СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С развитием технологий и появлением мощных вычислительных систем нейросети стали одним из самых распространенных инструментов в области искусственного интеллекта. Они обладают уникальной способностью обучаться на основе больших объемов данных и решать разнообразные задачи, включая распознавание образов. Одной из важных задач распознавания является распознавание цифр.

Актуальность выбранной темы обусловлена повышенным спросом на системы, способные эффективно распознавать цифровые образы. Такая разработка может применяться во многих сферах, включая финансы, транспорт, здравоохранение и другие, с целью улучшения точности, производительности и доступности распознавания цифровых данных.

Целью данного проекта является разработка нейросети, которая сможет распознавать цифры с высокой точностью и будет обладать дружественным и интуитивно понятным интерфейсом.

При разработке проекта были поставлены следующие задачи: изучение принципа работы нейросети, выбор архитектуры и разработка модели нейросети, обучение и оценка модели, разработка дружественного интерфейса, интеграция нейросети в интерфейс.

1 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

1.1 Понятие нейросети и нейрона

Нейросеть — это совокупность связанных искусственных нейронов, организованных в определенную структуру. Идея работы нейросети основана на процессах происходящий в биологических нейронах человеческого мозга.

Нейрон является базовым элементом нейросети. Он имеет несколько входов и один выход. Работу нейрона можно описать следующим образом: на вход нейрона подаются некоторые сигналы, умноженные на веса (числа, определяющие важность сигнала), далее считается их сумма. Сумма проходит через функцию активации и полученное значение подается на выход, а после к следующему нейрону.

Таким образом, связанные между собой нейроны объединяются в слои, среди которых выделяют входной, скрытый и выходной слой. Такая структура связанных между собой слоев образует нейронную сеть.

1.2 Функция активации

В процессе работы нейрона стоит более подробно остановиться на этапе его активации. Функция активации — это математической функция, которая вносит нелинейность в поведении нейрона и позволяет нейросети моделировать сложные зависимости в данных. Она играет особую роль в характере работы нейросети и является одним из важнейших параметров модели.

Среди самых распространенных функций активации в нейронных сетях встречаются сигмоидная функция, гиперболический тангенс и ReLU (Rectified Linear Unit). Выбор конкретной функции активации обычно зависит от задачи и архитектуры нейросети. Важно правильно подобрать функцию активации, так как она может существенно влиять на скорость обучения и точность работы нейросети.

1.3 Архитектуры нейронных сетей

Существует множество различных архитектур нейронных сетей, каждая из которых предназначена для решения определенных задач и имеет свои уникальные особенности.

1.3.1 Полносвязные нейронные сети

Полносвязные нейронные сети, также известные как многослойные персептроны, состоят из набора нейронов, организованных в слои. Каждый нейрон в предыдущем слое соединен со всеми нейронами в следующем слое. Это классический тип нейронных сетей, широко применяемый для различных задач, включая классификацию, регрессию и обработку изображений.

1.3.2 Сверточные нейронные сети

Сверточные нейронные сети часто используются для работы с изображениями. Они состоят из сверточных слоев, которые применяют карту-признаков для выделения особенностей в изображении, и пулинг слоев, которые уменьшают размерность данных. Архитектура сверточных нейронных сетей способствует эффективному извлечению признаков из изображений и широко применяется в области компьютерного зрения.

1.3.3 Рекуррентные нейронные сети

Рекуррентные нейронные сети разработаны для обработки последовательных данных, таких как текст или временные ряды. Они имеют обратные связи и сохраняют внутреннее состояние (память) для учета контекста и зависимостей между последовательными элементами. Рекуррентные нейронные сети позволяют

моделировать долгосрочные зависимости и широко применяются в задачах обработки естественного языка, машинного перевода и анализа временных рядов.

1.4 Плюсы и минусы нейросетей

Стоит выделить плюсы и минусы нейросетей как инструмента для решения нетривиальных задач:

Плюсы:

1. Нейросети могут моделировать и обнаруживать сложные нелинейные взаимосвязи в данных, что позволяет им решать задачи, в которых традиционные методы неэффективны.
2. Нейросети могут автоматически извлекать важные признаки из входных данных, что делает их гибкими и универсальными в различных областях.
3. Нейросеть, обученная на некоторых данных, способна давать довольно точные предсказания для произвольной информации, с которой она прежде не сталкивалась.
4. Нейросети являются универсальным инструментом для решения различных задач, включая классификацию, регрессию, обработку изображений, обработку естественного языка и многие другие.

Минусы:

1. Процесс обучения и работа нейросети может требовать значительных вычислительных мощностей, особенно для больших моделей.
2. Нейросети нуждаются в большом объеме качественных данных для обучения. Недостаток данных или неправильный формат данных может привести к низкой точности предсказания.
3. Нейросети имеют огромное количество параметров, для подбора которых требуется достаточное понимание математики.
4. В процессе обучения может возникнуть переобучение, то есть момент, когда модель сильно подстраивается под обучающую выборку, а на тестовых данных, не участвовавших в обучении, демонстрирует низкую

точность. Для решение данной проблемы требуется применение специальных методов.

1.5 Выбор архитектуры нейросети

Задача распознавания рукописных цифр является одной из фундаментальных и начальных задач в области машинного обучения. Поэтому при выборе архитектуры можно рассматривать несколько вариантов базовых структур нейронных сетей. В качестве оптимальной архитектуры для решения поставленной задачи была выбрана свёрточная модель нейронной сети, так как она широко используется для работы с двумерными данными, такими как изображения. По логике своей работы сверточная нейросеть способна выделять определенные признаки на изображении, что делает её подходящей для решения задачи распознавания цифр.

1.6 Пользовательский интерфейс

Для удобного взаимодействия программы и пользователя требовалось разработать дружественный графический интерфейс для взаимодействия между программой и пользователем без специальных технических навыков. Так как поставленная задача не требует сложного и многоуровневого интерфейса, при его разработке был сделан упор на простоту и понятность.

1.7 Использованное ПО

Разработка всех компонентов программы производилась на языке программирования Python как на одном из самых популярных и высокоуровневых. Для разработки интерфейса использовалась стандартная библиотека Tkinter. Для построения, обучения, тестирования нейросети использовалась высокоуровневая библиотека Keras. Также в ходе работы были задействованы следующие

дополнительные библиотеки для работы с изображениями и другими данными: Pillow, NumPy, OpenCV, SciPy.

1.8 Вывод

На исследовательском этапе проекта были получены знания о основных принципах работы, видах, шагах разработки и параметрах нейросетей. Были изучены библиотеки для реализации поставленной задачи. В качестве основного инструмента для разработки нейросети была выбрана библиотека Keras. Приобретенные знания помогли выбрать подходящую архитектуру нейросети, подобрать верные параметры, спроектировать модель и реализовать её в практической части проекта.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Общие задачи и результаты

В практической части проекта стояло несколько основных задач: построение эффективной модели нейросети, создание дружественного интерфейса и интеграция нейросети в интерфейс.

Приведем общие результаты, полученные при решении вышеупомянутых задач. На Рисунке 2.1 показан разработанный интерфейс приложения.

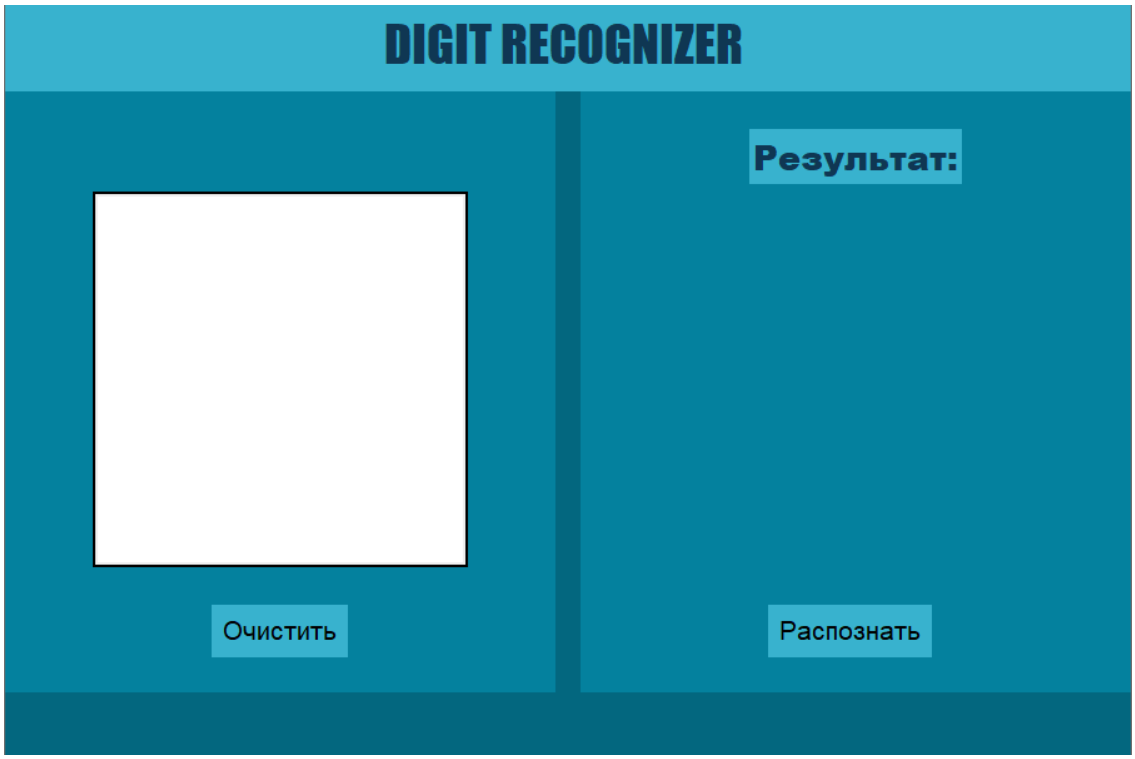

Рисунок 2.1 — Интерфейс приложения

Интерфейс приложения условно разделен на три части. В верхней части окна интерфейса написано название приложения. Левая часть интерфейса предназначена для ввода пользователем рукописной цифры от 0 до 9. Под холстом ввода расположена кнопка «Очистить», предназначенная для очисти области ввода. Правая часть интерфейса отвечает за вывод результата работы нейросети. Выходные данные представляют из себя саму цифру, которую распознала нейронная сеть и вероятность в процентах, что это именно эта цифра.

Результат появляется после того, как пользователь введет цифру и нажмет на кнопку «Распознать».

Продемонстрируем процесс работы приложения. Результат работы приложения при введённой цифре 8 изображен на Рисунке 2.2.

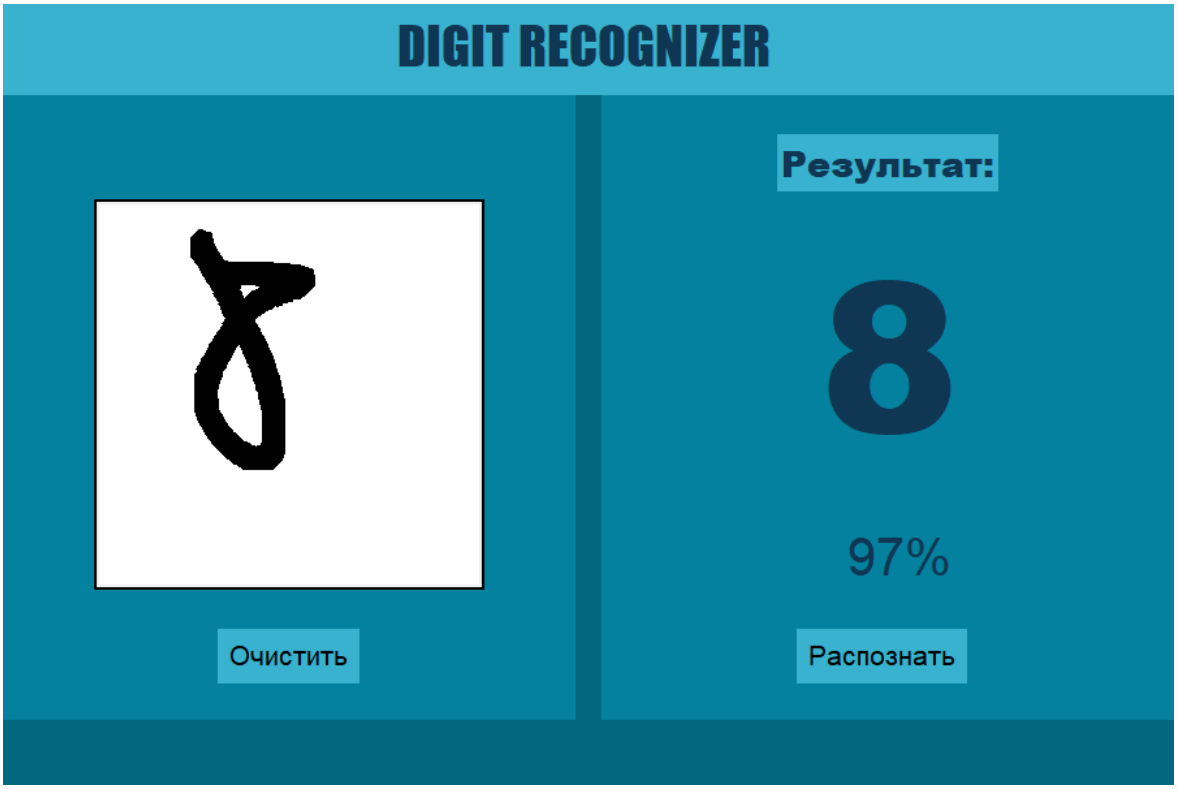

Рисунок 2.2 — Результат работы приложения

2.2 Построение модели нейронной сети

В рамках выполнения проектной работы была поставлена задача создать эффективную модель нейронной сети, которая позволит с высокой точностью определять рукописные цифры.

2.2.1 Библиотека Keras

В качестве основы для решения этой задачи была выбрана высокоуровневая библиотека Keras. Данная библиотека является эффективным и доступным инструментом для работы с искусственными нейронными сетями, так как она содержит реализации широко используемых алгоритмов и функций.

2.2.2 Сбор данных для обучения

Сбор данных для обучения традиционно является первым шагом для разработки нейронной сети. Данный шаг является одним из важнейших при проектировании нейросети и может быть достаточно затратным по времени и ресурсам. Но для задачи распознавания цифр существует огромная база данных образцов рукописных цифр MNIST (Modified National Institute of Standards and Technology). База данных MNIST состоит из 70 тыс. картинок в специальном формате, которые могут быть использованы для формирования обучающей и тестовой выборки.

2.2.3 Формат цифр MNIST

При обучении нейросети важно понимать в каком формате представлены обучающие данные. Формат цифр MNIST представляет из себя изображение 28x28 пикселей, на котором изображена белая цифру на черном фоне. Сама же цифра помещается в окно 20x20 пикселей. Пример цифры формата MNIST показан на Рисунке 2.3.

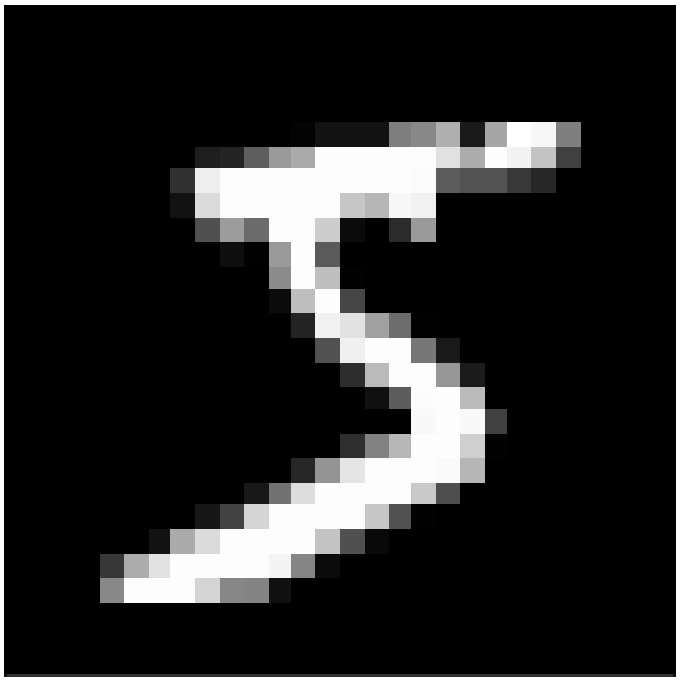

Рисунок 2.3 — Формат цифры MNIST

Знание устройства формата данных, на котором будет обучаться нейросеть, поможет в дальнейшем преобразовывать цифру, введенную пользователем к виду, который «понимает» нейросеть.

2.2.4 Подготовка данных для обучения

Перейдем к построению нейросети. Для начала импортируем библиотеки, которые будем использовать. Импорт необходимых библиотек представлен в Листинге 2.1.

Листинг 2.1 — Импорт необходимых библиотек

```
from tensorflow import keras
from keras import Sequential
from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
```

Выгрузим обучающие и тестовые данные из базы данных MNIST и нормализуем их (Листинг 2.2).

Листинг 2.2 — Выгрузка данных и нормализация

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train / 255
x_test = x_test / 255

y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

Данные x_train и x_test представляют из себя массив значений с от 0 до 255, где число определяет цвет этого пикселя, то есть 0 – черных, а 255 – полностью белый. Нормализованные данные заключены в диапазон от 0 до 1, что помогает нейросети более удобно работать с данными. Данные y_train и y_test также нормализуются и приводятся к формату вероятности.

2.2.5 Построение модели нейросети

Перейдем к построению модели нейросети. Как говорилось ранее для решения задачи распознавания цифр была выбрана сверточная архитектура нейронной сети, которая предназначена для работы с изображениями.

Реализация сверточной модели нейронной сети представлена в Листинге 2.3.

Листинг 2.3 — Сверточная модель нейросети

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

Опишем составляющие построенной архитектуры. Первый слой Conv2D является сверточным слоем с 32 фильтрами (картами-признаков), которые постепенно накладываются на кусочки изображения и определяют наличие того или иного признака на нём. Сами фильтры имеют размер 3x3 и функцию активации ReLU. На вход слоя подается изображение 28x28 пикселей с одном цветовым каналом (черно-былым).

Следующий слой MaxPooling2D или слой пулинга с размером 2x2. Он выполняет функцию уменьшения размерности данных и помогает извлечь наиболее значимые признаки, полученные из предыдущего слоя.

Далее используется сверточный слой с 64 фильтрами размером 3x3 и функцией активации ReLU. По логике работы аналогичен первому. После расположен еще один слой пулинга с размером 2x2.

Flatten это слой, преобразующий данные из двумерного формата в одномерный формат, то есть в одномерный вектор. Это используется для подачи данных на слои, работающие только с одномерными векторами, как например полносвязный слой.

Слой Dense представляет из себя полносвязные слой с 128 нейронами и функцией активации ReLU.

Наконец, последний полносвязный слой Dense с 10 выходами (равными количеству цифр) и функцией активации softmax. Данные слой подает на выход вероятность, того, что была распознана так или иная цифра от 0 до 9. Максимальное значения среди всех выходных вероятностей является лучшим «предсказанием»

нейросети. Цифра с наивысшей вероятностью и сама вероятность поступают в интерфейс как конечный «ответ» нейросети.

2.3 Обучение и тестирование

Перед началом обучения модели необходимо задать параметры обучения. В Листинге 2.4 приведена часть кода, на которой настраиваются параметры и запускается процесс обучения.

Листинг 2.4 — Обучение нейросети

```
model.compile(
optimizer="adam",
loss='categorical_crossentropy',
 metrics=['accuracy']
)

model.fit(x_train, y_train, batch_size=32, epochs=5, validation_split=0.2)
```

Сначала устанавливаются следующие параметры обучения: оптимизатор, функция потерь, метрика оценки точности.

Оптимизатор определяет способ обновления весов и минимизации функции потерь. Оптимизатор Adam является одним из самых широко используемых и эффективных алгоритмов обучения. Данный оптимизатор автоматические адаптирует скорость обучения и параметры весов. Это позволяет оптимизатору эффективно и быстро настроить веса модели.

Функция потерь используется для оценки отклонения между предсказанным значением и истинным результатом. При обучении данной модели используется категориальная кросс-энтропия функция потерь как одна из популярных функций в задачах многоклассовой классификации.

В качестве метрики оценки производительности модели используется метрика точности (accuracy), которая измеряет процент правильных предсказаний модели относительно общего числа примеров.

Далее передаются входные данные для обучения и устанавливаются следующие параметры: размер батча, количество эпох и параметр валидации.

Размер батча определяет количество примеров, которые будут одновременно обрабатываться моделью перед обновлением весов в процессе обучения.

Количество эпох отвечает за то, сколько раз обучающий набор данных будет проходить через модель нейросети при обучении.

Наконец, параметр валидации определяет сколько примеров из обучающей выборки будет использовано для тестирования модели в каждой эпохе. В данном случае это 20% обучающих данных.

Обученная модель должна быть протестирована данными, которые не учувствовали в обучении. Тестирование в библиотеке Keras происходит с помощью метода evaluate. Если модель показывает высокую точность, то её можно сохранить и использовать дальше для других данных. Процесс тестирования и сохранения модели показан в Листинге 2.5

Листинг 2.5 — Тестирование и сохранение модели

```
model.evaluate(x_test, y_test)
model.save('model.h5')
```

Полный код программы отражающий процесс построения, обучения и тестирования модели показан в Листинге 2.6.

Листинг 2.6 — Полный код программы

```
from tensorflow import keras
from keras import Sequential
from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

(x_train, y_train), (
x_test, y_test) = mnist.load_data()  # Выгружаем данные из mnist (x -
входные данные y - ожидаемый результат)

x_train = x_train / 255
x_test = x_test / 255
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)

model = Sequential([  # Сверточная архитектура нейросети
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

Продолжение Листинга 2.6

```
# Определение метрик, функции ошибки и оптимизации для обучения
model.compile(optimizer="adam", loss='categorical_crossentropy',
metrics=['accuracy'])


# Запуск обучения сети, установка кол-ва эпох и батча
# (эпоха - кол-ва итераций прогона обучающих данных, батч - часть данных
использующаяся в эпохе)
model.fit(x_train, y_train, batch_size=32, epochs=5, validation_split=0.2) #
Для небольшой сверточной модели

model.evaluate(x_test, y_test)

# Сохранения модели в файл
model.save('model.h5')
```

2.4 Обработка входных данных

2.4.1 Правила построение формата MNIST

Для того чтобы обученная нейронная сеть могла работать с произвольными данными требуется специальная обработка входных изображений. Другими словами, нужно преобразовать входные данные, полученные от пользователя, к формату на котором обучалась нейронная сеть и только потом их передавать.

Как было описано ранее нейросеть обучалась на формате, использованном в базе рукописных цифр MNIST. Сам формат строился по определенным правилам. Рисовалась белая цифра на черном фоне, которая умещалась в бокс 20x20 пикселей. Далее, вычислялся центр масс изображения, и оно помещалось на поле 28x28, так чтобы центр масс совпадал с центром поля.

2.4.2 Алгоритмы обработки изображения

Для преобразования входного изображения к формату MNIST использовались библиотеки OpenCV, NumPy и SciPy, содержащие алгоритмы для работы с изображениями и другими данными.

На Листинге 2.7 представлен полный код программы, отвечающий за преобразование изображения к формату MNIST.

Листинг 2.7 — Обработка изображения

```
from scipy.ndimage import center_of_mass
import math
import cv2
import numpy as np
from keras.models import load_model

# Загрузка обученной модели нейронной сети
model = load_model('model.h5')



# Вычисление центра масс и направления сдвига
def getBestShift(img):
    cy, cx = center_of_mass(img)

    rows, cols = img.shape
    shiftx = np.round(cols / 2.0 - cx).astype(int)
    shifty = np.round(rows / 2.0 - cy).astype(int)

    return shiftx, shifty


def shift(img, sx, sy):
    rows, cols = img.shape
    M = np.float32([[1, 0, sx], [0, 1, sy]])
    shifted = cv2.warpAffine(img, M, (cols, rows))
    return shifted


def recognize():
    img = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)
    gray = 255 - img
    # применяем пороговую обработку
    (thresh, gray) = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY |
cv2.THRESH_OTSU)

    # удаляем нулевые строки и столбцы
    while np.sum(gray[0]) == 0:
        gray = gray[1:]
    while np.sum(gray[:, 0]) == 0:
        gray = np.delete(gray, 0, 1)
    while np.sum(gray[-1]) == 0:
        gray = gray[:-1]
    while np.sum(gray[:, -1]) == 0:
        gray = np.delete(gray, -1, 1)
    rows, cols = gray.shape
    # изменяем размер, чтобы помещалось в box 20x20 пикселей
    if rows > cols:
        factor = 20.0 / rows
        rows = 20
        cols = int(round(cols * factor))
        gray = cv2.resize(gray, (cols, rows))
    else:
        factor = 20.0 / cols
        cols = 20
        rows = int(round(rows * factor))
        gray = cv2.resize(gray, (cols, rows))
    # расширяем до размера 28x28
    colsPadding = (int(math.ceil((28 - cols) / 2.0)), int(math.floor((28 -
cols) / 2.0)))
```

Продолжение Листинга 2.7

```
    rowsPadding = (int(math.ceil((28 - rows) / 2.0)), int(math.floor((28 -
rows) / 2.0)))
    gray = np.lib.pad(gray, (rowsPadding, colsPadding), 'constant')

    # сдвигаем центр масс
    shiftx, shifty = getBestShift(gray)
    shifted = shift(gray, shiftx, shifty)
    gray = shifted

    cv2.imwrite('image_mnist.jpg', gray)
    img = gray / 255.0
    img = np.array(img).reshape(-1, 28, 28, 1)  # конвертируем в картинку 28
на 28 пикселей
    prediction = model.predict(img)
    percent = prediction[0][np.argmax(prediction)]  # Результаты работы сети
    percent = int(percent * 100)
    digit = np.argmax(prediction)
    print("Predicted digit:", digit)
    print(percent)
    return digit, percent
```

Алгоритм обработки изображения и выдачи результата состоит из трех основных функций: getBestShift, shift и recognize.

Функция getBestShift вычисляет сдвиг изображения, чтобы его центр масс совпадал с центром изображения. Это позволяет корректно выравнивать и центрировать цифру перед распознаванием.

Функция выполняет сдвиг изображения на указанные значения sx и sy. Это используется для фактического сдвига цифры в соответствии с вычисленными сдвигами.

Функция recognize загружает изображение, преобразует его в оттенки серого и применяет пороговую обработку для получения бинарного изображения. Затем она обрезает нулевые строки и столбцы, чтобы удалить пустые области вокруг цифры. Затем изображение изменяется до размера 20x20 пикселей и затем до размера 28x28 пикселей. Затем происходит сдвиг центра масс изображения и сохранение результата в файл. Затем изображение нормализуется и преобразуется в массив, который передается в модель для предсказания. Далее возвращаются результаты работы нейросети для обработанного изображения. Эти данные подаются на интерфейс для отображения в приложении.

ЗАКЛЮЧЕНИЕ

В ходе данного проекта была успешно разработана нейросеть для распознавания цифр с дружественным интерфейсом. Основной целью проекта было создание эффективной модели, взаимодействие с которой происходило через пользовательский графический интерфейс.

В рамках работы были достигнуты все поставленные задачи. Был проведен обширный анализ данных, предварительную обработку изображений и подготовку набора данных для обучения. На этапе обучения модели была применена нейронная сеть с сверточной архитектурой, которая показала отличные результаты в распознавании цифр. Был спроектирован простой и понятный интерфейс для взаимодействия с пользователем.

Однако, в рамках дальнейших исследований и разработки проекта, есть несколько направлений для улучшения. Во-первых, можно провести дополнительные эксперименты с различными архитектурами нейронных сетей и параметрами обучения, чтобы улучшить точность распознавания.

Кроме того, стоит обратить внимание на расширение функциональности интерфейса. В дальнейшем можно добавить возможность распознавания и классификации не только цифр, но и других символов или объектов. Также можно улучшить пользовательский опыт, добавив функции для обработки и улучшения входных изображений, чтобы повысить точность распознавания в условиях с плохим качеством фотографий.

Предложенный проект может быть доработан и улучшен в будущем, открывая возможности для применения в различных сферах, где требуется автоматическое распознавание цифровых данных.

Таким образом, цель данного проекта была успешно достигнута, и разработанная нейросеть продемонстрировала высокую точность и эффективность в распознавании цифр.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Введение в нейросети / Ившин Вадим. URL: https://habr.com/ru/articles/342334/ (Дата обращения: 22.03.23)
2. Траск Эндрю Грокаем глубокое обучение. — СПб.: Питер, 2019. — 352 с.
3. Keras layers / Документация библиотеки Keras. URL: https://keras.io/api/layers/ (Дата обращения: 25.03.23)
4. MNIST digits classification dataset / Документация библиотеки Keras. URL: https://keras.io/api/datasets/mnist/ (Дата обращения: 25.03.23)
5. Нейросети и глубокое обучение, глава 1: использование нейросетей для распознавания рукописных цифр / Голованов Вячеслав. URL: https://habr.com/ru/articles/456738/ (Дата обращения: 03.04.23)
6. Гасников А. В. Современные численные методы оптимизации. Метод универсального градиентного спуска: учебное пособие. — М.: МФТИ, 2018. — 291 с.
7. Оптимизаторы в ML / Тяпкин Д. / URL: https://academy.yandex.ru/handbook/ml/article/optimizaciya-v-ml (Дата обращения: 06.04.23)