3. ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Рассмотрим подробно функционирование программы. Для этого проведем анализ основных блоков программы и рассмотрим их зависимости. А также проанализируем все функциональные компоненты, которые входят в состав кода программы, и рассмотрим назначение всех методов и переменных классов этих блоков.

В разрабатываемом приложении можно выделить следующие блоки:

– блок пользовательского интерфейса;
– блок разделения видеозаписей на кадры;
– блок обработки изображения;
– блок детектирования;
– блок стандартизации изображений;
– блок классификации;
– блок приведения полученных данных к одному формату;
– блок экспортирования результатов в базу данных.

Изначально пользователь попадает на главный экран, логика которого находится в классе `MainForm`. Здесь расположен весь основной графический интерфейс. В данном классе находятся все элементы, через которые пользователь взаимодействует с приложением. Здесь отображаются все информационные сообщения и выводятся управляющие элементы. Пользователь управляет программой в основном через нажатие на клавиши, расположенные в окне приложения. При нажатии происходит вызов необходимых функций, происходят определенные операции и пользователь получает какое-либо уведомление о завершении произошедшей операции.

На главном экране пользователь может указать на папку, в которой находятся видеозаписи, на которых необходимо произвести распознавание. После этого производится выбор места расположения для полученных изображений, которые будут сохранены во вложенные папки в соответствии с названиями исходных видеозаписей. Также при обработке исходных видеоматериалов есть возможность настройки частоты создания изображений. После этого пользователь выбирает набор необходимых преобразований и фильтров для улучшения качества распознавания. Затем он может выполнить операцию детекции областей изображений, на которых есть дорожные знаки. После этого производится запуск классификатора обнаруженных знаков. Результаты предыдущих этапов обработки приводятся к единому формату для стандартизации и упрощения экспортирования данных. И в конце происходит запись результатов в базу данных с заранее определенными полями. Так же все предыдущие этапы можно выполнить в полуавтоматическом режиме. Для этого вызывается специальное окно, в котором находятся все необходимые пути, настройки и другие возможные опции.

3.1.Классы разрабатываемого программного средства

3.1.1. Класс FFMPEGConverter

Этот класс является функциональной оберткой консольного вызова программы ffmpeg из одноименного набора библиотек для разбиения с определенной частотой видеозаписи на изображения.

Поле `sourcePath` хранит в себе путь к набору видеозаписей для распознавания.

В поле fps хранится частота преобразования видеозаписи, то есть количество кадров за секунду видеозаписи.

Метод `convertVidToImages` создает новый процесс, в котором через консоль Windows вызывает программу ffmpeg с определенным набором аргументов. Через аргументы передается расположение полный путь к видеозаписи и папки для сохранения полученных изображений, а также количество получаемых кадров за секунду.

Аргумент -qscale указывается для того, чтобы использовать фиксированную шкалу качества. Аргумент –r устанавливает частоту кадров. Пример команды вызова программы ffmpeg из консоли ОС Windows приведен на рисунке 3.1.

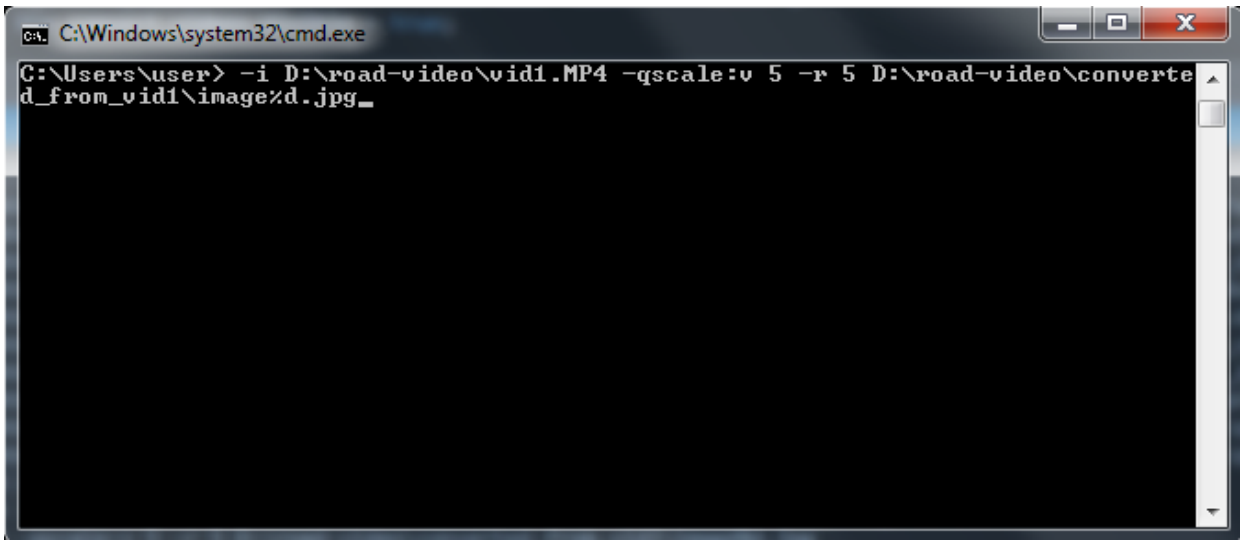

Рисунок 3.1 – Пример команды ffmpeg для раскадрирования видеозаписи

Метод `convertVidToSubs` создает новый процесс, в котором через консоль Windows вызывает программу ffmpeg с определенным набором аргументов. Через аргументы передается расположение видеозаписи, некоторые дополнительные параметры и путь к новому файлу для сохранения полученных изображений.

Аргументы -vn и -an используются, чтобы пропустить включение потоков видео и аудио соответственно, независимо от того, отображены ли они вручную или автоматически, за исключением тех потоков, которые являются выходами сложных фильтров.

Аргумент -map нужен для ручного управления выбором потока в каждом выходном файле.

Пример команды вызова программы ffmpeg из консоли ОС Windows приведен на рисунке 3.2.

Метод `convertAll` вызывает два предыдущих метода для полного преобразования набора видеозаписей в соответствующее число папок, содержащих изображения в формате .jpg и файлы субтитров в формате .txt. Он в зависимости от успешности выполнения включенных методов возвращает либо *`true`*, либо *`false`*.

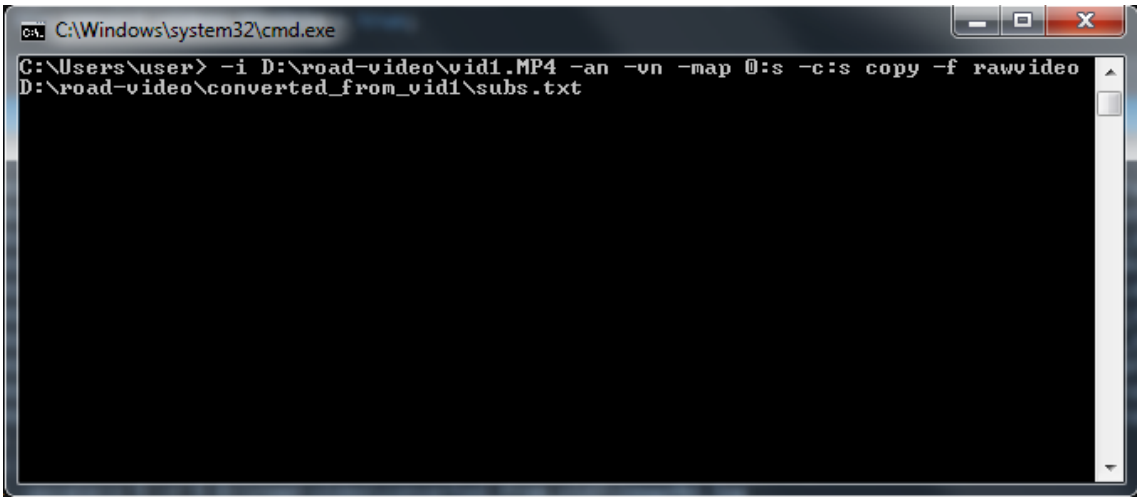

Рисунок 3.2 – Пример команды ffmpeg для получения субтитров из видеозаписи

Метод `ParseSubtitleFile` производит разбиение полученного текстового файла с субтитрами с помощью регулярного выражения. Каждое найденное соответствие приводится к объекту класса `MovementPoint` с помощью метода `ParseMovementPoint`. После обработки получается коллекция координат, которая используется в дальнейшей обработке.

В методе `ParseMovementPoint` происходит создание объекта класса `MovementPoint` путем разбиения текстовой строки по полям объекта класса.

Метод `ConvertDegreesAndDecimalMinutesStringToDecimal` необходим для приведения географических координат к пригодному для чтения десятичному виду.

3.1.2. Класс ImgOps

Данный класс содержит методы для преобразования и фильтрации изображений с целью увеличения количественно-качественных характеристик распознавания дорожных знаков. Он использует методы библиотеки EmguCV.

Используя метод `RGBtoGrey`, происходит преобразование изображения из цветного в градации серого. Это позволяет уменьшить количество цветовых каналов с 3 до 1, что позволяет использовать меньше вычислительных мощностей и уменьшает время распознавания. Он принимает изображение в цветовом пространстве RGB и возвращает изображение в градациях серого цвета.

Метод `RGBtoHSV` преобразует изображение из цветового пространства RGB в цветовое пространство HSV. Он принимает изображение в цветовом пространстве RGB и возвращает изображение в цветовом пространстве HSV.

Для того чтобы получить бинарное изображение используется метод `toBinary`. Он принимает изображение, а также порог бинаризации. На выходе получается изображение, в котором каждый пиксель принимает значение «1» либо «0». Если яркость пикселя исходного изображения меньше порога бинаризации, то в преобразованном изображении его значение будет равно нулю, если же больше – единице.

В методе `InterpolationResize` производится масштабирование изображения с использованием бинарной интерполяции. Данный метод на входе получает исходное изображение и необходимые размеры результата и возвращает полученное изображение.

Суть интерполяции заключается в использовании имеющихся данных для получения ожидаемых значений в неизвестных точках. Интерполяция изображений работает в двух измерениях и пытается достичь наилучшего приближения в цвете и яркости пикселя, основываясь на значениях окружающих пикселей. Бикубическая интерполяция рассматривает массив из 4x4 окружающих пикселей – всего 16. Поскольку они находятся на разных расстояниях от неизвестного пикселя, ближайшие пиксели получают при расчете больший вес. Бикубическая интерполяция производит значительно более резкие изображения, чем другие методы, и возможно, является оптимальной по соотношению времени обработки и качества на выходе. Результат увеличения с интерполяцией можно увидеть на рисунке 3.2(а).

а – исходное изображение; б – после увеличения с интерполяцией
Рисунок 3.2 Результат увеличения с интерполяцией

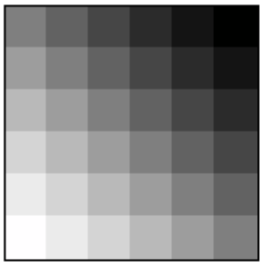

Рисунок 3.3 Результат увеличения без интерполяции

Для улучшения различимости элементов классифицируемого фрагмента изображения, содержащего дорожный знак необходимо применить метод `ContrastAlignment`. Метод принимает исходное изображение и возвращает масштабированное изображение. В данном методе используется контрастное выравнивание CLAHE (Contrast-limited adaptive histogram equalization).

Контрастное выравнивание CLAHE используется для изображений, имеющих неоднородное геометрическое распределения яркостей. Оно анализирует небольшие участки изображения и позволяет усилить локальный контраст. Для каждого пикселя рассматривается небольшая окрестность изображения, по которой строится функция преобразования, при этом все изображение, как таковое, не используется. Оно позволяет уменьшить неоднородность освещения дорожных знаков. Результат контрастного выравнивания можно увидеть на рисунке 3.4.

Рисунок 3.4 Результат применения контрастного выравнивания CLAHE

В методе `Filter` происходит фильтрация изображения, после которой на готовом изображении будут находиться только те области изображения, цвет которых находится в заданном HSV интервале. Метод принимает нижний и верхний порог фильтрации и возвращает полученное изображение.

3.1.3. Класс MovementPoint

В классе `MovementPoint` содержится сущность описывающая точку координат, пригодном для создания объектов при экспортировании полученных данных.

Поле `Lat` отображает географическую широту.

Поле `Lon` отображает географическую долготу.

Поле `Azimuth` содержит азимут в интервале от 0 до 360.

Поле `Date` позволяет узнать, когда были получены данные о координатах той либо иной точки.

3.1.4. Класс SignsHaarCascade

Этот класс выполняет распознавание знаков на основе выбранных каскадов Хаара. Класс содержит экземпляр класса `CascadeClassifier` из библиотеки EmguCV.

При создании объекта этого класса выбирается существующий на данном компьютере заранее обученный каскад в файле формата XML, который будет использоваться для детекции знаков в базе фотографий.

Метод `detectAll` принимает изображение, на котором необходимо найти знаки. Здесь вызывается функция `DetectMultiScale` класса `CascadeClassifier` из библиотеки EmguCV, который находит прямоугольные области в изображении, которые, вероятно, содержат объекты, для которых обучен каскад, и возвращает эти области в виде последовательности прямоугольников. Функция сканирует изображение несколько раз и в разных масштабах. Каждый раз он учитывает перекрывающиеся области на изображении. Также может быть применена некоторая эвристика для уменьшения количества анализируемых областей, например, алгоритм Кэнни.

Детектор границ Кэнни – оператор обнаружения границ изображения. Границы здесь отмечаются там, где градиент изображения приобретает максимальное значение. Они могут иметь различное направление, поэтому алгоритм Кэнни использует четыре фильтра для обнаружения горизонтальных, вертикальных и диагональных ребер в предварительно размытом для удаления шумов изображении. Результат работы данного детектора можно увидеть на рисунке 3.5.

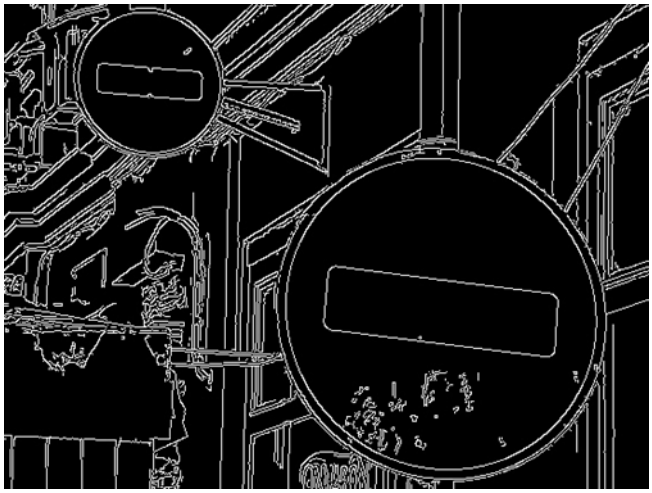

Рисунок 3.4 – Результат применения детектора границ Кэнни

3.1.5. Класс DetectFolder

Класс `DetectFolder` выполняет поиск дорожных знаков для каждой фотографии из выбранной папки. Также здесь имеется возможность производить обработку изображения несколькими каскадами, что увеличивает количество видов найденных знаков.

В методе `DetectAll`() производится проверка всех фотографий в выбранной директории на предмет присутствия на них дорожных знаков с помощью файлов каскадов Хаара и производит запись на жесткий диск найденных областей интереса (обрезанных частей изображения, на котором были обнаружены дорожные знаки) в соответствии с названиями исходных фотографий.

3.1.6. Класс OpenHaarCascadeFileDialog

Данный класс предназначен для создания окна, в котором пользователь выбирает файл формата XML, в котором хранится результат обучения каскада Хаара.

Главным методом этого класса является `openCascade`. При его вызове вызывается стандартное окно указания пути ОС Windows, с помощью которого пользователь указывает путь к нужному каскаду Хаара. Метод возвращает строку, в которой находится полный путь к выбранному каскаду.

3.1.7. Класс ImageFolder

Используя методы этого класса можно получить путь к директории, содержащей изображения в соответствующих форматах (jpg, png, bmp, и так далее).

Метод `Load()` производит открытие папки по выбранному пути и заполнения массива путей к изображениям из этой папки и возвращает количество найденных изображений.

Метод `GetCount`() возвращает количество изображений в определенной папке.

Метод `GetPath`() возвращает путь к определенной папке.

Метод `SetPath`() производит замену пути к папке, после которой заново получает доступ к изображениям в папке по новому пути.

Метод `GetAllImgs`() возвращает массив данных типа string в которых содержатся пути к изображениям из определенной папки.

Метод `GetImg`(`int`) возвращает путь к изображению с полученным номером.

Метод `Sort()` необходим для упорядочивания изображений по алфавиту для корректного соответствия имени файла с изображением и его координат.

3.1.8. Класс VideoFolder

Используя методы этого класса можно получить путь к директории, содержащей видеозаписи в формате MP4.

Метод `Load()` производит открытие папки по выбранному пути и заполнения массива путей к видеозаписям из этой папки и возвращает количество найденных видеозаписей.

Метод `GetCount`() возвращает количество видеозаписей в определенной папке.

Метод `GetPath`() возвращает путь к определенной папке.

Метод `SetPath`() производит замену пути к папке, после которой заново получает доступ к видеозаписям в папке по новому пути.

Метод `GetAllImgs`() возвращает массив данных типа string в которых содержатся пути к видеозаписям из определенной папки.

Метод `GetImg`(`int`) возвращает путь к видеозаписи с полученным номером.

3.1.9. Класс OpenVideoFolderFileDialog

Этот класс нужен для вывода на экран окна для выбора папки с видеозаписями, которые нужно использовать для пополнения базы данных новыми данными.

Главным методом этого класса является `openFolder`. При его вызове создается стандартное окно указания пути ОС Windows, с помощью которого пользователь указывает путь к папке. Из нее программа получает доступ к видеозаписям, которые необходимо обработать и выдать нужный результат.

Метод возвращает массив строк, в котором находится полный путь к видеозаписям из выбранной папки.

3.1.10. Класс OpenPictureFolderFileDialog

Этот класс используется для вывода на экран окна, которое используется для выбора папки с изображениями, которые нужно использовать для нахождения знаков.

Главным методом этого класса является `openFolder`. При его вызове создается стандартное окно указания пути ОС Windows, с помощью которого пользователь указывает путь к папке, из которой программа может получить доступ к изображениям. Метод возвращает массив строк, в котором находится полный путь к изображениям из выбранной папки.

3.1.11. Класс OpenPictureFileDialog

Этот класс используется для вывода на экран окна, которое используется для выбора изображения.

Главным методом этого класса является `openFile`. При его вызове создается стандартное окно указания пути ОС Windows, с помощью которого пользователь указывает путь к изображению. Метод возвращает строку, в котором находится полный путь к изображению.

3.1.12. Класс OpenTextFileDialog

Этот класс используется для вывода на экран окна, которое используется для выбора текстового файла формата txt.

Главным методом этого класса является `openFile`. При его вызове создается стандартное окно указания пути ОС Windows, с помощью которого пользователь указывает путь к текстовому файлу. Метод возвращает строку, в котором находится полный путь к текстовому файлу.

3.1.13. Класс OpenXmlFileDialog

Этот класс используется для вывода на экран окна, которое используется для выбора файла формата xml.

Главным методом этого класса является `openFile`. При его вызове создается стандартное окно указания пути ОС Windows, с помощью которого пользователь указывает путь к файлу. Метод возвращает строку, в котором находится полный путь к файлу.

3.1.14. Класс PhCoord_Connect

Данный класс служит для создания коллекции вида `PhotoData`, с помощью которой можно соотнести изображение с его географическими координатами.

Главный метод `Connect` выполняет создание коллекции с последующим её возвратом. Поиск координат производится по времени с первого кадра. Далее к нулевому отсчёту добавляется частота раскадровки видеозаписи с последующим созданием объекта и добавлением его в коллекцию `PhotoData`.

3.1.15. Класс ShapeDetection

В этом классе находятся функции поиска геометрических фигур на изображении.

Метод `detectShapes` принимает изображение, производит поиск геометрических фигур (треугольники, квадраты, прямоугольники, круги, шестиугольники), выделяет их и возвращает полученное изображение. Поиск производится с помощью метода контурной аппроксимации.

В методе `detectShape` производится поиск и выделение областей определенных фигур. При вызове функции указывается, какие фигуры нужно обнаружить, очертить и вернуть полученное изображение.

3.1.16. Класс Photo

Этот класс содержит описание изображения с помощью его номера в папке и его имени.

3.1.17. Класс CNN

Переменная net типа `Network<double>` хранит в себе слои нейронной сети (НС) и веса нейронов, находящихся на них.

В переменной `stepCount` находится счетчик итерации обучения сети.

Для проведения загрузки данных заранее обученной сети используется переменная `loadedJson`, в которой находится информация о НС в строковом формате, которая потом преобразуется к типу нейронной сети.

Поле `JsonToSave` хранит данные о сети в формате string для дальнейшего сохранения её данных.

Переменная `Aim` отражает минимальную точность, которую необходимо достичь при обучении НС.

Переменная `Acc` хранит достигнутую точность обученной НС.

В переменной `classes` типа int хранится количество слоев созданной НС.

Флаг `isNetLearned` отражает статус НС (обучена ли на данный момент НС).

Объект trainer класса `SgdTrainer` является необходимой для проведения обучения НС. В ней хранятся: скорость обучения НС, размер выборки для каждой стадии обучения, момент НС.

В переменной path хранится путь для сохранения либо загрузки данных сети.

Коллекция `testAccWindow` класса `CircularBuffer<double>` содержит в себе промежуточные данные о точности обучения сети на каждой стадии на основе тестирования тестовой выборки.

Коллекция `trainAccWindow` класса `CircularBuffer<double>` содержит в себе промежуточные данные о точности обучения сети на каждой стадии на основе тестирования тренировочной выборки.

Метод `CreateCNN` служит для создания нейронной сети, добавления необходимых слоев и инициализации классов распознаваемых дорожных знаков. Возвращает количество слоев созданной НС.

В методе `TeachCNN` происходит обучение нейронной сети методом градиентного спуска. Здесь создается обучающая и проверочная выборки и происходит корректировка весов каждого слоя до тех пор, пока точность распознавания сети не достигнет необходимого значения.

Метод `Train` вызывается на каждую итерацию обучения. Он производит изменение весов каждого слоя для улучшения качества распознавания.

Метод `Test` служит для проверки качества распознавания НС для тех весовых коэффициентов, которые на данный момент записаны в НС.

В методе `SaveCNN` происходит запись в файл по заранее определенному пути данных об обученной сети. Это выполняется с помощью приведения данных к стандартному виду, пригодному для дальнейшего использования. В имени полученного файла находятся наиболее важные данные о сети для выбора необходимой сети при загрузке.

При вызове метода `LoadCNN` производится загрузка и инициализация данных о сети, параметры которой находятся в файле по заданному пути.

Метод `Recognize` служит для непосредственной классификации дорожного знака на изображении. Для этого информацию об изображении в виде байтового массива пропускают через обученную сеть и на основе полученных результатов на выходном слое, и выбирается наибольшее значение слоя. Номер нейрона выходного слоя говорит о принадлежности изображения к данному классу дорожных знаков. Далее вызывается метод `GetClassNameFromNumber`, который на основе номера класса возвращает его названия для удобного восприятия пользователем.

Метод `GetLayersCount` возвращает количество слоев в созданной нейронной сети.

Метод `GetClassesCount` возвращает количество распознаваемых классов дорожных знаков в созданной нейронной сети.

Метод `GetAccuracy` возвращает точность заранее обученной нейронной сети.

Метод `IsLearned` возвращает флаг, который говорит о состоянии нейронной сети на предмет ее обученности.

3.1.18. Класс CircularBuffer

Данный класс служит для хранения промежуточных результатов обучения нейронной сети. Он является коллекцией шаблонов с возможностью добавления в нее новых элементов.

Переменная `buffer` является массивом шаблонов в котором хранятся элементы.

Переменная `nextFree` хранит номер следующей свободной ячейки для добавления нового элемента.

Метод `Add` производит добавление нового элемента и изменение номера следующего возможного для добавления элемента.

3.1.19. Класс DataSet

Этот класс производит создание набора данных для обучения нейронной сети. Он содержит в себе набор изображений в количестве заранее определенного размера пачки из обучающей выборки.

Метод `NextBatch` производит создание кортежа из набора обучающих изображений, набора тестовых изображений и массива соответствующих тестовой выборке номеров классов.

3.1.20. Класс Datasets

Данный класс содержит в себе объекты `Train` и `Test` класса `DataSet`, которые являются хранилищами обучающих и тестовых выборок для обучения сети.

В методе Load происходит чтение выборок с жёсткого диска и создание объектов Train и Test.

3.1.21. Класс ImageEntry

В классе ImageEntry содержится сущность описывающая изображение в виде, пригодном для создания выборок при обучении сети.

3.1.22. Класс ImageReader

Данный класс служит для считывания с диска компьютера изображений для создания выборки в виде байтового массива.

Метод Load производит инициализацию коллекций для последующего их заполнения.

Метод LoadImages выполняет операцию считывания для каждого найденного изображения в выбранной директории и его загрузки в коллекцию.

Метод LoadLabels выполняет операцию считывания названий для каждого найденного изображения в выбранной директории, в которых находится номер класса дополнение коллекции номерами классов.

3.1.23. Файл Enums

В этот файл отдельно вынесены используемые перечисления. Среди них:

– `XMLAccessMode`, в котором содержатся виды операций для взаимодействия с XML файлом, в котором находятся веса нейросети. Значение `Get` соответствует операции операции чтения, `Set` – операции записи.

– `NeuronType`, где перечисляются типы вычислений нейрона для соответствующего слоя. Значение `Convolutional` соответствует нейрону сверточного слоя, `Fullyconnected` – полносвязного, `Subsampling` – слоя субдискретизации, `Output` – выходного слоя.

– `NetworkMode` описывает режимы работы сети. Значение Train говорит о том, что сеть находится в режиме обучения, `Test` – тестирования, `Work` – непосредственно работы.

3.2. Структура организации Properties.Settings

Организация Properties.Settings – это xml файл, который можно найти в папке пользователя. Данный файл позволяет хранить и получать доступ к значениям, которые сохраняются между сеансами выполнения приложения. Эти значения называются *параметры*. Используя параметры, могут быть записаны пользовательские настройки или ценные сведения, которые приложению необходимо использовать.

В данном файле хранятся следующие настройки приложения:

– номер версии приложения;

– последний выбранный путь к папке с видеозаписями;

– последний выбранный путь к папке с изображениями;

– последний выбранный путь к обученному каскаду;

– путь к файлу формата XML, в котором хранятся веса нейронов для сверточной нейронной сети.

– флаг первого открытия приложения для отображения сообщения с инструкциями по пользованию.

3.3. Экспорт результатов в базу данных

3.3.1. Класс ResultExport

В этом классе происходит связь с базой данных, записи новых результатов и обновления уже существующих.

Поле `login` содержит имя пользователя для подключения к базе данных.

Поле `password` содержит пароль для подключения к базе данных.

Поле `dataSource` указывает значение, соответствующее атрибуту источника данных.

В методе `ConnectToDB` происходит подключение к базе данных с помощью связки логина и пароля и проверяется доступность базы данных. Метод возвращает значение `true`, если соединение установлено и база данных доступна, либо значение `false`, если произошла ошибка.

Метод `Export` производит создание в базе данных новой записи, содержащей данные обнаруженного знака.

Метод `Find` нужен для предотвращения повторной записи дорожного знака, если он был обнаружен на другой видеозаписи. Поиск происходит по географическим координатам, а также номеру и километру дороги. Если хотя бы одна пара значений (долгота и широта, либо номер дороги и километр) совпали, то происходит лишь обновление даты.

Метод `CloseConnection` производит закрытие соединения с базой данных.

3.3.2. Класс Result

Этот класс служит для стандартизации результатов обработки и распознавания.

Поле `id` отображает уникальный номер знака.

Поле `signClass` содержит номер класса, к которому принадлежит знак.

Поле `lattitude` показывает географическую широту.

Поле `longitude` показывает географическую долготу.

Поле `date` содержит время на видеозаписи, когда знак был заснят.

Поле `roadId` указывает номер дороги, на которой расположен знак.

Поле `roadKm` показывает километр от начала дороги, на которой расположен знак.

Метод `getId` возвращает уникальный номер знака в формате int.

Метод `getSignClass` возвращает номер класса знака в формате int.

Метод `getLattitude` возвращает географическую широту расположения знака в формате string.

Метод `getLongitude` возвращает географическую долготу расположения знака в формате string.

Метод `getDate` возвращает время на видеозаписи, когда дорожный знак был заснят, в формате DateTime.

Метод `getRoadId` возвращает уникальный номер дороги, на которой расположен знак в формате int.

Метод `getRoadKm` возвращает километр от начала дороги, на которой расположен знак, в формате double.

3.4. Хранение полученных результатов в базе данных Oracle

Для сохранения результатов в базе данных Oracle будет использоваться таблица определенными полями. Их можно увидеть в таблице 3.1.

Таблица 3.1 Структура таблицы в базе данных

Название поля	Описание поля
1	2
1 Id	Уникальный номер знака
2 Class	Номер класса знака
3 Lattitude	Географическая широта расположения знака
4 Longitude	Географическая долгота расположения знака
5 Date	Дата обнаружения знака на видеозаписи
6 RoadId	Id дороги, на которой расположен знак
7 RoadKm	Километр от начала дороги, на которой расположен знак