ДОКЛАДЫ РОССИЙСКОЙ АКАДЕМИИ НАУК. МАТЕМАТИКА, ИНФОРМАТИКА, ПРОЦЕССЫ УПРАВЛЕНИЯ, 2023, том 514, № 2, с. 126–137

УДК 517.54

ТЕХНИКИ СЖАТИЯ АКТИВАЦИЙ СЛОЕВ И ГРАДИЕНТОВ ДЛЯ РАСПРЕДЕЛЕННОГО ОБУЧЕНИЯ МОДЕЛЕЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

© 2023 г. М. И. Рудаков[1,2,*], А. Н. Безносиков[1,2,], Я. А. Холодов[1,***], А. В. Гасников[1,2,****]**

Представлено академиком РАН А.Л. Семеновым
Поступило 01.09.2023 г.
После доработки 15.09.2023 г.
Принято к публикации 18.10.2023 г.

Современные большие нейронные сети требуют для обучения огромных вычислительных ресурсов. В такой постановке параллелизация процесса обучения, когда последовательные слои модели разбиваются между устройствами, является популярным подходом для обучения больших моделей. Для уменьшения времени обмена данными между устройствами, часто являющимся узким местом в таких системах, применяется сжатие информации. В данной работе исследуется влияние одновременного сжатия активаций и градиентов в режиме параллелизации по модели на сходимость процесса обучения. Мы анализируем такие подходы, как квантизация и "жадное" TopK сжатие, а также экспериментируем с методами компенсации ошибки. Мы исследуем TopK сжатие с использованием подхода AQ-SGD с побатчевой компенсацией ошибки сжатия. Сравнения проводятся на задачах обучения ResNet18 и дообучения GPT-2. Полученные нами результаты показывают, что градиенты более чувствительны к степени сжатия, чем активации слоев модели. По нашим наблюдениям, $K = 10\%$ – это максимальный уровень сжатия TopK, который не оказывает сильного влияния на сходимость модели. Эксперименты также показывают, что модели, обученные с использованием сжатия TopK, хорошо работают только в том случае, если сжатие применяется и во время валидации. Мы обнаружили, что техники компенсации ошибки одновременно для активаций и градиентов не улучшают сходимость по сравнению с обычным сжатием. Наконец, применение подхода AQ-SGD с TopK сжатием сильнее, чем при $K = 30\%$, значительно ухудшает качество модели.

Ключевые слова: распределенное обучение, параллелизм модели, сжатие активаций, сжатие градиентов, техники компенсации ошибки

DOI: 10.31857/S2686954323601562, **EDN:** CMLGLL

1. ВВЕДЕНИЕ

Глубокие нейронные сети стали одним из важнейших направлений исследований в области компьютерных наук. Ежегодно публикуются сотни статей по машинному обучению, включая компьютерное зрение, обработку естественного языка, обучение с подкреплением и генеративные модели. Одной из причин такого успеха является колоссальный рост размеров моделей и обучающих наборов данных. Последние примеры включают модели GPT-4 [1] и BLOOM [2], содержащие более ста миллиардов обучаемых параметров сети. Для обучения таких архитектур также требуются терабайты данных: например, текстовый корпус ROOTS [3], используемый для обучения BLOOM, имеет объем 1.6 ТБ. Как следствие, современные модели требуют для обучения большого объема памяти CPU и GPU. Вычислительные ресурсы такого масштаба доступны лишь не-

[1]*Университет Иннополис, Иннополис, Республика Татарстан, Россия*
[2]*Московский физико-технический институт, Москва, Россия*
**E-mail: m.rudakov@innopolis.university*
***E-mail: beznosikov.an@phystech.edu*
****E-mail: ya.kholodov@innopolis.ru*
*****E-mail: gasnikov.av@mipt.ru*

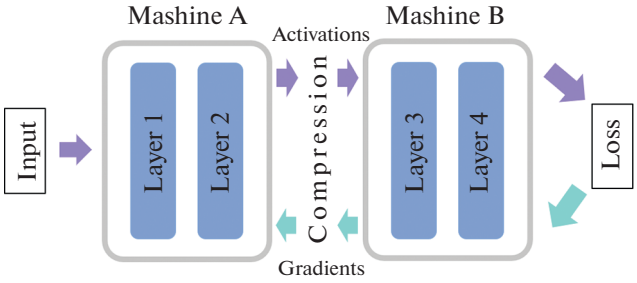

Рис. 1. Пример параллелизации обучения по модели с использованием сжатия. Степень параллелизации – два, с одним блоком сжатия. Активации сжимаются при прямом проходе, а градиенты – при обратном.

скольким крупным институтам и компаниям, которые могут позволить себе кластеры серверов с большой пропускной способностью сети. При этом менее значительные игроки также могут пробовать подступаться к использованию современных архитектур, объединяя свои ресурсы друг с другом по сети Интернет.

В любом случае обучение больших моделей невозможно без использования большого числа вычислительных устройств, а значит и без использования подхода распределенного обучения [4]. В такой постановке двумя основными парадигмами являются параллелизация по данным (data-parallel) и по модели (model-parallel), которые позволяют обучать большие нейронные сети, используя несколько устройств параллельно. При параллелизации по модели, или конвейерном обучении (model-parallel, MP/pipeline-parallel, PP), слои нейронной сети последовательно разделяются между устройствами. Этот подход, позволяет обучать модели, которые не помещаются на одном компьютере, что расширяет границы применения нейронных сетей. Несколько современных фреймворков поддерживают параллелизацию моделей, в том числе Megatron [5], Deepspeed [6] и Petals [7].

Однако медленная скорость соединения становится проблемой при параллелизации по модели [8]. Во время прямого прохода по вычислительному графу устройства отправляют векторы активаций слоев соседним в конвейере машинам. При обратном проходе передаются векторы соответствующих градиентов для обновления обучаемых параметров. Время обмена данными может стать узким местом, создавая простой устройств в ожидании данных, особенно если параллелизация модели происходит на машинах, расположенных по всему миру, через медленное сетевое соединение.

Выделим основные работы, связанные с компрессией пересылаемой информации при параллелизации по модели. Различные схемы квантизации применяются к активациям или градиентам [9, 10], или к обоим типам векторов [11]. Также в литературе анализируется работа методов сжатия для градиентов в случае распараллеливания по данным [12, 13], и лишь несколько работ исследуют сжатие активаций при параллелизации по модели [14, 15]. В ряде работ также используются методы компенсации ошибки для активаций [16] и градиентов [17]. Несмотря на достаточное количество работ, посвященных сжатию при параллелизации по модели, результаты, как правило, являются в основном эмпирическими, полученными отдельно для активаций или градиентов.

В данной работе мы представляем эксперименты по сжатию с использованием параллелизации модели. Основное внимание мы уделяем исследованию одновременного сжатия активаций и градиентов для различных методов компрессии. Кроме того, мы экспериментируем с техниками компенсации ошибки, известными из распределенного обучения с параллелизацией по данным, чтобы проверить их применимость вместе с “жадным” сжатием TopK.

Вклад нашей работы заключается в следующем:

1. Мы проводим эксперименты по сжатию активаций и градиентов при параллелизации по модели и эмпирически оцениваем сходимость для квантизации и TopK сжатий;

2. Мы экспериментируем с различными техниками компенсации ошибки при сжатии активаций и градиентов, включая EF, EF21 и AQ-SGD;

3. Мы оцениваем эффективность метода компенсации ошибки AQ-SGD для активаций с TopK-сжатием, и приходим к выводу, что он не улучшает сходимость модели по сравнению с обычным TopK сжатием.

2. МЕТОДОЛОГИЯ

В этом разделе мы приводим общую постановку экспериментов по сжатию передаваемой информации при обучении в режиме параллелизации по модели. Мы также описываем рассматриваемые операторы и схемы сжатия.

Постановка эксперимента: параллелизация обучения со сжатием активаций и градиентов

В наших экспериментах мы обучаем глубокие нейронные сети, используя подход параллелизации модели со сжатием между этапами конвейера. На рис. 1 представлен этот подход в случае, где вся модель разбивается на два устройства с одним этапом передачи данных между ними (степень параллелизации равна 2). Смежные звенья конвейера обмениваются активациями слоев во вре-

мя прямого прохода и градиентами этих активаций во время обратного прохода.

Мы используем операторы сжатия для передачи меньшего количества данных, снижая расходы на коммуникации. В наших экспериментах мы сжимаем как активации, так и градиенты, чтобы оценить, в каких пределах можно использовать сжатие без ущерба для сходимости и качества обучения модели.

Поскольку наша работа направлена только на оценку сходимости и качества, мы не проводим реального обучения модели на нескольких машинах. Вместо этого мы интегрируем сжатие непосредственно в модель, сжимая активации и градиенты там, где происходит обмен данными в реальной распределенной системе. Такой подход эквивалентен реальному распределенному обучению модели с точки зрения анализа сходимости.

Операторы сжатия

Для снижения коммуникационных затрат мы применяем различные операторы сжатия к активациям и градиентам. Мы анализируем сходимость операторов квантизации, TopK, операторов с компенсацией ошибки, а также подхода AQ-SGD и комбинаций перечисленных техник.

Квантизация

Квантизация – это метод сжатия, который заключается в преобразовании чисел с плавающей точкой в дискретный набор целочисленных значений. В наших экспериментах мы используем равномерную k-битную квантизацию с масштабированием. Входной вектор с действительными числами отображается на интервал [0; 1] с масштабированием min-max, а затем квантизуется на k уровней. Декомпрессия преобразует эти значения обратно к исходному масштабу. В наших экспериментах мы оцениваем квантизацию до 2, 4, 6, 8 бит для активаций и градиентов.

TopK Сжатие

TopK сжатие – это метод спарсификации с доказанной скоростью сходимости при параллелизации обучения по данным (data parallelism) [18]. Оператор TopK выбирает из входного вектора наибольшие K% значений (по абсолютной величине). При такой технике сжатия для передачи выбираются только наиболее важные данные, которыми, как предполагается, являются наибольшие по абсолютной величине активации и градиенты.

Наша реализация сжатия TopK имеет несколько особенностей. Во-первых, мы перестраиваем входной вектор для операции TopK таким образом, чтобы значения TopK выбирались для каждой порции данных (батчей) отдельно. Такая процедура позволяет избежать обнуления всех значений в некоторых батчах, что привело бы к некорректному обучению модели. Во-вторых, наибольшие K% значений выбираются по абсолютной величине, так как важны отрицательные и положительные изменения активаций и градиентов. Наконец, поскольку мы экспериментируем с параллелизацией по модели с точки зрения сходимости, мы не выполняем точную компрессию и декомпрессию значений при TopK сжатии. Вместо этого мы обнуляем значения, которые не входят в TopK.

Мы оцениваем уровни сжатия TopK (Top50%, Top30%, Top20%, Top10%, Top5%) для каждой задачи обучения, сжимая как активации, так и градиенты. Для задачи дообучения мы также пробуем

оператор, который переиспользует выбранные TopK индексы активаций для сжатия градиентов.

Методы компенсации ошибки

Компенсация ошибки (error feedback, EF) – это техника сжатия данных, широко используемая в обучении с параллелизацией по данным (data-parallel) [19–21]. Компенсация ошибки корректирует изменения, вносимые сжатием, путем передачи ошибки в следующих раундах коммуникации. Такой подход гарантирует, что в конечном итоге вся информация вносит свой вклад в процесс обучения.

Оригинальная версия техники компенсации ошибки, предложенная Seide и др. [19], сохраняет ошибку сжатия и добавляет ее во время следующего раунда обмена данными. При заданном векторе X и операции сжатия $\mathbb{C}$ ошибка вычисляется как $e = X - \mathbb{C}(X)$. На следующей итерации сжатое сообщение представляет собой новое передаваемое значение, добавленное к предыдущей ошибке, т.е. $\mathbb{C}(X + e)$.

Усовершенствованный вариант компенсации ошибки – EF21 – был представлен Richtarik и др. [21]. Он заключается в сжатии разности текущей посылки и предыдущего состояния буфера отправки. Поскольку по мере обучения вектор градиента должен стремиться к нулю, а вектор активаций – стабилизироваться у одного значения, то и сжимаемая стремится к нулю, а значит, сжатие таких значений приводит к уменьшению ошибки. При заданном векторе активаций X_{i+1} и ранее переданном значении g_i передаваемое сообщение имеет вид $\mathbb{C}(X_{i+1} - g_i)$, а значение буфера отправки обновляется как $g_{i+1} = g_i + \mathbb{C}(X_{i+1} - g_i)$.

Основываясь на идеях техник компенсации ошибки, мы также рассматриваем новый оператор сжатия EF-mixed для сжатия TopK. Идея заключается в сжатии наибольших по модулю $K/2\%$

значений из входного вектора X и $K/2\%$ текущего буфера ошибок. После этого входной вектор и буфер складываются и пересылаются, в результате чего получается не более $K\%$ ненулевых значений. Такой подход стремится получить наиболее важные значения из входных данных и буфера ошибки для активаций и градиентов.

Мы применяем подходы компенсации ошибки сжатия для обучения модели с параллелизацией по модели и использованием сжатия коммуникаций. Для сжатия активаций и градиентов используются алгоритмы EF, EF21 и EF-mixed с оператором сжатия TopK. Буфер ошибок является глобальным, т.е. накопленная ошибка добавляется к следующему батчу данных. Эксперименты с методами компенсации ошибки направлены на исследование того, приводит ли такая техника к лучшей сходимости при параллелизации по модели по сравнению с обычным сжатием активаций и градиентов.

AQ-SGD сжатие

AQ-SGD [16] – это техника, разработанная для сжатия активаций при параллелизации по модели, использующая вариацию компенсации ошибки сжатия. Аналогично EF21, AQ-SGD сжимает и передает изменение значений активации в процессе обучения. Однако буфер ошибок индивидуален для каждого батча, что приводит к большим затратам по памяти. В исходной работе в качестве сжатия используется квантизация для сжатия активаций и градиентов, а техника AQ-SGD применяется только для активаций.

В наших экспериментах мы используем подход AQ-SGD в сочетании со сжатием TopK. Мы оцениваем, остается ли AQ-SGD эффективным при смещенном сжатии TopK, применяемом для активаций слоев и градиентов. Мы тестируем сжатие Top50%, Top30%, Top20% и Top10% с компенсацией ошибки AQ-SGD для активаций и обычное сжатие TopK для градиентов.

Код экспериментов доступен на GitHub[1].

3. РЕЗУЛЬТАТЫ

В этом разделе мы представляем результаты экспериментов с различными методами сжатия активаций и градиентов при обучении глубоких нейронных сетей в режиме параллелизации по модели. Представлены эксперименты с моделью ResNet-18 и набором данных CIFAR-10. Эти эксперименты включают квантизацию, сжатие TopK, методы компенсации ошибки и эксперименты на основе подхода AQ-SGD. Также обсуждаются эксперименты по дообучению GPT-2 на наборе данных Wikitext со сжатием TopK.

Эксперименты с ResNet18 и CIFAR-10

В первой серии экспериментов проводилось обучение сверточной нейронной сети ResNet18 [22] на наборе изображений CIFAR-10 [23]. Качество классификации изображений измеряется точностью обучения (accuracy); также измеряется значение функции потерь при обучении и валидации.

Эксперименты проводились на одном графическом процессоре Tesla P100-PCIE-16GB со степенью параллелизации модели, равной 4, используя по 3 независимых оператора сжатия для активаций и градиентов. Количество эпох обучения было установлено на 100 с размером батча 100. Использовался оптимизатор SGD с моментумом 0.9 и весовым затуханием 5e-4. Скорость обучения регулируется планировщиком косинусного отжига с начальным значением 0.01 и $T_{\max} = 200$.

Наша реализация основана на коде репозитория pytorch-cifar[2], в котором сравниваются различные модели классификации изображений на наборе данных CIFAR-10.

Для замера базового качества было проведено обучение без сжатия. За пять запусков была достигнута средняя точность на тестовой выборке в 93.0%.

Квантизация

На рис. 2 показаны значения функции потерь при обучении, точность в несжатом и сжатом случае на тестовом наборе для экспериментов с квантизацией. В табл. 1 представлена наилучшая точность на тестовой выборке, достигнутая для каждого эксперимента по сжатию.

Во-первых, мы наблюдаем, что градиенты более чувствительны к квантизации, чем активации. Удовлетворительная тестовая точность достигается только при квантизации градиентов не менее чем до 6 бит, в то время как активации могут быть квантизованы до 2 или 4 бит. Такой результат согласуется с результатами, описанными Wang и др. [16], Bian и др. [14], где сообщается о слабой эффективности сжатия градиентов до 2 и 4 бит. Различие между сжатием активаций и градиентов может заключаться в их влиянии на обучение. Сжатие градиентов может серьезно нарушить оптимизацию и сходимость, в то время как сжатие активаций имеет менее серьезный эффект, если модель может адаптироваться к распределению сжатых активаций, и этих активаций достаточно для обучения.

[1] https://github.com/Glemhel/ActivationsGradientsCompressionForMP

[2] https://github.com/kuangliu/pytorch-cifar

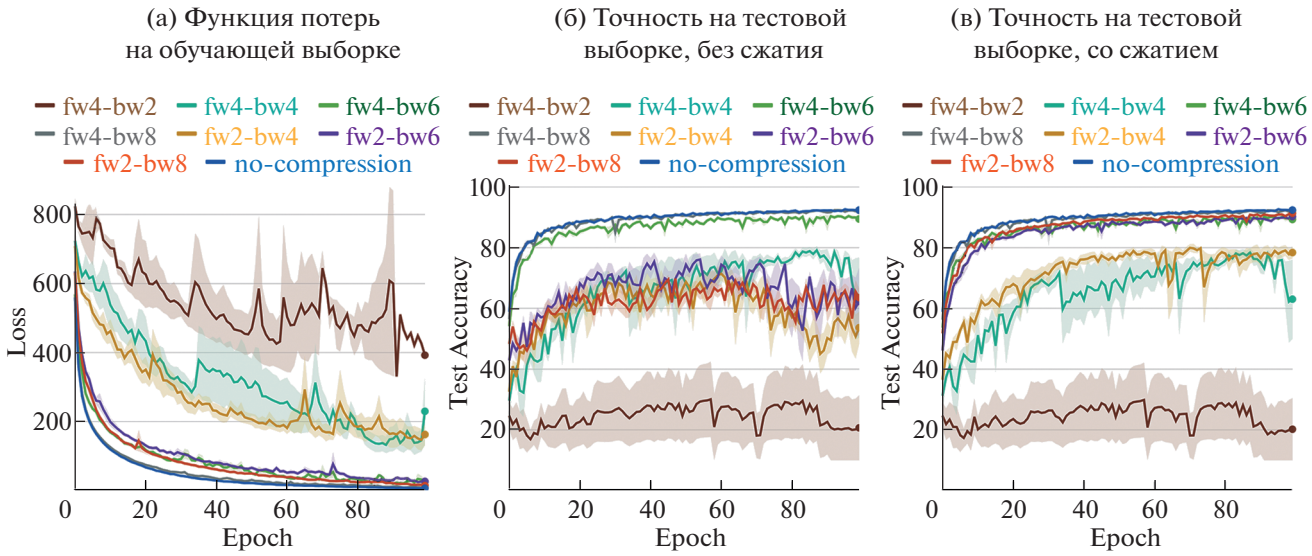

Рис. 2. Графики сходимости ResNet18 с квантизацией на CIFAR-10. fw[A]-bw[B] означает квантизацию активаций до A бит, градиентов до B бит. Каждая линия – среднее ± стандартное отклонение точности модели за 5 запусков модели, каждый запуск на 100 эпох обучения модели. Степень параллелизации модели равна 4, используется 3 операции сжатия.

Во-вторых, мы отмечаем заметную разницу в точности на тестовой выборке со сжатием и без сжатия. Точность модели со сжатием на 7–15 процентных пунктов выше, чем при применении без сжатия для квантизации активаций до 2 бит. Можно сделать вывод, что сжатие становится частью модели, поскольку отказ от сжатия снижает производительность модели. Когда в модель передаются несжатые активации, они непредсказуемо изменяют выход модели.

TopK сжатие

Результаты экспериментов со сжатием TopK представлены в табл. 2, а графики обучения – на рис. 3. Как и в случае с квантизацией, мы наблюдаем, что для сжатия TopK точность предсказания с компрессией выше, чем без компрессии. В то же время тестовая точность со сжатием остается удовлетворительной вплоть до Top10% сжатия; точность без сжатия оказывается ниже 90% уже при Top20%. В случае со сжатием TopK активации, которые обычно устанавливаются в 0 при сжатии, существенно меняют поведение модели при отсутствии сжатия. Поэтому для модели, обученной с помощью сжатия, применение сжатия является обязательным для достижения хороших результатов. Насколько нам известно, предыдущие работы не обнаруживали такой эффект сжатия в обучении с параллелизацией по модели. Более того, наши результаты показывают, что сжатие Top10% независимо для активаций и градиентов является хорошей техникой для применения в сверточных нейронных сетях типа ResNet.

Сжатие с методами компенсации ошибки

В табл. 3 и на рис. 4 приведены результаты обучения для экспериментов с методами компенсации ошибки. Результаты показывают, что использование компенсации ошибки в дополнение к сжатию TopK не улучшает точность модели. Значительное различие активаций и градиентов между батчами может быть причиной такого результата. Так, добавление ошибок одного обучающего примера в данные для примера из друго-

Таблица 1. Результаты обучения ResNet18 с квантизацией на CIFAR-10

Степень сжатия	Точность на тестовой выборке (%), без сжатия	Точность на тестовой выборке (%), со сжатием
Без сжатия	**93.00**	**93.00**
fw4-bw8	**93.10**	**92.95**
fw4-bw6	**91.93**	**91.76**
fw4-bw4	83.39	82.66
fw4-bw2	65.11	64.27
fw2-bw8	77.09	**92.05**
fw2-bw6	84.87	**91.59**
fw2-bw4	81.32	83.92

Примечание. fw[A]-bw[B] означает квантизацию активаций до A бит, градиентов – до B бит. В каждой ячейке – наилучший результат за 5 запусков, каждый запуск на 100 эпох обучения модели. Степень параллелизации модели равна 4, используется 3 оператора сжатия.

Таблица 2. Результаты обучения ResNet18 с TopK сжатием на CIFAR-10

Степень сжатия	Точность на тестовой выборке (%), без сжатия	Точность на тестовой выборке (%), со сжатием
Без сжатия	**93.00**	**93.00**
Top 50%	**93.16**	**93.38**
Top 30%	**91.26**	**93.06**
Top 20%	86.74	**92.73**
Top 10%	75.89	**91.87**
Top 5%	55.09	90.01
Top 2%	49.95	85.52

Примечание. В каждой ячейке – наилучший результат за 5 запусков, каждый запуск на 100 эпох обучения модели. Степень параллелизации модели равна 4, используется 3 оператора сжатия. Активации и градиенты сжимаются независимо друг от друга.

Таблица 3. Результаты обучения ResNet18 с TopK сжатием и компенсацией ошибки на CIFAR-10

Степень сжатия	Точность на тестовой выборке (%), без сжатия	Точность на тестовой выборке (%), со сжатием
Без сжатия	**93.00**	**93.00**
EF + Top 10%, warmup 20	**91.02**	89.85
EFmixed + Top 10%, warmup 20	61.37	90.97
EF21 + Top 5%	89.37	89.29
EF21 + Top 10%	90.83	**91.19**
EF21 + Top 10%, warmup 20	90.71	**91.77**

Примечание. Каждый запуск обучения модели на 100 эпох. В запусках Warmup 20 (base 20) используется предобученная модель за 20 эпох без сжатия. Степень параллелизации модели равна 4, используется 3 оператора сжатия. Активации и градиенты сжимаются независимо друг от друга, с глобальным буфером компенсации ошибки.

го целевого класса может существенно повлиять на буфер ошибки и качество предсказаний модели. Возможно, именно поэтому глобальный буфер компенсации ошибки не улучшает качество модели при параллелизации по модели.

Неожиданным результатом стало то, что техника компенсации ошибки выравнивает результаты точности предсказаний на тестовой выборке для несжатого и сжатого случая. Точность без сжатия падает всего на 1–2 процентных пункта при использовании сжатого варианта EF или EF21, в то время как при обычном сжатии TopK падение точности тестов составляет не менее 7–15 процентных пунктов. Мы предполагаем, что буфер EF рассматривается моделью как шум, и модель обучается игнорировать этот шум для активаций, что приводит к сопоставимым результатам на валидации без сжатия и со сжатием.

AQ-SGD со сжатием TopK

Наконец, в табл. 4 представлены результаты экспериментов на основе подхода AQ-SGD [16], в котором вместо квантизации используется сжатие TopK.

Графики обучения на рис. 5 свидетельствуют о том, что сходимость метода AQ-SGD со сжатием TopK не улучшается по сравнению с обычным сжатием TopK. Рост точности на тестовом множестве не соответствует исходному уровню без сжатия при сжатии Top10%. Мы предполагаем, что сходимость при использовании сжатия TopK в подходе AQ-SGD происходит медленно из-за смещенности оператора сжатия TopK. Кроме того, как было отмечено для методов с компенсацией ошибки, техника с побатчевой компенсацией ошибки AQ-SGD также достигает сравнимого качества предсказания на тестовой выборке вне зависимости от использования сжатия во время валидации.

Эксперименты с GPT-2 и Wikitext

Мы провели дообучение GPT-2-small [24] на наборе текстовых данных Wikitext [25], версия wikitext-2-raw-v1. Качество модели оценивалось по значению функции потерь и метрике perplexity (PPL). Мы дообучали модель в течение 4 эпох с размером батча 8. Использовался код от Hugging Face[3] с интегрированным в него сжатием. Эксперименты проводились на графическом процессоре Tesla P100-PCIE-16GB. Степень параллелизации модели была установлена на 4, с использованием 3-х операторов сжатия для активаций и для градиентов.

На рис. 6 показаны графики функции потерь при дообучении модели. В табл. 5 представлены результаты дообучения с применением сжатия TopK.

Во-первых, сжатие TopK начинает увеличивать значения функции потерь на тестовом множестве уже с Top20% сжатия. По сравнению с нашими экспериментами с ResNet18, где сжатие Top10% приводило к удовлетворительным результатам, текущая задача не выдерживает такого сжатия. Такое поведение может объясняться двумя факторами: архитектурными различиями между трансформерами и и сверточными сетями, а также использованием дообучения вместо обучения

[3] https://github.com/huggingface/transformers/blob/main/examples/pytorch/language-modeling/run_clm.py

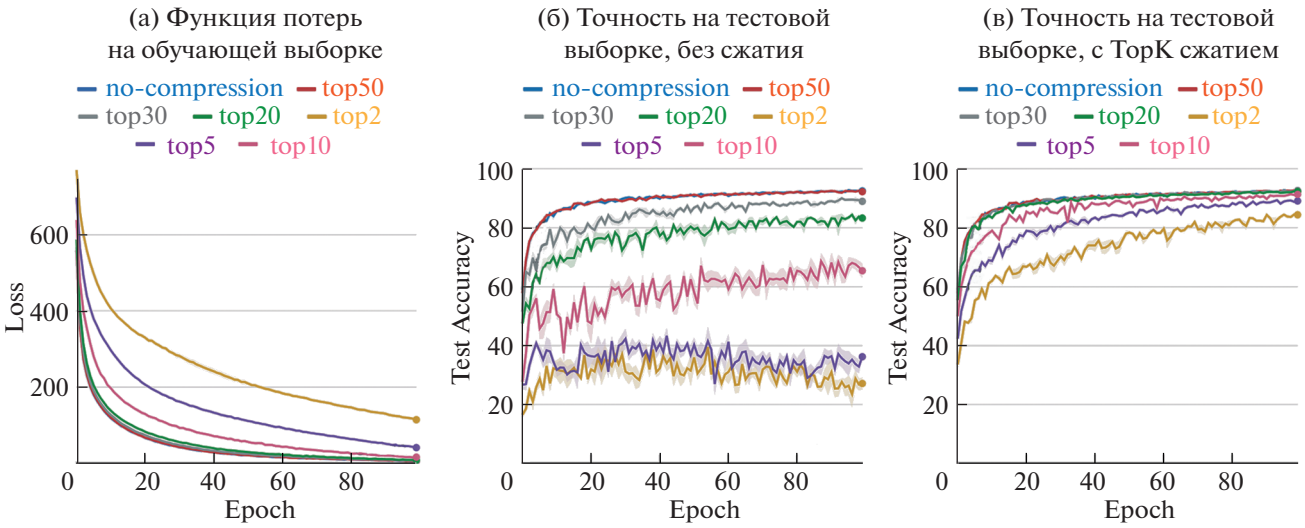

Рис. 3. Графики сходимости ResNet18 с TopK сжатием на CIFAR-10. Каждая линия – среднее ± стандартное отклонение точности модели за 5 запусков модели, каждый запуск на 100 эпох обучения модели. Степень параллелизации модели равна 4, используется 3 оператора сжатия. Активации и градиенты сжимаются независимо друг от друга.

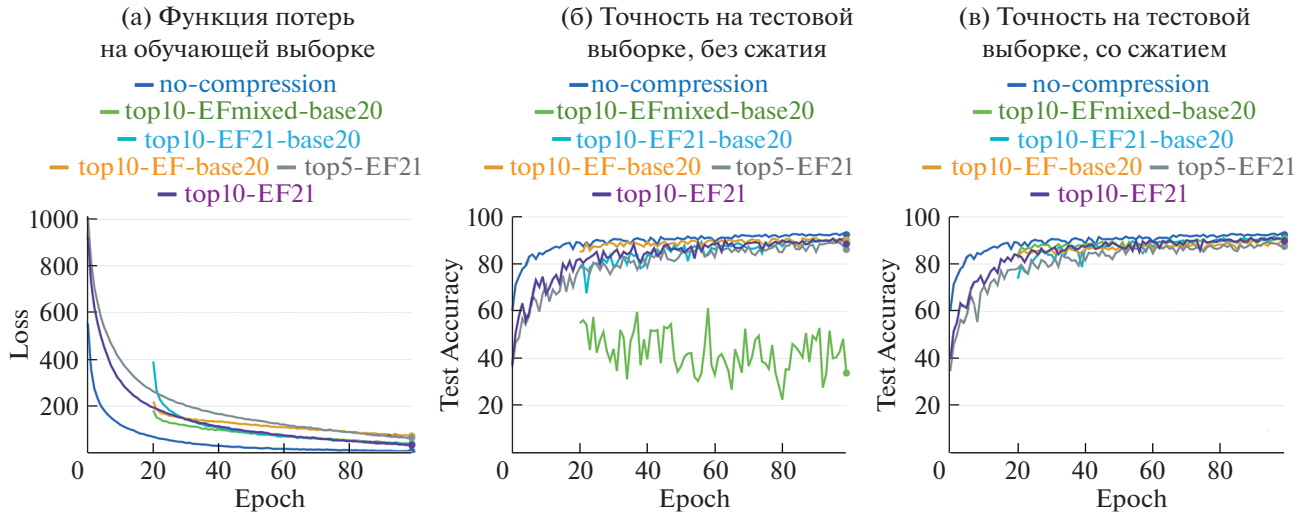

Рис. 4. Графики сходимости ResNet18 с TopK сжатием и компенсацией ошибки на CIFAR-10. Каждый запуск обучения модели на 100 эпох. В запусках Warmup 20 (base 20) используется предобученная модель за 20 эпох без сжатия. Степень параллелизации модели равна 4, используется 3 оператора сжатия. Активации и градиенты сжимаются независимо друг от друга, с глобальным буфером компенсации ошибки.

с нуля, что может препятствовать эффективному обучению распределений сжатия TopK.

Во-вторых, в сценарии дообучения сжатие активаций и градиентов независимо друг от друга приводит к очень большим потерям в оценке и даже к расхождению модели. Мы предполагаем, что такое поведение определяется тем, что мы используем предварительно обученную модель GPT-2. Для предварительно обученной модели обнуление большого количества активаций влияет на градиенты сильнее, чем обучающая выборка, на которой тренируется модель. На наш взгляд, для предотвращения такого поведения необходимо переиспользование индексов TopK% активаций для сжатия градиентов.

4. СВЯЗАННЫЕ РАБОТЫ

В этом разделе описываются смежные работы по распределенному обучению, в частности, по параллелизации по модели (model-parallel) и по данным (data-parallel), а также применение сжатия коммуникаций в таких сценариях.

Таблица 4. Результаты обучения ResNet18 с AQ-SGD и TopK сжатием на CIFAR-10

Степень сжатия	Точность на тестовой выборке (%), без сжатия	Точность на тестовой выборке (%), со сжатием
Без сжатия	**93.00**	**93.00**
AQ-SGD + Top 50%, warmup 10	**92.44**	**92.54**
AQ-SGD + Top 30%, warmup 10	**91.86**	**91.61**
AQ-SGD + Top 20%, warmup 10	90.82	87.8
AQ-SGD + Top 10%, warmup 10	85.91	84.16

Примечание. Каждый запуск обучения модели на 100 эпох. В запусках Warmup 10 используется предобученная модель за 10 эпох без сжатия. Степень параллелизации модели равна 4, используется 3 оператора сжатия. Активации и градиенты сжимаются независимо друг от друга. AQ-SGD используется только для активаций.

Параллелизация по данным (data parallelism)

Современные модели используют большие обучающие выборки данных, достигающие нескольких терабайт. В сценарии обучения на одном компьютере все примеры из обучающего набора данных должны итерироваться каждую эпоху, что занимает слишком много времени. Распределенное параллельное обучение разделяет обучающие данные между компьютерами [26]. Каждое устройство содержит полную копию обученной модели и обновляет локальную модель на каждом шаге обучения на своей части данных. Shallue и др. [27] показывают, что параллелизация по данным не оказывает негативного влияния на качество модели при практически одинаковом количестве итераций обучения. Параллелизация по данным широко используется в фреймворках для обучения больших моделей: Horovod [28], Pytorch Data-Parallel [29].

Параллелизация по модели (model parallelism)

Параллелизация по модели – это ортогональный подход. Модель разбивается на блоки слоев, каждый из которых хранится на отдельной машине. Размер назначенных блоков может варьироваться в соответствии с требованиями к памяти каждого устройства. Метод параллелизации по модели широко используется для обучения больших моделей в таких системах, как Megatron [5], Deepspeed [6], Petals [7].

Кроме того, в таком подходе к параллелизации обычно используется техника конвейеризации. Она позволяет распределенным системам наиболее эффективно использовать ресурсы за счет разбиения каждого батча данных на микробатчи, которые затем конвейеризуются на блочно-разделенной модели, как, например, в Gpipe [30], Xpipe [31], PipeDream [32]. Конвейерная обработка увеличивает использование ресурсов на каждой машине и уменьшает их простой.

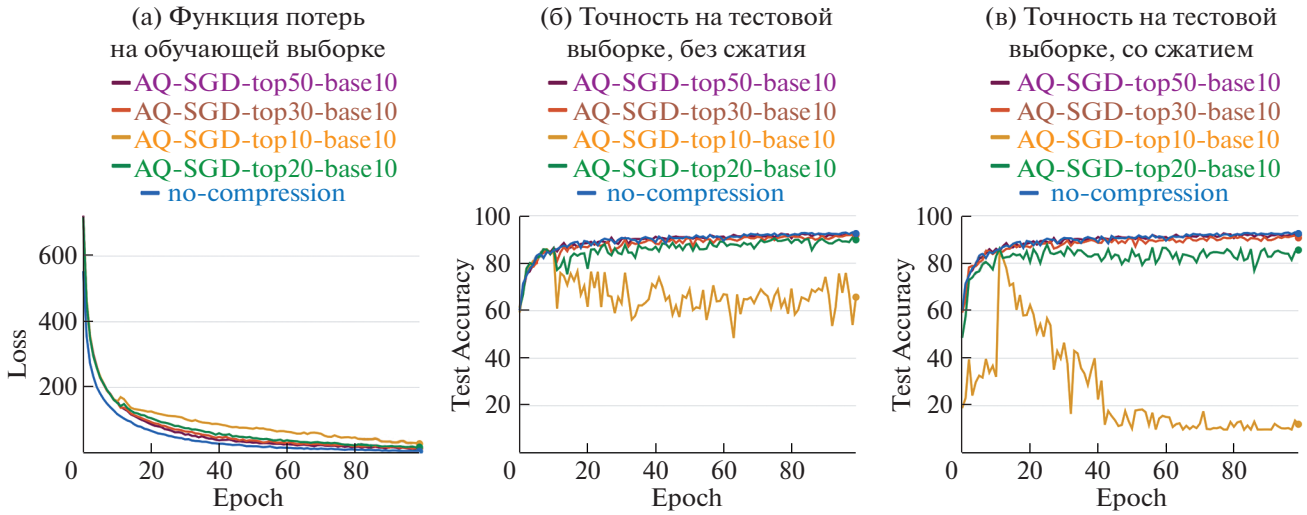

Рис. 5. Графики сходимости ResNet18 с AQ-SGD и TopK сжатием на CIFAR-10. Каждый запуск обучения модели на 100 эпох. В запусках Warmup 10 (base 10) используется предобученная модель за 10 эпох без сжатия. Степень параллелизации модели равна 4, используется 3 оператора сжатия. Активации и градиенты сжимаются независимо друг от друга. AQ-SGD используется только для активаций.

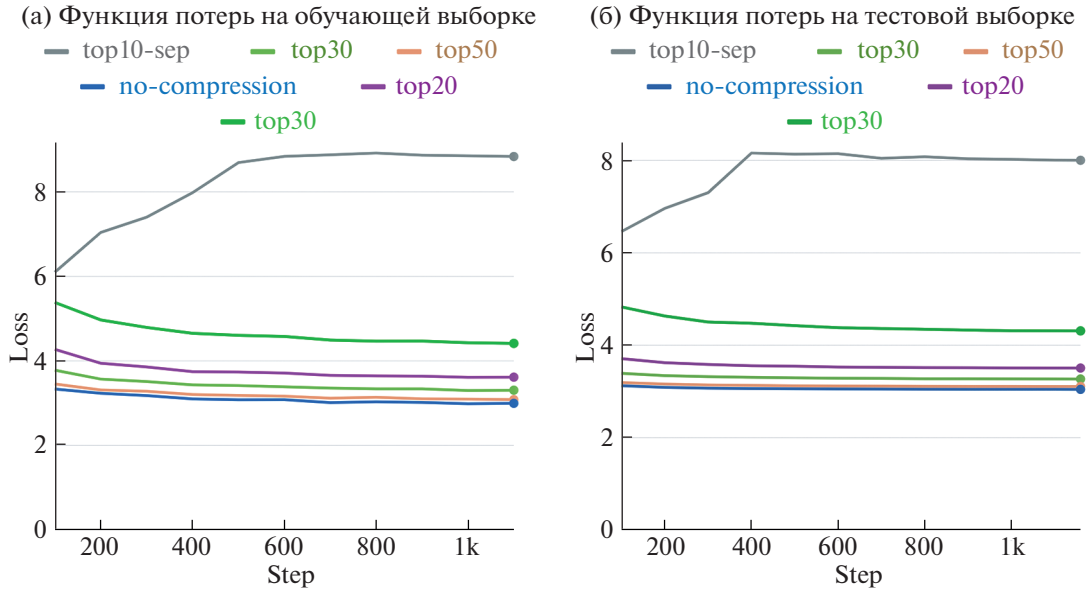

Рис. 6. Графики сходимости GPT-2 с AQ-SGD и TopK сжатием на Wikitext. Каждый запуск дообучения на 4 эпохи. Степень параллелизации модели равна 4, при этом используется 3 оператора сжатия. При сжатии TopK используются индексы TopK активаций для сжатия градиентов. TopK separate режим сжимает активации и градиенты независимо друг от друга.

Сжатие коммуникаций для распределенного обучения нейронных сетей

Узкое место в коммуникациях может оказаться критичным для распределенного обучения. Сжатие используется для уменьшения накладных расходов на пересылку. Обычно сжимаются активации слоев или их соответствующие градиенты, поскольку эта информация передается другим участникам системы распределенного обучения.

Сжатие градиентов

Сжатие градиентов естественным образом применяется в подходе параллелизации по данным. Были разработаны методы, позволяющие снизить коммуникационные затраты без существенного снижения качества модели.

Наиболее распространенным способом компрессии, занимаемого числами с плавающей точкой, является квантизация. В ряде работ, посвященных робастной квантизации, показано, что при определенных предположениях и используемых методиках обучение с квантизацией приводит к одинаковой сходимости и качеству модели [19, 33]. Alistarh и др. [34] разработали схему оценки сходимости градиентных методов со сжатием на основе квантизации. Методы квантизации с выбором уровней квантизации с учетом распределения достигают еще лучших практических результатов при обучении больших моделей [11, 35, 36]. Преимущество методов квантизации заключается в том, что они являются несмещенными, а значит, и в теории могут сходиться так же, как и при обучении без сжатия.

Другой класс методов сжатия основан на низкоранговой аппроксимации вектора градиентов. Wang и др. [37] предлагают сжатие путем спарсинга сингулярных значений градиентов. PowerSGD [38] также рассматривает сжатие градиентов за счет концепции power iteration. Подобные методы также применимы к параллелизму по модели: Фреймворк Optimus-CC показывает лучшие результаты по времени обучения при сжатии градиентов на основе PowerSGD [17]. Однако декомпо-

Таблица 5. Результаты дообучения GPT-2 с TopK сжатием на Wikitext

Степень сжатия	Значение функции потерь на тестовой выборке	Perplexity (PPL)
Без сжатия	**3.05**	**21.01**
Top 50%	**3.11**	**22.38**
Top 30%	**3.27**	**26.28**
Top 20%	3.51	33.32
Top 10%	4.31	74.51
Top 10% separate	8.0	2990.16

Примечание. Каждый запуск дообучения на 4 эпохи. Степень параллелизации модели равна 4, при этом используется 3 оператора сжатия. При сжатии TopK используются индексы TopK активаций для сжатия градиентов. TopK separate режим сжимает активации и градиенты независимо друг от друга.

зиция градиентов требует дополнительного вычислительного времени для сжатия.

Для борьбы с ошибками, возникающими при сжатии градиентов, были предложены методы компенсации ошибки (error feedback, EF), позволяющие улучшить теоретическую и практическую сходимость. Изначально предложенный Seide и др. [19] в качестве эвристики, EF показал себя как эффективный метод улучшения сходимости для задач с параллелизацией по данным. Более новый подход EF21 улучшает компенсацию ошибки EF, передавая изменения градиентов [21]. Алгоритмы MARINA [39] и DIANA [20] используют сжатие разностей градиентов для улучшения сходимости для сильно выпуклых и невыпуклых задач в постановке с параллелизацией по данным. Optimus-CC [17] также сжимает разности градиентов между мини-батчами, что напоминает подход EF21, но в контексте параллелизации по модели. В целом методы компенсации ошибки дают лучшие теоретические и практические результаты при параллелизации по данным, но лишь в некоторых работах EF используется при параллелизации по модели.

Наконец, для сжатия градиентов используются и операторы спрасификации. Alistarh и др. [18] показывают, что смещенные методы спарсификации, такие как TopK, могут приводить к лучшим результатам, чем несмещенные аналоги. Безносиков и др. [13] анализируют теоретические оценки смещенных методов сжатия. Они также предлагают улучшенное сжатие TopK с экспоненциальным дизерингом для достижения большей экономии на связи при сравнимой сходимости модели в обучении с параллелизацией по данным.

Сжатие активаций слоев

Хотя сжатие градиентов может применяться при параллелизации и по данным, и по модели, сжатие активаций обычно используется при применении или обучении модели в постановке с параллелизацией модели.

Существуют результаты, которые используют сжатие активаций для уменьшения потребления памяти, так как в процессе обучения необходимо хранить все промежуточные активации для подсчета градиентов. В AC-GC [10] используется квантизация по фиксированным уровням дискретизации для сжатия активаций в 15 раз со средней потерей точности всего 0.1 процентных пункта. Fu и др. [11] также используют квантизацию с учетом распределения для сжатия активаций и градиентов, уменьшая потребление памяти до 2–4 раз. Подход LLM-int8() [9] использует 8-битное квантование для ускорения применения больших языковых моделей без ухудшения качества. В этом подходе также учитываются значения данных-выбросов; они передаются в исходном формате fp32.

Работа AQ-SGD [16] представляет применение подхода с техникой компенсации ошибки сжатия активаций при параллелизации модели. Этот метод передает квантизованные разности активаций между блоками конвейера, достигая увеличения общей пропускной способности обучения до 8.5 раза в медленных сетях связи.

В экспериментальной работе по методам сжатия активаций, проведенной Bian и др. [14], сравниваются популярные компрессоры для обучения с параллелизацией по модели и делается ряд выводов о практическом применении сжатия активаций. В частности, отмечается, что автоэнкодерное сжатие, основанное на обучении, демонстрирует лучшие результаты по сходимости и пропускной способности обучения по сравнению с квантизацией и TopK-сжатием.

5. ЗАКЛЮЧЕНИЕ

В данной работе мы провели практический анализ сжатия активаций и градиентов для распределенного обучения в режиме параллелизации по модели для глубоких нейронных сетей. В частности, мы исследовали, как квантизация, сжатие TopK и подходы, основанные на компенсации ошибки, работают в разных задачах машинного обучения.

Как показали эксперименты с квантизацией, градиенты модели более чувствительны к сжатию, чем активации. Мы также эмпирически показали, что TopK сжатие как для активаций, так и для градиентов может быть применено не более чем на уровне $K = 10\%$ для достижения сопоставимого качества модели. В работе также исследовались методы компенсации ошибки. В частности, мы не наблюдали существенного улучшения сходимости при применении TopK сжатия активаций и градиентов при параллелизации по модели. Однако методы компенсации ошибки помогли преодолеть падение качества предсказаний модели без применения сжатия. Наконец, мы обнаружили, что подход AQ-SGD [16] не может быть применен с “агрессивным” сжатием TopK при степени сжатия TopK больше, чем 30%, для достижения удовлетворительной сходимости.

Следует отметить несколько потенциальных ограничений нашего исследования. Во-первых, поскольку параллелизация по модели обычно применяется для больших моделей, в данной работе отсутствуют эксперименты с обучением самых современных больших моделей. Во-вторых, в каждом эксперименте мы тестируем только одну конфигурацию пареллизации по модели и конфигурацию гиперпараметров.

На основе нашей работы мы предлагаем несколько направлений будущих исследований: эксперименты с современными языковыми моделями, уменьшение объема памяти подхода AQ-SGD и изучение других смещенных методов сжатия помимо TopK.

БЛАГОДАРНОСТИ

Мы выражаем глубокую благодарность самому незаменимому и самому скрытному сотруднику Yandex.Research за содержательные дискуссии в ходе наших исследований. Мы также благодарим Антона Антонова за помощь в предоставлении вычислительных ресурсов для проекта.

ИСТОЧНИКИ ФИНАНСИРОВАНИЯ

Исследования А. Безносикова были поддержаны Российским научным фондом (проект № 23-11-00229).

СПИСОК ЛИТЕРАТУРЫ

1. *Open A.I.*, GPT-4 Technical Report, 2023. arXiv: 2303.08774 [cs.CL].
2. *Scao T.L., Fan A., Akiki C. et al.* "BLOOM: A 176B-Parameter Open-Access Multilingual Language Model," arXiv preprint arXiv:2211.05100, 2022.
3. *Laurencon H., Saulnier L., Wang T. et al.* "The bigscience roots corpus: A 1.6 tb composite multilingual dataset," в Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2022.
4. *Verbraeken J., Wolting M., Katzy J., Kloppenburg J., Verbelen T., Rellermeyer J.S.* "A survey on distributed machine learning," Acm computing surveys (csur). 2020. Т. 53, № 2. С. 1–33.
5. *Shoeybi M., Patwary M., Puri R., LeGresley P., Casper J., Catanzaro B.* "Megatron-lm: Training multi-billion parameter language models using model parallelism," arXiv preprint arXiv:1909.08053, 2019.
6. *Rasley J., Rajbhandari S., Ruwase O., He Y.* "Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters," в Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020. С. 3505–3506.
7. *Borzunov A., Baranchuk D., Dettmers T. et al.* "Petals: Collaborative Inference and Fine-tuning of Large Models," arXiv preprint arXiv:2209.01188, 2022.
8. *Diskin M., Bukhtiyarov A., Ryabinin M. et al.* "Distributed deep learning in open collaborations," Advances in Neural Information Processing Systems. 2021. Т. 34. С. 7879–7897.
9. *Dettmers T., Lewis M., Belkada Y., Zettlemoyer L.* "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale," Advances in Neural Information Processing Systems. 2022. Т. 35. С. 30 318–30 332.
10. *Evans R.D., Aamodt T.* "Ac-gc: Lossy activation compression with guaranteed convergence," Advances in Neural Information Processing Systems. 2021. Т. 34. С. 27 434–27 448.
11. *Fu F., Hu Y., He Y. et al.* "Don't waste your bits! squeeze activations and gradients for deep neural networks via tinyscript," в International Conference on Machine Learning, PMLR, 2020. С. 3304–3314.
12. *Stich S.U., Cordonnier J.-B., Jaggi M.* "Sparsified SGD with memory," Advances in Neural Information Processing Systems. 2018. Т. 31.
13. *Beznosikov A., Horvath S., Richtarik P., Safaryan M.* "On biased compression for distributed learning," arXiv preprint arXiv:2002.12410, 2020.
14. *Bian S., Li D., Wang H., Xing E.P., Venkataraman S.* Does compressing activations help model parallel training? 2023. arXiv: 2301.02654 [cs.LG].
15. *Gupta V., Choudhary D., Tang P. et al.* "Training recommender systems at scale: Communication-efficient model and data parallelism," в Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021. С. 2928–2936.
16. *Wang J., Yuan B., Rimanic L. et al.,* "Fine-tuning Language Models over Slow Networks using Activation Quantization with Guarantees," Advances in Neural Information Processing Systems. 2022. Т. 35. С. 19 215–19 230.
17. *Song J., Yim J., Jung J. et al.* "Optimus-CC: Efficient Large NLP Model Training with 3D Parallelism Aware Communication Compression," в Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 2023. V. 2. С. 560–573.
18. *Alistarh D., Hoefler T., Johansson M., Konstantinov N., Khirirat S., Renggli C.* "The convergence of sparsified gradient methods," Advances in Neural Information Processing Systems. 2018. Т. 31.
19. *Seide F., Fu H., Droppo J., Li G., Yu D.* "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns," в Fifteenth annual conference of the international speech communication association, 2014.
20. *Mishchenko K., Gorbunov E., Takač M., Richtarik P.* "Distributed learning with compressed gradient differences," arXiv preprint arXiv:1901.09269, 2019.
21. *Richtarik P., Sokolov I., Fatkhullin I.* "EF21: A new, simpler, theoretically better, and practically faster error feedback," Advances in Neural Information Processing Systems. 2021. Т. 34. С. 4384–4396.
22. *He K., Zhang X., Ren S., Sun J.* "Deep residual learning for image recognition," в Proceedings of the IEEE conference on computer vision and pattern recognition, 2016. С. 770–778.
23. *Krizhevsky A., Hinton G. et al.*, "Learning multiple layers of features from tiny images," 2009.
24. *Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I. et al.*, "Language models are unsupervised multitask learners," OpenAI blog. 2019. Т. 1. № 8. С. 9.
25. *Merity S., Xiong C., Bradbury J., Socher R.* "Pointer Sentinel Mixture Models," в International Conference on Learning Representations, 2016.
26. *Krizhevsky A.* "One weird trick for parallelizing convolutional neural networks," arXiv preprint arXiv:1404.5997, 2014.

27. *Shallue C.J., Lee J., Antognini J., Sohl-Dickstein J., Frostig R., Dahl G.E.* "Measuring the effects of data parallelism on neural network training," arXiv preprint arXiv:1811.03600, 2018.

28. *Sergeev A., Del Balso M.* "Horovod: fast and easy distributed deep learning in TensorFlow," arXiv preprint arXiv:1802.05799, 2018.

29. *Li S., Zhao Y., Varma R. et al.,* "Pytorch distributed: Experiences on accelerating data parallel training," arXiv preprint arXiv:2006.15704, 2020.

30. *Huang Y., Cheng Y., Bapna A. et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," Advances in neural information processing systems. 2019. T. 32.

31. *Guan L., Yin W., Li D., Lu X.* "XPipe: Efficient pipeline model parallelism for multi-GPU DNN training," arXiv preprint arXiv:1911.04610, 2019.

32. *Harlap A., Narayanan D., Phanishayee A. et al.*, "Pipedream: Fast and efficient pipeline parallel dnn training," arXiv preprint arXiv:1806.03377, 2018.

33. *Bernstein J., Wang Y.-X., Azizzadenesheli K., Anandkumar A.* "signSGD: Compressed optimization for nonconvex problems," в International Conference on Machine Learning, PMLR, 2018. С. 560–569.

34. *Alistarh D., Grubic D., Li J., Tomioka R., Vojnovic M.* "QSGD: Communication-efficient SGD via gradient quantization and encoding," Advances in neural information processing systems. 2017. T. 30.

35. *Han S., Mao H., Dally W.J.* "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv preprint arXiv:1510.00149, 2015.

36. *Hong C., Kim H., Baik S., Oh J., Lee K.M.* "Daq: Channel-wise distribution-aware quantization for deep image super-resolution networks," в Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2022. С. 2675–2684.

37. *Wang H., Sievert S., Liu S., Charles Z., Papailiopoulos D., Wright S.* "Atomo: Communication-efficient learning via atomic sparsification," Advances in Neural Information Processing Systems. 2018. T. 31.

38. *Vogels T., Karimireddy S.P., Jaggi M.* "PowerSGD: Practical low-rank gradient compression for distributed optimization," Advances in Neural Information Processing Systems. 2019. T. 32.

39. *Gorbunov E., Burlachenko K.P., Li Z., Richt'arik P.* "MARINA: Faster non-convex distributed learning with compression," в International Conference on Machine Learning, PMLR, 2021. С. 3788–3798.

ACTIVATIONS AND GRADIENTS COMPRESSION FOR MODEL-PARALLEL TRAINING

M. Rudakov[a], A. Beznosikov[a,b], Y. Kholodov[a], and A. Gasnikov[a,b]

[a] *Innopolis University, Innopolis, Republic of Tatarstan, Russia*

[b] *Moscow Institute of Physics and Technology, Moscow, Russia*

Presented by Academician of the RAS A.L. Semenov

Large neural networks require enormous computational clusters of machines. Model-parallel training, when the model architecture is partitioned sequentially between workers, is a popular approach for training modern models. Information compression can be applied to decrease workers' communication time, as it is often a bottleneck in such systems. This work explores how simultaneous compression of activations and gradients in model-parallel distributed training setup affects convergence. We analyze compression methods such as quantization and TopK compression, and also experiment with error compensation techniques. Moreover, we employ TopK with AQ-SGD per-batch error feedback approach. We conduct experiments on image classification and language model fine-tuning tasks. Our findings demonstrate that gradients require milder compression rates than activations. We observe that $K = 10\%$ is the highest TopK compression level, which does not harm model convergence severely. Experiments also show that models trained with TopK perform well only when compression is also applied during inference. We find that error feedback techniques do not improve model-parallel training compared to plain compression, but allow model inference without compression with almost no quality drop. Finally, when applied with the AQ-SGD approach, TopK stronger than with $K = 30\%$ worsens model performance significantly.

Keywords: distributed learning, model parallelism, activation compression, gradient compression