Contents

A Importance-based DPP

In this section, we adapt importance-based pruning, which is traditionally popular for model parameter pruning, e.g. Han et al. (2015b); Li et al. (2018); Sun et al. (2023), into DPP.

Magnitude Pruning (MP). Magnitude pruning Li et al. (2018) drops model weights according to their magnitudes: less "important" weights with smaller magnitudes are dropped. To extend its application to pruning delta parameters we evaluate importance in terms of the magnitudes of delta parameters, represented as $\Delta S_{ij} := |\Delta W_{ij}|$.

Pruning based on both Weights and activations (WANDA). Recently, WANDA was introduced by Sun et al. (2023) and was empirically found to achieve state-of-the-art pruning performance for LLMs. It evaluates the importance of each model weight using a score metric that scales the magnitude of the weight by the Euclidean norm of the input feature, thus accounting for the input activations. We extend WANDA to encompass delta parameters by adjusting the importance score to $\Delta S_{ij} = |\Delta W_{ij}| \cdot \|\boldsymbol{x}_j\|_2$.

For their respective scores, MP and WANDA (and any other importance-sampling method adapted to DPP) maintain the top-k parameters in $\boldsymbol{\Delta_W}$ with the highest scores. Let $\mathcal{S}_k \subset [m] \times [n]$ be the set of indices (i, j) that correspond to weights with the top-k scores ΔS_{ij}. Moreover, let $k := (1-p)mn$ so that the method retains the same number of delta parameters as DARE does in expectation. Accordingly, denote the set of selected weights with respect to p as $\mathcal{S}_p := \mathcal{S}_k$. Then, putting in Eq. (1), the change for each output activation $i \in [m]$ can be expressed as:

$$h_i^{\text{diff}} = \sum\nolimits_{\{j:(i,j)\notin \mathcal{S}_p\}} \Delta W_{ij} x_j \,, \tag{4}$$

where the summation extends over all input dimensions j for which the (i, j) entry of $\boldsymbol{\Delta_W}$ is dropped due to a low score. An importance sampling method can perform effectively even for large $p \approx 1$ if the cumulative contributions from the summation are approximately zero out. This is guaranteed when the distribution of the entries $[\Delta W_{ij} x_i]_{j\in[n]}$ has a light tail and high peakedness. We validate this in Secs. C.5 and B and accordingly propose AdamR-L_1 fine-tuning to enhance their pruning performance (algorithm detailed in in Sec. C.7)

B Analysis on Two-layer Neural-network

Having gained analytical insights into the key factors that influence the performance of DPP methods in Sec. 3.2 and Sec. A, we now explore in detail how these factors influence DPP in a two-layer neural network. Concretely, for an input $\boldsymbol{x} \in \mathbb{R}^n$, the model output is $f(\boldsymbol{x}) = \boldsymbol{W}_o \, \mathrm{N}(\phi(\boldsymbol{W}_1 \, \mathrm{N}(\boldsymbol{x}) + \boldsymbol{b}_1)) + \boldsymbol{b}_o$. Here, N denotes layer normalization (RMSnorm Zhang & Sennrich (2019) in our case), ϕ is the ReLU activation function, $\boldsymbol{W}_o$ / $\boldsymbol{b}_o$ and $\boldsymbol{W}_1$ / $\boldsymbol{b}_1$ are the weights / biases of the output and the hidden layer respectively, and are all trainable (during both pre-training and fine-tuning). In our experiments, we pre-train the model on the CIFAR-10 dataset and use the SVNH dataset Sermanet et al. (2012) for the supervised fine-tuning task.

Influence of variance and mean statistics on DARE (Fig. 5a): Thm. 3.1 establishes how DARE's performance, approximated by the magnitude of h_i^{diff}, is influenced by the mean and variance statistics of $\{c_{ij}\}$. Specifically, smaller values are favorable for performance. To verify this, we compare performance of DARE when the model is fine-tuned using L_1/L_2 regularization with respect to the pre-trained parameters. Formally, recalling that $\boldsymbol{\Delta_\theta} = \boldsymbol{\theta}_F - \boldsymbol{\theta}_P$ denotes the delta parameter of all trainable weights, we use regularization that penalizes the following: $\|\boldsymbol{\Delta_\theta}\|_r$, $r \in \{1, 2\}$. L_2 regularization directly corresponds to a smaller total energy ($\sum_{ij} c_{ij}^2$) of the $\{c_{ij}\}$ parameters, which recall capture the degree of change in the output layer after fine-tuning. This enters the bound in Thm. 3.1 and suggests that L_2 regularization improves DARE's performance. On the other hand, L_1 regularization favors sparse $\boldsymbol{\Delta_\theta}$, thus also $\{c_{ij}\}$. Intuitively,

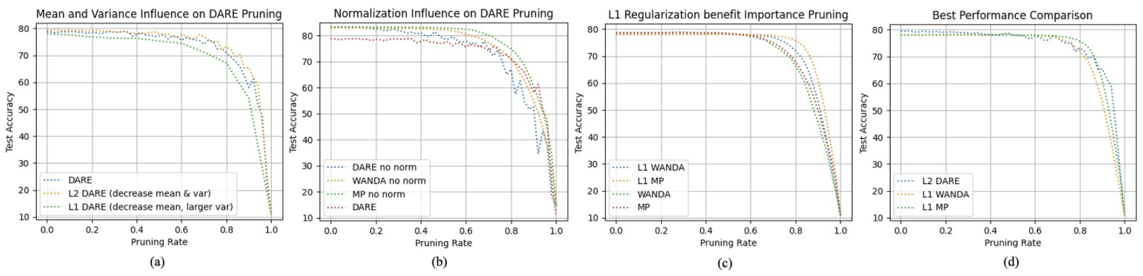

Figure 5: Controlled experiments of DPP performance on two-layer neural net. (a) Influence of variance and mean statistics on DARE. (b) Influence of normalization layer. (c) L_1 regularization for importance based pruning. (d) Methods with best-fitting regularization.

this increases the variance, thus we expect from Thm. 3.1 that L_1 regularization might hurt performance. Fig. 5 (a) shows the test accuracy of DARE for varying values of pruning rate p for three finetuning settings: no regularization (blue), L_2 regularization (orange), and L_1 regularization. The results confirm the insights above and thus support Thm. 3.1.

Influence of normalization layer (Fig. 5b): Normalization layers have become standard practice in deep learning due to their role in stabilizing training by normalizing the variance of activations, as established in Xu et al. (2019). In this study, we explore the impact of these layers on the performance of DARE. According to Thm. 3.1, reduced variance in activations (denoted as x_js in our theorem setup) leads to decreased variance in the $\{c_{ij}\}$ coefficients, which in turn is predicted to enhance DARE's effectiveness. This hypothesis is confirmed in Fig. 5 (b), where the absence of a normalization layer is shown to expedite the performance decline of DARE as pruning rates increase. Conversely, importance-based methods discussed in Sec. A demonstrate a higher tolerance to increased pruning rates without normalization. These observations are consistent with the discussions in Sec. A.

L_1 **regularization for importance based pruning (Figure 5c):** In Sec. A, we discussed how importance-based DPP methods are favored. Here, we demonstrate this is indeed the case by using L_1 regularization during finetuning to induce sparse delta parameters, which increases the variance. Concretely, as illustrated in Fig. 5c, introducing L_1 regularization improves the performance of importance-based methods. Note also that, in this scenario, magnitude pruning outperforms WANDA due to the absence of outlier features Wei et al. (2022); Sun et al. (2023).

Methods with best-fitting regularization(Fig. 5d): In Fig 5d, we compare three methods with their corresponding best-fit regularization. As discussed above, DARE benefits from L_2 regularization, while importance-based methods benefit from L_1 regularization. We find that that DARE with L_2 regularization is more robust with respect to the increase in pruning rate. On the other hand, importance based methods with L_1 regularization perform the best for medium-range pruning rates.

C Additional Details

C.1 Additional Related Work

Supervised fine-tuning of Language Models. Supervised fine-tuning (SFT) of pre-trained LLMs is designed to enhance their capabilities by training on task-specific data, establishing a standard in natural language processing Dodge et al. (2020); Zhao et al. (2023). SFT can be categorized into full fine-tuning Devlin et al. (2018); Liu et al. (2019) and parameter-efficient fine-tuning (PEFT) Ding et al. (2023); Li & Liang (2021); Hu et al. (2021); Deng et al. (2024). However, recent studiesChen et al. (2022); Lu et al. (2023);

Yao & Klimovic (2023) suggest that PEFT methods may not yet achieve the model quality of full parameter fine-tuning, particularly in high-resource tasks Chen et al. (2022). Additionally, Liu et al. (2024) indicates that most models on the Open LLM Leaderboard are derived from full parameter fine-tunes or their merges. Consequently, this paper focuses on full model fine-tuning.

Sparse finetuning. An orthogonal approach to DPP is sparse fine-tuning, which reduces the size of delta parameters by modifying the fine-tuning process itself. This is achieved through techniques such as iterative masking and fine-tuning to create sparse DP (Guo et al., 2020; Liao et al., 2023; Fu et al., 2023). In contrast, DPP methods like DARE Yu et al. (2023a) are primarily post-hoc procedures that focus on pruning the delta weights of models that have already been fine-tuned, making DPP particularly valuable for the many fine-tuned models available on platforms like Hugging Face.

Follow-up work on DARE. The recent development of the DARE (Drops delta parameters And REscales) technique has significantly advanced the efficiency of finetuned-model pruning. This method sets most delta parameters to zero, maintaining the efficacy of Supervised Fine-Tuning without loss of performance. Highlighted in Goddard et al. (2024) within Arcee's MergeKit, a toolkit for merging large LMs, DARE has shown great potential enhancing model merging processes and has various practical applications. It has been successfully implemented in projects such as Prometheus Kim et al. (2024), an open-source LM for evaluating other LMs, and in medical LMs like Biomistral Labrak et al. (2024), which develops LMs for medical applications. The technique also supports specialized domains, as seen in Akiba et al. (2024), which focuses on a Japanese LLM with advanced reasoning capabilities. These implementations highlight the broad applicability and effectiveness of DARE in enhancing model merging strategies, thus making our improvements particularly relevant and practical.

C.2 Implementation Details

For decoder LLMs, following Yu et al. (2023a), we set the temperature to 0.0 for greedy decoding and limit the maximum number of generated tokens to 1,024 on GSM8K. For encoder-based LMs, we fine-tune BERT-base-uncased and RoBERTa-base for 10 epochs using a warmup strategy and a learning rate of 1e-4. Experiments are conducted on NVIDIA A100 GPUs.

C.3 Proposed framework for DPP method selection.

We outline the procedure for selecting appropriate DPP methods in practice. As illustrated in Fig 4, if fine-tuning is not permitted and the data points (DPs) have large statistics, we recommend using WANDA for DPP (see Sections 5.3). If the DPs are not large, DARq with $1/q_v$ should be applied when a validation set is available, otherwise, DARq with $1/q_e$ is recommended (see Sec. 4.1). If the existing DPs are insufficient and fine-tuning is allowed, we suggest using AdamR-L_2 or L_1 with appropriate regularization weights to generate highly prunable DPs for DARq and MP, respectively (see Sec. C.7). Among the two, AdamR-L_2+DARq should be prefered due to flexibility and better performance as shown in Table 9 and Table 5.

C.4 Control Pruning with regularizaiton

In this section, we demonstrate the magnitude of regularization weight can control the degree of pruning that is required.

Fig 6 (a-d) illustrates the L_2 fintuned RoBERTa model with different regularization weights on SST2, MRPC, COLA and STS-B datasets respectively and we use pruning rate $p \in [0, 0.9, 0.99, 0.999]$ to prune the delta parameters. To separate different pruning rates in the figure, we set log scaled pruning $-log(1-p)/3 = [0, 0.33, 0.67, 1.00]$ in x axis. With moderate regularization the model has a close performance to models without regularizaiton (weight 0) (even better *i.e.*weight 0.01 in Fig 6 (a)). This indicate adding moderate

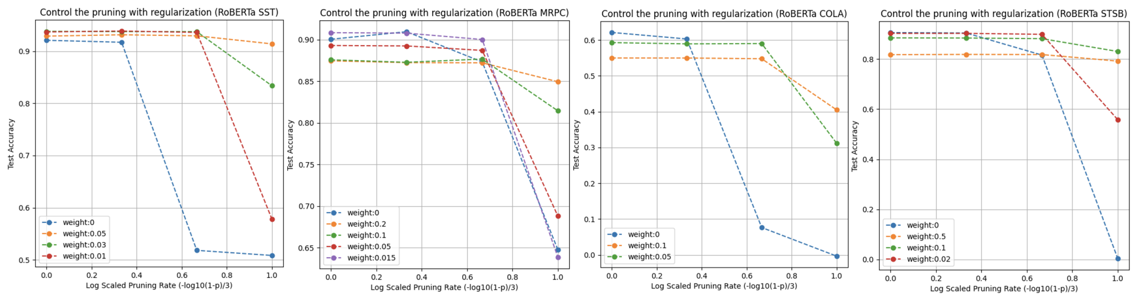

Figure 6: Control DARE pruning performance with L_2 regularizaiton on RoBERTa.(a-d) demonstrate results on SST2, MRPC, COLA, and STS-B, respectively. Aligning the degree of regularization proportionally to the pruning rate can lead to optimal performance.

regularization won't hurt the model performance. To achieve moderate pruning performance $p = 0.99$, we can choose moderate regularization. For example, in fig 6 (c), choose weight 0.05 achieves the best performance when pruning rate $p = 0.99$ and outperforms the no regularization by more than 40% . When there is a need for extreme delta parameters pruning, it is recommended to increase the regularization weight. As shown in Figure 6 (d), weight 0.1 achieves the best performance when pruning rate $p = 0.999$ and outperforms no regualrizaiton by 80 %. It is notable with L_2 regualrization, all dataset can achieve 99.9% delta parameter pruning with minor performance drop. Finally, as shown in Fig 6 (d), a strong regularization weight 0.5 although face original performance drop, but demonstrate robustness (minor performance drop) to different level of pruning rate.

Fig 7 (a-d) depicts L_2 fine-tuned BERT models with varied regularization weights on SST2, MRPC, COLA, and STS-B datasets, respectively. Pruning rates $p \in [0, 0.9, 0.99, 0.999]$ are used. Log-scaled pruning $-log(1-p)/3 = [0, 0.33, 0.67, 1.00]$ are plot to separate different rates on the x-axis. Moderate regularization yields performance close to models without regularization (weight 0), sometimes even better (e.g., weight 0.05 in Fig 7 (a)), suggesting it won't hurt model performance. For moderate pruning ($p = 0.99$), moderate regularization suffices. For instance, in Fig 7 (a), weight 0.05 achieves optimal performance, outperforming no regularization by about 40%. For extreme pruning needs, increase the regularization weight. As in Fig 7 (b), weight 0.1 achieves peak performance at $p = 0.999$, surpassing no regularization by 60%. Notably, with AdamR-L_2, all datasets can achieve 99.9% DPP with minimal performance drop. Lastly, Fig 7 (c) shows that a strong weight of 0.1, despite an initial performance drop, exhibits robustness (minor drop) across different pruning rates.

C.5 AdamR-L_1 Finetuning

In this section, we focus on MP (Han et al., 2015a) and WANDA (Sun et al., 2023), utilizing AdamR-L_1 to enhance pruning performance for importance-based methods.
MP: We demonstrate AdamR-L_1 results in Table 7: the pruning rate consistently increased on datasets when applying magnitude pruning on L_1 fintuned models. This indicates AdamR-L_1 is an effective strategy in producing highly prunbale DPs for MP.
WANDA: As shown in Table 7, AdamR-L_1 improved WANDA's performance in most datasets. However, in the SST2 and STS-B datasets (Table 7), adding AdamR-L_1 negatively impacted pruning performance, as indicated by the numbers in red. This is because WANDA considers outlier activations Dettmers et al. (2022) in LLMs and uses $\Delta S_{ij} = |\Delta W_{ij}| \cdot |x_j|$ as its metric. Thus, some important DPs may be pruned when AdamR-L_1 is applied due to their small $|x_j|$. This suggests that important DPs are not always aligned

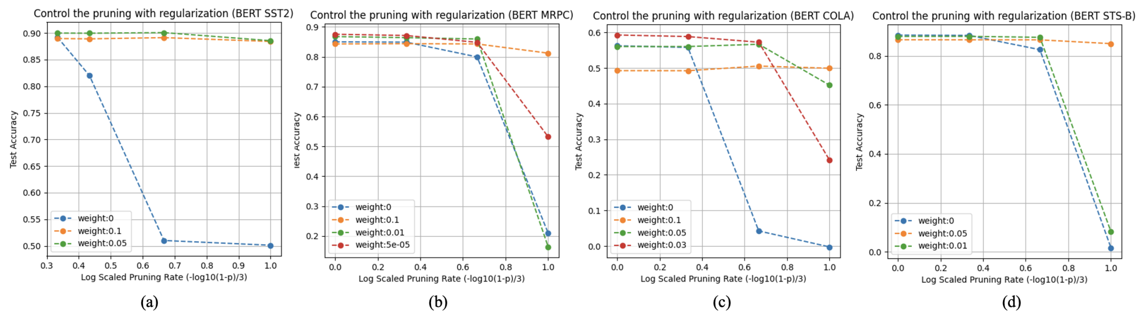

Figure 7: Control DARE pruning performance with L_2 regularization on BERT. (a-d) demonstrate results on SST2, MRPC, COLA, and STS-B, respectively. Aligning the degree of regularization proportionally to the pruning rate can lead to optimal performance.

Table 7: MP and WANDA with AdamR-L_1 on BERT and RoBERTa across various datasets

Models	SST2								COLA							
	p=0.0		p=0.90		p=0.95		p=0.99		p=0.0		p=0.90		p=0.95		p=0.99	
	original	L_1	original	L_1	original	L_1	original	L_1	original	L_1	original	L_1	original	L_1	original	L_1
BERT-MP	90.25	89.91	91.40	89.68	84.29	**89.22**	51.15	**83.14**	56.24	56.52	37.47	**47.49**	21.58	**44.59**	8.49	**12.30**
BERT-WANDA			90.51	88.53	86.93	79.93	52.64	**63.19**			37.34	**45.29**	19.24	**46.09**	8.37	**10.85**
RoBERTa-MP	92.09	92.55	90.27	**92.78**	84.86	**92.78**	68.00	**91.86**	62.10	60.16	24.88	**50.78**	6.56	**43.81**	0.00	**32.81**
RoBERTa-WANDA			92.12	91.74	92.35	89.45	76.15	72.82			34.64	**42.54**	0.07	**36.55**	0.00	**29.80**

Models	MRPC								STS-B							
	p=0.0		p=0.90		p=0.95		p=0.99		p=0.0		p=0.90		p=0.95		p=0.99	
	original	L_1	original	L_1	original	L_1	original	L_1	original	L_1	original	L_1	original	L_1	original	L_1
BERT-MP	85.01	84.35	66.14	**67.22**	15.81	**26.52**	15.81	15.81	88.51	85.12	82.79	**84.98**	42.31	**84.42**	55.80	**75.71**
BERT-WANDA			72.85	64.22	22.75	**23.49**	15.81	15.81			80.71	76.11	50.70	**70.80**	40.92	**54.21**
RoBERTa-MP	90.05	90.07	76.39	**85.17**	75.17	**78.93**	74.80	74.80	90.57	89.87	82.70	**87.43**	74.58	**86.25**	21.26	**58.53**
RoBERTa-WANDA			82.21	**84.68**	76.24	**78.86**	74.80	74.80			84.56	78.13	76.17	65.42	20.91	**25.87**

with outlier activations, and that AdamR-L_1 may not be the optimal strategy for improving WANDA. For further analysis of outlier activations, see Appendix D.

C.6 Scale of Delta Parameters

As demonstrated in Theorem 3.1, DARE is effective at pruning DPs when the mean and variance of c_{ij} are small, and it generally outperforms importance-based methods in such conditions, as shown in Table 12. While most fine-tuned LLMs typically introduce small DPs Yao & Klimovic (2023); Lee et al. (2019), and x is small, as indicated in Table 8, where $|\Delta W_{ij} x_j| \ll |\Delta W_{ij}|$, there are still instances where delta parameters are large. Specifically, Table 8 reports the magnitudes of ΔW and c_{ij} across various LLMs. For instance, the MetaMath-LLaMA-7B Yu et al. (2023b), which fine-tunes LLaMA2-7B using two large datasets—MetaMathQA Yu et al. (2023b) and Proof-Pile-2 Azerbayev et al. (2023)—yields a mean delta parameter magnitude of 0.017, significantly larger than that of Abel-7B and MetaMath-7B. As illustrated in Table 6, DARE experiences a substantial performance drop when the pruning rate exceeds 0.1.

C.7 Algorithm of AdamR

With a simple model, Sec. B demonstrated how L_1 and L_2 regularization can enhance DPP. We now show that when modifications to model finetuning are allowed, our approach can scale up to LLMs. In order to implement our regularization strategy, we propose replacing the weight decay of AdamW—the standard choice for training LLMs—with our custom regularization, as described in Sec. 5.2 and Sec. C.5. However, we do this carefully since Xie et al. (2024) found recently that weight decay can lead to large gradient norms in the final training phase, deteriorating convergence. To address this issue, following Xie et al. (2024) we

Table 8: Magnitude Statistics of ΔW_{ij} and $\Delta W_{ij} x_j$ of Llama2-7B fintuned LLMs on all dimensions and layers.

Models	Magnitude	
	$\mathrm{mean}(\lvert\Delta \mathrm{W}_{\mathrm{ij}}\rvert)$	$\mathrm{mean}(\lvert\Delta \mathrm{W}_{\mathrm{ij}}\mathrm{x}_{\mathrm{j}}\rvert)$
Abel-7B	7.3e-4 (4.3e-7)	2.89e-5 (5.41e-9)
MetaMath-7B	7.2e-4 (3.3e-7)	2.85e-5 (6.02e-9)
Wizardmath-7B	4.0e-4 (1.0e-7)	1.56e-5 (1.67e-9)
MetaMath-llema-7B	0.017 (1.9e-4)	7.0e-4 (3.02 e-6)

adjust our regularization strength based on the gradient norm. We call our new optimizer as AdamR. Note regularization here is based on the DP rather than simply on the weights of the finetuned model. The detailed algorithm, which we call AdamR, is presented in Algorithm 1.

Algorithm 1 AdamR

```
Input: Pre-trained model $\boldsymbol{\theta}_p$,
1: $\boldsymbol{g}_t = \nabla L(\boldsymbol{\theta}_{t-1})$
2: $\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$
3: $\boldsymbol{v}_t = \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2$
4: $\hat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1-\beta_1^t}$
5: $\hat{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1-\beta_2^t}$
6: $\bar{v}_t = \mathrm{mean}(\hat{\boldsymbol{v}}_t)$
7: $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta}{\sqrt{\hat{\boldsymbol{v}}_t+\epsilon}}\hat{\boldsymbol{m}}_t - \frac{\eta}{\sqrt{\bar{v}_t+\epsilon}}\lambda(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_p)$    ▷ $L_2$ regularization
8: $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta}{\sqrt{\hat{\boldsymbol{v}}_t+\epsilon}}\hat{\boldsymbol{m}}_t - \frac{\eta}{\sqrt{\bar{v}_t+\epsilon}}\lambda\mathrm{sign}(\boldsymbol{\theta}_{\mathrm{t}-1} - \boldsymbol{\theta}_{\mathrm{p}})$    ▷ $L_1$ regularization
```

Recall from Sec. B that L_2 regularization promotes a small mean and variance of delta parameters, while L_1 regularization encourages sparse updates on important parameters, leading to larger variance. We also emphasize that our regularization here is based on the DP rather than simply on the weights of the finetuned model.

C.8 ALGORITHMS OF RESCALING-PARAMETER MODIFICATION

Algorithm 2 Empirically Find q

Input: Delta Weight $\mathbf{\Delta}_\theta$, Input Dataset $\mathcal{D}_I$, Pre-trained Model θ_P, Pruning Rate p, Step Size Δq ($\Delta \eta$), Rounds N.

```
1:  initialization: $\text{error}_{min} = \text{inf}$
2:  for $t \leftarrow 1 \rightarrow N$ do
3:      if Find global $q$ then
4:          $q_t = 1 - p + t \times \Delta q$                                  ▷ $q \geq 1 - p$
5:          $\mathbf{\Delta}_\theta{}' = \mathcal{P}(\mathbf{\Delta}_\theta, q_t)$                         ▷ Prune and rescale by $q_t$
6:      else
7:          $\eta_t = t \times \Delta \eta$
8:          $\mathbf{q}_t$ = Algorithm 3($\eta_t$)                  ▷ Resolve per-layer rescaling factor.
9:          $\mathbf{\Delta}_\theta{}' = \mathcal{P}(\mathbf{\Delta}_\theta, \mathbf{q}_t)$                         ▷ Prune and rescale by $\mathbf{q}_t$
10:     end if
11:     $\theta' = \theta_P + \mathbf{\Delta}_\theta{}'$                      ▷ Add pruned delta parameters to $\theta_P$
12:     if $\mathcal{D}_I$ is Validation Dataset then
13:         $\text{error} = \mathbb{E}_{(x,y)\in D_I} \mathbb{1}(f(\theta', x) \neq y)$                 ▷ (1) Validation $q_v$
14:     else
15:         $\text{error} = \mathbb{E}_{x\in D_I} \text{abs}(f(\theta', x)^L - f(\theta_P + \mathbf{\Delta}_\theta, x)^L)$     ▷ (2) Last Layer's change $q_e$
16:     end if
17:     if $\text{error} \leq \text{error}_{min}$ then
18:         $\text{error}_{min} = \text{error}$ and $q_{best} = q_t$ ($\mathbf{q}_{best} = \mathbf{q}_t$)
19:     end if
20: end for
21: Return $q_{best}(\mathbf{q}_{best})$
```

Algorithm 3 Analytically calculate q

Input: Delta Weight $\mathbf{\Delta}_\theta$, Input x, Pre-trained Model θ_P, Pruning Rate p, Step Size Δq, Rounds $N \leq 1/\Delta q$, Empty q List $\mathbf{q} = [\,]$, Probability γ, Constant η:

```
1:  initialization: $x' = x$                                       ▷ Initialize layer input
2:  for $\mathbf{\Delta_W}, \mathbf{W}_P \in \mathbf{\Delta}_\theta, \theta_P$ do                      ▷ Loop each MLP layer
3:      initialization: $\text{error}_{min} = \text{inf}$
4:      for $t \leftarrow 1 \rightarrow N$ do
5:          $q_t = 1 - p + t \times \Delta q$                                  ▷ $q \geq 1 - p$
6:          $\text{error} = Eq\,(5)(\mathbf{\Delta_W}, q_t, \gamma, x', \eta)$     ▷ Evaluate the objective in Eq (5) for the given $q = q_t$
7:          if $\text{error} \leq \text{error}_{min}$ and $t \neq N$ then
8:              $\text{error}_{min} = \text{error}$ and $q_{best} = q_t$
9:          end if
10:     end for
11:     $\mathbf{q}.append(q_{best})$                               ▷ Add best $q$ for current layer
12:     $x' = f(\mathbf{\Delta_W} + \mathbf{W}_P, x)$     ▷ Calculate the output of the current layer of the fine-tuned model.
13: end for
14: Return $\mathbf{q}$
```

In this section, we propose and describe three strategies to determine the global rescaling factor. The per-layer rescaling factor follows a similar approach to the global one, with slight adjustments in determining the η and resolving $1/\mathbf{q}$ (see details in Algorithm 2 step 6.)

Empirically finding $1/q_v$ using validation data: In this approach, we use a validation dataset $\{x_v, y_v\} \in \mathcal{V}$ to empirically determine the best rescaling factor q. The goal is to solve the optimization problem $\arg\min_q \mathbb{E}_{\mathcal{V}}(f_q(x_v) = y_v)$, where f_q represents the pruned model rescaled by $1/q$. The process is as follows:

- *Optimization Process:* We iterate over a range of q values, adjusting the pruned model by rescaling with $1/q$. For each q, we measure the model's performance on the validation set by comparing the model's predictions $f_q(x_v)$ to the true labels y_v.
- *Outcome:* The q value that results in the best performance on the validation set is selected as $1/q_v$. This $1/q_v$ is expected to provide the best overall performance across all layers when evaluated against the validation set.
- *Algorithm Details:* This process is detailed in Algorithm 2 with objective (1), which outlines the specific steps taken to find $1/q_v$.

Empirically finding $1/q_e$ using output change: This method builds on the core idea from Thm 3.1, aiming to empirically determine a global rescaling factor q_e that balances the mean and variance in output embedding differences. Unlike the validation-based approach, this method does not rely on labeled data. Instead, it uses an unsupervised objective function based on the difference in the last layer's output embedding before and after pruning and rescaling. *Procedure:*

- *Data Usage:* We select a single fixed batch of data, which could be drawn from the training set or any natural dataset.
- *Inference:* The selected batch is passed through both the pruned and original models, and the output embeddings are compared.
- *Optimization:* The goal is to find the rescaling factor q_e that minimizes the difference between these embeddings, ensuring that the rescaled pruned model closely approximates the original model's output.

Efficiency: This method is highly efficient as it only requires inference on a single batch of data, avoiding the need for a full validation set. *Algorithm Details:* The specific steps for this method are outlined in Algorithm 2 with objective (2), which provides a detailed procedure for determining $1/q_e$.

C.9 Per-Layer $1/\mathbf{q}$

The challenge in naively implementing a per-layer tuning of the rescaling factor is that if q^ℓ were selected to optimize, say, the mean output change $|h_i^{\text{diff}}|$ averaged over all neurons in the ℓ-th layer, this would require L searches, which is computationally intractable. Instead, we turn to theory for a more efficient solution.

Our approach extends Thm.3.1, which bounds $|h_i^{\text{diff}}|$ for $q = 1 - p$, to obtain a high-probability bound for arbitrary q. This involves bounding both the mean (which is now non-zero) and the variance, then optimizing this bound over q for each layer. In view of Thm. 3.1, it is unsurprising that the bound depends on the first- and second-order statistics of the coefficients c_{ij}. Consequently, the optimal q can indeed vary per layer, depending on these statistics.

We need a high-probability bound that holds for all values of q, even as q becomes small (on the order of, but still larger than, $1 - p$). In more detail, following the machinery of Thm. 3.1 to bound $|h_i^{\text{diff}}|$, we apply Markov's inequality to the Laplace transform $\mathbb{E}[e^{\eta |h_i^{\text{diff}}|}]$, $\eta > 0$. After some careful calculations (see App.

E.2), for each fixed η, we can define $q^\ell(\eta)$ by solving a one-dimensional minimization problem involving η, p, and the statistics of c_{ij}^ℓ of the respective ℓ-th layer (which are measurable). Specifically, this is given as follows, for some probability of failure $\gamma \in (0, 1)$:

$$q(\eta) := \arg\min_q |\log \frac{2}{\gamma} + \eta \left(1 - \frac{1}{q}(1-p)\right) \sum_j c_{ij} + \frac{\eta^2 \Phi(p) \sum_j c_{ij}^2}{4q^2}| \tag{5}$$

In typical concentration bounds, one selects the optimal η by minimizing the upper bound obtained via Markov's inequality. However, this optimal η is inversely proportional to q^2. For the small q values we are interested in here (on the order of $1-p$), this makes Markov's inequality loose. Instead, we propose selecting η by a grid search over the mean output change of the last layer after rescaling with $q^L(\eta)$. This yields a value of $\eta = \eta_e$.

To give flexibility to the rescaling factors of other layers, allowing them to adjust based on the individual statistics of c_{ij}, while avoiding grid searches for all layers, we use the same value of $\eta = \eta_e$ and rescale each layer with $q^\ell(\eta_e)$. We denote the vector of selected per-layer rescaling factors as $1/\mathbf{q}_e = [1/q^1(\eta_e), \ldots, 1/q^L(\eta_e)]$. Additionally, by selecting η by optimizing the performance on validation set rather than the mean output change (denote this η_v), we can arrive at an alternative implementation, $1/\mathbf{q}_v = [1/q^1(\eta_v), \ldots, 1/q^L(\eta_v)]$, when labeled validation data are available.

C.10 Structural Pruning

Our random-based DARq can also be utilized for structural DPP for enhanced hardware efficiency (Shen et al., 2022; Yao & Klimovic, 2023). In this setup, we randomly select $a\%$ of the input dimensions of a MLP layer and randomly retain $b\%$ of the DPs within the selected dimensions, achieving a total structured DP pruning rate of $p = a \cdot b\%$. In Table 9 we use $a = 5, b = 20$, resulting in an equivalent pruning ratio $p = 0.99$. Our results show that DARq significantly outperforms DARE, achieving over a 30% improvement. AdamR-L_2 unlocks the potential of random-based DPP for structural pruning, improving DARE without L_2 finetuning by over 40%.

Table 9: Structural Pruning on BERT.

Dataset ($p = 0.99$)	SST2	COLA
DARE	51.00 (0.23)	6.25 (1.41)
Struct-DARq-$1/q_v$	83.40 (1.92)	39.09 (1.50)
Struct-DARq-L_2-$1/q_v$	**87.64 (0.47)**	**53.71 (1.74)**

C.11 Comparison to Sparse Finetuning

Sparse Finetuning (SFT) selects key weights by directly altering the fine-tuning process, using methods like iterative masking and fine-tuning to achieve sparse parameterization. In contrast, our DARq are primarily post-hoc techniques that prune the weights of models after they have been fine-tuned.
We compare DPP with sparse fine-tuning Guo et al. (2020), evaluating both on the BERT-Large model. Consistent with Guo et al. (2020), we assess the performance of vanilla DARE and our DARq with a rescaling factor of $1/q_v$, using a pruning rate of $p = 0.995$ (retaining only 0.5% of weights). As shown in Table 10, for MRPC and SST2, vanilla DARE performs slightly below the Diff-struct method, while DARq-q_v remains competitive with Diff-struct and surpasses Diff. For STS-B and COLA, vanilla DARE underperforms, but our DARq-q_v restores performance, making it competitive with Diff-struct and superior to Diff. Moreover, a key advantage of our DARq is that it is a post-hoc method, allowing for easy application to pre-fine-tuned models, and is simpler to implement.

Next, we conduct an additional experiment to examine how our proposed method—combining the post-hoc DARE with our in-training modification, AdamR-L_2—compares to sparse fine-tuning. As shown in Table 11,

Table 10: Comparison Post-hoc with SFT

$p = 0.995$	MRPC	STS-B	COLA	SST2
Full-tune	91.0	86.9	61.2	93.2
Diff	87.0	83.5	60.5	92.5
Diff-struct	89.7	**86.0**	**61.1**	**93.1**
DARE	89.6	0.00	53.2	90.1
DARq-$1/q_v$	**89.9**	84.2	60.1	92.2

Table 11: Comparison AdamR with SFT

$p = 0.999$	MRPC	STS-B
Full-tune	91.0	86.9
Diff	86.2	82.9
Diff-struct	88.2	85.2
DARE	70.2	0.00
AdamR-L_2 (ours)	**88.5**	**86.3**

Table 12: Similar to Table 2, **DARq consistently provides significant gains for decoder models**.

Dataset	GSM8K							
Methods	Abel-7B		MetaMath-7B		WizardMath-7B		MetaMath-Qwen2-0.5B	
	$p = 0.95$	$p = 0.99$	$p = 0.95$	$p = 0.99$	$p = 0.95$	$p = 0.99$	$p = 0.95$	$p = 0.99$
No Pruning	58.32		65.50		54.90		42.00	
WANDA	18.20	0.00	23.58	0.00	21.50	0.00	0.00	0.00
MP	15.16	0.00	22.82	0.00	12.00	14.50	0.00	0.00
DARE	37.30	0.00	58.22	0.00	47.10	0.00	0.00	0.00
DARq-$1/q_v$	**47.20**	20.20	59.05	**34.87**	49.81	**35.63**	**30.70**	**19.17**
DARq-$1/\mathbf{q}_v$	44.50	20.00	59.28	32.06	**50.64**	34.34	29.56	18.65
DARq-$1/q_e$	42.99	**21.30**	**60.34**	32.60	49.05	33.97	30.32	18.95
DARq-$1/\mathbf{q}_e$	42.84	19.63	59.28	28.05	**50.64**	33.58	28.80	17.66

for a pruning rate of $p = 0.999$, AdamR-L_2+DARE outperforms the baseline across both datasets. It's important to note that, similar to sparse fine-tuning, AdamR-L_2+DARE modifies the fine-tuning process. However, AdamR-L_2 only replaces the original AdamW optimizer while keeping the same task loss objective, making it a simpler implementation compared to methods like Diff, which require additional modifications.

C.12 Random-based Methods Generally Outperforms Importance-Based DPP

When DPs are not large, we show that random-based methods consistently outperform importance-based methods across all cases on decoder models, as presented in Table 12, with our DARq achieving more than a 30% improvement.

C.13 Broader impact

As discussed in Section 1 and demonstrated in prior work, DPP offers potential advantages for online serving, gradient communication in Federated Learning, and storage efficiency. Our systematic study of DPP methods and the proposed strategies that improve performance of existing methods can thus positively impact the usage of LLMs. We analyzed the impact of our approach on the efficiency of the machine learning system.

As stated in the introduction, a high pruning rate can benefit online serving, gradient communication in Federated Learning, and storage saving. To showcase this effectiveness, we compare the delta parameter loading time, the number of communication parameters, and the storage size at various pruning rates, as detailed in Table 13. Firstly, we found that the time to load the sparse compressed model decreases significantly as the pruning rate increases. In Federated Learning, only $p\%$ of the parameters need to be communicated, greatly enhancing communication efficiency. Additionally, when applied to storage saving, we further reduce memory usage by converting sparse tensors to CSR format, resulting in only 1.7 MB and 7.5 MB of storage with a 99.9% pruning rate for BERT and RoBERTa respectively. Consequently, pruning delta parameters can effectively enhance the efficiency of machine learning systems.

Table 13: Loading, Communication, Storage Efficiency with different pruning rate

Models	Loading Time (s)				Communication Size (# parameters)				CSR Storage Size (MB)			
	pruning rate p				pruning rate p				pruning rate p			
	No Pruning	0.9	0.99	0.999	No Pruning	0.9	0.99	0.999	No Pruning	0.9	0.99	0.999
BERT	0.143	0.120	0.035	0.023	$\approx$ 110 M	$\approx$ 11 M	$\approx$ 1.1 M	$\approx$ 0.11 M	417.7	108.7	11.4	1.7
RoBERTa	0.165	0.125	0.038	0.026	$\approx$ 125M	$\approx$ 12.5M	$\approx$ 1.25M	$\approx$ 0.125M	475.6	102.3	16.2	7.5

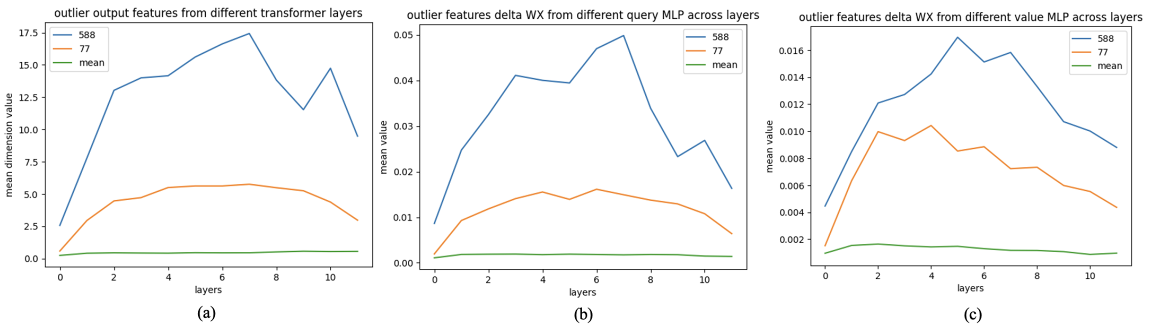

Figure 8: Analysis Outlier Features and the ΔWx with SST2 fintuned RoBERTa. Mean (green line) means average of all feature dimensions. The ΔWx is extremely small on average.

D Massive Activation and Outlier Features

Since Transformer-based models contain significantly large outliers that tend to concentrate in a few embedding dimensions Wei et al. (2022), and LLMs exhibit massive activations with high magnitude values on unique dimensions for specific tokens Sun et al. (2024), we analyze the influence of these large magnitude values in our study.

D.1 Outlier Features (Activations)

In this section, we empirically show the influence of outlier features. As shown in Table 7, WANDA outperforms MP on SST2, MRPC in original fintuned model, which indicates the validity of considering outlier features in DPP. Although outlier features will introduce larger activation inputs than normal features (Fig 8 (a)), with small delta parameter change, the mean and variance of $\Delta W_{ij} x_{ij}$ is still small (Fig 8 (b-c)) thus making DARE works well in delta parameter pruning. As a result, DARE results (Table 5 and Fig 7) outperforms that of WANDA and MP.

D.2 Massive Activation

The decoder based large language models (larger than 7-B) have massive activation Sun et al. (2024) that embrace large value on unique dimension on unique tokens (*i.e.*first token of a input). We leverage MetaMath-llema-7B Yu et al. (2023b) which finetune Llama2-7B with math datasets. We analyze each attention layer's input activations and identified the 2533-th,1415-th and 1512-th dimension as the massive activations, and shown in Fig 9 (a). It is notable 2533-th and 1415-th dimension is aligned with the pretrain model Llama2-7B's massive activation dimension Sun et al. (2024), but we have one new 1512-th dimension in MetaMath-llema-7B. When then studied the impact of layer normalization and found the massive activations is heavily downscaled by the layer normalizaiton's scale multiplier, which may fail to being considered as important by Wanda. We furtherly study the influence on removing the delta parameters that correspond to the those large manitude features by evaluating the results on GSM8K dataset. As shown in Table 14,

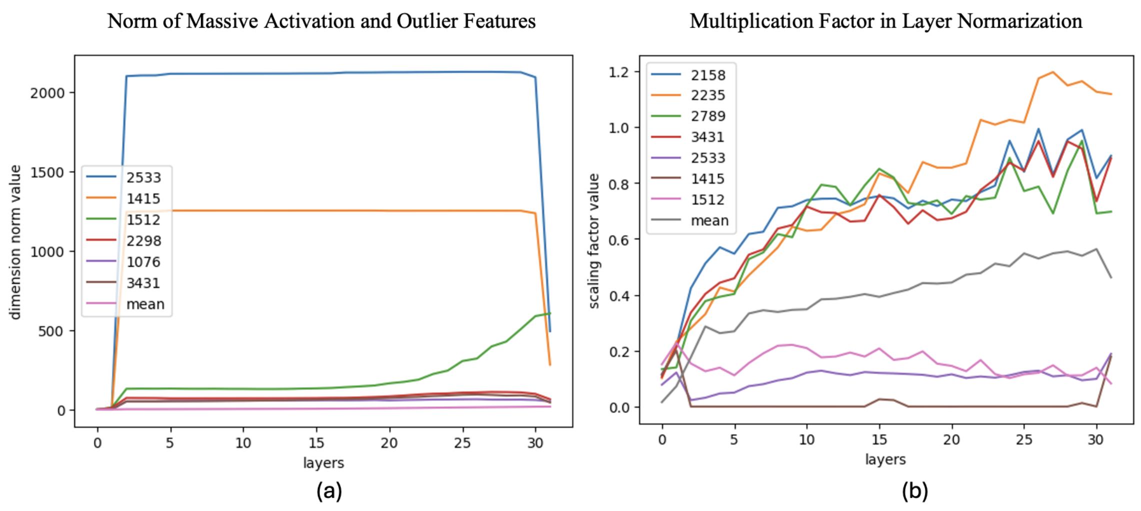

Figure 9: Analysis on Massive Activation and Outlier Features on MetaMath-llema-7B

removing the 1512-th massive activation features will bring significant performance drop. As a result, we suggest to maintain the delta parameters related to new massive activation features.

Table 14: Influnece of the delta parameters that correspond to large manitude features. Evaluation accuracy on GSM8K dataset.

ID	Prune Status	Accuracy
None	No Pruning	64.50
2533	Massive Activations	62.50
1512	Massive Activations	48.00
1415	Massive Activations	65.50
3431	Outlier Feature	61.00
2158	Outlier Feature	64.50
2350	Outlier Feature	62.50

E PROOFS

E.1 PROOF OF THEOREM 3.1

The proof employs Hoeffding's inequality for subgaussian random variables. However, a straightforward application results in suboptimal dependence of the bound on the drop-rate p. Instead, we utilize a refinement of Hoeffding's bound tailored for (generalized) Bernoulli random variables, as developed by Kearns and Saul Kearns & Saul (2013). For large values of p, we adopt the improved bound proposed by Berend and Kontorovich Berend & Kontorovich (2013). These adaptations are detailed below.

Recall from Eq. (2) that the entries of $\boldsymbol{h}^{\text{diff}}$ are given as

$$h_i^{\text{diff}} = \sum_{j=1}^{n} \Delta W_{ij} x_j - \sum_{j=1}^{n} \frac{1}{1-p} \delta_{ij} \Delta W_{ij} x_j = \sum_{j=1}^{n} \big(1 - \frac{1}{1-p}\delta_{ij}\big) \Delta W_{ij} x_j,$$

where δ_{ij} are iid $\text{Bernoulli}(1-\text{p})$ random variables. Fix any $i \in [m]$. Note h_i^{diff} is weighted sum of n iid random variables

$$A_{ij} := \big(1 - \frac{1}{1-p}\delta_{ij}\big).$$

We make the following observations about these random variables. First, a simple calculation gives that they are zero-mean, i.e. $\mathbb{E}[A_{ij}] = 0$ for all $j \in [n]$. Second, the random variables are bounded satisfying $-\frac{p}{1-p} \le A_{ij} \le 1$. Third, a simple calculations yields their variance $\mathbb{V}(A_{ij}) = p/(1-p)$.

Based on these, perhaps a first attempt in bounding $|h_i^{\text{diff}}|$ is applying Chebychev's inequality: For any $t > 0$,

$$\Pr(|h_i^{\text{diff}} - \mathbb{E}(h_i^{\text{diff}})| \ge t) \le \frac{\mathbb{V}(h_i^{\text{diff}})}{t^2}.$$

Using the mean and variance calculations above, this yields with probability at least $1-\gamma$:

$$|h_i^{\text{diff}}| \le \frac{1}{\sqrt{\gamma}} \sqrt{\frac{p}{1-p}} \sqrt{\sum_{j=1}^{n} \Delta W_{ij}^2 x_j^2}\,.$$

The drawback is of course that this scales poorly with γ.

The natural way to improve the dependence of the bound on γ is to use stronger concentration as materialized by a Chernoff bound. Since A_{ij}s are bounded, we can immediately apply Hoeffdings inequality for bounded random variables (Vershynin, 2020, Thm. 2.2.6) to find that for any $t > 0$:

$$\Pr\big(|h_i^{\text{diff}} - \mathbb{E}(h_i^{\text{diff}})| > t\big) \le 2\exp\left(-\frac{2t^2}{\frac{1}{(1-p)^2}\sum_{j\in[n]} c_{ij}^2}\right).$$

Therefore, for any $\gamma \in (0,1)$ the following holds with probability at least $1-\gamma$:

$$|h_i^{\text{diff}}| \le \left(\frac{1/\sqrt{2}}{(1-p)}\right) \sqrt{\sum_{j\in[n]} c_{ij}^2} \sqrt{\log\left(\frac{2}{\gamma}\right)}.$$

Note the significant improvement over Chebychev's bound with respect to the scaling factor γ. However, the bound remains relatively loose in terms of p for values of p near the extremes of its range. To see this note that as $p \approx 0$, for which $A_{ij} \approx 0$ and $\mathbb{V}(A_{ij}) = p/(1-p) \approx 0$. Conversely, Hoeffding's bound, which scales as $1/(1-p)$, does not decrease as p gets smaller. To get an improved bound, we can apply the Kearns-Saul ineqaulity (Kearns & Saul, 2013, Theorem 2), which is better suited for managing the characteristics of the distribution as p approaches extremal values.

Theorem E.1 (Kearns & Saul (2013)). *Let $R_j \in \{0,1\}$ be iid* $\text{Bernoulli}(1-\text{p}_\text{j})$ *for $j \in [n]$. Let also constants $\alpha_j, j \in [n]$. Then, for all $t > 0$*

$$\Pr\left(\Big|\sum_{j\in[n]} \alpha_j (R_j - (1-p_j))\Big| > t\right) \le 2\exp\left(-\frac{t^2}{\chi^2}\right), \tag{6}$$

where $\chi^2 := \sum_{j\in[n]} \alpha_j^2 \Phi(p_j)$, and

$$\Phi(x) = \frac{1-2x}{\log\left(\frac{1-x}{x}\right)}. \tag{7}$$

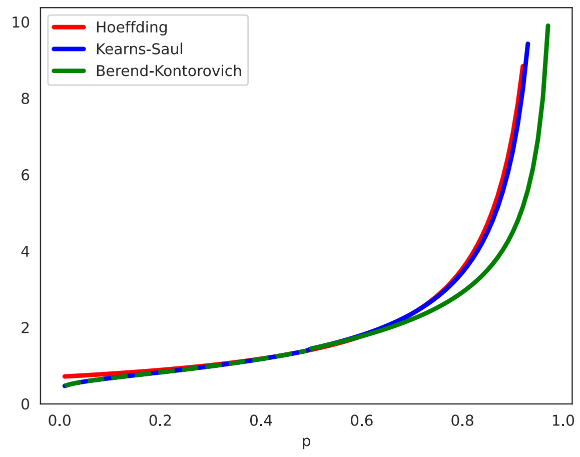

Figure 10: Visualization of the Hoeffding, Kearns-Saul and Berend-Kontorovich bounds. Note that the last is significantly improved over the vanilla Hoeffding bound for small/large values of p. Large values of p corresponding to large drop-rates are particularly interesting in the DPP setting. Note also that all three bounds diverge as $p \to 1$. This is because of the rescaling factor $1/(1-p)$. However, note that the Berend-Kontorovich bound diverges significantly slower. This further signifies the importance of using this versus the vanilla Hoeffding bound to provide an explanation on why DARE performs well even for large values of p. See proof of Thm. 3.1 for details.

Applied to our setting, let $R_j \leftarrow \delta_{ij}$ and $\alpha_j \leftarrow \frac{1}{1-p} c_{ij}$. Then, the sum on the LHS of (6) is equal to $-\sum_{j\in[n]} A_{ij}$ and $\chi^2 = \frac{\Phi(p)}{(1-p)^2}\sum_{j\in[n]} c_{ij}^2$. Put together, this gives for any $t > 0$:

$$\Pr\left(|h_i^{\text{diff}}| > t\right) \le 2\exp\left(-\frac{t^2}{\frac{1}{(1-p)^2}\Phi(p)\sum_{j\in[n]} c_{ij}^2}\right),$$

where $\Phi(p) := \frac{1-2p}{\log((1-p)/p)}$. Thus, for any $\gamma \in (0,1)$ the following holds with probability at least $1-\gamma$:

$$|h_i^{\text{diff}}| \le \left(\frac{\sqrt{\Phi(p)}}{1-p}\right)\sqrt{\sum_{j\in[n]} c_{ij}^2}\sqrt{\log\left(\frac{2}{\gamma}\right)}. \tag{8}$$

To see why this improves upon Hoeffding's bound, note that the following hold for $\Phi(p)$:

1. $\Phi(p) \le 1/2$ for all $p \in [0,1]$
2. $\Phi(0) \to 0$ as $p \to 0$ or $p \to 1$.

The first property shows that the bound is strictly better than Hoeffding's for all values of p. The second property shows that the bound approaches 0 as $p \approx 0$, which is intuitive since in that case $A_{ij} \approx 0$ and $\mathbb{V}(A_{ij}) \approx 0$. On the other hand, for $p \to 1$, it is easy to show that $\sqrt{\Phi(p)}/(1-p) \to +\infty$. Hence, both Hoeffding and this improved bound predict that $|h_i^{\text{diff}}|$ can be large as $p \to 1$.
It turns out that the Kearns-Saul bound can be further improved for $p \ge 1/2$. Such an improvement is

important in our setting, since large drop-rates $p \gg 1/2$ are of particular interest in DPP. The improved inequality is due to (Berend & Kontorovich, 2013, Lemma 5). When applied to our setting, it improves the first term in to $\sqrt{\frac{2p}{p-1}}$.
To arrive now at the advertised statement of the theorem simply rewrite $\sum_{j\in[n]} c_{ij}^2$ in terms of the mean and variance of the influence statistics

$$\frac{1}{n}\sum_{j=1}^{n} \Delta W_{ij}^2 x_j^2 = \frac{1}{n}\sum_{j=1}^{n} c_{ij}^2 = \bar{c}_i^2 + \sigma_i^2.$$

This completes the proof of the theorem.

For completeness, we visualize the three bounds that we discussed above in Figure 10. The lower the better, thus note how the Kearns-Saul bound improves over naive Hoeffding and the Berend-Kontorovich bound improves over both for $p > 1/2$.

E.2 Finding a rescaling factor balancing mean and variance

This section complements Sec. C.9 that describes the per-layer version of DARq. Specifically, we show how to derive Eq. (5).

We study a more general version of DARE that rescales with a parameter q that is not necessarily set to $1/(1-p)$. In this case, we have $A_{ij} = \left(1 - \frac{1}{q}\delta_{ij}\right) c_{ij}$, which we can rewrite as follows:

$$A_{ij} = \underbrace{\left(1 - \frac{1}{q}(1-p)\right) c_{ij}}_{=:\mu_{ij}} - \underbrace{\frac{1}{q} c_{ij}\left(\delta_{ij} - (1-p)\right)}_{=:R'_j}.$$

Following Kearns & Saul (2013), for any random variable X, it holds that $\Pr(X > 0) = \frac{1}{2}\mathbb{E}\left[1 + \frac{X}{|X|}\right]$. Observe that $\frac{1}{2}(1 + \frac{X}{|X|}) \leq e^{\eta X}$ for any positive number $\eta > 0, where \eta$ is a small positive number to ensure tightness of Eq. (9). Then, we obtain (alternatively, but equivalently, we can apply Markov's inequality to the Laplace transform):

$$Pr[\sum_j A_{ij} > \epsilon] \leq \mathbb{E}\left[e^{\eta \sum_j \mu_{ij} - \frac{1}{q} c_{ij}(\delta_{ij} - (1-p)) - \eta\epsilon}\right] \tag{9}$$

$$= e^{-\eta\epsilon + \eta\sum_j \mu_{ij}} \mathbb{E}\left[\prod_j e^{-\frac{\eta}{q} c_{ij}(\delta_{ij}-(1-p))}\right] = e^{-\eta\epsilon+\eta\sum_j \mu_{ij}} \prod_j \mathbb{E}\left[e^{-\frac{\eta}{q} c_{ij}(\delta_{ij}-(1-p))}\right]$$

$$= e^{-\eta\epsilon + \eta\sum_j \mu_{ij}} \prod_j (p e^{\frac{\eta}{q} c_{ij}(1-p)} + (1-p) e^{-\frac{\eta}{q} c_{ij} p})$$

$$\leq e^{\eta\left(\sum_j \mu_{ij} - \epsilon\right) + \frac{\eta^2}{4}\sum_j \frac{c_{ij}^2}{q^2}\Phi(p)}, \text{(Kearns \& Saul, 2013, Lemma 1)}. \tag{10}$$

Since $\frac{c_{ij}^2}{q^2}\Phi(p) \geq 0$, choosing the optimal $\eta = \frac{-2\left(\sum_j \mu_{ij} - \epsilon\right)}{\frac{\Phi(p)}{q^2}\sum_j c_{ij}^2}$ minimizes Eq. (10). However, this may result in a large η especially for small q that is of interest to us. When $\eta = \frac{-2\left(\sum_j \mu_{ij} - \epsilon\right)}{\frac{\Phi(p)}{q^2}\sum_j c_{ij}^2}$, for all $\epsilon > 0$, we have:

$$Pr[|\sum_j A_{ij}| > \epsilon] \leq 2e^{-\frac{q^2\left(\sum_j \mu_{ij} - \epsilon\right)^2}{\Phi(p)\sum_j c_{ij}^2}}. \tag{11}$$

Since $\mu_{ij} := \left(1 - \frac{1}{q}(1-p)\right) c_{ij}$, then with probability $1-\gamma$, we have:

$$\epsilon(q) = \begin{cases} \left(1 - \frac{1}{q}(1-p)\right)\sum_j c_{ij} - \frac{1}{q}\sqrt{\log(\frac{2}{\gamma})\sum_j c_{ij}^2 \Phi(p)}, & \text{if } \left(1 - \frac{1}{q}(1-p)\right)\sum_j c_{ij} - \epsilon > 0 \\ \left(1 - \frac{1}{q}(1-p)\right)\sum_j c_{ij} + \frac{1}{q}\sqrt{\log(\frac{2}{\gamma})\Phi(p)\sum_j c_{ij}^2}, & \text{if } \left(1 - \frac{1}{q}(1-p)\right)\sum_j c_{ij} - \epsilon \leq 0 \end{cases}$$

Since $\eta \geq 0$, the second condition is always satisfied. Therefore,

- If $\frac{1}{q}\sqrt{\log\left(\frac{2}{\gamma}\right)\sum_j c_{ij}^2\Phi(p)} = 0$, then $1-p$ is simply the optimal rescaling.
- If $\frac{1}{q}\sqrt{\log\left(\frac{2}{\gamma}\right)\sum_j c_{ij}^2\Phi(p)} > 0$, we can increase q to compute the optimal scaling factor:

$$\begin{aligned} q &= \arg\min_q \left(1 - \frac{1}{q}(1-p)\right)\sum_j c_{ij} + \frac{1}{q}\sqrt{\log(\frac{2}{\gamma})\Phi(p)\sum_j c_{ij}^2} \\ &= \arg\min_q \sum_j c_{ij} + \frac{1}{q}\left(\sqrt{\log(\frac{2}{\gamma})\Phi(p)\sum_j c_{ij}^2} - (1-p)\sum_j c_{ij}\right) \end{aligned}$$

Then the min value can be obtained by taking

$$q = 1 - p - \frac{\sqrt{\log(\frac{2}{\gamma})\Phi(p)\sum_j c_{ij}^2}}{\sum_j c_{ij}}$$

If $\sum_j c_{ij} > 0$ and $\sqrt{\log\left(\frac{2}{\gamma}\right)\Phi(p)\sum_j c_{ij}^2} > (1-p)\sum_j c_{ij}$, since the optimal $q > 0$, this suggests that increasing q towards infinity is favorable. However, given $\eta = \frac{-2\left(\sum_j \mu_{ij} - \epsilon\right)}{\sum_j \frac{c_{ij}^2}{q^2}\Phi(p)}$, η also tends to infinity, causing the loosening of Eq. (9). Thus, we propose selecting a fixed $\eta_\star$ to ensure the overall consistency of the inequalities in Eq. (9) and Eq. (10). Specifically, in our implementation of DARq with per-layer tuning, η_* is computed via grid search (see Sec. C.9). Substituting $\eta = \eta_\star$ into Eq. (10), we obtain:

$$\epsilon = \log\frac{2}{\gamma} + \eta_\star\left(1 - \frac{1}{q}(1-p)\right)\sum_j c_{ij} + \frac{\eta_\star^2\Phi(p)\sum_j c_{ij}^2}{4q^2}. \tag{12}$$

Given $\eta_\star$, we can find the optimal q to minimize $|\epsilon|$ in Eq. (12), which arrives at Eq. (5).