Оглавление

Введение

Жесты представляют собой тип видеообразов, которые компьютер может распознать благодаря моделям машинного обучения. В контексте компьютерного зрения руки являются сложными объектами, так как они активно жестикулируют, могут перекрывать друг друга, менять форму от раскрытой ладони до сжатого кулака и скрещивать пальцы. В отличие от лица, где есть явные активные точки, такие как глаза и рот, на руках такой однозначной информации нет, что затрудняет восприятие языка жестов.

Обучение машин распознавать движения в пространстве является задачей высокой сложности. Сначала требуется обнаружить объект в кадре. Модель машинного обучения выделяет целевой объект, опираясь на специальные критерии и параметры, которые нейросеть усваивает в процессе обучения.

Однако в данной научно-исследовательской работе будет рассмотрен более простой тип распознавателя жестов, в котором модель обучается на наборе данных

Глава 1

Постановка задачи

Описание модели

В данной научно-исследовательской работе описан более простой тип распознавателя жестов, в котором модель обучается на наборе данных, состоящем из фотографий легкоразличимых жестов. Модель основана на работе сверточных нейронных сетей.

Набор данных

1. Датасет взят на сайте Kaggle, доступ к которому можно получить по ссылке в списке литературы.
2. Жесты легкоразличимы
3. Изображения черно-белые
4. Каждое изображение имеет размерность 50x50 (можно изменить)
5. Метки кодируются числами от 1 до 20
6. Данные содержат обучающий и тестовый наборы

Метрика оценки

В качестве метрики оценки мы будем использовать точность. Точность – это отношение правильно классифицированных экземпляров к их общему количеству.

Моделирование

Для задачи классификации изображений были выбраны сверточные нейронные сети (Convolutional Neural Network - CNN).

Сверточные нейронные сети

Сверточные нейронные сети (CNN) – это особый класс нейронных сетей, совершенствованных для задачи анализа изображений и видеоданных. Их архитектура включает несколько ключевых компонентов, которые обеспечивают эффективное извлечение и классификацию признаков.

1. Входной слой (Input Layer)

Входной слой принимает исходное изображение, представленное в виде двумерного массива данных, где каждый элемент отражает цвет и яркость пикселя. Широкие размеры изображений позволяют сети изучать разнообразные визуальные паттерны.

2. Сверточные слои (Convolutional Layers)

Сверточные слои, являющиеся основными в CNN, предназначены для начального поиска признаков, таких как края и текстуры. В основе его работы лежит операция свертки, которая позволяет эффективно выделять важные характеристики изображений, такие как текстуры, края и углы. Основные элементы сверточных слоев:

Фильтры: Фильтры в сверточном слое представляют собой набор весов — небольшие матрицы, применяемые к входным данным. Матрицы весов калибруются для выявления особенностей в изображениях. Каждый фильтр осуществляет выборку специфических признаков, основанных на локальных изменениях.

Операция свертки: Каждый фильтр перемещается по изображению с заданным шагом (stride), при этом происходит поэлементное умножение значений фильтра на участок изображения, после чего результаты суммируются для получения значений пикселей, создавая карту признаков.

Карта признаков: Результат свертки формирует карту признаков, где каждый пиксель соответствует уровню активации конкретного фильтра. Чем выше значение, тем более выражена указанная особенность в соответствующей области.

Страйды и Паддинг: Страйды влияют на размер карты признаков: большие страйды уменьшают размерность, обеспечивая широкий, но менее детальный обзор, тогда как маленькие страйды создают более детализированную карту. Паддинг (padding) добавляет пиксели вокруг изображения, позволяя сохранять его пространственные размеры. Существуют различные стратегии паддинга: 'Same', сохранившая размер изображения, и 'Valid', не использующая паддинг и уменьшающая размер карты признаков.

3. Слои Пулинга (Pooling Layers)

Пулинг — это сверточный процесс, уменьшающий размерность карт признаков. Наиболее распространенные методы:

MaxPooling: Извлекает максимальные значения из подмодульных карт признаков, выявляя ключевые детали.

AveragePooling: Вычисляет среднее значение, что позволяет сохранить более общую информацию и уменьшить риск переобучения.

4. Выпрямляющий слой

Перед тем как передать карты признаков к полносвязным слоям, они преобразуются в одномерный вектор. Этот процесс называется выпрямлением (flattening) и необходим для дальнейшей классификации.

5. Полносвязные слои

В полносвязных слоях каждый нейрон соединен со всеми выходами предыдущих уровней. Это обеспечивает интеракцию всех высокоуровневых признаков, позволяя сети осуществлять более сложные операции и повышать точность классификации.

6. Выходной слой

На заключительном этапе находится выходной слой, использующий функцию активации, например, softmax. Она вычисляет вероятности принадлежности входного изображения к различным классам. Это позволяет сети сделать окончательное предсказание, основываясь на вероятностной интерпретации.

Глава 2

Реализация модели

Использованные библиотеки

(Реализация представлена в приложении 1)

TensorFlow — это опенсорсная библиотека, созданная Google, которая используется при разработке систем, использующих технологии машинного обучения. Эта библиотека включает в себя реализацию множества мощных алгоритмов, рассчитанных на решение распространённых задач машинного обучения, среди которых можно отметить распознавание образов и принятие решений. Основной API для работы с библиотекой реализован для Python.

Тензор (tensor) — это стандартный способ представления данных в системах глубокого обучения. Тензоры — это многомерные массивы, расширение двумерных таблиц (матриц) для представления данных, имеющих более высокие размерности.

Библиотека предоставляет удобные инструменты для работы со сверточными нейронными сетями. Один из них - Keras.

Keras - это высокоуровневый API для построения и обучения моделей включающий поддержку для TensorFlow. Keras делает упрощает использование TensorFlow, не жертвуя при этом гибкостью и производительностью. Данная библиотека является модульной, модуль, который используется для работы со сверточными нейронными сетями - это **layers**. В нем реализована система сверточных слоев (Conv2D для двумерных входных данных, таких как изображения), через которые будут пропускаться входные данные.

NumPy - это основная библиотека для математических расчетов в Python. Она предлагает объекты n-мерных массивов и функции для работы с массивами. Библиотека используется для работы с массивами, в которых представлены входные данные.

Matplotlib — это библиотека для визуализации данных в Python.

SciPy — это библиотека для научных и технических вычислений, которая строится на основе NumPy и предоставляет множество функций для численного интегрирования, оптимизации и обработки сигналов.
Библиотека **os** осуществляет доступ к файловой системе компьютера для обращения к набору данных.

Работа с проводником

(Реализация представлена в приложении 2)

Для доступа к датасету необходимо указать директории тренировочных и проверочных наборов данных. Для проверки, что данные распознаны верным образом, осуществляется вывод всех названий классов и количество фотографий в них.

Обработка изображений

(Реализация представлена в приложении 3)

Из библиотеки Keras импортируется класс ImageDataGenerator, который используется. для аугментации изображений.

Пиксели изображений нормализуются. Нормализация пикселей изображений — это процесс преобразования значений пикселей в более управляемый диапазон, обычно от 0 до 1. Нормализация помогает ускорить процесс обучения нейронных сетей, так модели легче обучаются, когда входные данные имеют более однородное распределение. Кроме того изображения в датасете могут иметь разные значения яркости и контрастности, что может повлиять на обучение модели. Нормализация помогает устранить такие различия, обеспечивая более согласованные входные данные. Так нормализация пикселей — это ключевой этап предобработки данных при работе с изображениями в задачах машинного обучения и глубокого обучения, который помогает повысить точность и стабильность модели.

Данные, предварительно разбитые на тренировочные и валидационные описываются методом класса ImageDataGenerator из библиотеки Keras datagen.flow from directory , который используется для загрузки изображений с диска и их аугментации в процессе обучения нейронной сети. Этот метод позволяет организовать данные в удобном формате.

Так данные имеют информацию о целевом размере изображения и размере батчей (небольших партий данных, используемая при обучении моделей машинного обучения). Так же указана многоклассовость набора и информация о том, что фотографии черно-белые.

Описание модели

(Реализация представлена в приложении 4)

Модель имеет класс keras.Sequential, который подразумевает последовательное расположение слоев.

В переменные слоев Conv2D записываются количество фильтров и размер ядра свертки - размер фильтра (количество пикселей, обрабатываемое за раз). Также задается функция активации свертки 'relu' (наиболее распространённая функция активации, которая заменяет все отрицательные значения на ноль, тем самым добавляя нелинейность в модель) и параметры входных данных (размерность и указание на то, что изображения в градациях серого). MaxPooling2D — это слой, который выполняет операцию максимального подвыборки (пулинга) на двухмерных данных, что помогает уменьшить размерность и извлечь наиболее значимые характеристики.

Слой Flatten преобразует входные данные из многомерного массива (например, 3D массива, представляющего изображение) в одномерный вектор. Это необходимо для подготовки данных для полносвязного слоя (Dense), поскольку Dense-слои ожидают вход в виде векторов.

Слой Dense представляет собой полносвязный слой, в котором каждое входное значение связано с каждым выходным нейроном. В паремтр ставится количество нейронов, а также функция kernel regularizer, добавляющая штрафы к весам в случае переобучения модели. Еще один слой Dense обычно используется как выходной слой в модели для многоклассовой классификации. В нем Функция активации softmax нормализует выходные значения так, чтобы они представляли собой вероятности принадлежности к каждому из классов (сумма вероятностей будет равна 1).

Разберем строку, где происходит компиляция модели в Keras. Оптимизатор 'adam' — это один из самых распространенных оптимизаторов, использующийся для обучения нейронных сетей. Функция потерь loss измеряет, насколько хорошо предсказания модели соответствуют истинным меткам. Задается метрика accuracy, которая отслеживается как точность классификации данных.

Модель в Keras обучается методом fit. В него передаются данные из функций, описанных в предыдущем разделе. Количество шагов за эпоху равно количеству образцов данных для обучения, деленному на размер батча. Результат выполнения метода fit сохраняется в переменной history. Этот объект содержит данные о ходе обучения — значения функции потерь и метрик для обучения и валидации по каждой эпохе.

В конце на вывод подаются переменные точности, потерь, потерь валидации и точности валидации.

Вывод графиков

(Реализация представлена в приложении 5)

На графиках изображены изменения точности и потерь между эпохами. Вывод графиков описан стандартными функциями.

Заключение

В данной работе были изучены современные подходы к распознаванию жестов с использованием методов глубокого обучения. Рассмотрены ключевые концепции, касающиеся архитектур сверточных нейронных сетей, принципов их работы, а также роли различных слоев в процессе извлечения и классификации характеристик из изображений жестов.

В процессе работы была разработана и обучена модель для распознавания жестов, основанная на сверточной нейронной сети. Модель продемонстрировала эффективную работу с использованием сверточных слоев, которые позволили выделить ключевые признаки, такие как текстуры и края жестов. С помощью операции максимального пулинга была проведена оптимизация размерности данных, что улучшило скорость обработки и уменьшило вероятность переобучения. Для повышения устойчивости модели использовалась регуляризация в полносвязных слоях.

В заключение, результаты, достигнутые в ходе экспериментов, подтверждают высокую точность распознавания жестов, что свидетельствует о перспективах применения глубоких нейронных сетей в данной области. Разработка эффективного распознавателя жестов может значительно упростить управление устройствами и сделать взаимодействие с ними более интуитивным, что имеет большое значение в современном мире технологий.

Список литературы

1. https://www.kaggle.com/datasets/aryarishabh/hand-gesture-recognition-dataset
2. V. I. Pavlovic, R. Sharma and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review"
3. L. Pigou, S. Dieleman, P.-J. Kindermans and B. Schrauwen, "Sign language recognition using convolutional neural networks"
4. S. Mitra and T. Acharya, "Gesture recognition: A survey"
5. Y. K. Meshram, "Hand Gesture Recognition using Deep Learning Techniques"

Приложение

Listing 1: Использованные библиотеки

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
import matplotlib.pyplot as plt
import os
import scipy
```

Listing 2: Работа с проводником

```
# Path to the dataset
train_dir = 'Path/to/training/dataset'
val_dir = 'Path/to/validation/dataset'

# Checking that the folder exists
print(os.listdir(train_dir))

# List of classes (folders' names)
for class_name in os.listdir(train_dir):
    print(f"Class: {class_name}, Number of pictures:
    {len(os.listdir(os.path.join(train_dir, class_name)))})")
for class_name in os.listdir(val_dir):
    print(f"Class: {class_name}, Number of pictures:
    {len(os.listdir(os.path.join(val_dir, class_name)))})")
```

Listing 3: Обработка изображений

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Creating of image generator
datagen = ImageDataGenerator(
    rescale=1./255,   # pixel normalization
)

train_generator = datagen.flow_from_directory(
    train_dir,
    target_size=(50, 50), # images can be compressed to reduce
                          # complexity
    batch_size=32,
```

```
    class_mode='categorical',
    color_mode='grayscale', # images are black and white
)

validation_generator = datagen.flow_from_directory(
    val_dir,
    target_size=(50, 50),
    batch_size=32,
    class_mode='categorical',
    color_mode='grayscale',
)
```

Listing 4: Модель

```
model = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu',
                                    input_shape=(50, 50, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(32, activation='relu', kernel_regularizer =
                                  keras.regularizers.l2(0.001)),
    layers.Dense(len(train_generator.class_indices),
                                          activation='softmax')
])

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

history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples
    // train_generator.batch_size,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples
    // validation_generator.batch_size,
    epochs=10
)

val_loss, val_accuracy = model.evaluate(validation_generator)
```

```
print(f"Validation accuracy: {val_accuracy:.4f}")
```

Listing 5: Графики

```
def plot_history(history):
    plt.figure(figsize=(12, 4))

    # Subplot for accuracy
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='accuracy')
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend(loc='lower right')
    plt.title('Training and Validation Accuracy')

    # Subplot for losses
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend(loc='upper right')
    plt.title('Training and Validation Loss')

    plt.show()

plot_history(history)
```

Listing 6: