Научная конференция «Суперкомпьютерные дни в России», 30 сентября 2025 г

An Evolutionary-Based Approach for Hardware-Aware Optimization of BERT-Like Models

Основанный на эволюционных алгоритмах подход к оптимизации BERT-подобных моделей с учетом особенностей целевой платформы

Вахрушев В.Ю., аспирант, Попова Н.Н.,доцент,
кафедра СКИ ВМК МГУ

Проблема эффективного сжатия трансформерных моделей

- Трансформерные модели используются практически для любых задач обработки текста - классификации, генерации, построения векторного представления слов и текстов
- Современные модели имеют сотни миллионов и даже миллиарды параметров
- Для их эффективного использования на маломощных платформах применяются несколько стандартных классов методов сжатия модели
- Нет стандартного подхода, который позволил бы эффективно комбинировать разные классы методов сжатия в едином алгоритме
- Современные методы сжатия редко используют в явном виде информацию о целевой платформе, на которой предполагается применять модель после сжатия

Используемые термины

- BERT - трансформерная архитектура, используемая для задач классификации, регрессии и векторизации текстов
- Дистилляция - метод сжатия модели, при котором модель относительно небольшого размера обучается с использованием результатов большой модели-учителя
- Квантизация - метод сжатия модели, при котором изменяется тип данных её весов (и, возможно, промежуточных результатов) - например, с fp32 на int8
- Обрезка - метод сжатия модели, при котором из нее удаляются наименее значимые слои или веса
- Инференс - процесс практического применения модели после обучения
- NAS - neural architecture search, метод поиска оптимальной конфигурации нейронной сети методом умного перебора

Цель работы

- **Цель:** реализация масштабируемого метода сжатия трансформерных BERT-подобных моделей, который был бы способен эффективно комбинировать методы обрезки, дистилляции и квантизации и при этом мог бы учитывать время инференса модели на целевой платформе
- **Задача:** разработка генетического алгоритма, использующего в качестве базовых операции методы сжатия моделей с учетом целевой платформы; исследование влияния значений гиперпараметров алгоритма на его эффективность.
- **Мотивация:** Наличие подобного метода, представляющего собой метаалгоритм, позволит эффективно сочетать лучшие практики из всех трех основных типов методов сжатия моделей, а целевая функция позволит явным образом оптимизировать сжимаемую модель под конкретную целевую платформу.

Существующие решения

- Методы дистилляции, квантизации и обрезки моделей сегодня активно развиваются
- Одним из самых популярных и универсальных решений для квантизации является библиотека **bitsandbytes**
- Существует множество дистиллированных предобученных BERT-моделей, основанных на идее сближения выходных распределений моделей ученика и учителя, изначально предложенной при обучении **DistilBERT**
- При обрезке трансформерных моделей сейчас используются в основном методы структурированной обрезки слоев внимания, реализованные, например, в библиотеке **TorchPruner**
- Также для сжатия трансформерных моделей используются **NAS**-алгоритмы (библиотека **Syne-Tune**)

Квантизация

- Квантизация - изменения типов весов (и, возможно, активаций) у модели.
- Для BERT-подобных моделей обычно хватает int8 квантизации
- Квантизация бывает во время обучения (quantization-aware traninig) и после (post-training)
- Будем рассматривать post-training квантизацию

Дистилляция

- Дистилляция - процесс обучения относительно небольшой модели-ученика с использованием значений выходного слоя модели-учителя (и иногда с использованием значений ее скрытых слоев)
- Для BERT-подобных моделей впервые была применена при обучении модели **DistilBERT**
- Сейчас существует множество модификаций **DistilBERT**, например, **TinyBERT**

DistilBERT

- Пусть L_{model} - функция потерь при обучении исходной BERT-модели. При обучении **DistilBERT** к нему добавляется слагаемое L_{dist}; $L_{dist} = L_{hidn} + L_{pred}$
- **DistilBERT** обучается на задачу предсказания маскированного токена
- $L_{hidn} = 1 - cos(H^S, H^T)$, где $cos(H^S, H^T)$ - косинусное расстояние между эмбеддингами последнего скрытого слоя модели учителя (H^T) и ученика (H^S)
- $L_{pred} = KL(z^T / t, z^S / t)$ - дивергенция Кульбака-Лейблера между выходными распределениями модели учителя (z^T) и ученика (z^S), поделенными на параметр температуры (t).

Обрезка

- Обрезка (**prunning**) представляет собой удаление наименее значимых параметров модели после или во время ее обучения
- Можно обрезать как отдельные веса модели (неструктурированная обрезка), так и нейроны, головы внимания (трансформеры) или даже целые слои (структурированная обрезка)
- То, какие параметры наименее значимы, определяется на основе критерия важности (чаще всего используется норма весов)
- Обрезать модель можно во время обучения или после него

Предложенный метод

Дано:

- Обученная базовая BERT-подобная модель
- Тренировочный и тестовый наборы данных для целевой задачи
- Целевая архитектура GPU, на которой предполагается применять сжатую модель

Необходимо построить модель, которая обладает наибольшим качеством на целевой задаче при наименьшем объеме занимаемой памяти и наибольшей скорости инференса на целевой платформе.

В предлагаемом подходе процесс сжатия модели представляет собой генетический алгоритм, популяцией которого являются варианты сжатой модели, а функция качества включает в себя все три требуемых функционала качества, которым можно явным образом задавать веса

Операторы в предложенном методе

- Мутация - структурная обрезка модели с последующей эпохой дообучения; также квантизация
- Скрещивание - эпоха дообучения модели с использованием исходной базовой модели
- Отбор происходит по значению функции качества

Мутация

- В предлагаемом подходе используется структурная обрезка K_1 последних слоев внимания модели или K_2 голов внимания на последнем слое
- Делаем обрезку последних слоев, поскольку если отрезать первые, дообучить модель станет гораздо тяжелее
- В качестве библиотеки для реализации обрезки используется **TorchPruner**
- После обрезки модель может быть квантизована (библиотека **bitsandbytes**) в один из трех типов данных (**int4, int8, fp16**) - однако также сохраняется копия весов модели в **fp32** для последующих операций дообучения

Скрещивание

- Скрещивание производится между моделью-кандидатом и базовой моделью с использованием подхода, похожего для обучения **DistilBERT**
- $L_{dist} = L_{hidn} + L_{pred}$
- $L_{hidn} = 1 - cos(H^S , H^T)$
- $L_{pred} = KL(z^T / t, z^S / t)$, однако здесь z - выходное вероятностное распределение для **целевой** задачи (если мы имеем дело с задачей классификации)
- В остальных случаях $L_{dist} = L_{hidn}$

Функция качества

F(network) = (Q(network) / Qmax)^{a} * (Tmin / T(network))^{b} * (Mmin / M(network))^{c} где:

- ***Q(network)*** - значение функции качества нейросети на валидационной выборке; ***Qmax*** - максимальное по текущей популяции нейросетей значение функционала;
- ***T(network)*** - время инференса нейронной сети на валидационной выборке на целевой платформе; ***Tmin*** - минимальное по текущей популяции нейросетей значение времени инференса
- ***M(network)*** - объем памяти, занимаемой нейросетью; ***Mmin*** - минимальное по текущей популяции нейросетей значение занимаемой памяти
- ***a, b, c*** - коэффициенты

На данный момент для вычисления ***T(network)*** необходимо напрямую инференсить модель-кандидат на целевой платформе

Общий вид алгоритма

1. Вычислить ***F(network)*** исходной модели; создать исходную популяцию, скопировав модель ***N*** раз
2. Для каждой модели из текущей популяции провести с заданными вероятностями операции мутации и скрещивания; вычислить функционал качества полученной модели; добавить полученную модель в популяцию
3. Провести отбор по функционалу качества из популяции, оставив *K* моделей с наибольшим значением функционала
4. Вернуться на шаг 2, если не достигнуто максимальное число эпох или требуемое качество модель

Реализация методов

Предложенные методы были разработаны с использованием технологии Python 3.11 в рамках программной библиотеки. Функционал библиотеки позволяет пользователю через интерфейс задать гиперпараметры метода, собственные функции обрезки и квантизации, отличные от заданных по умолчанию, а также поддерживает использование нескольких GPU (популяция моделей будет распределена по GPU равномерно)

Эксперименты

В качестве наборов данных для тестирования метода были взяты наборы данных CoLA, SST-2, и RTE из бенчмарка GLUE.

В качестве целевой платформы для инференса была выбрана GPU A100.

Предложенный подход применялся к BERT-модели и сравнивался с известными дистиллированными моделями (**DistilBERT, TinyBert, MiniLM**) и методом **NAS** (библиотека **Syne-Tune**).

В качестве платформы для экспериментов использовался суперкомпьютер МГУ-270. Эксперимент проводился на одном узле(GPU - Nvidia A100, 80 GB).

Гиперпараметры предложенного алгоритма (Genetic) для финальной таблицы подбирались по результатам экспериментов и были равны ***a=5, b=1, c=1, popSize=100, nEpoch=100.***

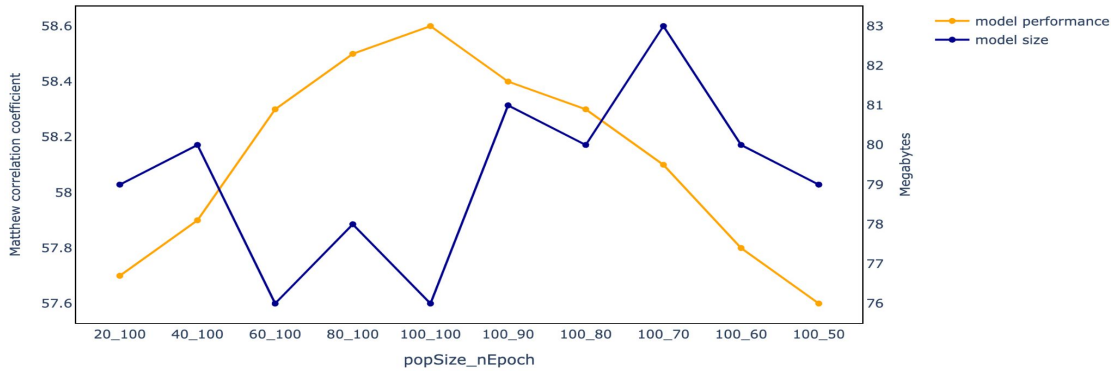
Экспериментальное исследование разработанных методов

Dataset	Method	Metric	Size, MB	Inference Time, minutes
CoLA	**Genetic**	58.7	76	1.16
	BERT	58.8	86	1.36
	DistilBERT	50.4	66	0.88
	TinyBert	48.8	20.7	1.13
	NAS	58.4	78	1.17
	MiniLM	53.7	33	1.06
SST-2	**Genetic**	90.8	83	1.27
	BERT	91.3	86	1.32
	DistilBERT	90.1	66	1.18
	TinyBert	86.2	20.7	0.98
	NAS	90.6	78	1.25
	MiniLM	89.9	33	1.05
QQP	**Genetic**	65.3	76	0.58
	BERT	65.2	86	0.6
	DistilBERT	62.8	66	0.56
	TinyBert	57.7	20.7	0.48
	NAS	65.2	78	0.57
	MiniLM	63.2	33	0.52

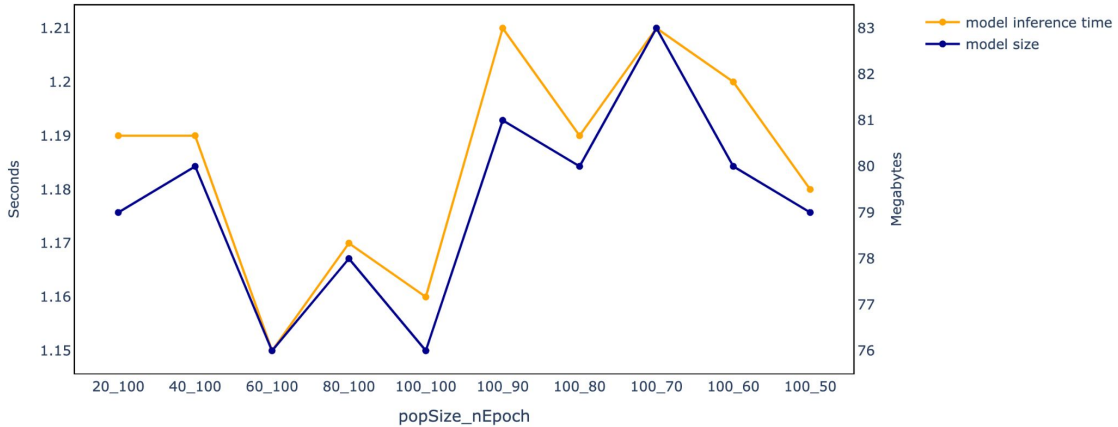
Сравнение качества методов

Исследование характеристик итоговой модели в зависимости от числа эпох и размера популяции

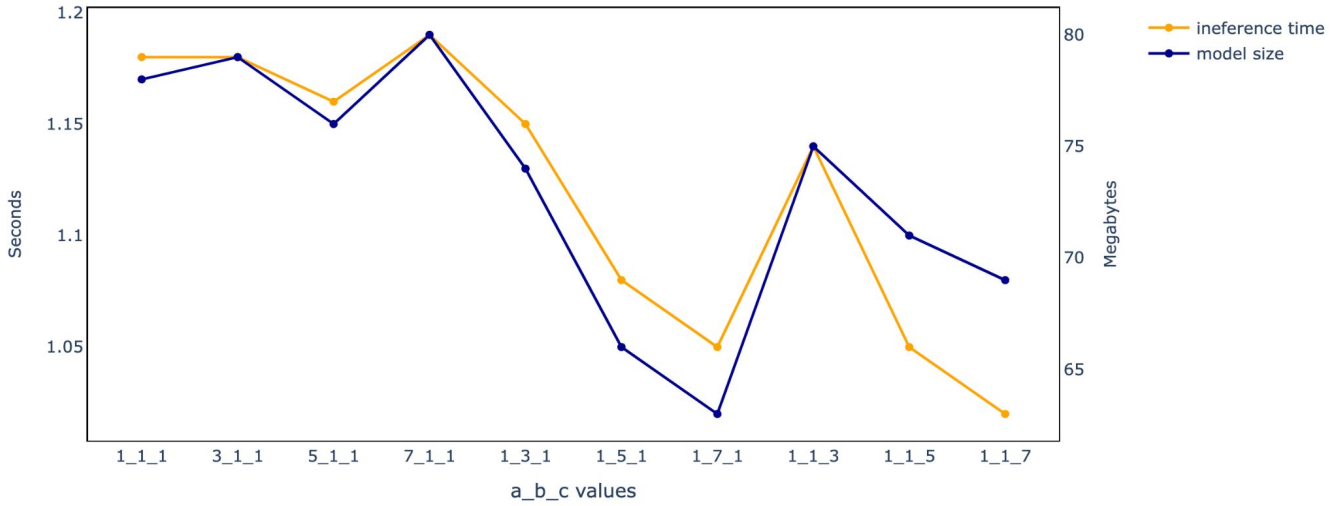
Зависимость качества и размера итоговой модели от размера популяции и числа эпох



Зависимость размера итоговой модели и времени ее инференса от размера популяции и числа эпох



Экспериментальное исследование разработанных методов



Зависимость размера итоговой модели и времени инференса от коэффициентов a, b, c

Выводы

- Для большинства целевых задач можно сжать модель практически без потери качества на целевой задаче

- Несмотря на то, что скорость инференса и размер модели скоррелированы, при непосредственной оптимизации одной из этих характеристик ее можно оптимизировать лучше, чем при непосредственной оптимизации второй

Заключение

Основные результаты работы:

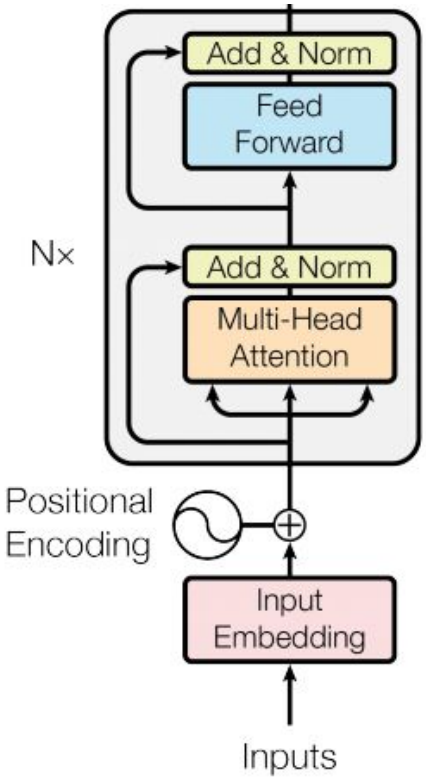
- Разработаны и исследованы методы сжатия моделей с использованием генетических алгоритмов с использованием современных методов компрессии
- Проведенные эксперименты показали эффективность предложенных методов в сравнении с дистиллированными моделями и методом NAS.
- Предложенные методы позволяют явным образом приоритезировать характеристики сжатой модели
- Реализованная библиотека является инструментом с открытым исходным кодом и позволяет эффективно масштабироваться на несколько GPU

СПАСИБО ЗА ВНИМАНИЕ!

Генетические алгоритмы

- Генетический алгоритм - эвристический метод приближенной оптимизации, основанный на понятиях популяции, естественного отбора и функции качества
- Популяция - набор объектов ***O***, обладающих определенными параметрами ***P***.
- Функция качества ***Q(O)*** оценивает качество объекта в рамках задачи
- Мутация - случайное преобразование параметров объекта P^{O} -> P'^{O}
- Скрещивание - порождение нового объекта с параметрами, полученными определенной комбинацией параметров двух объектов-родителей P^{O1} + P^{O2} -> P^{O3}
- Отбор - удаление из популяции объектов с самым низким значением функции качества
- Обычно этапы Скрещивание -> Мутация -> Отбор повторяются циклически до выполнения условия остановки

Структура слоя внимания



Планы на будущее

- Попробовать иные способы метаоптимизации - например, обучение с подкреплением
- Попробовать предложенные методы для LLM
- Масштабировать методы на несколько вычислительных узлов
- Разработать метод оценки времени выполнения модели на заданной архитектуре даже при ее отсутствии под рукой