Применение сверточных нейронных сетей для удаления шума с изображений

Юров Д.А. email: [yurov.dmitry.a@gmail.com](mailto:yurov.dmitry.a@gmail.com)

Барабанов В.Ф. email: [bvf@list.ru](mailto:bvf@list.ru)

Воронежский государственный технический университет

Свёрточная нейронная сеть (англ. convolutional neural network, CNN) - специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году, нацеленная на эффективное распознавание изображений. Она относится к технологиям глубинного обучения и основывается на чередовании свёрточных слоев (англ. convolution layers) и субдискретизирующих слоев (англ. subsampling layers) или слоёв подвыборки (англ. pooling layers), имитируя алгоритм работы человеческого зрения[1]. Структура сети является однонаправленной, обратные связи не используются. Для обучения используются стандартные методы, чаще всего метод обратного распространения ошибки, а в качестве функции активации нейронов можно использовать любую подходящую функцию.

В рамках данной работы было решено использовать функцию - «выпрямитель» (англ. ReLU, rectified linear unit), которая имеет вид

и реализует простой пороговый переход в нуле. Данная функция имеет следующий вид:

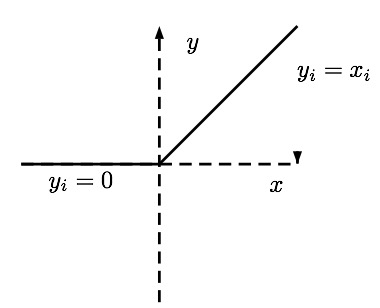


Рисунок 1. Общий вид функции активации ReLU

Данная функция хорошо подходит для использования в имплементациях в условиях ограниченных ресурсов из-за следующих особенностей:

* вычисление значения такой функции не требует такого количества ресурсов, как вычисление сигмоида или гиперболического тангенса, что позволяет использовать большее количество слоев и нейронов при меньших затратах;
* нейроны, использующие ReLU, не всегда достаточно надежны и в процессе обучения могут выходить из строя. Например, возможно такое обновление весов, при котором нейрон никогда больше не активируется. Если это произойдет, то, начиная с данного момента, градиент, проходящий через этот нейрон, всегда будет равен нулю. Количество деактивированных нейронов коррелирует со скоростью обучения и при слишком ее значении может оказаться, что до 40% нейронов никогда не активируются. В данном случае ограниченность ресурсов снижает влияние данной особенности, а фреймворк, на котором реализована сеть, способен сбрасывать веса нейронов, что, впрочем, может приводить к снижению эффективности работы.

Спроектированная сеть состоит из чередующихся слоев двух типов (всего 5 слоя) – сверточных двухмерных, так как целью является обработка двухмерных изображений, и субдискретизирующих двухмерных. Реализация данной структуры на языке Python с использованием библиотек Keras и TensorFlow выглядит следующим образом:

x = Conv2D(32, (3, 3), activation='relu', padding='same')(encoded)

x = UpSampling2D((2, 2))(x)

x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)

x = UpSampling2D((2, 2))(x)

decoded = Conv2D(1, (3, 3), activation='relu', padding='same')(x)

Здесь input\_img определяет размер тензора (под тензором в TensorFlow понимается абстрактный вектор или матрица, в которой все элементы имеют одинаковый тип), 28 на 28.

Обучение модели проходит на наборе данных, поставляемых вместе с библиотекой Keras – 60 тысяч изображений цифр от 0 до 9 различного вида. Обучение выполняется на протяжении 100 эпох (эпоха – полный цикл обработки набора данных), таким образом, было обработано порядка 6 миллионов изображений, что позволяет добиться погрешности 0.0898. Так как используется обучение без учителя, код выглядит следующим образом:

decoder = Model(input\_img, decoded)

decoder.compile(optimizer='adadelta', loss='binary\_crossentropy')

decoder.fit(x\_train\_noisy, x\_train,

epochs=1,

batch\_size=256,

shuffle=True,

validation\_data=(x\_test\_noisy, x\_test),

callbacks=[TensorBoard(log\_dir='/tmp/tb', histogram\_freq=0, write\_graph=False)])

Метод оптимизации ADADELTA – подвид метода градиентного спуска, впервые представленный в работе Мэтью Д. Цейлера[2].

Благодаря функции callback мы можем наблюдать динамику изменения погрешности неспосредственно во время обучения. Ниже представлен график изменения погрешности с каждой новой эпохой, благодаря которому можно заметить, что эффективность обучения снижается приблизительно на 90-йэпохе.

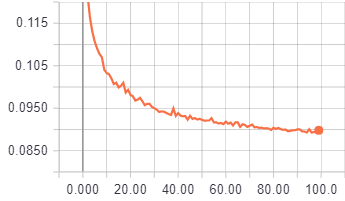


Рисунок 2. Динамика изменения погрешности.

В результате, после применения обученной сети к набору намеренно зашумленных изображений удается добиться высокой эффективности уже на сотой эпохе. На рисунке ниже в первом ряду представлены исходные изображения, во втором – намеренно зашумленные изображения, в третьем – результат обработки нейронной сетью зашумленных изображений.

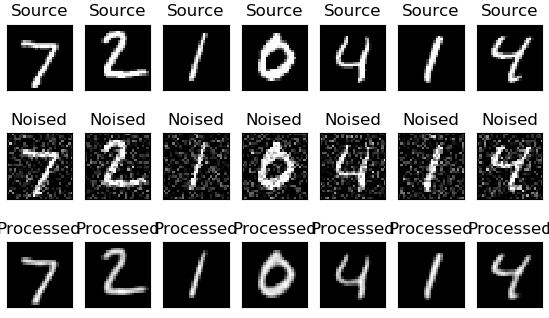


Рисунок 3. Результаты обработки данных обученной нейронной сетью

В дальнейшем данный подход будет использован для обработки полноцветных зашумленных изображений с большим разрешением.

Литература:

1. Sebastian Raschka Python Machine Learning / Sebastian Raschka - Birmingham, UK - Packt Publishing, 2015, 425 c.

2. ADADELTA: An Adaptive Learning Rate Method. URL [Электронный ресурс] Режим доступа: https://arxiv.org/abs/1212.5701 – Англ. 08.09.2017.