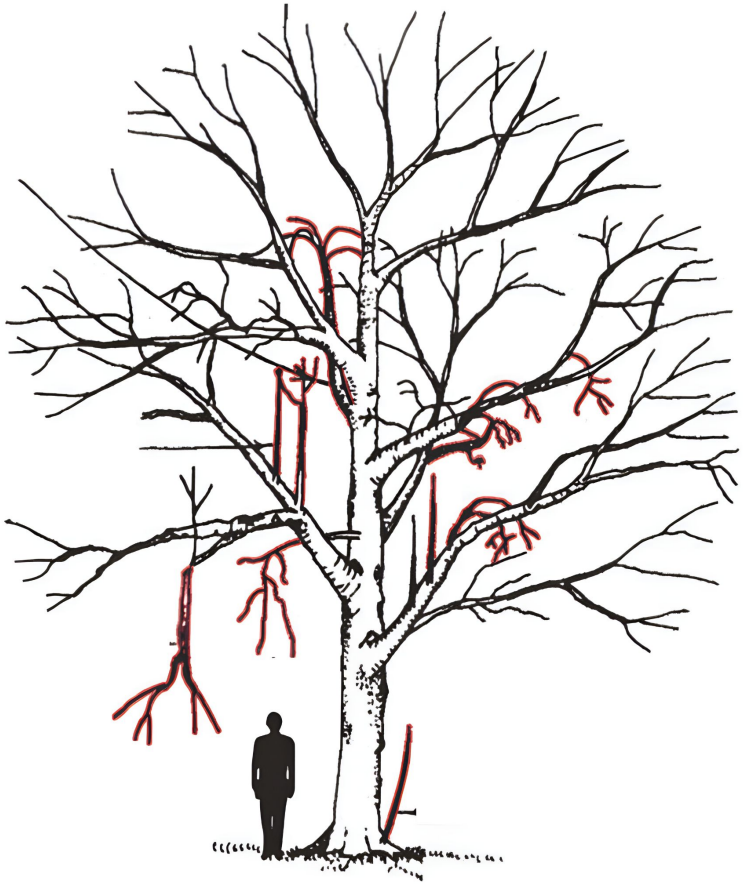
Спарсификация и дистилляция

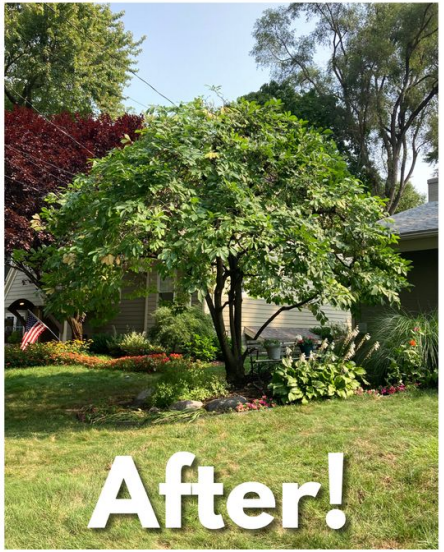
Дмитрий Осин @xaosina

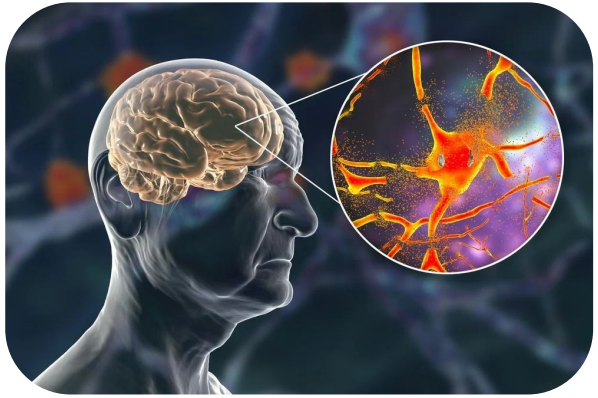
Содержание:

>>> Мотивация



Деревья часто несут большое количество мертвых ветвей, на которых нет ни листьев, ни плодов, что делает дерево излишне тяжелым. Эти мертвые ветки можно безопасно срезать, не затрагивая свинцовые ветки, которым в большинстве случаев полезно дополнительное пространство.





Обрезать или удалять связи, которые мы не используем - оптимизация из реальной жизни

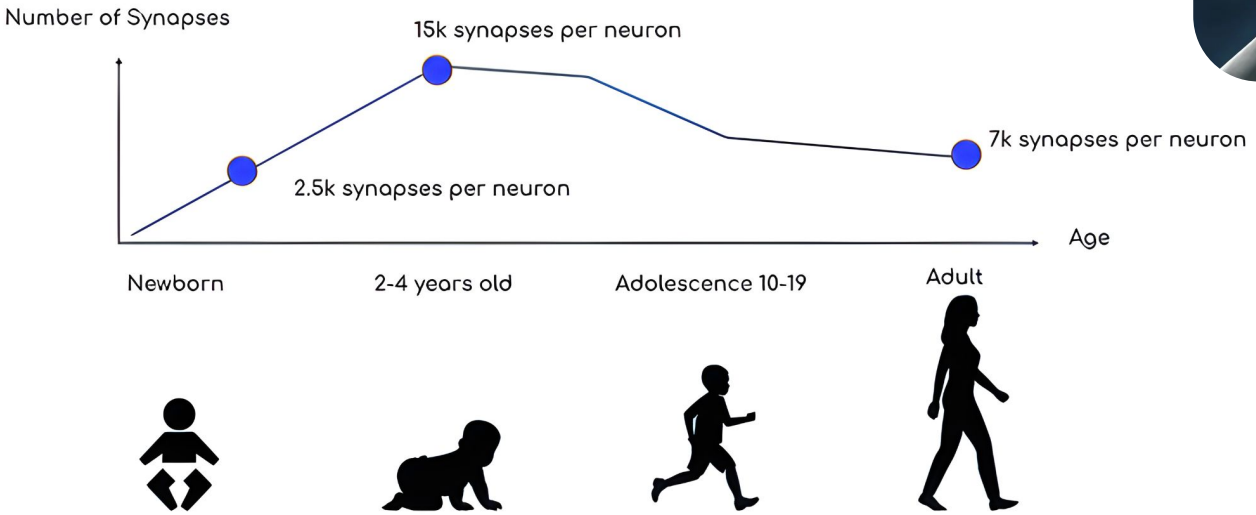


Image Source

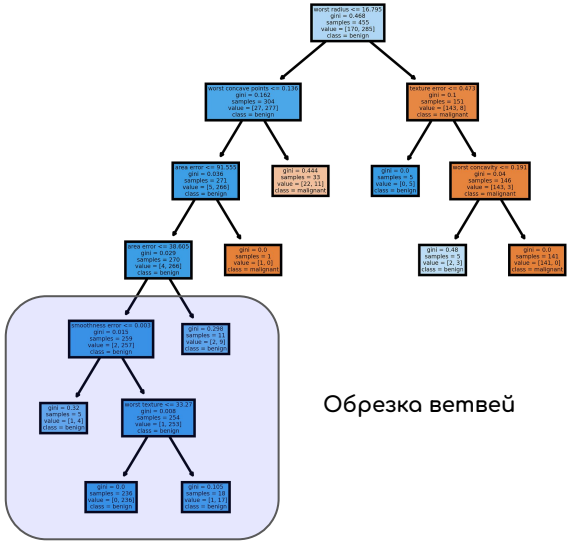
Прунинг
или
спарсификация

Прунинг приводит к спарсности

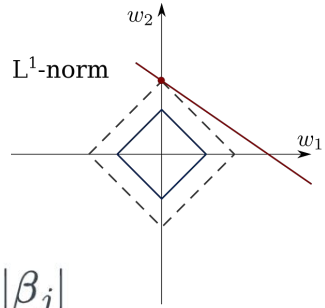
Хотя термины используются
взаимозаменяемо

Идея не нова

Деревья решений



Обрезка ветвей



Лассо регрессия

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Robust Principal Component Analysis - RPCA

$$W = S + L$$

S - разреженная

L - низкоранговая

Почему это работает и почему мы не можем с самого начала обучить меньшую модель?

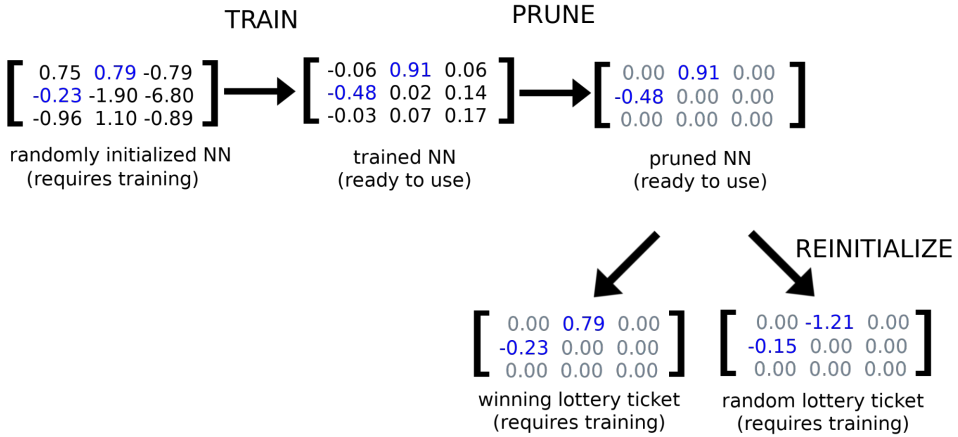
Модели оверпараметризованы
Но почему?

Инициализация

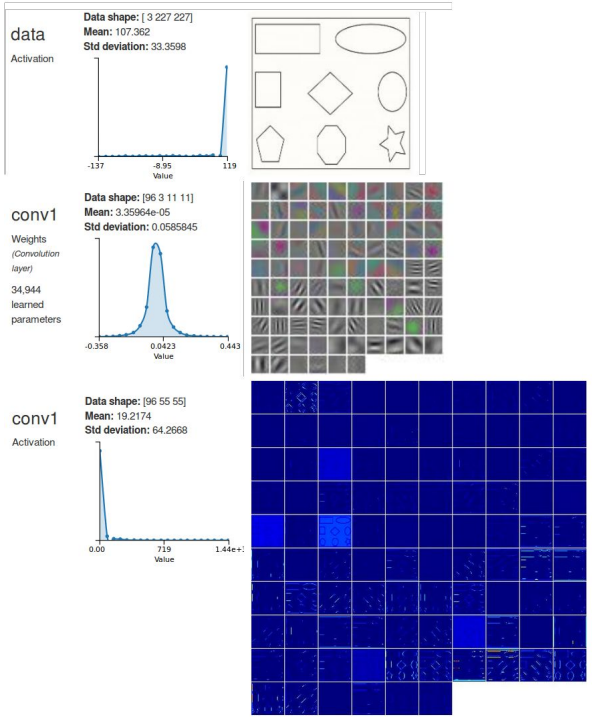
Динамика обучения (оптимизации)

Размер данных и их качество

Гипотеза статьи состоит в том, что внутри большой модели есть меньшая модель, которая, если ее обучать самостоятельно, будет соответствовать производительности более крупной модели.



[The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks]



Source

Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers

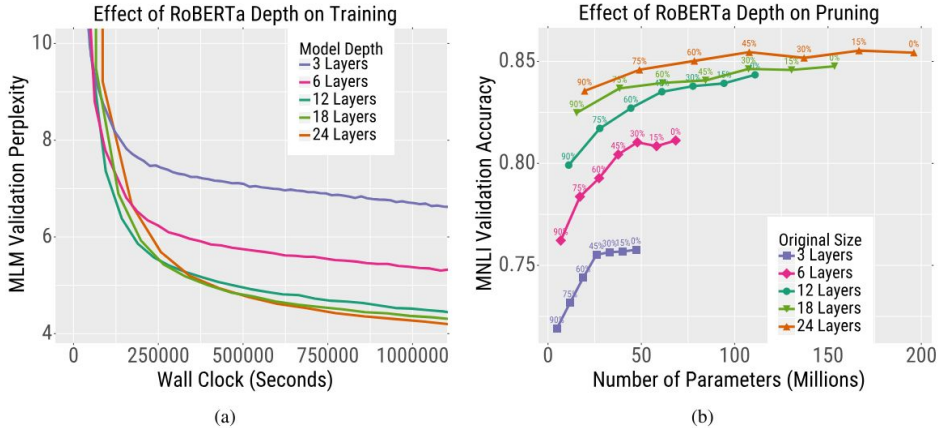


Figure 2. Increasing Transformer model size results in lower validation error as a function of *wall-clock time* and better test-time accuracy for a given *inference budget*. **(a)** demonstrates the training speedup for RoBERTa models of different sizes on the masked language modeling pretraining task. In **(b)**, we take RoBERTa checkpoints that have been pretrained for the *same* amount of wall-clock time and finetune them on a downstream dataset (MNLI). We then iteratively prune model weights to zero and find that the best models for a given test-time memory budget are ones which are trained large and then heavily compressed.

Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers

Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers

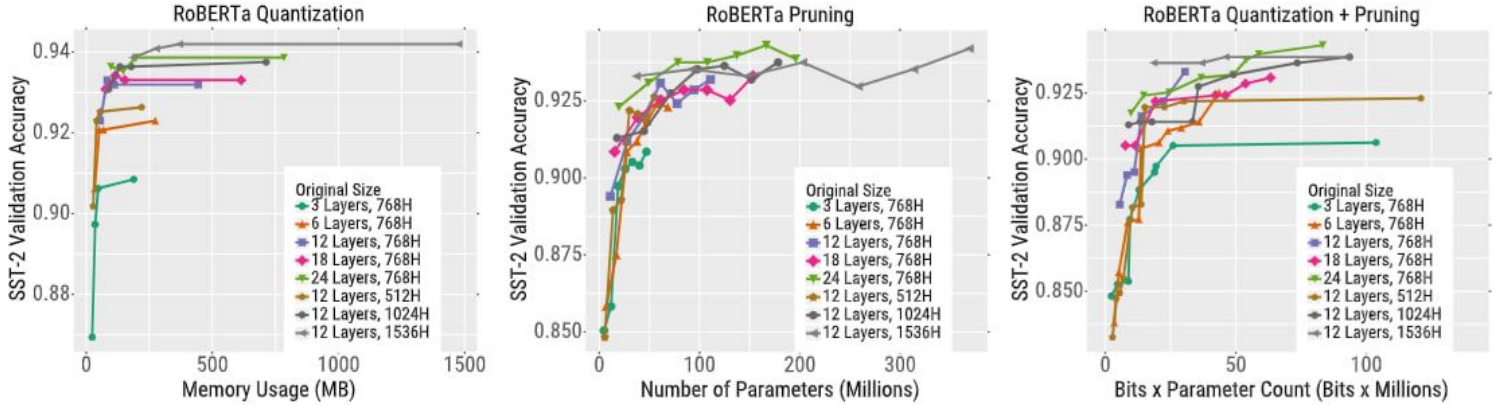


Figure 12. Compression for SST-2. For most budgets (x-axis), the highest accuracy SST-2 models are the ones which are trained large and then heavily compressed. We show results for quantization (left), pruning (center), and quantization and pruning (right).

Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers

Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers

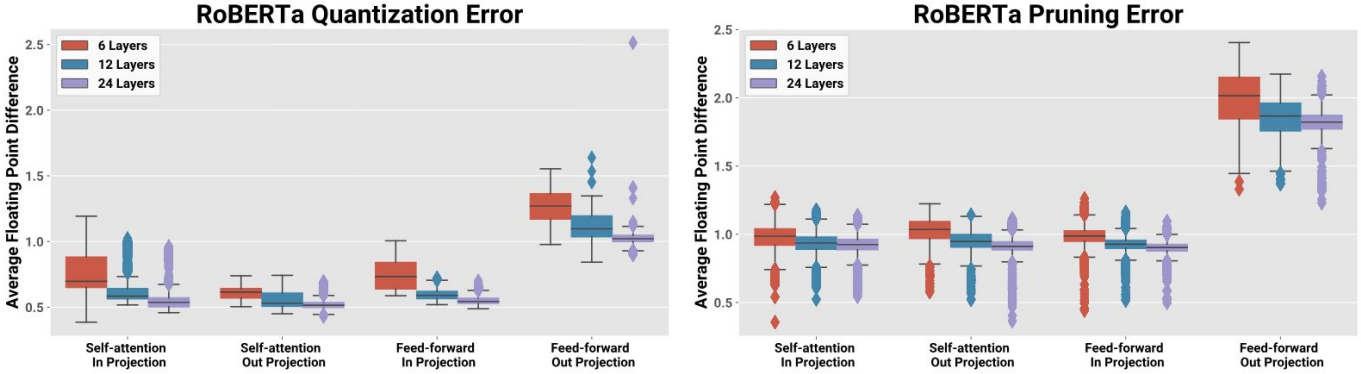
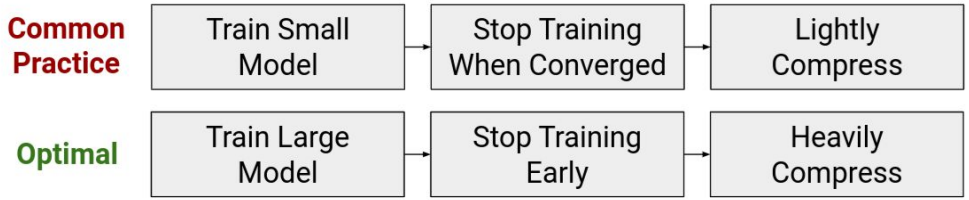


Figure 9. We finetune RoBERTa models of different sizes (6 layers, 12 layers, and 24 layers) on MNLI. We then quantize models to 4-bits or prune models to 60% sparsity. We plot the difference between the weights of the original and the quantized/pruned models averaged across different modules in the Transformer. The mean and variance of the weight difference after quantization (left) is consistently lower for the deeper models compared to the shallower models. The same holds for the difference after pruning (right). This shows that the larger model's weights are naturally easier to approximate with low-precision / sparse matrices than smaller models.

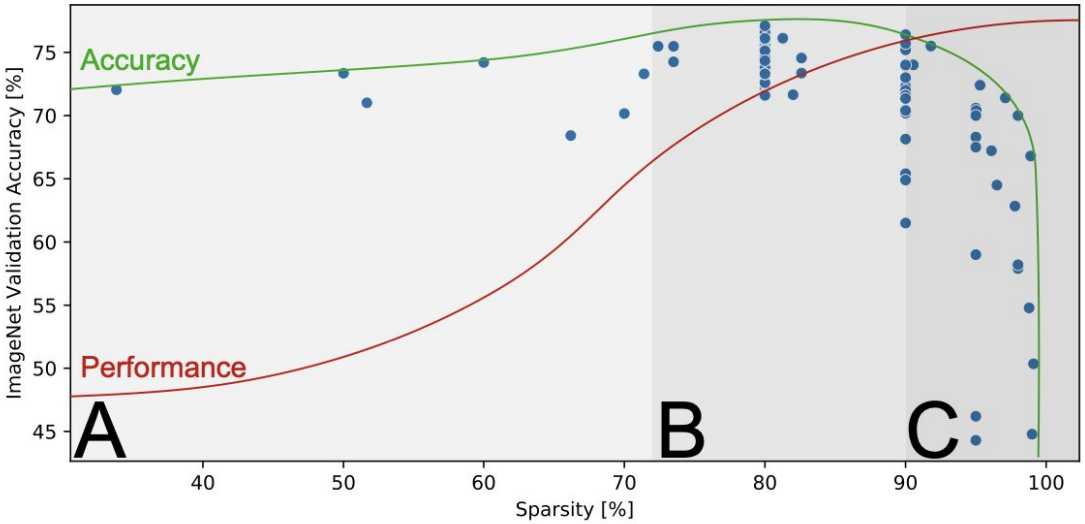
Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers

Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers



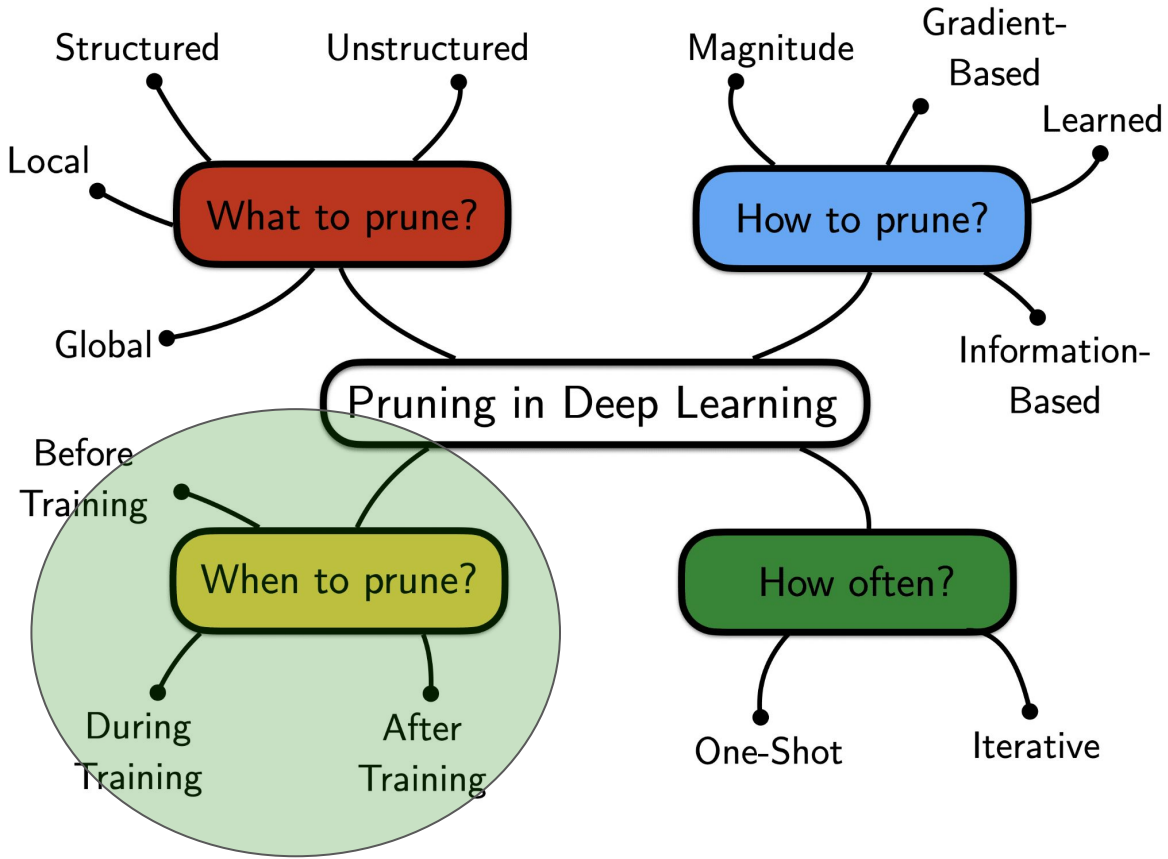
Resnet 50

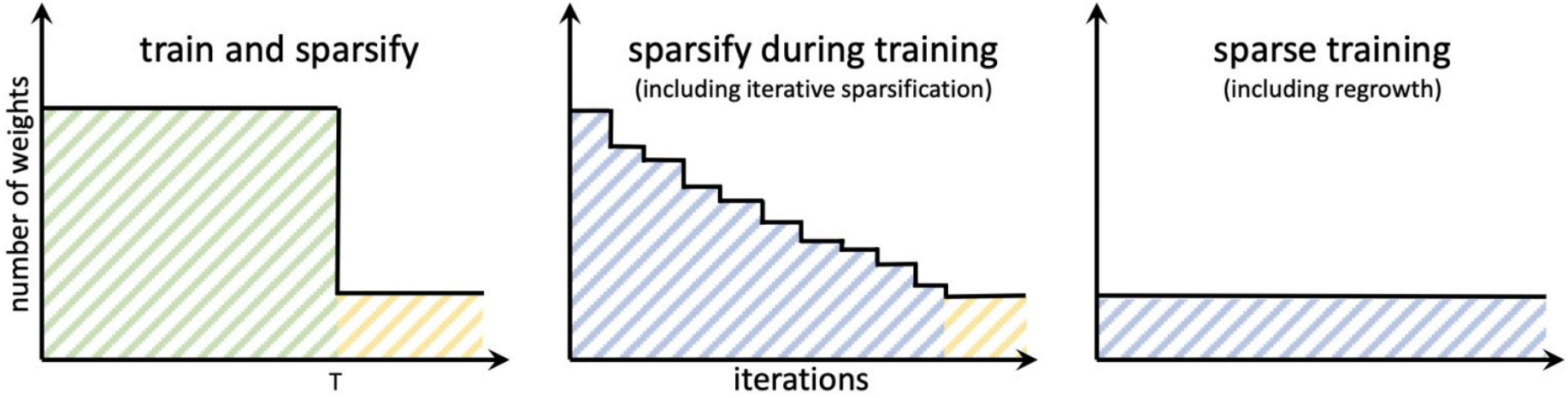
Иногда спарсификация улучшает качество



source

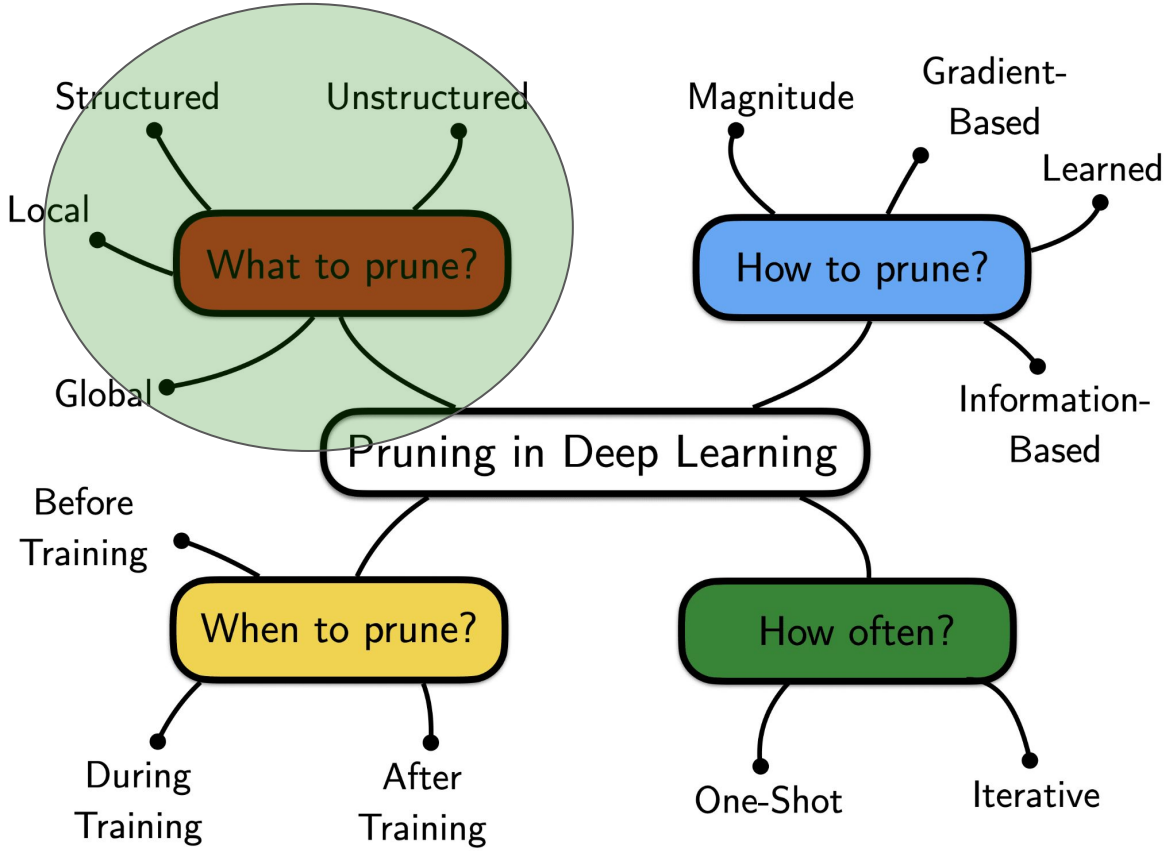
>>> Когда обрезать


Structured
Unstructured
Local
What to prune?
Global
Magnitude
Gradient-
Based
Learned
How to prune?
Information-
Based
Pruning in Deep Learning
Before
Training
When to prune?
During
Training
After
Training
How often?
One-Shot
Iterative

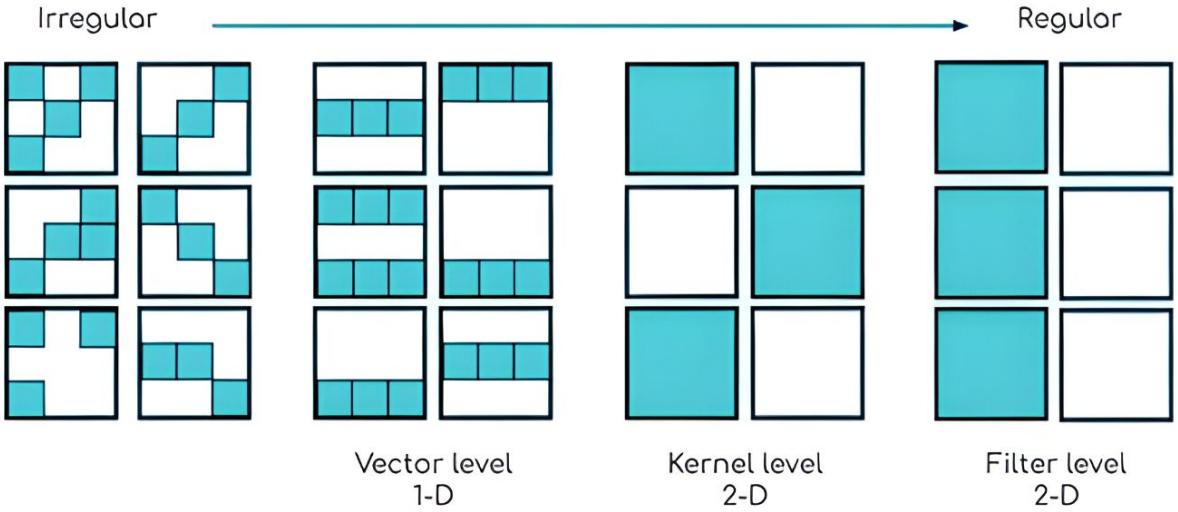


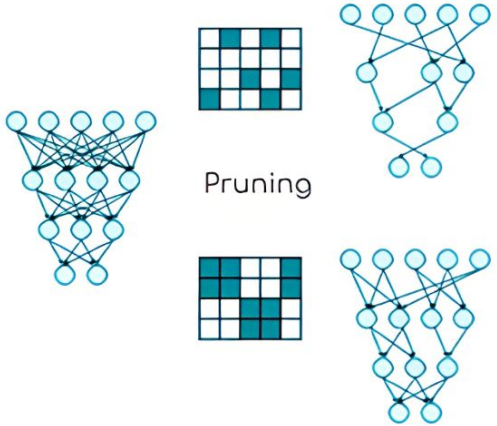
Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks

1. SNIP: Single-shot Network Pruning based on Connection Sensitivity
2. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science
3. Rigging the Lottery: Making All Tickets Winners
4. Picking Winning Tickets Before Training by Preserving Gradient Flow

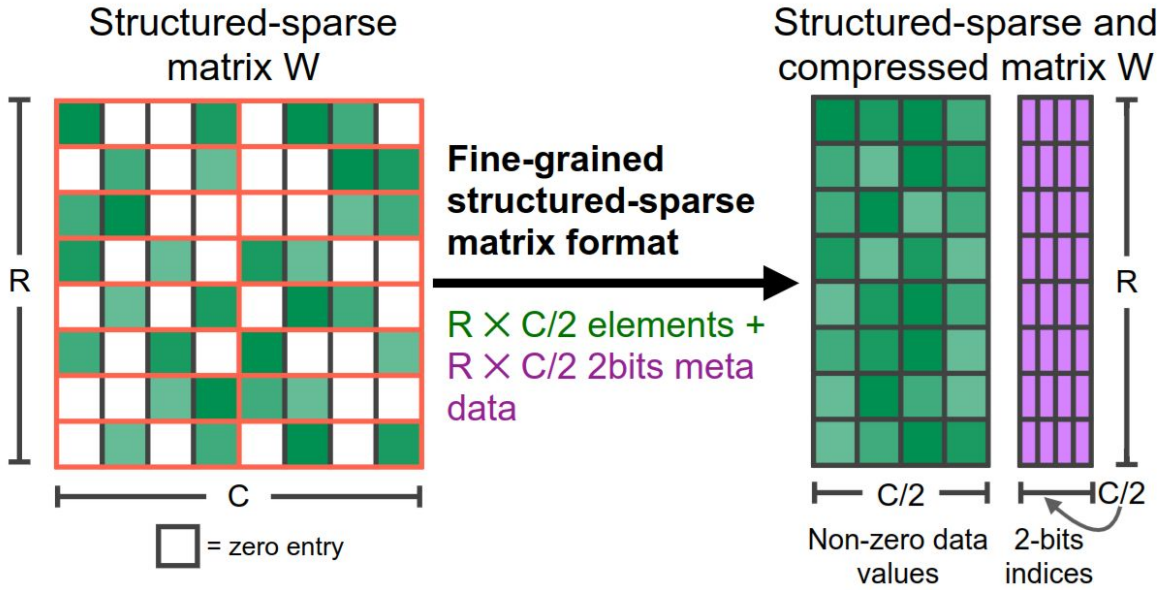

Pruning in Deep Learning
What to prune?
Structured
Unstructured
Local
Global
How to prune?
Magnitude
Gradient-Based
Learned
Information-Based
When to prune?
Before Training
During Training
After Training
How often?
One-Shot
Iterative

>>> Разреженные модели :
с точки зрения
аппаратного обеспечения


Irregular
Regular
Vector level
1-D
Kernel level
2-D
Filter level
2-D


Pruning

N:M разреженность — это структурированный шаблон разреженности, который хорошо работает с современной аппаратной оптимизацией графического процессора, в которой N из каждых M последовательных элементов являются нулями.

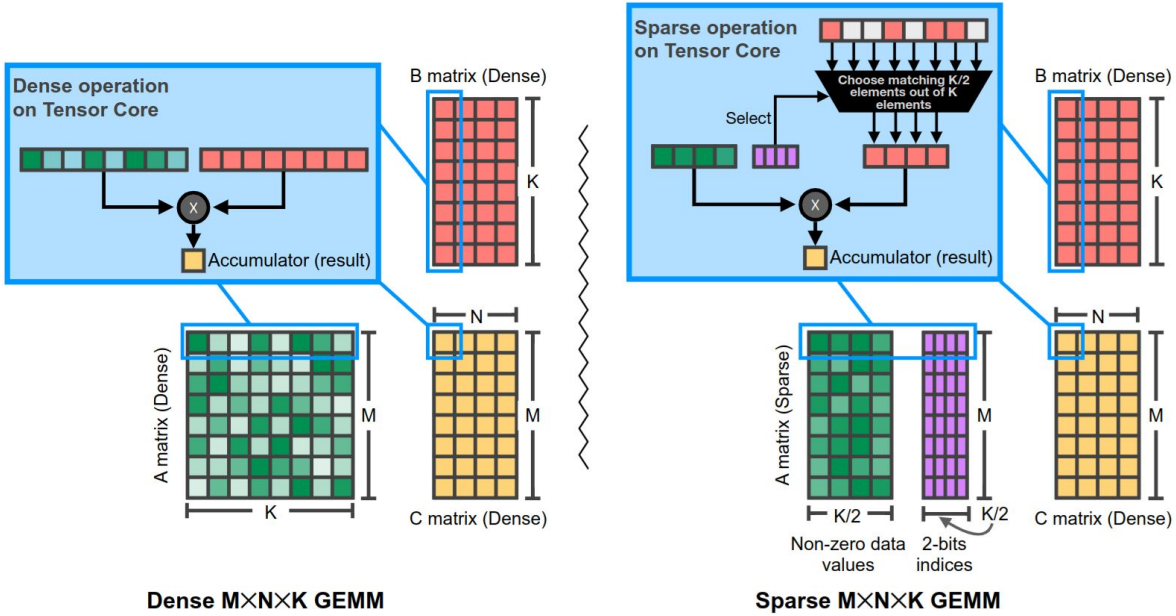


Accelerating Sparse Deep Neural Networks

Эффективность сжатого формата: использование формата CSR для неструктурированной разреженности может привести к увеличению накладных расходов на хранение из-за метаданных до 200 %.

Из-за размера блока из 4 значений разреженное хранилище 2:4
Формат требует только 2-битных метаданных на каждое значение, что ограничивает накладные расходы на хранение до 12,5% и 25% для значений 16b и 8b соответственно.

2:4 разреженность
~ 50% уменьшение весов
~ 2x ускорение математических вычислений

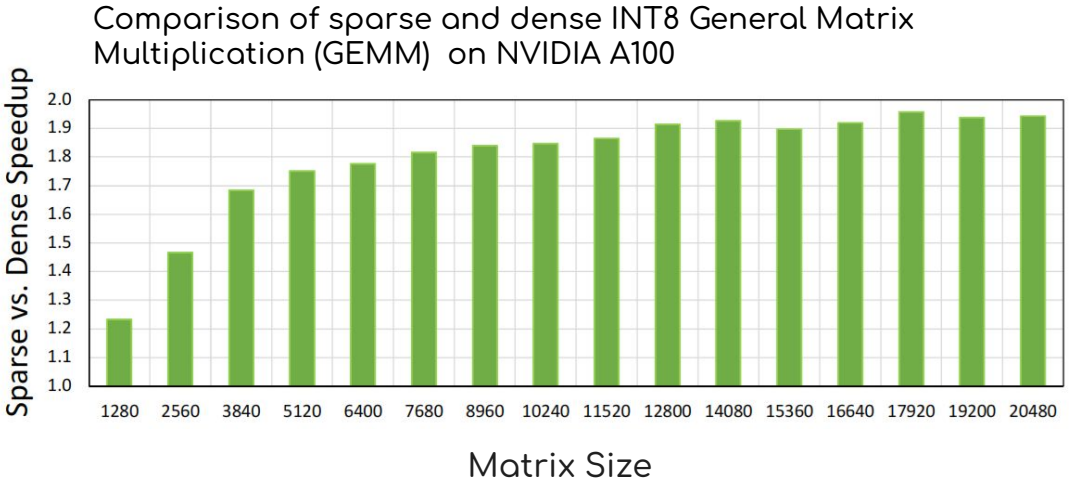


В настоящее время поддерживается только с:

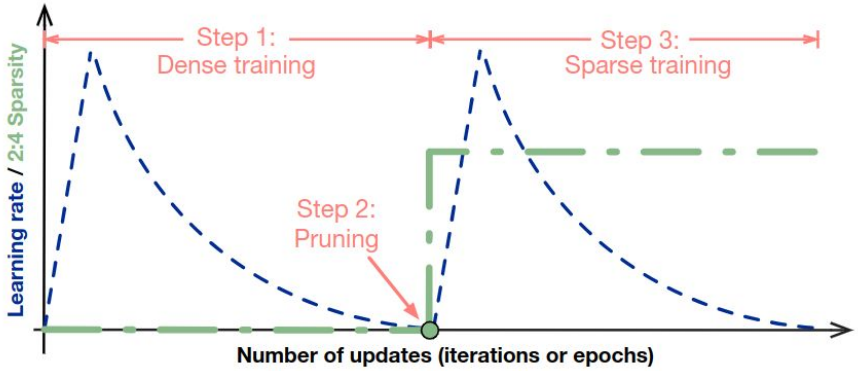
FP32, FP16, int8.

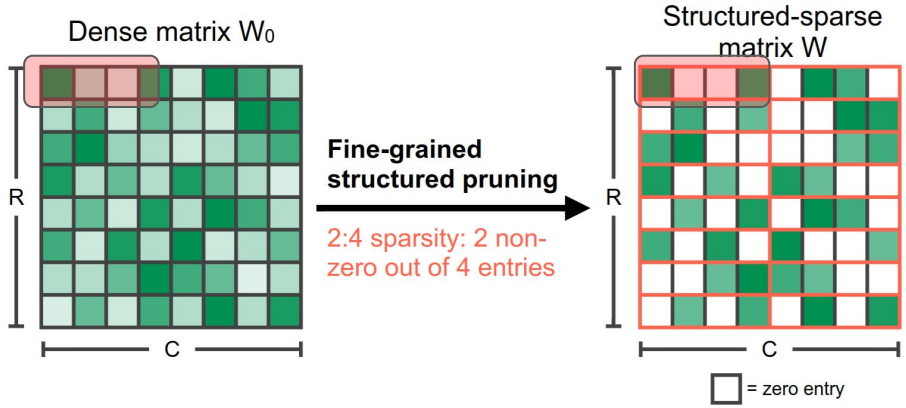
Accelerating Sparse Deep Neural Networks

2:4 разреженность
~ 50% уменьшение весов
~ 2x ускорение математических вычислений



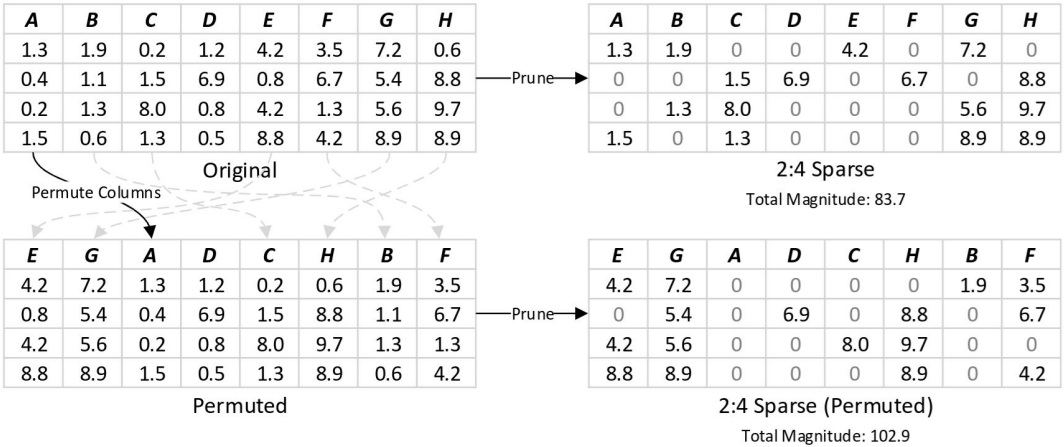
Accelerating Sparse Deep Neural Networks





Удалить два значения с наименьшей магнитудой(значением)
В каждой ячейке.

[Accelerating Sparse Deep Neural Networks]



Переставить столбцы, чтобы сохранить более важные веса с более высокой величиной.

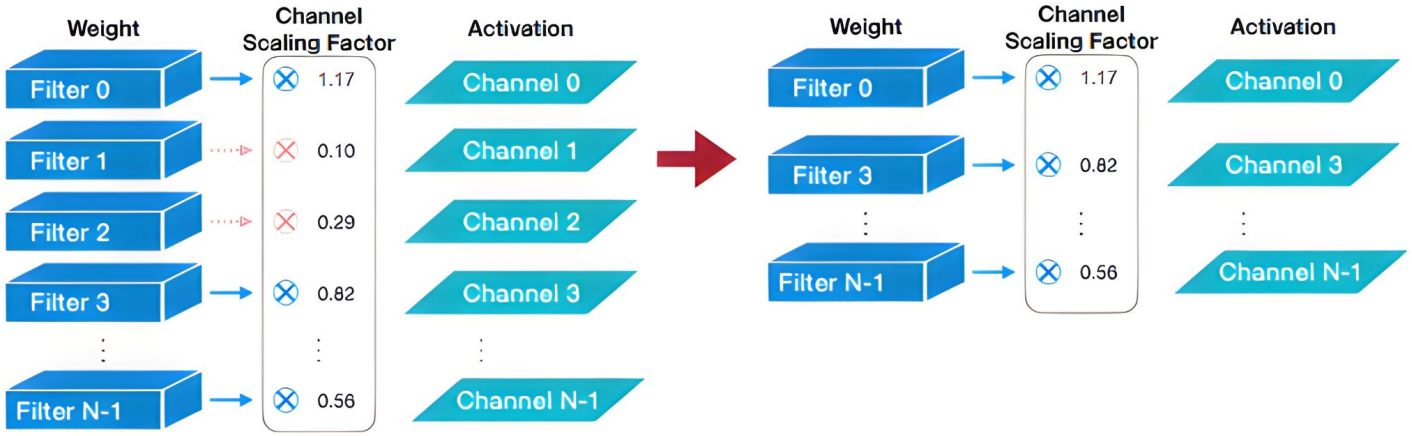
Оптимизация для увеличения общей магнитуды

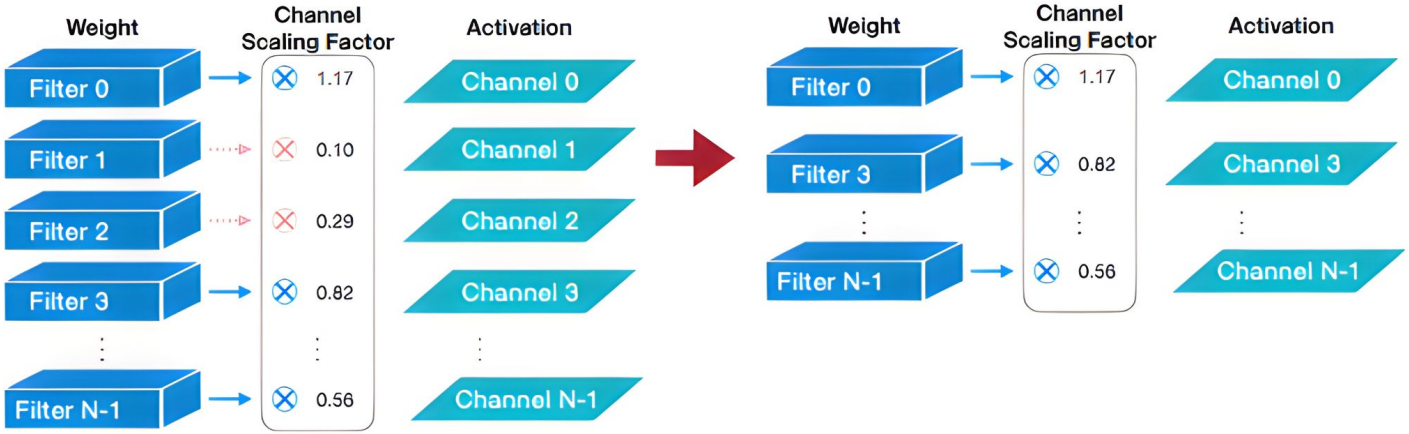
[Accelerating Sparse Deep Neural Networks]

Network	Accuracy		
	Dense FP16	Sparse FP16	Sparse INT8
ResNet-34	73.7	73.9	73.7
ResNet-50	76.1	76.2	76.2
ResNet-50 (SWSL)	81.1	80.9	80.9
ResNet-101	77.7	78.0	77.9
ResNeXt-50-32x4	77.6	77.7	77.7
ResNeXt-101-32x16	79.7	79.9	79.9
ResNeXt-101-32x16 (WSL)	84.2	84.0	84.2
DenseNet-121	75.5	75.3	75.3
DenseNet-161	78.8	78.8	78.9
Wide ResNet-50	78.5	78.6	78.5
Wide ResNet-101	78.9	79.2	79.1
Inception v3	77.1	77.1	77.1
Xception	79.2	79.2	79.2
VGG-11	70.9	70.9	70.8
VGG-16	74.0	74.1	74.1
VGG-19	75.0	75.0	75.0
SUNet-128	75.6	76.0	75.4
SUNet-7-128	76.4	76.5	76.3
DRN26	75.2	75.3	75.3
DRN-105	79.4	79.5	79.4

Accelerating Sparse Deep Neural Networks

Поканальная спарсификация

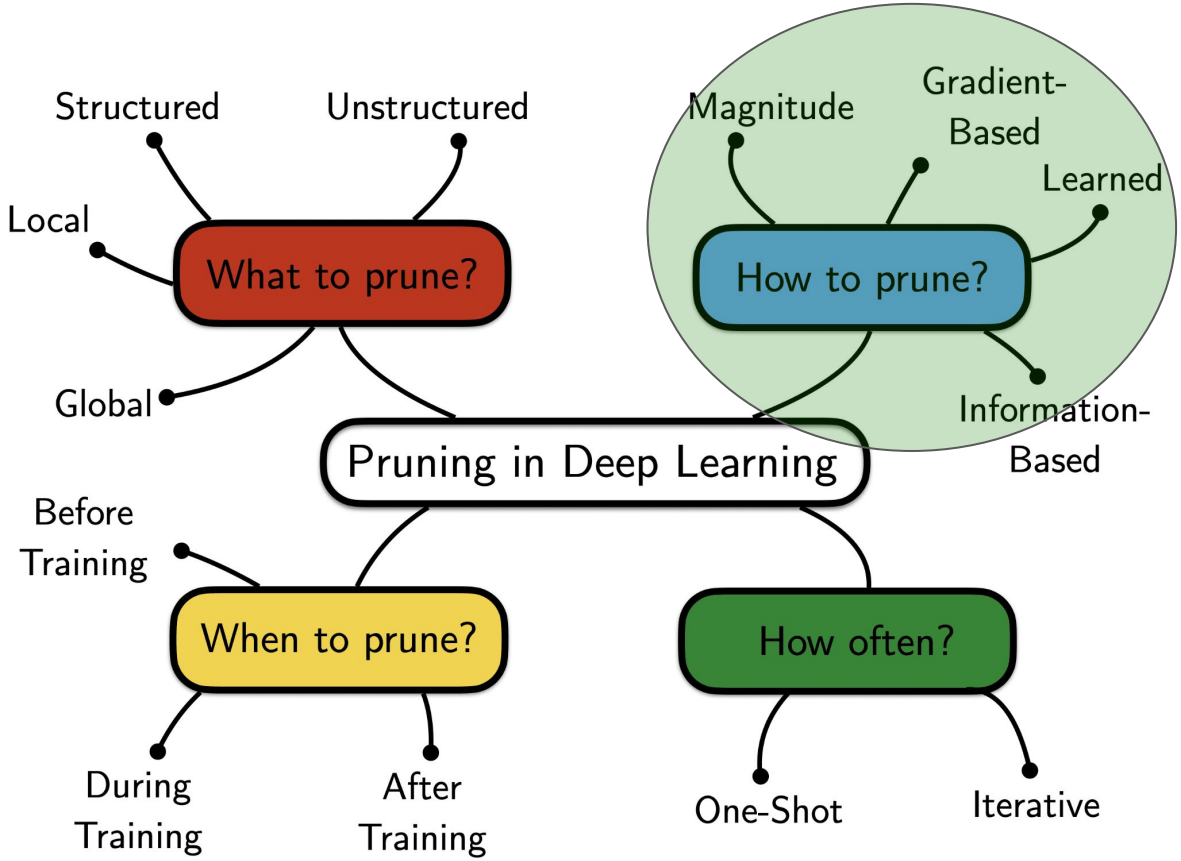




Использовать значения
BatchNorm для удаления каналов

$$\hat{x}_i = \frac{x_i - \mu_b^k}{\sqrt{(\sigma_b^k)^2 + \epsilon}}$$

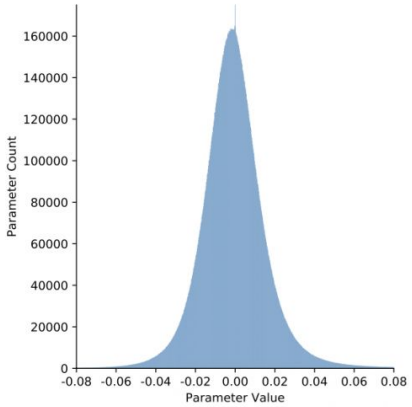
$$y = \hat{x}_i \gamma + \beta$$


Pruning in Deep Learning
What to prune?
Structured
Unstructured
Local
Global
How to prune?
Magnitude
Gradient-Based
Learned
Information-Based
When to prune?
Before Training
During Training
After Training
How often?
One-Shot
Iterative

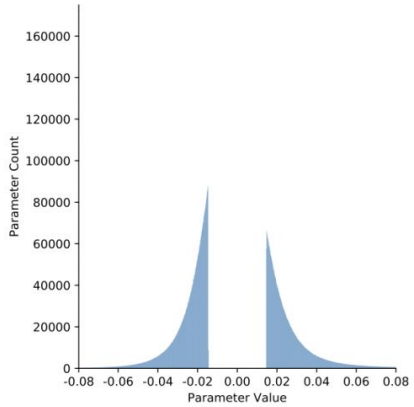
Как выбрать важность веса?

- Магнитуда весов
- На основе градиентов
- Разложение Тейлора
- ...

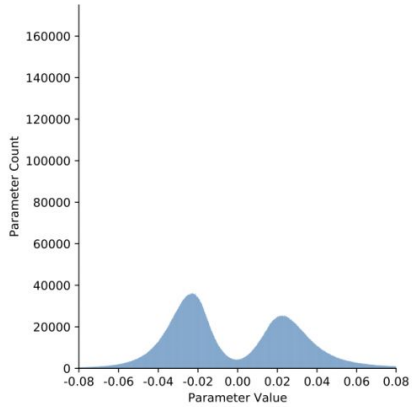
На основе магнитуды - не требует данных



(a) Dense Network (76.0%)



(b) 70% Pruned (36.1%)



(c) After 3-epoch Retraining (71.4%)

>>> Методы с дообучением с использованием данных

PRUNING CONVOLUTIONAL NEURAL NETWORKS FOR RESOURCE EFFICIENT INFERENCE

During pruning, we refine a subset of parameters which preserves the accuracy of the adapted network, $\mathcal{C}(\mathcal{D}|\mathcal{W}') \approx \mathcal{C}(\mathcal{D}|\mathcal{W})$. This corresponds to a combinatorial optimization:

$$\min_{\mathcal{W}'} \left| \mathcal{C}(\mathcal{D}|\mathcal{W}') - \mathcal{C}(\mathcal{D}|\mathcal{W}) \right| \quad \text{s.t.} \quad ||\mathcal{W}'||_0 \leq B, \tag{1}$$

where the ℓ_0 norm in $||\mathcal{W}'||_0$ bounds the number of non-zero parameters B in W'. Intuitively, if $\mathcal{W}' = \mathcal{W}$ we reach the global minimum of the error function, however $||\mathcal{W}'||_0$ will also have its maximum.

Molchanov, Pavlo, et al. "Pruning convolutional neural networks for resource efficient inference." *arXiv preprint arXiv:1611.06440* (2016).

PRUNING CONVOLUTIONAL NEURAL NETWORKS FOR RESOURCE EFFICIENT INFERENCE

$$\mathcal{C}(\mathcal{D}, h_i = 0) \quad = \quad \mathcal{C}(\mathcal{D}, h_i) - \frac{\delta \mathcal{C}}{\delta h_i} h_i + R_1(h_i = 0). \tag{5}$$

$$\Theta_{TE}(h_i) = \left|\Delta \mathcal{C}(h_i)\right| = \left|\mathcal{C}(\mathcal{D}, h_i) - \frac{\delta \mathcal{C}}{\delta h_i} h_i - \mathcal{C}(\mathcal{D}, h_i)\right| = \left|\frac{\delta \mathcal{C}}{\delta h_i} h_i\right|. \tag{7}$$

back-propagation. Θ_{TE} is computed for a multi-variate output, such as a feature map, by

$$\Theta_{TE}(z_l^{(k)}) = \left|\frac{1}{M} \sum_m \frac{\delta C}{\delta z_{l,m}^{(k)}} z_{l,m}^{(k)}\right|, \tag{8}$$

where M is length of vectorized feature map. For a minibatch with $T > 1$ examples, the criterion is computed for each example separately and averaged over T.

Molchanov, Pavlo, et al. "Pruning convolutional neural networks for resource efficient inference." *arXiv preprint arXiv:1611.06440* (2016).

>>> Методы БЕЗ дообучения с использованием данных: OBS, OBD, EBD

OBD - Optimal Brain Damage 1989 (LeCun)

OBS - Optimal Brain Surgeon 1993

EBD - Early Bird Damage 1996

Wood - Fisher 2020 (on estimates of the inverse Hessian)

M - FAC 2021 (on estimates of the inverse Hessian)

Optimal Bert Surgeon 2022

Assume that $W \in \mathcal{R}^d$ and Hessian is $W \in \mathcal{R}^{d \times d}$ The Taylor expansion of change in the training error is given by:

$$\mathcal{E}(W^*) = \mathcal{L}(W) - \mathcal{L}(W^*) = \frac{\mathcal{L}'(W^*)}{1!}(W - W^*) + \frac{\mathcal{L}''(W^*)}{1!}(W - W^*)^2 + ...$$

$$\mathcal{E}(W^*) = (\frac{\partial L}{\partial W})^T \cdot \delta W + \frac{1}{2}\delta W^T \cdot \mathcal{H} \cdot \delta W + \mathcal{O}(||\delta W||^3)$$

>> Detailed derivation and explanation can be found here

Assume that $W \in \mathcal{R}^d$ and Hessian is $W \in \mathcal{R}^{d \times d}$ The Taylor expansion of change in the training error is given by:

$$\mathcal{E}(W^*) = \mathcal{L}(W) - \mathcal{L}(W^*) = \frac{\mathcal{L}'(W^*)}{1!}(W - W^*) + \frac{\mathcal{L}''(W^*)}{1!}(W - W^*)^2 + ...$$

$$\mathcal{E}(W^*) = (\frac{\partial L}{\partial W})^T \cdot \delta W + \frac{1}{2}\delta W^T \cdot \mathcal{H} \cdot \delta W + \mathcal{O}(||\delta W||^3)$$

Given that Wvis well-optimized, it is reasonable in practice to assume that $\frac{\partial L}{\partial W^*} \approx 0$

$\delta W = W^* - W_0$ ay correspond to quantization or pruning.

For sparsification we have $\delta W = W^* - W^* \cdot M$, where M is a binary matrix.

For quantization we have $\delta W = W^* - Q(W^*)$, where Q is a quantization funciton.

Then, the change in loss incurred by pruning or quantizing a subset of weights can be expressed as

$$\mathcal{E}(W^*) \approx \frac{1}{2}\delta W^T \cdot \mathcal{H} \cdot \delta W$$

Assume that $W \in \mathcal{R}^d$ and Hessian is $W \in \mathcal{R}^{d \times d}$ The Taylor expansion of change in the training error is given by:

$$\mathcal{E}(W^*) = \mathcal{L}(W) - \mathcal{L}(W^*) = \frac{\mathcal{L}'(W^*)}{1!}(W - W^*) + \frac{\mathcal{L}''(W^*)}{1!}(W - W^*)^2 + ...$$

$$\mathcal{E}(W^*) = (\frac{\partial L}{\partial W})^T \cdot \delta W + \frac{1}{2}\delta W^T \cdot \mathcal{H} \cdot \delta W + \mathcal{O}(||\delta W||^3)$$

Computing $\mathcal{H}$ would require computing $\frac{1}{2}N^2$ elements, where N a number of parameters in a model.

Let's assume that change in $\mathcal{E}$ is caused by weights perturbations individually. And so that $\mathcal{H}$ is diagonal.

$$\mathcal{E}(W^*) \approx \frac{1}{2}\delta W^T \cdot \mathcal{H} \cdot \delta W \approx \frac{1}{2}\sum_i h_{ii}\delta W_i^2 \quad (1)$$

$$h_{ij} = \frac{\partial L^2}{\partial W_i \partial W_j} \quad (2)$$

Assume that $W \in \mathcal{R}^d$ and Hessian is $W \in \mathcal{R}^{d \times d}$ The Taylor expansion of change in the training error is given by:

$$\mathcal{E}(W^*) = \mathcal{L}(W) - \mathcal{L}(W^*) = \frac{\mathcal{L}'(W^*)}{1!}(W - W^*) + \frac{\mathcal{L}''(W^*)}{1!}(W - W^*)^2 + ...$$

$$\mathcal{E}(W^*) = (\frac{\partial L}{\partial W})^T \cdot \delta W + \frac{1}{2}\delta W^T \cdot \mathcal{H} \cdot \delta W + \mathcal{O}(||\delta W||^3)$$

Assume that $W \in \mathcal{R}^d$ and Hessian is $W \in \mathcal{R}^{d \times d}$ The Taylor expansion of change in the training error is given by:

$$\mathcal{E}(W^*) = \mathcal{L}(W) - \mathcal{L}(W^*) = \frac{\mathcal{L}'(W^*)}{1!}(W - W^*) + \frac{\mathcal{L}''(W^*)}{1!}(W - W^*)^2 + ...$$

$$\mathcal{E}(W^*) = (\frac{\partial L}{\partial W})^T \cdot \delta W + \frac{1}{2}\delta W^T \cdot \mathcal{H} \cdot \delta W + \mathcal{O}(||\delta W||^3)$$

Таким образом, мы можем выбрать веса для удаления, которые минимизируют изменение функции потерь:

$$\mathcal{E}(W^*) \approx \frac{1}{2}\delta W^T \cdot \mathcal{H} \cdot \delta W$$

Может использоваться для прунинга, квантования или любых других изменений весов.

Optimal Brain Damage - Algorithm:

- For each network weight W_i, determine its corresponding $h_{ii} = \frac{\partial L^2}{\partial W_i \partial W_i}$
- For each h_{ii} compute its saliency $\frac{1}{2} h_{ii} w_i^2$
- Sort all values by its saliency scores and delete or quantize the weights

>> Detailed derivation and explanation can be found here

Optimal Brain Surgeon

Find such δW that one of the weights becomes zero or such that one of the weights becomes quantize. Assume e_i is a unit vector for with ith element being modified.

- Pruning: $e_i^T \times \delta W + W_i = 0$
- Quantization: $e_i^T \times \delta W + (W_i - Q(W_i) = 0$

Then we need to optimize $\mathcal{E}$ with one of the constrains above, we consider pruning:

$$\min_{i \in W} \Big\{ \min_{\delta W} \big(\frac{1}{2} \delta W^T \cdot \mathcal{H} \cdot \delta W \big) | e_i^T \cdot \delta W + W_i = 0 \Big\}$$

$$\min_{i \in W} \left\{ \min_{\delta W} \left(\frac{1}{2} \delta W^T \cdot \mathcal{H} \cdot \delta W \right) | e_i^T \cdot \delta W + W_i = 0 \right\}$$

For that, we use Lagrange multlipliers:

$$L = \frac{1}{2} \delta W^T \cdot \mathcal{H} \cdot \delta W + \lambda (e_i^T \cdot \delta W + W_i)$$

$$\mathcal{H} \delta W + \lambda e_i^T = 0$$

$$e_i^T \cdot \delta W + W_i = 0$$

$$\lambda e_i^T \mathbf{H}^{-1} e_i^T = W_i$$

$$\lambda = \frac{W_i}{[\mathcal{H}^{-1}]_{ii}}$$

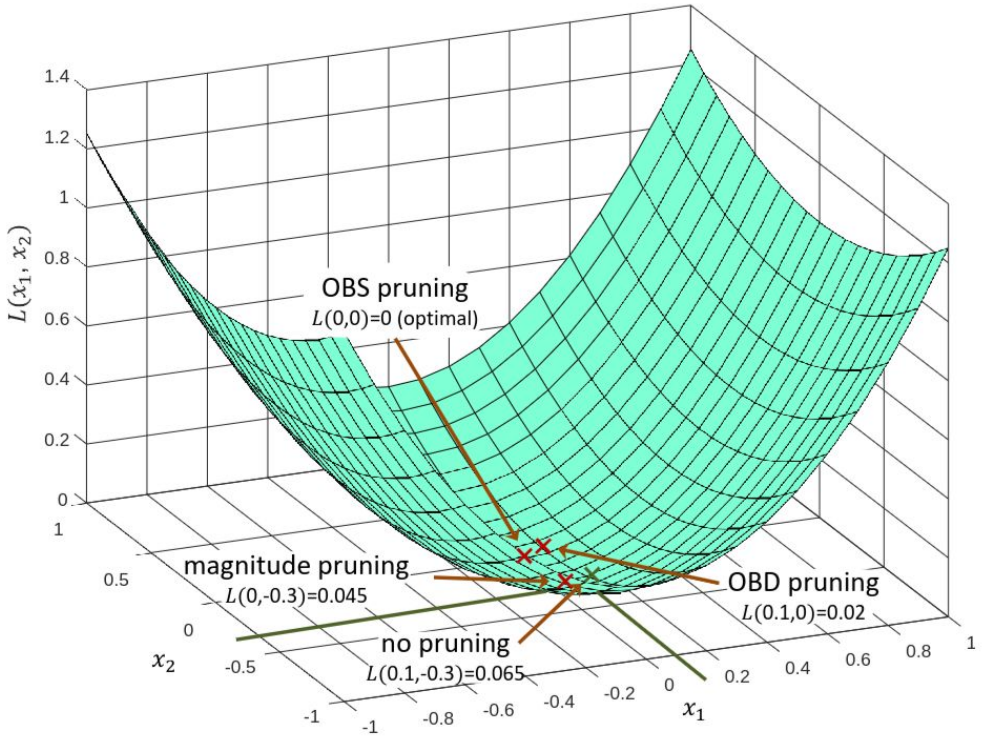
$$\delta W = -\frac{W_i}{[\mathcal{H}^{-1}]_{ii}} \mathcal{H}^{-1} \cdot e_i$$

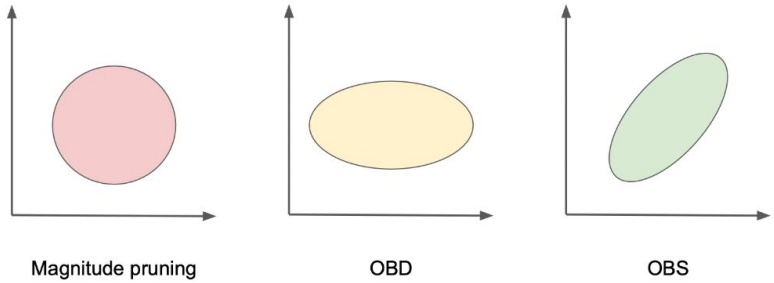
Как должны измениться другие веса, если мы удалим один вес.

$$\mathcal{E} = \frac{1}{2} \frac{W_i^2}{[\mathcal{H}^{-1}]_{ii}}$$

Optimal Brain Surgeon - Algorithm:

- Compute H^{-1}
- Fin weights that has the smallest importance $\mathcal{E} = \frac{1}{2} \frac{W_i^2}{[\mathcal{H}^{-1}]_{ii}}$
- Remove or quantize these weights
- Update remaining weights $\delta W = -\frac{W_i}{[\mathcal{H}^{-1}]_{ii}} \mathcal{H}^{-1} \cdot e_i$
- Repeat


OBS pruning
$L(0,0)=0$ (optimal)
magnitude pruning
$L(0,-0.3)=0.045$
no pruning
$L(0.1,-0.3)=0.065$
OBD pruning
$L(0.1,0)=0.02$
$L(x_1, x_2)$
x_1
x_2


Magnitude pruning
OBD
OBS

OBS & GPTQ For LLM

Теперь надо адаптировать эти решения для LLM

Проблемы:

- Вычислять H очень дорого
- Слишком много параметров обновлять
- После каждого изменения весов надо обновлять H

OBS & GPTQ For LLM

Теперь надо адаптировать эти решения для LLM

Проблемы:

- Вычислять H очень дорого
- Слишком много параметров обновлять
- После каждого изменения весов надо обновлять H

Algorithm 1 Prune $k \leq d_{\text{col}}$ weights from row $\mathbf{w}$ with inverse Hessian $\mathbf{H}^{-1} = (2\mathbf{X}\mathbf{X}^\top)^{-1}$ according to OBS in $O(k \cdot d_{\text{col}}^2)$ time.

$M = \{1, \ldots, d_{\text{col}}\}$
for $i = 1, \ldots, k$ **do**
  $p \leftarrow \text{argmin}_{p \in M} \frac{1}{[\mathbf{H}^{-1}]_{pp}} \cdot w_p^2$
  $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{H}^{-1}_{:,p} \frac{1}{[\mathbf{H}^{-1}]_{pp}} \cdot w_p$
  $\mathbf{H}^{-1} \leftarrow \mathbf{H}^{-1} - \frac{1}{[\mathbf{H}^{-1}]_{pp}} \mathbf{H}^{-1}_{:,p} \mathbf{H}^{-1}_{p,:}$
  $M \leftarrow M - \{p\}$
end for

Решения:

- Рассмотрим проблему послойно и разложем MSE между выходом обычного и обрезанного слоя
- Обновляем веса построчно

$$\mathcal{L} = ||(WX^T - \hat{W}X^T)||^2$$

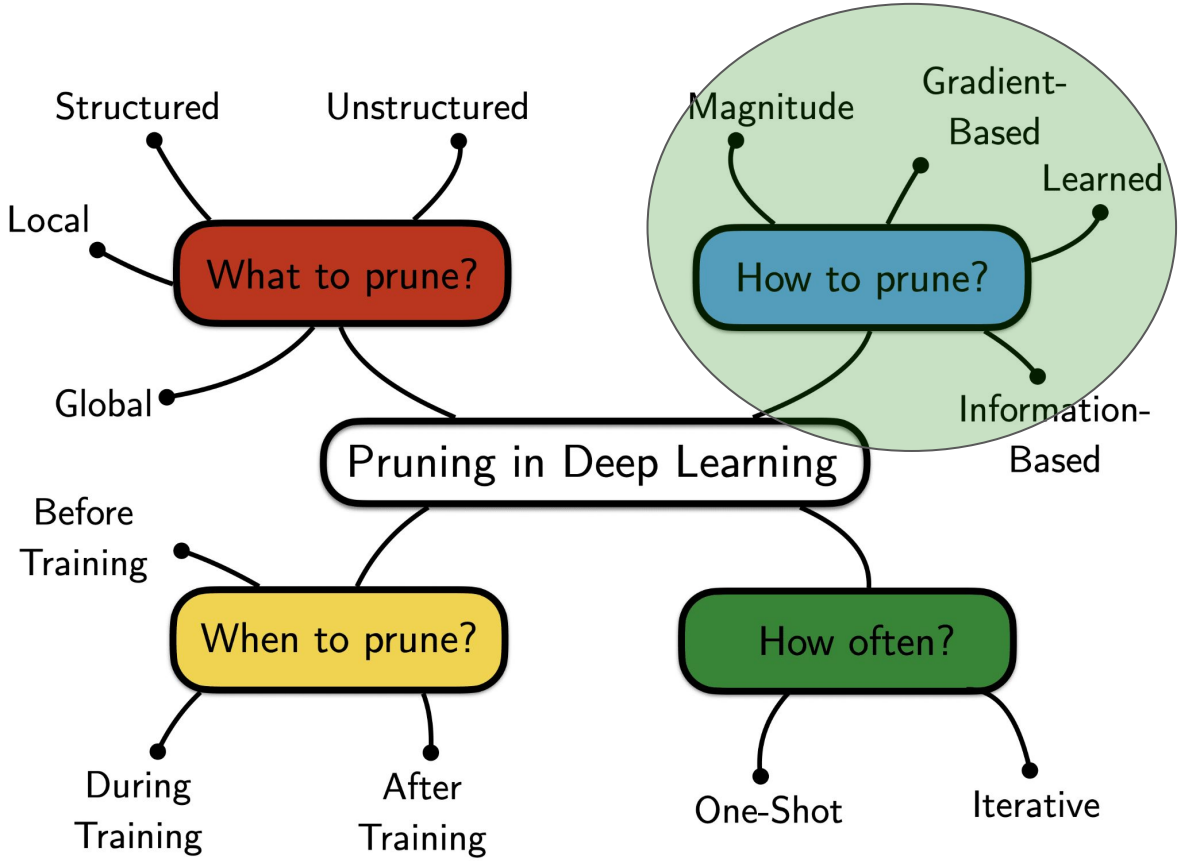
$$\mathcal{H} = \frac{\partial^2 \mathcal{L}}{\partial^2 W} = \frac{\partial^2}{\partial^2 W} ||(WX^T - \hat{W}X^T)||^2 = XX^T$$

Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning

OBS & GPTQ For LLM

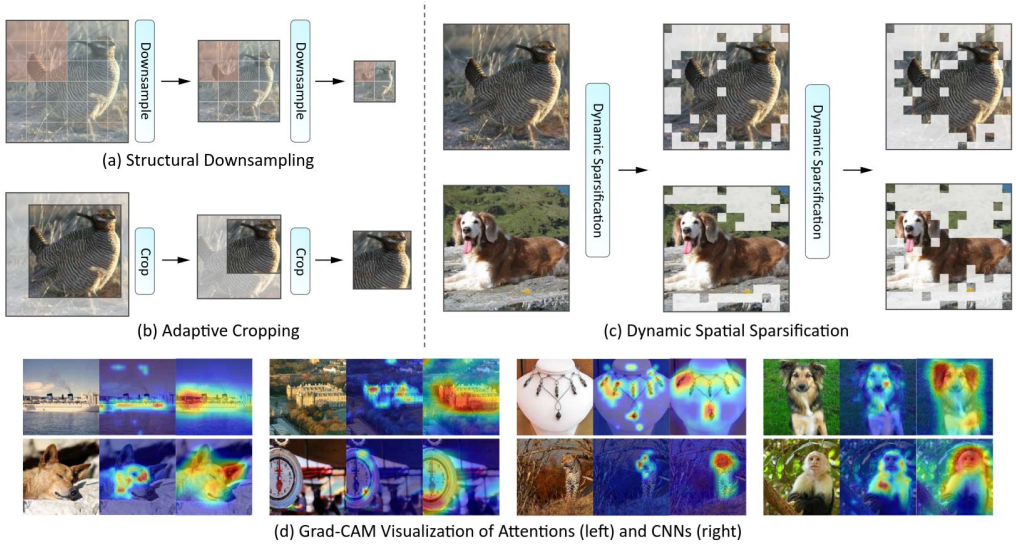
Method	ResNet50 – 76.13			YOLOv5l – 66.97			BERT – 88.53		
	2×	3×	4×	2×	3×	4×	2×	3×	4×
GMP	74.86	71.44	64.84	65.83	62.30	55.09	65.64	12.52	09.23
L-OBS	75.48	73.73	71.24	**66.21**	64.47	61.15	77.67	3.62	6.63
AdaPrune	75.53	74.47	72.39	66.00	64.88	62.71	87.12	70.32	18.75
ExactOBS	**75.64**	**75.01**	**74.05**	66.14	**65.35**	**64.05**	**87.81**	**85.87**	**82.10**

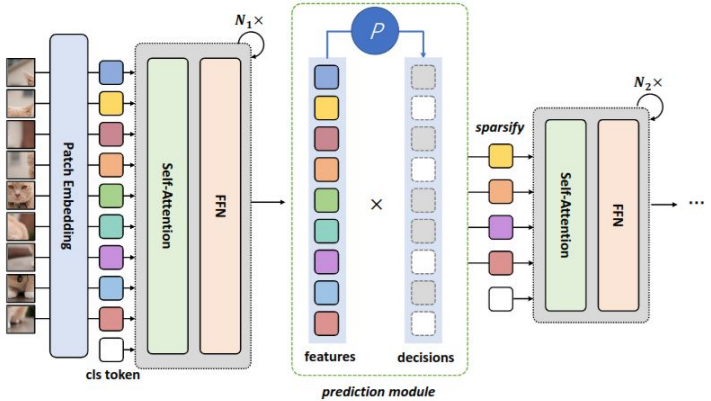
Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning


Structured
Unstructured
Local
What to prune?
Global
Magnitude
Gradient-Based
Learned
How to prune?
Information-Based
Pruning in Deep Learning
Before Training
When to prune?
During Training
After Training
How often?
One-Shot
Iterative

>>> Динамическая спарсификация

Динамическая спарсификация





Dynamic Spatial Sparsification for Efficient Vision Transformers and Convolutional Neural Networks

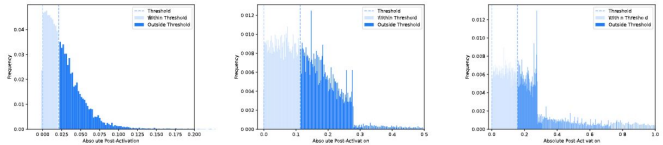
Динамическая спарсификация

$$\text{Gated-MLP}(x) := (\text{SiLU}(xW_{\text{gate}}) * (xW_{\text{up}}))W_{\text{down}}$$

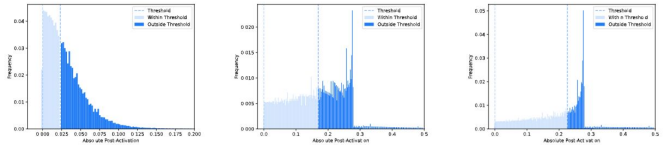
$$\text{CATS}_t(\text{SiLU}(xW_{\text{gate}})) = \begin{cases} \text{SiLU}(xW_{\text{gate}}) & |\text{SiLU}(xW_{\text{gate}})| \geq t \\ 0 & |\text{SiLU}(xW_{\text{gate}})| < t \end{cases}$$

Custom GPU Kernel 1 MLP using CATS

1: **Input:** threshold $t > 0$, hidden layer x, weights W_{gate}, W_{down}, and W_{up}
2: $v \leftarrow \text{SiLU}(xW_{gate})$
3: $\texttt{Mask} \leftarrow 1$ if $|v| \geq t$ else 0
4: $x_1 \leftarrow (xW_{\text{up}}[\texttt{Mask}] * v[\texttt{Mask}])$
5: $y \leftarrow x_1 W_{\text{down}}[\texttt{Mask}]$



(a) Llama2 Layer 0. (b) Llama2 Layer 15. (c) Llama2 Layer 31.



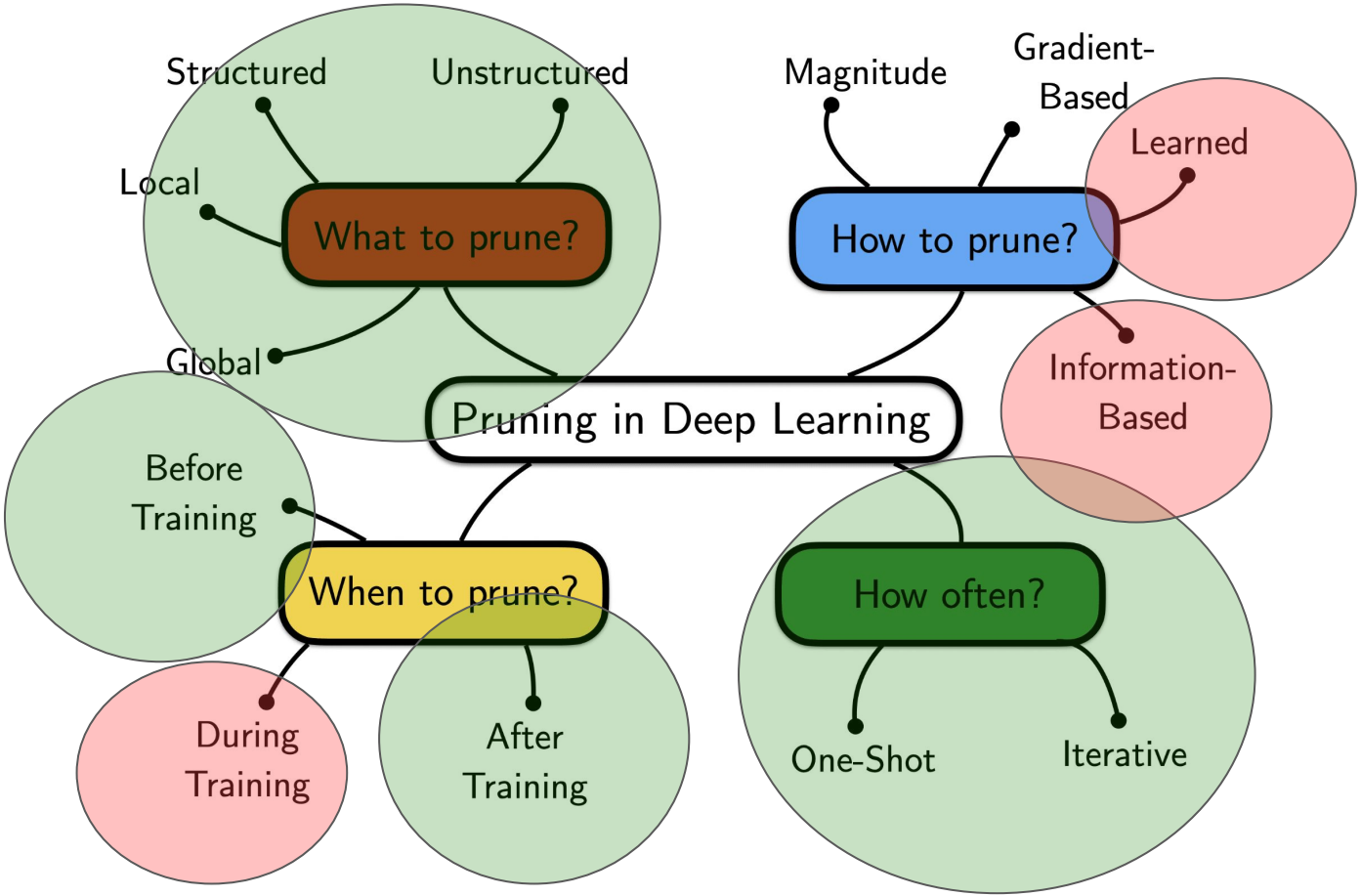
(d) Mistral7B Layer 0. (e) Mistral7B Layer 15. (f) Mistral7B Layer 31.

CATS: Contextually-Aware Thresholding for Sparsity in LargeLanguage Models

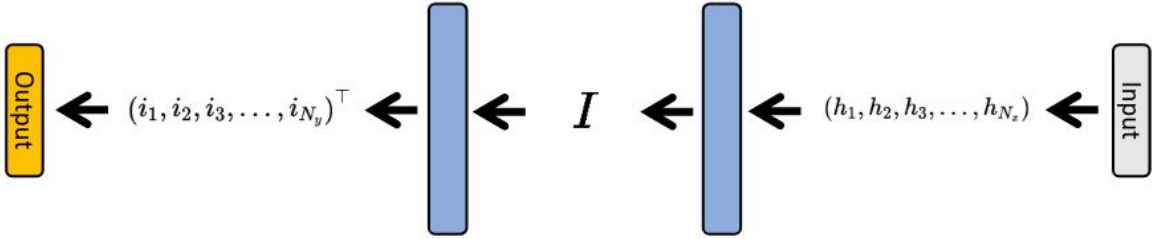
Динамическая спарсификация

Model ╲ Dataset	WG acc↑	PIQA acc↑	SciQ acc↑	QA acc↑	HS acc↑	BoolQ acc↑	Arc-E acc↑	Arc-C acc↑	Avg acc↑
Mistral-7B	0.7419	0.8069	0.959	0.3260	0.6128	0.8370	0.8085	0.5034	0.6994
CATS 50%	0.7245	0.8009	0.948	0.3200	0.6097	0.8193	0.7849	0.5043	0.6890
CATS 70%	0.7190	0.8003	0.929	0.292	0.6057	0.8028	0.7492	0.4693	0.6709
CATS 90%	0.5627	0.6001	0.422	0.212	0.3359	0.7086	0.3754	0.2773	0.4368
ReLUfication	0.5043	0.5092	0.236	0.142	0.2580	0.4208	0.2723	0.2415	0.3230
Llama2-7B	0.6906	0.7807	0.94	0.314	0.5715	0.7774	0.7630	0.4343	0.6589
CATS 50%	0.6748	0.7693	0.927	0.322	0.5711	0.7263	0.7441	0.4121	0.6433
CATS 70%	0.6693	0.7584	0.902	0.294	0.5500	0.6590	0.7008	0.3805	0.6143
CATS 90%	0.5738	0.6627	0.611	0.212	0.3848	0.6284	0.4566	0.2816	0.4764
ReLUfication	0.4893	0.5408	0.2570	0.154	0.2586	0.6003	0.2795	0.2406	0.3525

CATS: Contextually-Aware Thresholding for Sparsity in LargeLanguage Models

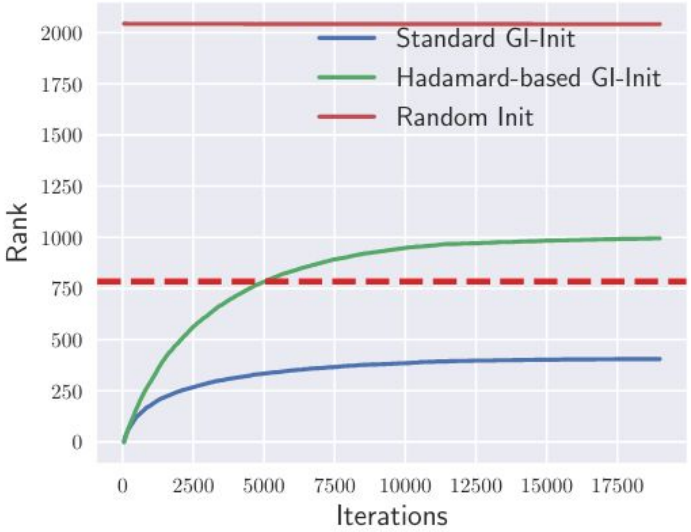

Pruning in Deep Learning
What to prune?
Structured
Unstructured
Local
Global
How to prune?
Magnitude
Gradient-Based
Learned
Information-Based
When to prune?
Before Training
During Training
After Training
How often?
One-Shot
Iterative

>>> Инициализация весов

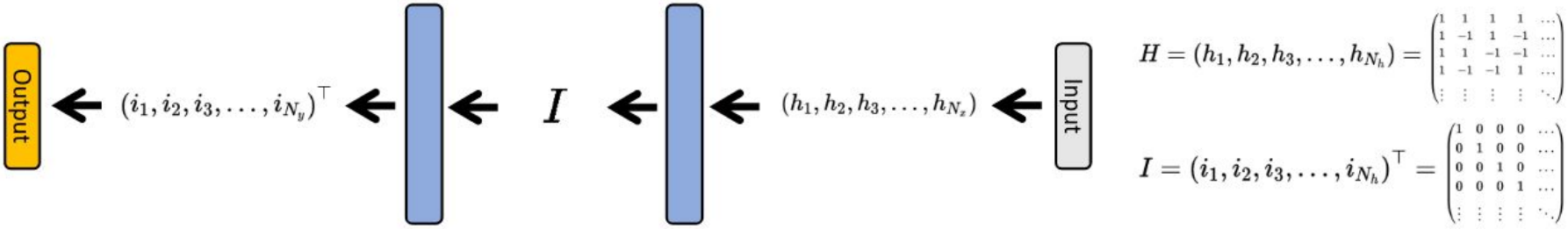


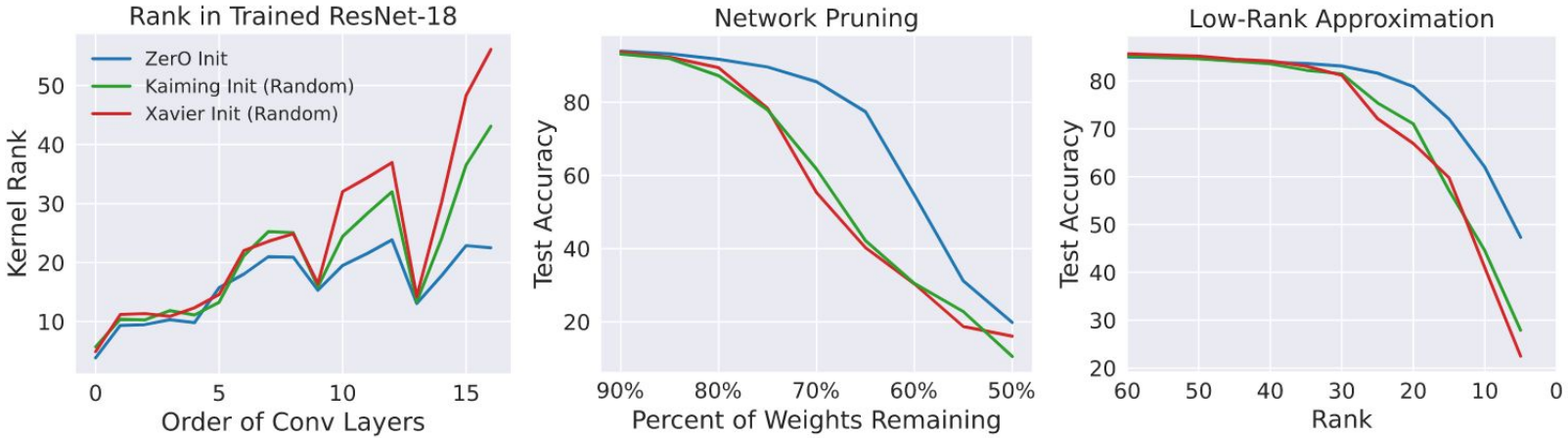
$$H = (h_1, h_2, h_3, \ldots, h_{N_h}) = \begin{pmatrix} 1 & 1 & 1 & 1 & \ldots \\ 1 & -1 & 1 & -1 & \ldots \\ 1 & 1 & -1 & -1 & \ldots \\ 1 & -1 & -1 & 1 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

$$I = (i_1, i_2, i_3, \ldots, i_{N_h})^\top = \begin{pmatrix} 1 & 0 & 0 & 0 & \ldots \\ 0 & 1 & 0 & 0 & \ldots \\ 0 & 0 & 1 & 0 & \ldots \\ 0 & 0 & 0 & 1 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$



ZerO Initialization: Initializing Neural Networks with only Zeros and Ones



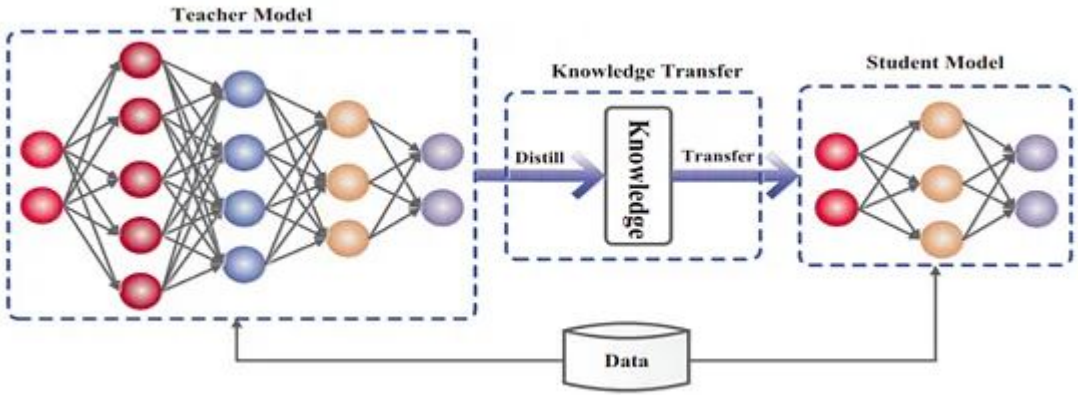


ZerO Initialization: Initializing Neural Networks with only Zeros and Ones

>>> Дистилляция

Эффективные модели тяжело учить, что мы можем использовать из того что уже имеем чтобы им помочь?

Knowledge distillation



Knowledge distillation

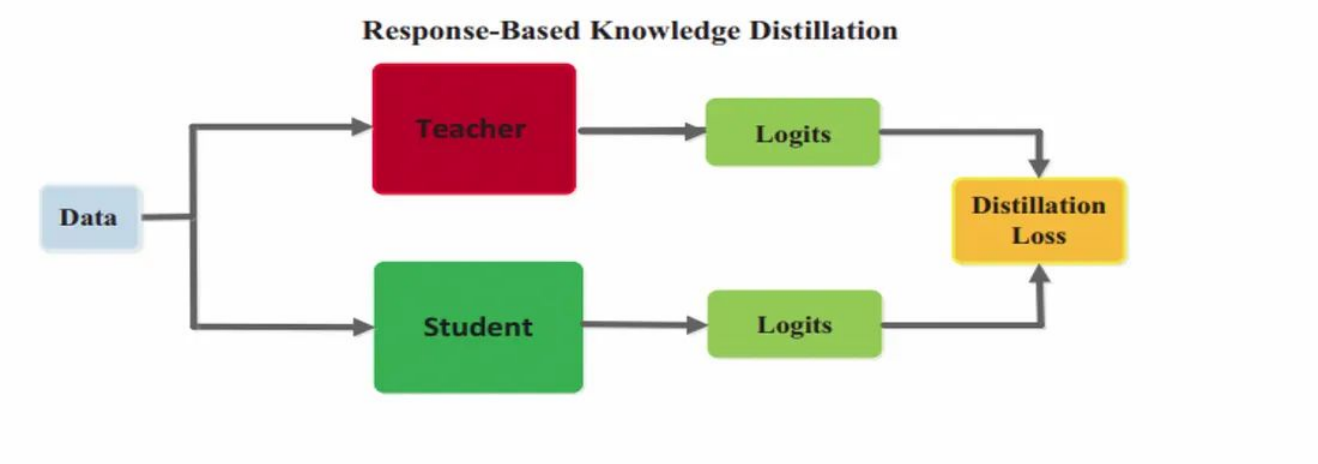
Вместо того, чтобы просто обучать маленькую/разреженную/квантизованную модель градиентным спуском на таргеты, мы будем учить ее копировать поведение большой/сложной модели - учителя.

Knowledge distillation

Используется там, где нужна эффективность:

1. Приложения на телефон
2. Автопилот для машин
3. Квантизованные модели
4. Разреженные модели

Knowledge distillation



Knowledge distillation