УДК 004

Анализ изображений с применением ML.NET

Фатеенков Данила Витальевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье описан процесс создания модели машинного обучения с использованием C# и библиотеки ML.NET для классификации изображений. Для классификации используются изображения с пятью разными видами цветов. Полученная модель используется в приложении, а также сравнивается с моделью, которая была сделана в программе для анализа данных Orange.
Ключевые слова: машинное обучение, анализ изображений, C#, ML.NET

Image analysis with ML.NET

Fateenkov Danila Vitalievich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the process of creating a machine learning model using C# and ML.NET library for image classification. Images with five different kinds of colors are used for classification. The resulting model is used in the application and is also compared with the model that was made in the Orange – data analysis program.
Keywords: machine learning, image analysis, C#, ML.NET

Научный руководитель:
Баженов Руслан Иванович
Приамурский государственный университет имени Шолом-Алейхема
к.п.н., доцент, зав. кафедрой информационных систем, математики и правовой информатики

1. Введение

1.1 Актуальность

Анализ изображений в настоящее время является одной из самых востребованных сфер машинного обучения. Благодаря анализу изображений искусственным интеллектом, появляется возможность диагностировать заболевания на ранних стадиях, находить трудноразличимые объекты на фотографиях с плохим качеством.

Анализ изображений в настоящее время востребован во многих сферах человеческой деятельности (не только в области IT). Одной из основных задач анализа изображений является классификация по определённому параметру. При решении задачи на классификацию изображений применяются нейронные сети, а также создаются отдельные системы.

Системы для анализа изображения востребованы, а их создание способно упростить и ускорить работу с данными.

1.2 Обзор исследований

А.В. Марков провёл анализ двух нейронных сетей East и PseNet, которые предназначены для анализа изображений [1]. Р.Ф. Долгачев проанализировал процесс классификации изображения с помощью машинного обучения и вывел основную проблему при обучении сверточной нейронной сети [2]. О.А. Степанова, А.А.Лебедев, В.В. Хрящев и А.Л. Приоров представили алгоритм детектирования патологий на основе сверточной нейронной сети SSD [3]. А.В. Матвеев, М.Ю. Машуков, А.В. Нартова, Н.Н. Санькова и А.Г. Окунев представили описание работы облачного сервиса DLgram01, предназначенного для автоматизированной обработки изображений [4]. М.В. Алейников и Н.М. Ершов реализовали на основе технологии Tesseract приложение для распознавания печатных текстов 19 века [5]. В области анализа данных также представлен метод определения формы невидимых частиц с помощью анализа динамических изображений [6].

1.3 Цель исследования

Цель исследования – разработать с использованием языка программирования C# и библиотеки ML.NET модель машинного обучения для анализа и классификации изображений.

2. Материалы и методы

Для реализации поставленной задачи используется язык программирования C# и библиотека ML.NET, а также набор изображений цветов, скачанный на сайте kaggle [7].

3. Результаты и обсуждения

Классификация изображений – задача из области компьютерного зрения. ML.NET предоставляет разработчику два сценария работы с изображениями: классификация изображений и распознавание объектов. Для реализации данных сценариев используется Model Builder – инструмент, позволяющий создавать модели машинного обучения на основе указанных параметров и набора данных (с дальнейшей интеграцией в проект разработчика).

Для анализа изображений существует большое количество моделей машинного обучения, одним видом которых являются нейронные сети. Нейронные сети принимают входные данные, на основе полученной

информации обучаются и классифицируют изображения, исходя из полученных знаний. В ML.NET используются нейронные сети для работы с изображениями. Также библиотека позволяет упростить процесс создания модели машинного обучения с помощью инструмента Model Builder.

Model Builder автоматизирует процесс разработки моделей машинного обучения и разделяет его на несколько этапов: выбор сценария (классификация текста, изображений, прогнозирование, кластеризация и др.), выбор набора данных, дальнейшее обучение модели и интегрирование готовой модели в проект.

При работе с изображениями разработчику представлены 3 среды для обучения модели: локально (с использованием мощностей центрального процессора), локально (с использованием графического процессора) и Azure (платформа для облачных вычислений) (рис. 1).

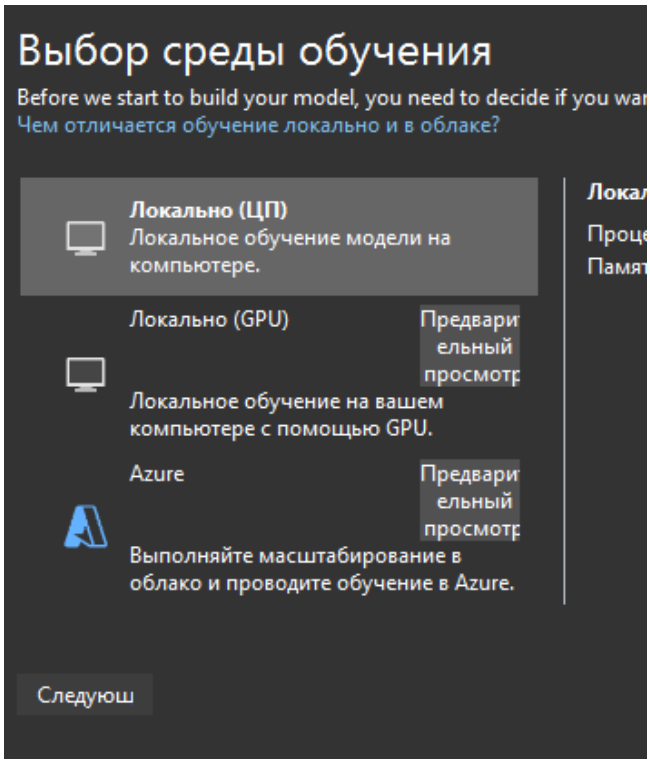

Рисунок 1. Этап с выбором среды обучения

Данные необходимо загружать в структурированном виде, так как модель будет ссылаться на названия каталогов для классификации изображений.

После загрузки изображений в Model Builder, разработчику будет показана структура набора данных (предварительный просмотр данных). Model Builder поддерживает png, jpg и gif форматы при работе с изображениями.

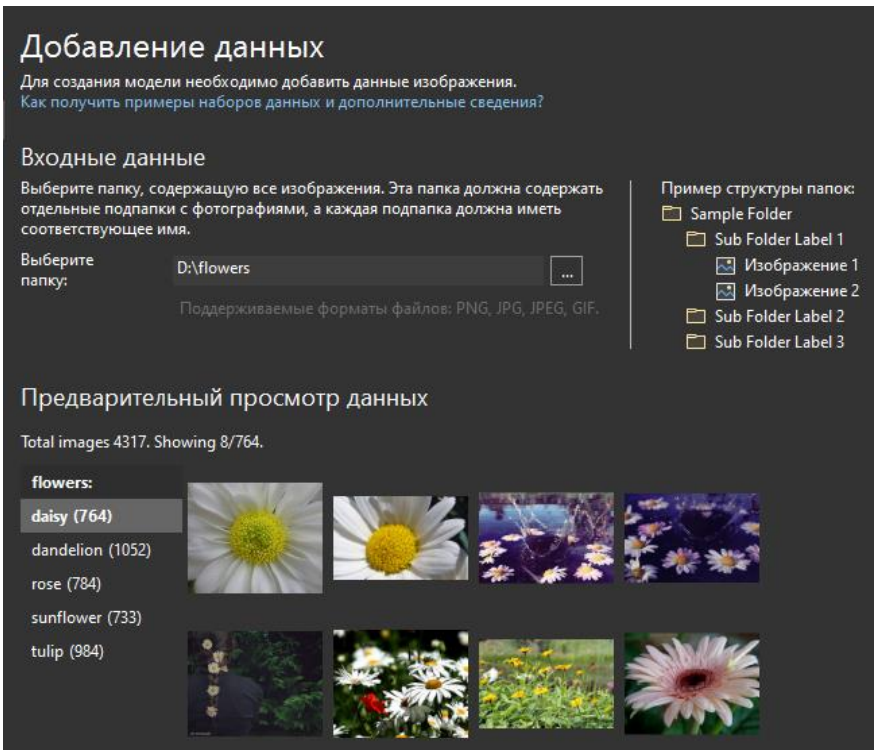

Рисунок 2. Этап добавления набора данных в Model Builder

Обучение модели может занять длительное время не только из-за количества данных, но и из-за необходимости установить дополнительные пакеты для анализа изображений. После завершения обучения, разработчику будет представлена наиболее подходящий для решения задачи алгоритм, а также его точность определения класса изображения (рис. 3).

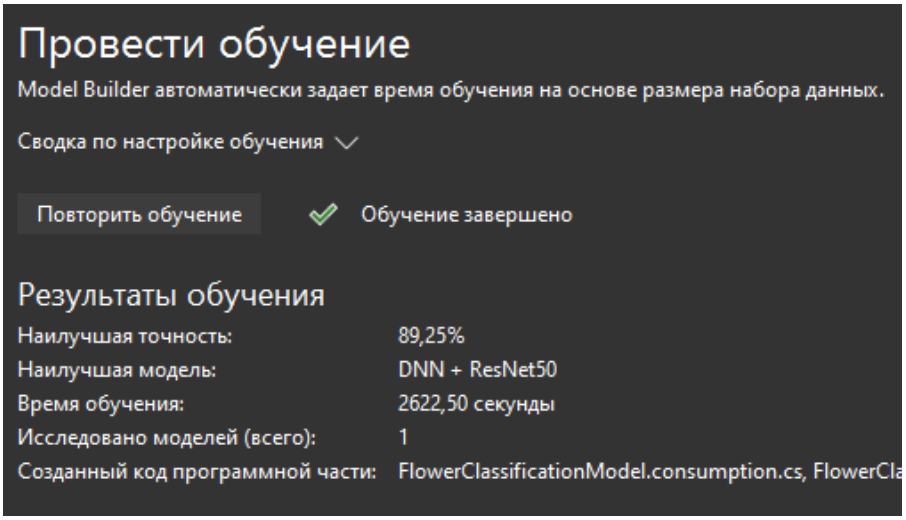

Рисунок 3. Результаты обучения модели

После завершения разработчику будет предложено протестировать модель и добавить своё изображение для анализа. После тестирования в проект можно добавить 2 ранее сгенерированных файла: consumption (содержит описание данных, также заданы функции для обработки и предсказания результата при загрузке изображений) и training (нужен для

обучения модели). Алгоритм на входе принимает один параметр: путь к изображению, представленный в виде строки:

```
FlowerClassification.ModelInput sampleData = new
FlowerClassification.ModelInput() {
     ImageSource =
@"D:\flowers\daisy\100080576_f52e8ee070_n.jpg",
};
```

sampleData является объектом созданного класса ModelInput, описанный в consumption. Данный класс содержит 2 поля: Label и ImageSource. Первое не используется при работе с моделью, а второе является путём к изображению, которое нужно проанализировать.

После завершения создания sampleData, объект передаётся в функцию Predict в качестве параметра и алгоритм анализирует изображение:

```
var predictionResult =
FlowerClassification.Predict(sampleData);
```

Для упрощения работы с моделью, стоит сделать простую форму, в которой пользователь сможет загрузить изображение и получить результат работы модели. Для этого на форме создаются 3 объекта: Image, Button и TextBlock. В Image отображается изображение, выбранное пользователем, при нажатии на кнопку открывается окно выбора изображения, а TextBlock содержит строку результата работы модели.

Необходимо создать функцию, которая при нажатии на кнопку будет открывать окно выбора изображения. Для создания диалогового окна для выбора необходимого файла создаётся объект класса OpenFileDialog. Так как модель работает с изображениями, то необходимо задать фильтр для расширения файлов, чтобы пользователь мог выбрать только изображение определённого формата. Для этого задаётся метод Filter:

```
OpenFileDialog dialog = new OpenFileDialog();
dialog.Filter = "Изображения(*png; *jpg)|*png; *jpg";
```

После выбора изображения необходимо его загрузить в ранее созданный объект Image через класс BitmapImage:

```
var imageFile = dialog.FileName;
ImageMain.Source = new BitmapImage(new Uri(imageFile));
```

Последним этапом в создании формы является создание объекта ModelInput модели машинного обучения и последующим вызовом функции Predict для анализа изображения. Делается это по аналогии с примером, который был представлен при генерировании кода для проекта:

```
var inputData = new FlowerClassification.ModelInput {
     ImageSource = imageFile
};

var predictedResult =
FlowerClassification.Predict(inputData);
```

В качестве параметра ImageSource используется выбранное пользователем изображение.

После анализа изображения необходимо вывести результат в TextBlock (рис. 4)

Рисунок 4. Пример работы формы для анализа изображений

Алгоритм выполняет свою работу хорошо и в большинстве случаев правильно определяет название цветка на изображении. Данную модель можно сравнить с моделью в Orange по параметру “Accuracy”. Для создания модели в Orange используется модуль для работы с изображениями “Image Analytics”. Для анализа набора изображений выбрана нейронная сеть VGG-16. После анализа нейронной сетью, изображения делятся на 5 групп (так как в наборе данных используется только 5 категорий цветков) (рис. 5).

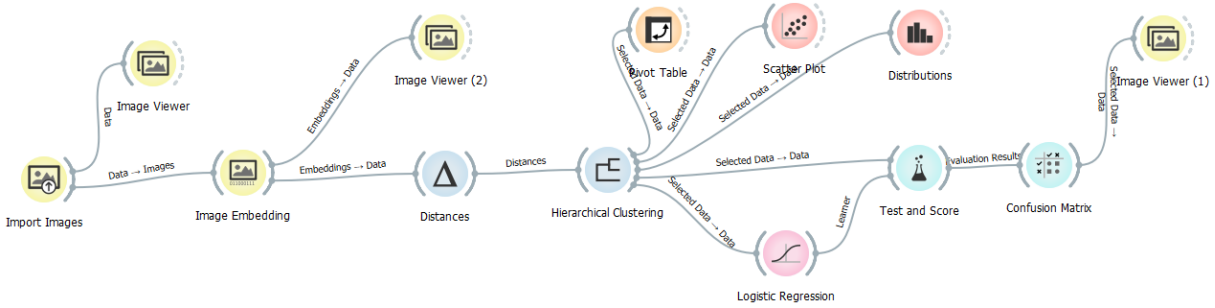

Рисунок 5. Модель для анализа изображений в Orange

Из полученных данных в Scatter Plot и Pivot Table можно увидеть распределение изображений по категориям и кластерам (рис. 6).

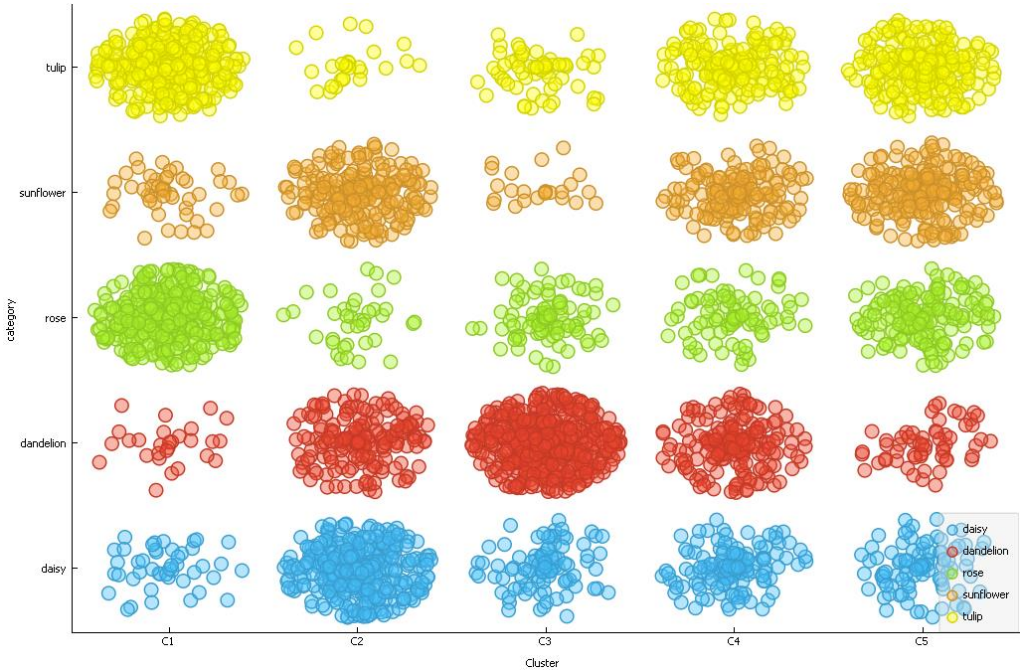

Рисунок 6. Распределение изображений по категориям и кластерам в Scatter Plot

Для модуля Distances выбрана метрика Absolute Pearson.

Так как использование одной нейронной сети и модуля Distances не дают необходимого результата (можно видеть только распределение по кластерам, но по каким параметрам и как изображения распределены пользователь узнать не сможет, если работает с большим количеством данных), то необходимо применить модель машинного обучения, чтобы увидеть конечный результат. Используя логистическую регрессию, можно распределить изображения из кластеров по категориям и увидеть точность модели. Тип регуляции для логистической регрессии – Ridge (рис. 7).

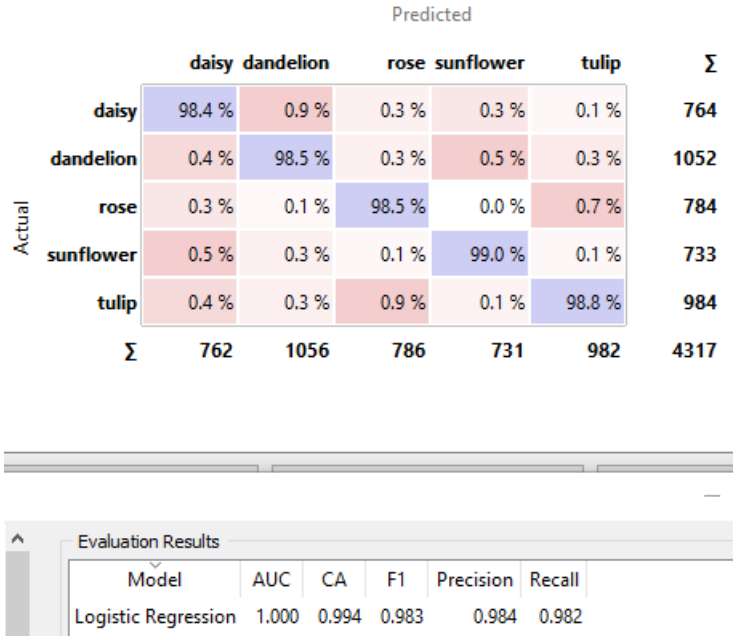

Рисунок 7. Параметры логистической регрессии и матрица путаницы

Точность модели, созданной на C#, сравнива с параметром CA – “Classification Accuracy”.

4. Выводы

Таким образом, ML.NET предоставляет разработчику эффективные средства для анализа данных. Model Builder позволяет автоматизировать процесс создания моделей для анализа и классификации изображений, а итоговый результат подходит по основному параметру (точности) для использования модели в проекте. Также можно отметить скорость работы алгоритма: обученная модель показывает хорошие результаты при работе с новыми изображениями.

В статье был описан процесс создания модели для анализа изображений с использованием ML.NET.

Библиографический список

1. Марков А.В. Проведение сравнительного анализа двух нейронных сетей east и psenet в задаче детекции текста на изображениях прейскурантов // Евразийский союз ученых. 2020. № 4-4 (73). С. 41-46
2. Долгачев Р.Ф. Анализ процесса классификации изображения с помощью машинного обучения // Альманах научных работ молодых ученых университета ИТМО. 2019. № 1. С. 72-74
3. Степанова О.А. и др. Использование сверточной нейронной сети ssd для детектирования патологий при эндоскопии желудка // Цифровая обработка сигналов и её применение. DSPA-2019. 2019. № 1. С. 533-537
4. Матвеев А.В. и др. Автоматический анализ изображений микроскопии с применением облачного сервиса DLGRAM01 // Физико-химические аспекты изучения кластеров, наноструктур и наноматериалов. 2021. № 13.

С. 300-311
5. Алейников М. В., Ершов Н. М. Применение методов машинного обучения в задачах распознавания печатных текстов 19 века // Системный анализ в науке и образовании: сетевое научное издание. 2021. № 1. С. 12–22. URL: http://sanse.ru/download/422.
6. Mathaes R. et al. Shape Characterization of Subvisible Particles Using Dynamic Imaging Analysis // Journal of Pharmaceutical Sciences. 2020. Т. 109. № 1. С.375-379
7. Flowers Recognition. URL: https://www.kaggle.com/alxmamaev/flowers-recognition (дата обращения: 20.12.2021)