УДК 004

Разработка модели распознавания оружия с помощью нейросети

Звайгзне Алексей Юрьевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Целью исследования является разработка эффективной модели распознавания оружия на основе нейросетевых технологий. Для этого были применены методы машинного обучения, такие как сверточные нейронные сети. Программа написана на языке программирования Python в онлайн-среде программирования Google Colab. В результате исследования была получена модель, которая с высокой точностью распознает различные типы оружия, что может быть полезно в обеспечении безопасности на практике.
Ключевые слова: модель распознавания оружия, свёрточная нейронная сеть, Python, Google Colab

Development of a weapon recognition model using a neural network

Zvaigzne Alexey Yurievich
Sholom-Aleichem Priamursky State University
Student

Abstract
The aim of the study is to develop an effective model of weapon recognition based on neural network technologies. To do this, machine learning methods such as convolutional neural networks were used. The program is written in the Python programming language in the Google Colab online programming environment. As a result of the study, a model was obtained that recognizes various types of weapons with high accuracy, which can be useful in ensuring security in practice.
Keywords: weapon recognition model, convolutional neural network, Python, Google Colab

1 Введение

1.1 Актуальность

Актуальность темы исследования состоит в том, что угроза террористических актов и нападений в различных местах массового скопления людей становится все более актуальной. Одним из наиболее опасных сценариев является нападение с использованием огнестрельного оружия. В связи с этим, создание системы распознавания огнестрельного оружия в руках человека становится важной задачей. Такая система может быть использована в различных местах массового скопления людей, таких как аэропорты, стадионы, концертные площадки, торговые центры и т.д., для обеспечения

безопасности и предотвращения террористических актов. При этом, создание такой системы позволит ускорить и улучшить работу служб безопасности, а также повысить уровень безопасности в обществе в целом. Также стоит отметить, что существующие системы безопасности, такие как металлодетекторы и сканеры, могут быть обойдены, если оружие находится в скрытом месте. В этом случае, система распознавания огнестрельного оружия в руках человека становится более эффективным средством обеспечения безопасности. Таким образом, создание системы распознавания огнестрельного оружия в руках человека является важной задачей в области обеспечения безопасности и защиты жизни людей.

1.2 Обзор исследований

А.А. Ярыгин обсуждает использование машинного обучения с подкреплением интеллектуальных агентов в задачах принятия решений. В статье рассматриваются технические основы обучения с подкреплением, исследуется интеграция обучения с подкреплением и глубокого обучения, а также обсуждается применение этих методов в различных областях, таких как робототехника и финансы [1].

П.М. Горбунов, Ю.А. Мацкевич и А.В. Чубарь описали автоматизацию выбора модели машинного обучения с использованием методов машинного обучения; обсуждали, как этот процесс можно оптимизировать и сделать более эффективным, чтобы сократить время и затраты, необходимые для выбора лучшей модели для данной задачи [2].

С. А. Сохина, С. А. Немченко раскрыли понятие машинного обучения, а также раскрыли его типы и виды; объяснили принципы функционирования различных видов машинного обучения [3].

В.Е. Садовников рассмотрел алгоритмы искусственного интеллекта в качестве основополагающего функции обработки изображений с видеонаблюдения [4].

А.В. Юдин и С. Рокхая изучили и продемонстрировали в практике различные методы искусственного интеллекта и машинного обучения, используемые для решения задач распознавания образов и интеллектуальной обработки изображений [5].

П. Пико-Валенсии, О. Винуэса и Х.А. Ольгадо-Терриза рассмотрели, как использование машинного обучения может стать доступнее для тех, кто не является профессионалом в данной области. В своих исследованиях они изучили различные методы, которые помогают упростить процесс создания и использования предиктивных моделей для пользователей без технического образования. [6].

К. Чаухан, С. Джани, Д. Таккар и другие посвятили свой труд автоматизированному машинному обучению (AutoML) и тому, как оно становится новой волной машинного обучения; обсудили важность AutoML для упрощения процесса машинного обучения и обеспечения его доступности для неспециалистов [7].

Р. Дивья и Д. П. Джей использовали методы квантовых вычислений в машинном обучении, а также предложили обзор методов оптимизации алгоритмов машинного обучения с использованием квантовых вычислений [8].

Р.М.Д. Д'Суза, С.Р. Дипти и К.А. Шримп применили алгоритм TensorFlow для обнаружения, классификации и резки фруктов и овощей [9].

Д. Баджпай и Л. ХЕ рассмотрели методы создания и обработки данных, а также их использование для обучения модели машинного обучения на примере игры Google T-Rex [10].

В своей книге Д.Р. Стюарт описал широкий круг тем, связанных с ИИ, включая интеллектуальных агентов, методы решения проблем, представление знаний, обработку естественного языка, обучение, восприятие, робототехнику [11].

Я. Гудфеллоу, Б. Иошуа и А. Курвилль в своей книге предоставили информацию по глубокому обучению, подобласти машинного обучения [12].

Р.С. Саттон и Э.Г. Барто в своей книге описали область обучения с подкреплением, представляющего собой тип машинного обучения, в котором агент обучается на основе своего взаимодействия с динамическим средами [13].

В.В. Круглов в своем учебном пособии изложил теорию искусственных нейронных сетей, аппаратуру нечеткой логики и так называемых гибридных нейронных сетей [14].

А.Н. Горбань обозревает методы решения задач с помощью нейронных сетей, реализованных на персональных компьютерах [15].

Е.С. Горбунов использовал нейронные сети для решения плохо формализованных задач в интеллектуальном анализе данных [16].

А.В. Близнюк рассмотрел методы обработки и хранения больших объемов видеоинформации с помощью облачных вычислений [17].

С.А. Эль-Хатиб провёл исследования о повышение точности и скорости решения задачи сегментации изображений с применением бионических моделей роевого интеллекта [18].

А.В. Новиков разработал новый метод кластерного анализа на базе осцилляторной нейронной сети для преодоления недостатков существующих методов и обеспечения параллельной реализации [19].

Э.Е. Тихонов в своей диссертации произвел сравнительную оценку качества прогнозирования классических и нейросетях методов прогнозирования [20].

1.3 Цель исследования

Целью данной работы является разработка нейронной сети для распознавания огнестрельного оружия в руках человека с использованием языка программирования Python и среды разработки Google Colab.

2 Материалы и методы

Для работы понадобится онлайн среда программирования Google Colab [21].

3 Результаты и обсуждение

Для начала работы нужно импортировать необходимые библиотеки (рис.1).

```
import os
import numpy as np
import cv2
from sklearn.utils import shuffle
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from google.colab import files
from keras.models import load_model
```

Рисунок 1. Импорт модулей

Функция load_image() загружает изображение из файла, изменяет его размер, преобразует в массив numpy, нормализует значения пикселей и возвращает его в виде массива img (рис.2).

```
# Функция для загрузки изображения из файла и преобразования его к нужному размеру'''
def load_image(filename, img_size):
    img = cv2.imread(filename)
    img = cv2.resize(img, img_size)
    img = np.expand_dims(img, axis=0)
    img = img.astype('float32') / 255
    return img
```

Рисунок 2. Функция для загрузки изображения из файла и преобразования его к нужному размеру

Переходим к инициализации необходимых переменных для загрузки и обработки изображений и подготовки их к обучению нейронной сети. Для этого нужно заранее загрузить в Google Диск папки с изображениями. Переменная with_weapon_dir содержит путь к папке с изображениями, на которых присутствует оружие. Переменная without_weapon_dir содержит путь к папке с изображениями, на которых отсутствует оружие. Переменная img_size определяет размер изображений, которые будут подаваться на вход нейронной сети. Переменные используются далее в коде для загрузки и предобработки изображений, а также для обучения нейронной сети на основе этих изображений (рис.3).

```
# Путь к папке с изображениями с оружием
with_weapon_dir = '/content/drive/MyDrive/with_weapon'
# Путь к папке с изображениями без оружия
without_weapon_dir = '/content/drive/MyDrive/without_weapon'
# Размер изображений, которые будут подаваться на вход нейронной сети
img_size = (224, 224)
```

Рисунок 3. Загрузка и корректировка датасета

Загружаем списки файлов в соответствующих папках и готовим их для дальнейшей обработки. Первая строка создает список файлов в папке with_weapon, используя функцию os.listdir(). Эта функция возвращает список всех файлов и папок, находящихся в указанной директории, в данном случае – в папке с изображениями, на которых присутствует оружие. Вторая строка создает список файлов в папке without_weapon, также используя функцию os.listdir(). Эта функция возвращает список всех файлов и папок, находящихся в указанной директории, в данном случае – в папке с изображениями, на которых отсутствует оружие (рис.4).

```
# Список файлов в папке with_weapon
with_weapon_files = os.listdir(with_weapon_dir)
# Список файлов в папке without_weapon
without_weapon_files = os.listdir(without_weapon_dir)
```

Рисунок 4. Подготовка данных

Создает два пустых списка images и labels, которые будут использоваться для хранения изображений и соответствующих меток классов. Список images будет содержать загруженные изображения, которые будут использоваться для обучения нейронной сети. Список labels будет содержать соответствующие метки классов, которые указывают, присутствует ли на изображении оружие или нет (рис.5).

```
# Список изображений и соответствующих меток классов
images = []
labels = []
```

Рисунок 5. Создание двух пустых списка images и labels

Загружаем и перерабатываем изображения с оружием, добавляем их в список images и соответствующие метки классов в список labels (рис.6).

```
# Загружаем изображения с оружием и добавляем их в список images, а метки классов - в список labels
for file_name in with_weapon_files:
    img = cv2.imread(os.path.join(with_weapon_dir, file_name))
    img = cv2.resize(img, img_size)
    images.append(img)
    labels.append(1)
```

Рисунок 6. Форматирование данных

Загружаем и перерабатываем изображения без оружия, добавляем их в список images и соответствующие метки классов в список labels (рис.7).

```
# Загружаем изображения без оружия и добавляем их в список images, а метки классов - в список labels
for file_name in without_weapon_files:
    img = cv2.imread(os.path.join(without_weapon_dir, file_name))
    img = cv2.resize(img, img_size)
    images.append(img)
    labels.append(0)
```

Рисунок 7. Форматирование данных

Преобразуем списки images и labels в массивы numpy (рис.8)

```
# Преобразуем списки images и labels в массивы numpy
images = np.array(images)
labels = np.array(labels)
```

Рисунок 8. Подготовка датасета

Перемешиваем массивы images и labels в случайном порядке (рис.9).

```
# Перемешиваем массивы images и labels в случайном порядке
images, labels = shuffle(images, labels, random_state=42)
```

Рисунок 9. Подготовка датасета

Преобразуем список меток классов labels в категориальный формат с помощью функции to_categorical() из библиотеки keras.utils (рис.10).

```
# Преобразуем метки классов в one-hot encoding
labels = to_categorical(labels)
```

Рисунок 10. Подготовка датасета

Делим данные на обучающую и тестовую выборки. Из которых 80% - обучающая, 20% - тестовая (рис.11).

```
# Делим данные на обучающую и тестовую выборки (80% - обучающая, 20% - тестовая)
split_idx = int(len(images) * 0.8)
train_images, test_images = images[:split_idx], images[split_idx:]
train_labels, test_labels = labels[:split_idx], labels[split_idx:]
```

Рисунок 11. Подготовка данных

Нормализуем значения пикселей в диапазоне от 0 до 1 (рис.12).

```
# Нормализуем значения пикселей в диапазоне от 0 до 1
train_images = train_images.astype('float32') / 255
test_images = test_images.astype('float32') / 255
```

Рисунок 12. Подготовка датасета

Создаем модель нейронной сети. Сначала создается объект Sequential(), который представляет собой последовательность слоев нейронной сети. Затем, последовательно добавляются слои с помощью метода add(). Первый слой - Conv2D - является сверточным слоем, который выполняет операцию свертки на входном изображении. У него 32 фильтра размером 3x3, функция активации - relu, и форма входного изображения задается параметром input_shape. Затем следует слой MaxPooling2D, который уменьшает размерность данных путем выбора максимального значения из окна размером 2x2. Далее, последовательно добавляются еще три сверточных слоя с увеличением количества фильтров (64, 128, 128), и каждый раз после сверточного слоя следует слой MaxPooling2D. После последнего сверточного

слоя добавляется слой Flatten(), который преобразует многомерный тензор в одномерный вектор, чтобы передать его в полносвязный слой. Затем добавляется полносвязный слой Dense с 512 нейронами и функцией активации relu, и слой Dropout, который помогает предотвратить переобучение модели, случайно обнуляя часть входных нейронов. Наконец, добавляется последний полносвязный слой с двумя нейронами и функцией активации softmax, который определяет вероятности принадлежности изображения к каждому из двух классов (рис.13).

```
# Создаем модель нейронной сети
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(img_size[0], img_size[1], 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
```

Рисунок 13. Создание нейронной сети

Компилируем нейронную сеть. Для этого определяем функцию потерь, оптимизатор и метрики для оценки ее производительности. После компиляции нейронная сеть готова к обучению на тренировочных данных (рис.14).

```
# Компилируем модель
model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
```

Рисунок 14. Компиляция нейронной сети

Обучаем модель на обучающей выборке. Функция fit() принимает несколько аргументов: train_images - тренировочные изображения, на которых будет обучаться нейронная сеть; train_labels - метки классов для тренировочных изображений; epochs - количество эпох (полных проходов) по тренировочным данным; batch_size - количество образцов, обрабатываемых за одну итерацию алгоритма обучения; validation_data - данные для проверки производительности нейронной сети на каждой эпохе обучения. В данном случае используются тестовые изображения и метки классов В качестве метрики используется функция потерь (рис.15).

```
# Обучаем модель на обучающей выборке
model.fit(train_images, train_labels, epochs=20, batch_size=32, validation_data=(test_images, test_labels))

Epoch 1/20
18/18 [==============================] - 77s 4s/step - loss: 0.7702 - accuracy: 0.5191 - val_loss: 0.6869 - val_accuracy: 0.5145
Epoch 2/20
18/18 [==============================] - 73s 4s/step - loss: 0.7952 - accuracy: 0.5699 - val_loss: 0.7157 - val_accuracy: 0.5145
Epoch 3/20
18/18 [==============================] - 73s 4s/step - loss: 0.6544 - accuracy: 0.6570 - val_loss: 0.5734 - val_accuracy: 0.7029
Epoch 4/20
18/18 [==============================] - 81s 5s/step - loss: 0.5665 - accuracy: 0.7314 - val_loss: 0.5930 - val_accuracy: 0.7174
Epoch 5/20
18/18 [==============================] - 75s 4s/step - loss: 0.5291 - accuracy: 0.7713 - val_loss: 0.6945 - val_accuracy: 0.6884
Epoch 6/20
18/18 [==============================] - 75s 4s/step - loss: 0.5297 - accuracy: 0.7731 - val_loss: 0.8863 - val_accuracy: 0.6087
Epoch 7/20
18/18 [==============================] - 75s 4s/step - loss: 0.5051 - accuracy: 0.7604 - val_loss: 0.5202 - val_accuracy: 0.7464
Epoch 8/20
18/18 [==============================] - 75s 4s/step - loss: 0.4579 - accuracy: 0.7895 - val_loss: 1.1578 - val_accuracy: 0.6449
Epoch 9/20
18/18 [==============================] - 74s 4s/step - loss: 0.5024 - accuracy: 0.8004 - val_loss: 0.5234 - val_accuracy: 0.7609
Epoch 10/20
18/18 [==============================] - 75s 4s/step - loss: 0.4756 - accuracy: 0.8022 - val_loss: 0.5641 - val_accuracy: 0.7246
Epoch 11/20
18/18 [==============================] - 73s 4s/step - loss: 0.4134 - accuracy: 0.8330 - val_loss: 0.5345 - val_accuracy: 0.7464
```

Рисунок 15. Обучение нейросети

Сохраняем модель в файл (рис.16).

```
# Сохраняем модель в файл
model.save('weapon_detection_model.h5')
```

Рисунок 16. Сохранение модели

Загружаем сохраненную модель из файла (рис.17).

```
# Загружаем сохраненную модель из файла
model = load_model('weapon_detection_model.h5')
```

Рисунок 17. Загрузка модели

Загружаем изображение для определения наличия огнестрельного оружия и проверки работоспособности нейросети (рис.18).

```
# Загружаем изображение для определения наличия огнестрельного оружия
uploaded = files.upload()
```

Выбрать файлы 16652748...oto-50.jpg

• 1665274856_39-podacha-blud-com-p-muzhchina-s-kruzhkoi-chaya-foto-50.jpg(

Saving 1665274856_39-podacha-blud-com-p-muzhchina-s-kruzhkoi-chaya-foto

Рисунок 18. Загрузка изображений пользователем

Используем предварительно обученную нейронную сеть. Пользователь загружает изображение, код загружает его, изменяет размер и классифицирует с помощью нейронной сети. В результате выполнения программы получаем ответ на вопрос: "Есть у человека огнестрельное оружие на данном изображении?" (рис.19).

```
# Преобразуем изображение к нужному размеру и типу данных
for filename in uploaded.keys():
    img = load_image(filename, img_size)

    # Предсказываем класс изображения
    prediction = model.predict(img)

    # Выводим результат
    if prediction[0][0] > prediction[0][1]:
        print("На изображении нет огнестрельного оружия.")
    else:
        print("На изображении есть огнестрельное оружие.")
```

Рисунок 19. Проверка нейронной сети пользователем

Попробуем использовать два разных изображения для проверки. На первом будет человек с огнестрельным оружием, а на втором без него (рис.20).

Рисунок 20. Изображение человека с огнестрельным оружием

После загрузки обработки изображения получаем следующие результаты (рис.21).

```
1/1 [==============================] - 0s 379ms/step
На изображении есть огнестрельное оружие.
```

Рисунок 21. Результаты по первому изображению

Используем изображение человека без оружия (рис.22).

Рисунок 22. Изображение человека без огнестрельного оружия

Получаем следующие результаты (рис.23).

```
1/1 [==============================] - 0s 78ms/step
На изображении нет огнестрельного оружия.
```

Рисунок 23. Результаты по второму изображению

Выводы

В ходе данной работы была разработана модель распознавания оружия с помощью нейросети. Был проведен обзор литературы и анализ существующих решений, были собраны и подготовлены данные для обучения нейронной сети, выбраны и настроены алгоритмы и методы обучения, разработана и обучена нейронная сеть, произведена оценка результатов и сделаны выводы.

В результате исследования было показано, что нейронная сеть может быть эффективным инструментом для распознавания огнестрельного оружия в руках человека. Была достигнута точность распознавания на уровне 75%, что является достаточным показателем.

У данной модели есть некоторые недостатки. Например, модель может неверно распознавать огнестрельное оружие у человека, если на изображении присутствуют другие объекты, которые похожи на оружие. В дальнейшем исследовании можно рассмотреть следующие направления:

- Улучшение точности распознавания оружия на изображениях путем увеличения объема обучающей выборки и использования дополнительных методов предобработки изображений.
- Расширение функциональности системы путем добавления возможности распознавания других типов оружия, например, холодного оружия.
- Адаптация системы для использования в реальном времени на видеопотоке, что позволит более эффективно контролировать безопасность в публичных местах.

Таким образом, данное исследование является важным шагом в развитии систем безопасности и контроля, и может быть применено в различных сферах, включая военную, полицейскую и частную безопасность. Посмотреть код нейросети можно по данной ссылке [22].

Библиографический список

1. Ярыгин А. А. Актуальные вопросы машинного обучения с подкреплением интеллектуальных агентов в задачах принятия решений //В сборнике: Автоматизация: проблемы, идеи, решения сборник статей по итогам Международной научно-практической конференции. 2017. С. 62.
2. Горбунов П. М., Мацкевич Ю. А., Чубарь А. В. Машинное обучение. Автоматизация подбора модели машинного обучения //Робототехника и искусственный интеллект. 2021. С. 155-160.
3. Сохина С. А., Немченко С. А. Машинное обучение. Методы машинного обучения //Современная наука в условиях модернизационных процессов: проблемы, реалии, перспективы. 2021. С. 165-168.
4. Садовников В. Е. Возможности видеонаблюдения с последующей обработкой изображения на основе алгоритмов искусственного интеллекта //Россия молодая. 2021. С. 31521.1-31521.4.
5. Юдин А. В., Рокхая С. Различные типы обработки изображений с помощью искусственного интеллекта //Общество-наука-инновации: сборник статей Международной научно-практической. 2021. С. 8.
6. Pico-Valencia P., Vinueza-Coli O., Olga do-Teresa J. A. Approximation of predictive machine learning models based on machine learning to non-technical users //Systems and Information Sciences: Materials of ICCIS 2020. – Springer International Publishing, 2021. pp. 3-15.
7. Chauhan K. et al. Automated Machine Learning: A New Wave of Machine Learning //2nd International Conference 2020 on Innovative Mechanisms for Industrial Applications (ICIMIA). IEEE, 2020. pp. 205-212.
8. Divya R., Peter J. D. Quantum machine learning: a comprehensive review of optimization of machine learning algorithms //The Fourth International Conference on Microelectronics, Signals and Systems (ICMSS) 2021. IEEE, 2021. pp. 1-6.
9. D'Souza R. M. D., Deepthi S. R., Shri K. A. Detection Classification and Cutting of Fruits and Vegetables Using Tensorflow Algorithm //Evolutionary Computing and Mobile Sustainable Networks: Proceedings of ICECMSN 2020. Springer Singapore, 2021. С. 713-720.
10. Bajpai D., He L. Custom Dataset Creation with Tensorflow Framework and Image Processing for Google T-Rex //2020 12th International Conference on Computational Intelligence and Communication Networks (CICN). IEEE, 2020. С. 45-48.
11. Russell S. Artificial intelligence. A modern approach. M.: Williams, 2006. 1024 p.
12. Гудфеллоу Я., Иошуа Б., Курвилль А. Глубокое обучение. М.: Litres, 2022.

– 800 с.
13. Саттон Р. С., Барто Э. Г. Обучение с подкреплением. М.: БИНОМ, 2011. 432 с.
14. Круглов В. В., Дли М. И., Голунов Р. Ю. Нечеткая логика и искусственные нейронные сети. М.: Физматлит, 2001. 198 с.
15. Горбань А. Н., Россиев Д. А. Нейронные сети на персональном компьютере. Новосибирск: Новосибирский филиал Федерального государственного унитарного предприятия "Академический научно-издательский и книгораспространительский центр "Наука", 1996. 276 с.
16. Горбунова Е. С. Реализация интеллектуальной системы распознавания эмоций с применением нейронных сетей: магистерская диссертация. М.: МГТУ им. Н.Э. Баумана, 2017. 124 с.
17. Близнюк А. В. Разработка и реализация Web-сервисов для сравнения и обработки видеоизображений: диссертация: дис. ... канд. физ.-мат. наук: 05.13.11. СПб., 2010. 100 с.
18. Эль-Хатиб С. А. И. Разработка и исследование методов сегментации изображений с применением бионических моделей: дис. … канд. тех. наук: 05.13.17. Таганрог, 2017. 178 с.
19. Новиков А. В. Нелинейная динамика осцилляторных нейронных сетей в задачах кластерного анализа: дис. ... канд. тех. наук: 05.13.01. СПб, 2016. 16 с.
20. Тихонов Э.Е. Методы и алгоритмы прогнозирования экономических показателей на базе нейронных сетей и модулярной арифметики: дис. ... канд. тех. наук: 05.13.18. Ставрополь, 2003. 19 с.
21. Google Colab URL: https://colab.research.google.com
22. Код модели распознавания оружия URL: https://colab.research.google.com/drive/1uXuGpL2lrkt-_5FGJ3UKT-MWwZwmYj-E?usp=sharing