МЕТОДЫ КВАНТОВАНИЯ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

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*

Введение

С развитием технологий нейронные сети постепенно начинают переходить с высоко производительных громоздких платформ на более компактные микроконтроллеры. Такой переход позволяет создавать малогабаритные устройства с низким энергопотреблением и меньшей стоимостью. Вместе с тем микроконтроллеры снабжены гораздо меньшим количеством оперативной и постоянной памяти и имеют меньшую частоту работы вычислительного ядра. Модели даже относительно простых нейронных сетей могут иметь достаточно большой размер по меркам современных микроконтроллеров. В добавок к этому, далеко не каждый микроконтроллер включает в себя такие модули, как блок вычислений с плавающей точкой. В связи с этими фактами необходим набор алгоритмов для оптимизации моделей нейронных сетей с целью их реализации на малопроизводительных устройствах.

Целью данной работы является обзор современных подходов для оптимизации и сжатия модели нейронной сети при использовании ее на микроконтроллерах.

Основная концепция квантования нейросетевых моделей

Основной задачей при обучении модели нейронной сети является оптимизация следующей функции:

$$F(W) = \frac{1}{N}\sum_{i=1}^{N} r(x_i, y_i, W), \quad (1)$$

где N – количество входных данных, (x_i, y_i) – эталонный набор вход-выходных данных, W – совокупность параметров нейросетевой модели, по которым происходит обучение (весовые значения), $r(x_i, y_i, W)$ – функция потерь.

Основная концепция квантования заключается в сжатии модели за счет снижения точности весов модели и дискретизации функций активации, а также в снижении затрат времени процессора при расчёте модели за счет применения естественных для аппаратного обеспечения представлений данных.

Сегодня для квантования в основном используется переход от значений формата *float32*, который широко используется при создании, обучении и просчёте моделей нейронных сетей, к целочисленному формату со снижением разрядности, обычно это *int8* или *int16* [1]. Такой переход позволяет сократить количество используемой памяти для хранения модели в среднем до 50 % [2], а также снизить нагрузку на вычислительное устройство.

Оператор квантования

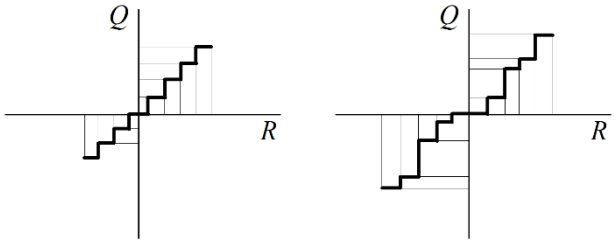
В первую очередь при выполнении квантования модели нейронной сети необходим оператор, с помощью которого будет возможно преобразовать значение из базового *float32* в значение с более низкой точностью. Такой оператор называют ***оператором квантования***.

Данный оператор необходим для преобразования из действительного значения *R* в квантованное значение *Q*. Преобразование может быть ***равномерным*** (2) или ***неравномерным*** (3) (см. рис. 1).

$$Q(R) = round(R/S) - Z, \quad (2)$$

$$Q(R) = q_i,\ R \in [r_i, r_{i+1}), \quad (3)$$

где R – действительное значение *float32*, представляющее собой выходное значение функции активации или значение веса модели, Z – смещение нуля в области квантованных значений, S – коэффициент преобразования действительного значения в квантованное, q_i – *i*–е квантованное значение, r_i – действительное значение, соответствующее *i*–му квантованному значению, *round(x)* – функция округления или отсечения дробной части.



а) *б*)

Рис. 1. Графическое представление способов квантования:
а) равномерное, б) неравномерное

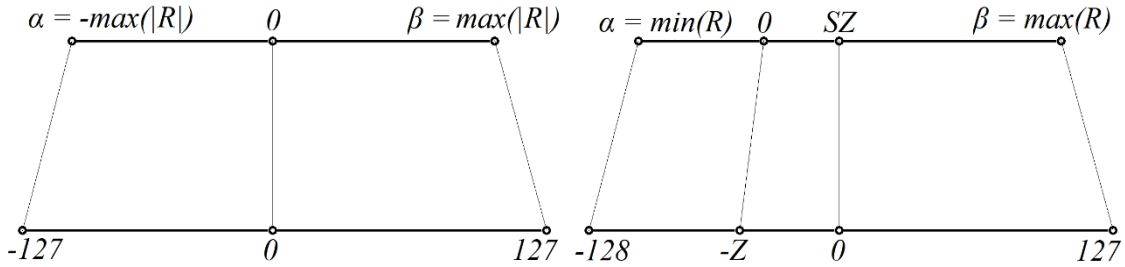
При неравномерном квантовании прибегают к использованию нелинейных зависимостей. Например, в работах [3, 4] был применен закон нормального распределения, а в [5] была использована логарифмическая зависимость при описании квантованных величин. Тем не менее, зачастую использование неравномерно квантования подход может оказаться не самым оптимальным из-за потенциальной сложности вычисления квантованных значений при нелинейных зависимостях (например, использующих логарифмическое или нормальное распределение). В связи с этим на практике зачастую прибегают к использованию равномерного квантования, при котором распределение квантованных значений подчиняется линейному закону.

При квантовании одним из важных вопросов является подбор параметра S в (2). Формально S описывается в следующем виде:

$$S = \frac{\beta - \alpha}{2^B - 1}, \tag{4}$$

где $[\beta; \alpha]$ – диапазон действительных значений, B – количество бит целочисленного типа квантованных значений.

При формировании параметра S существует два общепринятых подхода: *симметричное квантование* и *асимметричное квантование*. Графическое представление обоих подходов представлено на рис. 2.



а) *б*)

Рис. 2. Графическое представление способов квантования по подбору диапазонов действительных значений:
а) – симметричное, б) – асимметричное

Как видно из рис. 2, а, симметричное квантование подразумевает собой подбор равноудаленных от $Z = 0$ параметров α и β, что упрощает расчёт формулы (2), приводя её к следующему виду:

$$Q(R) = round(R/S). \tag{5}$$

Но в то же время при этом происходит потеря в точности из-за создания симметричных условий дополнительными действительными значениями. Это может быть особенно важно при квантовании ряда функций активации, например, часто используемой ReLU (Rectified Linear Unit), где существуют только неотрицательные действительные значения, что приводит к двукратному уменьшению диапазона квантованных значений.

Асимметричное квантование, в свою очередь, использует напрямую весь диапазон действительных значений, что усложняет вычисления, но при этом имеет более узкий диапазон действительных и

квантованных значений, что положительно сказывается на точности модели за счет отсутствия неиспользуемых квантованных значений [6].

Калибровка модели при квантовании

Основными параметрами, подлежащими квантованию, как было описано выше, являются веса модели и функции активации. Первые квантуются непосредственно на обученной модели, так как их значения являются фиксированными (исключая случаи, когда модель дополнительно обучается в процессе работы). Квантование функций активации является более сложной задачей, так как заранее неизвестны их выходные значения. Для этого производится калибровка модели с целью поиска лучшего диапазона значений $[\beta;\alpha]$ для параметра S из формулы (2).

В общем случае под калибровкой подразумевается аппроксимация функции активации на некотором диапазоне. Для этого через уже обученную модель пропускают выборку данных, по которым находят диапазон значений $[\beta;\alpha]$, после чего по найденному $[\beta;\alpha]$ происходит дискретизация функции активации.

Существует два основных подхода проведения калибровки: *динамическая калибровка* и *статическая калибровка*.

Статическая калибровка подразумевает наличие некоторой выборки данных [4, 7]. С помощью данной выборки происходит поиск $[\beta;\alpha]$ и дискретизация функции до того, как модель будет встроена в устройство и запущена. Это позволяет получить дискретную функцию еще до начала работы модели на целевом устройстве и не тратить процессорное время на дополнительные преобразования при расчёте модели.

При динамической калибровке, напротив, квантование функций активации происходит в реальном времени непосредственно по тем значениям, которые поступают с их выходов [6]. Данный подход позволяет добиться лучшей точности в сравнении со статической калибровкой за счет использования только тех данных, которые модель получает при работе. Однако такой подход значительно увеличивает время расчёта модели из-за необходимости дополнительно проводить дискретизацию для функций активации.

Несмотря на более высокую точность динамической калибровки, сегодня в подавляющем большинстве случае прибегают к статической калибровке. Это связано с дополнительной нагрузкой при расчете диапазона выходных данных при динамической калибровке, что может снизить частоту обработки модели.

Калибровка может проводиться как по целому слою модели [9], так и по каждому отдельному каналу (по каждой функции активации) отдельно [4, 8]. В этом смысле калибровку делят на *послойную калибровку* и на *поканальную калибровку*.

В случае послойной калибровки диапазон $[\beta;\alpha]$ для всех каналов в пределах слоя един. При поканальной калибровке данный диапазон уникален для каждого канала. Выбор послойной или поканальной калибровки зависит от целевой платформы. С одной стороны поканальная калибровка потребует дополнительной памяти для хранения диапазонов $[\beta;\alpha]$ и дополнительного процессорного времени. С другой стороны, послойная калибровка, как обозначено в [7], может привести к потери точности модели в тех случаях, когда имеется заметное различие в разбросе действительных значений различных функций активации в пределах слоя модели. В работе [7] рассматриваются различные подходы при выборе одного из данных методов. Помимо двух упомянутых базовых подходов, в [7] также предлагается гибридный вариант *квантования по группе каналов*, где общий диапазон действительных значений применяется к части каналов одного слоя модели.

Алгоритмы формирования квантованной модели с минимальной потерей точности

При квантовании модели нейронной сети зачастую происходит значительное снижение точности как следствие снижения разрешения параметров. Существует два основных метода формирования квантованной модели нейронной сети с минимальной потерей точности: *обучение с учетом квантования* (Quantization–Aware Training или QAT) и *квантование после обучения* (Post–Training Quantization или PTQ). Процесс квантования для обоих подходов представлен на рис. 3.

а) *б)*

Рис. 3. Алгоритмы методов формирования квантованных моделей:
а) обучение с учётом квантования, б) квантование после обучения

Обучение с учетом квантования использует готовую обученную модель нейронной сети. Процесс повторного обучения квантованной модели является итерационным. На каждой итерации происходит квантование весов и просчитывается прямое распространение сигнала по модели нейронной сети. После этого просчитывается обратное распространение ошибки, в ходе которого используются исходные значения с плавающей точкой. Использование исходных значений необходимо для возможности аккумулирования значений весов [10, 11]. Как только произошло формирование новых исходных значений весов происходит повторное квантование и цикл начинается снова. Основной минус такого подхода заключается в необходимости дополнительного обучения на выборке того же размера, что и исходная, то есть обучение будет производиться дважды. Тем не менее, указанным методом можно получить исходную или незначительно сниженную точность для квантованной модели [2]. Примеры использования данного подхода представлены в работах [10, 11].

Менее затратным в плане времени является второй способ – *квантование после обучения*. Его применение демонстрируется в [12, 13]. Данный подход требует относительно небольшую выборку для калибровки диапазонов $[\beta;\alpha]$ и расчета параметра оператора S квантования. После калибровки происходит квантование функций активации и весов модели с помощью оператора квантования. Хотя данный метод квантования модели очень прост и не требует больших вычислительных ресурсов, результирующая точность модели может снизиться в большей степени, чем при обучении с учётом квантования [2].

Сегодня оба рассмотренных алгоритма обработки модели нейронной сети поддерживаются в известных программных пакетах. Широко известный TensorFlow выполняет квантование из *float32* в *int8* и *int16* посредством как Post-Training Quantization (PTQ), так и Quantization-Aware Training (QAT) [2]. Такой же набор алгоритмов имеется и в программе PyTorch [8].

Заключение

В данной работе были представлены основные подходы, используемые при сжатии моделей нейронных сетей и их адаптации к малопроизводительному аппаратному обеспечению. Были описаны базовые принципы квантования и алгоритмы, позволяющие проводить квантование с минимальными потерями в точности.

Представленная работа является кратким обзором современных методов квантования. В дальнейшем планируется практическая апробация описанных в работе методов с целью интеграции нейросетевых моделей в микроконтроллеры различных семейств с оценкой производительности.

Список использованных источников

1. Tripathy J.R., Tripathy H.K., Nayak S.S. Artificial Neural Network Implementation in Microchip PIC 18F45J10 8-Bit Microcontroller // International Journal of Engineering and Advanced Technology. – 2014. – Vol. 19, № 4 – P. 131–135.

2. TensorFlow Models // Model optimization: сайт. – URL: https://www.tensorflow.org/lite/performance/model_optimization.

3. 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.

4. Benoit J., Skirmantas K., Bo C., Menglong Z., Tang M., Howard A., Hartwing A., Kalenichenko D. Quantization and training of neural networks for efficient integer-arithmetic-only inference // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2018. – P. 2704-2713.

5. Zhou A., Yao A., Guo Y., Xu L., Chen Y. Incremental network quantization: Towards lossless cnns with low-precision weights // International Conference on Learning Representations. – 2017.

6. Nagel M., Fournarakis M., Amjad R. A., Bondarenko Y., van Baalen M., Blankevoort T. A White Paper on Naural Network Quantization. – 2021.

7. Yao Z., Dong Z., Zheng Z., Gholami A., Yu J., Tan E., Wang L., Huang Q., Wang Y., Mahoney M. Hawqv3: Dyadic naural network quantization // Proceedings of the 38th International Conference on Machine Learning. – 2021. – P. 11875-11886.

8. Huang Q., Wang D., Dong Z., Gao Y., Cai Y., Li T., Wu B., Keutzer K., Wawrzynek J. Codenet: Efficient deployment of input-adaptive object detection on embedded fpgas // The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. – 2021. – P. 206-216.

9. Krishnamoorthi R. Quantizing deep convolutional networks for efficient inference: A whitepaper. – 2018.

10. Gysel P., Motamedi M., Ghiasi S. Hardware-oriented approximation of convolutional neural networks // Workshop contribution at ICLR 2016 – 2016.

11. Gysel P., Pimentel J., Motamedi M., Ghiasi S. Ristello: A framework for empirical study of resource-efficient inference in convolutional neural networks // IEEE transactions on neural networks and learning systems. – 2018. – P. 5784–5789.

12. 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.

13. 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.