Содержание

ГЛАВА 3

Несколько интересных проектов

Не играя, не научишься.

В этой главе мы исследуем ряд оригинальных идей. Они не связаны с пониманием основ нейронных сетей, поэтому не смущайтесь, если кое-что будет вам непонятно.

Поскольку глава предназначена больше для развлечения, мы ускорим темп, хотя там, где это потребуется, будут даваться простые пояснения.

Собственный рукописный текст

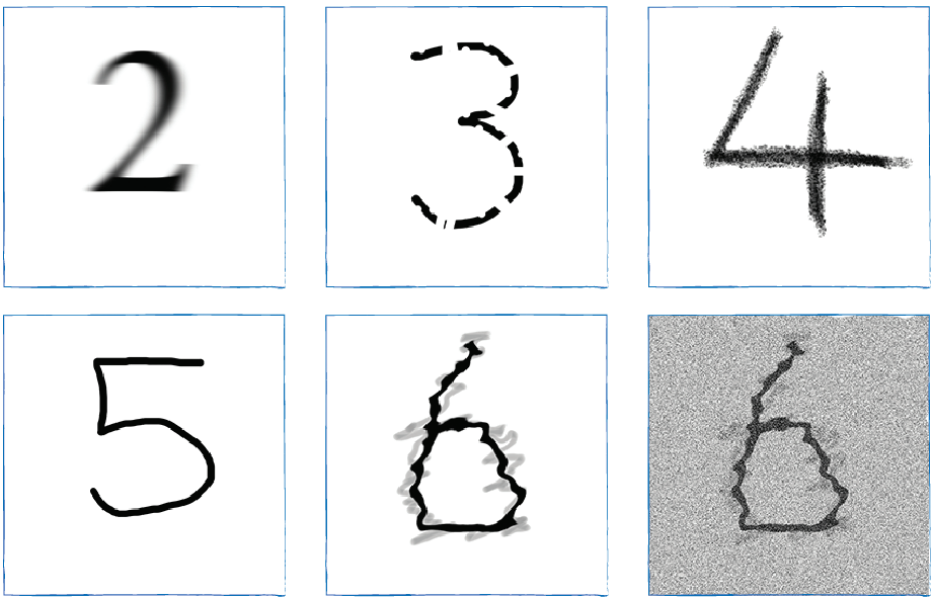
В главе 2 мы распознавали изображения рукописных цифр из базы данных MNIST. А почему бы не использовать собственный рукописный текст?

В этом эксперименте мы создадим свою тестовую базу данных, используя собственноручно написанные цифры. Кроме того, мы попытаемся сымитировать различные виды почерка, а также зашумленные и прерывистые изображения, чтобы посмотреть, насколько хорошо с ними справится наша нейронная сеть.

Для создания изображений можно использовать любой графический редактор. Вы не обязаны делать это с помощью дорогостоящей программы Photoshop. Вполне подойдет альтернативный вариант в виде бесплатной программы с открытым исходным кодом GIMP (www.gimp.org), доступной для компьютеров с операционными системами Windows, Mac и Linux. Вы даже можете написать цифры карандашом на листе бумаги, а затем отсканировать или сфотографировать их с помощью смартфона или фотокамеры. Единственным требованием является то, что изображение цифры должно быть

квадратным (длина равна ширине) и представленным в формате PNG. Этот формат можно выбрать в списке допустимых форматов большинства графических редакторов при выполнении команды меню Файл⇨Сохранить как или Файл⇨Экспорт.

Вот как выглядят некоторые из подготовленных мною изображений.



Цифру “5” я написал маркером. Цифра “4” написана мелом. Цифру “3” я написал маркером, но намеренно сделал линию прерывистой. Цифра “2” взята из типичного газетного или книжного шрифта, но немного размыта. Цифра “6” как бы покрыта рябью, словно мы видим ее как отражение на поверхности воды. Последнее изображение совпадает с предыдущим, но в него добавлен шум, чтобы намеренно затруднить распознавание цифры нейронной сетью.

Все это забавно, но здесь есть и серьезный момент. Ученых изумляет поразительная способность человеческого мозга сохранять функциональность даже после того, как он испытал повреждения. Предполагают, что в нейронной сети приобретенные знания распределяются между несколькими связями, а потому даже в условиях повреждения некоторых связей остальные связи могут успешно

функционировать. Это также означает, что они могут достаточно хорошо функционировать и в случае повреждения или неполноты входного изображения. Таковы возможности нашего мозга, и именно это мы хотим протестировать с посеченной цифрой “3”, изображенной на приведенной выше иллюстрации.

Прежде всего, мы должны создать уменьшенные версии PNG-изображений, масштабировав их до размера 28×28 пикселей, чтобы привести в соответствие с использовавшимися ранее данными MNIST. Для этого можете использовать свой графический редактор.

Для чтения и декодирования данных из распространенных форматов изображений, включая PNG, мы вновь используем библиотеки Python. Взгляните на следующий простой код.

```
import scipy.misc
img_array = scipy.misc.imread(image_file_name, flatten=True)

img_data = 255.0 - img_array.reshape(784)
img_data = (img_data / 255.0 * 0.99) + 0.01
```

Функция `scipy.misc.imread()` поможет нам в получении данных из файлов изображений, таких как PNG- или JPG-файлы. Чтобы использовать библиотеку `scipy.misc`, ее необходимо импортировать. Параметр `flatten=True` превращает изображение в простой массив чисел с плавающей запятой и, если изображение цветное, переводит цветовые коды в шкалу оттенков серого, что нам и надо.

Следующая строка преобразует квадратный массив размерностью 28×28 в длинный список значений, который нужен для передачи данных нейронной сети. Ранее мы делали это не раз. Новым здесь является вычитание значений массива из 255,0. Это необходимо сделать, поскольку обычно коду 0 соответствует черный цвет, а коду 255 — белый, но в наборе данных MNIST используется обратная схема, в связи с чем мы должны инвертировать цвета для приведения их в соответствие с соглашениями, принятыми в MNIST.

Последняя строка выполняет уже знакомое вам масштабирование данных, переводя их в диапазон значений от 0,01 до 1,0.

Образец кода, демонстрирующего чтение PNG-файлов, доступен на сайте GitHub по следующему адресу:

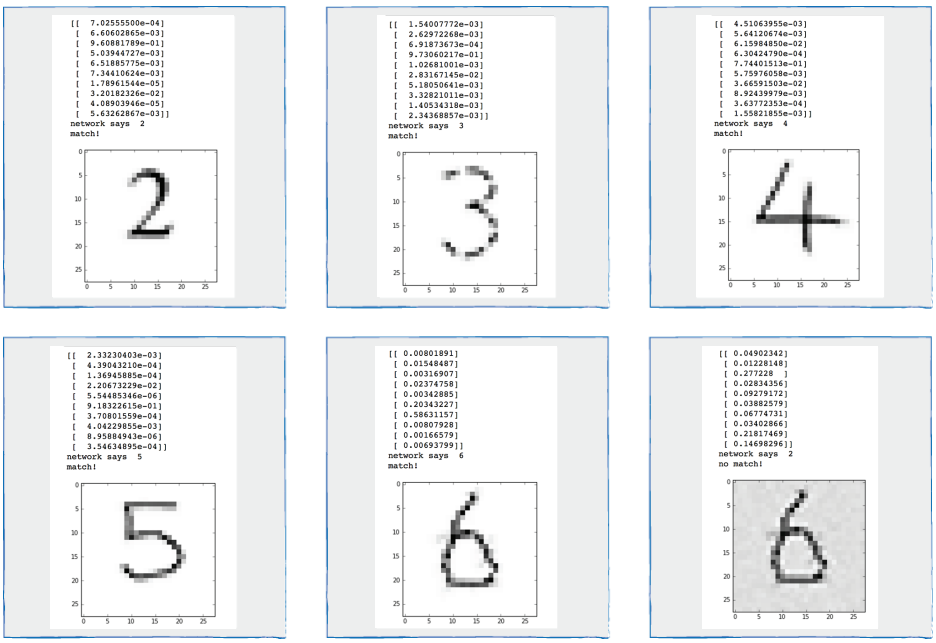
https://github.com/makeyourownneuralnetwork/makeyourownneuralnetwork/blob/master/part3_load_own_images.ipynb

Нам нужно создать версию программы, использовавшейся ранее для создания базовой нейронной сети и ее обучения с помощью набора данных MNIST, но теперь мы будем тестировать программу с использованием набора данных, созданного на основе наших изображений.

Новая программа доступна на сайте GitHub по следующему адресу:

https://github.com/makeyourownneuralnetwork/makeyourownneuralnetwork/blob/master/part3_neural_network_mnist_and_own_data.ipynb

Работает ли она? Конечно, работает! Следующая иллюстрация демонстрирует результаты опроса сети с использованием наших изображений.



Как видите, нейронной сети удалось распознать все изображения, включая намеренно поврежденную цифру “3”. Не удалось распознать лишь зашумленную цифру “6”.

Проведите эксперименты с подготовленными вами изображениями, включая изображения рукописных цифр, и убедитесь в том, что нейронная сеть действительно работает.

Также проверьте, как далеко можно зайти с испорченными или искаженными цифрами. Вы будете поражены гибкостью нашей нейронной сети.

Проникнем в "мозг" нейронной сети

Нейронные сети полезны при решении тех задач, для которых трудно придумать простой и четкий алгоритм решения с фиксированными правилами. Представьте себе только, каким должен быть набор правил, регламентирующих процедуру определения рукописной цифры, представленной изображением! Вы согласитесь, что создать такие правила довольно нелегко, а вероятность того, что наши попытки окажутся успешными, весьма мала.

Загадочный черный ящик

Завершив обучение нейронной сети и проверив ее работоспособность на тестовых данных, вы, по сути, получаете **черный ящик**. Фактически вы не знаете, как вырабатывается ответ, но все же он вырабатывается!

Незнание внутреннего механизма не всегда является проблемой, если главное для вас — ответы и вас не интересует, каким образом они получены. Но именно этим недостатком страдают рассматриваемые методы машинного обучения: обучение не всегда означает понимание сути задачи, которую черный ящик научился решать.

Давайте разберемся, можем ли мы заглянуть внутрь нашей простой нейронной сети и увидеть, чему она научилась, чтобы визуализировать знания, приобретенные ею в процессе тренировки.

Мы могли бы проанализировать весовые коэффициенты, в которых, в конце концов, и сосредоточено все, чему учится сеть. Но вряд ли эти данные будут нести в себе полезную для нас информацию, особенно если учесть, что нейронная сеть работает таким образом, чтобы распределить то, чему она учится, по различным связям. Это обеспечивает устойчивость сети к повреждениям, как это свойственно биологическому мозгу. Маловероятно, что удаление одного или

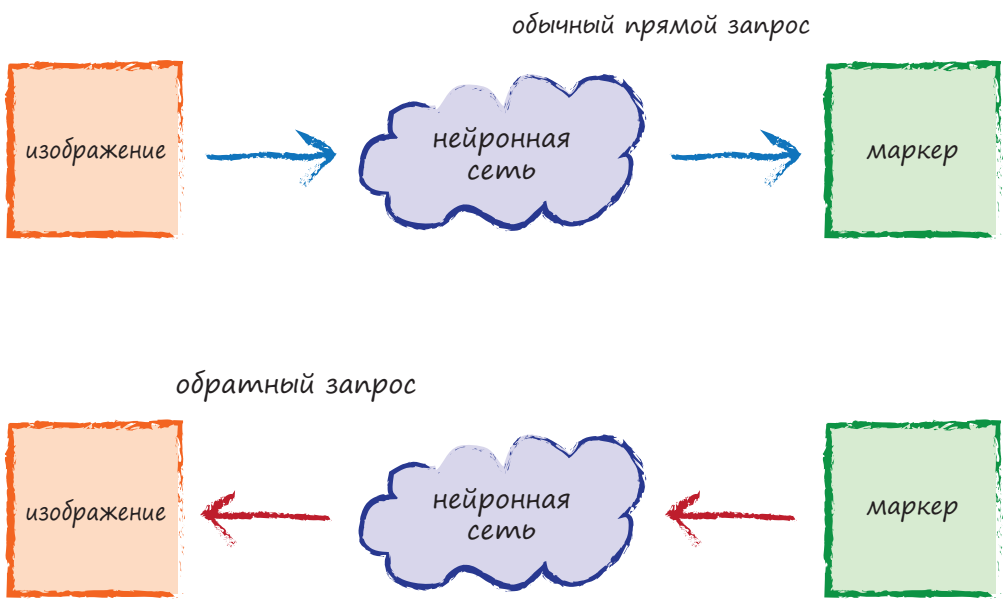
даже большего количества узлов полностью лишит сеть возможности нормально работать.

Рассмотрим одну безумную идею.

Обратные запросы

Обычно мы задаем тренированной сети вопрос, и она выдает нам ответ. В нашем примере вопросом является изображение рукописной цифры. Ответом является маркер, представляющий число из диапазона значений от 0 до 9.

А что, если развернуть этот процесс наоборот? Что, если мы подадим маркер на выходные узлы и проследим за распространением сигнала по уже натренированной сети в обратном направлении, пока не получим на входных узлах исходное изображение? Следующая диаграмма иллюстрирует процесс распространения обычного запроса и описанный только что процесс распространения **обратного запроса**.



Мы уже знаем, как распространять сигналы по сети, сглаживая их весовыми коэффициентами и рекомбинируя на узлах, прежде чем применять к ним функцию активации. Весь этот механизм работает также для сигналов, распространяющихся в обратном направлении,

за исключением того, что в этом случае используется обратная функция активации. Если **y = f(x)** — функция активации для прямых сигналов, то ее обратная функция — **x = g(y)**. Для логистической функции нахождение обратной функции сводится к простой алгебре:

$$y = 1/(1 + e^{-x})$$
$$1 + e^{-x} = 1/y$$
$$e^{-x} = (1/y) - 1 = (1-y)/y$$
$$-x = \ln[(1-y)/y]$$
$$x = \ln[y/(1-y)]$$

Эта функция называется **logit**, и библиотека Python `scipy.special` предоставляет ее как **scipy.special.logit()**, точно так же, как и логистическую функцию **scipy.special.expit()**.

Прежде чем использовать обратную функцию активации `logit()`, мы должны убедиться в допустимости сигналов. Что это означает? Вы помните, что сигмоида принимает любое значение и возвращает значение из диапазона от 0 до 1, исключая граничные значения. Обратная функция должна принимать значения из того же самого диапазона — от 0 до 1, исключая сами значения 0 и 1, — и возвращать значение, которое может быть любым положительным или отрицательным числом. Для этого мы просто берем все значения в слое, к которым собираемся применить функцию `logit()`, и приводим их к допустимому диапазону. В качестве такового я выбрал диапазон чисел от 0,01 до 0,99.

Соответствующий код доступен на сайте GitHub по следующему адресу:

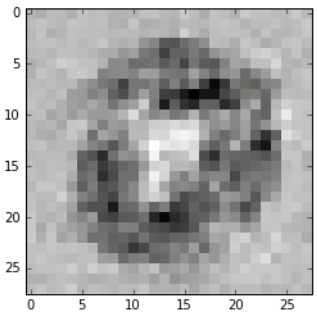
https://github.com/makeyourownneuralnetwork/makeyourownneuralnetwork/blob/master/part3_neural_network_mnist_backquery.ipynb

Маркер “0”

Посмотрим, что произойдет, если выполнить обратный запрос для маркера “0”, т.е. мы предоставляем выходным узлам значения, равные 0,01, за исключением первого узла, соответствующего маркеру “0”, которому мы предоставляем значение 0,99. Иными словами,

мы передаем на выходные узлы массив чисел `[0.99, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01]`.

Ниже показано изображение, полученное на входных узлах.



Это уже интересно! Так "видит" картинку "мозг" нашей нейронной сети.

Как расценивать полученное изображение? Как его следует интерпретировать?

Основное, что сразу же бросается в глаза, — округлая форма изображения. Это соответствует истине, поскольку мы интересовались у нейронной сети, каков тот идеальный вопрос, ответом на который будет "0".

Кроме того, на изображении можно выделить темные, светлые и серые области.

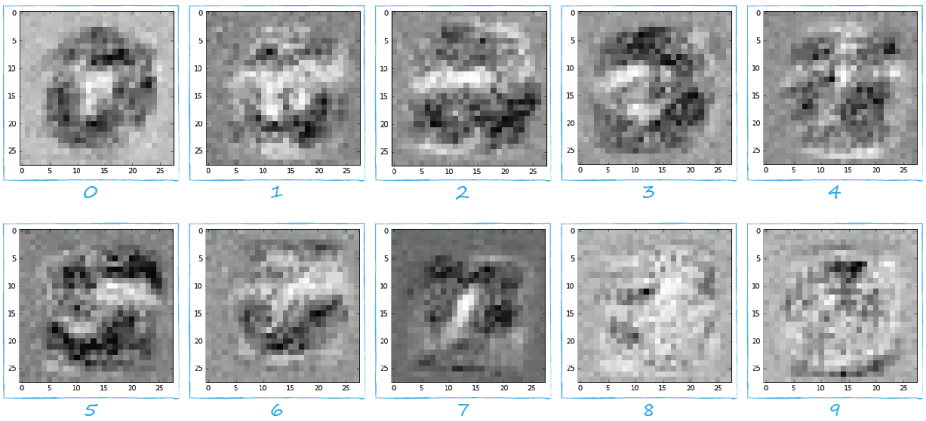
- **Темные области** — это те части искомого изображения, которые, если обвести их маркером, свидетельствуют в пользу того, что ответом следует считать "0". Форма их контура действительно напоминает цифру "0".
- **Светлые области** — это те участки искомого изображения, которые следует считать не закрашенными, если остановиться на версии, что ответом является "0". Это предположение представляется разумным, поскольку эти участки располагаются внутри контура цифры "0".
- **Серые области** в основном не несут никакой информации для нейронной сети.

Итак, в общих чертах нам удалось достигнуть некоторого понимания того, как именно сформировалось умение сети классифицировать изображения с маркером “0”.

Нам выпала редкостная удача заглянуть внутрь нейронной сети, поскольку в случае сетей с более сложной структурой и бо́льшим количеством слоев, а также в случае более сложных задач вряд ли можно рассчитывать на столь же легкую интерпретацию результатов.

Остальные изображения

Ниже представлены результаты обратного запроса для остальных случаев.



Вот это да! Мы вновь получили довольно-таки интересные изображения. Они словно представляют собой снимки мозга нейронной сети, полученные с помощью компьютерной томографии.

По поводу этих изображений можно сделать некоторые замечания.

- Маркер “7” довольно отчетливо распознается как цифра “7”. Если вы обведете карандашом темные пиксели изображения, то получите достаточно наглядное подтверждение этого. Также заметно выделяется “белая” область, где не должно быть закрашенных элементов. Оба этих фактора вместе указывают на то, что в данном случае мы имеем дело с цифрой “7”.

- То же самое справедливо для маркера “3”, поскольку здесь налицо те же два фактора: схожесть контура темных пикселей, если обвести его карандашом, с цифрой “3” и наличие белых областей в тех местах, где они и должны быть в цифре “3”.
- Маркеры “2” и “5” интерпретируются аналогичным образом.
- Случай маркера “4” интересен наличием фигуры, напоминающей четыре квадранта, и белых областей.
- Изображение, полученное для маркера “8”, очень напоминает снеговика, “голова и туловище” которого сформированы двумя белыми областями, что в целом соответствует цифре “8”.
- Изображение для маркера “1” ставит нас в тупик. Создается впечатление, что сеть уделила больше внимания тем областям, которые должны оставаться белыми, чем тем, которые должны быть закрашены. Ну что ж, это то, чему научилась сеть на примерах.
- Изображение для маркера “9” вообще неразборчиво. В нем нет четко выделенных темных областей и каких-либо фигур, образованных белыми областями. Это результат того, чему сеть научилась на предоставленных ей примерах, обеспечивая в целом точность на уровне 97,5%. Глядя на данное изображение, напрашивается вывод, что, возможно, сеть нуждается в дополнительных тренировочных примерах, которые помогли бы ей лучше распознавать образцы цифры “9”.

Итак, вам была предоставлена увлекательная возможность заглянуть во внутренний мир нейронной сети и, что называется, увидеть, как работает ее мозг.

Создание новых тренировочных данных: вращения

Оценивая тренировочные данные MNIST, следует сказать, что они содержат довольно богатый набор рукописных начертаний цифр. В нем встречается множество видов почерка и стилей, причем как хороших, так и плохих.

Нейронная сеть должна учиться на как можно большем количестве всевозможных вариантов написания цифр. Удачно то, что в этот

набор входит также множество форм написания цифры “4”. Одни из них искусственно сжаты, другие растянуты, некоторые повернуты, в одних верхушка цифры разомкнута, в других сомкнута.

Разве не будет полезно создать дополнительные варианты написания и использовать их в качестве тренировочных примеров? Как это сделать? Нам трудно собрать коллекцию из тысячи дополнительных примеров рукописного написания той или иной цифры. Вернее, мы могли бы попытаться, но это стоило бы нам огромных усилий.

Но ведь ничто не мешает нам взять существующие примеры и создать на их основе новые, повернув цифры по часовой стрелке или против, скажем, на 10 градусов. В этом случае мы могли бы создать по два дополнительных примера для каждого из уже имеющихся. Мы могли бы создать гораздо больше примеров, вращая образцы на разные углы, но мы ограничимся углами +10 и −10 градусов, чтобы посмотреть, как работает эта идея.

И вновь большую помощь в этом нам окажут расширения и библиотеки Python. Функция `scipy.ndimage.interpolation.rotate()` поворачивает изображение, представленное массивом, на заданный угол, а это именно то, что нам нужно. Описание функции можно найти здесь:

```
https://docs.scipy.org/doc/scipy-0.16.0/reference/
generated/scipy.ndimage.interpolation.rotate.html
```

Вспомните, что наши входные значения представляют собой одномерный список длиной 784 элемента, поскольку мы спроектировали нашу нейронную сеть таким образом, чтобы она принимала длинный список входных сигналов. Мы должны реорганизовать этот список в массив размерностью 28×28, чтобы повернуть изображение, а затем проделать обратную операцию по преобразованию массива в список из 784 входных сигналов, которые мы подадим на вход нейронной сети.

В приведенном ниже коде показан пример использования функции `scipy.ndimage.interpolation.rotate()` в предположении, что у нас уже имеется массив **`scaled_input`**, о котором шла речь ранее.

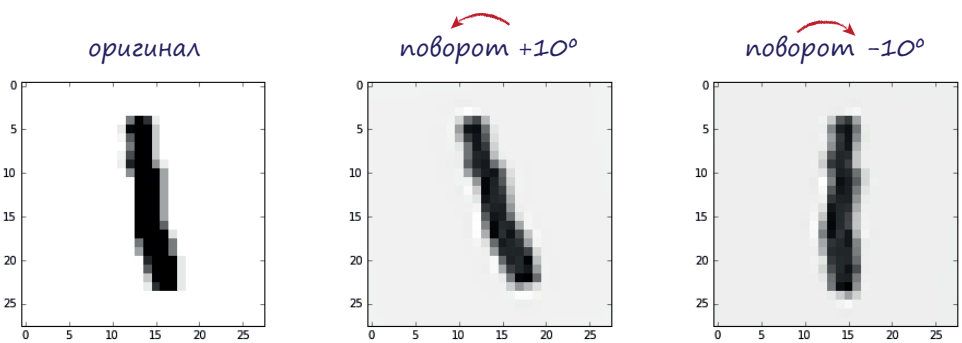
```
# создание повернутых на некоторый угол вариантов изображений

# повернуть на 10 градусов против часовой стрелки
inputs_plus10_img =
```

```
↳scipy.ndimage.interpolation.rotate(scaled_input.reshape(28,28),
↳10, cval=0.01, reshape=False)
# повернуть на 10 градусов по часовой стрелке
inputs_minus10_img =
↳scipy.ndimage.interpolation.rotate(scaled_input.reshape(28,28),
↳-10, cval=0.01, reshape=False)
```

В этом коде первоначальный массив `scaled_input` преобразуется в массив размерностью 28×28. Параметр `reshape=False` сдерживает излишнюю рьяность библиотеки в ее желании быть как можно более полезной и сжать изображение таким образом, чтобы после вращения все оно уместилось в массиве и ни один его пиксель не был отсечен. Параметр `cval` — это значение, используемое для заполнения элементов массива, которые не существовали в исходном массиве, но теперь появились. Мы откажемся от используемого по умолчанию значения 0,0 и заменим его значением 0,01, поскольку во избежание подачи на вход нейронной сети нулей мы используем смещенный диапазон входных значений.

Запись 6 (седьмая по счету) малого тренировочного набора MNIST содержит рукописное начертание цифры “1”. Вот как выглядят ее исходное изображение и две его повернутые вариации, полученные с помощью нашего кода.



Результаты очевидны. Версия исходного изображения, повернутая на +10 градусов, является примером почерка человека, “заваливающего” текст влево. Еще более интересна версия оригинала, повернутая на −10 градусов, т.е. по часовой стрелке. Эта версия

располагается даже ровнее по сравнению с оригиналом и в некотором смысле более представительна в качестве изображения для обучения.

Создадим новый блокнот Python с имеющимся кодом нейронной сети, но с дополнительными тренировочными примерами, созданными путем поворота исходных изображений на 10 градусов в обе стороны. Этот код доступен на сайте GitHub по следующему адресу:

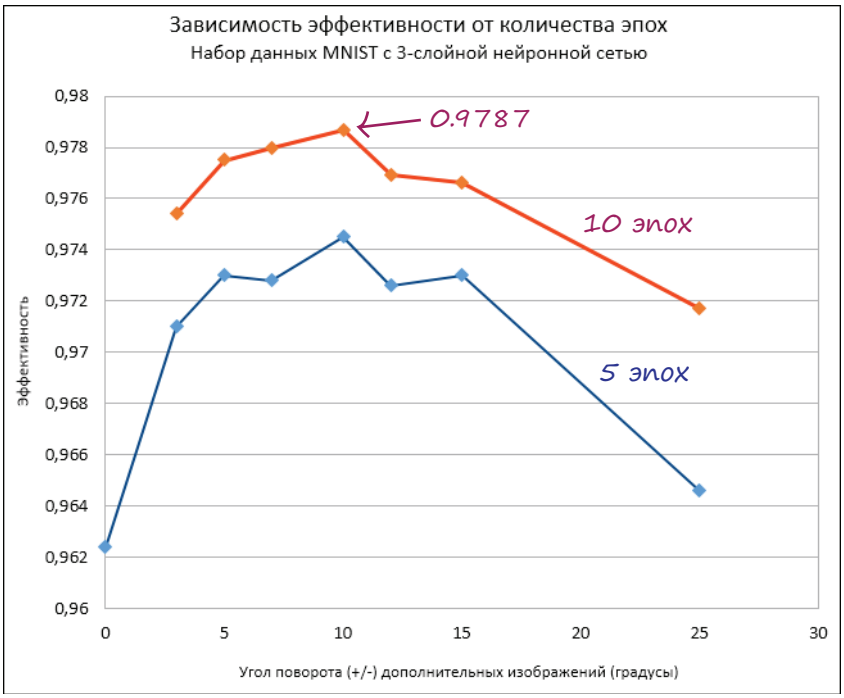
https://github.com/makeyourownneuralnetwork/makeyourownneuralnetwork/blob/master/part3_neural_network_mnist_data_with_rotations.ipynb

Запуск этого кода с использованием коэффициента обучения 0,1 и всего лишь одной тренировочной эпохи дает показатель эффективности, равный **0,9669**. Это значительное улучшение по сравнению со значением 0,954, полученным без дополнительных повернутых изображений. Такой показатель уже попадает в число лучших из тех, которые опубликованы на сайте Яна Лекуна (http://yann.lecun.com/exdb/mnist/).

Запустим серию экспериментов, изменяя количество эпох, чтобы проверить, можно ли еще больше улучшить полученный показатель эффективности обучения. Кроме того, **уменьшим коэффициент обучения до 0,01**, поскольку, предоставив гораздо больше тренировочных данных и тем самым увеличив общее время обучения, мы можем позволить себе более осторожные шаги обучения меньшей величины.

Не забывайте о том, что мы не ожидаем получить точность распознавания 100%, поскольку, вероятно, существует естественный предел точности, обусловленный спецификой архитектуры нейронной сети или полнотой тренировочных данных, в связи с чем мы вряд ли можем получить точность выше примерно 98%. Под “спецификой архитектуры нейронной сети” здесь подразумевается выбор количества узлов в каждом слое, количества скрытых слоев, функции активации и т.п.

Ниже представлены графики, отражающие зависимость эффективности нейронной сети от угла поворота дополнительных тренировочных изображений. Для сравнения показана также точка данных, соответствующая отсутствию дополнительных примеров.



Как видите, для пяти эпох наилучший результат равен **0,9745**, или 97,5% точности. Это явное улучшение по сравнению с предыдущим примером.

Также следует отметить, что с увеличением угла поворота изображения точность падает. Это вполне объяснимо, поскольку при больших углах поворота результирующие изображения фактически вообще не представляют цифры. Представьте цифру “3”, повернутую на 90 градусов, т.е. положенную на бок. Это будет вовсе не тройка. Поэтому, добавляя тренировочные примеры с чрезмерно большими углами поворота, мы снижаем качество тренировки, потому что добавляемые примеры являются ложными. Пожалуй, оптимальным углом поворота для дополнительных тренировочных изображений является угол 10 градусов.

Для десяти эпох рекордное пиковое значение точности составляет **0,9787**, или почти **98%**! Это поистине ошеломляющий результат для простой нейронной сети такого рода. Не забывайте о том, что мы не использовали никаких изощренных математических трюков в отношении нейронной сети или данных, как это делают некоторые

люди. Мы придерживались предельной простоты и тем не менее достигли результатов, которыми по праву можем гордиться.

Отличная работа!