Распознавание жестов человека нейросетями и применение в быту.

Инженерные науки, Информационные науки

 Мбоу сош имени Героя Советского союза Александра Тимофеевича Бодряшова

Рп. Шемышейка

Выполнил: Пилюгин Артём Эдуардович

Целью данного проекта является разработка программы (нейросети) и её последующее обучение распознаванию жестов человека, что может пригодится человеку в быту или людям с ограниченными возможностями.

CNN или сверточные нейронные сети являются наиболее часто используемыми алгоритмами для задач классификации изображений. Классификатор изображений берет фотографию или видео в качестве входных данных и классифицирует их по одной из возможных категорий, которые он был обучен идентифицировать. У них есть приложения в различных областях, таких как автомобили без водителя, оборона, здравоохранение и т. д.

Существует множество алгоритмов классификации изображений, и в этом эксперименте мы рассмотрим один из самых популярных алгоритмов в этом жанре, который называется SqueezeNet от DeepScale.

Для решения данной задачи в ходе проекта использовался язык программирование Python, а также множество сторонних библиотек для работы с нейросетями. Помимо этого использовалось оборудование способное принимать сигналы при помощи программы. Набор изображений для обучения нейросети снимался на камеру.

В нашей финальной программе мы будем выполнять некоторые действия. Вот примерный список действий, которые я решил сделать. В зависимости от конкретных требований это количество элементов в этом списке будет другим. Далее я для простоты сопоставлю эти действия с именем категории, чтобы использовать их позже в нашей программе:

1. Увеличить громкость
2. Уменьшить громкость
3. Полное отключение звуков/включение
4. Пуск/Пауза

5.Открытие Google Chrome

Помимо этих пяти категорий, в программе также должна быть еще одна категория (категория = ничего), которая будет использоваться в окончательной программе, когда пользователь показывает жест рукой, который модель не распознает, или когда пользователь не вводит жесты.



2.Сбор изображений для обучения

Следующий шаг — подготовить тренировочные изображения для каждой из этих категорий. Чтобы собрать набор обучающих данных (изображения), я буду использовать веб-камеру. Чтобы упростить задачу, я использовал изображения с простым незагроможденным фоном.

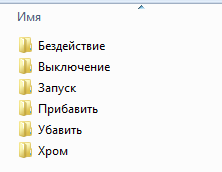
Следующая программа на Python будет использовать библиотеку OpenCV для выполнения этого действия и сохранения этих изображений в папке с именем — «training\_images». Программа принимает два входных параметра:



А) число изображений, которое нужно снять

Б) Название категории(прибавить/убавить звук и т.д)

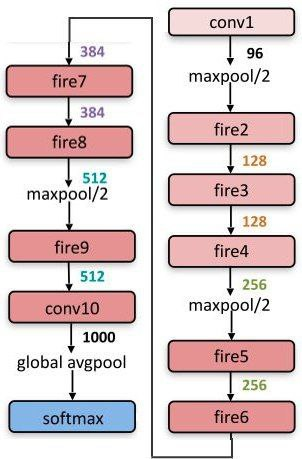
Программа останавливается после захвата такого количества изображений. Второй параметр указывает название категории, к которой принадлежат эти изображения. Программа создает подпапку по названию категории внутри нашей папки «training\_images» и сохраняет в ней все изображения.



После запуска программы, включится трансляция с веб-камеры. При помещении руки в пределы белого прямоугольника и нажатии клавишы «s», начнётся съёмка руки.

3.Обучение нейросети

Я собираюсь использовать SqueezeNet для этой демонстрации. SqueezeNet — популярная предварительно обученная модель для задач классификации изображений, чрезвычайно легкая и с впечатляющим уровнем точности. В процессе обучения я буду использовать библиотеку Keras. Keras — это простая и мощная библиотека Python, которая широко используется в глубоком обучении.



Это значительно упростило обучение моделей нейронных сетей.

Давайте посмотрим на некоторые параметры, используемые для нашего обучения.

Последовательный: я буду использовать последовательную модель, что означает, что слои расположены в виде линейного стека (последовательности). Слои в модели добавляются в качестве аргументов в этот конструктор.

Частота отсева: нейронные сети, обученные на небольших наборах данных, часто имеют тенденцию к чрезмерной подгонке и, следовательно, с меньшей вероятностью будут точными на новых данных. Теоретически,

лучший способ обучить модель может заключаться в том, чтобы попробовать как можно больше различных комбинаций различных значений параметров, а затем взять среднее значение этих отдельных результатов, чтобы получить обобщенный результат. Но это потребует много времени и вычислительных ресурсов для многократного обучения модели с несколькими комбинациями этих параметров.

. Чтобы обойти это, был введен коэффициент отсева. Здесь некоторые блоки/узлы в слое (из входных слоев или скрытых слоев, но не из выходного слоя) «отбрасываются», что приводит к исчезновению входящих и исходящих соединений этого узла. Проще говоря, когда это делается несколько раз во время тренировки,

различное количество узлов удаляется из слоя, из-за чего слой выглядит по-разному с точки зрения количества узлов и его соединений с предыдущим слоем (грубо имитируя несколько моделей с разными конфигурациями слоев). Хорошее значение Dropout rate для слоя составляет от 0,5 до 0,8. В моем случае я нашел 0.

5, чтобы дать мне лучший результат после того, как я попробовал несколько разных значений. Обратите внимание, что отсев используется, чтобы избежать чрезмерной подгонки. Если точность модели низкая, этого параметра можно было бы избежать.

Узлы: количество нейронов/узлов в нашем выходном слое равно количеству классов, которое я пытаюсь предсказать.

Форма ввода: Форма ввода, которую требует SqueezeNet, составляет не менее 224 X 224 (с 3 каналами для RGB). В этой программе я использовал 225 X 225.

Выбор оптимизатора и потери: здесь я выбрал оптимизатор Адама. Функция потерь выбирается в зависимости от типа проблемы. Например: - для проблемы бинарной классификации,

[loss=’binary\_crossentropy’] лучше подходит, а для задачи классификации нескольких классов выбирается [loss=’categorical\_crossentropy’]. В то время как для проблемы регрессии можно выбрать [loss=’mse’]. Поскольку у нас проблема классификации нескольких классов, я буду использовать categorical\_crossentropy.

Количество эпох: количество эпох — это количество раз, когда весь набор данных проходит через нейронную сеть во время обучения. Для этого нет идеального числа, и оно зависит от данных. В данном случае я начал с 10, а использовал 15. Чем больше число, тем больше время обучения.

Функция активации: я использую функцию активации ReLU, которая является наиболее распространенной функцией активации, используемой в нейронных сетях из-за ее простоты вычислений (среди других преимуществ). Эта функция возвращает ноль для отрицательных входных данных и само значение для положительных входных данных. Для большинства современных нейронных сетей

ReLU — это функция активации по умолчанию. Другими функциями активации, которые также используются, являются Sigmoid, Tanh и т. д.

Объединение: в CNN обычно добавляют слой объединения после слоя свертки и активации.

Входное изображение дискретизируется или преобразуется в версию с низким разрешением, чтобы сохранить только важные детали и удалить менее важные детали.

Softmax: слой softmax используется в многоклассовой классификации непосредственно перед выходным слоем. Это дает вероятность принадлежности входного изображения к определенной категории.

После обучения я сохраню параметры обученной модели в файл, который я буду использовать позже во время тестирования модели.

4.Тестирование модели

тестовых изображений модель спрогнозировала правильно. Однако, если Для следующих изображение содержит слишком много других элементов на заднем плане, оно не всегда точное.



Возможные будущие улучшения:

В дальнейшем планируется устранение недочётов программы, создание большего датасета изображений, увеличение точности распознавания.

*Литература:*

1. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size (https://arxiv.org/abs/1602.07360)

2. Review: SqueezeNet (Image Classification) (https://towardsdatascience.com/review-squeezenet-image-classification-e7414825581a)

3. Keras API reference (https://keras.io/api/)

4. Распознавание жестов с помощью Raspberry Pi и OpenCV (https://microkontroller.ru/raspberry-pi-projects/raspoznavanie-zhestov-s-pomoshhyu-raspberry-pi-i-opencv/)

5. OpenCV (https://opencv.org/)