Lark: Low-Rank updates after knowledge localization for Few-shot Class-Incremental Learning

Supplementary Material

6. More Details for Lark

Additional details of this work are provided in the supplementary materials. Specifically, Section 6 presents detailed information on the Lark method, while Section 7 demonstrates the effectiveness of the proposed method through extensive experiments.

6.1. Implementation of Nearest Neighbor Classifier

In this work, we employ ViT-B/16 as the backbone and utilize the Nearest Neighbor Classifier in accordance with CLOSER [37] and OrCo [1].

6.1.1. Classifier of CLOSER

In this work, the implementation of the classifier head primarily reflects the its subsequent Classifier Replacement (CR) strategy. Specifically, the CR designs the following two stages:

Initial Training Stage During the training on base classes, the entire network consists of a feature extractor $f_\theta(\cdot)$ and a classification layer. The weights of the classification layer are denoted as ϕ. The training objective is a softmax cross-entropy (SCE) loss, where the logits are computed using cosine similarity scaled by $1/\tau$. The loss function is formulated as:

$$L_{\text{ce}} = -\frac{1}{B}\sum_{i=1}^{B}\log\left(\frac{\exp\left(\frac{1}{\tau}\cos(z_i,\phi_{y_i})\right)}{\sum_j \exp\left(\frac{1}{\tau}\cos(z_i,\phi_j)\right)}\right), \quad (11)$$

where $z_i = f_\theta(x_i)$ represents the feature extracted from input x_i, and τ is a temperature parameter controlling the logit distribution. The classifier (i.e., the classification layer) is designed only to distinguish between classes.

Classifier Replacement Strategy (CR) After the initial training phase on base classes is finished, we freeze the feature extractor (i.e., f_θ) and adopt a classifier replacement strategy. Specifically, the original classification layer weights ϕ are replaced with class prototypes. Each class prototype ϕ'_j is given by:

$$\phi'_j = \frac{1}{N_j}\sum_{n=1}^{N_j} f_\theta\left(x_j^{(n)}\right), \quad (12)$$

where N_j is the number of samples in class j. For new input samples, classification scores are then obtained by computing the cosine similarity between their features and the class prototypes.

6.1.2. Classifier of OrCo

OrCo achieve linear classification by generating a set of mutually orthogonal pseudo-targets on the hypersphere. First, randomly generate a set of vectors $T = \{t_i\}_{i=1}^{|T|}$, where the number of vectors satisfies $|T| \geq C$, and C is the number of classes. Then, to ensure these pseudo-targets mutually orthogonal, we adopt a special loss function $L_{TG}(T)$:

$$L_{TG}(T) = \frac{1}{|T|}\sum_{i=1}^{|T|}\log\sum_{j=1}^{|T|} e^{t_i\cdot t_j/\tau_0}, \quad (13)$$

where τ_0 is a temperature parameter (set to 0.8 in this work). The loss function aims to maximize the angles between the pseudo-targets, ensuring they are uniformly distributed on the hypersphere and mutually orthogonal. Finally, we obtain a set of mutually orthogonal pseudo-targets by optimizing the loss function $L_{TG}(T)$ for 1000 iterations. Once generated, these pseudo-targets are fixed and remain unchanged during the entire training process.

Using the pseudo-targets, the nearest neighbor classification head leverages their orthogonality and similarity computations with sample features. The implementation details are as follows:

Sample Feature Extraction. Given a sample x, we extract its feature representation using the backbone network. Specifically, we use the class token $z_m^{(L)}$ from the last layer.

Matching with Pseudo-Targets. Calculate the similarity between the sample feature and each pseudo-target via the inner product.

Classification Rule. Classify the sample x into the class corresponding to the pseudo-target with the highest similarity to its feature $z_m^{(L)}$:

$$\text{class}(x) = \underset{j}{argmax}\, z_m^{(L)}\cdot t_j. \quad (14)$$

6.2. Perturbations Added to Hidden States

We collect all the hidden states $H = \{h_a^{(l)}, h_m^{(l)}\}_{l=1}^{L}$ of the original sample x from the backbone network. Next, we sequentially applied noise to the hidden states at each layer and measured the impact of that layer on the model's output by computing the gradient of the target function with respect to it. Specifically, taking the output $h_a^{(l)}$ of the *MHSA* module at layer l as an example, we uniformly added a perturbation ϵ to each token:

$$\widetilde{h}_a^{(l)} = h_a^{(l)} + \epsilon = [z_a^{(l)} + \epsilon; p_{a,1}^{(l)} + \epsilon; p_{a,2}^{(l)} + \epsilon; ...; p_{a,N}^{(l)} + \epsilon], \quad (15)$$

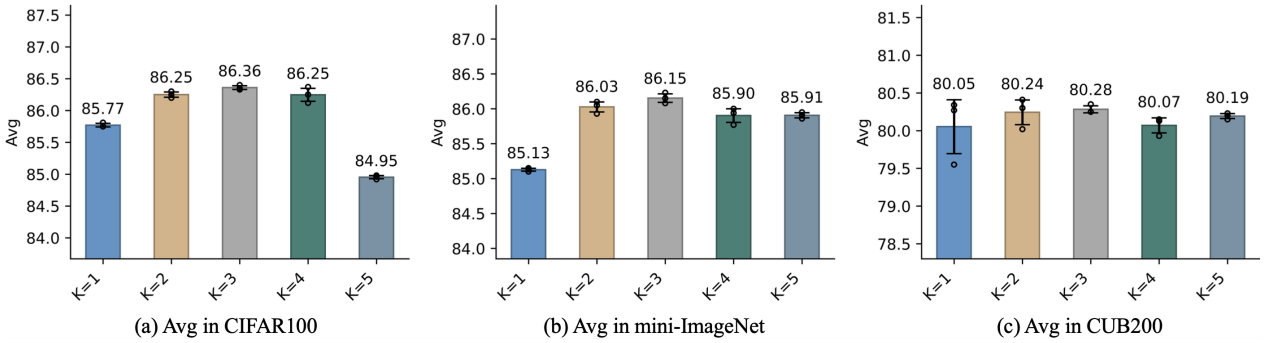


Figure 8. Parameter investigation experiments on the number of editable layers, measured by average accuracy **(Avg)**. The number marked on each bar represents the average value obtained from three experiments.

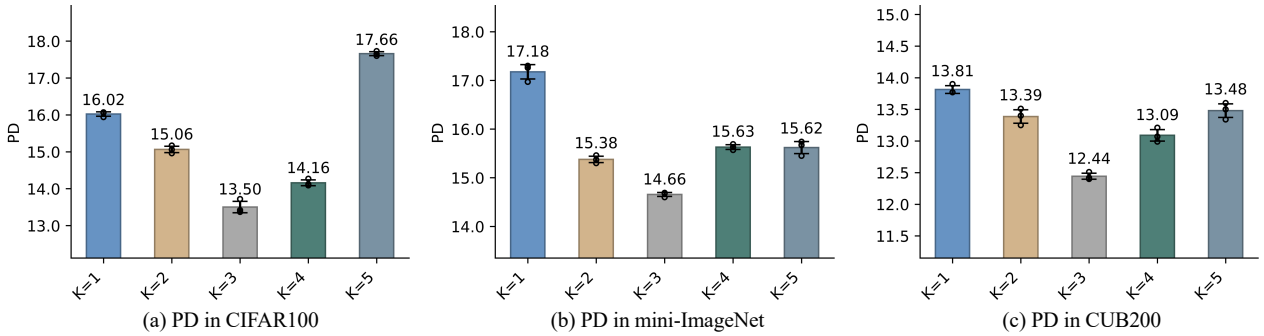


Figure 9. Parameter investigation experiments on the number of editable layers, measured by performance drop rate **(PD)**. The number marked on each bar represents the average value obtained from three experiments.

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The perturbed hidden states are collected as $\widetilde{H} = \{h_a^{(1)}, h_m^{(1)}, \ldots, \widetilde{h}_a^{(l)}, \widetilde{h}_m^{(l)}, \ldots, \widetilde{h}_a^{(L)}, \widetilde{h}_m^{(L)}\}$. It is important to note that the hidden states of the preceding layers remain in their original states, ensuring that only the specific layer's impact is evaluated and avoiding interference from noise in earlier layers. Additionally, the residual connections in the encoder allow the perturbation at the specific layer to influence the hidden states of subsequent layers through forward propagation, thereby enabling a more accurate assessment of that layer's influence.

6.3. Lowest-K for Parameter Selection

Figures 8 and 9 present the results obtained on different datasets with varying values of the editable layer count (K) set to 1, 2, 3, 4, and 5. Each value of K is tested in three separate experiments to ensure the reliability of the results. It is evident that, across all three datasets, a moderate number of editable layers (typically K=2,3,4) yields the best performance, avoiding both too few layers (which limits the model's learning capacity) and too many layers (which increases the risk of catastrophic forgetting). Notably, when K=3, the best performance was consistently achieved across all datasets.

6.4. Reason for SVD

Summing or averaging multiple Rank-One matrices typically results in a matrix that is no longer Rank-One, posing challenges for maintaining low-rank constraints. In this section, we approximate the resultant matrix using SVD to obtain a Rank-One matrix [3, 19]. We provide a rigorous mathematical formulation of this method and elaborate on its rationale and effectiveness.

Problem Definition. Given $N+1$ Rank-One matrices $\{\Delta W_i\}_{i=0}^{N}$, the total weight update matrix $\{\Delta W_i\}_{i=0}^{N}$ is expressed as:

$$\Delta W = \sum_{i=0}^{N} \Delta W_i. \tag{16}$$

Although each ΔW_i is Rank-One, the rank of the matrix ΔW may exceed one due to the subadditivity property of matrix rank, i.e.:

$$\text{rank}(\Delta W) \leq \sum_{i=0}^{N} \text{rank}(\Delta W_i). \tag{17}$$

Thus, to obtain a Rank-One approximation of ΔW, we impose a rank constraint. Specifically, we apply SVD to decompose the matrix $\Delta W \in \mathbb{R}^{m \times n}$.

Rank-One Approximation via SVD. The SVD of the matrix ΔW can be expressed as:

$$\Delta W = U \Lambda V^T, \tag{18}$$

where,$U \in \mathbb{R}^{m \times m}$ is the orthogonal matrix of left singular vectors; $V \in \mathbb{R}^{n \times n}$ is the orthogonal matrix of right singular vectors; $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r)$ is the diagonal matrix of singular values, with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > 0$, and $r = \text{rank}(\Delta W)$.

To approximate ΔW as a rank-one matrix, we retain only the largest singular value $\lambda_{\max} = \lambda_1$ and its corresponding left singular vector $u_{\max}$ and right singular vector $v_{\max}$. The rank-one approximation is then given by:

$$\Delta W \approx \lambda_{\max} u_{\max} v_{\max}^T, \tag{19}$$

where $\lambda_{\max}$ is the largest singular value of ΔW; $u_{\max}$ is the left singular vector corresponding to $\lambda_{\max}$; $v_{\max}$ is the right singular vector corresponding to $\lambda_{\max}$.

By retaining the principal singular components of ΔW, we achieve the best rank-one approximation.

Theoretical Proof. The optimality of this rank-one approximation is guaranteed by the Eckart-Young-Mirsky theorem [32]. This theorem states that, in terms of the Frobenius norm, the best Rank-k approximation of a matrix can be achieved by retaining the largest k singular values and their corresponding singular vectors.

Specifically, for k=1, our goal is to minimize the following error:

$$E = \|\Delta W - \lambda_{\max} u_{\max} v_{\max}^T\|_F, \tag{20}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. By retaining only the largest singular value λmax and its corresponding singular vectors $\lambda_{\max} u_{\max} v_{\max}^T$ captures the dominant information in ΔW, thereby achieving the optimal rank-one approximation in theory.

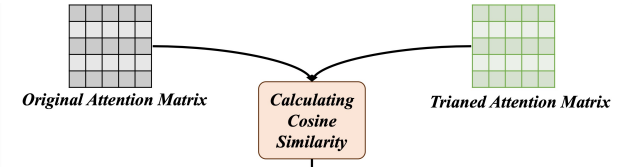
This method captures the dominant information in ΔW while significantly reducing computational complexity and ensuring the low-rank constraint of the result matrix. In practical applications, this approach exhibits good performance and stability. Furthermore, minimizing the error under the Frobenius norm guarantees that the approximation matrix has the smallest possible deviation from the original matrix [59].

7. More Details for Experiments

7.1. Details of Regular Training

To obtain the target vectors for editing, we perform regular training on the model to be edited. During this process, only the selected parameter matrices are updated, while the remaining parameters remain frozen. We then input the training data from the incremental session into the model and apply the cross-entropy loss to enable the model to quickly adapt to new knowledge. Each sample undergoes 50 iterations, with an initial learning rate of 0.1, and cosine scheduling is employed.

7.2. Changes in the Attention Matrix



	No.1	No.2	No.3	No.4	No.5
Sample 1	0.8321	0.8136	0.7884	0.7741	0.8072
Sample 2	0.8217	0.8429	0.8720	0.8817	0.9093
Sample 3	0.7958	0.8173	0.8325	0.8575	0.8627
Sample 4	0.8305	0.8581	0.8622	0.8823	0.8829
Sample 5	0.8052	**0.7617**	0.8438	0.8612	0.8087

Figure 10. Flowchart for calculating the similarity of attention matrices between the models before and after training on CIFAR100 Session 1 data.

Figure 10 presents the cosine similarities of samples in the models before and after training. This subsection details the entire computation process. The trained model refers to the model trained on the session 1 training data of CIFAR100; the untrained model has not undergone this training. We then input 25 test samples into both models, extract the attention matrices from the MHSA module of the sixth layer, and compute the similarities sequentially.

7.3. Comparison with SOTA and PEFT Insights

To further validate the effectiveness of our method, we conduct a comprehensive comparison with prompt-based methods (ASP [28], PriViLege [39]) and parameter-efficient fine-tuning approaches such as LoRA [15] and CKPD [26]. As shown in Table 6, our method, Lark, is evaluated both under the standard FSCIL setting (without using any pre-trained weights) and under the same pre-trained setting as ASP (denoted as Lark*). This dual evaluation ensures fair comparisons across different methodological assumptions.

Method	CIFAR100		CUB200		mini-ImageNet		MP (M)	LP (M)	FLOPs (G)
	Avg	PD	Avg	PD	Avg	PD			
ASP*	**89.00**	5.50	83.83	3.60	/	/	89.61	7.15	49.84
PriViLege*	88.08	4.82	77.50	7.13	95.27	2.58	197.07	14.4	77.13
CKPD*	88.62	5.35	84.95	3.72	95.36	3.50	87.43	0.11	35.99
LoRA*	83.26	14.17	76.81	15.33	84.73	14.46	86.78	0.29	27.40
Lark*	88.92	**4.29**	**85.21**	**3.36**	**95.83**	**2.34**	**86.60**	**0.02**	**22.17**
CKPD	/	/	79.00	11.49	90.66	9.95	/	/	/
Lark	86.03	7.74	82.00	9.77	88.15	9.43	/	/	/

Table 6. Results of ASP, PriViLege, and CKPD are obtained from their papers. Methods marked with * use pre-trained weights. LoRA fine-tunes all layers' q and v. Model Params (MP), Learnable Params (LP) and FLOPs are computed using the thop library on CIFAR100 with one sample.

Frozen	Localization	Rank-All	Rank-One	Acc. in each session (%) ↑									Avg ↑	PD ↓
				Base	1	2	3	4	5	6	7	8		
✔	✘	✘	✘	93.78	87.26	86.26	84.28	82.24	79.86	76.53	74.72	73.79	82.08	19.99
✘	✘	✘	✘	93.78	85.62	83.17	81.07	78.28	73.59	69.33	67.16	63.97	77.33	29.81
✘	✔	✘	✘	93.78	86.85	84.61	82.31	80.01	79.35	76.92	74.37	71.46	81.07	22.32
✘	✔	✔	✘	93.78	88.64	88.10	85.07	83.49	81.77	81.73	79.64	77.19	84.38	16.59
✘	✔	✘	✔	**93.78**	**90.45**	**88.89**	**87.31**	**85.55**	**84.24**	**84.19**	**82.54**	**80.28**	**86.36**	**13.50**

Table 7. Ablation study of different modules on **CIFAR100** in this work. The best results are **bolded**. **Avg** is the average accuracy across all sessions, and **PD** is the performance drop rate.

Frozen	Localization	Rank-All	Rank-One	Acc. in each session (%) ↑									Avg ↑	PD ↓
				Base	1	2	3	4	5	6	7	8		
✔	✘	✘	✘	93.58	86.26	83.17	81.51	78.53	77.59	76.11	75.56	75.03	80.82	18.55
✘	✘	✘	✘	93.58	85.18	81.98	79.18	76.38	73.59	76.79	73.99	70.19	78.98	23.39
✘	✔	✘	✘	93.58	85.83	82.05	78.47	76.95	75.27	74.38	74.05	72.21	79.20	21.37
✘	✔	✔	✘	93.58	89.04	86.46	86.37	85.24	81.78	80.96	80.23	77.63	84.59	15.95
✘	✔	✘	✔	**93.58**	**90.12**	**88.46**	**87.19**	**86.94**	**84.55**	**83.54**	**81.85**	**79.12**	**86.15**	**14.46**

Table 8. Ablation study of different modules on **mini-ImageNet** in this work. The best results are **bolded**. **Avg** is the average accuracy across all sessions, and **PD** is the performance drop rate.

Lark* achieves the best overall performance across all datasets in both average accuracy (Avg) and performance degradation (PD), with only a marginal 0.08% Avg shortfall compared to ASP on CIFAR100. Remarkably, Lark* accomplishes this using just 0.02M trainable parameters, less than 1% of ASP's, and under half of its computational cost. In the non-pretrained setting, our method consistently outperforms CKPD, demonstrating the robustness and adaptability of Lark in strict FSCIL scenarios. Furthermore, Lark surpasses LoRA across all datasets, achieving higher Avg and lower PD, while offering substantially improved storage and computational efficiency. This advantage arises from Lark's selective parameter update mechanism, in contrast to LoRA's indiscriminate fine-tuning approach.

From a broader perspective, our method sits at the intersection of Model Editing (ME) and Parameter-Efficient Fine-Tuning (PEFT). While PEFT approaches like LoRA prioritize scalability and efficiency through minimal trainable branches, ME techniques aim to inject knowledge with precision and minimal disruption. Both rely on low-rank updates to constrain parameter changes. Our method synergizes the strengths of both paradigms by identifying key parameters and updating them via a rank-1 matrix, achieving efficient yet targeted model adaptation.

7.4. More Detailed ablation Experiments

Tables 7, 8 and 9 present the results of the ablation experiments of Lark in OrCo-ViT on CIFAR100, mini-ImageNet and CUB200. The experimental results indicate that, under the condition of limiting the magnitude of parameter matrix updates, OrCo-ViT demonstrates a significant advantage. Moreover, updating only the selected parameters is clearly superior to updating all parameters globally. After incorporating the Rank-One constraint into our method, a notable improvement was observed in both datasets, surpassing OrCo-ViT.

7.5. Setup for Hand Keypoint Detection

Due to the lack of accurately labeled data [6], hand keypoint detection often involves pre-training on the synthetic dataset RHD [24], followed by transfer learning on the target domain data. We consider the challenges of this task to be describable as a few-shot incremental learning scenario. Therefore, we selected three datasets: RHD [62], H3D [58], and FHD [63]. RHD is a synthetic dataset, while H3D and FHD are datasets from real-world scenarios.

We designate RHD as the Base Session, where all data are utilized. It contains 41,000 training samples and 2,000 test samples. The H3D dataset is set as Session 1, from which we select approximately the top 1% of training samples (200 samples) as training data, with 3,200 test samples. The FHD dataset is set as Session 2, selecting approximately the top 1% of training samples (1,000 samples) as training data, with 32,000 test samples.

This setup not only simulates the difficulty of obtaining large amounts of labeled data in practical applications but also allows us to investigate how to effectively perform domain adaptation and generalization of models under few-shot conditions.

7.6. More Results on Hand Keypoint Detection

Figures 11 and 12 present additional visualization results.

Frozen	Localization	Rank-All	Rank-One	Acc. in each session (%) ↑											Avg ↑	PD ↓
				Base	1	2	3	4	5	6	7	8	9	10		
✔	✘	✘	✘	87.22	83.49	82.90	81.93	80.16	78.34	76.41	76.25	73.03	72.17	72.43	78.58	14.79
✘	✘	✘	✘	87.22	82.80	80.74	78.55	77.39	76.42	74.37	73.06	72.52	69.04	67.28	76.31	19.94
✘	✔	✘	✘	87.22	83.07	81.67	80.39	79.53	78.01	76.58	74.75	72.08	71.65	69.18	77.65	18.04
✘	✔	✔	✘	87.22	83.19	82.53	81.49	80.57	79.33	78.40	75.94	75.26	74.93	73.42	79.30	13.80
✘	✔	✘	✔	**87.22**	**84.48**	**83.46**	**82.77**	**81.54**	**80.69**	**79.43**	**77.61**	**75.62**	**75.45**	**74.78**	**80.28**	**12.44**

Table 9. Ablation study of different modules on **CUB200** in this work. The best results are **bolded**. **Avg** is the average accuracy across all sessions, and **PD** is the performance drop rate.

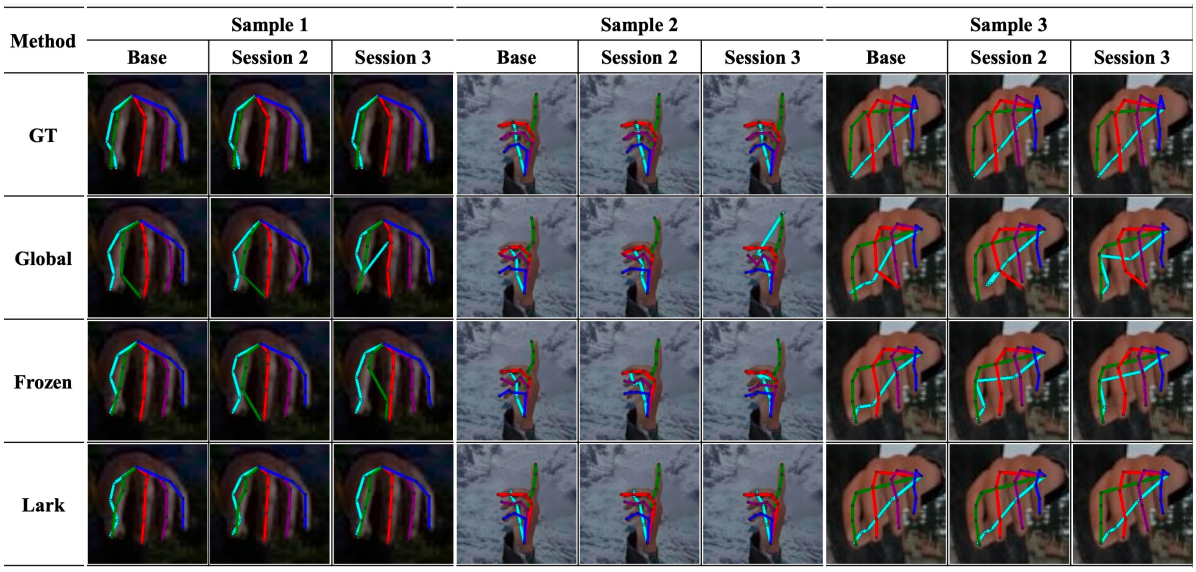


Figure 11. Detection performance of samples across different sessions. All samples are from the RHD dataset. Samples are trained and learned in the Base session but are no longer trained in Session 2 and Session 3. The performance variation of each sample in the same row under different methods reflects the ability of different methods to overcome catastrophic forgetting.

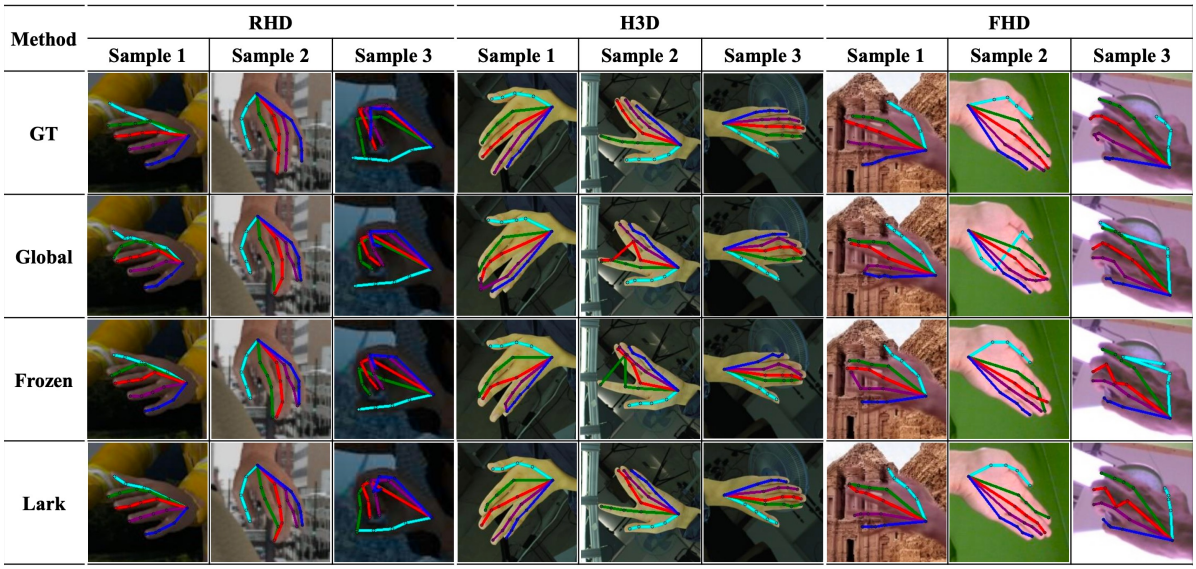


Figure 12. Detection performance of samples under different methods. Three samples from each dataset are selected for demonstration. The differences in each sample within the same column reflect the ability of different methods to learn new knowledge.