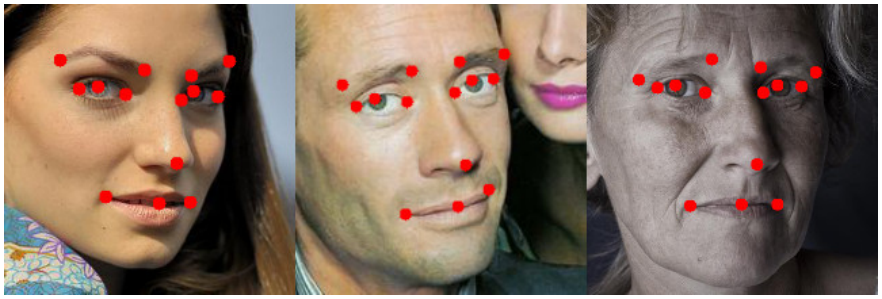
Поиск ключевых точек на лице

Азамат Мифтахов, Влад Шахуро, Владимир Гузов, Даниил Киреев

Обзор задания

В данном задании предлагается реализовать регрессию ключевых точек лица с использованием сверточной нейронной сети. Для реализации используйте библиотеку Keras, которая является частью TensorFlow. Библиотека работает на CPU и GPU. Если у вас есть CUDA-совместимая видеокарта, мы советуем использовать её, тогда обучение нейросети будет идти на порядок быстрее.



Базовая часть

Архитектура нейронной сети

На вход нейронной сети подается трехканальное изображение фиксированного размера. Размер входного слоя определяет компромисс между скоростью обучения и работы нейронной сети и качеством распознавания. Таким образом, все изображения обучающей выборки нужно привести к фиксированному размеру и перевести координаты соответствующих им ключевых точек. Изображения перед подачей на вход нейросети рекомендуется нормализовать. Для этого можно для каждого канала попиксельно посчитать среднее и дисперсию по всей выборке и нормализовать каждое изображение с помощью посчитанных значений.

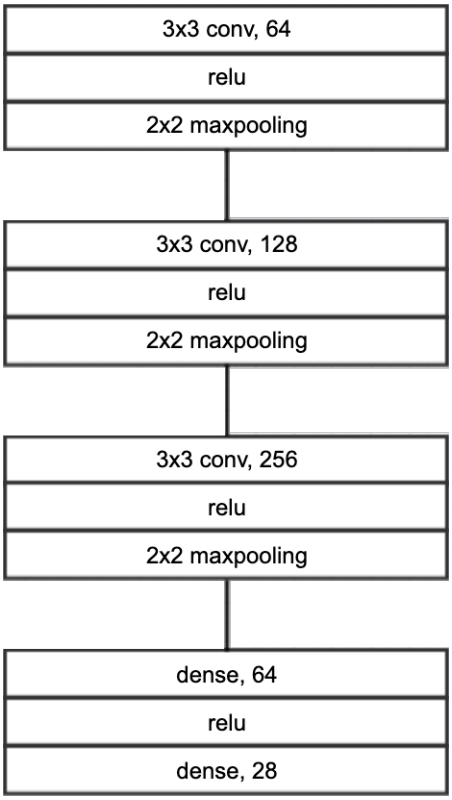
Слои нейронной сети

- Слой свертки – набор обучаемых фильтров. Фильтр представляет из себя окно, которое скользит по всей области предыдущего слоя и находит определенные признаки объектов. Размер фильтра обычно берут в пределах от 3×3 до 7×7. Предпочтительнее выбирать несколько слоев с маленьким размером фильтра (чередуя свертки с нелинейными функциями активации), чем один слой с большим. Это позволяет уменьшить количество обучаемых параметров модели.
- Слой активации – некоторая нелинейная функция, которая применяется к выходу сверточного слоя. Наиболее популярные функции активации: Sigmoid – $f(x) = \frac{1}{(1+e^{-x})}$ и ReLU – $f(x) = \max(0, x)$

- Слой пулинга – уплотнение карты признаков, при котором непересекающиеся группы пикселей, сжимаются в один пиксель. Наиболее популярная функция пулинга – максимум из области 2x2. Операция пулинга позволяет существенно уменьшить пространственный объём изображения.

- Полносвязный слой – это слой обычного многослойного персептрона. После нескольких прохождений свертки изображения и уплотнения с помощью пулинга система перестраивается к более абстрактным картам признаков. В итоге остаётся набор каналов, хранящих самые абстрактные понятия, выявленные из исходного изображения. Эти данные объединяются и передаются на обычную полносвязную нейронную сеть, которая тоже может состоять из нескольких слоёв. При этом полносвязные слои уже утрачивают пространственную структуру пикселей.

В качестве базовой можно использовать следующую архитектуру:



Стохастический градиентный спуск

Для настройки весов нейронной сети используется стохастический градиентный спуск. Скорость и качество обучения зависит от параметров градиентного спуска:

- Темп обучения(learning rate) – это параметр градиентных алгоритмов обучения нейронных сетей, позволяющий управлять величиной коррекции весов на каждой итерации. Выбирается в диапазоне от 0 до 1. Ноль указывать бессмысленно, поскольку в этом случае корректировка весов вообще производиться не будет. При большом learning rate алгоритм будет работать быстрее, однако может снизиться точность настройки модели, либо алгоритм может перестать сходиться. Малые значения learning rate соответствуют меньшему шагу. В этом случае число шагов обучения, требуемое для поиска экстремума, как правило увеличивается, но возрастает и точность настройки алгоритма, что потенциально уменьшает ошибку обучения. На практике коэффициент скорости обучения обычно подбирают экспериментально. Для темпа

обучения обычно задают расписание, например, делить lr на 10 когда ошибка на тренировочной выборке перестала уменьшаться. Постепенное уменьшение темпа обучения позволяет лучше приблизиться к локальному минимому.

- Момент (momentum). При использовании момента градиент для спуска определяется как взвешенная сумма градиентов на предыдущей и текущей итерации. Такое определение градиента позволяет избегать некоторых локальных минимумов. Существует также момент Нестерова, который является модификацией обычного момента. Значение момента выбирается в интервале от 0 до 1, на практике берется значение близкое к 0.9. При слишком маленьком значении момента будут происходить колебания около локального минимума, при слишком большом значении обучение сильно замедлится.

- Minibatch. Для лучшей сходимости алгоритма градиентный шаг делается по нескольким примерам, множество этих примеров называется минибатчем. Градиентный шаг вычисляется как усредненное значение градиента по всем примерам минибатча. Наилучшая сходимость алгоритма достигается при минибатче, равном всей обучающей выборке, однако на практике данный подход не применяется, так как такой минибатч часто не помещается в памяти компьютера , а алгоритм требует значительных временных затрат. Поэтому размер минибатча следует выбирать исходя из задачи и располагаемых ресурсов.

- Метод оптимизации. Кроме классического стохастического градиентного спуска существуют его различные модификации:

  – RMSprop. Данный метод уменьшает колебания и автоматически подбирает learning rate отдельно для каждого параметра.

  – Adam. Данный метод совмещает в себе преимущества momentum и RMSprop.

- Время обучения в эпохах. Эпоха — количество итераций, которое необходимо, чтобы просмотреть всю обучающую выборку с помощью минибатчей. Останавливать обучение имеет смысл, когда ошибка на тестовой выборке перестает уменьшаться.

Мониторинг обучения

Метрикой качества является среднеквадратичная ошибка:

$$Q(w, X, Y) = \frac{1}{n} \sum_{i=1}^{n} (f(w, x_i) - y_i)^2,$$

где $f(w, x)$ — функция, вычисляемая нейросетью. Для мониторинга обучения нейронной сети обычно используется валидационная выборка. Для ускорения обучения качество на валидационной выборке проверяется через фиксированное количество итераций или в конце эпохи.

Тестирование

Для оценки качества модели из публичной выборки выделяется набор тестовых данных, который не участвует в обучении. Если качество модели на тренировочной выборке значительно лучше, чем на тестовой, это может говорить о переобучении.

Интерфейс программы, данные и скрипт для тестирования

Необходимо реализовать две функции: `train_detector`, обучающую модель детектора, и `detect`, проводящую детектирование ключевых точек на изображениях с обученной моделью. Функция детектирования возвращает словарь размером N, ключи которого — имена файлов, а значения —

массив чисел длины 28 с координатами точек лица $[x_1, y_1, \ldots, x_{14}, y_{14}]$. Здесь N — количество изображений.

Скрипт для тестирования обучает детектор и оценивает качество детектирования путем подсчета среднеквадратичной ошибки err на изображении, приведенном к размеру 100×100 пикселей. Ошибка err на скрытой выборке конвертируется в итоговый балл:

$err \leqslant 20$ — 2 балла,

$err \leqslant 18$ — 3 балла,

$err \leqslant 16$ — 4 балла,

$err \leqslant 14$ — 5 баллов,

$err \leqslant 12$ — 6 баллов,

$err \leqslant 10$ — 7 баллов,

$err \leqslant 8$ — 8 баллов,

$err \leqslant 6$ — 9 баллов,

$err \leqslant 5$ — 10 баллов.

Для обучения алгоритма выдается публичная выборка лиц с размеченными ключевыми точками. Разрешается использовать только эти данные для обучения. Другие внешние данные и библиотеки с готовыми моделями использовать нельзя.

Программа тестируется на двух тестах. В каждом из тестов нейросеть сначала обучается с флагом `fast_train=True` в функции `train_detector`. Функция обучения с этим флагом должна работать недолго, не больше 5 минут. Для этого поставьте 1 эпоху обучения и несколько батчей. Обученная модель для тестирования не используется, этот этап необходим только для проверки работоспособности функции обучения. При работе функция не должна писать в какие-либо файлы (например, сохранять обученную модель), т.к. в проверяющей системе папка с решением монтируется только для чтения (read-only). Затем в первом тесте алгоритм тестируется на публичной выборке, во втором тесте — на скрытой выборке. Для тестирования загружается обученная модель `facepoints_model.hdf5`. Запуск функции на публичной обучающей выборке с флагом `fast_train=False` должен позволять воспроизвести сданную в `hdf5`-файле модель с небольшой погрешностью, связанной со случайностью в процессе обучения. Решения без функции обучения могут быть не засчитаны. Обратите внимание, что изображения как в тренировочной, так и в тестовой выборке могут быть черно-белыми, а не только цветными. Эту ситуацию нужно корректно обрабатывать (например, делать из одноканального изображения трехканальное путем дублирования канала).

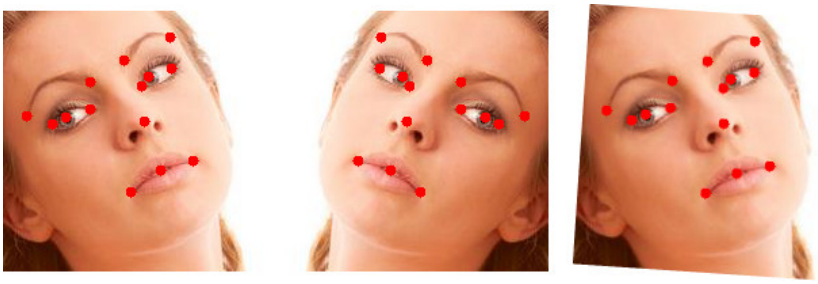
Результаты второго теста и итоговый балл скрыты до окончания срока сдачи задания. Итоговый балл считается по последней посылке с ненулевой точностью. Для уменьшения количества потребляемой памяти в Keras можно при обучении и тестировании использовать генераторы батчей (функции `model.fit_generator` и `model.predict_generator`).

Продвинутая часть

Базовой архитектуры нейросети достаточно, чтобы получить среднеквадратичную ошибку $err \leqslant 20$ и минимальный положительный балл за решение. Для повышения качества предсказания нейросети можно использовать следующие приемы:

Изменение архитектуры Поэкспериментируйте с количеством слоев и колиечством нейронов на каждом слое.

Размножение данных



Для предотвращения переобучения и увеличения точности распознавания нейронных сетей тренировочная выборка обычно размножается (data augmentation). Количество обучающих данных при этом возрастает на порядки. Данные можно размножить заранее и записать на диск или размножать в процессе обучения с использованием функций-предобработчиков. Изображения человеческих лиц можно размножить следующими случайными преобразованиями:

- зеркальное отражение относительно горизонтальной оси,
- поворот на небольшой угол,
- кадрирование с сохранением всех точек лица на изображении.

Не забывайте при этом применять преобразования и к координатам точек лица.

Batch normalization Batch normalization — метод, который позволяет повысить производительность и стабилизировать работу искусственных нейронных сетей. Суть данного метода заключается в том, что некоторым слоям нейронной сети на вход подаются данные, предварительно обработанные и имеющие нулевое математическое ожидание и единичную дисперсию.

- Достигается более быстрая сходимость моделей, несмотря на выполнение дополнительных вычислений.
- Становится возможным использование более высокого темпа обучения, так как пакетная нормализация гарантирует, что выходы узлов нейронной сети не будут иметь слишком больших или малых значений.
- Модели становятся менее чувствительны к начальной инициализации весов.

Batch normalization имеет смысл использовать до или после функции активации.

Dropout Для предотвращения переобучения рекомендуется использовать дропаут. Идея дропаута состоит в том, что во время обучения мы случайным образом обнуляем связи некоторых нейронов в сети. Дропаут для слоя нейросети определяется вероятностью p, с которой зануляется каждый нейрон слоя. Dropout ипользуют в полносвязных и сверточных слоях после функции активации.

Weight decay Для предотвращения переобучения в нейросетях можно использовать L_1 или L_2-регуляризацию. К функции потерь прибавляется слагаемое, штрафующее слишком большие значения весов:

$$Q'(w, X, Y) = Q(w, X, Y) + \frac{\lambda}{2}||w||^2.$$

Здесь λ — параметр (обычно называется затуханием весов), определяющий силу регуляризации.

Полезные ресурсы

Материалы стэнфордского курса по нейронным сетям
Dropout: A Simple Way to Prevent Neural Networks from Overfitting — статья про дропаут
Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift — статья про batch normalization