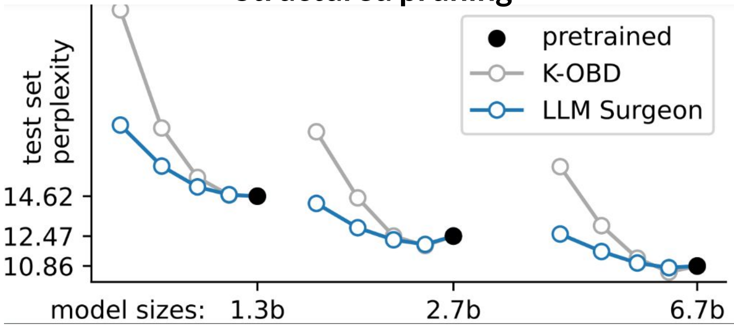
The LLM Surgeon

A general framework for pruning large neural models

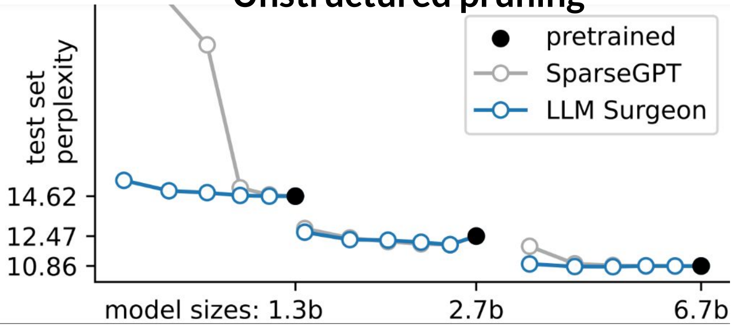
Tycho F.A. van der Ouderaa, Markus Nagel, Mart van Baalen, Yuri M. Asano, Tijmen Blankevoort

In ICLR 2024

Structured pruning

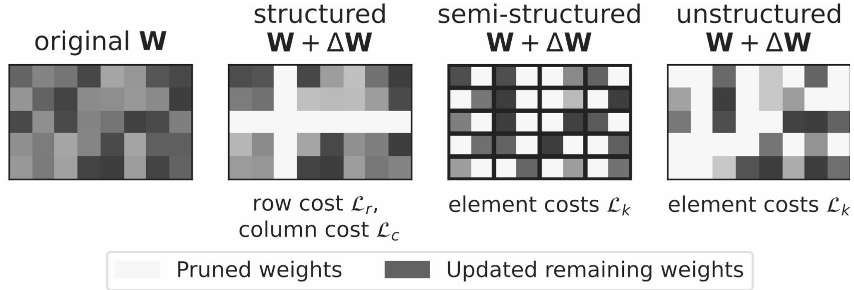


Unstructured pruning



Pruning of large neural models

Any structure type



Novelty

- Uses gradient info. Removal cost and updates in terms of final loss.
- Modern Hessian approximations
- Scalable to LLMs

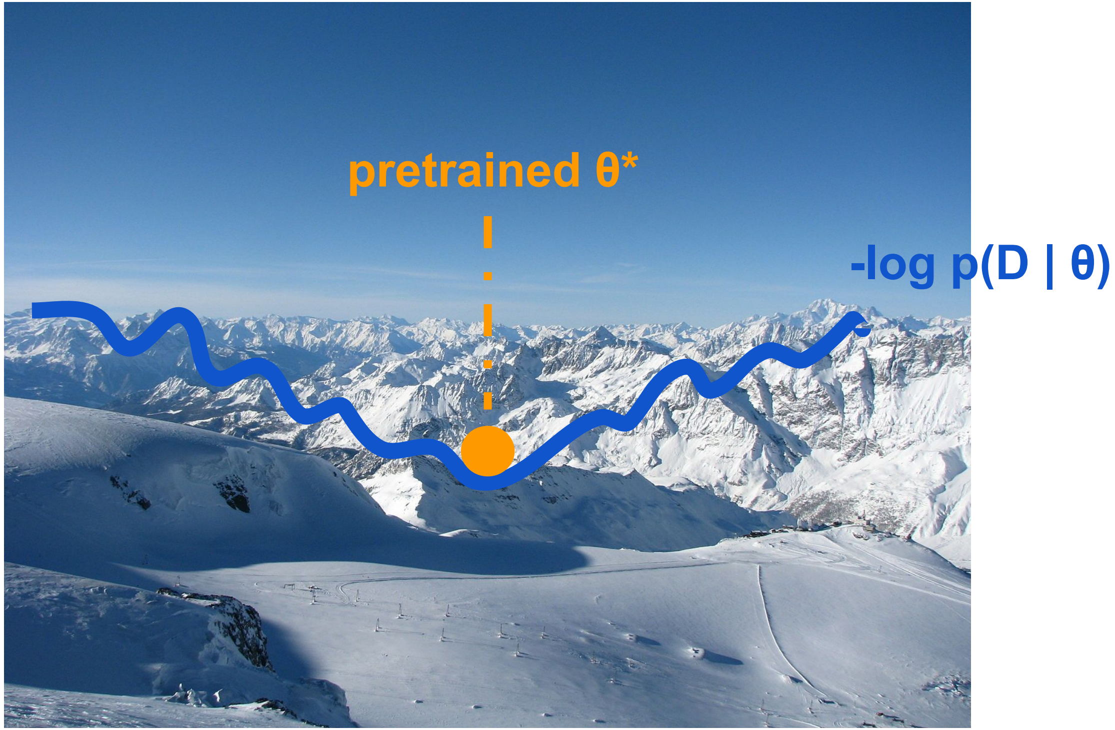
- First to achieve 20-30% structured (!) LLM pruning with performance loss.
- Also state-of-the-art results in unstructured and semi-structured pruning.

Growing interest in structured LLM pruning. Notably, concurrent work by ETH Zurich / Microsoft Research AI:
Ashkboos, Saleh, et al. "Slicegpt: Compress large language models by deleting rows and columns." *(2024)*

A tale of pruning in the loss surface

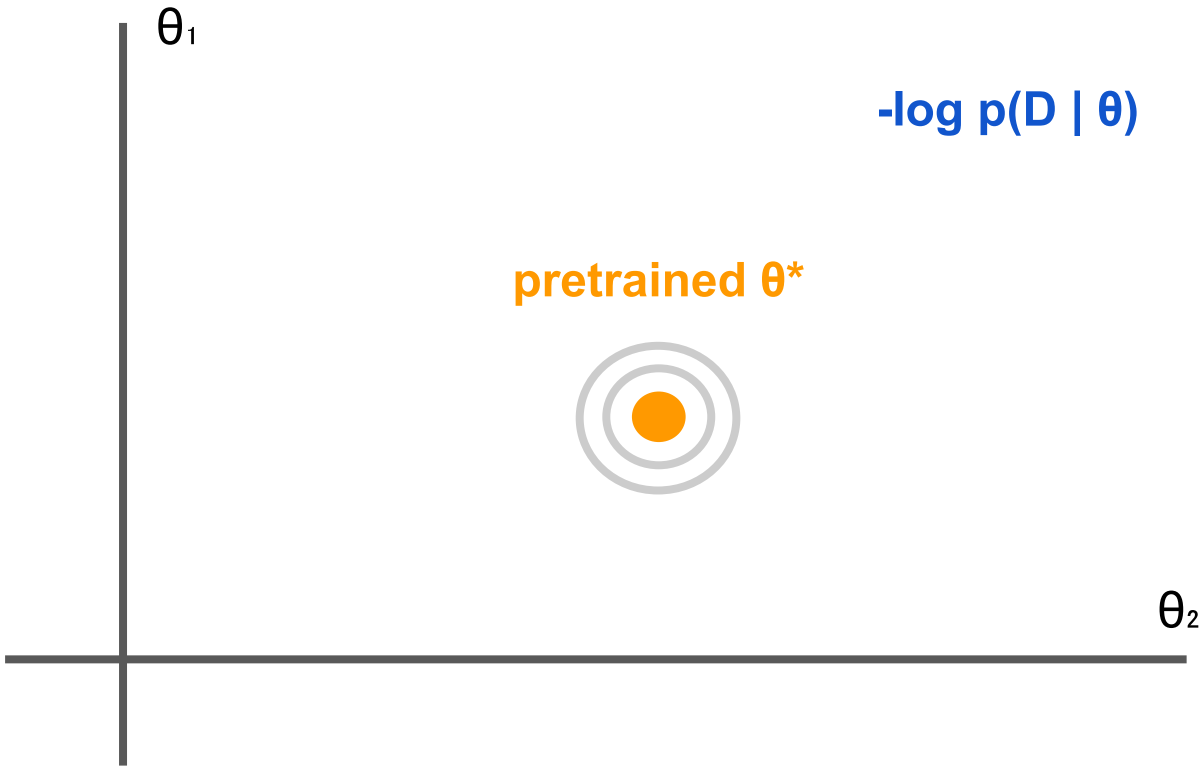
Alps view from Matterhorn Glacier Paradise. (source: Wikipedia)

A tale of pruning in the loss surface

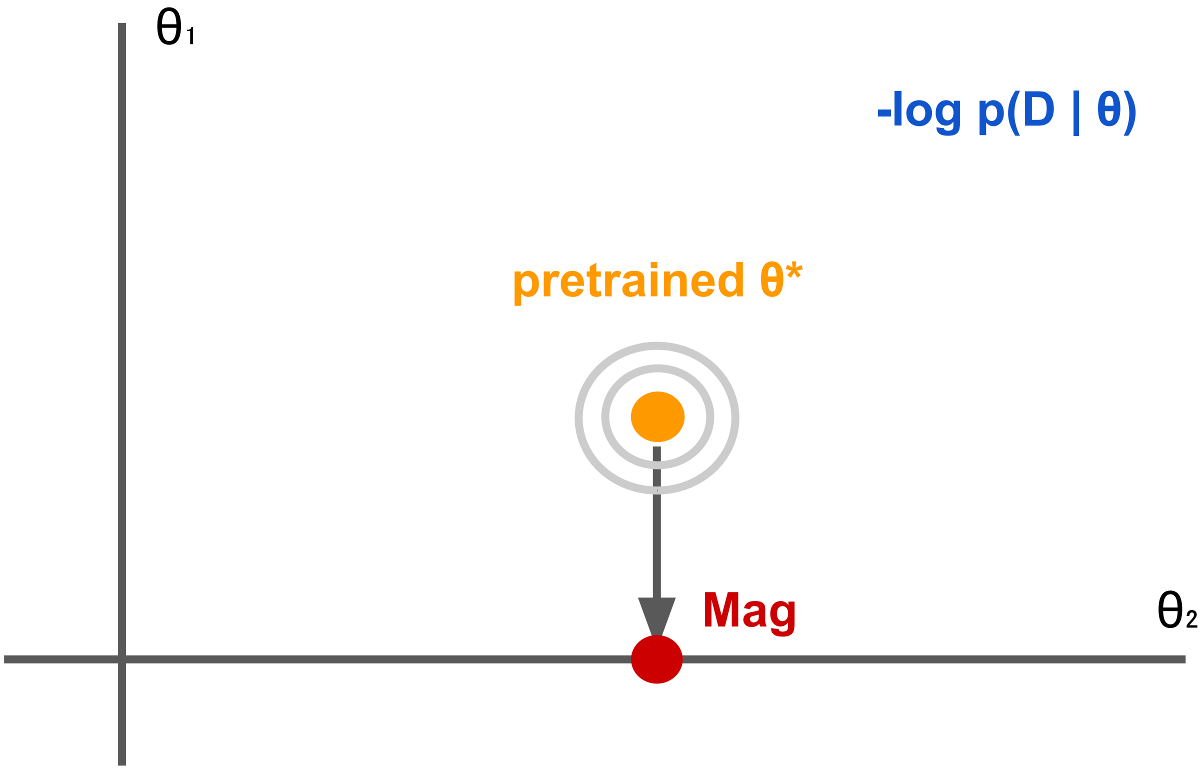


Alps view from Matterhorn Glacier Paradise. (source: Wikipedia)

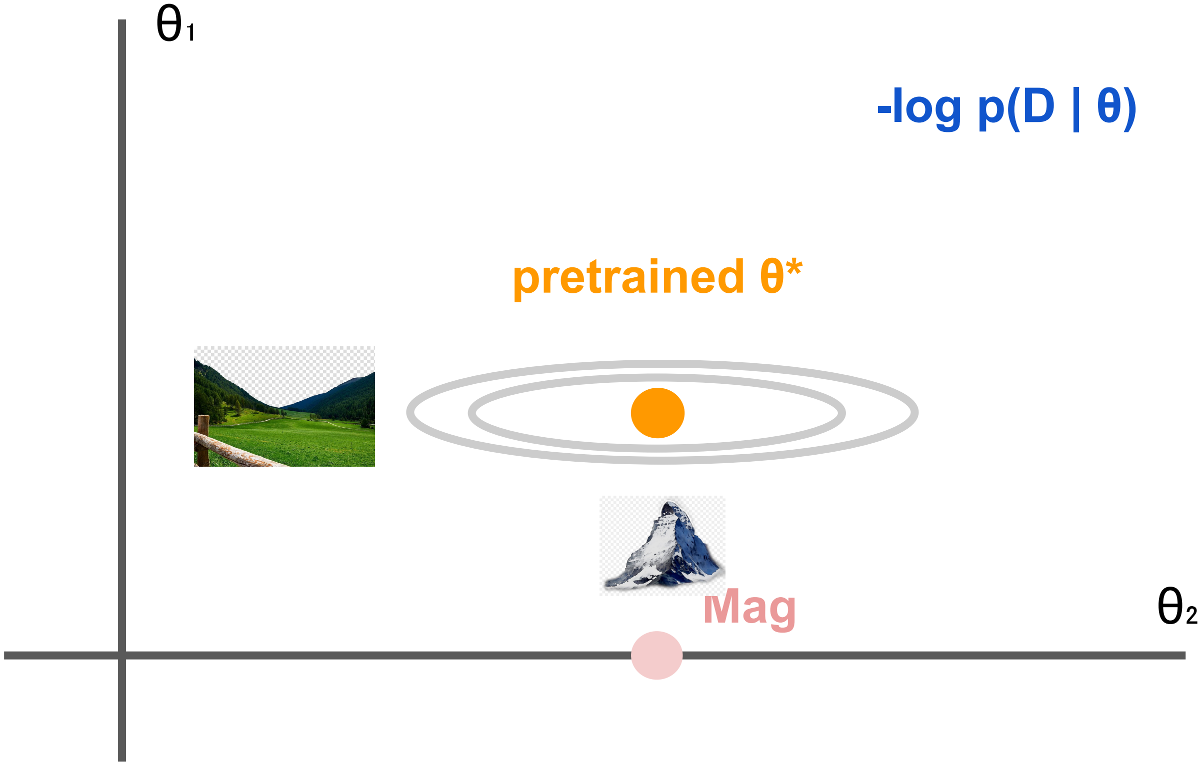
Geometric view on pruning



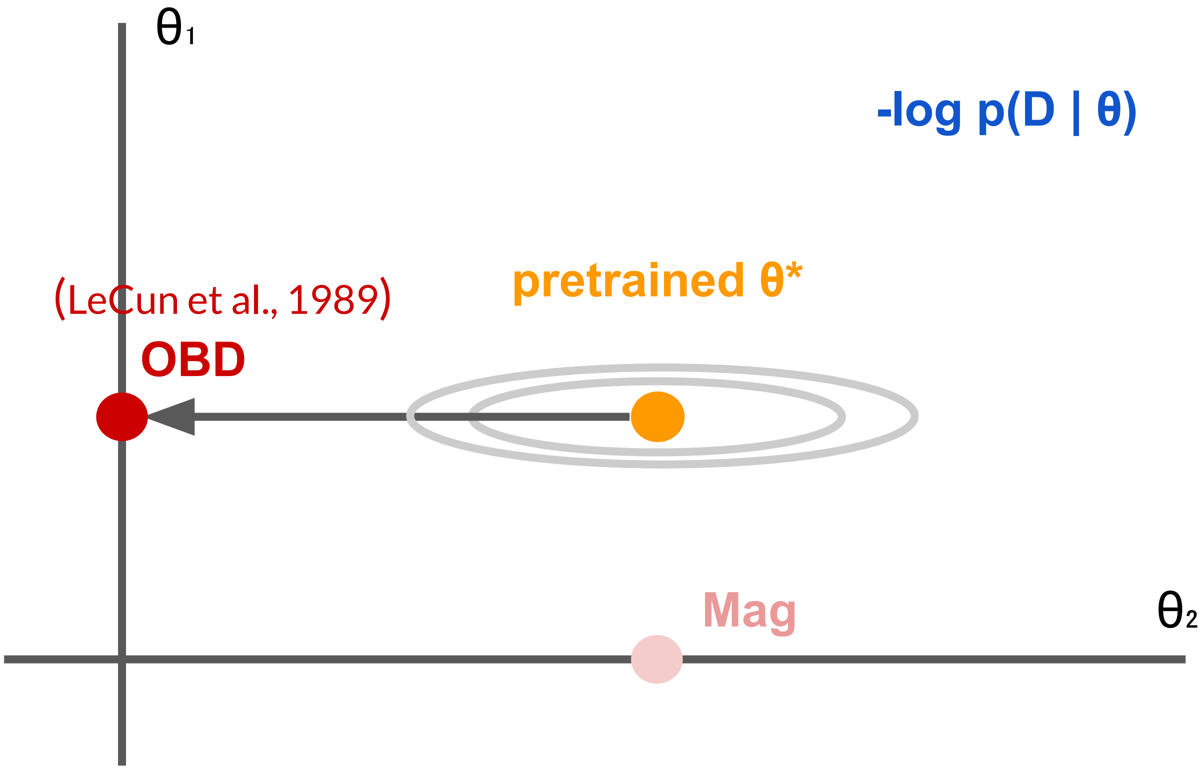
Geometric view on pruning



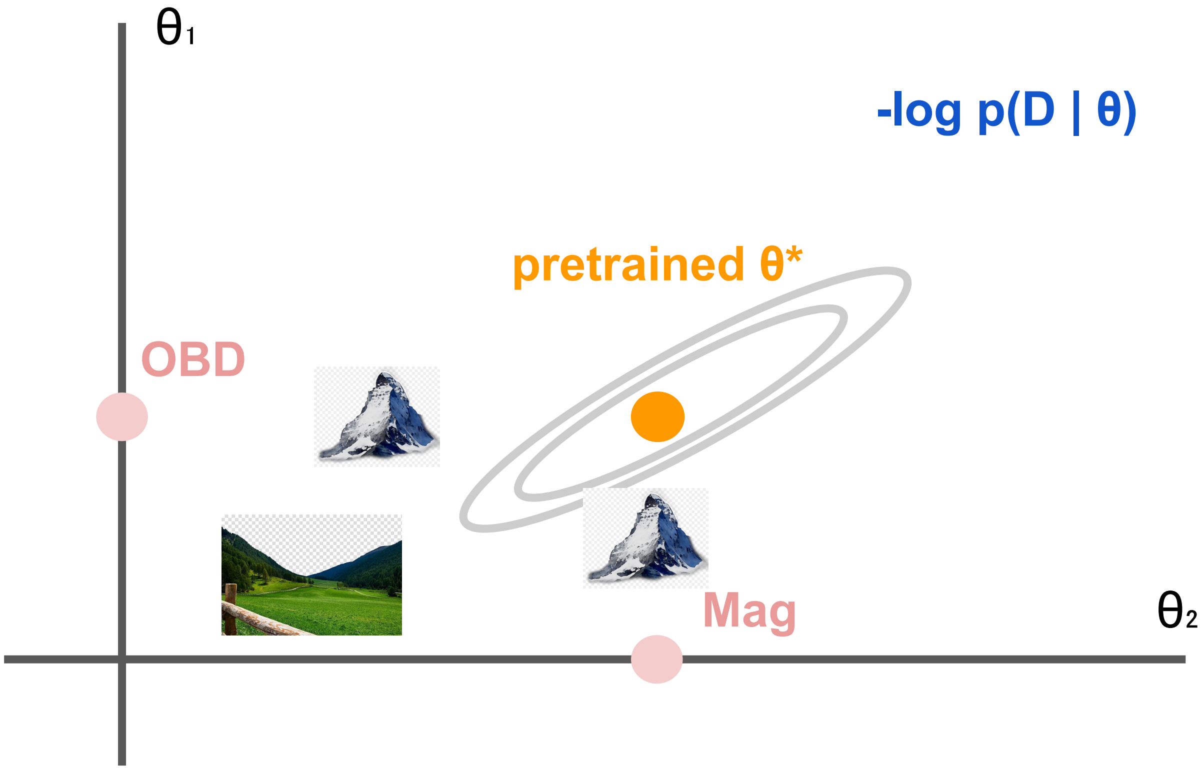
Geometric view on pruning



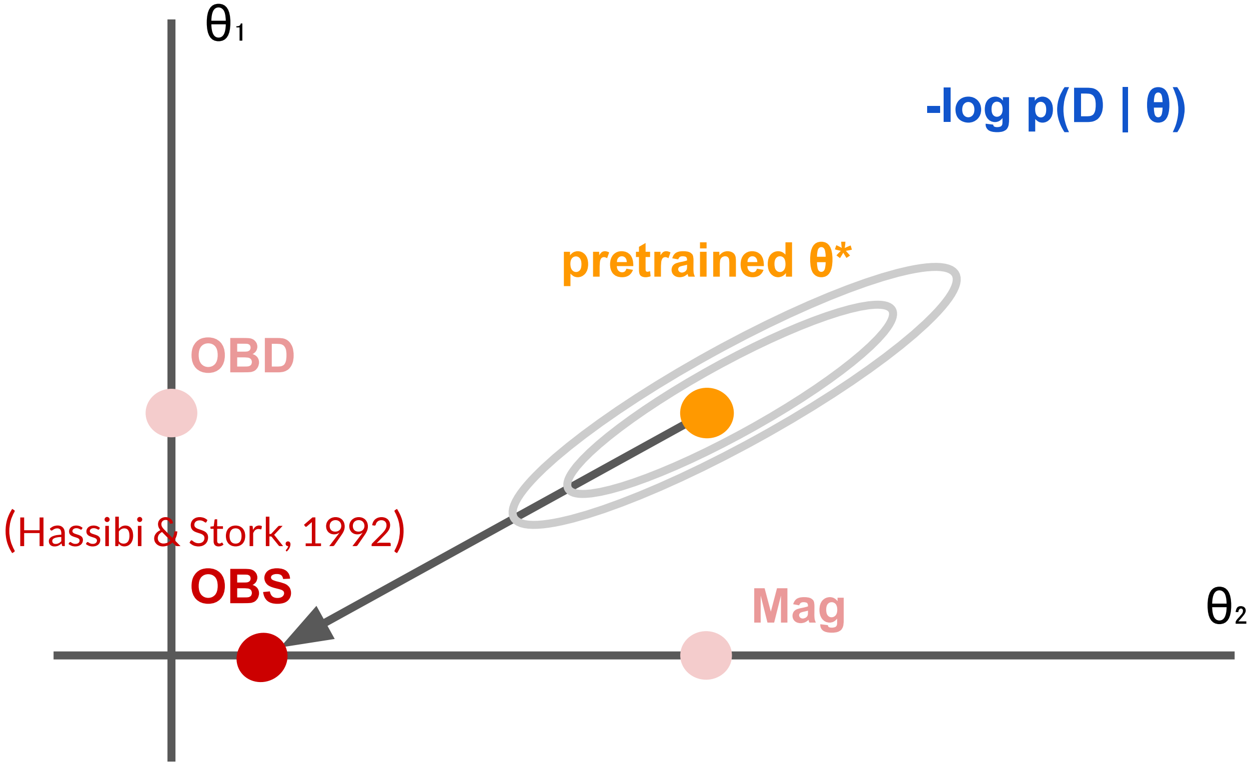
Geometric view on pruning



Geometric view on pruning



Geometric view on pruning

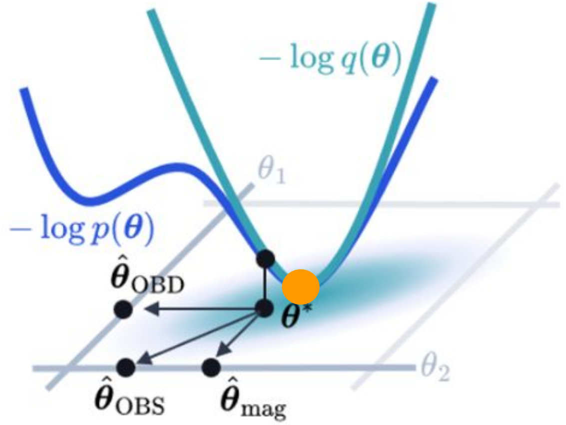


Closed-form constraint optimization

Solve the following quadratic constraint optimization problem (OBS: Hassibi & Stork, 1992)

$$\arg\min_{\Delta\boldsymbol{\theta}} \frac{1}{2}\Delta\boldsymbol{\theta}^T \boldsymbol{F} \Delta\boldsymbol{\theta}$$

$$\text{s.t. } \boldsymbol{e}_k^T \Delta\boldsymbol{\theta} + \boldsymbol{e}_k^T \boldsymbol{\theta} = 0, \forall k \in \mathcal{K}$$



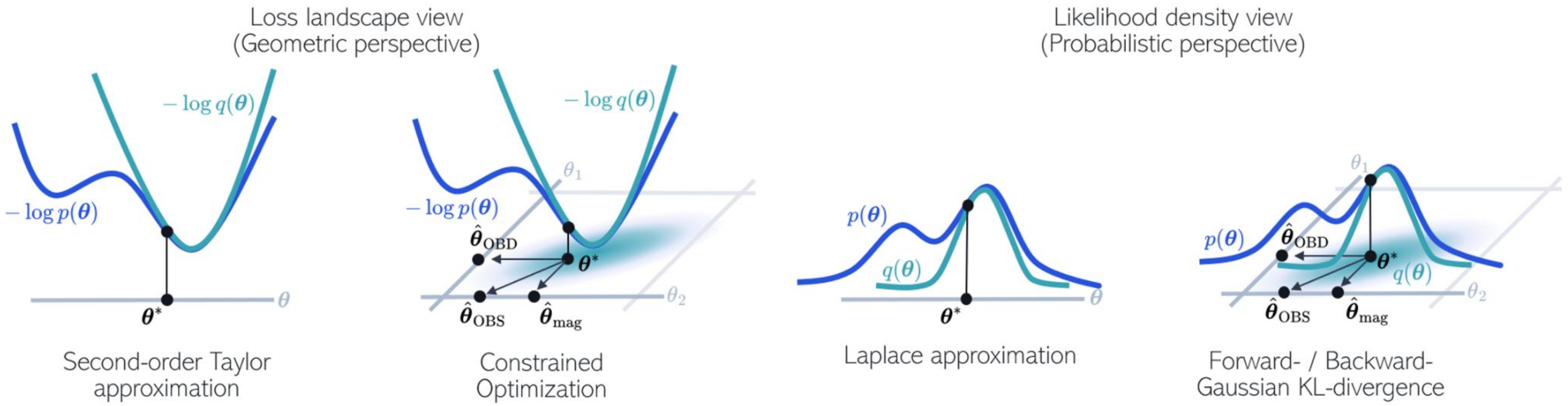
General solution (in LLM context: Kurtic et al. (2022))

$$\mathcal{L} = \frac{1}{2}(\boldsymbol{E}_K\boldsymbol{\theta}^*)^T \left(\boldsymbol{E}_K\boldsymbol{F}^{-1}\boldsymbol{E}_K^T\right)^{-1} \boldsymbol{E}_K\boldsymbol{\theta}$$

$$\Delta\boldsymbol{\theta} = -\boldsymbol{F}^{-1}\boldsymbol{E}_K^T \left(\boldsymbol{E}_K\boldsymbol{F}^{-1}\boldsymbol{E}_K^T\right)^{-1} \boldsymbol{E}_K\boldsymbol{\theta}$$

One slide on the probabilistic perspective...

Actually loss is regularised: **$-\log p(D \mid \theta) + \log p(\theta)$** by a log prior.
Prior variance plays critical role in implementation as `damping' term.



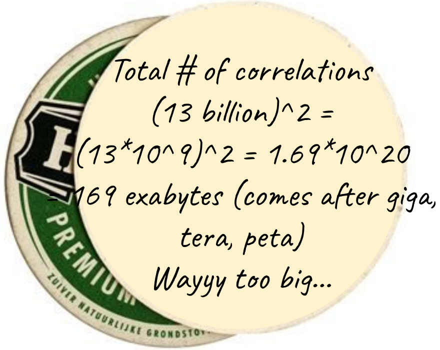
- We perform a Laplace approximation of the likelihood or posterior.
- Good pruning is all about correlations! Avoid the mean-field assumption.

Curse of squaring a large number

The Hessian of a 13 billion parameter LLM contains 1.69 × 10^20 elements!

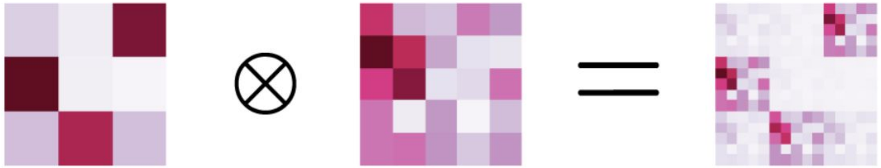
13 billion

13 billion



Waaaayyy to big…

Kronecker-factors



The Kronecker product ⊗ operates on two matrices of arbitrary size and results in a block matrix.

Nice way to write factorisations/decompositions for tensors.

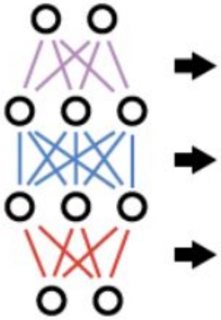
Very natural operation, and broadcasted multiplication under a reshuffling:

```
(A.view(3, 1, 3, 1) * B.view(1, 4, 1, 4)).view(12, 12)
```

Often pops up in factorisations/decompositions of tensors. Keeps math clean.

Modern Hessian approximations

The Hessian of a 13 billion parameter LLM contains 1.69e+20 elements!



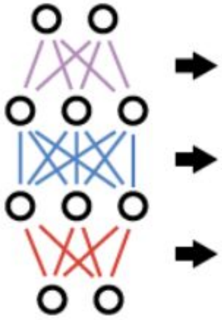
$A_1 \otimes G_1$	$A_1 \otimes G_2$	$A_1 \otimes G_3$
$A_2 \otimes G_1$	$A_2 \otimes G_2$	$A_2 \otimes G_3$
$A_3 \otimes G_1$	$A_3 \otimes G_2$	$A_3 \otimes G_3$

Note: No approximations yet!

Waaaayyy to big…

What often happens in prior work

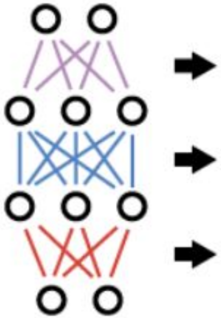
Most pruning works ignore `layer-wise' interactions, BUT make it completely `local'.



$A_1 \otimes I$	0	0
0	$A_2 \otimes I$	0
0	0	$A_3 \otimes I$

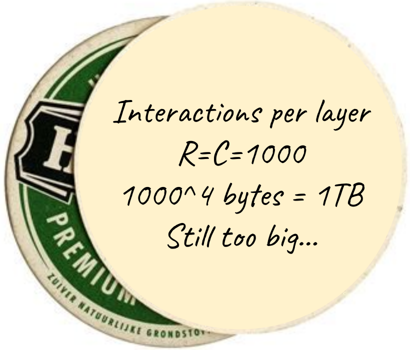
- Very cheap.
- No gradient info.
- Ignores final loss. (equivalent to summing local squared losses on output of each layer)

Modern Hessian approximations



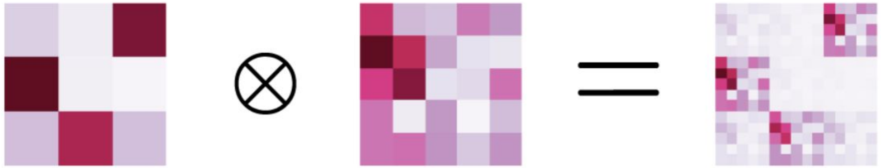
$$\begin{bmatrix} A_1 \otimes G_1 & 0 & 0 \\ 0 & A_2 \otimes G_2 & 0 \\ 0 & 0 & A_3 \otimes G_3 \end{bmatrix}$$

Still quite big…



$$\boldsymbol{F}_l = \sum_{n=1}^{N} \mathbb{E}\Big[\underbrace{(\boldsymbol{g}_{l,n}\boldsymbol{g}_{l,n}^T) \otimes (\boldsymbol{a}_{l,n}\boldsymbol{a}_{l,n}^T)}_{RC \times RC}\Big]$$

Modern Hessian approximations

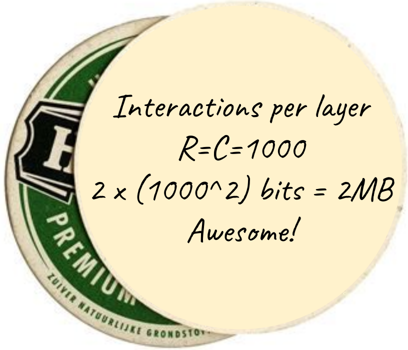


The Kronecker product ⊗ operates on two matrices of arbitrary size and results in a block matrix.

Assume independent input and outputs (KFAC: Martens & Grosse,

$$\underbrace{\mathbb{E}[\boldsymbol{g}_{l,n}\boldsymbol{g}_{l,n}^T \otimes \boldsymbol{a}_{l,n}\boldsymbol{a}_{l,n}^T]}_{\text{(RC x RC)}} \approx \underbrace{\mathbb{E}[\boldsymbol{g}_{l,n}\boldsymbol{g}_{l,n}^T]}_{\text{(R x R)}} \otimes \underbrace{\mathbb{E}[\boldsymbol{a}_{l,n}\boldsymbol{a}_{l,n}^T]}_{\text{(C x C)}}$$

Still quite big… Great! Great!



Implementation using *hooks*:

During all fwd and bwd passes, maintain aggregates of activations (aa^T) and gradients (gg^T).

Aggregates can be moved to ram, if needed.

Modern Hessian approximations

Constraint optimization problem

Solve the following quadratic constraint optimization problem (OBS: Hassibi & Stork, 1992)

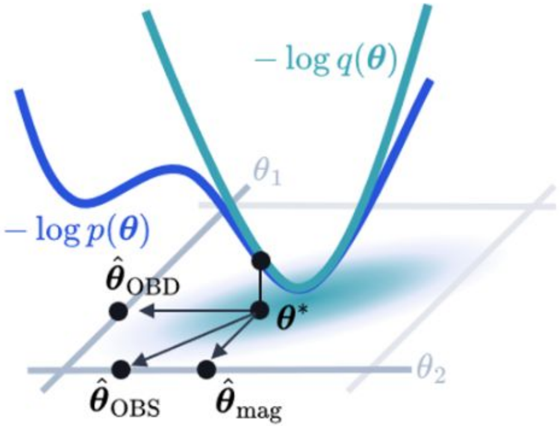
$$\arg\min_{\Delta\boldsymbol{\theta}} \; \frac{1}{2}\Delta\boldsymbol{\theta}^T \boldsymbol{F} \Delta\boldsymbol{\theta}$$

$$\text{s.t. } \boldsymbol{e}_k^T \Delta\boldsymbol{\theta} + \boldsymbol{e}_k^T \boldsymbol{\theta} = 0, \forall k \in \mathcal{K}$$

General solution (in LLM context: Kurtic et al. (2022))

$$\mathcal{L} = \frac{1}{2}(\boldsymbol{E}_K \boldsymbol{\theta}^*)^T \left(\boldsymbol{E}_K \boldsymbol{F}^{-1} \boldsymbol{E}_K^T\right)^{-1} \boldsymbol{E}_K \boldsymbol{\theta}$$

$$\Delta\boldsymbol{\theta} = -\boldsymbol{F}^{-1}\boldsymbol{E}_K^T \left(\boldsymbol{E}_K \boldsymbol{F}^{-1} \boldsymbol{E}_K^T\right)^{-1} \boldsymbol{E}_K \boldsymbol{\theta}$$



Paper provides derivations for all structures {*unstructured*, *semi-structured*, *structured*}

Algorithm outline for structured pruning

1. **Compute removal cost** for each row and column

$$\mathcal{L}_r = \underline{\frac{1}{2}\frac{\boldsymbol{\theta}_r^T \boldsymbol{A} \boldsymbol{\theta}_r}{[\boldsymbol{G}^{-1}]_{rr}}, \quad \mathcal{L}_c = \frac{1}{2}\frac{\boldsymbol{\theta}_c^T \boldsymbol{G} \boldsymbol{\theta}_c}{[\boldsymbol{A}^{-1}]_{cc}}}$$

Scales not in #elements anymore, but only in #rows and #columns!

2. **Global thresholding** by sorting all costs and selecting op X% for removal
3. **Update** remaining weights using correlated weight updates

$$\Delta \boldsymbol{W} = -\underline{\overline{\boldsymbol{W}}(\boldsymbol{E}_{C'}\boldsymbol{A}^{-1}\boldsymbol{E}_{C'}^T)^{-1}(\boldsymbol{A}^{-1}\boldsymbol{E}_{C'}^T)}$$

$$\Delta \boldsymbol{W} = -\underline{\boldsymbol{G}^{-1}\boldsymbol{E}_{R'}^T(\boldsymbol{E}_{R'}\boldsymbol{G}^{-1}\boldsymbol{E}_{R'}^T)^{-1}\overline{\boldsymbol{W}}}$$

(among new results)

Can be efficiently implemented by indexing rows/cols.

Scales not in #elements anymore, but only in #rows and #columns!

4. **Repeat** for multiple shots

Pseudo code

Algorithm 1 LLM Surgeon (*structured*)

Input: initial weights $\boldsymbol{\theta}^0$, target size α, and data $\mathcal{D}$

For shot t in $[1, 2, \ldots, T]$

- **Compute:** approximate curvature $\boldsymbol{G}, \boldsymbol{A}$ from data $\mathcal{D}$ ▷ section 3.1
- **Compute:** costs per row/column $\mathcal{L}_r, \mathcal{L}_c$ from $\boldsymbol{G}, \boldsymbol{A}$ ▷ section 3.2
- **Compute:** threshold τ using $\mathcal{L}_r$ and $\mathcal{L}_c$ given target size α_t ▷ section 3.3
- **Select:** rows and columns to remove $\boldsymbol{E}_R, \boldsymbol{E}_C$ based on τ ▷ section 3.3
- **Compute:** weight update $\Delta\boldsymbol{\theta}^{t-1}$ based on $\boldsymbol{E}_R, \boldsymbol{E}_C$ and $\boldsymbol{G}, \boldsymbol{A}$ ▷ section 3.4
- **Update:** remaining weights $\boldsymbol{\theta}^t \leftarrow \boldsymbol{\theta}^{t-1} + \Delta\boldsymbol{\theta}^{t-1}$ ▷ section 3.5
- **Optionally:** $\boldsymbol{\theta}^t \leftarrow$ low-rank update$(\boldsymbol{\theta}^t)$ ▷ section 3.6

Output: compressed weights $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^T$

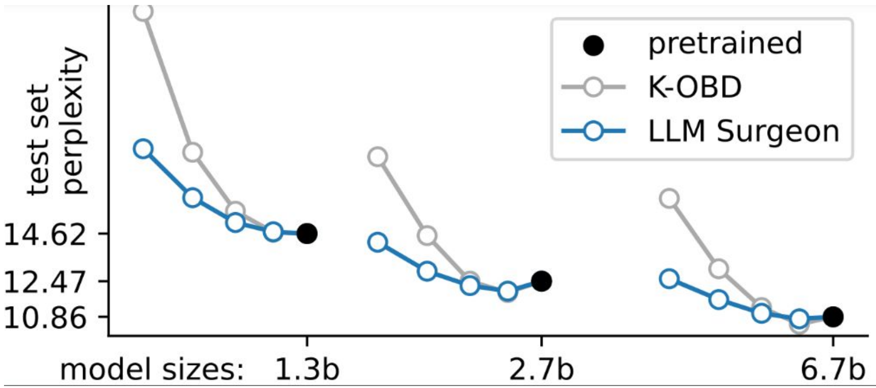
Optionally, can be interleaved with first-order LoRA corrections.

Useful trick: absorb in between to allow
increase rank of sum of LoRA updates!

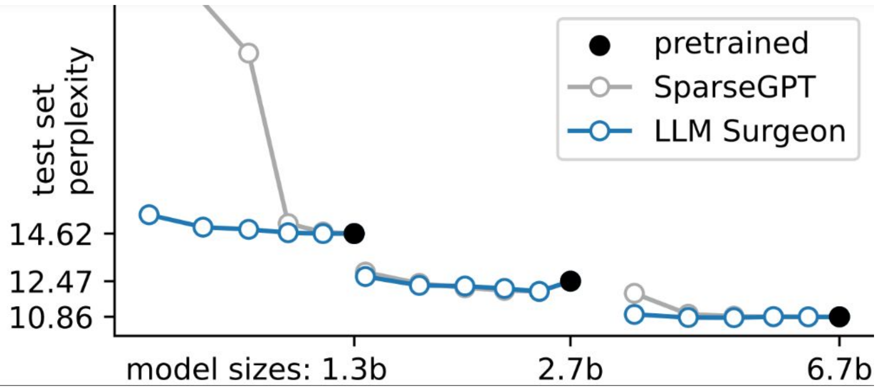
Results

Interpolate model sizes

Structured pruning



Unstructured pruning



Results

Quantitative benchmark

Structured pruning results

Table 1: Structured compression of large language models on wikitext-2 data.

Method	Target size	Test performance (PPL)				
		OPT (125m)	OPT (1.3b)	OPT (2.7b)	OPT (6.7b)	Llama-v2 (7b)
Baseline	100%	**27.65**	**14.62**	**12.47**	**10.86**	**5.12**
Magnitude	90%	767.2	894.4	1229	3464	36746
$\boldsymbol{I} \otimes \boldsymbol{I}$	80%	4685	(1278)	2788	16747	347960
	70%	17970	(3098)	9255	17312	41373
L-OBD	90%	33.3	20.76	17.69	27.20	14259
diag($\boldsymbol{I} \otimes \boldsymbol{A}$)	80%	94.14	1392	3236	7570	15630
multi shot	70%	545.6	2147	7233	7628	21386
K-OBD	90%	27.97	14.68	11.96	10.53	5.48
diag($\boldsymbol{G} \otimes \boldsymbol{A}$)	80%	29.89	15.63	12.47	11.28	9.14
multi shot	70%	36.54	18.29	14.53	13.03	15.43
	60%	47.54	24.65	18.09	16.21	28.03
	50%	75.95	37.68	26.68	25.54	46.64
LLM Surgeon (**ours**)	90%	28.29	14.73	12.00	10.82	5.43
$\boldsymbol{G} \otimes \boldsymbol{A}$	80%	29.37	15.27	12.37	11.22	7.29
within row/col cor. Δ	70%	32.46	16.60	13.16	11.83	10.85
	60%	39.82	19.40	14.79	12.94	16.67
	50%	51.48	23.81	18.01	15.38	25.62
LLM Surgeon (**ours**)	90%	28.01	14.70	12.02	10.77	5.25
$\boldsymbol{G} \otimes \boldsymbol{A}$	80%	28.73	15.12	12.27	11.02	6.18
full cor. Δ	70%	31.82	16.24	12.92	11.64	7.83
	60%	38.47	18.45	14.23	12.58	10.39
	50%	49.78	22.95	17.15	14.90	15.38

Unstructured pruning results

Table 4: Unstructured compression of large language models on wikitext-2 data.

Method	Target size	Test performance (PPL)				
		OPT (125m)	OPT (1.3b)	OPT (2.7b)	OPT (6.7b)	Llama-v2 (7b)
Baseline	100%	**27.65**	**14.62**	**12.47**	**10.86**	**5.12**
Magnitude	90%	27.62	14.69	12.60	10.88	5.18
$\boldsymbol{I} \otimes \boldsymbol{I}$	80%	28.53	15.68	13.18	11.26	5.37
	70%	52.88	140.2	15.22	12.22	6.03
L-OBD	90%	29.70	16.24	14.44	13.43	6.09
diag($\boldsymbol{I} \otimes \boldsymbol{A}$)	80%	32.18	21.92	23.35	39.85	116.2
single shot	70%	49.08	204.7	274.8	810.4	6549
K-OBD	90%	27.64	14.62	12.09	36.89	5.13
$\boldsymbol{G} \otimes \boldsymbol{A}$	80%	27.62	14.37	130220	39928	5.19
single shot	70%	27.92	220.1	23097	19506	5.60
	60%	29.24	13783	10331	33896	9.20
	50%	34.43	7311	10495	91506	118.6
SparseGPT	90%	27.93	14.69	12.00	10.86	5.49
$\boldsymbol{I} \otimes \boldsymbol{A}$	80%	28.18	15.07	12.05	10.86	5.58
	70%	28.93	22.77	12.17	10.89	5.71
	60%	30.20	25.07	12.37	10.98	5.94
	50%	33.17	26.77	12.88	11.92	6.51
LLM Surgeon (**ours**)	90%	27.69	14.62	12.01	10.86	5.13
$\boldsymbol{G}_1 \otimes \boldsymbol{A}_1$	80%	27.83	14.66	12.14	10.87	5.20
full cor. Δ	70%	28.35	14.81	12.25	10.82	5.36
multi shot	60%	28.98	14.91	12.28	10.83	5.66
	50%	30.30	15.47	12.68	10.97	6.08

Semi-structured (2:4) pruning results

Table 3: Semi-structured 2:4 compression for large language models on wikitext-2 data.

Method	$\boldsymbol{F} \approx$	Target size	Test performance (PPL)			
			OPT (125m)	OPT (1.3b)	OPT (2.7b)	OPT (6.7b)
Baseline		100%	**27.65**	**14.62**	**12.47**	**10.86**
Magnitude	$\boldsymbol{I} \otimes \boldsymbol{I}$	50%	342.04	379.57	1106.01	187.29
L-OBD	diag($\boldsymbol{I} \otimes \boldsymbol{A}$)	50%	87.26	44.92	41.40	27.36
K-OBD	diag($\boldsymbol{G} \otimes \boldsymbol{A}$)	50%	68.74	27.22	20.23	15.55
SparseGPT	$\boldsymbol{I} \otimes \boldsymbol{A}$	50%	45.51	29.44	14.92	13.01
LLM Surgeon (**ours**)	$\boldsymbol{G} \otimes \boldsymbol{A}$	50%	44.64	25.10	14.64	12.10

Similar findings for performance on downstream tasks!

Results

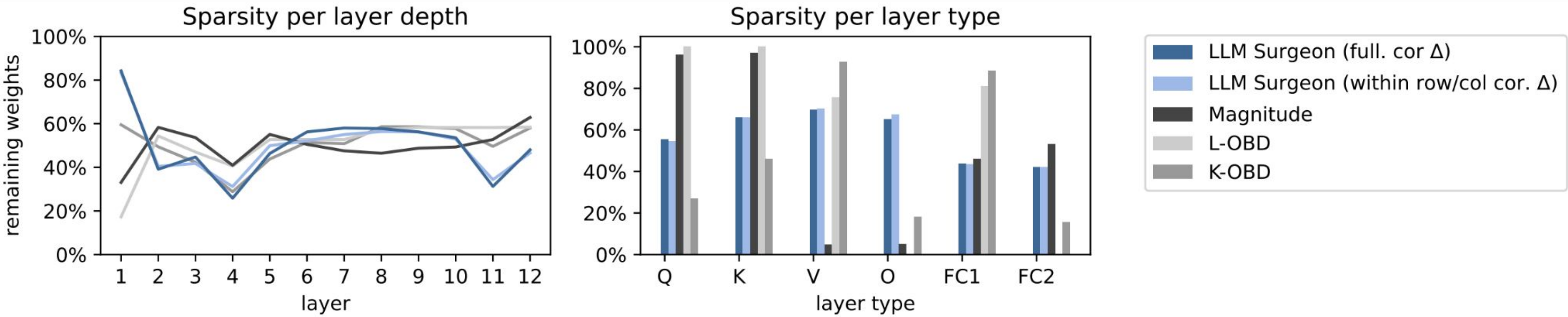
Task-specific compression

Can be used to project existing pretrained models to tailored smaller model.

	evaluation dataset				mask equivalence (%)			
target	EN	FR	DE	IT	EN	FR	DE	IT
Pretrained	**27.66**	**22.54**	**24.32**	**27.66**				
EN	**47.46**	172.9	181.1	169.1	1.00	0.74	0.70	0.72
FR	113.4	**28.44**	35.02	34.90	0.74	1.00	0.87	0.90
DE	142.1	35.15	**27.49**	38.49	0.70	0.87	1.00	0.87
IT	123.7	31.85	33.78	**30.58**	0.72	0.90	0.87	1.00

Results

Analysing sparsification



THE LLM SURGEON

Tycho F.A. van der Ouderaa[1*], **Markus Nagel**[2], **Mart van Baalen**[2],
Yuki M. Asano[3], **Tijmen Blankevoort**[2]
[1]Imperial College London , [2]Qualcomm AI Research[†], [3]QUVA Lab, University of Amsterdam

Tycho van der Ouderaa
Twitter/X: tychovdo
Email: tychovdo@gmail.com
Web: tychovdo.ai

Imperial College London

UNIVERSITY OF OXFORD

Qualcomm

Beyond independent inputs and outputs

Nearest Kronecker product with Kronecker power iteration

Algorithm 4 Kronecker power method. Finds $\widetilde{\boldsymbol{G}}, \widetilde{\boldsymbol{A}}$ nearest Kronecker product $||\boldsymbol{F} - \widetilde{\boldsymbol{G}} \otimes \widetilde{\boldsymbol{A}}||_F$.

Input: Initialise $\widetilde{\boldsymbol{g}}^0 = \mathbf{1}, \widetilde{\boldsymbol{a}}^0 = \mathbf{1}$ (or using estimates of previous shot).
Input: Set iterations I (or I=1 if using estimates from previous shot)
Output: $\widetilde{\boldsymbol{G}}, \widetilde{\boldsymbol{A}}$
  for iteration i in $[1, 2, \ldots, I]$ **do**
    Compute: $\widetilde{\boldsymbol{g}}^i = \frac{\mathcal{R}(\widetilde{\boldsymbol{F}})\widetilde{\boldsymbol{a}}^{i-1}}{||\mathcal{R}(\widetilde{\boldsymbol{F}})\widetilde{\boldsymbol{a}}^{i-1}||_2}$, with $\mathcal{R}(\widetilde{\boldsymbol{F}})\widetilde{\boldsymbol{a}}^{i-1} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{a}_n^T \widetilde{\boldsymbol{A}}^{i-1} \boldsymbol{a}_n \mathrm{vec}(\boldsymbol{g}_n \boldsymbol{g}_n^T)$
    Compute: $\widetilde{\boldsymbol{a}}^i = \frac{\mathcal{R}(\widetilde{\boldsymbol{F}})^T \widetilde{\boldsymbol{g}}^i}{||\mathcal{R}(\widetilde{\boldsymbol{F}})^T \widetilde{\boldsymbol{g}}^i||_2}$, with $\mathcal{R}(\widetilde{\boldsymbol{F}})^T \widetilde{\boldsymbol{g}}^i = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{g}_n^T \widetilde{\boldsymbol{G}}^i \boldsymbol{g}_n \mathrm{vec}(\boldsymbol{a}_n \boldsymbol{a}_n^T)$
    Compute: $\sigma^i = ||\widetilde{\boldsymbol{a}}^i||_2$
  end for
  Return: $\widetilde{\boldsymbol{G}} = \sqrt{\sigma^i}\mathrm{mat}(\widetilde{\boldsymbol{g}}), \widetilde{\boldsymbol{A}} = \sqrt{\sigma^i}\mathrm{mat}(\widetilde{\boldsymbol{a}})$.