ИСПОЛЬЗОВАНИЯ НЕЙРОННЫХ СЕТЕЙ В ВЕБ ПРИЛОЖЕНИИ

Калиновский М.Г.
Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь
Камлач П.В. – канд. техн. наук, доцент

Данная работа описывает процесс создания и обучения нейронной сети, а также портирования ее для использования в веб приложении.

В мире существует огромное количество задач в этом мире, которые человек пытается решить. Одним из стандартных решений какой-либо проблемы является создание алгоритма. Алгоритмы были разделены по классам сложности. NP – полные задачи являются одним из самым известных классов, которых показал, что для некоторых задачи на данный момент невозможно найти эффективного алгоритма. Для других же задач выражения решения с помощью алгоритмов вообще является невозможным, например, распознавание изображений.

Еще одно направление, которое бурно развивалось в 20 веке – искусственный интеллект. Таким образом были предложены понятия нейронной сети и математическая модель для ее представления. Из-за определённых трудностей произошел спад к этой теме, но к которому вернули в 2007, когда Джеффри Хинтоном создал алгоритм глубокого обучения многослойный нейронных сетей. На текущий момент эксперименты с нейронными сетями активно проводятся, а некоторые достижение, как распознавание классов объектов на изображении, уже активно используются, например, в камерах. Но серьезных нативных программ для персональных компьютеров, использующих нейронный сети редки, в связи с бурным развитием нейронных сетей и трудоемкостью создания программ для персональных компьютеров.

При создании первого IPhone у разработчиков не было возможности писать нативные приложения для него. Стив Джобс, говорил, чтоб разработчики использовали веб технологии для создания приложений. Во время выпуска первого IPhone веб технологии были сильно ограничены, чтоб в дальнейшем заставило Apply добавить возможность создавать нативные приложения. Сейчас браузеры стали полноценной средой выполнения кода, для телефонов говорят про прогрессивные веб-приложения, а технология WebAssembly позволяет запускать нативные приложения, написанные для персональных запускать в браузере практически на любом устройстве. Что облегчает возможность создания серьёзных приложений, использующих нейронные сети.

В данной статье будет продемонстрирована возможность использовать веб технологий для создания приложения использующий нейронные сети. Для этого была выбрана одна из простых задач в плане решения и реализации с помощью нейронных сетей – распознавание цифры на картинке. Также имеется достаточно большая база данных, MNIST, для обучения нейронной сети.

Сама нейронная сеть будет состоять из 3 слоев. Каждый слой состоит из нейронов, которые связаны между собой. Эта связь выражается в виде весовых коэффициентов, показывающих степень значимости между этими нейронами. Другим числом, который показывает уже значимость самого нейрона является смещение. Оно определенно для каждого нейрона отдельно. Так образом математически нейрон, который принимает неограниченное количество входные данный и выдает один ответ, можно представить в виде:

$$\sum_j w_j x_j + b \tag{1}$$

где w_j – это вес для j связи; x_j – это j входное значение; b – смещение нейрона.

В данном случает ответ линейно зависит от входных данных. Но можно использовать разные функции для активации нейрона. Одной из первых использовали ступенчатую функцию и такой нейрон называет перцептроном. Ни линейная функция, ни ступенчатая не является хорошим выбор для обучения нейронной сети, так как небольшое изменение, как входных данных, так и весов и отклонением, приводят к большим изменяем ответа. Удобней использовать функцию сигмоид, которая возвращает вещественное число от 0 до 1.

Входной слой состоит из 784 нейронов по числу пикселей, которое входит изображении цифры. На вход приходит число от 0 до 1, которая показывает, является ли пиксель частью изображения цифры или же это фон. Всего будет один скрытый слой состоящий из 30 нейронов. Скрытым называется слой, который находится между входным и выходным слоями. Выходной слой состоит из 10 нейронов, каждый из которых возвращает число от 0 до 1, и чем выше это число, тем больше нейронная сеть уверена, что это ответ.

Для компиляции кода нейронной сети в WebAssembly, который запускается в браузере используется Emscripten. Основным языком для разработки нейронных сетей является Python, но пока

WebAssembly не поддерживает такие вещи, как сборщик мусора, поэтому эффективно можно использовать языки с мануальным управлением памятью: C, C++ или Rust. Так что в данной работе был взят код на Python, который использовал использует математическую библиотеку NumPy, и переписан на C. Такой код оказался менее эффективный в плане быстродействия, чем Python с уже оптимизированными библиотеками. В таком случае выгодней компилировать Python в WebAssembly, когда появится такая возможность. До тех пор можно экспериментировать и обучать нейронные сети на Python, затем генерировать файл с уже готовыми весовыми коэффициентами и смещениями, например, в формате JSON, и на основе этого файла уже код на C в браузере будет создавать сеть и использовать ее.

Начальное значения для обучения нейронной сети важны, так как уменьшают время обучения и позволяют получить более высокие показатели. Одним из самых простых способов и эффективныъ выбора начальных значений является использования случайных чисел. При обучение нейронную сеть можно представить в виде функцию от множества элементов. Для которой нужно найти веса и смешения, дающий максимальный результат, то есть нужно найти желательно глобальный экстремум функции. При очередной итерации мы вычисляем разницу между полученным ответом и правильным, вычисляем градиент, который изменяет весовые коэффициенты и отклонения на небольшое значение в сторону экстремума функции. После этого записав полученные веса и смещения в файл, написать на C функции для создания нейронная сети и функция, которая будет высчитывать функцию активацию для каждого нейрона и передавать ответ на следующий слой. Для этого компилируем код командой: emcc –O2 -o main.html -g -s "EXPORTED_FUNCTIONS=['_create_mnist_data', '_create_network', '_feedforward', '_SGD']" -s EXTRA_EXPORTED_RUNTIME_METHODS='["ccall", "cwrap"]'. Где параметры: “-O2” показывает степень оптимизации; “-o main.html” – показывает выходной файл; "EXPORTED_FUNCTIONS” – функции, которые должны быть сохранены при компиляции даже, если они не используются; “EXTRA_EXPORTED_RUNTIME_METHODS” – позволяет добавить функции, которые предоставляет Emscripten и дает удобства работы с скомпилированным кодом. Также в команду нужно передать путь к файлу с C кодом.

После компиляции можно создать отдельный скрипт на JavaScript, который будет загружать конфигурационный файл, вызывать скомпилированные функции отвечать за визуальную часть. Для демонстрации было создано небольшое веб приложение, которое загружает тестовые картинки с цифрами, позволяет выбрать картинку, которая выводится вместе с ответом нейронной сети и правильным ответом (рисунок 1). Готовый код можно найти по ссылке: https://github.com/malekylik/web-neural-networks.

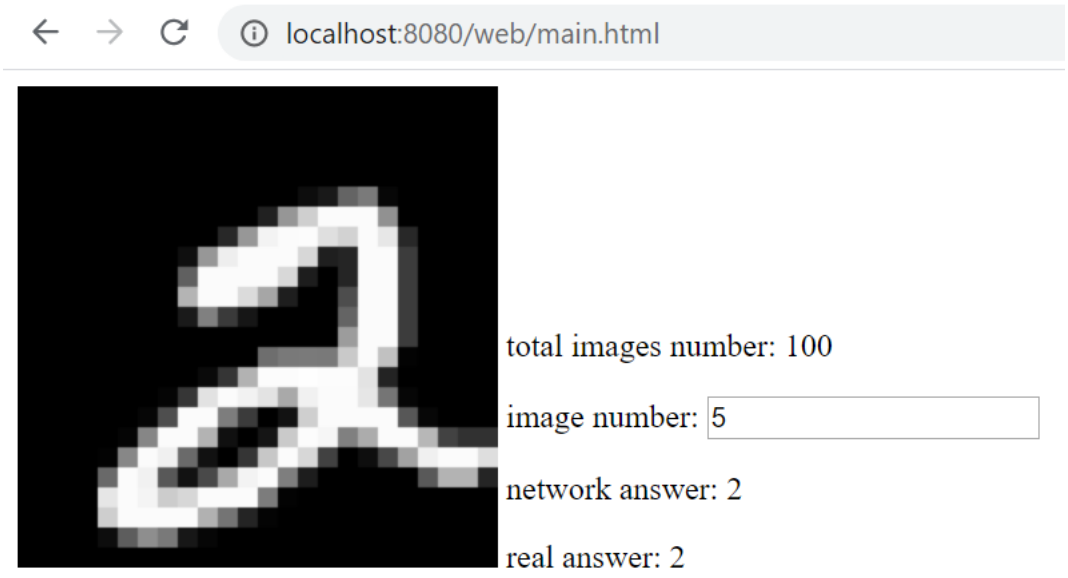

Рисунок 1 – Пример использования нейронной сети для определения цифры на картинке запущенное в Google Chrome

Список использованных источников:

1. Neural Networks and Deep Learning / Michael Nielsen [Электронный ресурс]. – Режим доступа: http://neuralnetworksanddeeplearning.com/index.html.

2. Emscripten [Электронный ресурс]. – Режим доступа: https://emscripten.org.