

Создание искусственного интеллекта на примере чат-бота идентификации покемонов

26.03.2019

Алиса Кускова

Школа 2026, 10 класс "Л"

г. Москва, 2019 г

Обзор

1. Что такое искусственный интеллект

Чтобы дать определение понятию “искусственный интеллект”, необходимо в первую очередь определиться со значением самого слова “интеллект”. Согласно наиболее упортебляемому определению, интеллект - способность системы в ходе самообучения создавать программы для решения задач определенного класса сложности и решать эти задачи [12]. Соответственно, искусственным интеллектом можно будет назвать техническую или программную систему, обладающую интеллектом и способную правильно интерпретировать внешние данные для достижения определенных целей [7].

2. Чем он полезен, для чего применяется

Преимущества искусственного интеллекта - способность автоматизировать процессы, ранее требовавшие необходимость человеческого участия и высокую скорость и точность анализа большой объемов информации. В сочетании с отсутствием подверженности человеческому фактору, эти свойства позволяют использовать ИИ практически во всех основных сферах человеческой жизни.

В медицине искусственный интеллект применяется для диагностики различных заболеваний, анализа генетической предрасположенности к патологиям, некоторые системы способны моделировать лекарства. В промышленности машины могут осуществлять сбор деталей и бухгалтерские расчеты. Постоянно усложняющиеся и улучшающиеся алгоритмы позволяют искусственному интеллекту исполнять роль консультанта при общении с покупателями или анализировать резюме для повышения эффективности подбора персонала. Большую популярность набирают системы “умного дома” и роботы-пылесосы. [13]

Часто используемой задачей для искусственного интеллекта является классификация - определение принадлежности исследуемого образца к той или иной группе или классу.

Цели

1. Целью проекта является разработка искусственного интеллекта, способного распознавать новые образы, отсутствующие в стандартных базах моделей, используя вычислительные ресурсы бытового компьютера, на примере распознавания изображений покемонов.
2. Обеспечение возможности практического использования создаваемой нейросети людьми без специальных знаний и оборудования в виде бота для telegram.

Актуальность

Развитие возможностей искусственного интеллекта позволяет ему все больше проникать в различные сферы жизни людей, решая не только научные задачи, но и прикладные бытовые, обучающие и развлекательные.

Существует порядка тысячи разновидностей покемонов, и любителям серии игр и мультфильмов нелегко держать в памяти облик и названия их всех.

Данный проект позволит сделать первый шаг для создания удобного средства распознавания покемонов по изображению, а автору проекта позволит на практике изучить алгоритмы классификации изображений и разработки чат-ботов для социальных сетей и средств обмена сообщениями.

Используемое техническое оборудование

Одним из критериев для выбора алгоритма машинного обучения и параметров построения модели явилась вычислительная мощность компьютера, находящегося в распоряжении автора проекта:

1. Процессор Intel Core i5-8600K 3,6 ГГц.
2. Размер оперативной памяти 16 Гб.
3. Операционная система Windows 10, 64-разрядная.
4. Дискретная видеокарта - отсутствует.

Этапы разработки

I. Выбор и установка программных и технических средств для создания ИИ

В качестве языка программирования для разработки был выбран Python 3, из-за частоты его использования в научной среде, в том числе и при написании алгоритмов для искусственного интеллекта, а также по той причине, что автор работы изучает данный язык программирования.

Для осуществления математических расчетов использовалась библиотека scikit-learn, содержащая огромное количество алгоритмов анализа и обработки данных, построения математических моделей.

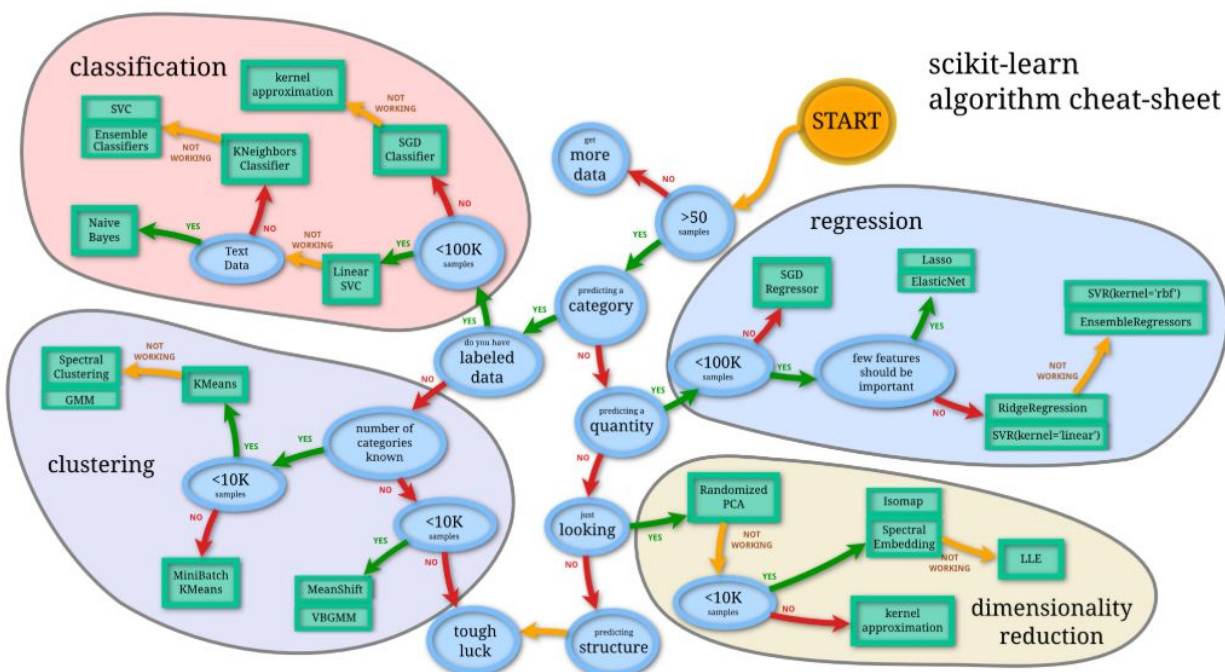
Обработка изображений происходила с помощью библиотеки scikit-image.

Для получения и отправки сообщений в Telegram использовалась библиотека pyTelegramBotAPI.

Для удобства использования Python в среде Windows, использовался дистрибьютив WinPython (<http://winpython.github.io/>), содержащий множество встроенных научных библиотек. Этот дистрибьютив не требует установки, для начала работы достаточно просто его скачать.

II. Разработка алгоритма обучения искусственного интеллекта

Используя путеводитель по популярным алгоритмам ИИ (за исключением нейронных сетей) [3] и примеры из публикаций по распознаванию образов [8, 10, 11], было решено, что алгоритм классификации SVM (иногда называемый SVC) подходит для решения поставленной задачи.

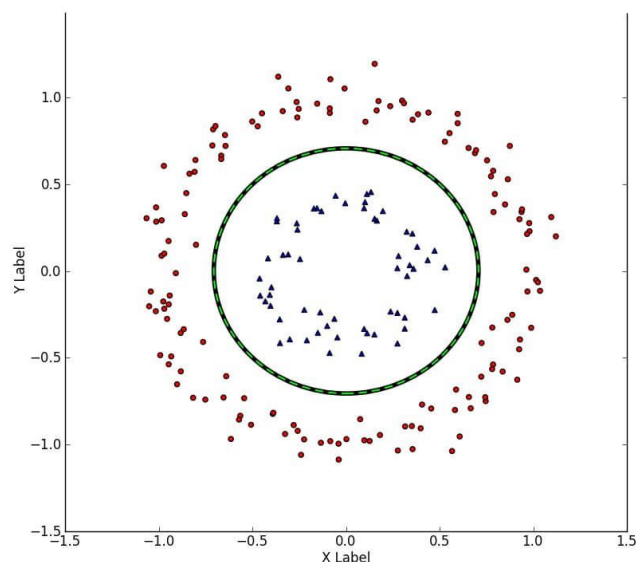
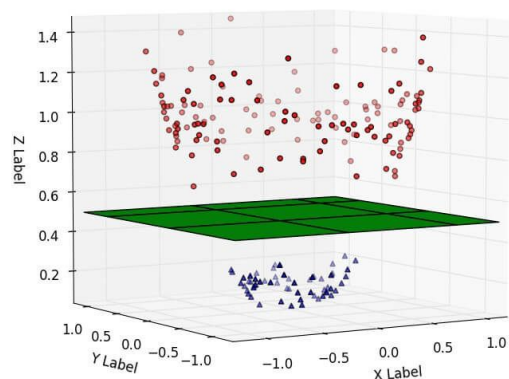


Исходными данными для алгоритма SVM являются многомерные числовые вектора, описывающие характеристики исследуемых объектов. Для каждого такого объекта указывается принадлежность к тому или иному классу. Алгоритм SVM располагает точки, на которые указывают вектора, в многомерном пространстве и вычисляет уравнения поверхностей которые разделяли бы точки так, чтобы точки одного класса оказывались бы рядом друг с другом, а точки разных классов оказались бы разделены. Уравнения разделяющих поверхностей и являются моделью классификации.

Чтобы выяснить, к какому классу относится новый неизвестный объект, представленный в виде вектора, алгоритм проверяет, в какой области между разделяющими поверхностями находится новая точка и выдает свой прогноз по классу исследуемого объекта.

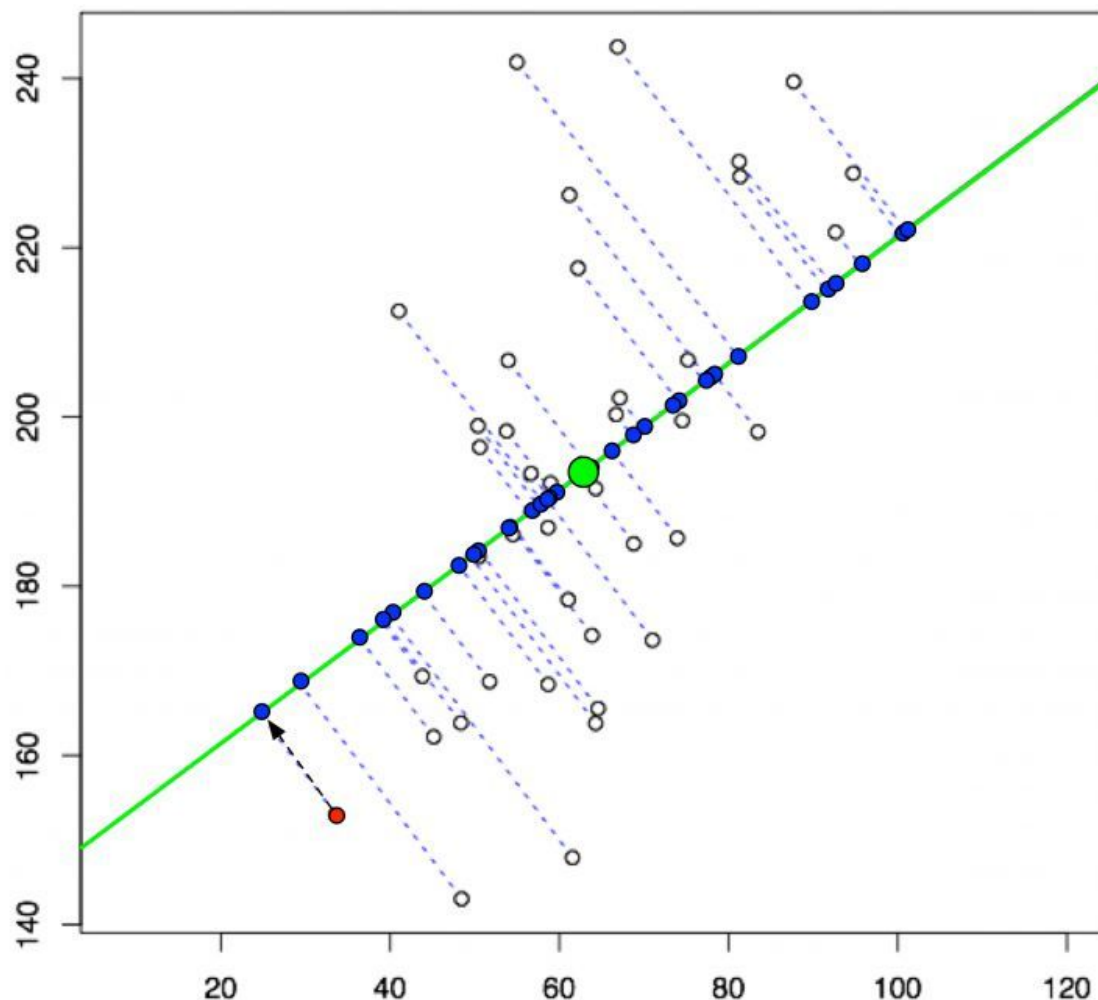
Пример разделения объектов классификатором на две категории:

- 1) по трем характеристикам объекта (трехмерный вектор)
- 2) по двум характеристикам (двухмерный вектор)



В нашем случае характеристики каждого изображения описываются вектором, имеющим столько же измерений, сколько точек содержится в изображении. Для картинки формата 256 x 256 пикселей требуется построение пространства, содержащего 65 536 измерений. По понятным причинам проиллюстрировать это пространство автор не решается. Для уменьшения размерности векторов использовался алгоритм PCA, позволяющий уменьшать количество измерений с минимально возможными потерями полезной информации. В нашем алгоритме отбрасываются не более 20% данных, при этом размерность вектора уменьшается до 18 - 82 измерений в зависимости от количества видов покемонов и количества изображений данного вида.

Алгоритм PCA работает следующим образом:

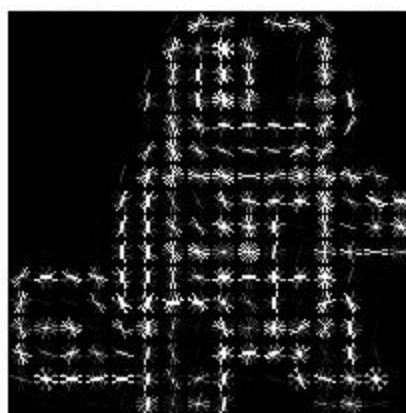
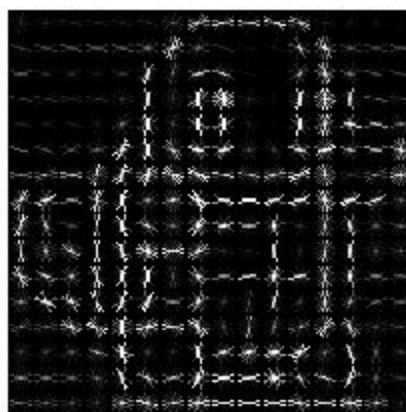


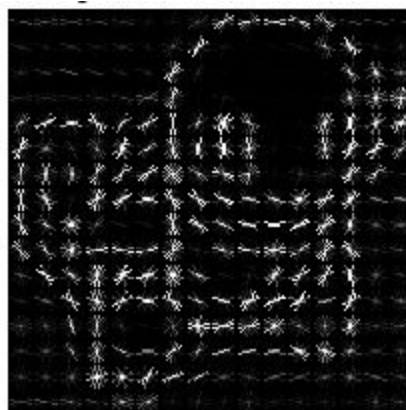
Белые точки показывают исходные вектора в двухмерном пространстве, имеющие, естественно, два измерения. Алгоритм PCA находит такое новое измерение (зеленая прямая), на которую можно спроецировать исходные точки с наименьшим искажением их расположения (синие точки). Теперь каждая точка может быть описана вектором из одного измерения, указывающим положение точки на зеленой прямой. Таким образом при каждом применении алгоритм PCA позволяет уменьшить количество измерений на одно. Количество погрешности, которое вносит алгоритм, определяется расстоянием от исходных точек до новой оси (черная стрелка от красной точки до зеленой линии), и поэтому можно заранее задать максимальную допустимую погрешность, вносимую алгоритмом в данные.

Для работы с алгоритмами, описанными выше, необходимо привести изображения в форму многомерного вектора.

Сначала изображения приводятся к единому размеру путем вписывания в квадрат 256 x 256 точек. Далее они обрабатываются методом HOG (Histogram of Oriented Gradients) [6] - гистограмма направленных градиентов - изображение разбивается на блоки, сравнивается яркость соседних блоков и формируется массив векторов, показывающих направление и величину изменения яркости между блоками. Это позволяет выделить наиболее существенные части изображения - контуры где происходит изменение цвета, а более однородный фон убрать из изображения.

Результат работы алгоритма HOG:





Таким образом, обучение модели происходит в следующем порядке:

1. Обучающее изображение вписывается в квадрат 256 x 256 пикселей
2. Алгоритм HOG выделяет контуры и преобразует изображение в многомерный вектор
3. Алгоритм PCA уменьшает размерность вектора, отбрасывая малозначимые данные
4. Алгоритм SVM вычисляет уравнения поверхностей, разделяющих векторы разных типов покемонов

В приложении 2 приведен текст программы teach.py, реализующей данный процесс обучения.

Пример вывода на консоль при запуске программы teach.py:

```
Запуск обучения
6.77 сек занял сбор и преобразование образцов изображений
0.37 сек занял расчет количества главных компонентов
Количество главных компонентов векторов для 80% точности: 82
Идет процесс проецирования полных векторов изображений на оси
главных компонентов
0.01 сек занял расчет проекций главных компонентов векторов
Найдены наилучшие параметры модели:
SVC(C=1, cache_size=200, class_weight='balanced', coef0=0.0,
    decision_function_shape='ovr', degree=3,
    gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=True,
    random_state=None,
```

```
shrinking=True, tol=0.001, verbose=False)
```

0.91 сек заняло обучение модели

Обучение завершено. Всего потребовалось 8.15 сек

Обученная модель сохраняется на диск в виде файла, что позволяет сохранить результаты вычислений и позволяет не проводить обучение заново каждый раз при запуске бота.

III. Тестирование искусственного интеллекта

Одновременно с обучением модели осуществляется проверка качества ее работы на тестовых образцах, для чего из обучающих образцов отбирается 20% изображений, которые не используются для обучения, а используются только для пробного распознавания и по результатам такого распознавания вычисляется качество модели.

Пример вывода программы teach.py в процессе проверки модели:

Запуск тестирования качества распознавания на тестовых образцах

0.0 сек заняло тестирование модели

	precision	recall	f1-score	support
Бульбазавр	1.00	1.00	1.00	1
Генгар	0.67	0.67	0.67	6
Джигглипафф	1.00	0.80	0.89	5
Дигглет	1.00	0.75	0.86	4
Дитто	1.00	1.00	1.00	5
Пикачу@run	1.00	0.50	0.67	2
Пикачу@stand	0.44	1.00	0.62	4
Сквиртл	1.00	1.00	1.00	5
Тангела	1.00	1.00	1.00	6
Чармандер	1.00	0.50	0.67	4
Черизард	1.00	1.00	1.00	2
micro avg	0.84	0.84	0.84	44
macro avg	0.92	0.84	0.85	44
weighted avg	0.90	0.84	0.85	44

```
[ [1 0 0 0 0 0 0 0 0 0 0 0]
  [0 4 0 0 0 0 2 0 0 0 0 0]
  [0 0 4 0 0 0 1 0 0 0 0 0]
  [0 1 0 3 0 0 0 0 0 0 0 0]
  [0 0 0 0 5 0 0 0 0 0 0 0]
  [0 0 0 0 0 1 1 0 0 0 0 0]
  [0 0 0 0 0 0 4 0 0 0 0 0]
  [0 0 0 0 0 0 0 5 0 0 0 0]
  [0 0 0 0 0 0 0 0 6 0 0 0]
  [0 1 0 0 0 0 1 0 0 2 0 0]
  [0 0 0 0 0 0 0 0 0 0 2 0]]
```

Таким образом, взвешенная точность определения тестовых образцов составила 90%

IV. Использование алгоритма идентификации покемонов

Определение изображений покемонов реализовано в программе `check_image.py` следующим образом:

1. С диска загружается ранее рассчитанная модель классификации
2. Изображение неизвестного покемона приводится в векторный вид аналогично изображениям образцов
3. Выполняется функция классификатора `predict_proba`, который определяет расположение вектора определяемого изображения относительно поверхностей, разделяющих классы
4. Функция вычисляет вероятность принадлежности изображения к каждому типу покемонов
5. Этот алгоритм повторяется для зеркально отраженного изображения
6. Отбираются типы покемонов, к которым с наибольшей вероятностью принадлежит изображение и выводятся пользователю

Текст программы приведен в приложении 3.

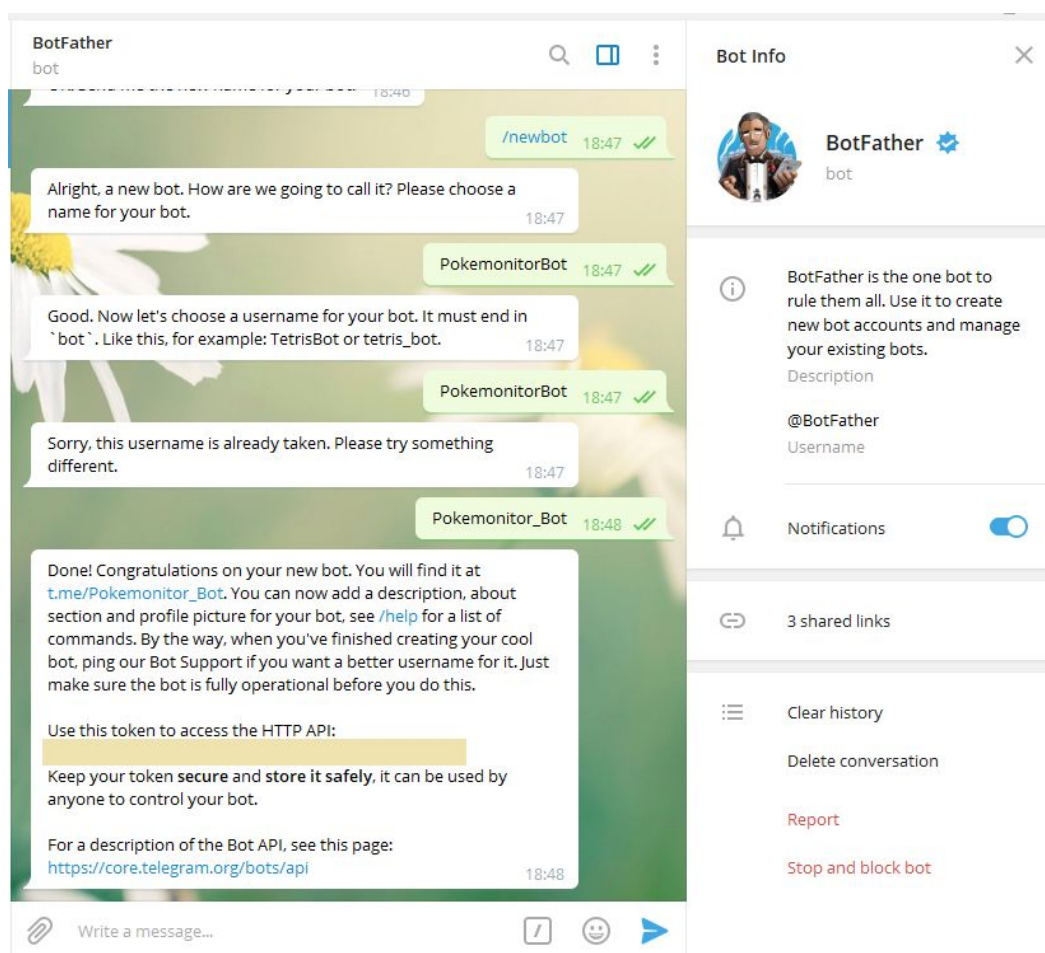
V. Разработка чат-бота

Для простого и свободного доступа к разрабатываемому искусственному интеллекту было решено не создавать собственный интернет сайт или мобильное приложение, а создать и подключить бота к популярному сервису обмена сообщениями Telegram. Telegram был выбран за простоту создания и легкость подключения бота.

Для обеспечения возможности подключения бота к Telegram, чтобы пользователи могли его находить и обмениваться сообщениями, был получен идентификационный ключ по инструкции [2]:

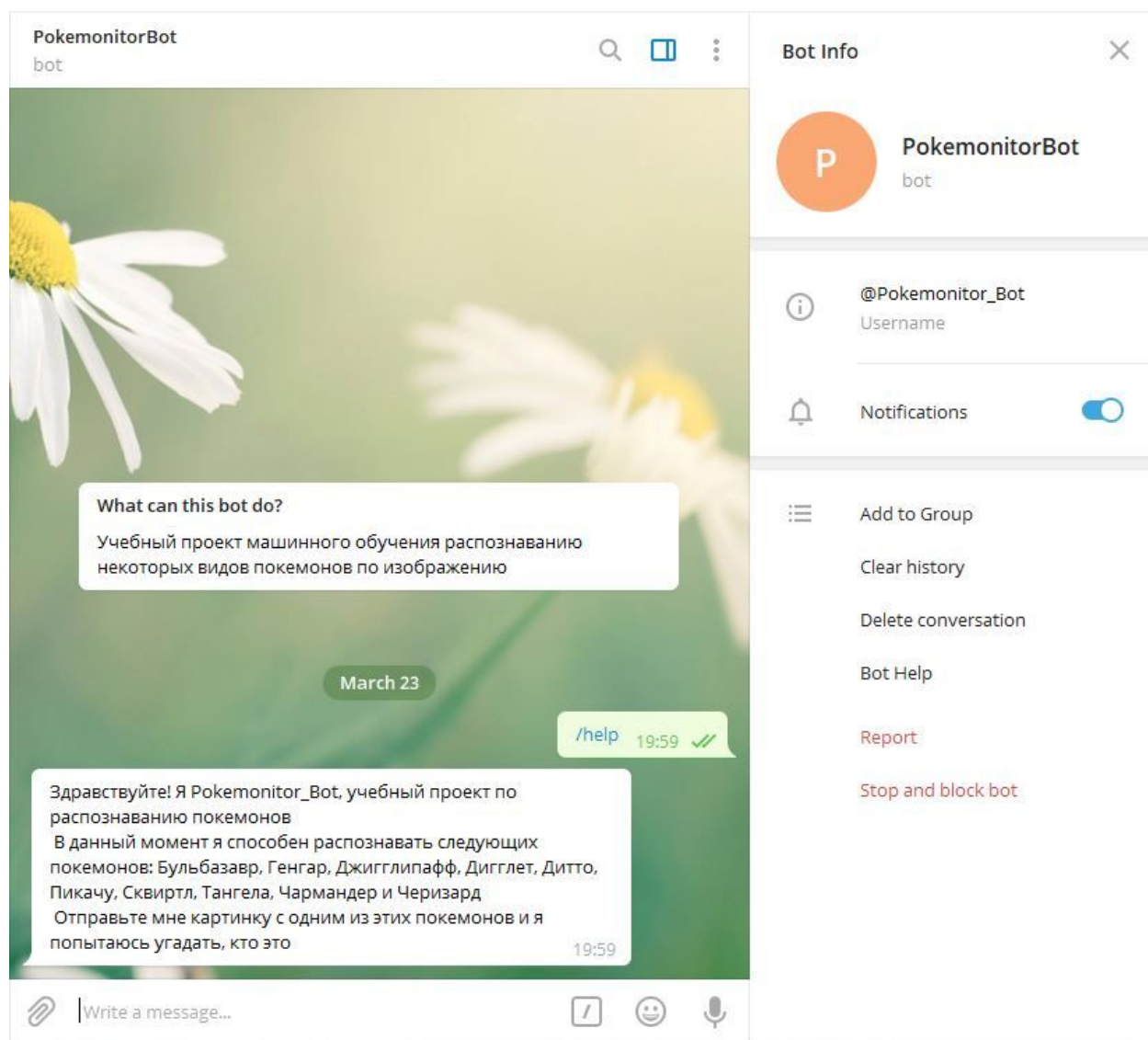
В Telegram боту @BotFather была дана команда /newbot для регистрации нового бота, а затем были придуманы имя и логин для своего бота.

Был зарегистрирован бот с именем Pokemonitor_Bot и получен секретный ключ для его аутентификации.



Для разработки программного кода бота использовалась бесплатная библиотека pyTelegramBotAPI [5].

При начале общения пользователей с ботом, он представляется и сообщает порядок работы.


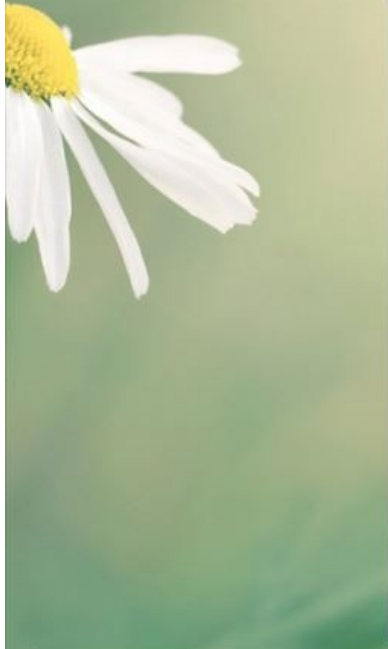


Когда бот получает от пользователя изображение, он передает его программе `check_image.py`, которая возвращает результат классификации.

Результат работы бота с искусственным интеллектом:

PokemonitorBot
bot

Пикачу, Сквиртл, Тангела, Чармандер и Черизард
Отправьте мне картинку с одним из этих покемонов и я попытаюсь угадать, кто это 19:59





Sèance (no i'm not obsessed with tua)
Photo
спасибо, получил! 20:03


Картинка похожа на Пикачу@stand(29.02%) 20:03


Write a message...


Bot Info

 **PokemonitorBot**
bot

 **@Pokemonitor_Bot**
Username

 Notifications ☒

 1 photo

 Add to Group

Clear history

Delete conversation

Bot Help


Report


Stop and block bot


PokemonitorBot
bot


Search, Add, More


Bot Info X

 **PokemonitorBot**
bot

 **@Pokemonitor_Bot**
Username

 Notifications ☒

 2 photos

 Add to Group


Clear history

Delete conversation

Bot Help


[Report](#)

[Stop and block bot](#)


 **Séance (no i'm not obsessed with tua)**
Photo

спасибо, получил! 20:03

Картинка похожа на Пикачу@stand(29.02%) 20:03







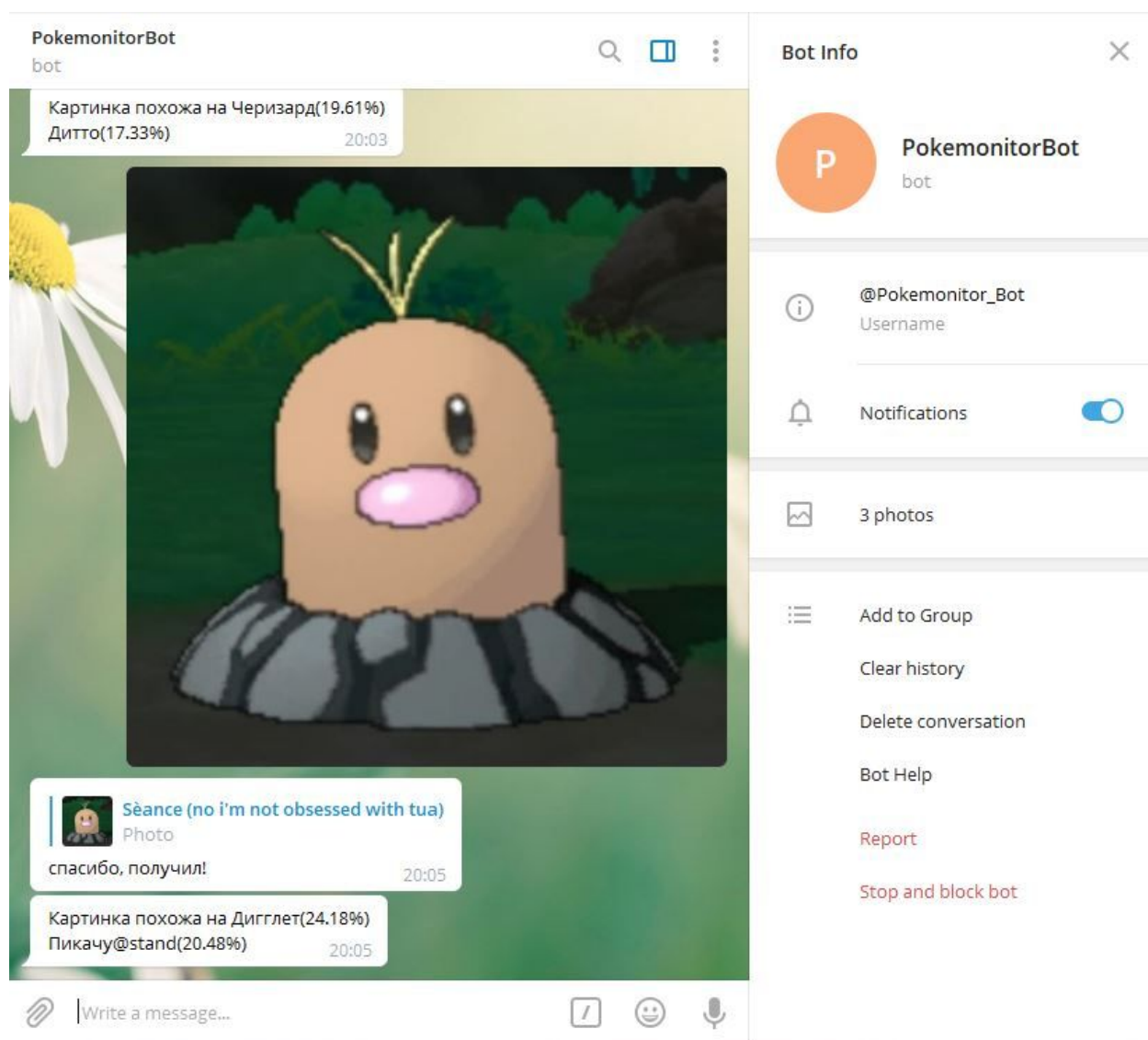
Made With VivaVideo

 **Séance (no i'm not obsessed with tua)**
Photo

спасибо, получил! 20:03

Картинка похожа на Черизард(19.61%)
Дитто(17.33%) 20:03

 Write a message...   



Результат

Были разработаны следующие программные библиотеки:

- ❖ bot.py - непосредственно бот для Telegram, осуществляющий диалог с пользователем
- ❖ check_image.py - классифицирует полученное изображение
- ❖ image_modify.py - библиотека, приводящая изображения к единому размеру и формату, а также выявляющая на них контуры объектов и преобразующая их в векторное представление.
- ❖ process_images.py - подготавливает образцы обучающих изображений для тренировки модели
- ❖ teach.py - обучает модель ИИ классифицировать изображения на подготовленных образцах

Файловая система проекта:

- ❖ `trained_model` - папка, содержащая обученную модель и справочные таблицы
 - `pokemon_model.names` - справочник соответствия номера категории изображения и названия покемона
 - `pokemon_model.pca` - хранение параметров метода pca (метод главных компонент, сокращает число измерений вектора)
 - `pokemon_model.pkl` - хранения параметров метода svm, он же svc (метод опорных векторов, находит уравнения поверхностей, разделяющих категории)
- ❖ `pokemon` - необработанные изображения для обучения модели
- ❖ `pokemon_processed` - изображения, приведенные к единому виду и формату для обучения модели
- ❖ `bot_downloaded_images` - изображения, которые бот получает от пользователя

В ходе работы было осуществлено следующее:

1. Собраны и при помощи созданной программы *process_images.py* автоматически обработаны образцы изображений персонажей Pokemon для приведения их к единому размеру и типу, наиболее подходящему для обучения.
2. Создана программа *teach.py* для создания модели классификации изображений по собранным образцам на языке Python с использованием библиотеки `scikit-learn`. Изображения обрабатывались методом HOG (гистограмма направленных градиентов) для выявления основных контуров объектов и представлялись в виде многомерного вектора, который при помощи PCA (метода главных компонент) упрощался, оставляя только существенные для отбора размерности, и, используя SVC (классификацию методом опорных векторов), определялись параметры модели для использования при распознавании.
3. Используя программу *teach.py* создания модели, была обучена модель классификации покемонов на собранных образцах, качество работы которой было проверено на случайно отобранных классифицированных образцах.
4. Разработан алгоритм *check_image.py* классификации изображения по обученной модели. Подобно алгоритму обучения, алгоритм классификации обрабатывает проверяемое изображения, приводя к единому размеру и типу, преобразует его в многомерный вектор, уменьшает размерность вектора

методом PCA, при помощи SVC находит степень схожести вектора с классами, имеющимися в обученной ранее модели.

5. Разработан алгоритм *bot.py* для получения изображений от пользователей через мессенджер Telegram, классификации их алгоритмом *check_image.py*, передачи пользователям ответного сообщения с указанием классов покемонов, наиболее схожих с полученным изображением.
6. Разработанный бот подключен к telegram и проведена проверка его работоспособности.

Выводы

Данная работа на практике показала, что разработка алгоритмов искусственного интеллекта доступна даже для школьников благодаря бесплатным и доступным программным средствам и библиотекам, реализующим необходимые математические функции. Обучение простой модели классификации и распознавания изображений занимает на обычном бытовом компьютере несколько минут, а распознавание новых изображений по уже обученной модели - доли секунды.

Разработанный бот для мессенджера Telegram, оснащенный искусственным интеллектом, способен классифицировать обработанные рисованные изображения персонажей комиксов, мультфильмов и игр Pokemon с точностью до 90%, что достаточно для развлекательных и учебных целей.

При распознавании необработанных изображений, где изображение персонажа занимает менее 85% площади всего изображения, где изображено несколько персонажей или персонажи находятся в положении или позициях, отсутствующих в обучающих образцах, точность распознавания резко уменьшается до порядка 20-40%.

Дальнейшее развитие

Работа над проектом может быть продолжена в следующих направлениях:

1. Улучшение качества распознавания образов:
 - Увеличение количества распознаваемых видов покемонов

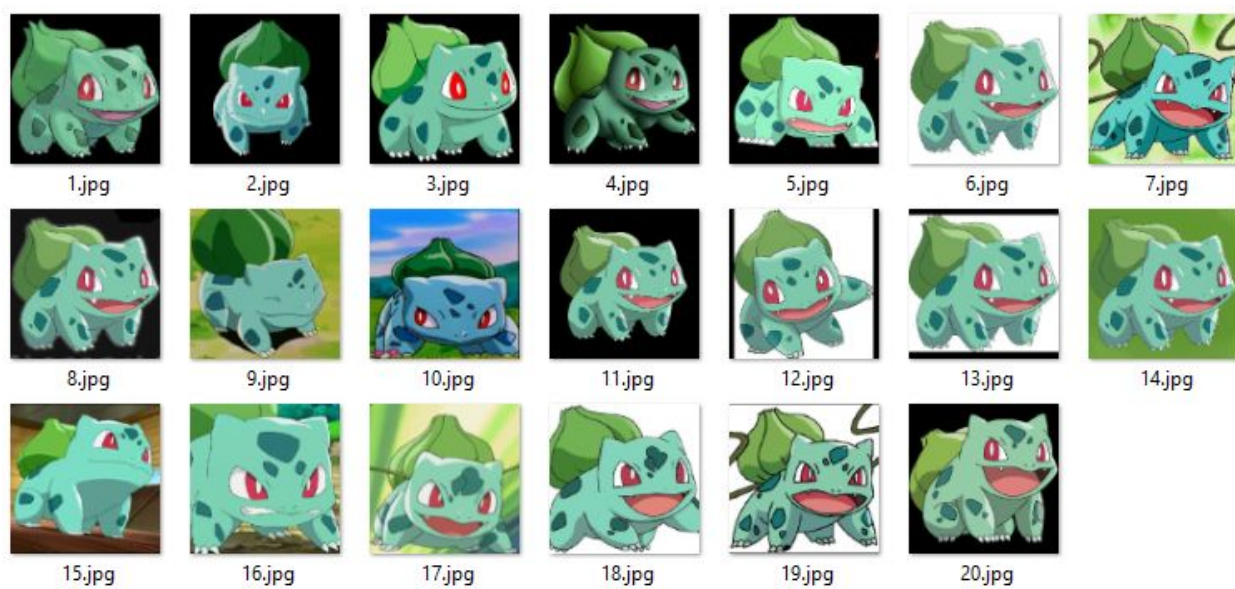
- Увеличение количества обучающих образцов изображений для каждого вида покемонов.
 - Сравнение различных алгоритмов классификации и настраиваемых параметров их работы для выбора наилучшего из них.
 - Применение нейронных сетей в различных вариантах: готовых моделей на нейронных сетях или с дообучением готовых моделей собственными образцами.
2. Расширение возможностей:
- Добавление способности самообучения бота на основе присылаемых ему изображений.
 - Вывод пользователям дополнительной справочной информации по распознанным покемонам.
 - Распознавание на видео встречающихся покемонов.
 - Поиск на видео кадров с заданным видом покемона.

Исходный код проекта выложен в общий свободный доступ на <https://github.com/alice-kuskova/pokemonibot>

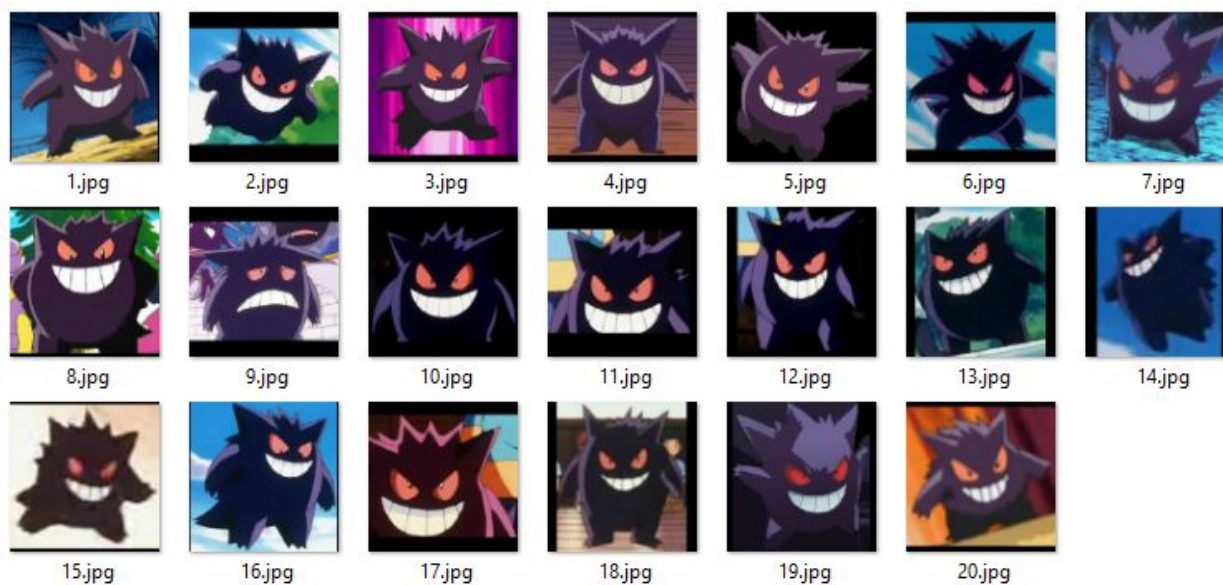
Приложения

Приложение 1. Примеры обучающих изображений

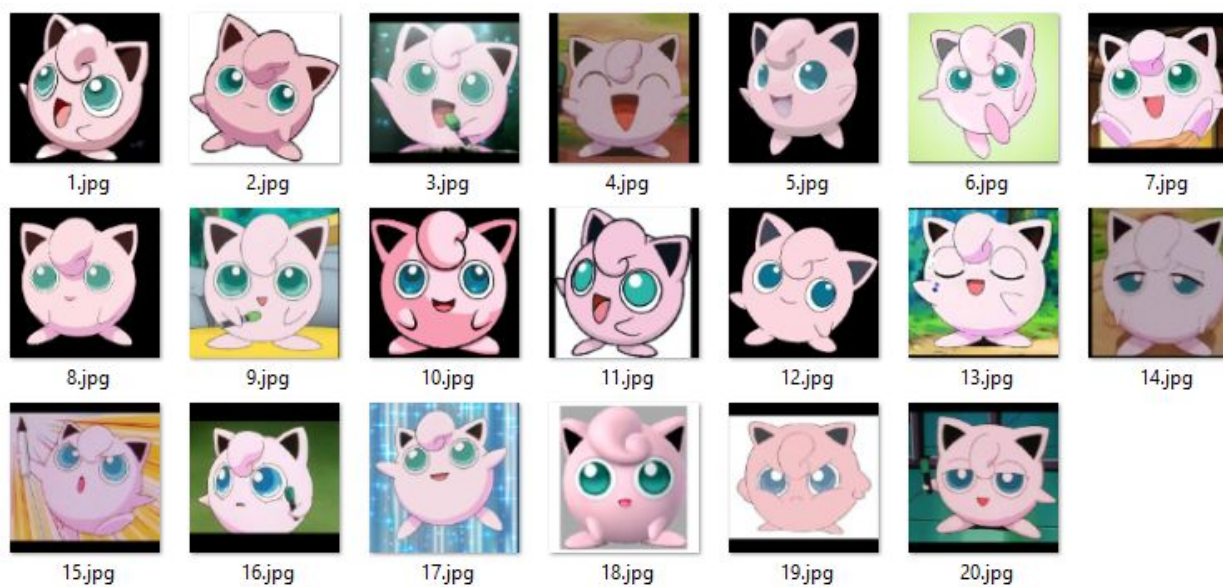
Бульбазавр



Генгар



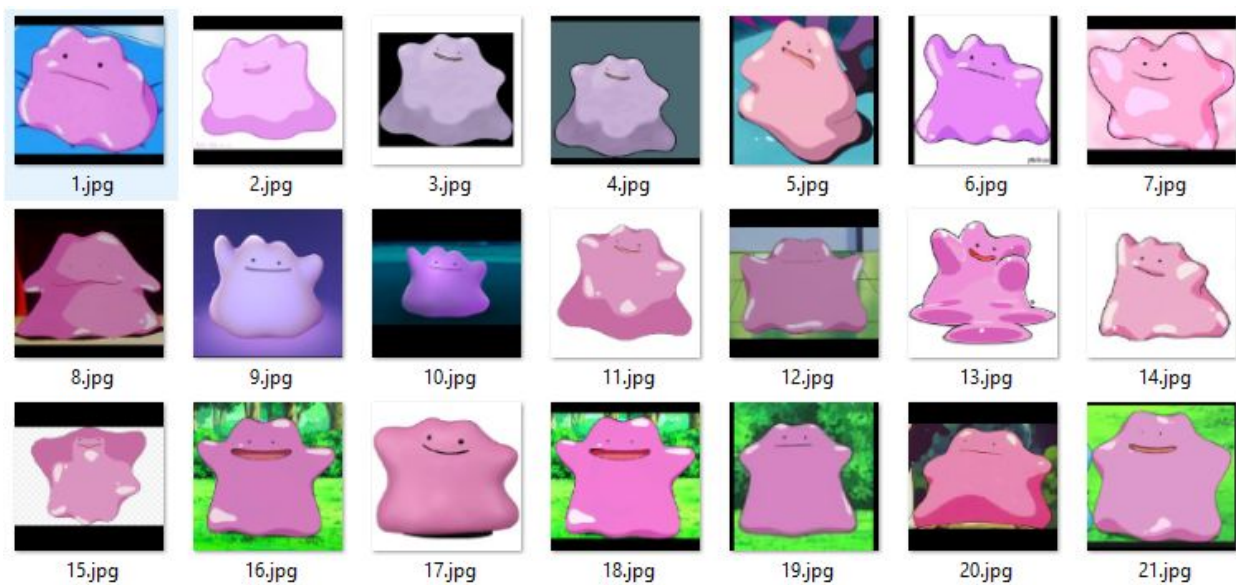
Джигглипафф



Дигглет



Дитто



Пикачу



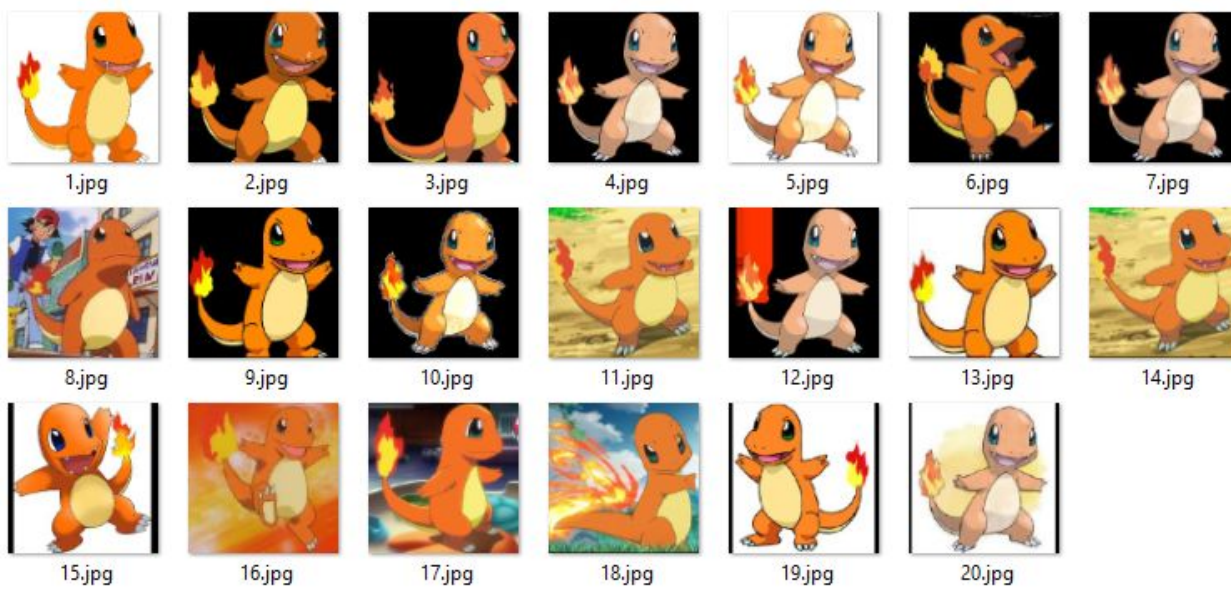
Сквиртл



Тангела



Чармандер



Черизард



1.jpg



2.jpg



3.jpg



4.jpg



5.jpg



6.jpg



7.jpg



8.jpg



9.jpg



10.jpg



11.jpg



12.jpg



13.jpg



14.jpg



15.jpg



16.jpg



17.jpg



18.jpg




19.jpg



20.jpg

Список литературы

1. 3.4. Model persistence // scikit [Электронный ресурс]. URL: https://scikit-learn.org/0.20/modules/model_persistence.html.
2. Bots: An introduction for developers // Telegram APIs [Электронный ресурс]. URL: <https://core.telegram.org/bots#6-botfather>.
3. Choosing the right estimator // scikit [Электронный ресурс]. URL: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html.
4. Comparing anomaly detection algorithms for outlier detection on toy datasets // scikit [Электронный ресурс]. URL: https://scikit-learn.org/0.20/auto_examples/plot_anomaly_comparison.html#sphx-gl-r-auto-examples-plot-anomaly-comparison-py.
5. Eternnoir eternnoir/pyTelegramBotAPI // GitHub [Электронный ресурс]. URL: <https://github.com/eternnoir/pyTelegramBotAPI>.
6. Histogram of Oriented Gradients // Histogram of Oriented Gradients - skimage v0.15.dev0 docs [Электронный ресурс]. URL: https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html#sphx-gl-r-auto-examples-features-detection-plot-hog-py.
7. Kaplan A., Haenlein M. Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence // Business Horizons. 2019. № 1 (62). С. 15–25.
8. Petukhov D., Petukhov D. Pokémon Recognition // Medium [Электронный ресурс]. URL: <https://medium.com/@dimart/pokémon-recognition-d3ad5cad61e>.
9. Principal component analysis // Wikipedia [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Principal_component_analysis.

- 
10. sklearn.svm.SVC — scikit-learn 0.20.3 documentation // scikit [Электронный ресурс]. URL:
<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
 11. Введение в машинное обучение с помощью Python и Scikit-Learn // / Блог компании MLClass / Хабр [Электронный ресурс]. URL:
<https://habr.com/ru/company/mlclass/blog/247751/>.
 12. Ильясов Ф. Н. Разум искусственный и естественный // Известия АН Туркменской ССР, серия общественных наук. 1986. № 6. С. 46—54.
 13. Области применения искусственного интеллекта // Mentamore [Электронный ресурс]. URL:
<https://mentamore.com/covremennye-texnologii/oblasti-primeneniya-iskusstvenno-go-intellekta.html>.