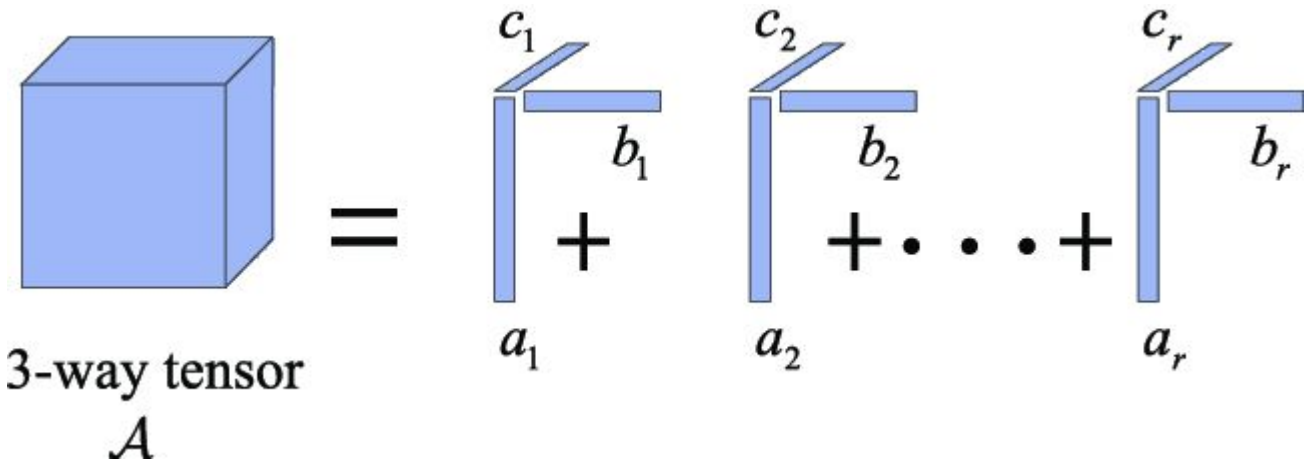
Compressing CNNs with CP decomposition

Никита Загайнов, Павел Кривенко, Хусейин Агаджанов



Релевантные работы

https://arxiv.org/abs/1701.07148	База нашего проекта. Мы реализовывали данную статью в форме библиотеки	[1]
https://arxiv.org/abs/2112.10855	Алгоритм, используемый для вычисления CP разложения	[2]
https://arxiv.org/abs/2005.13746	Подробнее о том, как происходит свертка разложенным ядром	[3]
https://arxiv.org/abs/1412.6553v3	В работе приведена одна из первоначальных идей со сжатием моделей	[4]

Background: CP decomposition

$$\mathbf{X} = \mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C} + \mathbf{E}$$

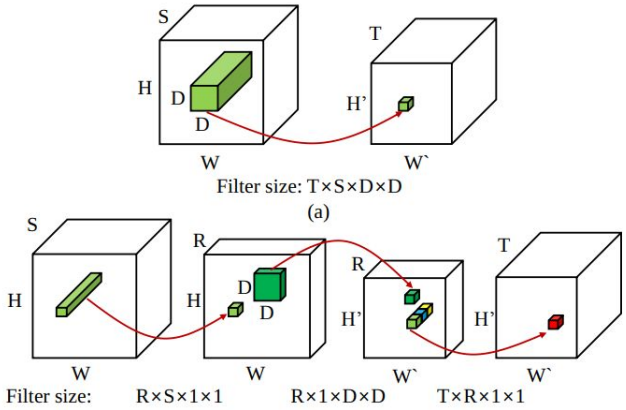
1. $\min_{\mathbf{A}} \|\mathbf{X} - \mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C}\|$
2. $\min_{\mathbf{B}} \|\mathbf{X} - \mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C}\|$
3. $\min_{\mathbf{C}} \|\mathbf{X} - \mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C}\|$

- Реальное название - CANDECOMP/PARAFAC or canonical polyadic (CP)
- Ранг тензора – NP - сложная задача
- Алгоритм - Alternating Least Squares (ALS) [2]
- Применяется в визуализации, PCA, **сжатии** данных, и в других целях
- Свертка с CP-разложенным ядром легко реализуема

Мотивация и цель проекта

В работе [1] приведен алгоритм для эффективного сжатия сверточных нейросетей с много значащими результатами, однако в работе приводятся результаты экспериментов только над моделью AlexNet, и не прилагается код для воспроизведения результатов.

Мы ставим целью проекта реализовать предложенный алгоритм в форме библиотеки на **PyTorch**, предназначенную для сжатия моделей CNN, а также воспроизведение с ее помощью результатов данной статьи и обобщение на дополнительных моделях и сценариях.



[1]

Алгоритм сжатия CNN

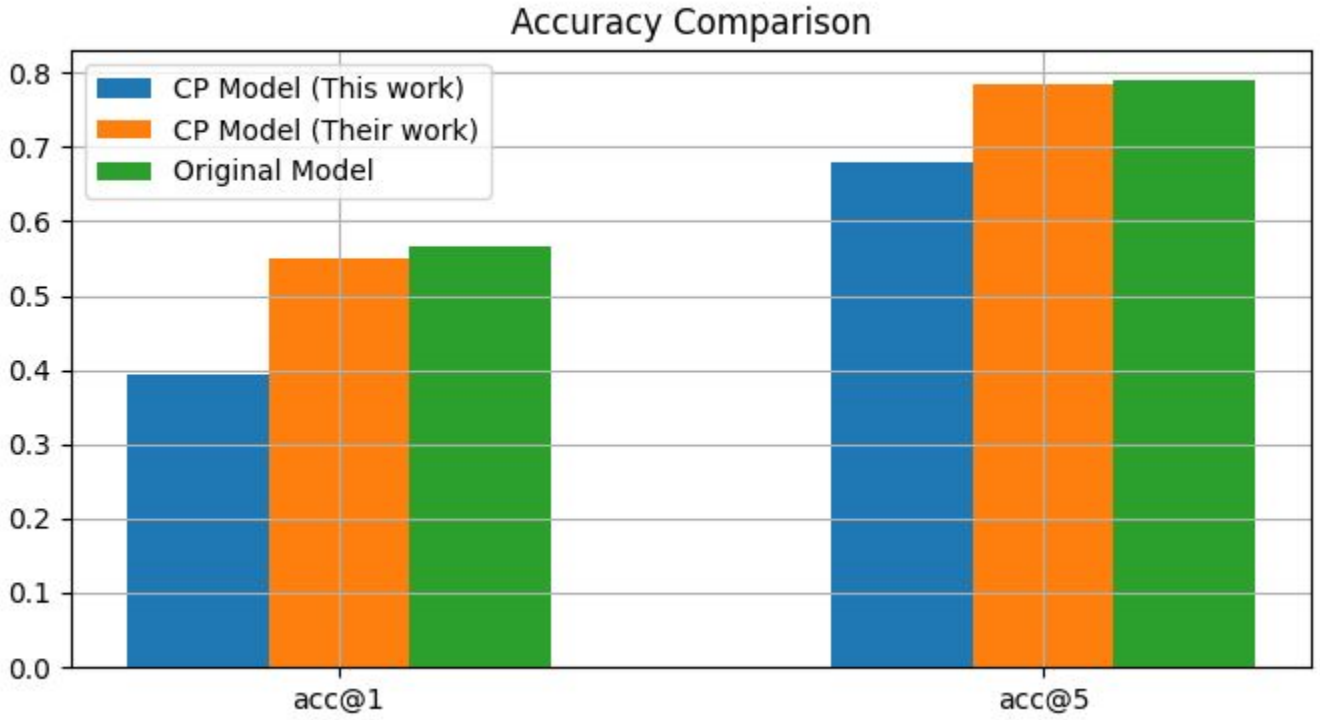
Мы используем метод, описанный в [1]

- Определяем ранг каждого слоя, измеряя их “чувствительность”: разложить этот слой с малым рангом и посмотреть на лосс
- После определения ранга, последовательно раскладываем каждый слой
- Важная деталь: чтобы не потерять точность, дообучаем модель после каждой операции разложения

Воспроизведение результатов [1]

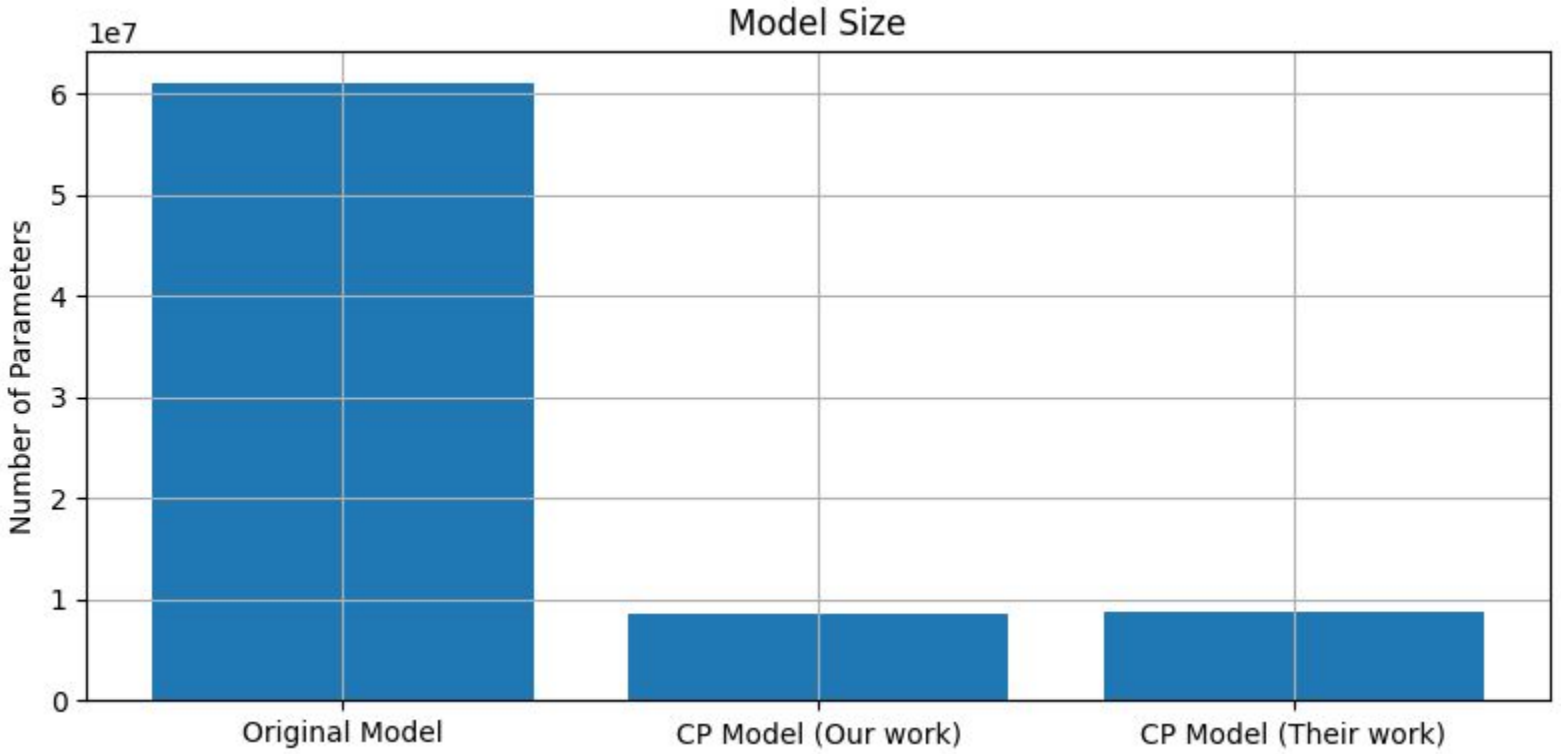
	Conv1	Conv2	Conv3	Conv4	Conv5	Conv Total	FC1	FC2	FC3	FC Total
Loss from [1]	5.38	11.89	11.86	13.68	15.17	**57.99**	28.69	21.50	20.31	**70.40**
Rank from [1]	69	154	153	178	196	**750**	365	275	260	**900**
Loss (ours)	8.34	7.46	8.66	7.16	6.66	**38.28**	6.68	6.77	5.51	**18.96**
Rank (ours)	164	147	169	140	130	**750**	318	321	261	**900**
Layer size (1e3)	23	307	664	885	590	**2468**	37749	16777	4096	**58622**

Воспроизведение результатов [1]



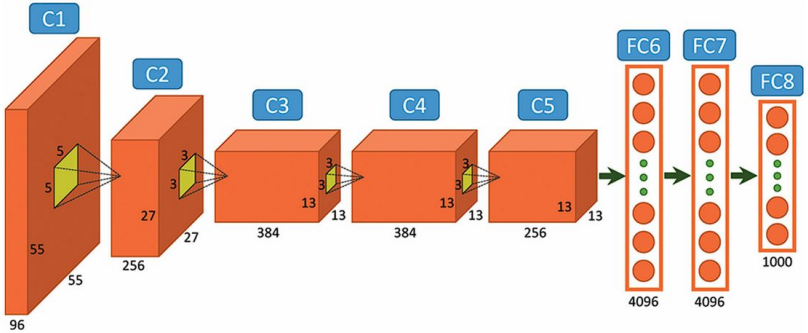
Сравнение с [1]

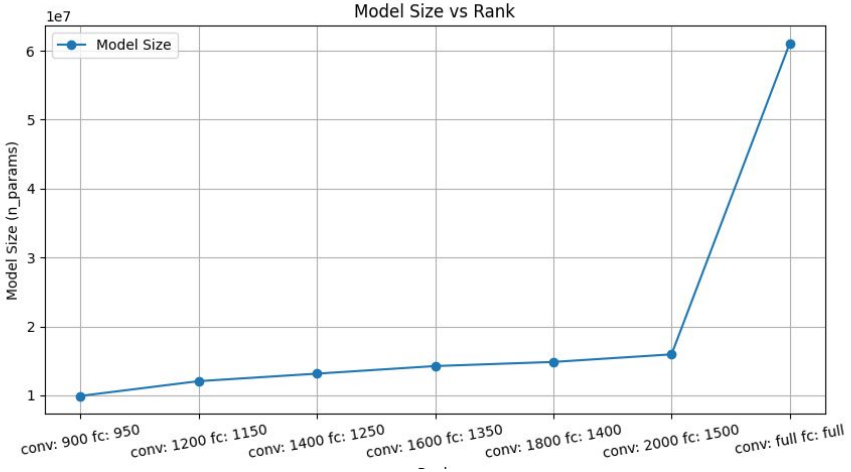
Воспроизведение результатов [1]

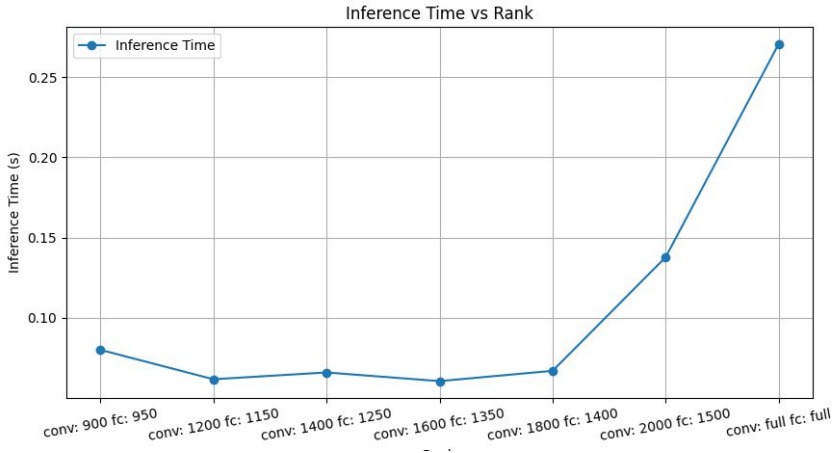


Сравнение с [1]

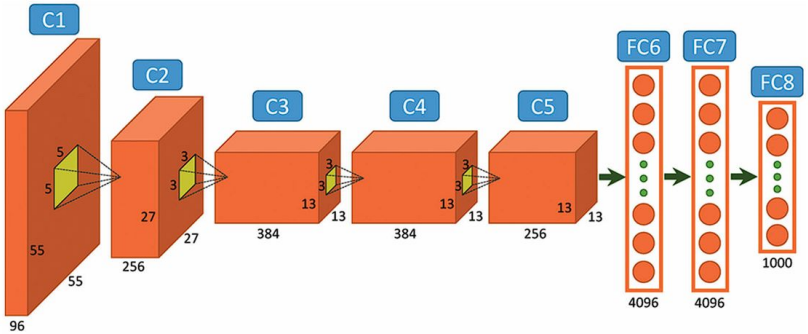
Эксперименты: AlexNet

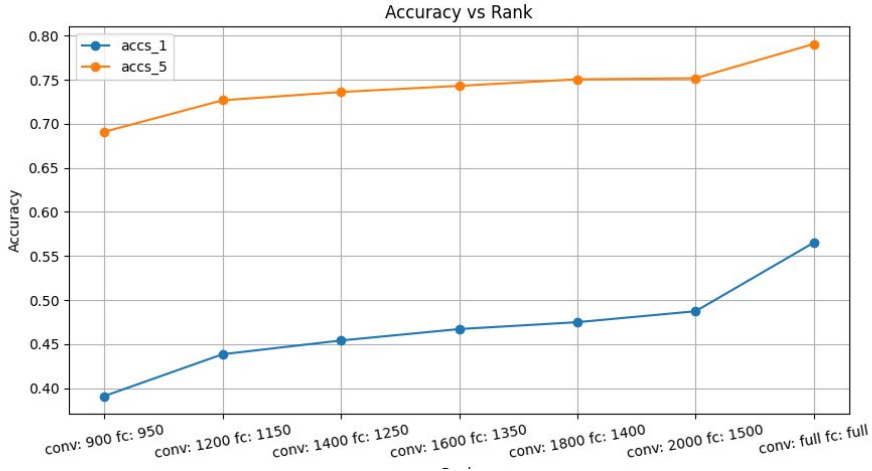


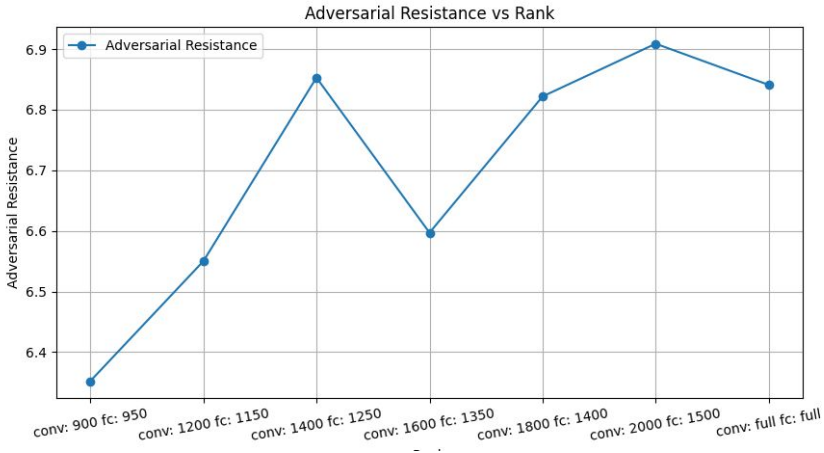




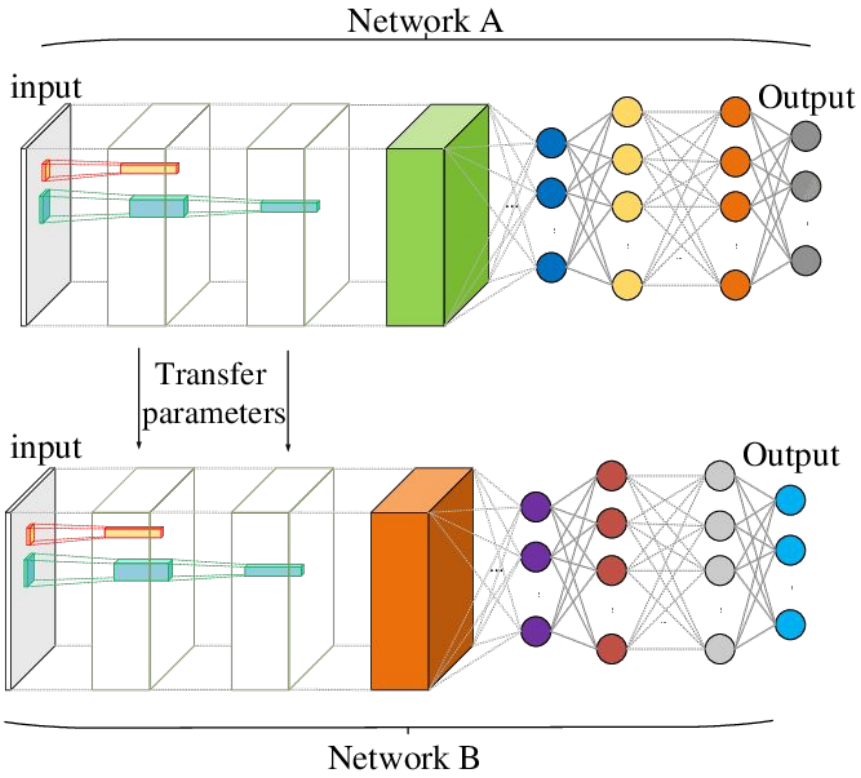
Эксперименты: AlexNet



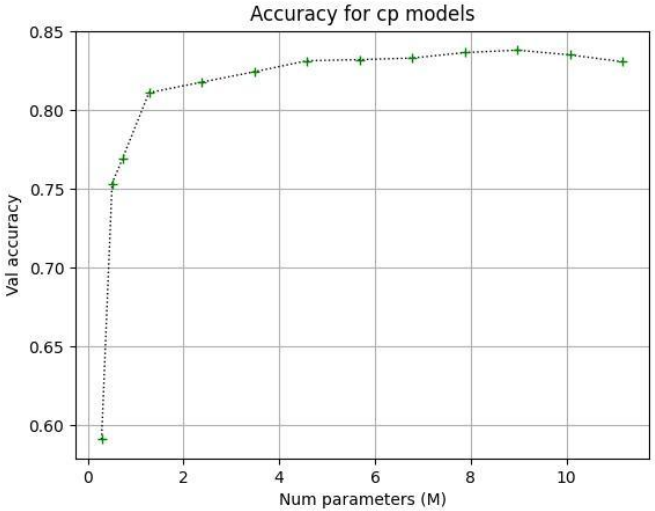


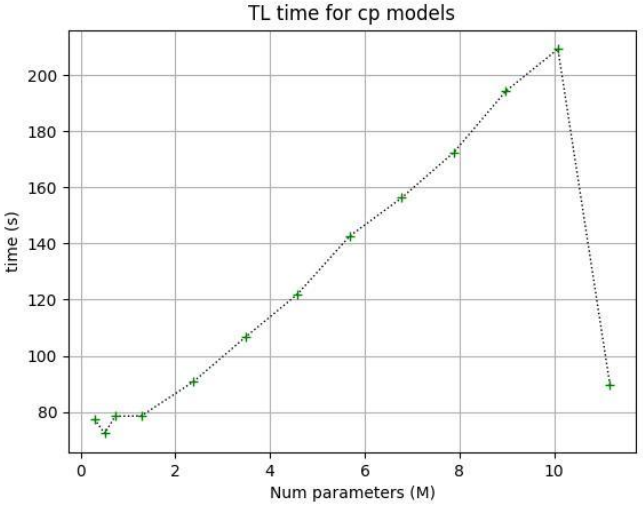


Transfer Learning

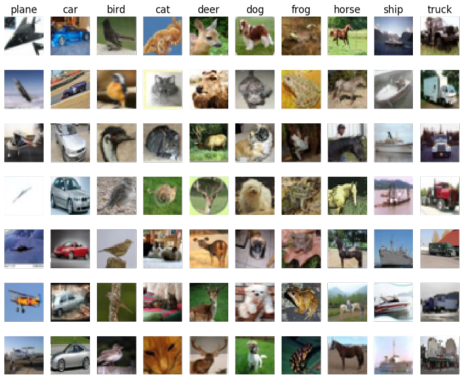


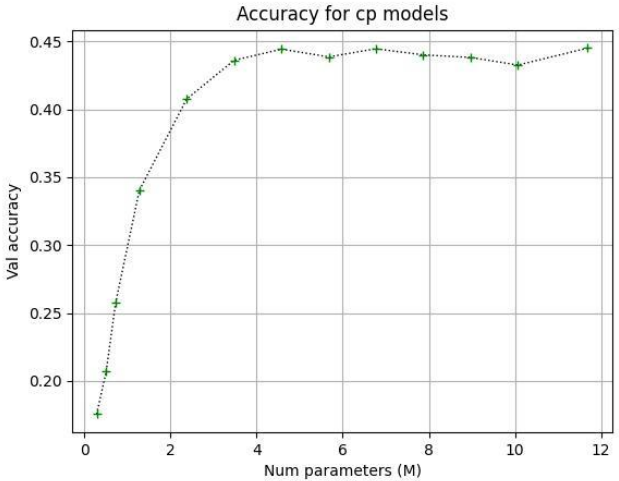
Transfer Learning. Fashion MNIST

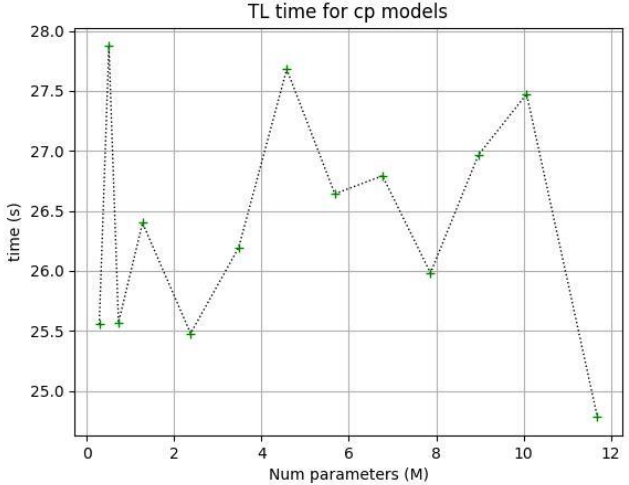




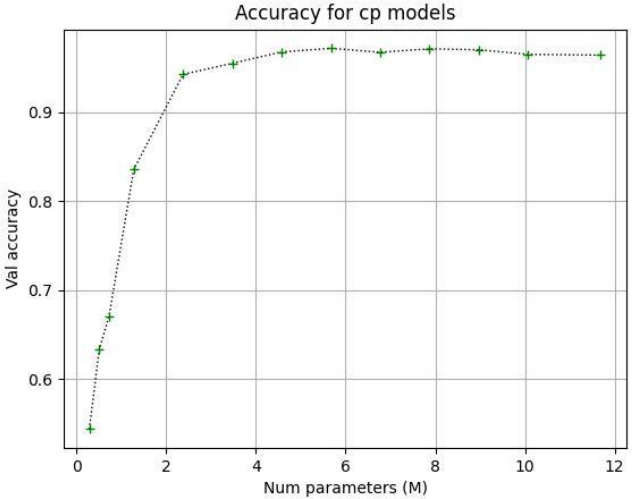
Transfer Learning. CIFAR10

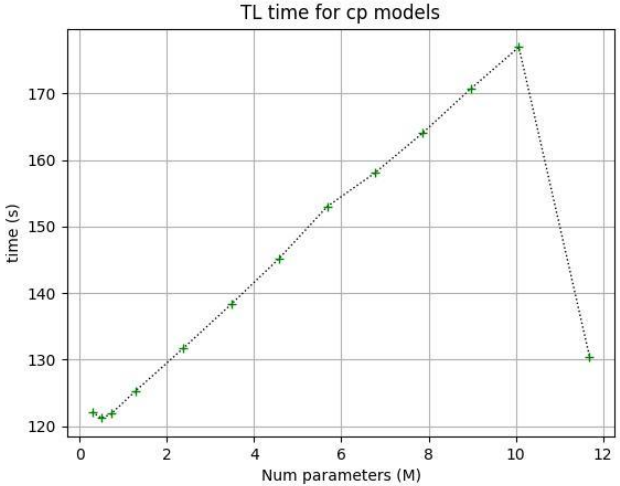






Transfer Learning. Cats vs Dogs





Результаты и Выводы

- Этим проектом мы представляем реализацию алгоритма из [1] и частичное воспроизведение их результатов
- CP разложение позволяет сильно уменьшить размер и увеличить скорость модели без больших потерь по качеству
- CP разложенные модели можно использовать в Transfer Learning

Спасибо за внимание!

GitHub: https://github.com/SqVaD-NLA2024/conv-cp-decomposition