Сжатие нейронных сетей для предсказания временных рядов

Ахтырченко Михаил, Брескану Никита, Дмитрий Каратыщев, Егор Сильверстов

Ноябрь, 2024

Описание моделей

Исследовались 3 модели: TimesNet [1], Transformer [2] и iTransformer [3].

Для них будут применяться различные алгоритмы сжатия. Задача: уменьшить число параметров модели, при этом не сильно уменьшив её качество (accuracy) на тестовом датасете.

Модель	# параметров	Accuracy	Loss
TimesNet	895442	0.810	0.494
Transformer	164354	0.771	0.495
iTransformer	134338	0.751	0.545

Таблица 1: *Модели*

Спарсификация

Будет исследоваться малоранговое приближение, gradient-based и magnitude-based методы [4].

Исследование структуры TimesNet

Спарсификация проводилась только для модели TimesNet. Перед тем как применять спарсификацию, стоит ознакомиться со структурой модели.

Тип слоя	# параметров	% от суммарного
Conv2d	879456	98.2
Linear	13026	1.5
Other	2960	0.3

Таблица 2: *Структура TimesNet по числу параметров. Основную часть модели занимают Conv2d слои.*

Видно, что модель состоит практически полностью из свёрточных слоёв (2). Спарсификация будет применяться только к свёрточным и линейным слоям.

Описание Finetune

После спарсификации модель дообучалась 10 эпох и брался лучший результат по тестовой выборке. Это немного нечестно, но к сожалению, датасет не предусматривает валидационную выборку.

Структурная спарсификация

Малоранговое приближение Свёрточный слой Conv2d имеет следующую структуру весов:

$$W_{\text{conv}} \in \mathbb{R}^{C_o \times C_i \times K_h \times K_w}$$

, где C_o, C_i — число выходных и входных каналов, K_w, K_h - размеры окна.

Далее применяется следующее разделение весов:

$$W'_{\text{conv}} = \text{reshape}(W_{\text{conv}}) \in \mathbb{R}^{C_o \times C_i \cdot K_h \cdot K_w}$$
$$U, S, V^T = \text{SVD}(W'_{\text{conv}})$$
$$W^1_{\text{conv}} = \text{reshape}(V^T) \in \mathbb{R}^{r \times C_i \times K_h \times K_w}$$
$$W^2_{\text{conv}} = US \in \mathbb{R}^{C_o \times r \times 1 \times 1}$$

Таким образом, свёрточный слой W_{conv} разбился на 2 слоя W^1_{conv} и W^2_{conv}. Первой слой переводит изображение в r каналов и имеет такое же окно, как и исходный слой. Второй слой имеет окно 1×1 и переводит из r каналов в нужное число выходных каналов C_o.

Для линейных слоёв всё существенно проще:

$$W_{\text{linear}} \in \mathbb{R}^{N_o \times N_i}$$
$$U, S, V^T = \text{SVD}(W_{\text{linear}})$$
$$W^1_{\text{linear}} = V^T \in \mathbb{R}^{r \times N_i}$$
$$W^2_{\text{linear}} = US \in \mathbb{R}^{N_o \times r}$$

Таким образом, в свёрточных слоях происходит малоранговое приближение по числу каналов, а в линейных — по числу признаков.

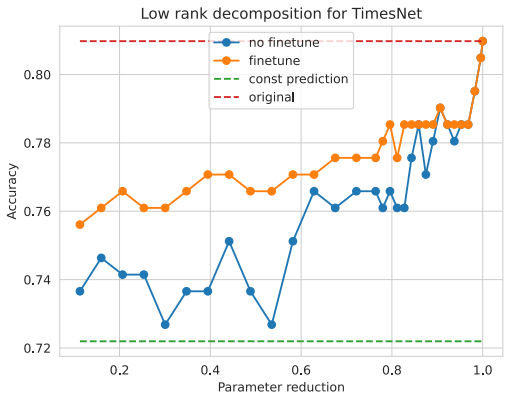

Рис. 1: *Результаты для малорангового приближения*

Gradient-based Рассматривается ещё один способ структурной спарсификации — выкидывание свёрточных слоёв по присвоенным им показателям важности. Эти показатели вычисляются как средняя норма градиента для выходного канала. Градиент суммарный по всей тренировочной выборке. То есть, чем больше градиент течёт через слой, тем он важнее.

Теперь, можно отсортировать слои по их важности и полностью удалять самые неважные.

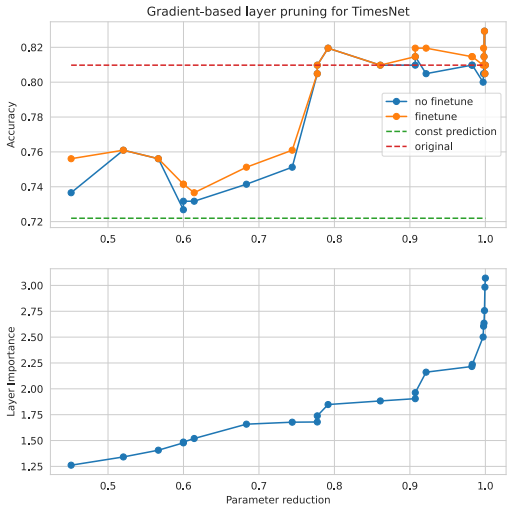

Рис. 2: *Gradient-based спарсификация*

Неструктурная спарсификация

Magnitude-based Магнитудный способ спарсификации очень простой: в каждом слое (линейном и свёрточном) зануляется фиксированная доля весом с минимальным модулем. При этом, разумееется, при дообучении на занулённые веса навешена маска, поэтому после каждой эпохи эти веса снова зануляются.

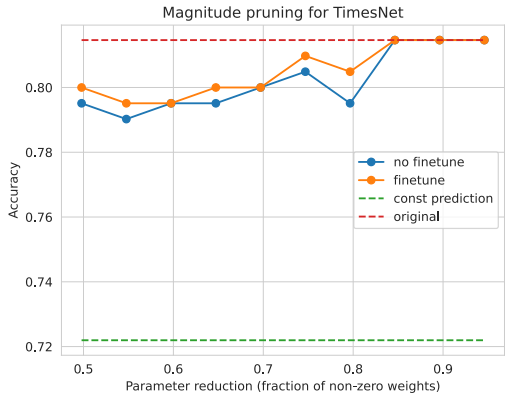

Рис. 3: *Magnitude-based спарсификация*

Сравнение уровней 50% и 75%

В обоих уровнях 50% и 75% наибольший Accuracy имеет магнитудный метод (таблицы 3, 4). Однако стоит понимать, что в нём идёт только зануление весов, поэтому такое сравнение немного нечестное. В остальных 2 методах уменьшение параметров честное.

Метод	% params	ACC	ACC (finetune)
low-rank	48.9	0.736	0.766
gradient-based	52.0	0.761	0.761
magnitude-based	49.7	**0.795**	**0.800**

Таблица 3: *Сравнение методов для уровня разреженности 50%*

Метод	% params	ACC	ACC (finetune)
low-rank	76.4	0.766	0.775
gradient-based	74.4	0.751	0.761
magnitude-based	74.6	**0.805**	**0.810**

Таблица 4: *Сравнение методов для уровня разреженности 75%*

Вывод

Удалось применить 3 метода спарсификации. Магнитудный метод лучше всего уменьшает параметры, однако зануление весов меньше увеличит скорость работы, чем явное удаление параметров. Интересным наблюдением оказалось то, что свёрточные слои TimesNet с ядром 1 x 1 имеют наименьшую важность, и их удаление не уменьшало качество, т.е. можно халявно убрать 10% параметров (2). Судя по результатам, для TimesNet gradient-based importance pruning лучше, чем low-rank approximation. Оба метода реализованы и честно уменьшают число параметров путём замены слоёв.

Дистилляция знаний

Общие условия экспериментов

В качестве модели учителя использовалась предобученная модель TimesNet, а в качестве студентов использовались модели Transformer и iTransformer. В качестве параметров моделей и обучения были выбраны параметры из предоставленного ноутбука. В качестве целевой задачи использовалась задача классификации для датасета Heartbeat.

KL knowledge distillation [5]

$(tLogit_{il})_{i=1,l=1}^{N,L}$ — логиты предсказния учителя, (1)

$(sLogit_{il})_{i=1,l=1}^{N,L}$ — логиты предсказния ученика, (2)

$(r_i)_{i=1}^{N}$ — настоящие значения, (3)

$$(s_{il})_{l=1}^{L} = \underset{l=\overline{1L}}{\mathbf{Softmax}}(\frac{sLogit_{il}}{T})- \tag{4}$$

вероятности предсказания учеником,

$$(t_{il})_{l=1}^{L} = \underset{l=\overline{1L}}{\mathbf{Softmax}}(\frac{tLogit_{il}}{T})- \tag{5}$$

вероятности предсказания учителем

Данный метод дистилляции предлагает заменить исходную функцию потерь на:

$$\mathcal{L} = (1-\alpha)\cdot\frac{1}{N}\sum_{i=1}^{N}\mathbf{CrossEntropy}(s_i, r_i)+ \\ +\alpha\cdot\frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{L}t_{il}(\log(t_{il})-\log(s_{il})) \tag{6}$$

В экперименте использовался перебор α по сетке от 0.0 до 1.0 с 10 шагами, $T = 2$. Отметим, что случае $\alpha = 0$ мы используем только кроссэнтропию, что равносильно исходной задаче, при $\alpha = 0$ мы не используем таргет, лишь ответы учителя.

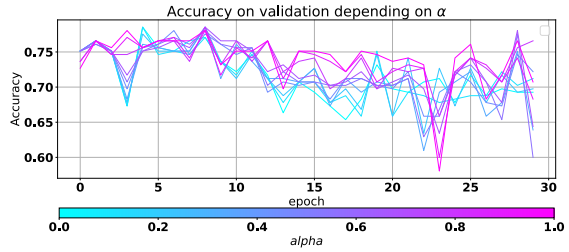

Рис. 4: *Результаты для дистилляции TimesNet в Transformer с применением KL добавки*

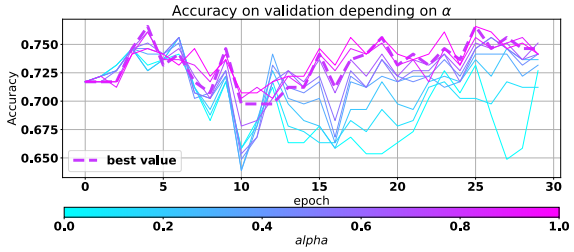

Рис. 5: *Результаты для дистилляции TimesNet в iTransformer с применением KL добавки*

Как видно из графиков, добавление KL добавки в лосс делает обучение стабильнее и уменьшает переобучение. KL дивиргенция позволяет модели ученику не переобучаться под ответы, так как в отличие от кросс энтропии, для которой важна вероятность только истинного класса, в данном случае мы приближаем распределение ученика к распределению учителя целиком и не даем ему выродиться, как происходит при переобучении. В силу малого количества тестовых примеров и сильного переобучения начиная с 9-11 эпохи заметить значительного прироста в метрике не удалось, хотя лучшие значения на валидации на обоих моделях достигаются при $\alpha \sim 0.7$.

OFA knowledge distillation [6]

В данном методе предлагается использовать не только выходные логиты модели но и добавить промежуточные логиты, достигается это добавлением проекционного оператора на выходы промежуточных слоев. Пусть всего будет F выходов, включая основной выход модели

$$(tLogit_{il})_{i=1,l=1}^{N,L} - \text{логиты предсказния учителя,} \tag{7}$$

$$(sLogit_{ilf})_{i=1,l=1,f=1}^{N,L,F} - \text{логиты предсказния ученика,} \tag{8}$$

$$(r_i)_{i=1}^{N} - \text{настоящие значения,} \tag{9}$$

$$(s_{ilf})_{l=1}^{L} = \underset{l=\overline{1L}}{\mathbf{Softmax}}(\frac{sLogit_{il}}{T})- \tag{10}$$

вероятности предсказания учеником,

$$(t_{il})_{l=1}^{L} = \underset{l=\overline{1L}}{\mathbf{Softmax}}(\frac{tLogit_{il}}{T})- \tag{11}$$

вероятности предсказания учителем

Данный метод дистилляции предлагает заменить исходную функцию потерь на:

$$\mathcal{L} = (1-\text{ofa\_loss\_weight}-\text{kd\_loss\_weight})\cdot \\ \frac{1}{N}\sum_{i=1}^{N}\mathbf{CrossEntropy}(s_i, r_i)+ \\ \text{kd\_loss\_weight}\cdot\frac{1}{N}\sum_{i=1}^{N}\mathbf{OFA}((s_{ilF})_{l=1}^{L}, t_i, \gamma_F)+ \\ \text{ofa\_loss\_weight}\cdot\frac{1}{N}\sum_{i=1}^{N}\sum_{f=1}^{F}\mathbf{OFA}((s_{ilf})_{l=1}^{L}, t_i, \gamma_f) \tag{12}$$

$$\mathbf{OFA}(s,t,\gamma) = -(1+t_{ir_i})^{\gamma}\cdot\log(s_{ir_i})+ \\ \frac{1}{L}\sum_{l=1}^{L}t_{il}(\log(t_{il})-\log(s_{il})) \tag{13}$$

В эксперименте использовался перебор по сетке с 5 шагами внутри симплекса, задающегося координатами (ofa_loss_weight, kd_loss_weight, 1-kd_loss_weight-ofa_loss_weight). $T = 2, \gamma_f = 1.0, f = \overline{1,F}$. Такой выбор γ обусловлен наблюдениями авторов статьи, у которых оптимальные значения γ принимали значения в диапазоне от 1.0 до 1.5. В качестве проекционного оператора использовался 1 слой MLP с Layer нормализацией и функцией активацией Gelu перед ним. Отметим, что в случае kd_loss_weight $= 0$, ofa_loss_weight $= 0$ мы используем только кроссэнтропию, ofa_loss_weight $= 0$ мы используем только кроссэнтропию и KL дивергенцию ($\gamma = 1$)). При проведении экспериментов использовался код из github репозитория авторов статьи.

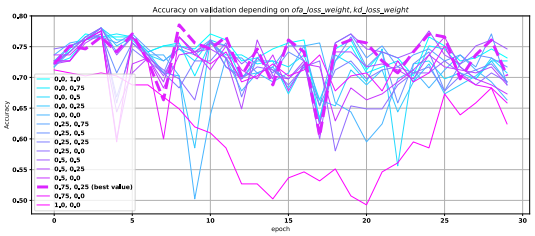

Рис. 6: *Результаты для дистилляции TimesNet в Transformer с применением OFA-KD loss*

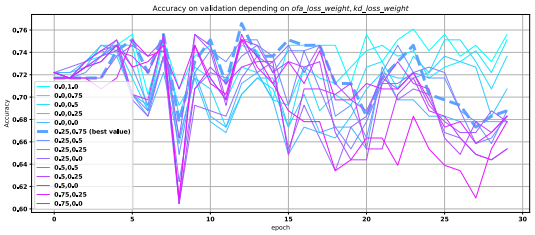

Рис. 7: *Результаты для дистилляции TimesNet в iTransformer с применением OFA-KD loss*

Как видно, при добавлении в loss логитов с промежуточных слоев, в случае iTransformer вновь появляется переобучение, в силу того, что проекционные операторы имеют существенное количество параметров, так как применяются к выходу размерности ($seq\_len \cdot d\_model$). Поэтому для данного метода дистилляции лучше использовать Transformer. В силу причин перечисленных в предыдущих пунктах, заметного улучшения метрики достич не получилось, хотя наилучшее значение для Transformer достигается при соотношении 3 к 1 между lossами с промежуточных слоев и последнего, без использования кроссэнтропии. С другой стороны для iTransformer, соотношение 1 к 3 в силу сильного переобчения, из-за которого требуется большая добавка KL лосса, который как понятно из предыдущего метода борется с переобучением.

Сравнение 2х методов

Сравнив 2 метода дистилляции на 2х моделях можно сделать следующие выводы:

- 2 ой метод дает больше свободы и обобщает 1 ый метод, причем данное обобщение дает незначительный прирост в метриках, поэтому при возможности стоит использовать второй метод
- OFA-KD не стоить использовать с моделями-учениками, которые при создании логитов для задачи классификации, в качестве проекционного оператора используют оператор количество входов которого зависит от длины последовательности. Это может привести к сильному переобучению
- Transformer предпочтительнее iTransformer, по причине того, что в качестве проекционного оператора использует оператор количество входов которого не зависит от длины последовательности.
- Дистилляция дает незначительный прирост в метриках, но при этом в отдельных случаях хорошо борется с переобучением.

Квантизация

Post Training Quantization

Для квантизации моделей предлагается использовать квантильную квантизацию.

Пусть имеется массив w (если веса какой-то части модели - тензор, то вытянем в массив, ниже описано как именно этот процесс происходил с TimesNet) - веса какой-то части модели, тип $float32$ преобразуется в $int4$ и $int8$ с помощью следующего алгоритма

- Значения сортируются w по возрастанию
- В зависимости от типа $int4$ или $int8$ массив делится на 16 или 256 равных частей (если длина не делится на 16 или 256, то количество элементов в одной части округляется вверх, считая что последняя часть содержит меньше остальных значений; если длина меньше 16 или 256, то алгоритм делит на то количество частей, какова длина массива) соответственно
- Для каждой части считается среднее значение
- Каждой части (соответственно и среднему значению части) ставится в соответствие целое число
- Вещественные числа исходного $float32$ массива заменяются целыми числами ($int4$, $int8$) - номерами тех частей, в которые они попали

При инференсе модели через целые значения каждому весу присваивается вещественное среднее значение той части, в которую оно попало при квантизации. Плюсом такого подхода является то, что он позволяет кодировать веса в целых числах с учетом дисбаланса вещественных значений, таким образом экономя память более эффективно. Минусом такого подхода является то, что он требует дополнительных вычислений при инференсе и хранения дополнительной информации.

Для модели TimesNet все свертки были квантованы в рамках ядер (то есть w - вытянутое 1 ядро), для полносвзяных слоев w - вытянутая матрица 1 слоя, для весов, которые уже хранятся в виде одномерных векторов w, квантование проводилось без дополнительных преобразований. Результат сравнения качества на тесте датасета *Heartbit* и веса модели при разной степени квантизации показан на Таблице 5. Качество после квантизации стало хуже ($int8$ на 0.005 хуже), однако при $int4$ занимаемой памяти в **8** раз меньше (при $int8$ выигрыш по памяти почти в 3 раза меньше), но каче-

Таблица 5: *Сравнение качества обученного TimesNet после квантильной квантизации*

Модель	Accuracy	Память (Mb)
TimesNet fp32	0.8097	3.4158
TimesNet int8	0.8048	1.5436
TimesNet int4	0.7951	0.4831

ство хуже на 0.014, что может быть не такой большой потерей в конкретной задаче.

Quantization Aware Training

Для QAT тоже использовалась квантильная квантизация (обазначим её $Q(.)$), при инференсе требуется деквантизовать приближенными значениями (обозначим $R(.)$). Работу модели обозначим $F(.,.)$, где первый аргумент - входной объект (x), второй - веса (W). Пусть на шаге n у модели квантованные веса W_n, используется следующая схема:

$$\overline{W_n} = R(W_n)$$

$$y = F(x, \overline{W_n})$$

$$W_{n+1} = Q(\overline{W_n} - \alpha \frac{Loss(x,y)}{\partial \overline{W_n}})$$

Изначально W_0 случайно инициализируем. Стоит отметить, что такой алгоритм отличается от STE небольшой поправкой в последнем равенстве, где градиент вычитается из матрицы $\overline{W_n}$ и сам градиент брался по ней же, такая поправка мотивирована тем, что возможно обучение будет более стабильным, в задании не было четко прописано, что реализация QAT должна быть для существующего алгоритма, поэтому была предложена такая модификация SET в качестве еще одного способа QAT.

Результаты тренировки видны на рис. 8 (график прерывается на 20-ой эпохе так как обучение остановилось по критерию останова). Без QAT обучение стабильнее, при $int8$ модель улучшается со временем, но при $int4$ обучение слишком нестабильно и модель медленно учится. Результаты по качеству работы на тесте датасета *Heartbit* после обучения (обучения, которое на рис. 8), где по колонкам техника при обучении (обычное обучение fp32 или тип в QAT), по строкам квантизация после обучения (тип в PTQ) или не квантизация (при значении fp32), на пересечении Accuracy, представлены в таблице 6. Так как модель из результатов в Таблице 5 обучалась дольше, то у нее видна разница при квантизации, но обучение в Таблице 6 было менее долгим и разницы не видно, поэтому на этот результат ориентироваться нельзя. Так как обучение $int8$ было более стабильно (чем у $int4$) и оно учитывало квантизацию при обучении, то можно заметить, что уже после PTQ точность сильно не меняется, именно этим полезен QAT, имеется возможность обучать

Рис. 8: *Сравнение процесса обучения модели при разной квантизации*

Таблица 6: *Сравнение качества обученного TimesNet, по колонкам техника при обучении (обычное обучение fp32 или тип в QAT), по строкам квантизация после обучения (тип в PTQ) или не квантизация (при значении fp32), на пересечении Accuracy*

Модель	fp32	QAT int8	QAT int4
fp32	0.7609	0.7609	0.7414
PTQ int8	0.7609	0.7560	0.7219
PTQ int4	0.7609	0.7707	0.7024

модель, закладывая информацию о квантизации в процесс обучения (в процессе обучения есть понимание о качестве модели после квантования).

Исходя из результатов PTQ и QAT можно сделать вывод, что только для PTQ лучше использовать $int4$, так как в качестве теряется не пропорционально меньше, чем в весе модели, а для QAT и уже последующего PTQ - $int8$, так как обучение стабильнее и быстрее чем у $int4$, однако обучаться нужно дольше, чем без квантизации.

Parameter-efficient fine-tuning

QLoRA

Метод QLoRA (*Quantized Low-Rank Adaptation*) [7] представляет собой технику уменьшения вычислительных затрат при адаптации больших языковых моделей (LLM). Данный метод сочетает два ключевых подхода: **квантование** параметров модели и использование **низкоранговых адаптаций (LoRA)**. Основной целью QLoRA является снижение аппаратных требований для обучения и инференса, сохраняя при этом высокую точность.

LoRA (*Low-Rank Adaptation*) [8] позволяет уменьшить количество обновляемых параметров, сохраняя

основные параметры модели неизменными и добавляя низкоранговые матрицы. Это достигается за счёт добавления корректировок в параметры модели через низкоранговую аппроксимацию.

Математическая основа LoRA

Пусть $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ — параметры слоя в модели. Вместо обновления всей матрицы $\mathbf{W}$, LoRA добавляет низкоранговую матрицу $\Delta\mathbf{W}$:

$$\mathbf{W}' = \mathbf{W} + \Delta\mathbf{W},$$

где $\Delta\mathbf{W}$ аппроксимируется как произведение двух низкоранговых матриц:

$$\Delta\mathbf{W} = \mathbf{BA}, \quad \mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}, \ \mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$$

Здесь $r \ll \min(d_{\text{out}}, d_{\text{in}})$ — ранг аппроксимации.

Во время обучения обновляются только $\mathbf{A}$ и $\mathbf{B}$, а основная матрица $\mathbf{W}$ остаётся неизменной. Это значительно снижает вычислительную сложность и объём хранимых градиентов.

Преимущества LoRA

- Снижение числа параметров, которые необходимо обновлять.
- Поддержка мультимодального обучения: одна базовая модель может быть адаптирована под разные задачи.
- Совместимость с другими методами, такими как квантование.

Совмещение LoRA и квантования

При использовании QLoRA модель начинает с квантованных параметров $\hat{\mathbf{W}}$, а добавляемая низкоранговая матрица $\Delta\mathbf{W}$ используется для внесения адаптивных изменений. Полная формула для слоя принимает вид:

$$\mathbf{W}' = \hat{\mathbf{W}} + \mathbf{BA}.$$

Обучение QLoRA

Во время обучения:

- Основные параметры $\hat{\mathbf{W}}$ остаются неизменными.
- Обновляются только низкоранговые матрицы $\mathbf{A}$ и $\mathbf{B}$.
- Градиенты вычисляются только для $\mathbf{A}$ и $\mathbf{B}$, что снижает вычислительную сложность.

Практическая реализация

Для реализации метода QLoRA были созданы модули-обёртки для линейных и свёрточных слоёв исходной модели. Основная модель при этом квантуется, после чего её слои заменяются соответствующими слоями-обёртками. В соответствующих модулях были имплементированы два основных параметра - это обучаемые матрицы A и B. После обучения на инференсе основная матрица весов обновляется через сложение с BA. Это позволяет использовать модель без лишних расчётов.

Для реализации обёртки для свёрточных слоёв была использована статья LoRA for Convolution layers

Эксперименты

В качестве первого эксперимента проводился подбор минимального значения ранга, при котором количество параметров модели, сквантизованной в 4 бита, будет совпадать с точки зрения количества бит с количеством параметров модели, сквантизованной в 8 бит. Полученный результат - $rank = 7$ при количестве параметров 4-х битной модели 227150.25 и количестве параметров 8-ми битной модели 223860.5

Второй эксперимент проводился с целью оценки производительности реализованного метода QLoRA на предобученной модели TimesNet. Приведём результаты валидационного лосса для разных рангов. 9

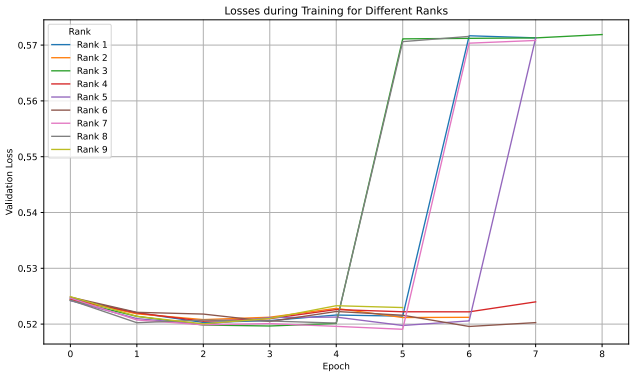

Рис. 9: *Результаты валидационного лосса при обучении QLoRA для разных рангов*

Приведём также результаты точности на тестовой выборке после обучения для разных рангов. 10

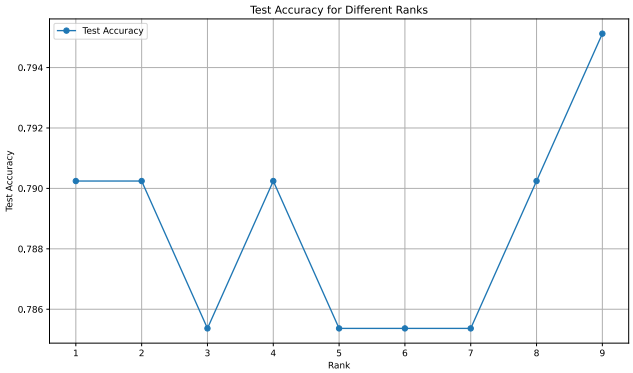

Рис. 10: *Результаты точности на тестовой выборке после обучения QLoRA*

Видно, что эксперименты вышли не особо удачными, видно явное переобучение, что может быть связано как с небольшими размерами выборки, так и возможными ошибками в чистой реализации метода QLoRA. Однако важно отметить, что fine-tuning проводился на той же выборке, на которой проводилось само обучение, что, очевидно, ведёт к переобучению. Время обучения не замерялось, однако, субъективно, файн-тьюнинг через QLoRA работает гораздо быстрее.

QDoRA: Weight-Decomposed Low-Rank Adaptation

Метод QDoRA (*Weight-Decomposed Low-Rank Adaptation*) [9] представляет собой расширение концепции Low-Rank Adaptation (LoRA) для повышения эффективности адаптации крупных моделей. В отличие от стандартного LoRA, QDoRA фокусируется на декомпозиции весов и параметров с использованием не только низкоранговых матриц, но и так называемой магнитуды $m = ||W + BA||_c$. Здесь W уже квантованная.

Основная идея QDoRA Пусть $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ — веса слоя модели. Вместо прямой адаптации весов вводится их декомпозиция на две компоненты при каждом шаге обучения. Сначала веса также квантуются в $\mathbf{W}'$, как и в QLoRA, а потом добавляется магнитуда:

$$\mathbf{W}_{new} = m \frac{\mathbf{W}' + \mathbf{BA}}{||\mathbf{W}' + \mathbf{BA}||_c}$$

То есть при инициализации модуля считается по-столбцовая норма соответствущих весов, а при каждом шаге обучения она дополнительно нормируется на значение, написанное в формуле. После обучения магнитуда пересчитывается.

Практическая реализация Для реализации метода QDoRA использовались следующие вещи:

1. **Квантование исходных весов**: Проводилось квантильное квантование, описанное выше в данном отчёте.
2. **Реализация модулей для линейного и свёрточного слоёв**. Аналогично QLoRA, были реализованы классы-обёртки для DoRA.
3. Реализация аналогична QLoRA за исключением того, что используется дополнительный параметр магнитуды, которая нормирует наши веса.

Эксперименты Для проверки эффективности метода QDoRA проведён эксперимент с предобученной моделью TimesNet на задаче классификации временных рядов. Были исследованы разные значения ранга r.

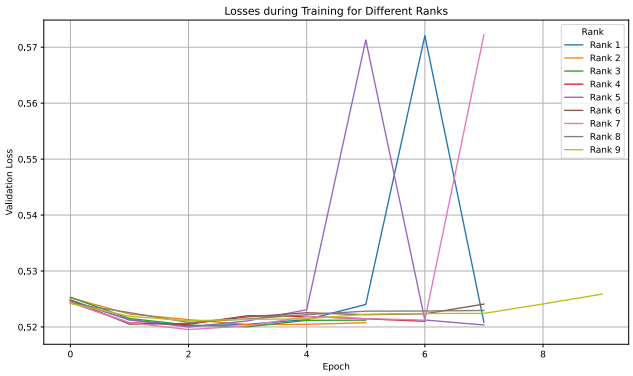

Рис. 11: *Валидационные потери при обучении QDoRA для различных значений ранга r.*

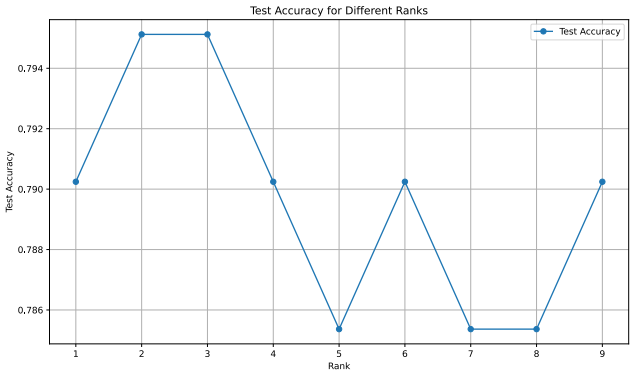

Рис. 12: *Точность на тестовой выборке после обучения QDoRA для различных значений ранга r.*

Как видно из графиков 11 и 12, QDoRA показывает более высокие значения accuracy, нежели QLoRA, однако, так как задание предполагало дообучение на той же выборке данных, на которой проводилось обучение самой модели, то и результатом стало сильное переобучение.

Приведём также график времени обучения модели для разного значения рангов 13.

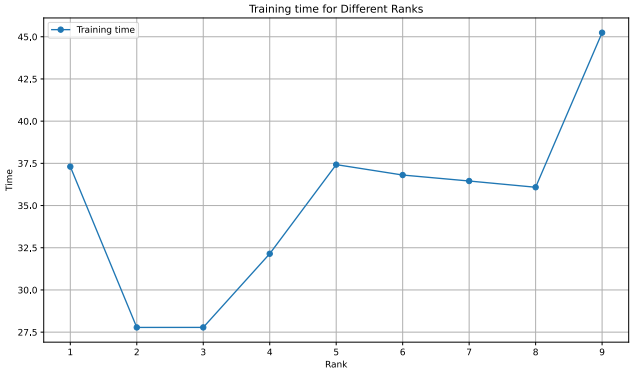

Рис. 13: *Время обучения QDoRA для разного значения рангов r*

Дообучение занимает гораздо меньше времени, чем

если бы мы дообучали модель со всеми весами без использования QDoRA.

Выводы по QLoRA и QDoRA Несмотря на сильное переобучение в экспериментах, методы показывают себя неплохо с точки зрения относительного изменения метрики accuracy, а также времени обучения. Реализация данных методов над предобученными моделями позволяет производить более быстрое дообучение с меньшими накладными расходами, что позволяет сделать вывод о значимости данных методов. Также стоит сказать, что в силу переобучения не удалось найти минимальное значение ранга, при котором точность превышает предобученную модель, сквантизованную в 8 бит, так как точность при дообучении падает.

Выводы и сравнение методов

Сложно сравнивать полученные методы сжатия, т.к. они направлены на разные цели:

Цель дистилляции — сделать обучение другой модели более стабильным

Цель спарсификации — уменьшить число параметров.

Цель квантизации — уменьшить вес модели (в битах)

Цель PEFT — сделать finetune модели более эффективным.

С точки зрения соотношения перфоманса и числа бит квантизация сильно лучше, чем спарсификация. Наверное, лучше всего использовать их совместно: вначале спарсификацию, а потом квантизацию.

Вклад участников

- Ахтырченко Михаил: Дистилляция (и импортирование моделей)
- Брескану Никита: Спарсификация (и импортирование моделей)
- Сильвестров Егор: Квантизация
- Каратыщев Дмитрий: PEFT

Список литературы

[1] Haixu Wu и др. «Timesnet: Temporal 2d-variation modeling for general time series analysis». В: *arXiv preprint arXiv:2210.02186* (2022).

[2] A Vaswani. «Attention is all you need». В: *Advances in Neural Information Processing Systems* (2017).

[3] Yong Liu и др. «itransformer: Inverted transformers are effective for time series forecasting». В: *arXiv preprint arXiv:2310.06625* (2023).

[4] Sunil Vadera и Salem Ameen. «Methods for pruning deep neural networks». В: *IEEE Access* 10 (2022), с. 63280—63300.

[5] Geoffrey Hinton. «Distilling the Knowledge in a Neural Network». В: *arXiv preprint arXiv:1503.02531* (2015).

[6] Zhiwei Hao и др. «One-for-all: Bridge the gap between heterogeneous architectures in knowledge distillation». В: *Advances in Neural Information Processing Systems* 36 (2024).

[7] Tim Dettmers и др. *QLoRA: Efficient Finetuning of Quantized LLMs*. 2023. arXiv: 2305.14314 [cs.LG]. URL: https://arxiv.org/abs/2305.14314.

[8] Edward J. Hu и др. «LoRA: Low-Rank Adaptation of Large Language Models». В: *CoRR* abs/2106.09685 (2021). arXiv: 2106.09685. URL: https://arxiv.org/abs/2106.09685.

[9] Shih-Yang Liu и др. *DoRA: Weight-Decomposed Low-Rank Adaptation*. 2024. arXiv: 2402.09353 [cs.CL]. URL: https://arxiv.org/abs/2402.09353.