A. Mathematical supplement

A.1. Minimizing the trainable parameters in KRAdapter

Khatri-Rao products: Our main theorem is that an arbitrary matrix W of size $m \times n$ can be approximated by the Khatri-Rao product of two matrices. In order to see how to do this we will need the column-wise vectorization operator **vec**.

Definition A.1. Let **vec** denote the column-wise vectorization operator defined as follows. Given a matrix $A = [A_1 \cdots A_n]$ of size $m \times n$, where each A_i has size $m \times 1$, we define $\mathbf{vec}(A)$ to be the matrix of size $mn \times 1$ defined by

$$\mathbf{vec}(A) = \begin{bmatrix} A_1 \\ \vdots \\ A_n \end{bmatrix} \tag{A.1}$$

Theorem A.2. *Let W be a matrix of size $m \times n$ such that $rank(W) = r$. Then there exists matrices $\overline{U}$ of size $m \times r$ and $\overline{V}$ of size $n \times r$ and a vector σ of size $r \times 1$ such that*

$$\mathbf{vec}(W) = (\overline{V} \odot \overline{U})\sigma. \tag{A.2}$$

Proof. We apply the SVD to W to obtain the decomposition $W = USV^T$. We can then write

$$W = \sum_{i=1}^{r} u_i v_i^T \sigma_i \tag{A.3}$$

where u_i is the ith column of U, v_i is the ith column of V and σ_i is the ith singular value of S. We then observe that if we vectorize (A.3) we obtain

$$\mathbf{vec}(W) = \sum_{i=1}^{r} \mathbf{vec}(u_i v_i^T)\sigma_i. \tag{A.4}$$

The we note that since u_i and v_i are column vectors we have

$$\mathbf{vec}(u_i v_i^T) = v_i \otimes u_i. \tag{A.5}$$

This gives

$$\mathbf{vec}(W) = \sum_{i=1}^{r} (v_i \otimes u_i)\sigma_i. \tag{A.6}$$

If we define $\overline{U} = [u_1 \cdots u_r]$ and $\overline{V} = [v_1 \cdots v_r]$ then by definition of the Khatri-Rao product we have

$$\mathbf{vec}(W) = (\overline{V} \cdot \overline{U})\sigma \tag{A.7}$$

where $\sigma = (\sigma_1 \cdots \sigma_r)^T$. $\square$

Theorem A.2 shows that we can use Khatri-Rao products to approximate matrices. The importance of this approximation is that if were to use Khatri-Rao products for weights of a neural model, we get a parameter efficient decomposition of the weight matrix.

In general, we can apply Khatri-Rao products to approximate weight matrices as follows: Given a pretrained base weight of shape $d_{out} \times d_{in}$ we can take two matrices U and V of shapes $k_1 \times d_{in}$ and $k_2 \times d_{in}$ respectively and consider the Khatri-Rao product $U \odot V$ of shape $k_1 k_2 \times d_{in}$ where $k_1, k_2 > 0$. In order to get the shape right we need to take $k_2 = \frac{d_{out}}{k_1}$. Then the total number of parameters will be $(k_1 + \frac{d_{out}}{k_1})d_{in}$. This is minimized when $k_1 = \sqrt{d_{out}}$ so that $k_2 = \sqrt{d_{out}}$, which follows from the following lemma.

Lemma A.3. *Let $f(x) = (\frac{m}{x} + x)n$ for $x > 0$ where $m, n > 0$. Then f has a minimum at the point $x = \sqrt{m}$.*

Proof. Differentiating we see that $f'(x) = n + -\frac{mn}{x^2}$. Setting this to zero to find critical points gives

$$n + -\frac{mn}{x^2} = 0 \Rightarrow \frac{m}{x^2} = 1 \tag{A.8}$$

which gives $x = \pm\sqrt{m}$. Since we are assuming $x > 0$ we have that $x = \sqrt{m}$ is a critical point. To understand what type of critical point this is we take the double derivative and find $f''(x) = \frac{2mn}{x^3}$. We then have that

$$f''(\sqrt{m}) = \frac{2mn}{m^{3/2}} = \frac{2n}{\sqrt{m}} > 0. \tag{A.9}$$

This tells us the critical point $x = \sqrt{m}$ is a minimum point. $\square$

If d_{out} is not a perfect square we can take $k_1 = \lfloor d_{out} \rfloor$. Then U has size $\sqrt{d_{out}} \times d_{in}$ and V has shape $\sqrt{d_{out}} \times d_{in}$. The total parameters are $2\sqrt{d_{out}}d_{in}$ which is much smaller than $d_{out}d_{in}$. We thus see that by using a Khatri-Rao product we obtain a low parameter approximation for the adaptors that have parameters in $\mathcal{O}(\sqrt{d_{out}}d_{in})$ which is much less than $\mathcal{O}(d_{out}d_{in})$ when d_{out} and d_{in} are large.

To further enhance parameter efficiency, we typically choose d_{in} to be the smaller dimension of the original weight matrix $\mathbf{W}_0$. If $d_{out} < d_{in}$, we then transpose the resulting update to be applied to $\mathbf{W}_0$.

A.2. Proving the Khatri-Rao of two random matrices is full rank

We can compare the construction of a matrix $\mathbf{W} \in \mathbb{R}^{d_{in} \times d_{in}}$ from $\mathbf{U} \in \mathbb{R}^{k \times d_{in}}$ and $\mathbf{V} \in \mathbb{R}^{k \times d_{in}}$ where $k << d_{in}$ using a Khatri-Rao product compared to standard low rank approximations used in models such as LoRA. In the context of LoRA the matrices U and V would be multiplied as follows $U^T V$ to produce a matrix of size $d_{in} \times d_{in}$. Since $k < d_{in}$ and assuming U and V have rank k, we then have by properties of the rank of a product that

$$Rank(U^T V) = k. \tag{A.10}$$

In particular there are no conditions we can impose on U and V such that $U^T V$ has rank greater than k. However, by taking a Khatri-Rao product we will show that under suitable conditions we can obtain a matrix with much larger rank $= min(k^2, d_{in})$. To show this we need the following lemma borrowed from from Albert *et al.* [2] (lemma D.2) that we rewrite here to allow this proof to be self-contained.

Lemma A.4. *Let $\{X_1, \ldots, X_n\}$ denote n vectors in $\mathbb{R}^m$ where $n \leq m$ drawn i.i.d from a Gaussian or uniform distribution. Then with probability 1 $\{X_1, \ldots, X_n\}$ will be linearly independent.*

Proof. We first note that any measure defined via a Gaussian or Uniform probability distribution is absolutely continuous with respect to the Lebesgue measure. Meaning they have the same sets of measure zero as the Lebesgue measure.

We then prove the case that $\{X_1, \ldots, X_n\}$ are vectors of unit length. Since the vectors were drawn independently, we can first assume we drew X_1. The probability that this is the zero vector is 0 w.r.t the Lebesgue measure on the closed unit ball $B_N(0)$ about the origin in $\mathbb{R}^N$ and hence any other measure absolutely continuous to it. Then draw X_2 and note that the probability that X_2 lies in $span\{X_1\} \cap B_N(0)$ is also 0 since $span\{X_1\} \cap B_N(0)$ forms a set of 0 Lebesgue measure in $B_N(0)$. Continuing in this way we find that $\{X_1, \ldots, X_n\}$ will be linearly independent with probability 1.

For the general case where $\{X_1, \ldots, X_n\}$ are not drawn to have unit length i.e. drawn on the sphere in $\mathbb{R}^N$, we simply note that we can draw each one and then divide by its norm producing one of unit length. Since normalizing by the norm doesn't affect linear independence we get by the above case that $\{X_1, \ldots, X_n\}$ must be linearly independent with probability 1. $\square$

We now prove theorem 3.1.

Proof. Let $\mathbf{U} \in \mathbb{R}^{k \times d_{in}}$ and $\mathbf{V} \in \mathbb{R}^{k \times d_{in}}$ where $k \leq d_{in} \leq k^2$ be matrices whose entries are chosen i.i.d. from a standard Gaussian or uniform distribution. Since $k \leq d_{in}$ write $d_{in} = nk + p$ where $0 \leq p < k$ i.e. p is the remainder when we divide d_{in} by k. Note that since the entries of U and V are chosen i.i.d from a Gaussian or uniform distribution we have with probability 1 that none of the columns of U are multiples of each other and none of the columns of V are multiples of each other. Furthermore, using lemma A.4 we have with probability 1 that the k column vectors $\{U_1, \dots, U_k\}$ are linearly independent, as well as the second k column vectors $\{U_{k+1}, \dots, U_{2k}\}$, and continuing in this way each batch of k column vectors $\{U_{(i-1)k+1}, \dots, U_{ik}\}$ for $1 \leq i \leq n$ are linearly independent and the final p vectors $\{U_{nk+1}, \dots, U_{nk+p}\}$ are linearly independent. We can also assume the same for the columns vectors of V.

We now observe that because

$$\{U_{(i-1)k+1}, \dots, U_{ik}\} \tag{A.11}$$

is linearly independent and

$$\{V_{(i-1)k+1}, \dots, V_{ik}\} \tag{A.12}$$

is linearly independent for $1 \leq i \leq n$ and

$$\{U_{nk+1}, \dots, U_{nk+p}\} \tag{A.13}$$

are linearly independent and

$$\{V_{nk+1}, \dots, V_{nk+p}\} \tag{A.14}$$

are linearly independent. We have that

$$\{U_{(i-1)k+1} \otimes V_{(i-1)k+1}, \dots, U_{ik} \otimes U_{ik}\} \tag{A.15}$$

are linearly independent for $1 \leq i \leq n$ and that

$$\{U_{nk+1} \otimes V_{nk+1}, \dots, U_{nk+p} \otimes V_{nk+p}\} \tag{A.16}$$

are linearly independent. This uses the fact that given a collection of p linearly independent vectors $x_1, \dots, x_p$ in $\mathbb{R}^q$ and another collection of p linearly independent vectors $\{y_1, \dots, y_p$ in $\mathbb{R}^q\}$ the collection of Kronecker products $\{x_1 \otimes y_1, \dots, x_p \otimes y_p\}$ in $\mathbb{R}^{q^2}$ are linearly independent.

Furthermore, since none of the column vectors of U are multiplies of the others and none of the column vectors of V are multiples of its others we have that the set of vectors

$$\{U_1 \otimes V_1, \dots, U_{nk+1} \otimes U_{nk+1}, \dots, U_{nk+p} \otimes U_{nk+p}\} \tag{A.17}$$

are linearly independent. Then observe that the Khatri-Rao product $U \odot V$ has columns precisely given by (A.17) and thus the columns of $U \odot V$ are linearly independent. Since $d_{in} \leq k^2$ we have that $rank(U \odot V) = d_{in}$ as required. □

B. Further empirical differences with Kronecker adapter

Because Kronecker adapters (Krona) also uses a form of Kronecker products for PEFT, we propose here to highlight more differences between KRAdapter and Krona in terms of effective rank at initialization. Empirical evidence from vision and language tasks presented in the main paper indicates that KRAdapter consistently attains higher effective ranks than Krona given comparable trainable parameter budgets. While a comprehensive theoretical justification for this observation is beyond the scope of this empirical study, we undertake a controlled numerical experiment to analyze the effective rank and singular value distribution resulting from matrix approximation using both Kronecker and Khatri-Rao products. Specifically, we generate random matrices of dimensions relevant to transformer architectures, ranging from ViT-L/14 attention heads (768, 1024) to LLama 3.1 attention heads (4096, 4096). We configure KRAdapter-style factorization based on its inherent shape-dependent parameter allocation. To ensure a fair comparison, we then tune Krona's hyperparameters to precisely match the number of trainable parameters used by the KRAdapter configuration. The parameters are then initialized using a kaiminig uniform initialization strategy. Figure 4 presents the singular value decomposition and effective rank for both factorization methods across these matrix dimensions. Our analysis reveals that the Khatri-Rao product yields a consistently higher effective rank (10-50%) and a more gradual decay in the singular value spectrum compared to the Kronecker product. This suggests a more uniform distribution of singular values, indicative of richer representational capacity. These empirical findings substantiate the observed performance advantages of KRAdapter in vision and language tasks, highlighting the superior effective rank achieved by Khatri-Rao product factorization for parameter-efficient adaptation.

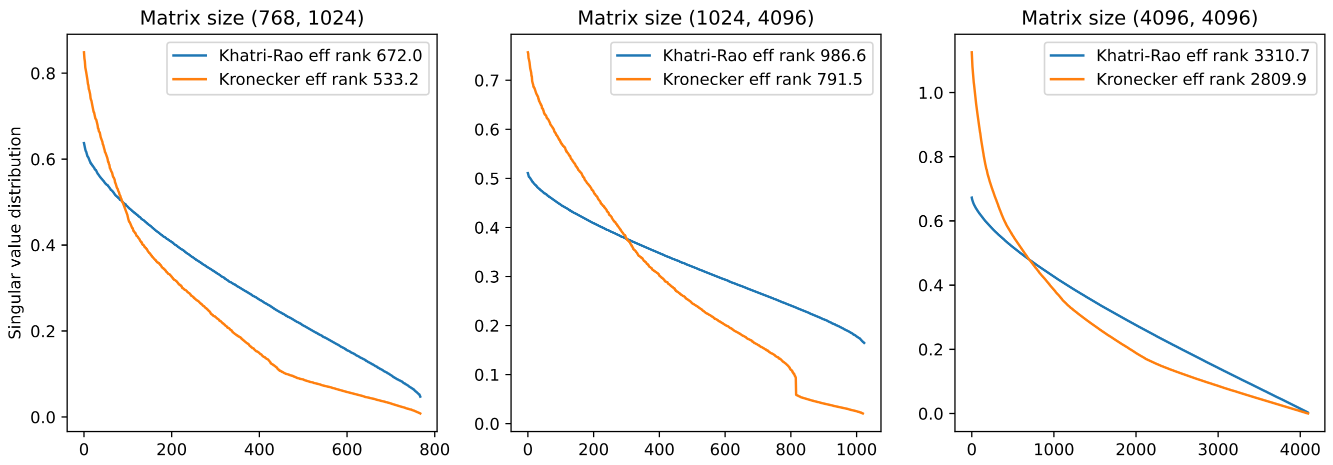

Figure 4. We compare the singular value distribution and effective rank resulting from a parameter-efficient construction of a matrix of set size using either Khatri-Rao or Kronecker products. For an equivalent amount of randomly initialized parameters, the Khatri-Rao produces a matrix with a smooth, more balanced svd sprectrum, resulting in a higher effective rank.

C. Details about the toy experiments

C.1. Training details

The matrices used in our experiments have a size of 1024×768. We aim to match the number of parameters of our proposed KRAdapter as closely as possible. Minor discrepancies in parameter counts for other methods arise due to the inherent structural differences of each adaptation technique. Specifically, the number of parameters for each method is: LoRA and SinLoRA: $50,176$, Krona: $50,700$, RandLoRA: $49,168$, and KRAdapter: $49,152$. We train models using the AdamW optimizer [1] for 100 iterations with a fixed learning rate of 10^{-2}. The AdamW optimizer is used with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay=0.01). Our training objective is to minimize the mean of the squared error between the predicted and target matrices.

C.2. Matrix generation

We generate six different synthetic patterns, each designed to probe specific aspects of parameter-efficient fine-tuning algorithms. **Normally-distributed Random Matrix** generated from a standard normal distribution. This serves as a baseline representing a high-rank weight matrix, testing the algorithms' general approximation capability. **Sparse Random Matrix (90% Sparsity)**, a normally distributed random matrix where 90% of elements are randomly set to zero. This baseline simulates scenarios where pre-trained models contains crucial parameters that should not be modified during fine-tuning. **PCA-Whitened Random Matrix**, a random matrix transformed using Principal Component Analysis (PCA) whitening. This process de-correlates the random features, assessing how well algorithms can generate highly de-correlated representations. **Low-Rank Matrix** constructed by taking a normally distributed random matrix and zeroing out all but the top one fourth of singular values. Tests the ability of full-rank algorithms to model low-rank matrices. **CLIP ImagNet fine-tune delta (Vision or Language)**, obtained by the element-wise difference between the pre-trained CLIP-ViT-L/14 weights and the weights obtained after standard fine-tuning on ImageNet (also known as task vector [63]). The weight difference is extracted from the last attention layer of either the vision or language backbone. This pattern represents a realistic target weight for LoRA when adapting pre-trained transformer weights, allowing to assess performance on real-world fine-tuning. **High/Low frequency features** where rows are generated using up to 5 superposed sinusoidal functions, with frequencies linearly increasing along the rows. The high frequencies are contained between $[1000, 10000]$ Hz while the low frequencies are contained between $[1, 100]$ Hz. This structured pattern assesses the algorithms' bias towards feature frequencies.

For the normally-distributed random and identity matrices, we respectively use the `torch.randn` [2] and `torch.eye` [3]

[1]https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html
[2]https://pytorch.org/docs/main/generated/torch.randn.html
[3]https://pytorch.org/docs/main/generated/torch.eye.html

functions to generate matrices of the desired size.

PCA-Whitened Random Matrix: We generate a normally-distributed random matrix using `torch.randn` and then perform PCA whitening. This involves multiplying the data by the eigenvectors of the covariance matrix, effectively decorrelating the features. We then scale each row of the resulting matrix by the square root of the corresponding eigenvalue to normalize the variance.

High/Low frequency features: Each row of the matrices is generated by sampling a sinusoid function $\mathbf{f}(f, t) = \sin(2\pi f t)$ over one second. The frequency f increases linearly from 1 Hz for the first row to $1,000$ Hz for the last row ($1,000$ to $10,000$ for the high frequencies). This creates a matrix where each row represents a sinusoid with a different frequency.

C.3. Visualization

We propose a visualization of the achieved reconstruction for each PEFT algorithm in Figure 5 for smaller 128×128 matrices. For the finetuned weights, we select the first 128 rows and columns.

D. Effective rank

To further investigate the intrinsic dimensionality of each method we report the average effective ranks averaged across attention layers post fine-tuning in Table 4. Specifically, the effective rank [47] of a matrix M is calculated as $\exp(-\sum_i S_i^n \log S_i^n)$, where $S_i^n = \frac{S_i}{\sum_i S_i}$ represents the sum-normalized singular values of M. An effective rank close to the mathematical rank indicates that the weight matrix makes full use of the available spectrum to significantly modify the space in a wide range of directions. We report that the effective rank of KRAdapter is consistently higher than that of other theoretically full-rank algorithms.

	LoRA	SinLoRA	RandLoRA	Krona	KRAdapter	Max rank
ViT-B-32	4.5	21.9	494.8	518.5	**705.9**	768
ViT-L-16	13.1	31.8	587.0	744.0	**959.7**	1024
LLama3.1	16.8	24.0	562.3	734.2	**970.6**	1024
Qwen2.5-7B	8.5	18.7	247.6	310.5	**486.6**	512

Table 4. Effective ranks of full-rank PEFT algorithms for vision or language architectures.

E. Training times and VRAM usage

We find that all algorithms use comparable amounts of VRAM during training except for RandLoRA which comes at the cost of a slight increase in training time. We report training time results in Table 5 for various ViT architecture for 1 epoch on ImageNet and LLama3-8b for the commonsense reasoning task for 4 epochs (160k samples in total). Note that PEFT algorithms are trained on attention layers only. Although not reported in this table, DoRA's training time is comparable to RandLoRA.

F. CLIP classification

F.1. Dataset details

We fine-tune pre-trained vision-language architectures on 11 vision datasets. For few-shot learning experiments, we consistently train models for 10 epochs. In contrast, for 50% and 100% fine-tuning scenarios, we follow [2, 63] and adjust the number of training epochs for the full fine-tuning baseline based on convergence behavior, aiming for optimal performance. We do not perform early stopping as we do not observe significant over-fitting. All algorithms use the same training samples and training

Algorithm	ViT-B/32	ViT-L/14	ViT-H/14	LLama3-8B	Qwen2.5-7B
LoRA	16.8 mins	134.1 mins	215.5 mins	243.3 mins	222.2 mins
SinLoRA	16.8 mins	136.9 mins	216.6 mins	246.2 mins	224.3 mins
RandLoRA	16.7 mins	138.4 mins	225.5 mins	265.3 mins	235.2 mins
Krona	16.6 mins	135.9 mins	217.2 mins	250.4 mins	227.5 mins
KRAdapter	16.5 mins	137.5 mins	220.1 mins	247.6 mins	226.3 mins
FT	21.1 mins	167.9 mins	270.5 mins	Not trained	Not trained

Table 5. Comparison of training time for one epoch on ImageNet on CLIP architectures and LLama3-8B, Qwen2.5-7B for one epoch on the commonsense reasoning dataset

epochs. Detailed specifications of the 11 datasets, including number of training samples and the specific number of epochs used, are reported in Table 6.

#	Datasets	Classes	Splits			Epochs
			train	*val*	*test*	
(1)	Cars	196	7,330	814	8,041	35
(2)	DTD	47	3,384	376	1,880	76
(3)	EuroSAT	10	21,600	2,700	2,700	12
(4)	SUN397	397	17,865	1,985	19,850	14
(5)	Food101	101	70,750	5,000	25,250	15
(6)	Caltech101	101	6,941	694	1,736	10
(7)	FGVCAircraft	100	3,334	3,333	3,333	60
(8)	Flowers102	102	1,020	1,020	6,149	40
(9)	OxfordIIITPet	37	3,312	368	3,669	5
(10)	UCF101	101	7,639	1,898	3,783	20
(11)	ImageNet	1,000	1,276,167	5,000	50,000	1

Table 6. Vision datasets used for the image classification experiments

F.2. Classic datasets

We report detailed average results for the classic datasets of Table 6 for ViT-B/32, ViT-L/14 and ViT-H/14 in Table 7.

	ViT-B/32						ViT-L/14						ViT-H/14					
Shots	1	4	16	50%	100%	Avg.	1	4	16	50%	100%	Avg.	1	4	16	50%	100%	Avg.
LoRA	60.93	66.11	69.47	74.53	77.48	69.70	74.82	78.65	81.64	85.59	88.17	81.77	79.82	80.91	83.00	86.39	88.51	83.73
SinLoRA	60.36	67.93	72.31	75.90	78.38	70.98	75.43	80.09	82.69	86.14	88.03	82.48	79.96	82.59	84.66	86.49	88.22	84.38
RandLoRA	59.40	68.98	73.91	78.57	81.99	72.57	76.26	81.60	84.28	87.92	89.93	84.00	81.40	84.19	86.52	89.48	90.83	86.48
Krona	58.64	68.94	73.86	78.05	81.12	72.12	75.75	81.52	84.47	88.11	89.85	83.94	79.74	84.03	86.68	89.62	90.81	86.18
KRAdapter	58.86	69.28	74.80	79.67	82.74	73.07	76.39	81.97	85.14	88.79	90.46	84.55	81.18	84.75	87.10	89.62	90.76	86.68
FT	58.90	70.03	75.52	80.31	83.42	73.64	77.39	80.96	84.97	87.91	90.03	84.25	77.39	80.96	84.97	87.91	90.03	83.65

Table 7. Parameter-efficient vision-language CLIP tuning for image classification.

F.3. VTAB1k

The Visual Task Adaptation Benchmark (VTAB) [62] is a collection of datasets used to evaluate the capacity of PEFT algorithms to adapt large pretrained models to 3 categories of tasks.

F.3.1. Dataset presentation

Natural subset *Caltech101* [34] focuses on classifying images of 102 object categories, including common objects and a background class. *CIFAR-100* [30] is a natural image classification dataset with 100 classes. The *DTD* dataset [10] involves classifying textural patterns across 47 classes. *Flowers102* [4] is dedicated to classifying 102 flower species found in the UK. *Pets* [43] is a dataset for classifying cat and dog breeds, containing 37 classes. *Sun397* [58] is a scenery classification benchmark with 397 hierarchically structured classes.

Specialized subset *SVHN* [42] is a dataset for classifying street-view house numbers with 10 classes. *EuroSAT* [21] consists of Sentinel-2 satellite imagery for land use classification into 10 classes. *Resisc45* [8] is a remote sensing image scene classification dataset with 45 classes. *Patch Camelyon* [53] is a large dataset of histopathologic scans for binary classification of metastatic tissue presence. The *Retinopathy* dataset [15] focuses on predicting the severity of Diabetic Retinopathy on a 0-4 scale from high-resolution retina images.

Structured subset The *CLEVR* [27] datasets utilize images from a visual question answering task, with the 'count' variant predicting the number of objects and the 'distance' variant predicting the depth of the closest object. The *dSprites* [38] dataset, originally designed for disentanglement learning, is repurposed for location and orientation prediction tasks of simple 2D shapes. Similarly, the *SmallNORB* [32] dataset, containing images of 3D toys, is used for predicting azimuth and elevation angles of the objects. *DMLab* [3] provides 3D navigation environments where the task is to classify distances to reward objects, and finally, *KITTI* [17] involves predicting the depth of vehicles in real-world driving scenes. These tasks require models to reason about object counts, distances, orientations, and locations, spanning both 2D and 3D visual understanding which presents significant challenges for CLIP architectures.

F.3.2. Prompt design

Although the prompts design for the natural and part fo the specialized subset is straight forward, these are not evident for the structed subset especially when the classification is discrete. We settle on the class names described in Table 8 as we find they perform better than random for the zero-shot models and allow to see an improved performance with stronger zero-shot models. The final prompt we train CLIP with is "An image of a {classname}.' and we train with the SimCLR [6] augmentations.

F.3.3. Detailed results

Tables 9 report per dataset detailed results for PEFT algorithms using the ViT-{B/32,L/14,H/14} architectures respectively

F.4. OOD datasets

We evaluate the out-of-distribution (OOD) generalization of image classification models trained on ImageNet [31], using datasets that probe model robustness under various distribution shifts with the standard ImageNet test set:

ImageNet-A (Naturally Adversarial) [23] comprises 7,500 real-world images from 200 ImageNet classes that are confidently misclassified by standard models, yet easily recognizable by humans. ImageNet-A assesses robustness to naturally occurring, subtle adversarial examples present in real-world data, highlighting vulnerabilities beyond synthetic adversarial attacks.

ImageNet-R (Renditions) [22] contains 30,000 images across 200 ImageNet classes, featuring artistic renditions like paintings, sketches, and sculptures. It evaluates robustness to significant stylistic domain shifts, testing if models generalize beyond photographic images and capture semantic content despite variations in visual style.

ImageNet-Sketch [56] presents a more extreme domain shift with 50,000 black and white sketches across all 1,000 ImageNet classes. ImageNet-Sketch serves as a stress test, evaluating a model's ability to generalize to drastically different image modalities and rely on high-level semantic understanding rather than low-level image features.

ImageNet-v2 [46] is not an OOD dataset in the same sense but an updated test set collected using the original ImageNet methodology. It aims to provide a more reliable evaluation by mitigating potential test set contamination and overfitting to the original ImageNet validation set. We study three subsets including "Freq" (Matched Frequency) which replicates the original

Dataset	Class names
CAMELYON	’with no metastatic tissue’, ’containing metastatic tissue’
RETINOPATHY	’with no diabetic retinopathy’, ’with mild diabetic retinopathy’, ’with moderate diabetic retinopathy’, ’with severe diabetic retinopathy’, ’with extreme diabetic retinopathy’
CLEVR_COUNT	’3 items’, ’4 items’, ’5 items’, ’6 items’, ’7 items’, ’8 items’, ’9 items’, ’10 items’
CLEVR_DIST	’congested’, ’larger’, ’large’, ’normal’, ’small’, ’tiny’
DSPRITES_LOC	’0-6 percent x axis’, ’6-12 percent x axis’, ’12-18 percent x axis’, ’18-25 percent x axis’, ’25-31 percent x axis’, ’31-37 percent x axis’, ’37-43 percent x axis’, ’43-50 percent x axis’, ’50-56 percent x axis’, ’56-62 percent x axis’, ’62-68 percent x axis’, ’68-75 percent x axis’, ’75-81 percent x axis’, ’81-87 percent x axis’, ’87-93 percent x axis’,’93-100 percent x axis’
DSPRITES_ORIENT	’shape rotated 0-22.5 degrees clockwise’, ’shape rotated 22.5-45.0 degrees clockwise’, ’shape rotated 45.0-67.5 degrees clockwise’, ’shape rotated 67.5-90.0 degrees clockwise’, ’shape rotated 90.0-112.5 degrees clockwise’, ’shape rotated 112.5-135.0 degrees clockwise’, ’shape rotated 135.0-157.5 degrees clockwise’, ’shape rotated 157.5-180.0 degrees clockwise’, ’shape rotated 180.0-202.5 degrees clockwise’, ’shape rotated 202.5-225.0 degrees clockwise’, ’shape rotated 225.0-247.5 degrees clockwise’, ’shape rotated 247.5-270.0 degrees clockwise’, ’shape rotated 270.0-292.5 degrees clockwise’, ’shape rotated 292.5-315.0 degrees clockwise’, ’shape rotated 315.0-337.5 degrees clockwise’, ’shape rotated 337.5-360.0 degrees clockwise’
SMALLNORB_AZIMUT	’shape rotated by 0-20 degrees clockwise’, ’shape rotated by 20-40 degrees clockwise’, ’shape rotated by 40-60 degrees clockwise’, ’shape rotated by 60-80 degrees clockwise’, ’shape rotated by 80-100 degrees clockwise’, ’shape rotated by 100-120 degrees clockwise’, ’shape rotated by 120-140 degrees clockwise’, ’shape rotated by 140-160 degrees clockwise’, ’shape rotated by 160-180 degrees clockwise’, ’shape rotated by 180-200 degrees clockwise’, ’shape rotated by 200-220 degrees clockwise’, ’shape rotated by 220-240 degrees clockwise’, ’shape rotated by 240-260 degrees clockwise’, ’shape rotated by 260-280 degrees clockwise’, ’shape rotated by 280-300 degrees clockwise’, ’shape rotated by 300-320 degrees clockwise’, ’shape rotated by 320-340 degrees clockwise’, ’shape rotated by 340-360 degrees clockwise’
SMALLNORB_ELEVATION	’object photographed with a 30 degrees elevation’, ’object photographed with a 35 degrees elevation’, ’object photographed with a 40 degrees elevation’, ’object photographed with a 45 degrees elevation’, ’object photographed with a 50 degrees elevation’, ’object photographed with a 55 degrees elevation’, ’object photographed with a 60 degrees elevation’, ’object photographed with a 65 degrees elevation’, ’object photographed with a 70 degrees elevation’
DMLAB	’obstructed’, ’large’, ’bigger’, ’normal’, ’smallest’, ’empty’
KITTI	’congested’, ’close’, ’distant’, ’empty’

Table 8. CLIP Prompts for the Structed subset of VTAB-1k and + Camelyon and Retinopathy. We train the VL CLIP models with the prompt ”An image of a {classname}.”

	Natural							Specialized				Structured												
	CIFAR-100	Caltech101	DTD	Flowers102	Pets	SVNH	Sun397	Camelyon	EuroSAT	Resisc45	Retinopathy	Clevr-Count	Clevr-Dist	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Ele	DMLab	KITTI-Dist	**Mean Nat.**	**Mean Spe**	**Mean Struc**	**Group Mean**	**All Mean**
ViT-B/32																								
Zero-shot	41.5	78.8	41.7	66.7	87.7	25.8	59.5	60.7	31.4	53.8	55.7	25.1	17.4	6.7	8.1	6.2	11.6	20.2	39.8	57.4	50.4	16.9	41.6	38.9
LoRA	48.6	84.0	60.4	76.8	84.0	89.1	51.3	83.8	95.0	83.1	68.5	67.4	45.1	36.8	46.1	22.4	39.9	50.5	53.3	70.6	82.6	45.2	66.1	62.4
SinLoRA	49.0	84.7	60.6	84.2	84.4	90.3	50.6	84.5	95.0	83.9	69.0	69.3	53.1	78.5	51.5	22.4	41.2	51.9	56.8	72.0	83.1	53.1	69.4	66.4
RandLoRA	48.6	87.4	68.7	88.0	85.9	91.7	49.0	84.8	93.0	87.3	64.6	63.8	58.1	82.0	54.3	23.1	32.8	54.3	56.5	74.2	82.4	53.1	69.9	67.1
Krona	48.5	86.7	66.5	86.3	86.0	91.6	50.1	84.5	93.4	86.5	69.3	70.5	57.4	82.2	53.7	23.4	29.4	54.8	54.6	73.7	83.4	53.3	70.1	67.1
KRAdapter	52.3	88.4	70.3	88.9	87.0	91.7	53.5	84.9	93.1	88.4	69.4	69.4	58.3	79.9	54.0	25.3	32.1	54.1	53.0	76.0	84.0	53.3	71.1	68.1
FT	51.2	87.2	68.1	89.0	85.7	92.0	56.4	84.6	93.1	87.2	69.2	68.5	58.2	80.3	54.7	24.2	32.1	54.1	53.0	75.7	83.5	53.1	70.8	67.8
ViT-L/14																								
Zero-shot	55.9	80.9	52.5	78.9	93.3	56.4	64.2	54.7	42.6	66.2	23.9	19.0	22.5	6.6	6.8	5.5	9.3	21.1	17.6	68.9	46.9	13.6	43.1	40.9
LoRA	64.9	87.9	75.2	96.8	92.6	94.7	63.9	86.0	95.4	91.8	74.7	85.5	43.4	74.0	54.5	22.0	39.8	58.1	60.1	82.3	87.0	54.7	74.6	71.6
SinLoRA	65.9	88.2	75.7	97.0	92.5	94.7	63.7	86.8	95.6	92.0	72.8	86.5	45.5	84.7	60.9	22.1	38.0	60.4	56.4	82.5	86.8	56.8	75.4	72.6
RandLoRA	63.4	89.1	76.7	97.3	93.8	94.7	64.5	86.7	94.8	92.2	74.2	81.0	60.2	84.5	60.9	25.8	35.7	60.4	58.1	82.8	87.0	58.3	76.0	73.4
Krona	64.5	89.8	77.1	97.5	92.9	95.4	66.9	86.9	95.4	92.7	74.5	79.1	56.4	83.9	61.5	26.5	35.7	58.9	58.1	83.4	87.4	57.5	76.1	73.3
KRAdapter	70.0	89.8	78.5	98.0	93.7	95.2	68.5	86.5	95.1	93.0	73.0	81.7	55.3	79.1	62.0	25.3	34.5	59.5	60.1	84.8	86.9	57.2	76.3	73.6
FT	63.2	89.9	76.0	97.8	92.8	94.2	68.1	86.6	95.4	92.3	75.1	81.8	49.9	84.7	64.6	27.2	37.2	60.4	60.3	83.2	87.4	58.3	76.3	73.6
ViT-H/14																								
Zero-shot	65.9	83.4	63.5	79.6	94.6	45.5	74.7	54.5	52.9	70.9	23.4	34.9	22.6	6.1	8.9	5.9	11.2	15.2	37.8	72.4	50.4	17.8	46.9	44.8
LoRA	68.9	89.5	78.6	97.4	92.2	94.6	67.6	86.9	95.7	91.9	72.4	79.7	40.8	86.6	62.6	26.6	39.2	58.3	60.3	84.1	86.7	56.8	75.9	73.1
SinLoRA	69.0	89.8	78.0	97.4	92.3	94.7	68.5	87.3	95.5	92.0	72.7	87.1	41.8	87.4	62.9	28.3	39.2	60.0	58.6	84.2	86.9	58.2	76.4	73.8
RandLoRA	66.4	90.8	79.0	97.2	92.2	94.0	67.6	86.8	95.6	92.0	74.5	84.0	59.1	85.4	60.5	28.9	38.4	60.5	58.8	83.9	87.3	59.5	76.9	74.3
Krona	68.1	92.2	79.6	97.7	92.6	94.7	69.7	87.5	95.0	92.5	72.0	80.6	54.4	86.3	61.2	28.4	36.8	59.0	57.7	84.9	86.7	58.0	76.6	74.0
KRAdapter	71.2	92.5	80.0	98.1	93.0	94.4	71.6	86.1	95.6	93.1	73.3	84.5	53.3	84.0	60.3	27.2	36.4	59.5	59.6	85.8	87.0	58.1	77.0	74.4
FT	66.5	90.4	77.8	97.5	92.5	94.3	78.9	84.5	95.4	88.9	70.3	76.0	35.0	44.3	49.4	14.8	26.2	47.2	56.4	85.4	84.8	43.7	71.3	67.7

Table 9. Accuracies training on VTAB1k benchmark. We report per dataset accuracies as well as category-wise averages. Base networks are ViT CLIP models in version - B/32, L/14 and H/14 where both vision and language backbones are trained.

validation set's label distribution, "Top" (Top-5 Accuracy Matched) which matches the top-5 accuracy of a reference model, and "Thresh" (Thresholded) which uses a higher worker agreement threshold for potentially cleaner labels.

F.4.1. Detailed OOD results

Table 10 reports detailed per-dataset accuracies for the OOD experiments on ImageNet.

G. Commonsense reasoning

G.1. Dataset details

We test on 8 commonsense reasoning datasets. These benchmarks encompass a range of cognitive skills, including answering yes/no questions BoolQ [11]), addressing common-sense physics inquiries (PIQA [5]), understanding social dynamics (SIQA [49]), completing multi-choice scenarios (HellaSwag [61]), binary solutions to finish sentences (WinoGrande [48]),

	ImageNet (ID)	ImageNetA	ImageNetSketch	ImageNetR	ImageNetV2Thresh	ImageNetV2Top	ImageNetV2Freq	CIFAR100	**OOD**	**Improve ID**	**Improve OOD**	**Ratio**	**Effrank**	**Spectral**	**Fro**
ViT-B/32															
ZS	62.64	32.28	40.78	66.56	62.93	68.23	55.28	62.26	55.47	n/a	n/a	n/a	n/a	n/a	n/a
LoRA	72.16	28.6	42.82	66.10	70.80	76.32	62.52	63.79	58.71	9.52	3.2	0.34	20.4	19.6	4.1
SinLoRA	72.84	28.43	42.39	64.48	71.34	76.97	62.44	64.24	58.61	10.20	3.1	0.31	276.8	223.5	14.7
RandLoRA	72.01	27.55	42.05	64.31	71.00	76.72	61.8	65.64	58.44	9.37	3.0	0.31	464.6	46.2	5.7
Krona	71.88	28.51	42.38	66.06	71.04	76.48	62.16	65.95	58.94	9.24	3.5	0.38	584.0	44.1	4.9
KRAdapter	72.52	30.32	43.67	67.84	71.39	77.23	62.59	66.32	59.91	9.88	4.4	0.45	696.0	7.3	2.3
FT	75.54	25.71	42.35	64.31	73.41	78.7	64.85	62.68	58.86	12.9	3.4	0.26	590.4	0.7	0.8
ViT-L/14															
ZS	75.44	70.77	59.6	87.73	75.86	79.05	69.75	76.15	74.13	n/a	n/a	n/a	n/a	n/a	n/a
LoRA	83.34	70.73	61.36	86.81	82.22	84.98	76.06	78.08	77.18	7.90	3.05	0.39	22.6	46.2	6.6
SinLoRA	82.8	69.45	59.99	85.01	81.45	84.61	75.22	78.67	76.34	7.36	2.21	0.30	734.6	61.6	7.1
RandLoRA	82.78	68.96	59.66	85.02	81.84	84.93	75.32	77.95	76.24	7.34	2.11	0.28	605.5	159.9	11.8
Krona	84.08	72.09	61.40	87.17	82.87	85.55	76.48	78.88	77.78	8.64	3.65	0.42	755.2	42.7	5.02
KRAdapter	83.64	73.03	61.95	87.85	82.79	85.72	76.5	79.32	78.17	8.20	4.04	0.49	920.9	9.8	2.8
FT	85.05	68.13	60.30	86.00	83.41	86.14	77.28	75.74	76.71	9.61	2.58	0.27	758.8	1.0	1.0
ViT-H/14															
ZS	77.94	59.36	66.53	89.29	77.59	81.30	70.93	84.74	75.74	n/a	n/a	n/a	n/a	n/a	n/a
LoRA	83.65	56.76	64.38	85.62	82.54	85.91	76.21	79.89	75.90	5.71	0.22	0.04	27.7	70.0	7.4
SinLoRA	83.55	58.72	65.93	87.90	82.85	86.02	76.4	81.69	77.07	5.61	1.40	0.29	968	90.6	8.1
RandLoRA	82.94	57.04	63.4	84.90	81.84	85.17	75.13	80.58	75.43	5.00	-0.24	-0.05	752.3	508.3	21.21
Krona	85.02	64.33	65.78	87.52	83.62	86.55	77.15	82.43	78.20	7.08	2.52	0.36	882.4	123.3	9.2
KRAdapter	84.57	65.67	67.15	89.01	83.38	86.51	76.96	83.23	78.84	6.63	3.17	0.48	1140.2	32.8	5.5
FT	84.88	64.23	67.26	89.68	81.96	85.07	75.44	84.88	78.36	6.94	2.68	0.39	935.4	4.3	1.9

Table 10. Detailed results on OOD generalization with efficient rank

tackling both simpler and more complex elementary science questions (ARC-e and ARC-c [12]), and engaging in multi-stage reasoning (OBQA [40]). This collection of datasets presents different challenges, ranging from understanding the nuances of language and employing everyday knowledge to making inferences about the physical and social world. For a deeper exploration of these datasets, we redirect readers to the work of Hu et al. [25].

G.2. Training details

The models are trained using the Transformers library from Hugging Face[4]. We followed implementation specifics detailed by Albert et al. [2], whose code is publicly available[5]. The training lasts for four epochs, utilizing a learning rate of 1×10^{-4} and a base scaling coefficient of 2 for α weights. To combat overfitting we use dropout with a probability of 0.05 for each adapter layer. Unless otherwise specified, hyper-parameters were kept consistent across different architectures and algorithms. We train on the multi-choice tasks SIQA, ARC-C, ARC-E and OBQA and test on all tasks.

[4]https://huggingface.co

[5]https://github.com/PaulAlbert31/RandLoRA

H. GLUE

We further report results tuning RoBERTa [37] on the General Language Understanding Evaluation (GLUE) [54] dataset (see appendix H). We train for the SST-2, MRPC, COLA, QNLI, RTE and STS-N tasks. We report Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for the remaining tasks. We train the key and value matrix in the attention layers of a pretrained RoBERTa-large [37] network configuration with 355M parameters originally and perform 5 runs to report average performance and one standard deviation. We train each run for 10 epochs with a learning rate of 10^{-4}.Results are reported in Table 11 where we find that KRAdapter slightly outperforms other algorithms on average although results are very close. In this setting, the margin for improvement is small as the task is an easy binary classification. This translates to all PEFT algorithms producing results within an error margin of each other. KRAdapter however performs competitively in this setting as well.

LoRA	95.6 ± 0.2	88.7 ± 0.9	64.3 ± 1.2	94.6 ± 0.2	79.1 ± 4.0	91.8 ± 0.4	85.7 ± 0.9
SinLoRA	96.1 ± 0.1	88.9 ± 0.9	63.4 ± 0.9	93.6 ± 0.6	83.7 ± 0.4	91.8 ± 0.1	86.3 ± 0.2
RandLoRA	95.7 ± 0.3	88.7 ± 0.4	63.9 ± 1.3	93.9 ± 0.3	81.7 ± 2.3	91.8 ± 0.2	85.9 ± 0.3
Krona	95.8 ± 0.2	88.0 ± 0.8	59.6 ± 0.8	94.3 ± 0.2	78.7 ± 2.4	91.6 ± 0.3	84.7 ± 0.4
KRAdapter	95.9 ± 0.4	89.2 ± 0.6	64.6 ± 0.6	94.1 ± 0.3	82.5 ± 0.7	92.0 ± 0.3	86.4 ± 0.1

Table 11. Results on GLUE datasets with the RoBERTa-large model.

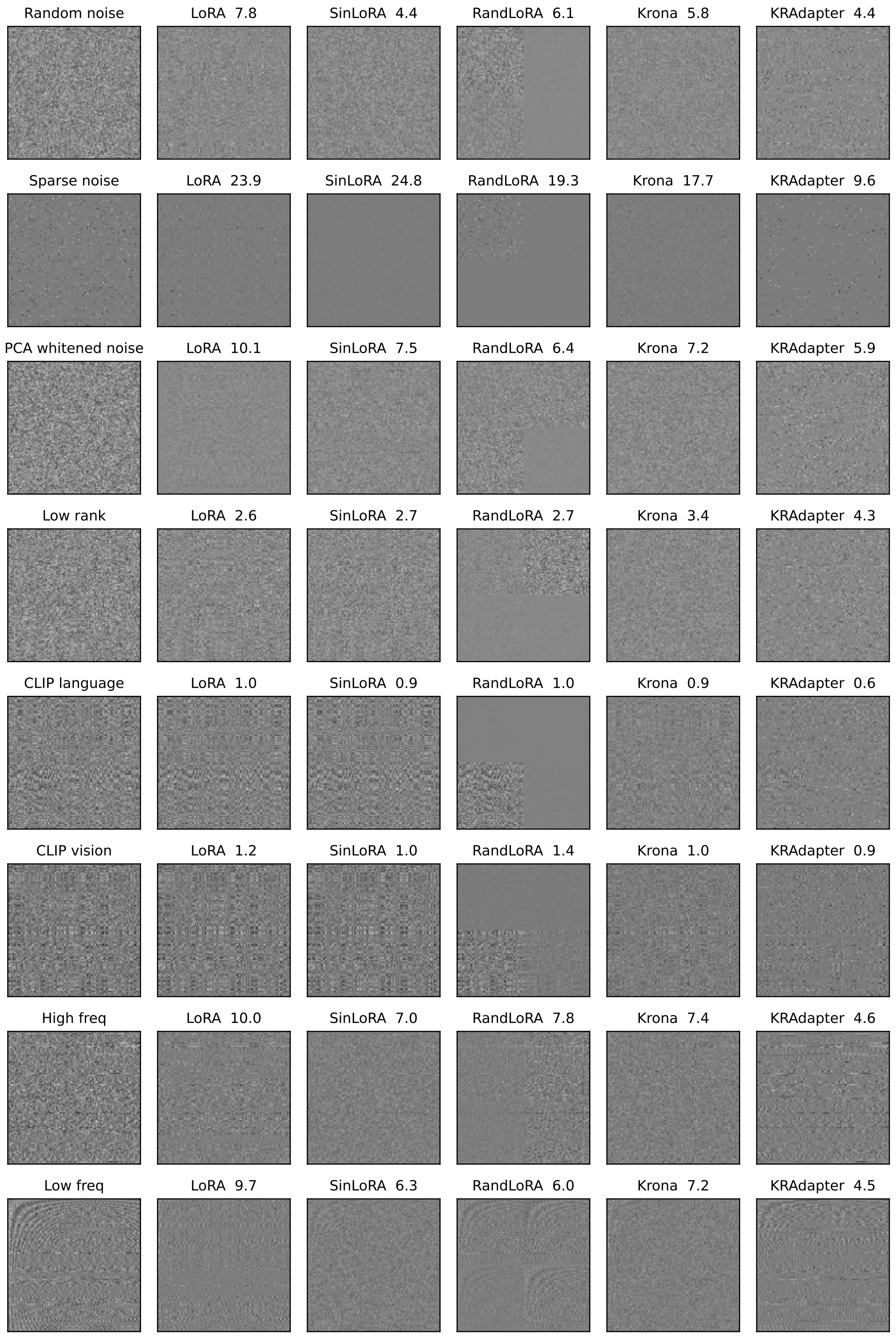

Figure 5. Toy experiment. We evaluate the capacity of PEFT methods to produce specific types of weight matrices. We report the generated matrices according to the target (left) and the absolute element-wise nuclear error. Lower is better. All algorithms train at least the same amount of parameters as KRAdapter