СРАВНЕНИЕ МЕТОДОВ СЖАТИЯ МОДЕЛИ НЕЙРОННОЙ СЕТИ ПРИ ИСПОЛЬЗОВАНИИ НА МИКРОКОНТРОЛЛЕРЕ

Иванов Е.А.[1], Мамонова Т.Е.[2]
[1]Томский Политехнический Университет, ИШИТР, А3–36, eai13@tpu.ru
[2]Томский Политехнический Университет, к.т.н., доцент ОИТ, ИШИТР, stepte@tpu.ru

Аннотация

В работе представлено практическое применение методов оптимизации модели нейронной сети для применения на малопроизводительном устройстве (микроконтроллере). Описываются следующие методы квантования модели нейронной сети: после обучения, дообучения с учетом квантования, а также прореживания. Полученные модифицированные модели нейронной сети проходят оценку скорости обсчета на базе микроконтроллера STM32F405.

Ключевые слова: квантование, микроконтроллер, нейронная сеть, методы сжатия нейронной сети.

Введение

Нейронные сети повсеместно внедряются в повседневную жизнь и претерпевают переход с габаритных платформ на более компактные устройства. Это позволяет создавать носимые аппараты с пониженным энергопотреблением и малой стоимостью. Однако, данные устройства являются менее производительными, что вынуждает разработчиков прибегать к оптимизации структуры нейронной сети. Специализированные алгоритмы позволяют не только снизить требуемый объем памяти устройства, но и повысить скорость обсчета модели за счет сжатия модели и использования эффективных средств обсчета нейронной сети.

Целью данной работы является исследование современных методов оптимизации количества занимаемой памяти и скорости вычисления при использовании нейросетевых моделей

Описание исследования

Для исследования используется модель нейронной сети, обученная распознавать рукописные цифры. Обучение проводилось по данным из базы MNIST [1]. Модель была подвергнута трем методам оптимизации: квантование после обучения, обучение с учетом квантования, а также прореживание модели. По итогу каждого способа оптимизации было произведено сопоставление точности, занимаемого объема памяти, а также скорости обсчета.

В качестве целевой платформы использовался микроконтроллер STM32F405, поскольку данная модель обладает достаточным количеством оперативной и постоянной памяти для хранения получаемых в ходе экспериментов моделей. На базе данного микроконтроллера была проведена оценка скорости обсчета нейронных сетей для каждого варианта оптимизации. Ядро микроконтроллера было настроено на частоту работы 160 МГц. На базе данного микроконтроллера имеется аппаратный модуль вычисления чисел с плавающей точкой. Модели, используемые для данного микроконтроллера, будут размещены в постоянной памяти, объем которой у данного микроконтроллера (1 МБ) позволяет экспериментировать с размером используемой нейронной сети. Встроенного количества оперативной памяти (192 кБ) с запасом хватает для обсчета нейросетевой модели.

Обучение исходной модели нейронной сети

Обучение исходной модели производилось на основе базы изображений рукописных цифр MNIST. Для исследования была выбрана существующая архитектура нейронной сети [2]:

1) входной слой размера 28 на 28, по размеру изображений;
2) слой свертки с размером ядра свертки 3 на 3 и функцией активации ReLU;
3) слой подвыборки по максимальному значению с шагом 2 в каждом измерении;
4) выходной полносвязный слой размера 10 на 1.

Общий размер выборки составил 70000 изображений. Данные для обучения и валидации были разбиты в пропорциях 6 к 1 соответственно. Обучение производилось в 3 эпохи. Результирующая точность составила 97,95 %. Размер исходной модели составил 84 896 байт.

Квантование после обучения

При квантовании модели после обучения существует обширное количество опций. Базовый подход квантования заключается в снижении разрешения весов модели [3]. Такой подход позволяет проводить сокращение общего объема модели, а также увеличение производительности в случае использования конечных значений целочисленного типа.

Общепринятым форматом весов модели при обучении является формат float32, который использует 4 байта для хранения своего значения. Для возможности снижения количества занимаемой памяти необходимо перейти на представление данных меньшей разрядности – float16, uint8.

Использование float16 требует вычислений с плавающей точкой, что поддерживается не всеми платформами. Данный подход используется в тех случаях, когда модель нейронной сети крайне чувствительна к точности значений весов. При использовании данного подхода к квантованию результирующая модель сжимается в немногим меньше, чем в 2 раза, что следует из соотношения разрядностей форматов данных (float32 требует 4 байта для хранения, в то время как для хранения float16 требуется 2 байта).

Использование форматов представления данных uint8 позволяет сократить вычислительную нагрузку на ЦП конечной платформы, так как отсутствует необходимость проведения вычислений с плавающей точкой. Однако, по сравнению с float16, данный формат хуже отражает изначальное значение весов, что может сказаться на точности для чувствительных к данным показателям моделей. Помимо увеличения скорости обсчета нейронной сети, формат uint8 позволяет сократить объем используемой моделью памяти вплоть до 4 раз.

Помимо стандартных подходов к квантованию, когда изменяются только статические параметры модели (веса), существует и широко применяется подход дополнительного квантования функций активации [4]. Квантование функций активации представляет собой более сложный процесс, а в случае, когда нет доступа к репрезентативной выборке данных – невозможный. В данном подходе происходит формирование диапазона выходных данных для наборов функций активации (рис. 1).

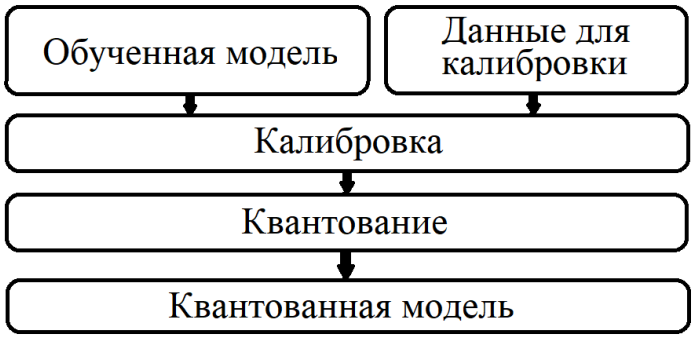

Рис. 1. Обобщённая схема калибровки при квантовании функций активации нейронной сети

Для формирования указанных диапазонов производится калибровка модели, при которой через нее пропускается небольшая часть (обычно 5-10 % от размера обучающей выборки) обучающих данных. На основании этих данных происходит квантование функций активации, что позволяет вычислять их значения в формате целых чисел, без использования плавающей точки. Данный способ может незначительно увеличить конечный размер модели, при этом позволяя получить весомый выигрыш в скорости обсчета на платформах, не поддерживающих аппаратные вычисления дробных значений.

Все вышеописанные варианты квантования модели сведены в таблице 1, представленной ниже.

Таблица 1

Показатели, полученные различными методами квантования после обучения

Метод квантования	**Размер конечной модели, байт**	**Точность модели, %**	**Усредненное время одного обсчета модели, мс**
Квантование весов float16	44716	97,62	109,63
Квантование весов uint8	24136	97,12	93,17
Квантование весов uint8 и функций активации int8	24672	97,08	95,56

По вышеприведенным данным можно сделать вывод относительно скорости обсчета модели. Действительно, квантование позволило в 4 раза снизить объем памяти, занимаемый моделью, однако, дополнительное квантование функций активации привело к увеличению скорости обсчета модели. Это может быть связано с наличием у используемого микроконтроллера модуля аппаратного обсчета чисел с плавающей точкой, что позволяет проводить расчет функций активации в формате с плавающей точкой быстрее, чем дополнительный обсчет блока квантора.

Прореживание модели

Подход прореживания модели позволяет уменьшить как размер конечной модели, так и вычислительную сложность за счет устранения части параметров (зануления весов) [5]. Прореживание происходит итеративно – для этого формируется так называемое расписание прореживания (pruning schedule). В расписании определяется количество шагов, по истечении которых происходит зануление части весов и дообучение модели. Модель итеративно прореживается до тех пор, пока не достигнет необходимой (заданной изначально) прореженности. В текущем случае для нашей модели было применено прореживание в течении двух эпох. Целевое количество зануляемых весов было задано равным 50 %.

Прореживание модели дало следующий результат: количество обнуленных параметров модели составило 49,98 %, точность модели незначительно снизилась. Можно заметить, что размер модели практически не изменился от изначального, однако был получен значительный выигрыш в производительности за счет прореживания параметров модели.

Поскольку после зануления части весов форматы представления параметров модели сохранились в виде float32, то возможно получить значительное сжатие при использовании квантования весов в uint8. В данном случае квантование дало дополнительное сокращение размера при практически идентичной точности.

Общие результаты прореживания модели представлены ниже, в таблице 2.

Таблица 2

Показатели модели нейронной сети, полученные с помощью прореживания параметров

Метод	**Размер конечной модели, байт**	**Точность модели, %**	**Усредненное время одного обсчета модели, мс**
Прореживание (зануление 50% параметров)	84 820	96,81	106,69
Прореживание с последующим квантованием в uint8	24 064	96,80	90,812

Прореживание модели дало прирост в производительности для исходной модели. При последующем квантовании удалось еще больше сократить время обсчета. Результат закономерен,

поскольку часть весов модели были прорежены. При этом получилось незначительное снижение точности по сравнению с методом квантования после обучения. Размер модели по сравнению с исходным также сократился в 4 раза, подобно тому, как это было при квантовании после обучения.

Обучение с учетом квантования

Обучение с учетом квантования выполняется путем дополнительного обучения модели с учетом возможности последующего квантования (рис. 2) [6, 7]. При этом, в конечной модели сохраняется исходный формат представления данных float32, однако при дальнейшем квантовании точность получится наиболее близкой к исходной, так как в модели на этапе обучения была учтена возможность квантованных весов и функций активации. Более того, при квантовании функций активации нет необходимости в дополнительной калибровке модели, так как она уже была откалибрована на этапе обучения.

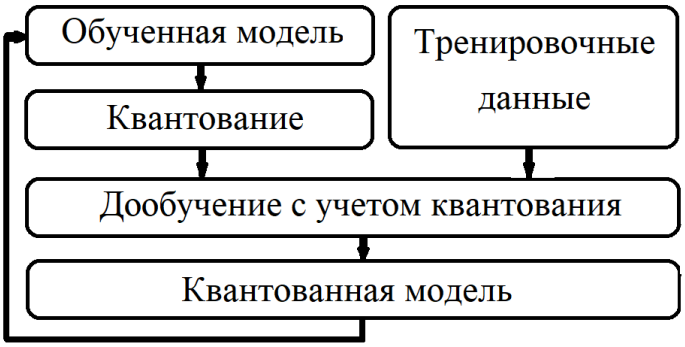

Рис. 2. Обобщенная схема дообучения модели нейронной сети с учетом квантования

Для дополнительного обучения с учетом квантования была использована 1/60 (1000) часть исходных тренировочных данных, которая была разделена на тренировочную и валидационную выборки в соотношении 9 к 1 соответственно.

Модель, полученная непосредственно после дополнительного обучения, с форматом представления данных float32 имела размер 84 896 байт при точности в 97,90 %.

После применения стандартного квантования весов и функций активации в uint8 размер модели сократился почти в 4 раза при лучшей (по сравнению с обычным квантованием после обучения с калибровкой) точности.

Общие результаты квантования с учетом обучения представлены ниже, в таблице 3.

Таблица 3

Показатели модели нейронной сети, полученные с помощью дополнительного обучения с учетом квантования

Метод	**Размер конечной модели, байт**	**Точность модели, %**	**Усредненное время одного обсчета модели, мс**
Формат представления float32	84 896	97,90	109,66
Формат представления весов и функций активации uint8	25 216	97,87	95,45

Обучение с учетом квантования показало сохранение точности. Время обсчета как исходной, так и квантованной модели получилось схожим с методом квантования после обучения. Данный факт обосновывается тем, что модели в обоих подходах имеют идентичную структуру, однако, в случае с обучением с учетом квантования удалось сохранить более высокую точность.

Заключение

Итогом предоставленной работы стало сравнение трех фундаментальных подходов сжатия модели нейронной сети. Лучшим по количеству занимаемой памяти стал метод прореживания модели с последующим квантованием (24 064 байт). Этот же метод позволил получить самую «быструю» модель (один обсчет модели занимает 90,812 мс). Однако, при этом происходит значительное (в сравнении с остальными методами) снижение точности – до 96,80%. Лучшим по сохранению точности методом оказалось обучение с учетом квантования с последующим квантованием. Данный метод значительно снизить размер модели (до 25 216 байт) при практически не изменившейся точности (97,87%).

Список использованных источников

1. MNIST Dataset // Kaggle: сайт. – 2019. – URL: https://www.kaggle.com/datasets/hojjatk/mnist-dataset
2. Tensorflow – Post-training Dynamic Range Quantization // Github: сайт. – 2022. – URL: https://github.com/tensorflow/tensorflow/blob/master/tensorflow/lite/g3doc/performance/post_training_quant.ipynb
3. Fang J., Shafiee A., Abdel-Aziz H., Thorsley D., Georgiadis G., Hasoun J. Near-lossless post-training quantization of deep neural networks via piecewise linear approximation – 2020.
4. Nagel M., Fournarakis M., Amjad R.A., Bondarenko Y., van Baalen M., Blankevoort T.A White Paper o Neural Network Quantization – 2021.
5. Sun M., Liu Z., Bair A., Zico Kolter J. A Simple and Effective Pruning Approach for Large Language Models // ICLR 2024 – 2024. – P. 778-800.
6. Cai Z., He X., Sun J., Vasconcelos N. Deep learning with low precision by half-wave gaussian quantization // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition – 2017. – P. 5918-5926.
7. Fang J., Shafiee A., Abdel-Aziz H., Thorsley D., Georgiadis G., Hasoun J. Post-training piecewise linear quantization for deep neural networks // European Conference on Computer Vision. – Springer, 2020. – P. 69-86.