МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ОМСКИЙ ГОСУДАРСТВЕННОЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Прикладная математика и фундаментальная информатика»

Реферат123

Содержание

**Введение**

Проблема восстановления цифровых изображений возникает в следствие многих причин – неисправность носителя, ошибки при передаче файла по сети, искажение из-за ошибок операционной системы и так далее. Визуально это может проявляться как появление темных областей на изображении, появление зашумленности, отсутствие части изображения. Для избавления изображения от повреждений используются два основных подхода. Первый подход – это использование сглаживающих фильтров, второй – поиск поврежденных пикселей в изображении и их восстановление. Использование фильтров дает хороший визуальный результат, но приводит к искажению изображения. Восстановление изображения по пикселям является более затратным по времени способом, однако потенциал этого метода является большим, так как затрагиваются только поврежденные части изображения, а следовательно и искажения изображения будут меньшими в сравнении с первым подходом.

Существует множество алгоритмов для восстановления поврежденных пикселей, однако практически все они основаны на анализе неповрежденных пикселей, расположенных по соседству с поврежденным. Поиск цвета для поврежденного пикселя происходит с помощью разных методов, например, с помощью арифметического среднего, или с помощью метода иерархий [1].

Однако при большой зашумленности изображений, либо при отсутствии крупной его части (более 30%) результат данных алгоритмов будет иметь визуальные искажения. Большей устойчивостью потенциально обладает метод восстановления пикселей с использованием нейронных сетей.

Нейронные сети – математические модели, а так же их программные или аппаратные реализации, построенные по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма[2].

Основные понятия и определения

Самоорганизующиеся карты Кохонена

Алгоритм восстановления изображений(я пока что не решил,выносить ли это в отдельный раздел или же включить в раздел с картами кохонена)

**2 Программа**

**Основные возможности**

В разрабатываемом приложении должны быть реализованы следующие основные функции:

1. приведение изображения к внутреннему формату программы;
2. обучение нейронной сети;
3. возможность сохранения состояния сети;
4. загрузка обученной сети;
5. обозначение поврежденной области;
6. восстановление изображения;
7. сохранение восстановленного изображения;
8. сбор различных параметров работы для проведения анализа результата.

Для реализации приложения был выбран язык программирования С# и среда разработки Microsoft Visual Studio 2012.

**Внутренний формат изображения**

Изображение по сути является матрицей, состоящей из пикселей, но для обучения сети нельзя подавать целое изображение, поэтому было решено составлять вектор входных значений из набора пикселей, находящихся вокруг какой-либо точки, выбранной заранее. Выбор этой точки можно производить разными способами – либо делать это случайным образом, либо путем наложения некоторой сетки на изображение с ячейками, размерность которых равна размерности вектора веса нейрона. В обоих случаях необходимо выполнять проверку на наличие поврежденных пикселей в наборе, и если такие пиксели имеются, то данный набор не подходит для обучения.

**Обозначение поврежденной области**

Поврежденная область цифрового изображения может быть различного характера, либо это локализованная область, внутри которой находятся поврежденные пиксели, либо изображение полностью зашумлено, то есть искажено полностью. Для выделения области в первом случае было решено добавить в интерфейс поля, в которые будут водиться координаты четырех точек, ограничивающих поврежденную область, то есть сама область будет иметь прямоугольную форму, это сделано для упрощения реализации и чтобы избежать фактора ошибки конечного пользователя, который будет выделять эту область. В любом случае перед началом обучения поврежденная область будет проверяться внутри программы для выделения точных границ. Если изображение зашумлено полностью, то перед началом работы необходимо будет оставить поля, описанные выше незаполненными.

**Сохранение состояния обученной сети**

Поскольку разрабатываемое приложение предназначено для многоразового использования, следует сохранять состояние обученной карты, а так же обеспечить пользователю возможность загрузки ранее сохраненного состояния карты. Для этих целей можно использовать два основных варианта – это сохранение состояния в базу данных, либо сохранение состояния в файл. Первый способ требует проектирования базы данных, создание таблиц и связей между ними, а так же реализации функций внутри программы для работы с это базой данных. Данный способ не является целесообразным, так как объем информации, которую необходимо сохранить, является небольшим. Более простым является второй способ. В ходе работы было решено использовать файлы в формате XML для сохранения состояния карты.

XML – рекомендованный Консорциумом Всемирной паутины (W3C) язык разметки. Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому). XML разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов программами и одновременно удобный для чтения и создания документов человеком, с подчёркиванием нацеленности на использование в Интернете[3].

В языке С# существует специальный объект для работы с XML-файлами. Этот объект находится в пространстве имен «System.Xml.Schema» и представлен двумя основными классами – «XmlSchemaElement» и «XElement». Первый служит для создания схемы, второй – непосредственно для создания элементов файла. В рамках концепции XML каждый нейрон будет сохранен в файл как отдельный элемент, и его параметры, такие как вектор веса и координаты будут сохранены как свойства этого элемента. В приложении Х показано итоговое содержание созданного программой XML-документа.

**Результаты работы программы**

Восстановленное изображение будет являться результатом работы программы и будет сохраняться как отдельный файл в указанную пользователем директорию. Так же, помимо изображения, для оценки результатов следует сохранить все входные данные, такие как:

1. размерность вектора веса нейрона;
2. количество итераций обучения сети;
3. количество нейронов в карте;
4. количество поврежденных пикселей в процентах;
5. время, затраченное на обучение сети;
6. время, затраченное на восстановление изображения;
7. информация об исходном изображении (высота и ширина в пикселях).

Сохранение всех вышеперечисленных параметров позволит производить калибровку программы с целью повышения эффективности. Данные будут сохраняться отдельным файлом в формате XML.

**Интерфейс приложения**

Учитывая все возможности и функции проектируемого приложения, описанные выше, был спроектирован интерфейс приложения. Для простоты реализации было решено использовать одно окно для интерфейса, поскольку основных функциональных блоков интерфейса немного.

Условно главное окно можно разделить на несколько блоков:

1. блок настроек нейронной сети;
2. блок выделения поврежденной области;
3. блок добавления поврежденного изображения;
4. блок просмотра результатов работы.

С помощью онлайн – инструмента moqups.com был разработан макет главного окна приложения, он представлен на рисунке хх.

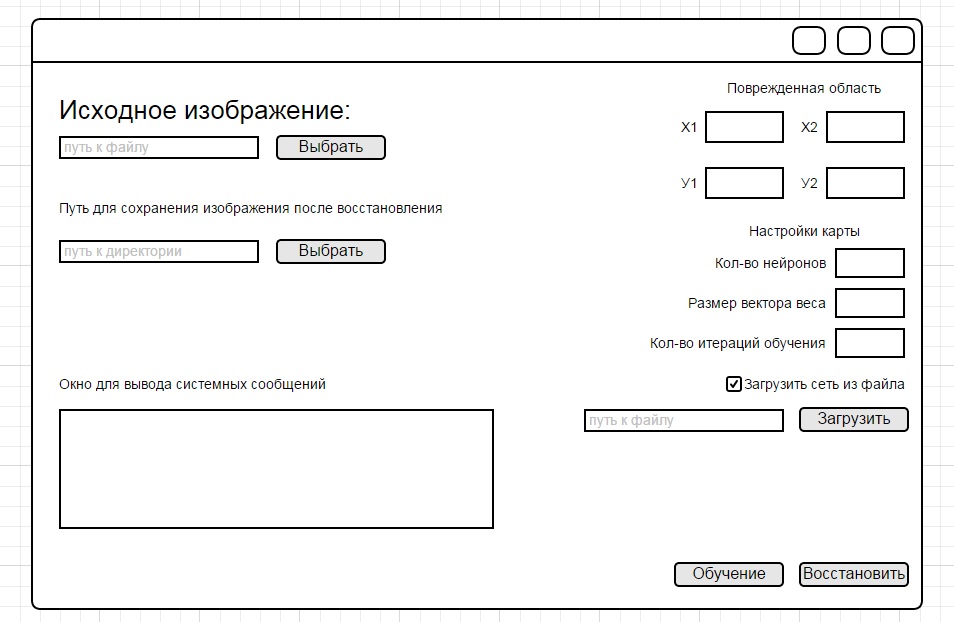


Рисунок хх – Макет главного окна приложения

Для создания главного окна приложения по макету, Microsoft Visual Studio 2012 обладает встроенным инструментом, который называется «Конструктор форм». Этот инструмент позволяет создавать окна приложения и добавлять на него нужные элементы. Чтобы окно соответствовало макету необходимо использовать следующие элементы управления:

1. Label;
2. TextBox;
3. RichTextBox;
4. Button;
5. CheckBox.

Каждый из этих элементов является классом, обладающий своими свойствами, событиями и методами.

Окно для вывода системных сообщений будет служить для отладки приложения, в конечной версии приложения оно будет выполнять другие функции. На этапе отладки в нем будут выводиться сообщения об ошибках и предупреждения по мере их возникновения. Далее, после окончания процесса отладки, в нем будут появляться сообщения об окончании обучения сети, сообщения о завершении восстановления изображения.

Возможна ситуация, когда пользователю нужно восстановить изображение по уже обученной сети, либо если пользователю нужно подать другую картинку для обучения уже ранее инициализированной сети. Для этих целей в макете предусмотрен checkbox – элемент, который выполняет роль следующей условной конструкции – если возвращаемое значение этого элемента перед началом процесса обучения сети эквивалентно логическому значению «true», то вместо параметров, вводимых в блоке «Настройки карты», будут использоваться параметры, полученные из XML файла, путь к которому должен быть указан пользователем заранее. В ходе написания кода приложения было разработан интерфейс, представленный на рисунке хх.

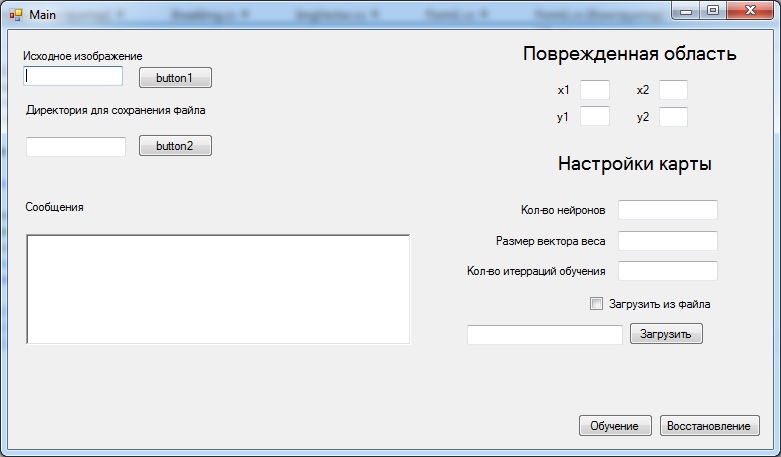


Рисунок хх – Основное окно приложения

**Классы С#, необходимые для реализации приложения.**

Язык C# обладает встроенным набором классов для работы с изображениями. Основной класс, который будет использоваться для работы с изображениями – это класс Bitmap. Он позволяет инкапсулировать точечный рисунок GDI+, состоящий из пикселей графического изображения и атрибутов рисунка[4]. Другими словами, он позволяет манипулировать изображением или получать какие-либо его свойства. Например, можно получить высоту и ширину изображения в пикселях, получить горизонтальное и вертикальное разрешение изображения, манипулировать палитрой цветов, манипулировать пикселями изображения (менять цвет пикселя, получить его координаты и так далее), а так же получить дополнительную информацию об изображении (формат исходной картинки, название, заголовки и так далее). Потребуются следующие свойства класса Bitmap:

1. height;
2. width.

Первое свойство получает высоту изображения в пикселях, второе – ширину. Эти свойства понадобятся при создании экземпляров классов, которые будут написаны для работы с векторами входных данных, а так же при сохранении восстановленного изображения.

Из всей совокупности методов рассматриваемого класса следует выделить основные – это методы SetPixel(), GetPixel(), Save(). Первый метод задает цвет определенного пикселя на изображении, входные параметры этого метода – это координаты нужного пикселя, а так же цвет, который будет записан в него в виде объекта класса Color. Второй метод возвращает цвет пикселя в виде объекта класса Color по заданным координатам, которые передаются в метод в качестве параметров. Последний метод сохраняет объект Bitmap как цифровой графический файл в любом расширении, допустимом в языке C#. Входные параметры этого метода – это название файла и путь к директории, в которую необходимо сохранить файл.

Так же в приложении будет использован класс Color, позволяющий работать с цветами пикселей. Он включает в себя возможность преобразования цветов из различных форматов (например, из формата «rgb») в свой внутренний формат, который используется классом Bitmap для работы с цветами пикселей.

**Модули программы**

В соответствии с описанными ранее возможностями программа была разбита на следующие отдельные составляющие:

1. приведение исходного изображения к векторному виду для обучения сети;
2. обучение нейронной сети;
3. восстановление изображения и сохранение его в файл;
4. сохранение состояния нейронной сети;
5. загрузка нейронной сети из файла;
6. функции для создания XML – разметки.

Результаты и их анализ и выводы

Заключение

Список использованных источников

Приложения(коды,листинги,изображения)