**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Саратовский государственный технический университет**

**имени Гагарина Ю.А.»**

Факультет Институт прикладных информационных технологий и коммуникаций

Направление Информационные системы и технологии

Кафедра Прикладные информационные технологии

**КУРСОВАЯ РАБОТА**

по дисциплине «*Компьютерные методы моделирования искусственных нейронных сетей*»

на тему «Разработка прикладной нейросетевой экспертной системы для распознавания животных»

Комиссия по защите:

доцент кафедры ПИТ Кузьмин А.К.

старший преподаватель кафедры ПИТ Печкин И.О.

Курсовая работа защищена на оценку «\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись члена комиссии)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись члена комиссии)

Саратов 2024

**Замечания**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись члена комиссии)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись члена комиссии)

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Саратовский государственный технический университет**

**имени Гагарина Ю.А.»**

Кафедра «Прикладные информационные технологии»

ЗАДАНИЕ

на выполнение курсовой работы

по дисциплине «*Компьютерные методы моделирования искусственных нейронных сетей*»

студенту ИнПИТ группы б1 ИФСТ-31

Кошелевой Алины Денисовны

В курсовой работе необходимо:

1. Для моделирования нейросетевой модели (или какой-либо её составной части) должен использоваться свободно-распространяемый фреймворк. Выбор фреймворка свободный, но рекомендуется использовать Google TensorFlow или Keras.
2. Решаемая прикладная задача должна реализовываться как задача классификации (хотя допускается также кластеризация или моделирование ассоциативной памяти).
3. Проблематика решаемой задачи должна быть реальной. Для обучения нейронной сети должен использоваться реальный набор данных (датасет), опубликованный в сети Интернет. Рекомендуется использовать платформу kaggle.com, которая предоставляет доступ к большому количеству реальных прикладных задач и соответствующим наборам для обучения нейросетевых моделей.
4. Задача должна быть трудно формализуемой, для которой непригодны классические методы принятия решений. Другими словами, использование технологий машинного обучения должно быть обосновано для выбранной задачи.
5. Графический интерфейс системы должен быть законченным и предоставлять возможность пользователю в интуитивно-понятной манере обучать нейронную сеть, подавать ей на вход данные и получать от неё ответ.

Дата выдачи: 6 февраля 2024 г.

Срок выполнения: 31 мая 2024 г.

Руководитель:\_\_\_\_\_\_\_\_\_ доц. Кузьмин А.К.

Студент:\_\_\_\_\_\_\_\_\_\_\_\_\_ Кошелева А.Д.

**Содержание**

[1. Описание предметной области и решаемой прикладной задачи **Ошибка! Закладка не определена.**](#_Toc105349486)

[1.1 Постановка задачи **Ошибка! Закладка не определена.**](#_Toc105349487)

[1.2 Обоснование использования для решения данной задачи технологий машинного обучения **Ошибка! Закладка не определена.**](#_Toc105349488)

[1.3 Список входных и выходных факторов системы **Ошибка! Закладка не определена.**](#_Toc105349489)

[2. Описание использованного для обучения датасета 7](#_Toc105349490)

[3. Описание выбранной нейросетевой архитектуры 1](#_Toc105349491)1

[4. Описание выбранного фреймворка **Ошибка! Закладка не определена.**](#_Toc105349492)

[5. Подробное описание разработанного программного обеспечения **Ошибка! Закладка не определена.**](#_Toc105349493)

[6. Описание процесса и результатов обучения **Ошибка! Закладка не определена.**](#_Toc105349494)

[7. Демонстрация работы приложения 2](#_Toc105349494)1

[Заключение 2](#_Toc105349495)3

[Список литературы 2](#_Toc105349496)4

**Описание предметной области и решаемой прикладной задачи**

Данная работа посвящена разработке нейросетевой модели для распознавания животных. Она предназначена для автоматического распознавания видов животных по входному изображению. Это задача, требующая высокой точности, учитывает наличие множества видов с характеристическими отличиями в размере, форме, окраске. Нейросеть может распознавать различные формы, размеры, цвета, текстуры тела, и отличительных особенностей. Данная система может пригодится исследователям, у которых имеется большой объем фотографий животных, чтобы наблюдать за изменениями какого-либо вида. Нейросеть справится намного быстрее и точнее чем человек, поэтому необходимо создать модель машинного обучения для классификации животных.

**Обоснование использования для решения данной задачи технологий машинного обучения**

Использование технологии машинного обучения для решения задачи классификации животных имеет несколько причин, объясняющих целесообразность ее применения.

Во-первых, обработка и классификация огромного количества изображений животных - задача, требующая большого количества времени и усилий, если рассматривать традиционные методы, такие как ручная классификация. Так как решение данной задачи требует тщательной подготовки данных, применения алгоритмов предварительной обработки, создания и обучения нейросетевой модели, то для этого использовались современные методы обучения с учителем.

Во-вторых, преимущества машинного обучения заключаются в его способности к обучению на основе большого объема данных, а также в его способности к адаптации для решения различных задач.

Использование машинного обучения для классификации животных позволяет увеличить точность и надежность классификации.

**Список входных и выходных факторов системы**

Входная картинка представляет собой изображение животного. Входные данные передаются нейронной сети в формате многомерного массива, соответствующего пиксельному изображению. Выход нейросети состоит из 90 нейронов, каждый из которых представляет степень принадлежности изображённой на входной картинке животного к определенному виду.

**Описание использованного для обучения датасета**

Для обучения используются тренировочный датасет и валидационный. Все датасеты были взяты с сайта kaggle.com.

Тренировочный датасет включает в себя 2534 фотографий 90 видов:

1. антилопа
2. барсук
3. летучая мышь
4. медведь
5. пчела
6. жук
7. бизон
8. кабан
9. бабочка
10. кошка
11. гусеница
12. шимпанзе
13. таракан
14. корова
15. койот
16. краб
17. ворона
18. олень
19. собака
20. дельфин
21. осел
22. стрекоза
23. утка
24. орел
25. слон
26. фламинго
27. муха
28. лиса
29. коза
30. золотая рыбка
31. гусь
32. горилла
33. кузнечик
34. хомяк
35. заяц
36. еж
37. бегемот
38. птица-носорог
39. лошадь
40. колибри
41. гиена
42. медуза
43. кенгуру
44. коала
45. божьи коровки
46. леопард
47. лев
48. ящерица
49. омар
50. комар
51. мотылек
52. мышь
53. осьминог
54. окапи
55. орангутанг
56. выдра
57. сова
58. бык
59. устрица
60. панда
61. попугай
62. пелеканообразные
63. пингвин
64. свинья
65. голубь
66. дикобраз
67. опоссум
68. енот
69. крыса
70. северный олень
71. носорог
72. кулик
73. морской конек
74. тюлень
75. акула
76. овца
77. змея
78. воробей
79. кальмар
80. белка
81. морская звезда
82. лебедь
83. тигр
84. индейка
85. черепаха
86. кит
87. волк
88. вомбат
89. дятел
90. зебра

**Описание выбранной нейросетевой архитектуры**

Для реализации нейросети использовалась архитектура DNN + ResNet50.

DNN(deep neural network) напоминает структуру многослойного перцептрона. А сам наш перцептрон называется ResNet50.

ResNet50 - это нейронная сеть, которая состоит из 50 скрытых слоев и использует структуру остаточного соединения. В обычных случаях мы можем иметь базовое отображение с нелинейной функцией H (x) от входа к выходу. Допустим, вместо H (x) используйте нелинейную функцию F (x), которая определяется как H (x)-x. На выходе второго слоя весов (справа) мы арифметически добавляем x к F (x). Затем пропускаем F (x) + x через выпрямленный линейный блок (ReLU). Это позволяет нам переносить важную информацию с предыдущего уровня на следующие уровни. Поступая таким образом, мы можем предотвратить проблему исчезающего градиента. Даже если это подключение выглядит как дополнение к стандартному подходу CNN, удивительно, что оно ускоряет обучение сети.

ReLu - это нелинейная функция активации, которая широко используется в глубоком обучении. Она преобразует входное значение в значение от 0 до положительной бесконечности.

**Краткое описание выбранного фреймворка**

В реализации искусственной нейронной сети был выбран фреймворк ML.NET. Это мультиплатформенная платформа разработки (Windows, Mac OS и Linux), с открытым исходным кодом и бесплатная, для создания всех видов приложений: настольных, мобильных, веб-приложений, игр и для Интернета вещей. Данный фреймворк позволяет добавлять в приложения .NET возможности машинного обучения в автономном и подключённом режиме. Это позволяет вам использовать различные языки, редакторы и библиотеки для различных типов возможных приложений. Среди языков: C#, F#, Visual Basic.

Это так же отличная платформа для кросс-разработки, поскольку она позволяет приложениям, созданным на вышеупомянутых языках, запускаться изначально в любой совместимой операционной системе благодаря различным интегрированным реализациям, среди которых: Чистое ядро, .NET Framework, Xamarin / Mono.

ML.NET содержит базовый набор API, общих для всех реализаций .NET. Кроме того, каждая реализация может также предоставлять дополнительные API-интерфейсы, специфичные для операционных систем, в которых они будут работать. Например, .NET Framework — это реализация .NET только для Windows, которая включает API-интерфейсы для доступа к реестру Windows.

ML.NET имеет огромную библиотеку (экосистему пакетов), которая расширяет его функциональность. Чтобы использовать их, вы можете использовать NuGet, менеджер пакетов, созданный специально для «.NET» содержащий более 90.000 XNUMX упаковок.

**Подробное описание разработанного программного обеспечения, включая архитектурную схему, диаграмму классов и основных графических форм**

*AnimalClassification.consumption.cs*

В классе *AnimalClassification* есть два подкласса *ModelInput*, *ModelOutput*, которые отвечают за вход картинки и выход результата.

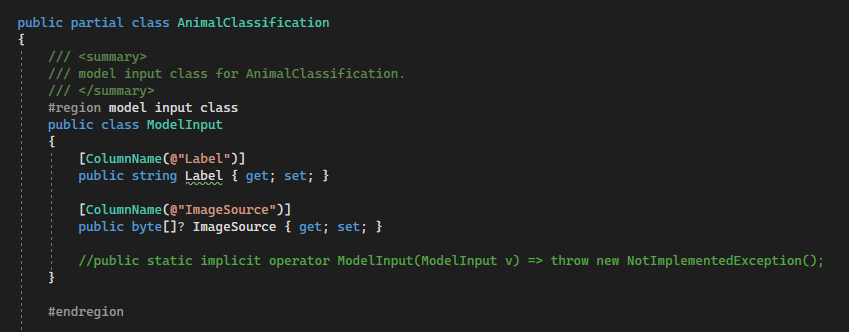


Рисунок 1. класс ModelInput

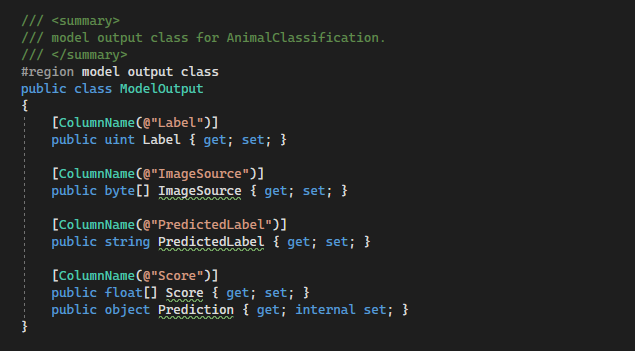


Рисунок 2. класс ModelOutput

Далее у класса *AnimalClassification* есть путь до нашей обученной модели **(**Path.GetFullPath("AnimalClassification.zip")), методы CreatePredictEngine, который отвечает за создание специального контекста, куда загружается обученная модель, для предсказания, и Predict, который и предсказывает наши результаты.

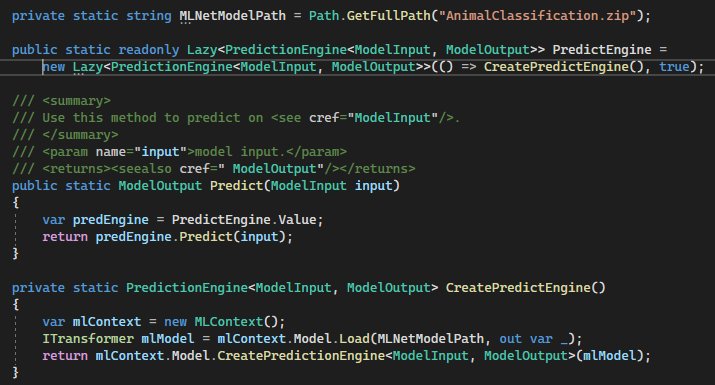


Рисунок 3. Логика предсказания

*AnimalClassification.training.cs*

Метод *RetrainPipeline* переобучает модель, используя конвейер, сгенерированный в рамках процесса обучения.

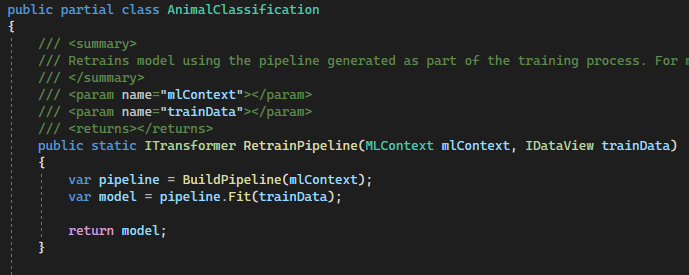


Рисунок 4. RetrainPipeLine

Метод *BuildPipeline* создайте конвейер, который используется в model builder.

В этом методе происходит конфигурация процесса обработки данных с помощью конвейерных преобразований данных

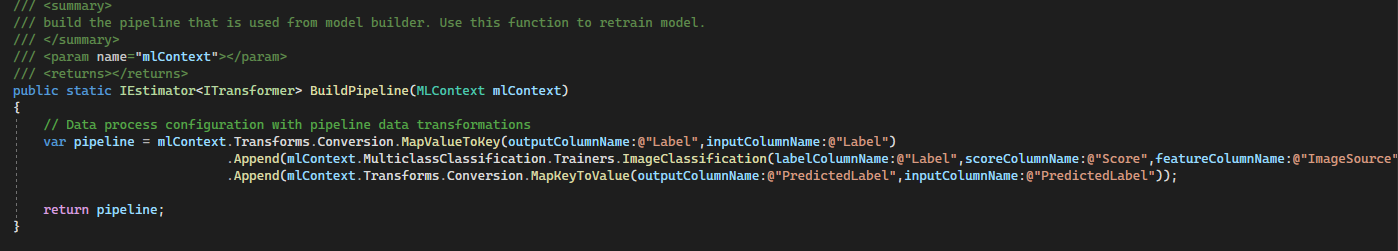


Рисунок 5. BuildPipeline

**MainWindow.xaml**

Класс *MainWindow* отвечает за отображение интерфейса. Метод *OpenImageButton\_OnClick* - обработчик кнопки, в котором происходит загрузка картинки с помощью *OpenFileDialog*.

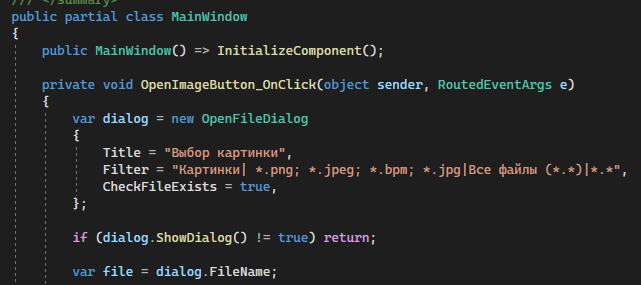


Рисунок 6. Обработчик кнопки

Далее мы преобразуем наше файловое изображение в *BitmapImage*. И конвертируем данные в byte[]. После в метод *Predict* мы прокидываем нашу картинку. И выводим результат предсказания в *ResultText.Text.*

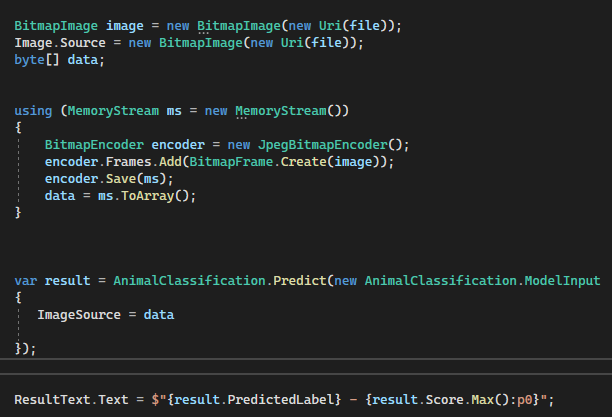


Рисунок 7. Преобразование, предсказание, результат

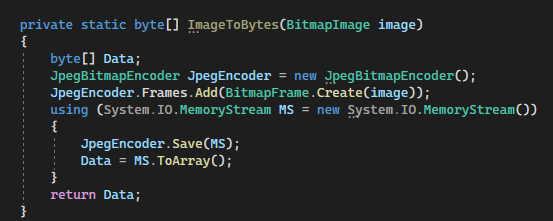


Рисунок 8. Конвертер из string в byte[]

**Описание процесса и результатов обучения**

Библиотека ML.NET была использована для обучения, и даёт возможность обучить модель. ML.NET из графического окна. Пошаговое обучение модели. Первый шаг – выбор сценария обучения.

Сценарии:

1. Data classification – классификация текстовых данных на несколько категорий.
2. Прогнозирование значений – прогнозирование числовых значений на основе данных (регрессия).
3. Классификация изображений – классификация изображений на несколько категорий.
4. Рекомендация – создание списка предлагаемых элементов для определенного пользователя.
5. Обнаружение объектов – обнаружение и идентификация объектов на изображениях.

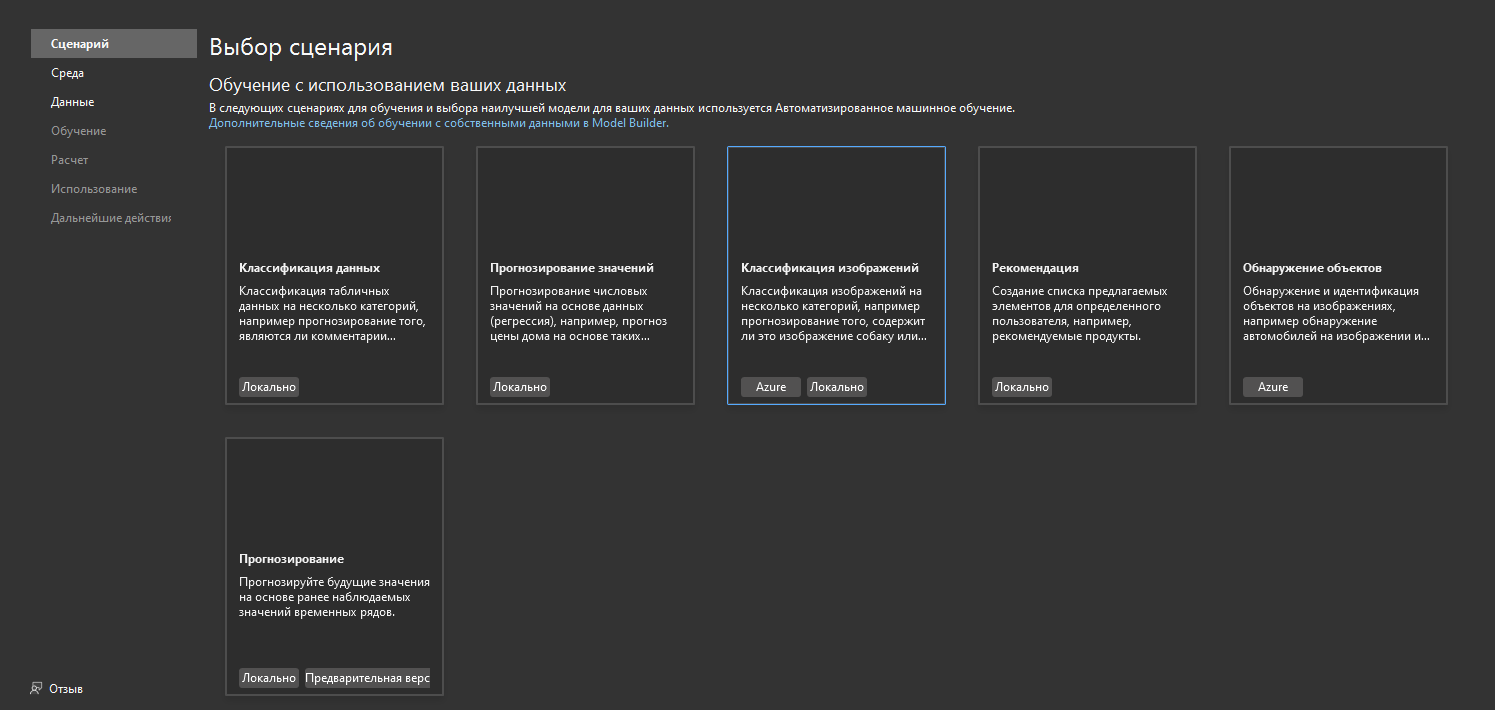


Рисунок 9. Выбор сценария

Второй шаг – выбор среды обучения.

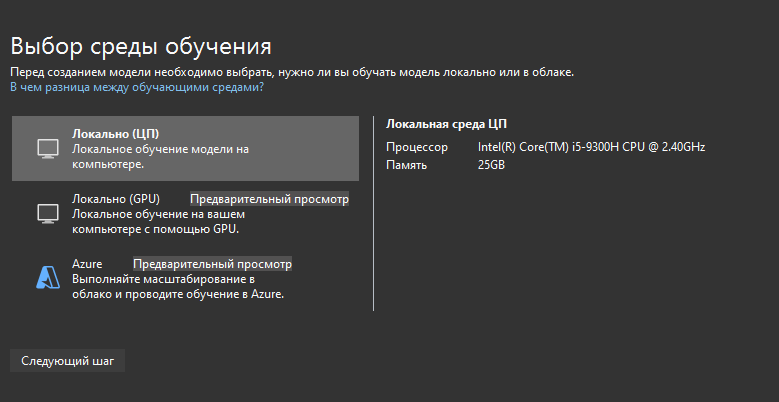


Рисунок 10. Выбор среды обучения

Третий шаг – добавление данных. Перед тем как загружать данные, надо разделить датасет на training и testing. Загружаем тренировочные данные.

(в AnimalClassification\_ConsoleApp -> AnimalClassification.mbconfig)

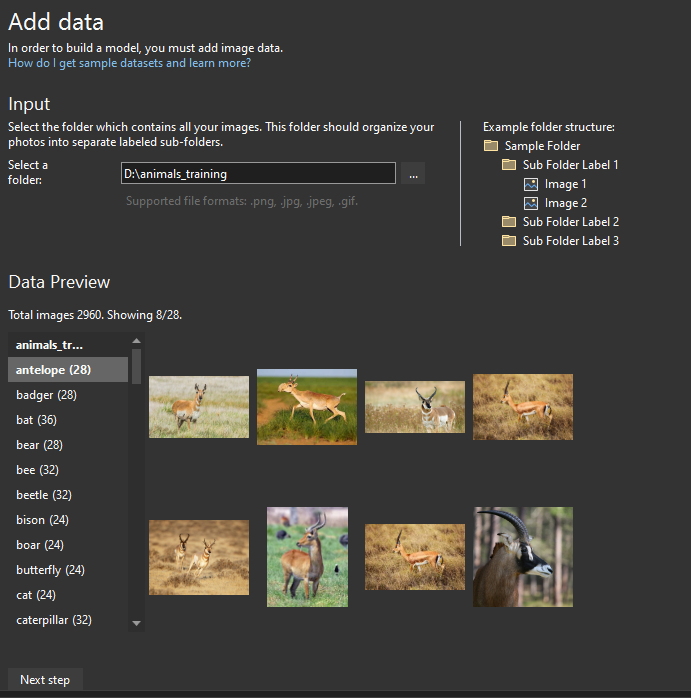


Рисунок 11. Добавление данных

Четвёртый шаг – обучение.

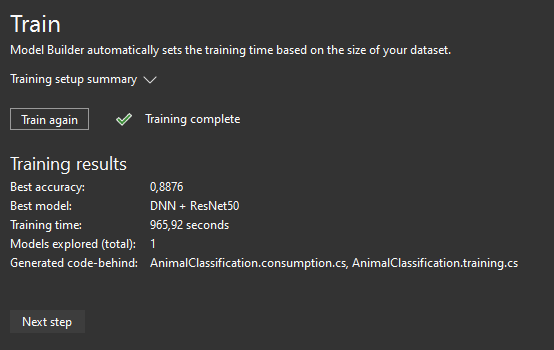


Рисунок 12. Обучение завершилось

Наша сеть обучилась с точностью 88%. Обучалась она 956 сек – 9,3 минуты.

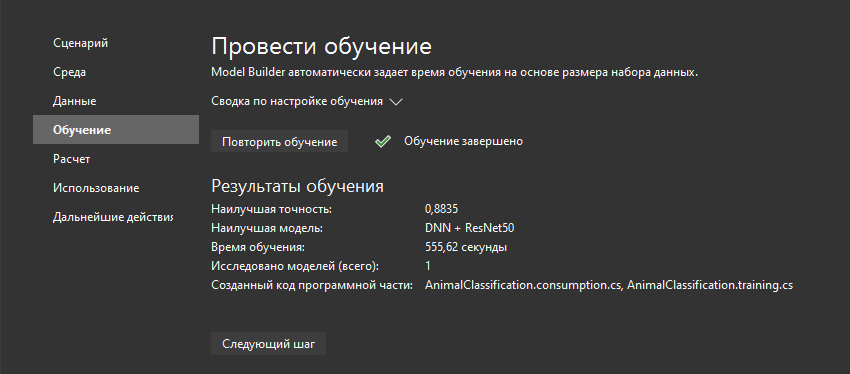
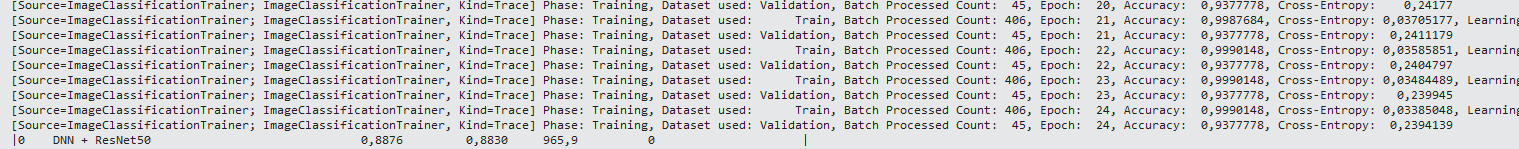


Рисунок 13. Результаты обучения

Количество эпох составило 24. На промежуточных этапах точность достигала 99%



Риснуок 14. Эпохи, точность и перекрестная энтропия

Четвёртый шаг был последним, однако, фреймворк предлагает еще несколько шагов. Расчёт – результаты обучения, название модели и возможность попробовать модель, загрузить фотографию для классификации.

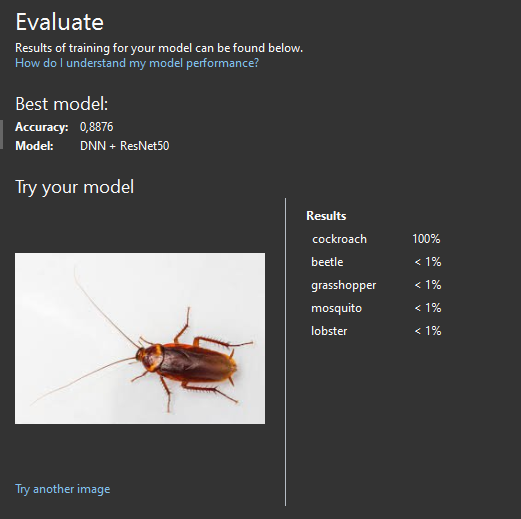


Рисунок 15. Расчет обучения

**Демонстрация работы приложения**

Запускаем Visual Studio

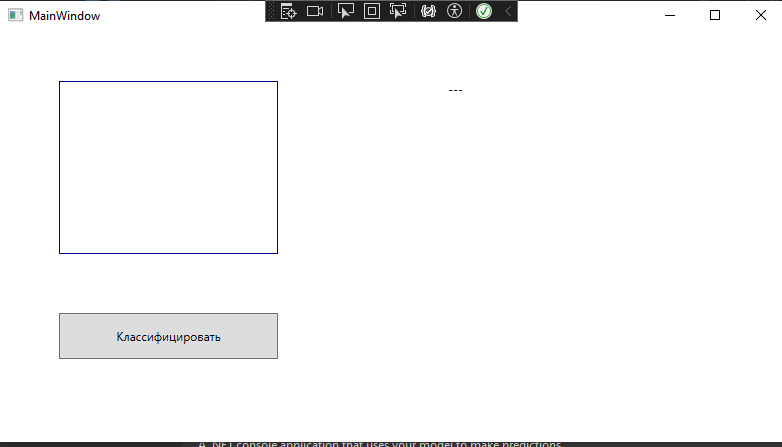


Рисунок 16. Главный экран приложения

Нажимаем на кнопку «Классифицировать»

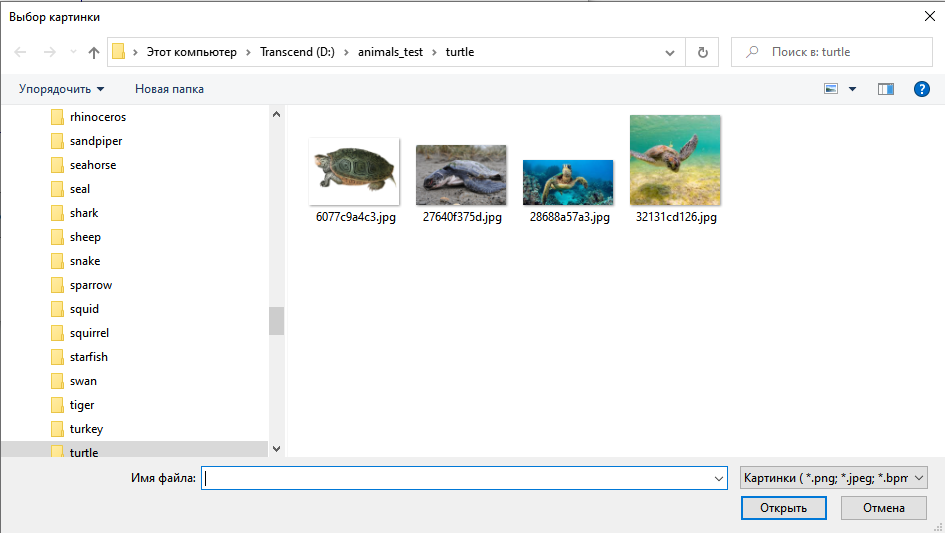


Рисунок 17. Выбираем изображение

Смотрим результат и видим, что приложение определила морскую звезду правильно, точность 100%.

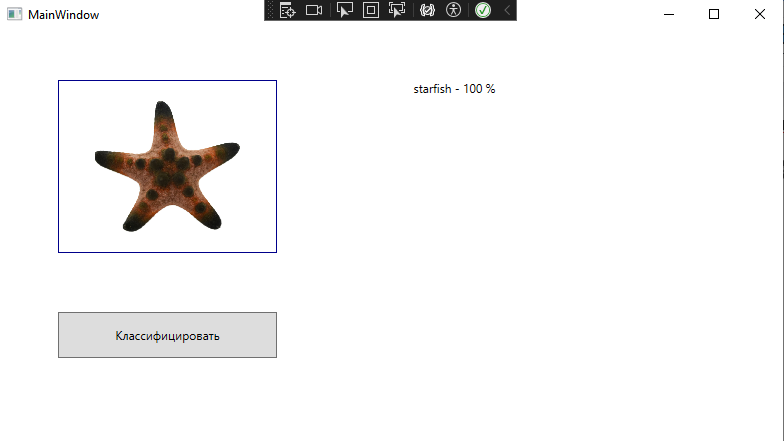


Рисунок 18. Результат

Сделаем повторную проверку, и выберем Морского конька.

Приложение определило картинку правильно, точность 88%

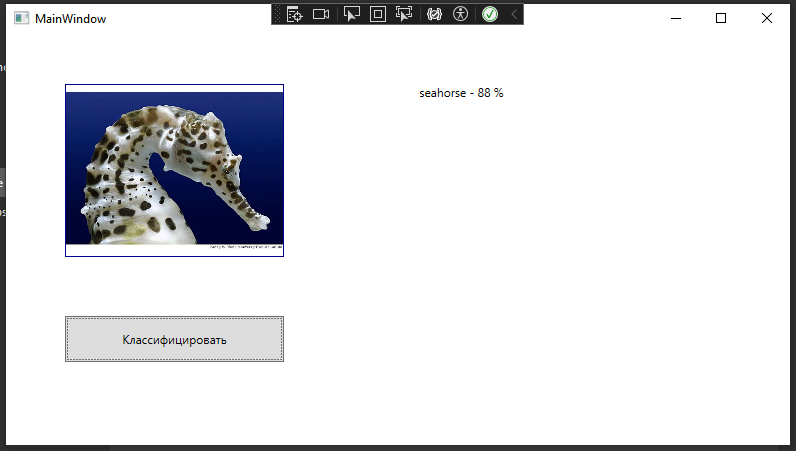


Рисунок 19. Результат

**Заключение**

В процессе написании курсовой работы была реализована прикладная нейросетевая экспертная система для определения животных. Описана предметная область и решаемая прикладная задача. Так же произведено обоснование использования для решения задачи технологий машинного обучения. Был описан датасет и выбранная нейросетевая архитектура.

Конкретная прикладная нейросетевая экспертная система представляет возможность распознать разные виды животных, начиная с кота и заканчивая морским коньком. Такое распознавание может помочь людям быстро определять животных для большего ознакомления.

**Список использованных источников**

1. Share Delightful Machine Learning Apps // Gradio URL: https://gradio.app (дата обращения: 01.03.2024).
2. Start with more than a blinking cursor // Kaggle URL: https://www.kaggle.com (дата обращения: 28.03.2024).
3. Создавайте модели машинного обучения производственного уровня с помощью TensorFlow // TensorFlow URL: https://www.tensorflow.org/?hl=ru (дата обращения: 02.03.2024).
4. Simple. Flexible. Powerful. // Keras URL: https://keras.io (дата обращения: 02.05.2024).
5. Types of artificial neural networks // Wikipedia URL: https://en.wikipedia.org/wiki/Types\_of\_artificial\_neural\_networks (дата обращения: 02.03.2024).
6. Classification in Machine Learning: What it is & Classification Models // Simplelearn URL: https://www.simplilearn.com/tutorials/machine-learning-tutorial/classification-in-machine-learning (дата обращения: 02.03.2024).
7. What are Classification and Regression in ML? // Medium URL: https://medium.com/swlh/what-are-classification-and-regression-3677987b9422 (дата обращения: 02.03.2024).
8. Фреймворк ML.NET [Электронный ресурс] // BLOG.DESDELINUX.NET // URL: https://blog.desdelinux.net/ru/net-ml-net-plataformas-codigo-abierto-microsoft/ (дата обращения: 03.03.2024)
9. МL.NET [Электронный ресурс] // DOTNET.MICROSOFT.COM // URL: <https://dotnet.microsoft.com/en-us/apps/machinelearning-ai/ml-dotnet> (дата обращения: 03.03.2024)
10. WPF [Электронный ресурс] // METANIT.COM // URL: https://metanit.com/sharp/wpf/1.php (дата обращения: 04.03.2024).