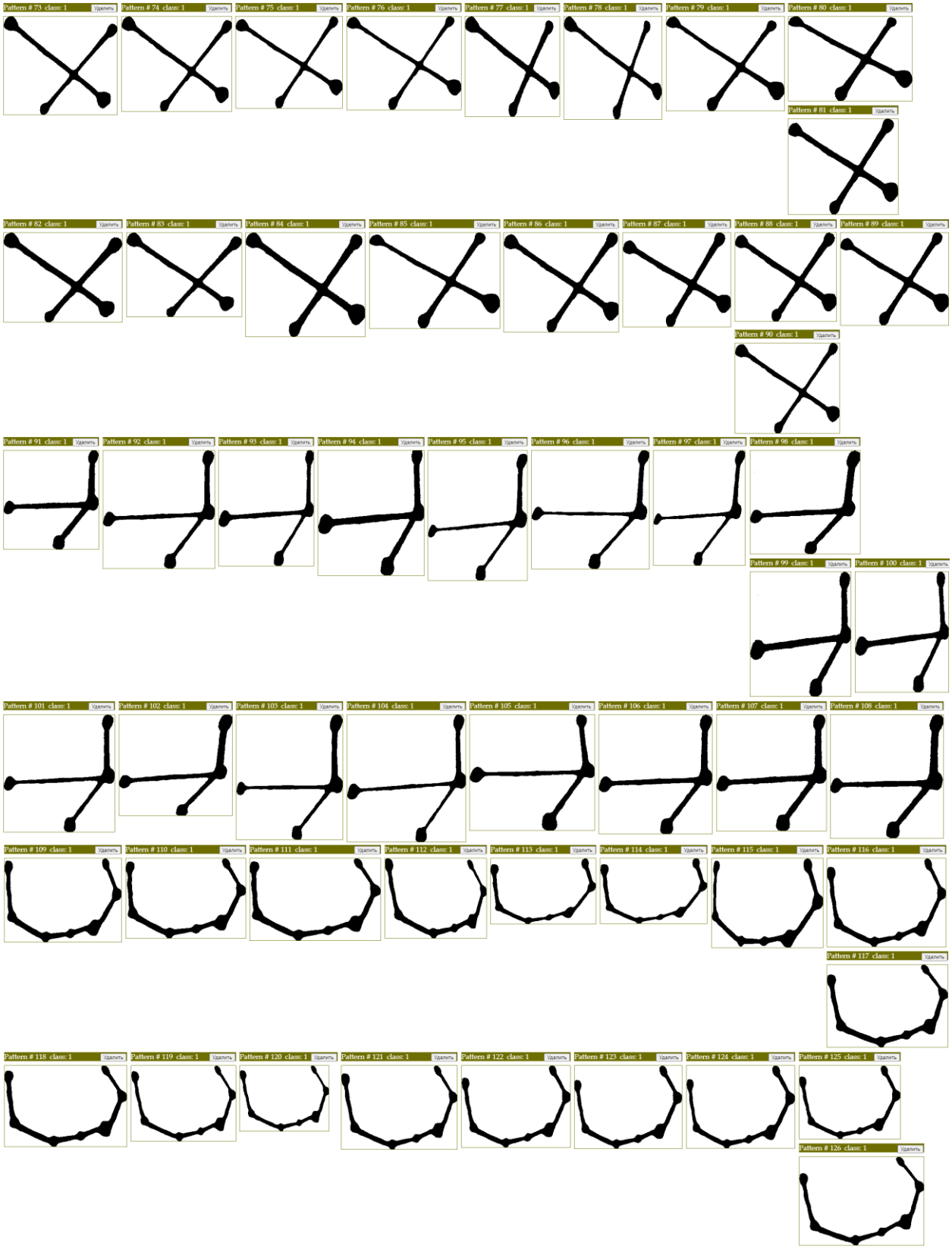


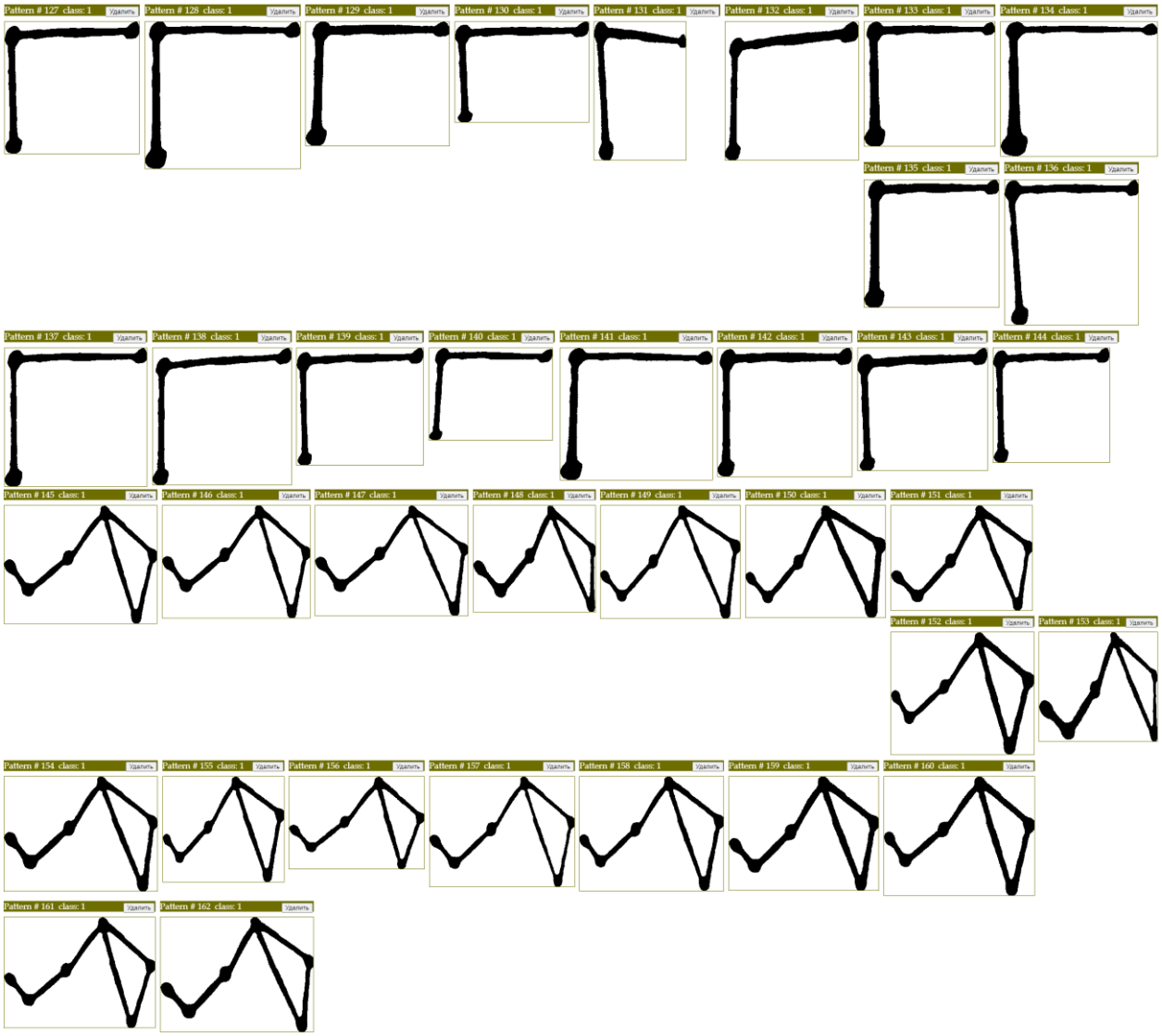
Б.5 Тренировочная выборка для обучения нейросети:



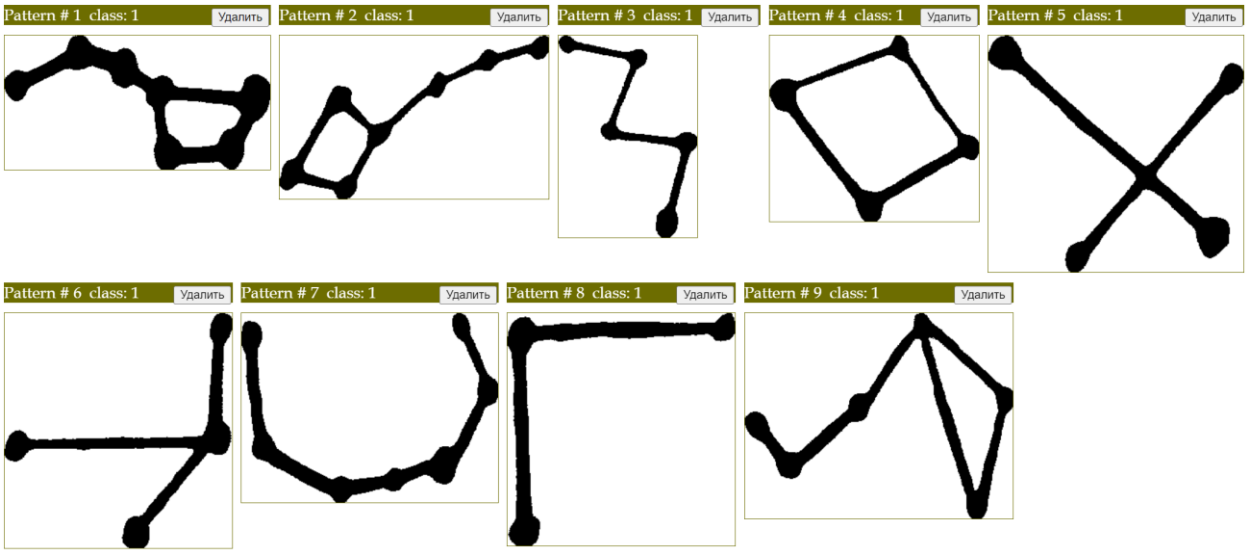
(пропал при фотографировании)




Pattern # 73 class: 1 Удалить
Pattern # 74 class: 1 Удалить
Pattern # 75 class: 1 Удалить
Pattern # 76 class: 1 Удалить
Pattern # 77 class: 1 Удалить
Pattern # 78 class: 1 Удалить
Pattern # 79 class: 1 Удалить
Pattern # 80 class: 1 Удалить
Pattern # 81 class: 1 Удалить
Pattern # 82 class: 1 Удалить
Pattern # 83 class: 1 Удалить
Pattern # 84 class: 1 Удалить
Pattern # 85 class: 1 Удалить
Pattern # 86 class: 1 Удалить
Pattern # 87 class: 1 Удалить
Pattern # 88 class: 1 Удалить
Pattern # 89 class: 1 Удалить
Pattern # 90 class: 1 Удалить
Pattern # 91 class: 1 Удалить
Pattern # 92 class: 1 Удалить
Pattern # 93 class: 1 Удалить
Pattern # 94 class: 1 Удалить
Pattern # 95 class: 1 Удалить
Pattern # 96 class: 1 Удалить
Pattern # 97 class: 1 Удалить
Pattern # 98 class: 1 Удалить
Pattern # 99 class: 1 Удалить
Pattern # 100 class: 1 Удалить
Pattern # 101 class: 1 Удалить
Pattern # 102 class: 1 Удалить
Pattern # 103 class: 1 Удалить
Pattern # 104 class: 1 Удалить
Pattern # 105 class: 1 Удалить
Pattern # 106 class: 1 Удалить
Pattern # 107 class: 1 Удалить
Pattern # 108 class: 1 Удалить
Pattern # 109 class: 1 Удалить
Pattern # 110 class: 1 Удалить
Pattern # 111 class: 1 Удалить
Pattern # 112 class: 1 Удалить
Pattern # 113 class: 1 Удалить
Pattern # 114 class: 1 Удалить
Pattern # 115 class: 1 Удалить
Pattern # 116 class: 1 Удалить
Pattern # 117 class: 1 Удалить
Pattern # 118 class: 1 Удалить
Pattern # 119 class: 1 Удалить
Pattern # 120 class: 1 Удалить
Pattern # 121 class: 1 Удалить
Pattern # 122 class: 1 Удалить
Pattern # 123 class: 1 Удалить
Pattern # 124 class: 1 Удалить
Pattern # 125 class: 1 Удалить
Pattern # 126 class: 1 Удалить



Б.6 Контрольная выборка для обучения нейросети:



Приложение В

ЛИСТИНГ ПРОГРАММНЫХ МОДУЛЕЙ

Листинг программы NS 1, обучающей персептрон:

Б.1 Файл App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
        </configuration>
```

Б.2 Файл Program.cs

```
using System;
using System.Windows.Forms;

namespace NS_1
{
  internal static class Program
  {
    /// <summary>
    /// Главная точка входа для приложения.
    /// </summary>
    [STAThread]
    static void Main()
    {
      Application.EnableVisualStyles();
      Application.SetCompatibleTextRenderingDefault(false);
      Application.Run(new Form1());
    }
  }
}
```

Б.3 Файл Form1.cs

```
using System;
using System.IO;
using System.Windows.Forms;

namespace NS_1
{
    public partial class Form1 : Form
    {
        Random realRnd = new Random();

        int k = 0;
        double w0;
        double wx;
        double wy;
        double nu;
        point[] p = new point[54];
        int epoch = 0;
        int count = 0;
        public class point
        {
            public double X;
            public double Y;
            public double W; //1-W1,0-W2
```

```
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            chart1.ChartAreas[0].AxisX.Minimum = -0.1;
            chart1.ChartAreas[0].AxisX.Maximum = 1.1;
            chart1.ChartAreas[0].AxisX.Interval = 0.2;
            chart1.ChartAreas[0].AxisY.Minimum = -0.1;
            chart1.ChartAreas[0].AxisY.Maximum = 1.1;
            chart1.ChartAreas[0].AxisY.Interval = 0.2;
            string str = "";
            string[] text;
            try
            {
                // Открываем файл для чтения
                using (StreamReader sr = new StreamReader("2.txt"))
                {
                    // Считываем содержимое файла и сохраняем в переменную str
                    str = sr.ReadToEnd();
                }
            }
            catch (IOException exc)
            {
                // Обработка возможных ошибок чтения файла
                MessageBox.Show("Ошибка чтения файла: " + exc.Message);
            }
            text = str.Split(',');
            int q = 0;
            for (int i = 0; i < text.GetLength(0); i += 4)
            {
                p[q] = new point();
                p[q].W = double.Parse(text[i + 1]);
                p[q].X = double.Parse(text[i + 2]);// Класс point заполнить данными
образа из практической работы No2.;
                p[q].Y = double.Parse(text[i + 3]);// Класс point заполнить данными из
практической работы No2.;
                //pics.Add(new picture(int.Parse(text[i + 1]), int.Parse(text[i + 2]),
int.Parse(text[i + 3])));
                q++;
            }
            Normalization(p);
        }

        public void GenParams()
        {
            w0 = (double)realRnd.Next(200) / 100 - 1;
            wx = (double)realRnd.Next(200) / 100 - 1;
            wy = (double)realRnd.Next(200) / 100 - 1;
            nu = realRnd.NextDouble();
            label1.Text = "w0 = " + w0 + ";\r\nwx = " + wx + ";\r\nwy = " + wy;
        }
        public void Normalization(point[] _p)
        {
            double minX = _p[0].X, maxX = _p[0].X;
            double minY = _p[0].Y, maxY = _p[0].Y;
            for (int i = 1; i < _p.Length; i++)
            {
                if(minX > _p[i].X) minX = _p[i].X;
                if(maxX < _p[i].X) maxX = _p[i].X;
                if(minY > _p[i].Y) minY = _p[i].Y;
                if(maxY < _p[i].Y) maxY = _p[i].Y;
            }
            for (int i = 0; i < _p.Length; i++)
            {
```

```
            _p[i].X = (_p[i].X - minX) / (maxX - minX);
            _p[i].Y = (_p[i].Y - minY) / (maxY - minY);
        }
    }
    public Form1()
    {
        InitializeComponent();
        GenParams();
    }
    public void Perseptron(int n)
    {
        double S;
        int y;
        for (int i = 0; i < n / 2; i++)
        {
            S = w0 + wx * p[i].X + wy * p[i].Y;
            if (S < 0) y = 1;
            else y = 0;
            if (y == 1 && p[i].W != 1)
            {
                k++;
                wx = wx + nu * p[i].X;
                wy = wy + nu * p[i].Y;
                w0 = w0 + nu;
            }
            else if (y == 0 && p[i].W != 0)
            {
                k++;

                wx = wx - nu * p[i].X;
                wy = wy - nu * p[i].Y;
                w0 = w0 - nu;
            }
        }
        for (int i = 27; i < 27 + n / 2; i++)
        {
            S = w0 + wx * p[i].X + wy * p[i].Y;
            if (S < 0) y = 1;
            else y = 0;
            if (y == 1 && p[i].W != 1)
            {
                k++;
                wx = wx + nu * p[i].X;
                wy = wy + nu * p[i].Y;
                w0 = w0 + nu;
            }
            else if (y == 0 && p[i].W != 0)
            {
                k++;

                wx = wx - nu * p[i].X;
                wy = wy - nu * p[i].Y;
                w0 = w0 - nu;
            }
        }
    }
    private void button1_Click(object sender, EventArgs e)
    {
        count = 0;
        for (int i = 0; i < 9; i++)
            chart1.Series["Созвездия"].Points.AddXY(p[i].X, p[i].Y);
        for (int i = 27; i < 27+9; i++)
            chart1.Series["Фоны"].Points.AddXY(p[i].X, p[i].Y);
        chart1.Series[2].Points.Clear();
```

```
        chart1.Series[2].Points.AddXY(0, (-wx * 0 - w0) / wy);
        chart1.Series[2].Points.AddXY(20, (-wx * 20 - w0) / wy);
        button5.Enabled = true;
        button1.Enabled = false;
        epoch = 1;
    }
    private void button2_Click(object sender, EventArgs e)
    {
        count = 0;
        for (int i = 0; i < 18; i++)
            chart1.Series["Созвездия"].Points.AddXY(p[i].X, p[i].Y);
        for (int i = 27; i < 27 + 18; i++)
            chart1.Series["Фоны"].Points.AddXY(p[i].X, p[i].Y);
        chart1.Series[2].Points.Clear();
        chart1.Series[2].Points.AddXY(0, (-wx * 0 - w0) / wy);
        chart1.Series[2].Points.AddXY(20, (-wx * 20 - w0) / wy);
        button5.Enabled = true;
        button2.Enabled = false;
        epoch = 2;
    }
    private void button3_Click(object sender, EventArgs e)
    {
        count = 0;
        for (int i = 0; i < 27; i++)
            chart1.Series["Созвездия"].Points.AddXY(p[i].X, p[i].Y);
        for (int i = 27; i < 27 + 27; i++)
            chart1.Series["Фоны"].Points.AddXY(p[i].X, p[i].Y);
        chart1.Series[2].Points.Clear();
        chart1.Series[2].Points.AddXY(0, (-wx * 0 - w0) / wy);
        chart1.Series[2].Points.AddXY(20, (-wx * 20 - w0) / wy);
        button5.Enabled = true;
        button3.Enabled = false;
        epoch = 3;
    }
    private void button4_Click(object sender, EventArgs e)
    {
        chart1.Series[0].Points.Clear();
        chart1.Series[1].Points.Clear();
        chart1.Series[2].Points.Clear();
        epoch = 0;
        count = 0;
        k = 0;
        GenParams();
        button1.Enabled = true;
        button2.Enabled = true;
        button3.Enabled = true;
        button5.Enabled = false;
    }
    private void button5_Click(object sender, EventArgs e)
    {
        count++;
        k = 0;
        if (epoch == 1)
            Perseptron(18);
        else if (epoch == 2)
            Perseptron(36);
        else
            Perseptron(54);

        if (k != 0)
        {
            button5_Click(sender, e);
        }
        else if (epoch == 1)
```

```
            {
                button2.Enabled = true;
            }
            else if (epoch == 2)
            {
                button3.Enabled = true;
            }
            button5.Enabled = false;
            chart1.Series[2].Points.Clear();
            label1.Text = "w0 = " + w0 + ";\r\nwx = " + wx + ";\r\nwy = " + wy;
            chart1.Series[2].Points.AddXY(0, (-wx * 0 - w0) / wy);
            chart1.Series[2].Points.AddXY(20, (-wx * 20 - w0) / wy);
            File.WriteAllText("weights1.txt", w0 + ";" + wx + ";" + wy);
        }
    }
}
```

4. Файл Form1.Designer.cs

```
namespace NS_1
{
    partial class Form1
    {
        /// <summary>
        /// Обязательная переменная конструктора.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        /// <param name="disposing">истинно, если управляемый ресурс должен быть удален;
иначе ложно.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Код, автоматически созданный конструктором форм Windows

        /// <summary>
        /// Требуемый метод для поддержки конструктора — не изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        private void InitializeComponent()
        {
            System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea1 = new
System.Windows.Forms.DataVisualization.Charting.ChartArea();
            System.Windows.Forms.DataVisualization.Charting.Legend legend1 = new
System.Windows.Forms.DataVisualization.Charting.Legend();
            System.Windows.Forms.DataVisualization.Charting.Series series1 = new
System.Windows.Forms.DataVisualization.Charting.Series();
            System.Windows.Forms.DataVisualization.Charting.Series series2 = new
System.Windows.Forms.DataVisualization.Charting.Series();
            System.Windows.Forms.DataVisualization.Charting.Series series3 = new
System.Windows.Forms.DataVisualization.Charting.Series();
            this.chart1 = new System.Windows.Forms.DataVisualization.Charting.Chart();
            this.button1 = new System.Windows.Forms.Button();
```

```
            this.button2 = new System.Windows.Forms.Button();
            this.button3 = new System.Windows.Forms.Button();
            this.button4 = new System.Windows.Forms.Button();
            this.button5 = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            ((System.ComponentModel.ISupportInitialize)(this.chart1)).BeginInit();
            this.SuspendLayout();
            //
            // chart1
            //
            this.chart1.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
            | System.Windows.Forms.AnchorStyles.Left)
            | System.Windows.Forms.AnchorStyles.Right)));
            chartArea1.Name = "ChartArea1";
            this.chart1.ChartAreas.Add(chartArea1);
            legend1.Name = "Legend1";
            this.chart1.Legends.Add(legend1);
            this.chart1.Location = new System.Drawing.Point(23, 8);
            this.chart1.Name = "chart1";
            series1.ChartArea = "ChartArea1";
            series1.ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;
            series1.Legend = "Legend1";
            series1.Name = "Созвездия";
            series2.ChartArea = "ChartArea1";
            series2.ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;
            series2.Legend = "Legend1";
            series2.Name = "Фоны";
            series3.ChartArea = "ChartArea1";
            series3.ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
            series3.Legend = "Legend1";
            series3.Name = "Series3";
            this.chart1.Series.Add(series1);
            this.chart1.Series.Add(series2);
            this.chart1.Series.Add(series3);
            this.chart1.Size = new System.Drawing.Size(676, 398);
            this.chart1.TabIndex = 0;
            this.chart1.Text = "chart1";
            //
            // button1
            //
            this.button1.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.button1.Location = new System.Drawing.Point(705, 10);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(105, 29);
            this.button1.TabIndex = 1;
            this.button1.Text = "Генер 18";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // button2
            //
            this.button2.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.button2.Location = new System.Drawing.Point(705, 45);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(105, 29);
```

```
            this.button2.TabIndex = 2;
            this.button2.Text = "Генер 36";
            this.button2.UseVisualStyleBackColor = true;
            this.button2.Click += new System.EventHandler(this.button2_Click);
            //
            // button3
            //
            this.button3.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.button3.Location = new System.Drawing.Point(705, 80);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(105, 29);
            this.button3.TabIndex = 3;
            this.button3.Text = "Генер 54";
            this.button3.UseVisualStyleBackColor = true;
            this.button3.Click += new System.EventHandler(this.button3_Click);
            //
            // button4
            //
            this.button4.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.button4.Location = new System.Drawing.Point(705, 115);
            this.button4.Name = "button4";
            this.button4.Size = new System.Drawing.Size(105, 29);
            this.button4.TabIndex = 4;
            this.button4.Text = "Очистить";
            this.button4.UseVisualStyleBackColor = true;
            this.button4.Click += new System.EventHandler(this.button4_Click);
            //
            // button5
            //
            this.button5.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.button5.Location = new System.Drawing.Point(705, 150);
            this.button5.Name = "button5";
            this.button5.Size = new System.Drawing.Size(105, 29);
            this.button5.TabIndex = 5;
            this.button5.Text = "Обучить";
            this.button5.UseVisualStyleBackColor = true;
            this.button5.Click += new System.EventHandler(this.button5_Click);
            //
            // label1
            //
            this.label1.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(711, 199);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(35, 13);
            this.label1.TabIndex = 6;
            this.label1.Text = "label1";
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(868, 450);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.button5);
            this.Controls.Add(this.button4);
```

```
            this.Controls.Add(this.button3);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.chart1);
            this.Name = "Form1";
            this.Text = "НС 1";
            this.Load += new System.EventHandler(this.Form1_Load);
            ((System.ComponentModel.ISupportInitialize)(this.chart1)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.DataVisualization.Charting.Chart chart1;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.Button button4;
        private System.Windows.Forms.Button button5;
        private System.Windows.Forms.Label label1;
    }
}
```

Листинг программы NS 2, обучающей нейросеть:

Б.5 Файл App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
    </startup>
       </configuration>
```

Б.6 Файл Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace NS_2
{
    internal static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Б.7 Файл NeuralNet.cs

```
using System;
using System.Text;
```

```
using System.Collections.Generic;
using System.IO;
using System.Windows.Forms;

namespace NS_2
{
    public class NeuralNet
    {
        const int img_size = 32;
        const int hid_count = 192;
        const int learn_img_count = 9;
        const int training_img_count = 162;//162

        double[][] inWeight = new double[img_size * img_size][];//входные веса 1024 на
192

        double[][] hidWeight = new double[hid_count][];//веса скрытого слоя 192 на 9
        double[] hidBias = new double[hid_count];//смещение скрытого слоя 192
        double[] hidOutput = new double[hid_count];//выходные числа входного слоя 192

        double[] outBias = new double[learn_img_count];//смещение выходного слоя 9
        public double[] output = new double[learn_img_count];//выходные значения сети 9

        public double learningRate = 0.05;

        public NeuralNet()
        {
            Random random = new Random();
            //Инициализация весов методом Xavier
            double inWeightScale = Math.Sqrt(1.0 / (inWeight.Length)); // Масштаб для
входных весов
            double hidWeightScale = Math.Sqrt(1.0 / (hidWeight.Length)); // Масштаб для
весов скрытого слоя

            for (int i = 0; i < inWeight.Length; i++)
            {
                inWeight[i] = new double[hid_count];
                for (int j = 0; j < inWeight[i].Length; j++)
                    inWeight[i][j] = random.NextDouble() * 2 * inWeightScale -
inWeightScale;
            }
            for (int i = 0; i < hidWeight.Length; i++)
            {
                hidWeight[i] = new double[learn_img_count];
                for (int j = 0; j < hidWeight[i].Length; j++)
                    hidWeight[i][j] = random.NextDouble() * 2 * hidWeightScale -
hidWeightScale;

                hidBias[i] = random.NextDouble() * 2 - 1;
            }
            for (int i = 0; i < outBias.Length; i++)
                outBias[i] = random.NextDouble() * 2 - 1;
        }

        public double Sigmoid(double x)
        {
            double num = 1 / (1 + Math.Exp(-x));
            return num;
        }

        public void Run(int[] _image)
        {
            //for (int n = 0; n < training_img_count; n++)//номер картинки
            //{
```

```
            //считаем выходные числа входного слоя
            //hidOutput[n] = new double[img_size * img_size];
            for (int i = 0; i < hidOutput.Length; i++)//номер нейрона
            {
                double sum = 0;
                for (int j = 0; j < inWeight.Length; j++)
                {
                    sum += _image[j] * inWeight[j][i];// i or j
                }
                hidOutput[i] = Sigmoid(sum + hidBias[i]);
            }

            //считаем выходные числа сети
            //output[n] = new double[learn_img_count];
            for (int i = 0; i < learn_img_count; i++)//номер нейрона
            {
                double sum = 0;
                for (int j = 0; j < hidWeight.Length; j++)
                {
                    sum += hidOutput[j] * hidWeight[j][i];//
                }
                output[i] = Sigmoid(sum + outBias[i]);
            }
        }

        public void BackProp(int[] _image, int n)
        {
            //for (int n = 0; n < training_img_count; n++)//номер картинки
            //{
            double[] outputError = new double[learn_img_count]; //ошибка выходного
значения
            int[] answers = new int[learn_img_count];           //правильный ответ
            answers[n / 18] = 1;
            //вычисление ошибок
            for (int i = 0; i < learn_img_count; i++)
                outputError[i] = (answers[i] - output[i]) * output[i] * (1 - output[i]);

            double[] hidErrors = new double[hid_count];//ошибка веса
            for (int i = 0; i < hidOutput.Length; i++)
            {
                double error = 0;
                for (int j = 0; j < hidWeight[i].Length; j++)
                {
                    error += outputError[j] * hidWeight[i][j];
                    //обновление весов скрытого слоя
                    hidWeight[i][j] += learningRate * outputError[j] * hidOutput[i];
                }
                hidErrors[i] = error * hidOutput[i] * (1 - hidOutput[i]);
                hidBias[i] += learningRate * hidErrors[i];
            }
            for (int i = 0; i < inWeight.Length; i++)
            {
                for (int j = 0; j < inWeight[i].Length; j++)
                    inWeight[i][j] += learningRate * hidErrors[j] * _image[i];
            }

            for (int i = 0; i < learn_img_count; i++)
                outBias[i] += learningRate * outputError[i];
            //}
        }

        public void Learning(int[][] _images, int epoches, TextBox tb, int round)
        {
            for (int epoch = 1; epoch <= epoches; epoch++)
```

```
            {
                double errorCount = 0;
                for (int n = 0; n < training_img_count; n++)//номер картинки
                {
                    Run(_images[n]);

                    int[] answers = new int[learn_img_count];//правильный ответ
                    answers[n / 18] = 1;
                    for (int i = 0; i < learn_img_count; i++)
                        errorCount += Math.Pow(answers[i] - output[i], 2);
                    BackProp(_images[n], n);
                    Rounding(round);
                }
                Console.WriteLine("Эпоха: " + epoch + " ошибок " + errorCount);
                tb.AppendText("Эпоха№ " + epoch + " ошибок: " + errorCount + "\r\n");
            }
        }

        public void Rounding(int round)
        {
            for (int i = 0; i < inWeight.Length; i++)
            {
                for (int j = 0; j < inWeight[i].Length; j++)
                    inWeight[i][j] = Math.Round(inWeight[i][j], round);
            }
            for (int i = 0; i < hidWeight.Length; i++)
            {
                for (int j = 0; j < hidWeight[i].Length; j++)
                    hidWeight[i][j] = Math.Round(hidWeight[i][j], round);
                hidBias[i] = Math.Round(hidBias[i], round);
            }
            for (int i = 0; i < outBias.Length; i++)
                outBias[i] = Math.Round(outBias[i], round);

        }
        public void WriteAll()
        {
            List<string> text = new List<string>();
            for (int i = 0; i < inWeight.Length; i++)
            {
                StringBuilder str = new StringBuilder();
                for (int j = 0; j < inWeight[i].Length - 1; j++)
                    str.Append(inWeight[i][j].ToString() + ';');
                str.Append(inWeight[i][inWeight[i].Length - 1].ToString());
                text.Add(str.ToString());
            }
            for (int i = 0; i < hidWeight.Length; i++)
            {
                StringBuilder str = new StringBuilder();
                for (int j = 0; j < hidWeight[i].Length - 1; j++)
                    str.Append(hidWeight[i][j].ToString() + ';');
                str.Append(hidWeight[i][hidWeight[i].Length - 1].ToString() + ';');
                text.Add(str.ToString());
            }
            text.Add(string.Join(";", hidBias));
            text.Add(string.Join(";", outBias));
            File.WriteAllLines("weights.txt", text.ToArray());
        }

        public void ReadAll()
        {
            string text = File.ReadAllText("weights.txt");
            string[] str = text.Split(new char[] { '\n', '\r' },
StringSplitOptions.RemoveEmptyEntries);
```

```
            string[] line;
            for (int i = 0; i < inWeight.Length; i++)
            {
                line = str[i].Split(';');
                for (int j = 0; j < inWeight[i].Length; j++)
                    inWeight[i][j] = double.Parse(line[j]);
            }

            for (int i = inWeight.Length; i < hidWeight.Length + inWeight.Length; i++)
            {
                line = str[i].Split(';');
                for (int j = 0; j < hidWeight[i - inWeight.Length].Length; j++)
                    hidWeight[i - inWeight.Length][j] = double.Parse(line[j]);
            }
            line = str[hidWeight.Length + inWeight.Length].Split(';');
            for (int j = 0; j < hidBias.Length; j++)
                hidBias[j] = double.Parse(line[j]);
            line = str[hidWeight.Length + inWeight.Length + 1].Split(';');
            for (int j = 0; j < outBias.Length; j++)
                outBias[j] = double.Parse(line[j]);
        }
    }
}
```

Б.8 Файл Form1.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Windows.Forms;
using Newtonsoft.Json.Linq;
using System.Drawing.Imaging;

namespace NS_2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        const int img_size = 32;
        const int hid_count = 192;
        const int learn_img_count = 9;
        const int training_img_count = 162;

        int[][] images = new int[training_img_count][];//Хранит 162 картинки в виде
строки из 0 и 1 размером 32*32

        NeuralNet nn = new NeuralNet();

        private void Form1_Load(object sender, EventArgs e)
        {
            string str = "";
            try
            {
                // Открываем файл для чтения
                using (StreamReader sr = new StreamReader("patterns.txt"))
                {
                    // Считываем содержимое файла и сохраняем в переменную str
                    str = sr.ReadToEnd();
                }
                JObject jo = JObject.Parse(str);
```

```
            int img_count = 0;
            foreach (KeyValuePair<string, JToken> pattern in jo)
            {
                JToken token = pattern.Value;
                JArray matrix = (JArray)token["matrix"];
                int height = (int)token["height"];
                int width = (int)token["width"];

                int[][] matrixdata = new int[height][];
                for (int i = 0; i < (int)token["height"]; i++)
                    matrixdata[i] = matrix[i].ToObject<int[]>();
                /*string text = "";
                for (int i = 0; i < height; i++)
                    for (int j = 0; j < width; j++)
                        text += matrixdata[i][j].ToString();
                textBox1.AppendText(text);*/
                Bitmap bm = new Bitmap(height, width);
                int pixel;
                for (int i = 0; i < height; i++)
                    for (int j = 0; j < width; j++)
                    {
                        pixel = matrixdata[i][j];
                        Color color = Color.FromArgb(pixel, pixel, pixel);
                        bm.SetPixel(i, j, color);
                    }
                bm = new Bitmap(bm, img_size, img_size);
                matrixdata = new int[img_size][];
                for (int i = 0; i < img_size; i++)
                {
                    matrixdata[i] = new int[img_size];
                    for (int j = 0; j < img_size; j++)
                    {
                        matrixdata[i][j] = bm.GetPixel(i, j).R;
                        //text += matrixdata[i][j];
                    }
                    //text += "\r\n";
                }
                //textBox1.AppendText(text);
                images[img_count] = new int[img_size * img_size];
                for (int i = 0; i < img_size; i++)
                    for (int j = 0; j < img_size; j++)
                        images[img_count][i * img_size + j] = matrixdata[i][j];
                img_count++;
            }
            textBox2.Text = "1";
            textBox3.Text = "10";
            textBox4.Text = "3";
            //SaveBinaryImages(images);
        }
        catch (IOException exc)
        {
            // Обработка возможных ошибок чтения файла
            MessageBox.Show("Ошибка чтения файла: " + exc.Message);
            Close();
        }
    }
    private void button1_Click(object sender, EventArgs e)
    {
        double d_num;
        int i_num3, i_num4;
        if (double.TryParse(textBox2.Text, out d_num))
        {
            nn.learningRate = d_num;
```

```
                    if      (int.TryParse(textBox3.Text,      out      i_num3)      &&
int.TryParse(textBox4.Text, out i_num4))
                        nn.Learning(images, i_num3, textBox1, i_num4);
                    else
                        MessageBox.Show("Число эпох и округление должны быть целыми
числами!");
                }
                else
                    MessageBox.Show("Неправильно введёна скорость обучения!");
            }

            private void button2_Click(object sender, EventArgs e)
            {
                nn.WriteAll();
            }

            private void button3_Click(object sender, EventArgs e)
            {
                nn.ReadAll();
            }

            private void button4_Click(object sender, EventArgs e)
            {
                string text = File.ReadAllText("test.txt");
                int[][] control_images = new int[learn_img_count][];
                JObject jo = JObject.Parse(text);
                int img_count = 0;
                foreach (KeyValuePair<string, JToken> pattern in jo)
                {
                    JToken token = pattern.Value;
                    JArray matrix = (JArray)token["matrix"];
                    int height = (int)token["height"];
                    int width = (int)token["width"];

                    int[][] matrixdata = new int[height][];
                    for (int i = 0; i < (int)token["height"]; i++)
                        matrixdata[i] = matrix[i].ToObject<int[]>();
                    Bitmap bm = new Bitmap(height, width);
                    int pixel;
                    for (int i = 0; i < height; i++)
                        for (int j = 0; j < width; j++)
                        {
                            pixel = matrixdata[i][j];
                            Color color = Color.FromArgb(pixel, pixel, pixel);
                            bm.SetPixel(i, j, color);
                        }
                    bm = new Bitmap(bm, img_size, img_size);
                    matrixdata = new int[img_size][];
                    for (int i = 0; i < img_size; i++)
                    {
                        matrixdata[i] = new int[img_size];
                        for (int j = 0; j < img_size; j++)
                            matrixdata[i][j] = bm.GetPixel(i, j).R;
                    }
                    control_images[img_count] = new int[img_size * img_size];
                    for (int i = 0; i < img_size; i++)
                        for (int j = 0; j < img_size; j++)
                            control_images[img_count][i    *    img_size    +    j]    =
matrixdata[i][j];
                    img_count++;
                }
                textBox1.AppendText("Попытка не пытка!\r\n");

                for (int i = 0; i < learn_img_count; i++)
```

```
                    {
                        nn.Run(control_images[i]);

                        double maxOut = nn.output[0];
                        int maxI = 0;

                        for (int j = 1; j < learn_img_count; j++)
                            if (maxOut < nn.output[j])
                            {
                                maxOut = nn.output[j];
                                maxI = j;
                            }
                        if (i == 0)
                            if (i == maxI)
                                textBox1.AppendText("Большая       медведица       угадана
верно.\r\n");
                            else
                                textBox1.AppendText("Большая медведица не угадана.\r\n");
                        if (i == 1)
                            if (i == maxI)
                                textBox1.AppendText("Малая медведица угадана верно.\r\n");
                            else
                                textBox1.AppendText("Малая медведица не угадана.\r\n");
                        if (i == 2)
                            if (i == maxI)
                                textBox1.AppendText("Кассиопея угадана верно.\r\n");
                            else
                                textBox1.AppendText("Кассиопея не угадана.\r\n");
                        if (i == 3)
                            if (i == maxI)
                                textBox1.AppendText("Лира угадана верно.\r\n");
                            else
                                textBox1.AppendText("Лира не угадана.\r\n");
                        if (i == 4)
                            if (i == maxI)
                                textBox1.AppendText("Лебедь угадана верно.\r\n");
                            else
                                textBox1.AppendText("Лебедь не угадана.\r\n");
                        if (i == 5)
                            if (i == maxI)
                                textBox1.AppendText("Орёл угадан верно.\r\n");
                            else
                                textBox1.AppendText("Орёл не угадан.\r\n");
                        if (i == 6)
                            if (i == maxI)
                                textBox1.AppendText("Северная корона угадана верно.\r\n");
                            else
                                textBox1.AppendText("Северная корона не угадана.\r\n");
                        if (i == 7)
                            if (i == maxI)
                                textBox1.AppendText("Волосы Вероники угаданы верно.\r\n");
                            else
                                textBox1.AppendText("Волосы Вероники не угаданы.\r\n");
                        if (i == 8)
                            if (i == maxI)
                                textBox1.AppendText("Весы угаданы верно.\r\n");
                            else
                                textBox1.AppendText("Весы не угаданы.\r\n");

                    }
                    int hehe = 0;
                    for (int q = 0; q < 111; q++)
                    {
                        for (int i = 0; i < learn_img_count; i++)
```

```
                    {
                        nn.Run(control_images[i]);

                        double maxOut = nn.output[0];
                        int maxI = 0;

                        for (int j = 1; j < learn_img_count; j++)
                            if (maxOut < nn.output[j])
                            {
                                maxOut = nn.output[j];
                                maxI = j;
                            }
                        if (i == maxI)
                            hehe++;
                    }
                }
                textBox1.AppendText("Шанс правильного ответа: " + (hehe * 100.0 / 999)
+ "%.");
                //SaveBinaryImages(control_images);
            }

            public void SaveBinaryImages(int[][] _images)
            {
                int scale = 16;
                for(int n = 0; n < _images.Length; n++)
                {
                    Bitmap bm = new Bitmap(img_size * scale, img_size * scale);
                    for(int y = 0; y < img_size; y++)
                        for(int x = 0; x < img_size; x++)
                        {
                            Color color = _images[n][y * img_size + x] == 1 ? Color.Black
: Color.White;

                            for (int i = 0; i < scale; i++)
                                for (int j = 0; j < scale; j++)
                                    bm.SetPixel(x * scale + i, y * scale + j, color);
                        }
                    string path = n.ToString() + ".png";
                    bm.Save(path, ImageFormat.Png);
                }

            }
        }
    }
```

Б.9 Файл Form1.Designer.cs

```
namespace NS_2
{
    partial class Form1
    {
        /// <summary>
        /// Обязательная переменная конструктора.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        /// <param name="disposing">истинно, если управляемый ресурс должен быть удален;
иначе ложно.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
```

```
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Код, автоматически созданный конструктором форм Windows

        /// <summary>
        /// Требуемый метод для поддержки конструктора — не изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        private void InitializeComponent()
        {
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.button1 = new System.Windows.Forms.Button();
            this.button2 = new System.Windows.Forms.Button();
            this.button3 = new System.Windows.Forms.Button();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.textBox3 = new System.Windows.Forms.TextBox();
            this.button4 = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.textBox4 = new System.Windows.Forms.TextBox();
            this.SuspendLayout();
            //
            // textBox1
            //
            this.textBox1.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
            | System.Windows.Forms.AnchorStyles.Left)
            | System.Windows.Forms.AnchorStyles.Right)));
            this.textBox1.Location = new System.Drawing.Point(14, 9);
            this.textBox1.Multiline = true;
            this.textBox1.Name = "textBox1";
            this.textBox1.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
            this.textBox1.Size = new System.Drawing.Size(247, 269);
            this.textBox1.TabIndex = 0;
            //
            // button1
            //
            this.button1.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.button1.Location = new System.Drawing.Point(267, 10);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(111, 28);
            this.button1.TabIndex = 1;
            this.button1.Text = "Обучить";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // button2
            //
            this.button2.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.button2.Location = new System.Drawing.Point(267, 44);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(111, 28);
            this.button2.TabIndex = 2;
            this.button2.Text = "Сохранить веса";
            this.button2.UseVisualStyleBackColor = true;
```

```
            this.button2.Click += new System.EventHandler(this.button2_Click);
            //
            // button3
            //
            this.button3.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.button3.Location = new System.Drawing.Point(267, 78);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(111, 28);
            this.button3.TabIndex = 3;
            this.button3.Text = "Считать веса";
            this.button3.UseVisualStyleBackColor = true;
            this.button3.Click += new System.EventHandler(this.button3_Click);
            //
            // textBox2
            //
            this.textBox2.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.textBox2.Location = new System.Drawing.Point(267, 166);
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(108, 20);
            this.textBox2.TabIndex = 4;
            this.textBox2.Text = "Скорость обучения";
            //
            // textBox3
            //
            this.textBox3.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.textBox3.Location = new System.Drawing.Point(267, 205);
            this.textBox3.Name = "textBox3";
            this.textBox3.Size = new System.Drawing.Size(108, 20);
            this.textBox3.TabIndex = 5;
            this.textBox3.Text = "Кол-во эпох";
            //
            // button4
            //
            this.button4.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.button4.Location = new System.Drawing.Point(267, 112);
            this.button4.Name = "button4";
            this.button4.Size = new System.Drawing.Size(111, 28);
            this.button4.TabIndex = 6;
            this.button4.Text = "Проверка";
            this.button4.UseVisualStyleBackColor = true;
            this.button4.Click += new System.EventHandler(this.button4_Click);
            //
            // label1
            //
            this.label1.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(272, 150);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(104, 13);
            this.label1.TabIndex = 7;
            this.label1.Text = "Скорость обучения";
            //
            // label2
            //
```

```
            this.label2.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(272, 189);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(67, 13);
            this.label2.TabIndex = 8;
            this.label2.Text = "Кол-во эпох";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(272, 228);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(67, 13);
            this.label3.TabIndex = 9;
            this.label3.Text = "Округление";
            //
            // textBox4
            //
            this.textBox4.Location = new System.Drawing.Point(267, 244);
            this.textBox4.Name = "textBox4";
            this.textBox4.Size = new System.Drawing.Size(108, 20);
            this.textBox4.TabIndex = 10;
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(387, 296);
            this.Controls.Add(this.textBox4);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.button4);
            this.Controls.Add(this.textBox3);
            this.Controls.Add(this.textBox2);
            this.Controls.Add(this.button3);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.textBox1);
            this.Name = "Form1";
            this.Text = "Созвездия";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.Button button4;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.TextBox textBox4;
    }
}
```

Б.10 Файл RIA KP.html

```
<!DOCTYPE html>
<html lang="ru">
<head>
<meta charset="UTF-8">
<title>КП Созвездия</title>
<style type=text/css>
video#camera-stream {background-color: black;}
canvas#hidden-canvas {display: none;}
canvas#output-canvas {background-color: black;}
canvas#hidden-box-canvas {display: none;}
canvas#output-box-canvas {background-color: black; box-shadow: rgb(0,0,0) 1px 1px 20px;}
.subtitlestyle {color: white; font: 14pt Book Antiqua; background:rgb(109, 109, 0); padding:
0.4em 0.6em;
margin-top: 0.6em;}
</style>
</head>
<body>
<video id="camera-stream"></video>
<canvas id="hidden-canvas"></canvas>
<canvas id="output-canvas"></canvas>
<div style="clear:left;"></div>
<input style="float:left; font: 14pt Book Antiqua; padding: .4em 0.8em; border-radius: 12px;"
type="button" value="Сфотографировать" id="buttonSnap" onclick="snapVideo()">
<div style="font: 14pt Book Antiqua; padding: .4em 0.8em;" id="idPatternNote"></div>
<div style="clear:left;"></div>
<canvas id="hidden-box-canvas"></canvas>
<canvas id="output-box-canvas"></canvas>
<div id="panel0" style="clear:both;"></div>
<div style="font: 18pt Book Antiqua;"><br>Соответстующий эталон:</div>
<!-- Copyright (c) 2023 goodarget -->
<script src="neuralnetwork.js"></script>
</body>
</html>
```

Б.11 Файл neuralnetwork.js

```
globalVideoPlay = false;
globalBoxTop = 65;
globalBoxLeft = 150;
globalBoxWidth = 350;
globalBoxLineWidth = 20;
globalBoxLineHalf = Math.round(globalBoxLineWidth / 2);
globalParamX1 = 0;
globalParamX2 = 0;
globalParamWidth = 0;
globalParamHeight = 0;
globalMatrix = [];
const video = document.querySelector('#camera-stream');
const hiddenCanvas = document.querySelector('#hidden-canvas');
const outputCanvas = document.querySelector('#output-canvas');
const hiddenBoxCanvas = document.querySelector('#hidden-box-canvas');
const hiddenContext = hiddenCanvas.getContext('2d');
const outputContext = outputCanvas.getContext('2d');
const hiddenBoxContext = hiddenBoxCanvas.getContext('2d');
const constraints = {
video: {
width: 640,
height: 480
}
};
```

```
const processFrame = () => {
if (globalVideoPlay || globalVideoPlay == undefined) {
return;
}
globalVideoPlay = true;
const { videoWidth: width, videoHeight: height } = video;
if (width && height) {
hiddenCanvas.width = width;
hiddenCanvas.height = height;
outputCanvas.width = width;
outputCanvas.height = height;
hiddenContext.drawImage(video, 0, 0, width, height);
outputContext.drawImage(video, 0, 0, width, height);
var imageData0 = hiddenContext.getImageData(globalBoxLeft, globalBoxTop, globalBoxWidth,
globalBoxWidth);
var pixels0 = imageData0.data;
var imageData = hiddenContext.createImageData(globalBoxWidth, globalBoxWidth);
var pixels = imageData.data;
var txh = 90;
for (var i = 0; i < pixels0.length; i += 4) {
var grayscalePixel = ((0.3*pixels0[i]) + (0.59*pixels0[i+1]) + (0.11*pixels0[i+2]));
if (grayscalePixel > txh) {
pixels[i] = grayscalePixel;
pixels[i+1] = grayscalePixel;
pixels[i+2] = grayscalePixel;
pixels[i+3] = 255;
}
}
outputContext.putImageData(imageData, globalBoxLeft, globalBoxTop);
outputContext.lineWidth = globalBoxLineWidth;
outputContext.strokeStyle = "rgba(0,255,0,0.3)";
outputContext.lineCap = "square";
outputContext.moveTo(globalBoxLeft - globalBoxLineHalf, globalBoxTop);
outputContext.lineTo(globalBoxLeft - globalBoxLineHalf, globalBoxTop + globalBoxWidth -
globalBoxLineHalf);
outputContext.moveTo(globalBoxLeft - globalBoxLineHalf, globalBoxTop - globalBoxLineHalf);
outputContext.lineTo(globalBoxLeft + globalBoxWidth - globalBoxLineHalf, globalBoxTop -
globalBoxLineHalf);
outputContext.moveTo(globalBoxLeft + globalBoxWidth + globalBoxLineHalf, globalBoxTop -
globalBoxLineHalf);
outputContext.lineTo(globalBoxLeft + globalBoxWidth + globalBoxLineHalf, globalBoxTop +
globalBoxWidth - globalBoxLineHalf);
outputContext.moveTo(globalBoxLeft - globalBoxLineHalf, globalBoxTop + globalBoxWidth +
globalBoxLineHalf);
outputContext.lineTo(globalBoxLeft + globalBoxWidth + globalBoxLineHalf, globalBoxTop +
globalBoxWidth + globalBoxLineHalf);
outputContext.stroke();
}
globalVideoPlay = false;
window.requestAnimationFrame(processFrame);
};
if (navigator.webkitGetUserMedia) {
navigator.mediaDevices.getUserMedia(constraints).then(function (stream) {
video.srcObject = stream;
video.play();
}, function (err) {
console.error(err);
});
} else {
navigator.mediaDevices.getUserMedia(constraints).then(function (stream) {
video.srcObject = stream;
video.play();
```

```
})
.catch(function (err) {
console.error(err);
});
}
video.addEventListener('play', function () {
window.requestAnimationFrame(processFrame);
console.log('Live video!');
});
function snapVideo() {
var canvas = document.querySelector('#output-box-canvas');
var context = canvas.getContext('2d');
var raw = {width: globalBoxWidth, height: globalBoxWidth};
hiddenBoxCanvas.width = raw.width;
hiddenBoxCanvas.height = raw.height;
canvas.width = raw.width;
canvas.height = raw.height;
var imageData0 = outputContext.getImageData(globalBoxLeft, globalBoxTop, globalBoxWidth,
globalBoxWidth);
var pixels0 = imageData0.data;
var imageData = hiddenBoxContext.createImageData(globalBoxWidth, globalBoxWidth);
var pixels = imageData.data;
var boxMatrix = [];
for (var i = 0; i < raw.height; i++) {
var vector = [];
for (var j = 0; j < raw.width; j++) {
vector.push(0);
}
boxMatrix.push(vector);
}
var d = 4 * raw.width;
for (var i = 0; i < pixels0.length; i += 4) {
var y0 = Math.floor(i/d);
var x0 = Math.floor((i - d*y0)/4);
if (pixels0[i] != 0 ) {
pixels[i] = 255;
pixels[i+1] = 255;
pixels[i+2] = 255;
pixels[i+3] = 255;
boxMatrix[y0][x0]=0;
} else {
boxMatrix[y0][x0]=1;
}
}
context.putImageData(imageData, 0, 0);
var tx = 3;
var segment = {Top: 5, Left: 5, Bottom: raw.height - 5, Right: raw.width - 5};
var histogramX = vectorSumX({width: raw.width, height: raw.height, array: boxMatrix});
for (var i = 0; i < histogramX.length; i++) { if (histogramX[i] > tx) { segment.Left = i;
break; } };
for (var i = histogramX.length - 1; i > -1 ; i--) { if (histogramX[i] > tx) { segment.Right =
i; break; } };
var histogramY = vectorSumY({width: raw.width, height: raw.height, array: boxMatrix});
for (var i = 0; i < histogramY.length; i++) { if (histogramY[i] > tx) { segment.Top = i;
break; } };
for (var i = histogramY.length - 1; i > -1 ; i--) { if (histogramY[i] > tx) { segment.Bottom
= i; break; } };
var measure = (segment.Bottom - segment.Top) * (segment.Right - segment.Left);
var sum = 0;
var segmentMatrix = [];
for (var i = segment.Top; i < segment.Bottom + 1; i++) {
var vector = [];
```

```
for (var j = segment.Left; j < segment.Right + 1; j++) {
var a = boxMatrix[i][j];
sum = sum + a;
vector.push(a);
}
segmentMatrix.push(vector);
}
globalMatrix = segmentMatrix;

Read(function(weights) {
  globalParamX1 = sum;
  globalParamX2 = measure;
  globalParamWidth = segment.Right - segment.Left;
  globalParamHeight = segment.Bottom - segment.Top;;
  var panelOut = document.getElementById('panel0');
  fetch("weights1.txt")
    .then(function(response) {
      if (response.ok) {
        return response.text();
      } else {
        throw new Error("Error: " + response.status);
      }
    })
    .then(function(content) {
      var values = content.split(";");
      var w0 = parseFloat(values[0].replace(",", "."));
      var wx = parseFloat(values[1].replace(",", "."));
      var wy = parseFloat(values[2].replace(",", "."));
    console.log("w0 = " + w0);
    console.log("wx = " + wx);
    console.log("wy = " + wy);
  var S = w0 + wx * ((sum - 4518) / (113939 - 4518)) + wy * ((measure - 53544) / (121801 - 53544));
  console.log("S = " + S);
  if(S < 0)
  {
    var scaledMatrix = Scale(globalMatrix);
    console.log(scaledMatrix);

    var predictedIndex = Predict(scaledMatrix, weights);
    console.log(predictedIndex);
    var constellation;
switch (predictedIndex) {
  case 0:
    show_image('images/0.png', 512, 512);
    constellation = "Большая медведица";
    break;
  case 1:
    show_image('images/1.png', 512, 512);
    constellation = "Малая медведица";
    break;
  case 2:
    show_image('images/2.png', 512, 512);
    constellation = "Кассиопея";
    break;
  case 3:
    show_image('images/3.png', 512, 512);
    constellation = "Лира";
    break;
  case 4:
    show_image('images/4.png', 512, 512);
    constellation = "Лебедь";
    break;
```

```
  case 5:
    show_image('images/5.png', 512, 512);
    constellation = "Орла";
    break;
  case 6:
    show_image('images/6.png', 512, 512);
    constellation = "Северная корона";
    break;
  case 7:
    show_image('images/7.png', 512, 512);
    constellation = "Волопас";
    break;
  case 8:
    show_image('images/8.png', 512, 512);
    constellation = "Весов";
    break;
  default:
    constellation = "...";
}

    var patternNoteElement = document.getElementById("idPatternNote");
    patternNoteElement.innerHTML = "Обнаружен образ созвездия " + constellation;
    panelOut.innerHTML = 'Параметры X1 "Сумма единичных пикселей": ' + sum + '<br>' +
  'Параметры X2 "Площадь сегмента, пикселей": ' + measure +'<br>';
  }
  else
  {
    var patternNoteElement = document.getElementById("idPatternNote");
    patternNoteElement.innerHTML = "Обнаружен фон.";
    panelOut.innerHTML = 'Параметры X1 "Сумма единичных пикселей": ' + sum + '<br>' +
  'Параметры X2 "Площадь сегмента, пикселей": ' + measure +'<br>';
  var previousImage = document.getElementById("dynamic-image");
    if (previousImage) {
        previousImage.remove(); // Удаляем предыдущий элемент картинки
    }
  }
})
.catch(function(error) {
  console.error("Error:", error);
});

});

context.lineWidth = 5;
context.strokeStyle = "rgba(0,0,255,0.3)";
context.lineCap = "square";
context.moveTo(segment.Left, segment.Top);
context.lineTo(segment.Right, segment.Top);
context.moveTo(segment.Left, segment.Bottom);
context.lineTo(segment.Right, segment.Bottom);
context.moveTo(segment.Left, segment.Top);
context.lineTo(segment.Left, segment.Bottom);
context.moveTo(segment.Right, segment.Top);
context.lineTo(segment.Right, segment.Bottom);
context.stroke();
}

function show_image(src, width, height) {
    // Проверяем, существует ли предыдущий элемент картинки
    var previousImage = document.getElementById("dynamic-image");
    if (previousImage) {
        previousImage.remove(); // Удаляем предыдущий элемент картинки
```

```
    }

    var img = document.createElement("img");
    img.id = "dynamic-image"; // Устанавливаем уникальный идентификатор для нового элемента картинки
    img.src = src;
    img.width = width;
    img.height = height;

    // Добавляем новую картинку в <body> тег
    document.body.appendChild(img);
}
function vectorSumY (data) {
var vector = [];
for (var y = 0; y < data.height; y++) {
var sum = 0;
for (var x = 0; x < data.width; x++) {
sum = sum + data.array[y][x];
}
vector.push(sum);
}
return vector;
}
function vectorSumX (data) {
var vector = [];
for (var x = 0; x < data.width; x++) {
var sum = 0;
for (var y = 0; y < data.height; y++) {
sum = sum + data.array[y][x];
}
vector.push(sum);
}
return vector;
}

function Read(callback) {
  fetch("weights2.txt")
    .then(function(response) {
      if (response.ok) {
        return response.text();
      } else {
        throw new Error("Error: " + response.status);
      }
    })
    .then(function(content) {
      var lines = content.split("\n");

      var inWeight = [];
      var hidWeight = [];
      var hidBias = [];
      var outBias = [];

      for (var i = 0; i < lines.length; i++) {
        var values = lines[i].split(";");

        if (i < 1024) {
          var inputRow = [];
          for (var j = 0; j < values.length; j++) {
            inputRow.push(parseFloat(values[j].replace(",", ".")));
          }
          inWeight.push(inputRow);
        } else if (i >= 1024 && i < 1216) {
          var hiddenRow = [];
```

```
          for (var j = 0; j < values.length; j++) {
            hiddenRow.push(parseFloat(values[j].replace(",", ".")));
          }
          hidWeight.push(hiddenRow);
        } else if (i === 1216) {
          for (var j = 0; j < values.length; j++) {
            hidBias.push(parseFloat(values[j].replace(",", ".")));
          }
        } else if (i === 1217) {
          for (var j = 0; j < values.length; j++) {
            outBias.push(parseFloat(values[j].replace(",", ".")));
          }
        }
      }

      var weights = {
        inWeight: inWeight,
        hidWeight: hidWeight,
        hidBias: hidBias,
        outBias: outBias
      };

      callback(weights);
    })
    .catch(function(error) {
      console.error("Error:", error);
    });
}

function Scale(globalMatrix) {
  var originalRows = globalMatrix.length;
  var originalCols = globalMatrix[0].length;

  var targetRows = 32;
  var targetCols = 32;

  var scaledMatrix = [];

  for (var i = 0; i < targetRows; i++) {
    var row = [];
    for (var j = 0; j < targetCols; j++) {
      var originalRow = Math.floor(i * originalRows / targetRows);
      var originalCol = Math.floor(j * originalCols / targetCols);

      var value = 0;
      if (originalRow >= 0 && originalRow < originalRows && originalCol >= 0 && originalCol < originalCols) {
        value = globalMatrix[originalRow][originalCol];
      }

      row.push(value);
    }
    scaledMatrix.push(row);
  }

  return scaledMatrix;
}

function MatrixToArray(matrix) {
  var width = matrix.length;
  var height = matrix[0].length;
  var normalized = new Array(width * height);
```

```
  for (var i = 0; i < width; i++) {
    for (var j = 0; j < height; j++) {
      normalized[i * 32 + j] = matrix[i][j];
    }
  }

  return normalized;
}

function Run(input, weights) {
  var inWeight = weights.inWeight;
  var hidWeight = weights.hidWeight;
  var hidBias = weights.hidBias;
  var outBias = weights.outBias;

  var hidOutput = new Array(192);
  var output = new Array(9);

  // Вычисление выходов скрытого слоя
  for (var i = 0; i < hidOutput.length; i++) {
    var sum = 0;
    for (var j = 0; j < input.length; j++) {
      sum += input[j] * inWeight[j][i];
    }
    hidOutput[i] = Sigmoid(sum + hidBias[i]);
  }

  // Вычисление выходов сети
  for (var i = 0; i < output.length; i++) {
    var sum = 0;
    for (var j = 0; j < hidOutput.length; j++) {
      sum += hidOutput[j] * hidWeight[j][i];
    }
    output[i] = Sigmoid(sum + outBias[i]);
  }

  return output;
}

function Sigmoid(x) {
  return 1 / (1 + Math.exp(-x));
}

function Predict(image, weights) {
  var input = MatrixToArray(image);
  var output = Run(input, weights);
  var maxOutput = 0;
  var predictedIndex = 0;

  for (var i = 0; i < output.length; i++) {
    if (output[i] > maxOutput) {
      maxOutput = output[i];
      predictedIndex = i;
    }
  }

  return predictedIndex;
}
```

Приложение Г

ЭКРАННЫЕ ФОРМЫ

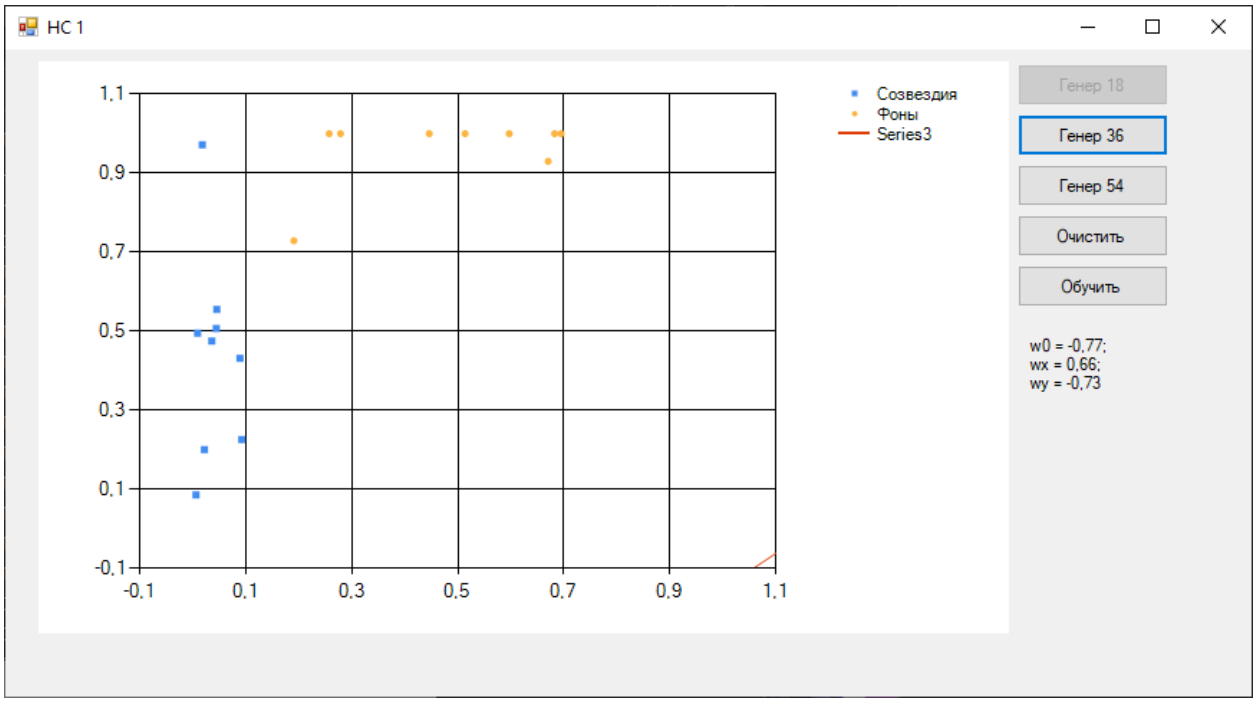


Рисунок Г.1 – Выборка из 18 образов

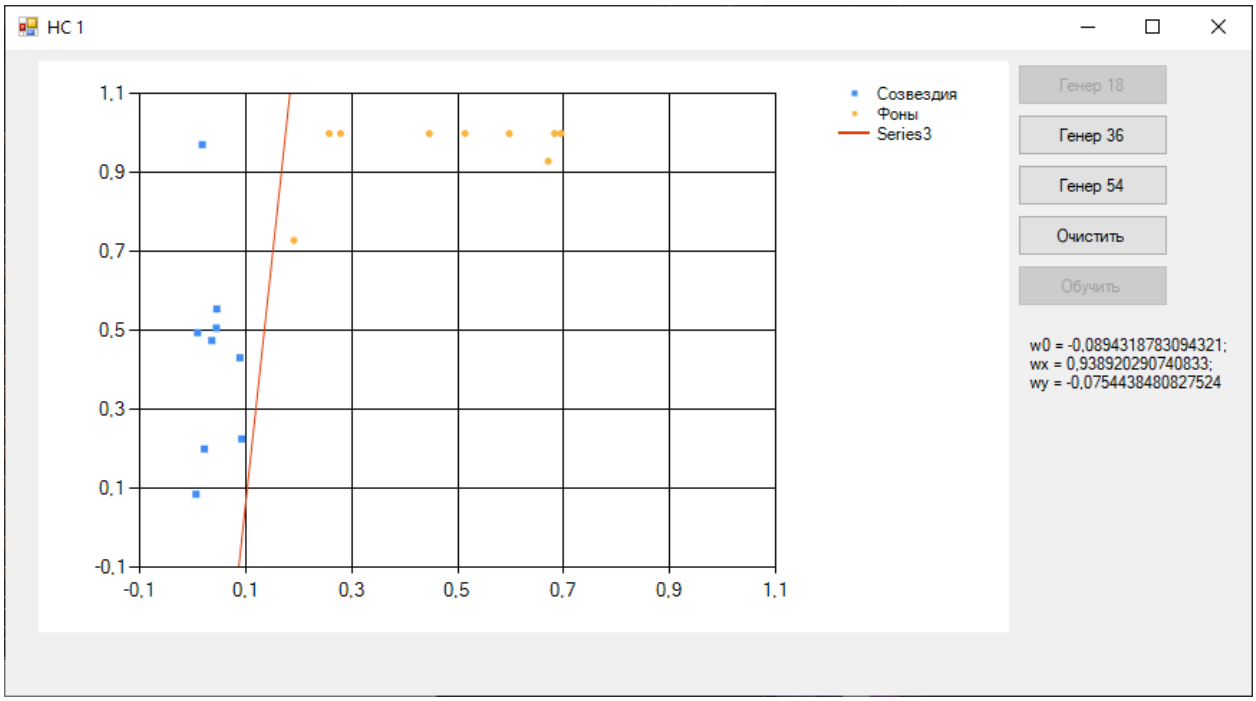


Рисунок Г.2 – Пример успешного обучения

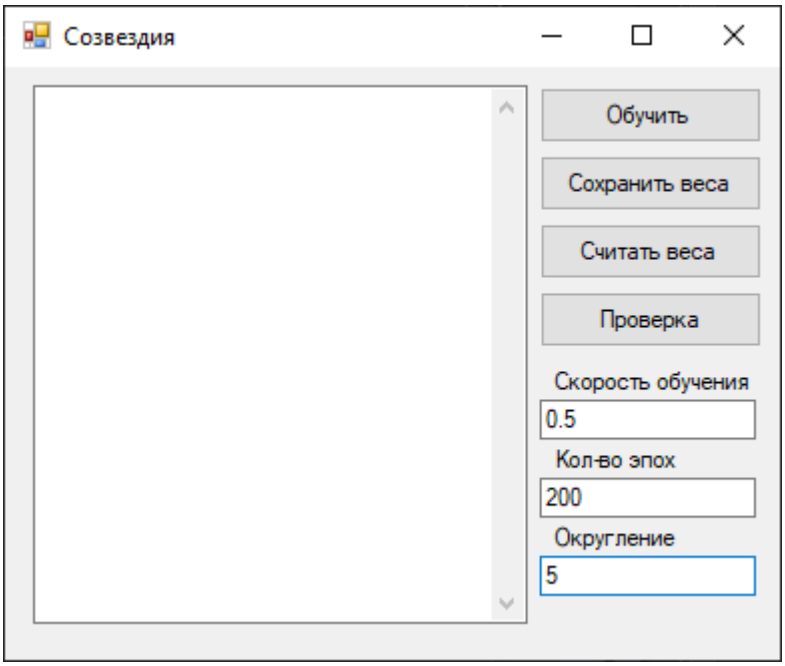


Рисунок Г.3 – Окно программы для обучения нейросети

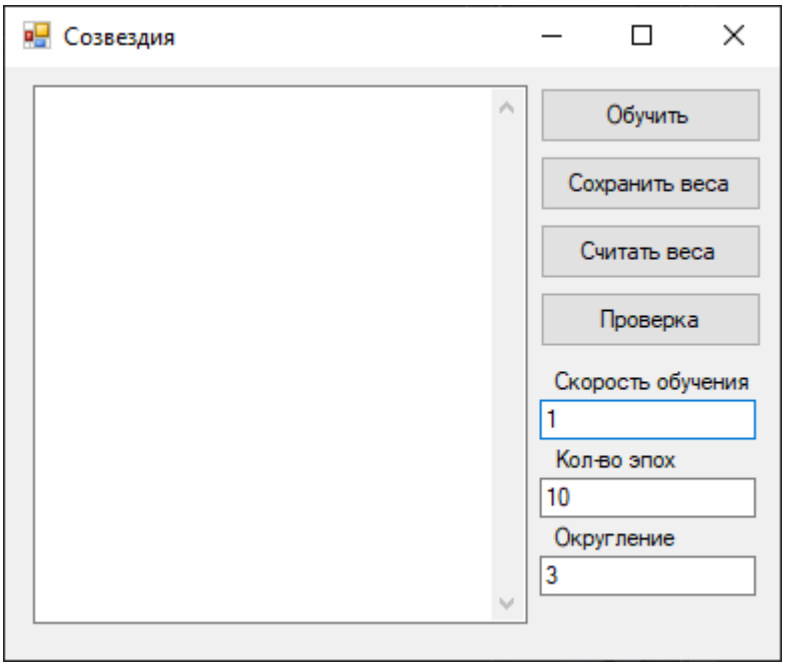


Рисунок Г.4 – Подобранные данные

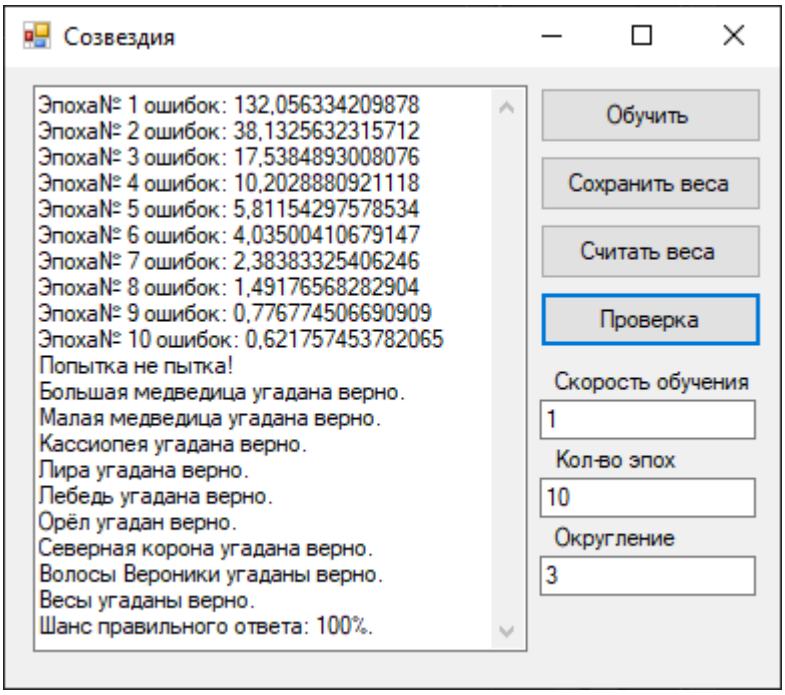


Рисунок Г.5 – Успешно обученная нейросеть

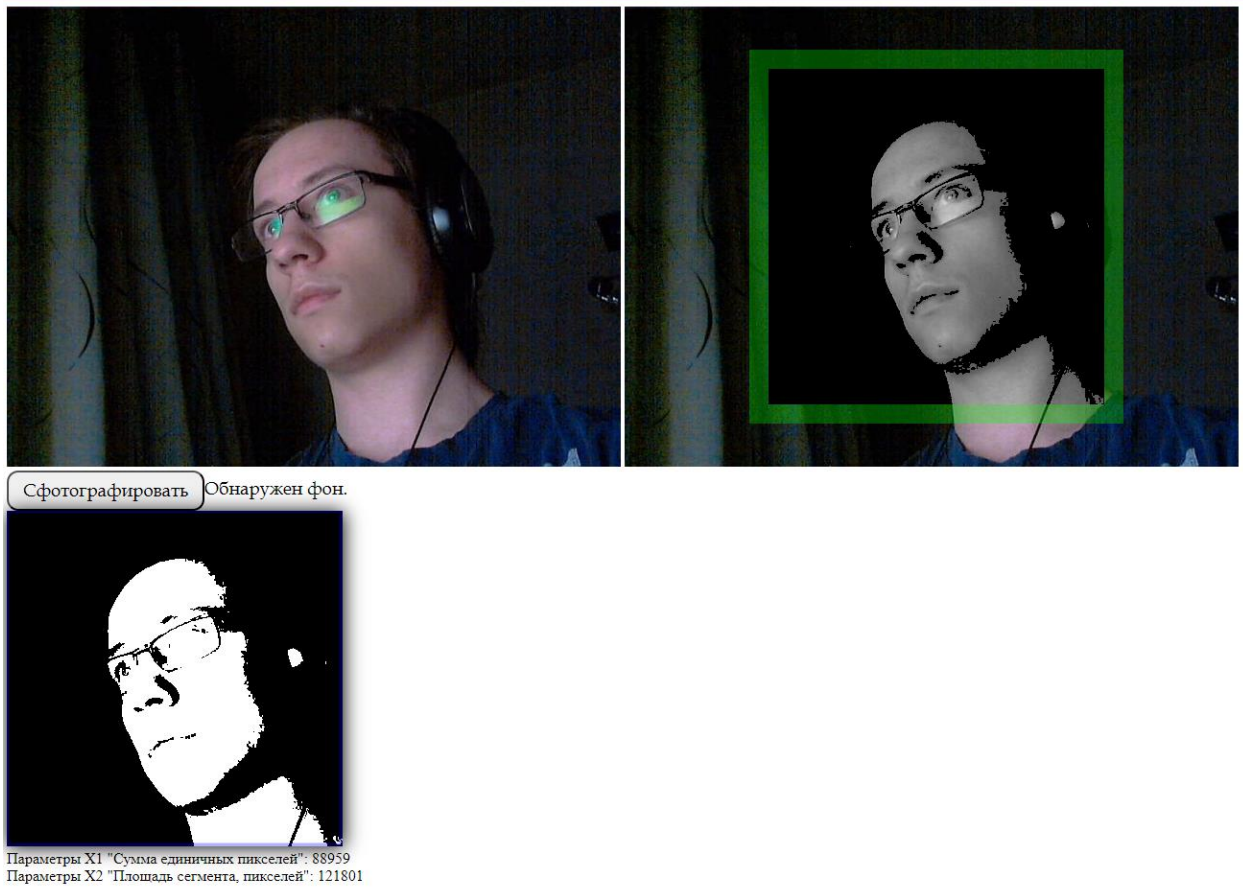


Рисунок Г.6 – Обнаружен фон.

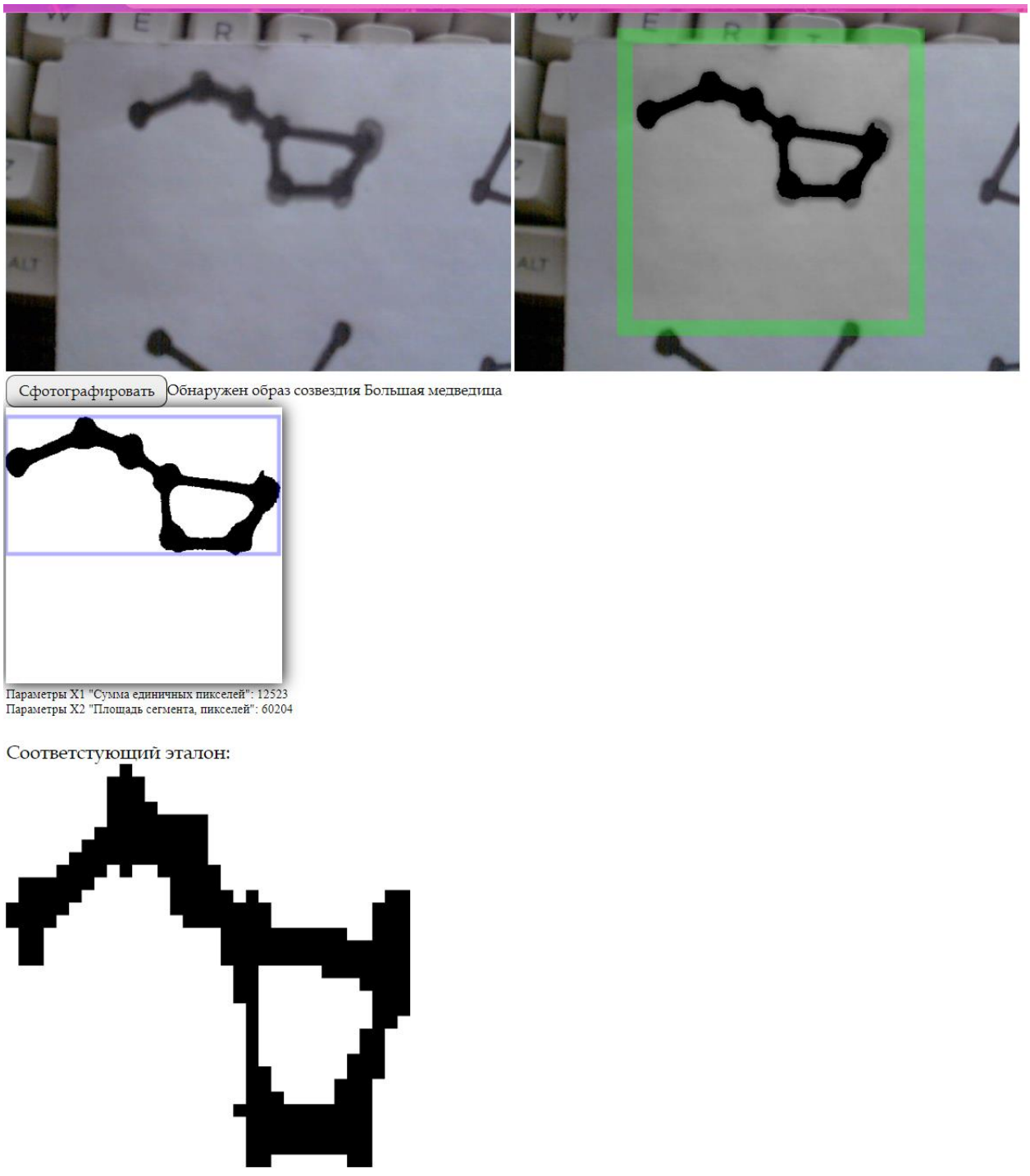
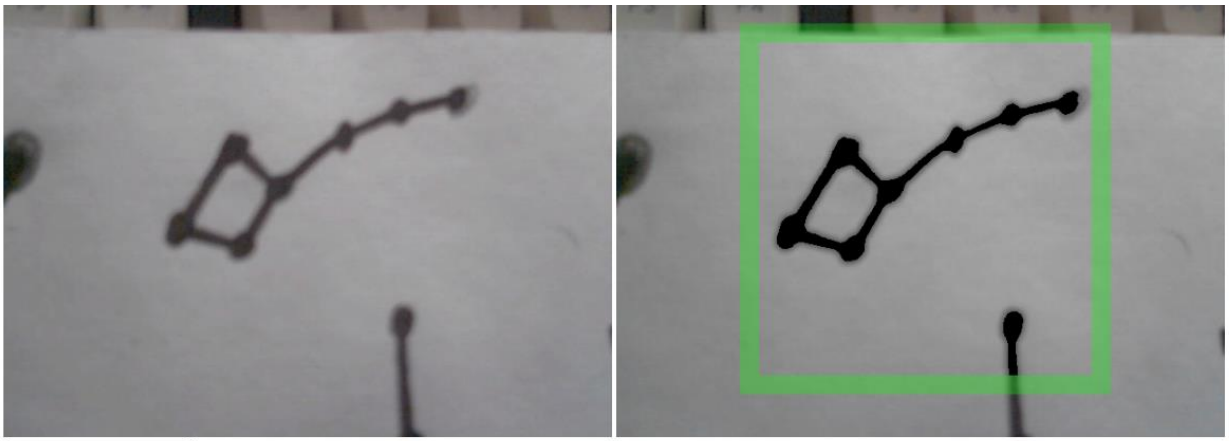
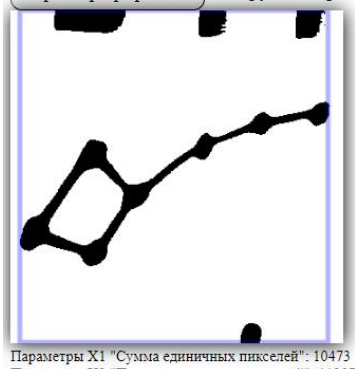


Рисунок Г.7 – Обнаружен образ созвездия Большая медведица



Сфотографировать Обнаружен образ созвездия Малая медведица



Параметры X1 "Сумма единичных пикселей": 10473
Параметры X2 "Площадь сегмента, пикселей": 113076

Соответствующий эталон:

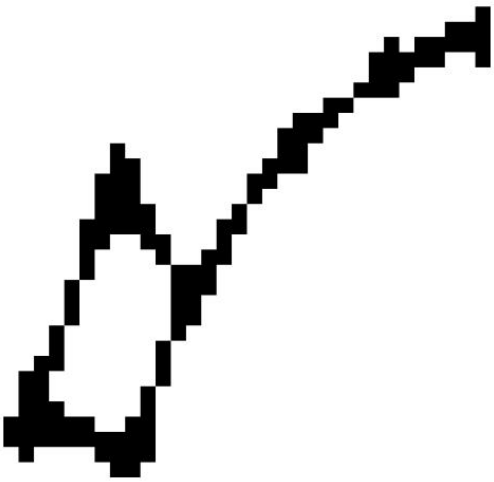


Рисунок Г.8 – Обнаружен образ созвездия Малая медведица

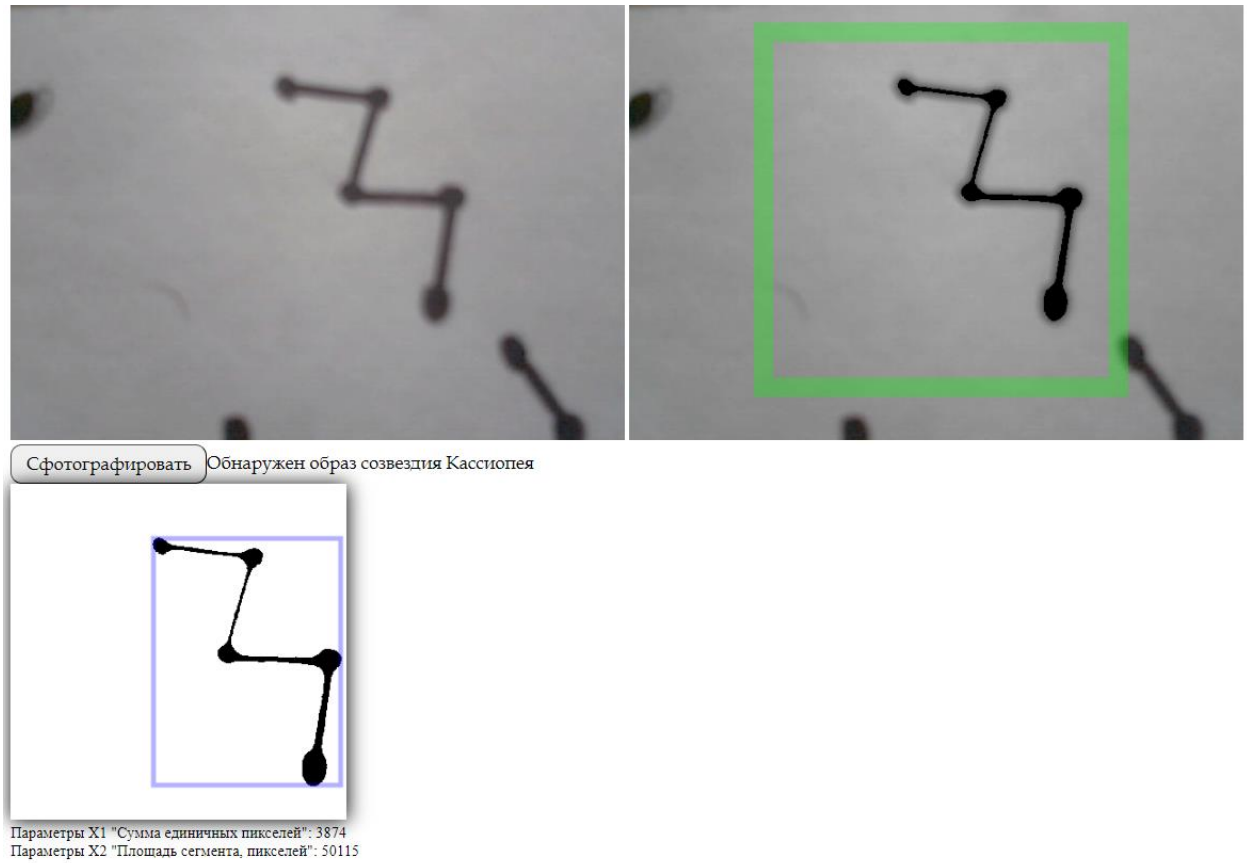


Рисунок Г.9 – Обнаружен образ созвездия Кассиопея

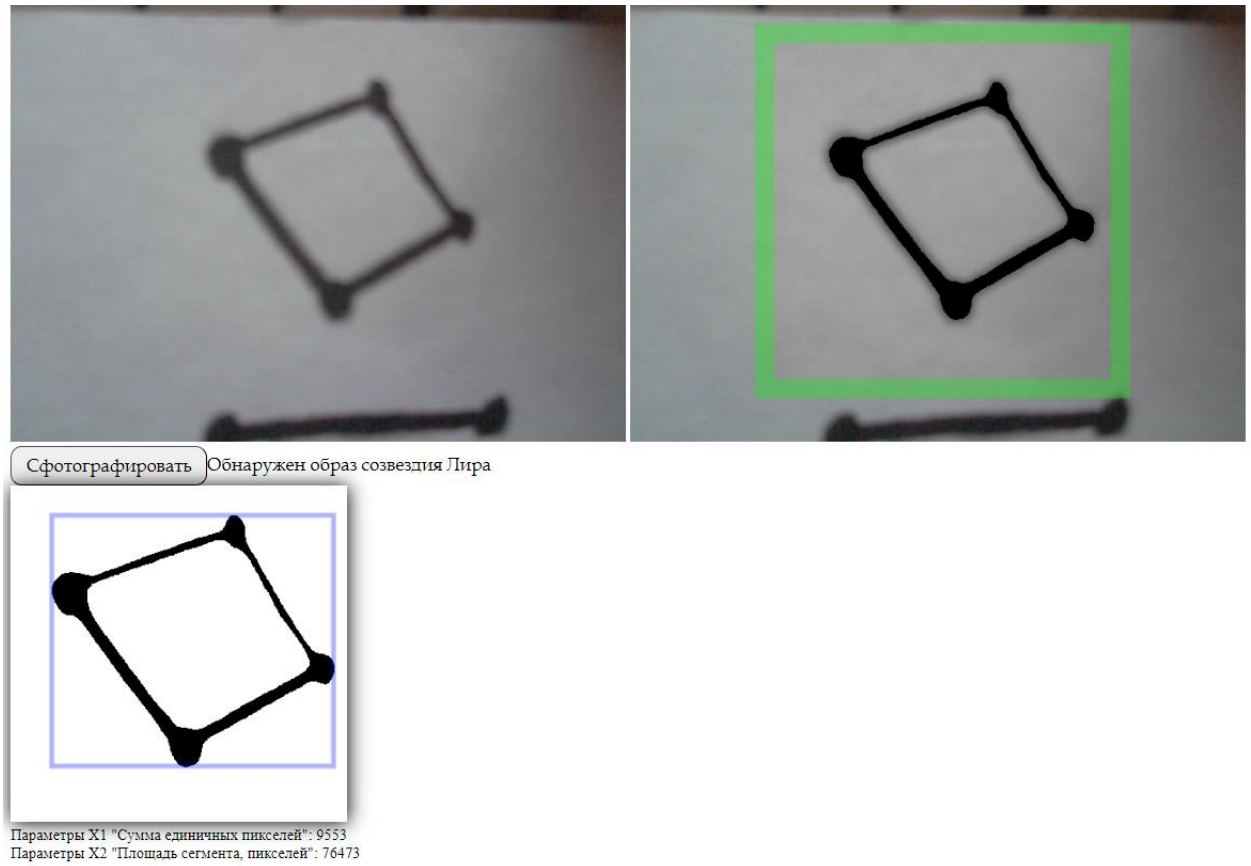


Рисунок Г.10 – Обнаружен образ созвездия Лира

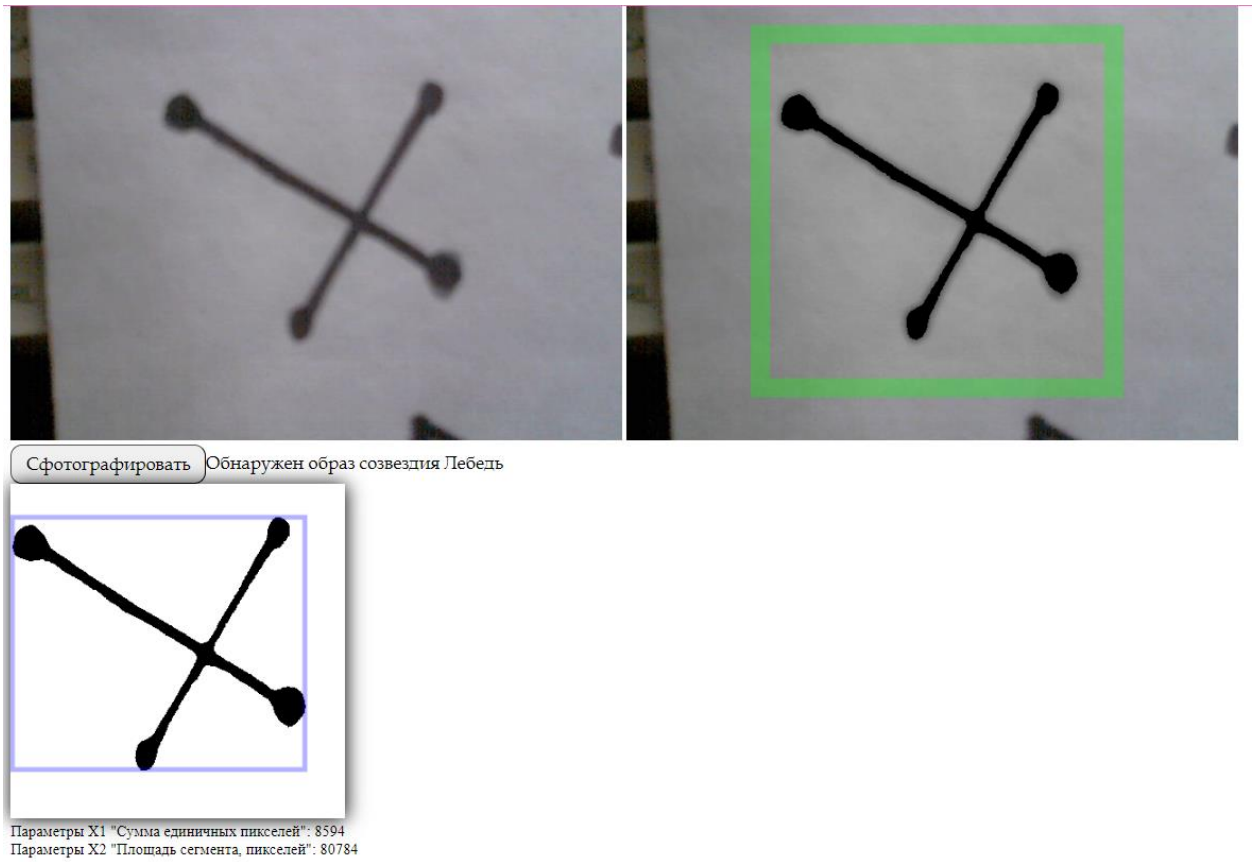


Рисунок Г.11 – Обнаружен образ созвездия Лебедь

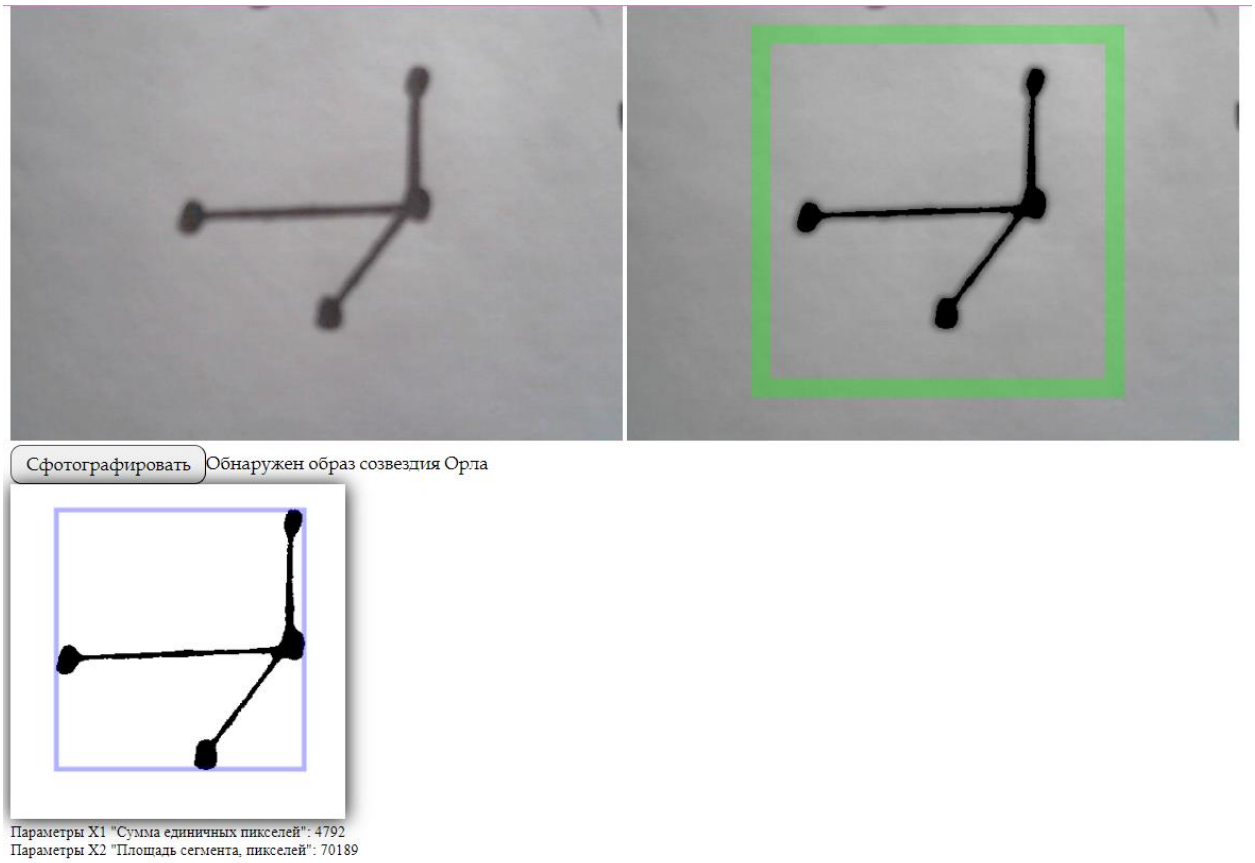


Рисунок Г.12 – Обнаружен образ созвездия Орла

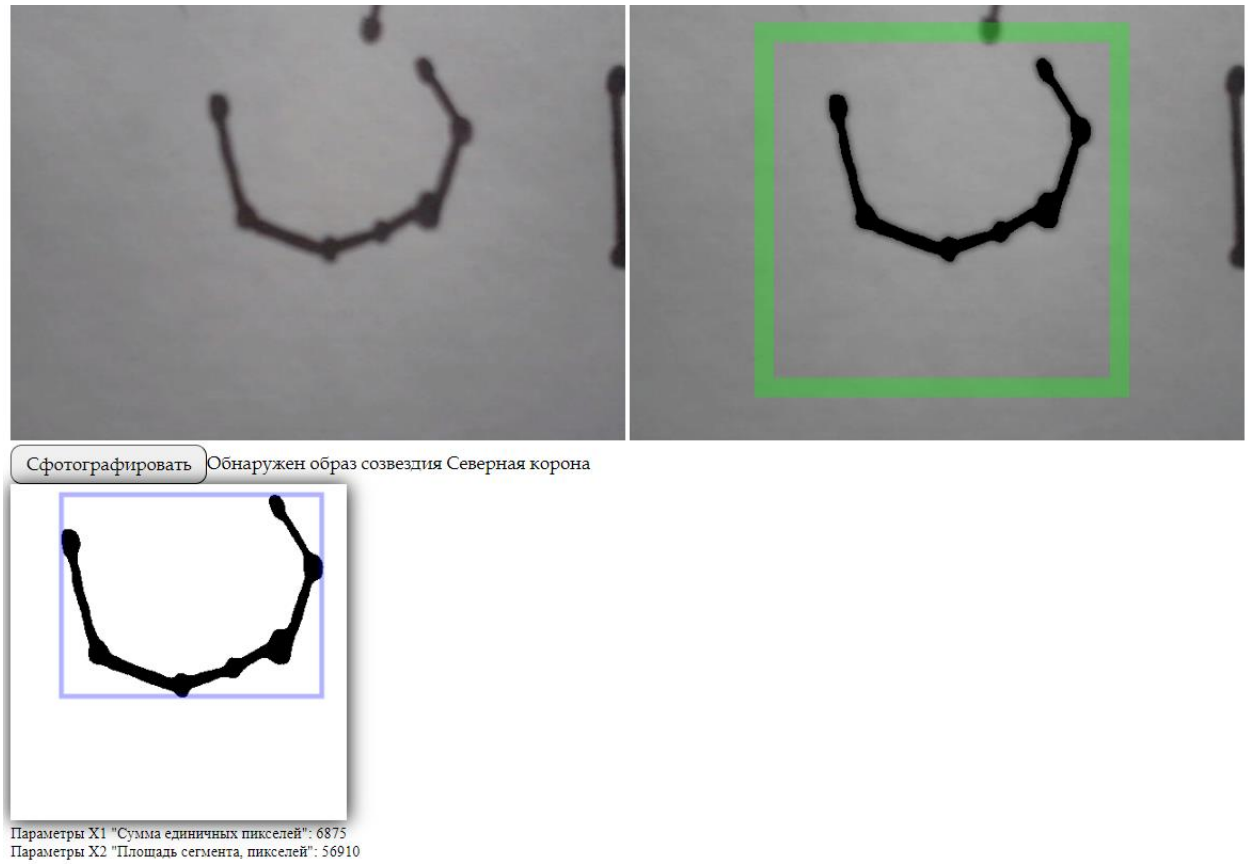


Рисунок Г.13 – Обнаружен образ созвездия Северная корона

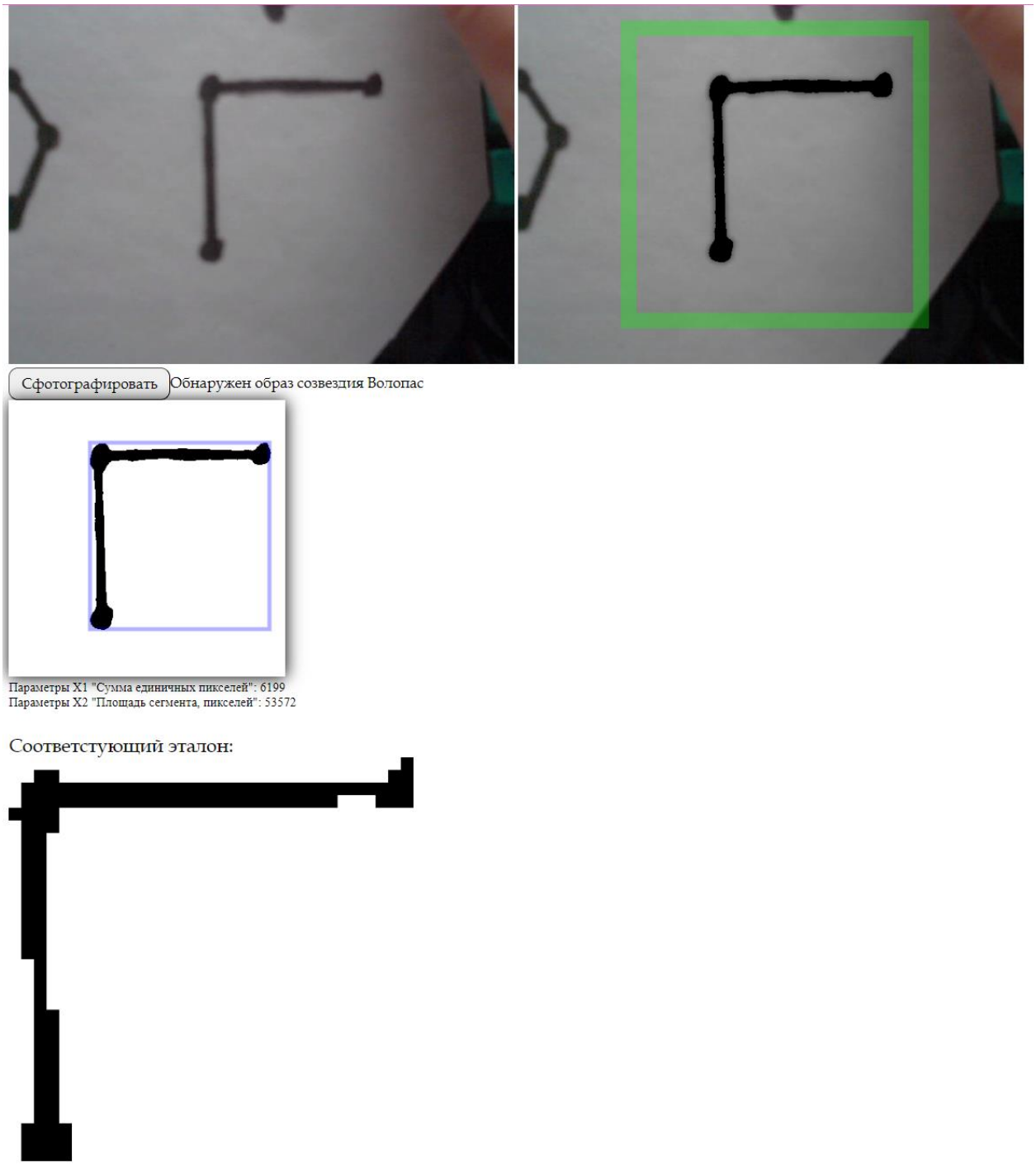


Рисунок Г.14 – Обнаружен образ созвездия Волопас

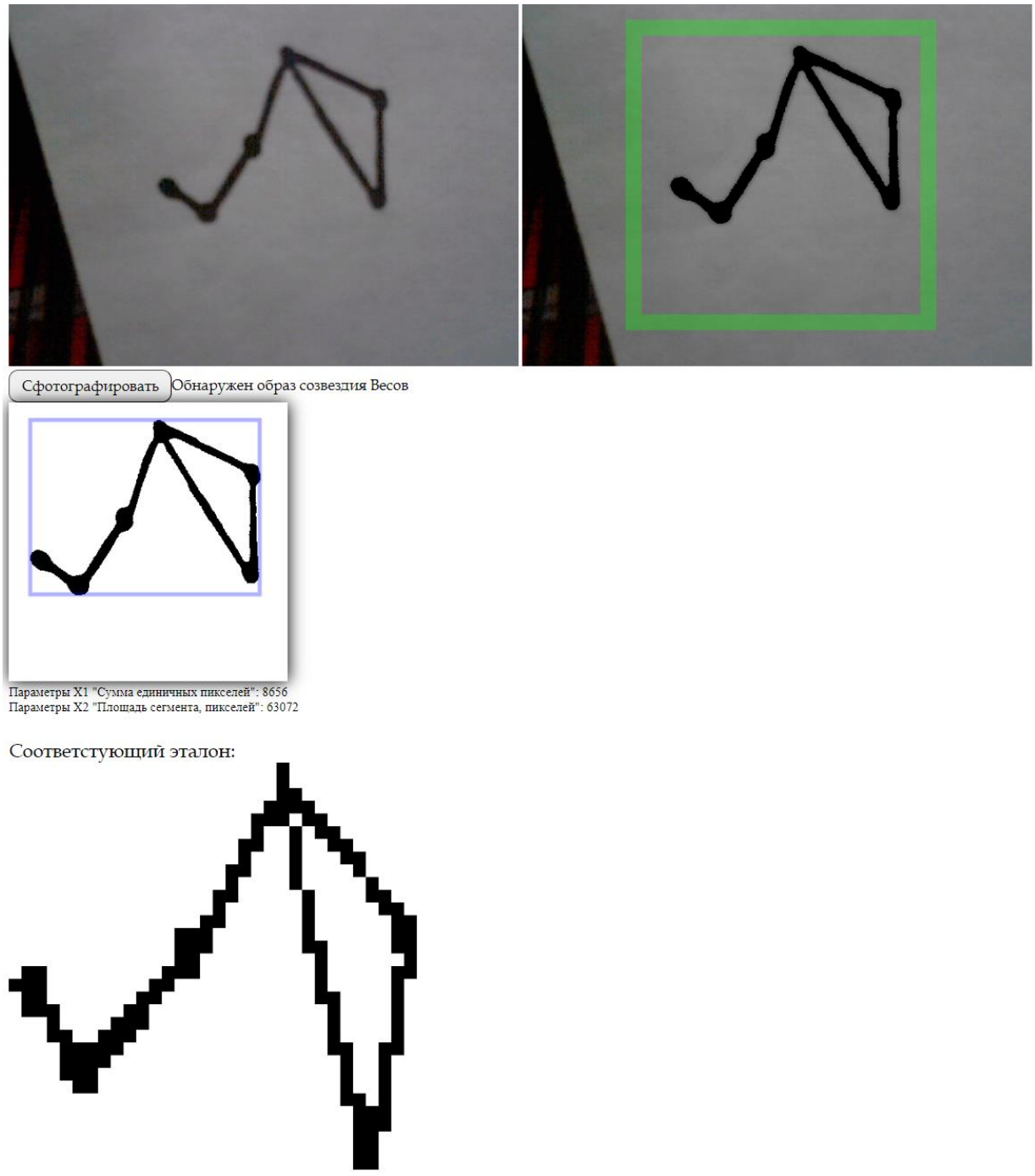


Рисунок Г.15 – Обнаружен образ созвездия Весов