A. Proofs

A.1. Derivation of Eqn. (7) with NWGM approximation

The work of [1] gave the approximation to the expectation $\mathbb{E}_F[g(F)]$ as,

$$\mathbb{E}_F[g(F)] \approx \prod_F g(F)^{p(F)}, \tag{1}$$

based on which we have the expectation of the softmax function $\mathbb{E}_F[\sigma(g(F))]$ as,

$$\begin{aligned}\mathbb{E}_F[\sigma(g(F))] &\approx \frac{\prod_F \exp(g_y(F))^{p(F)}}{\sum_k \prod_F \exp(g_k(F))^{p(F)}} \\ &= \frac{\exp(\sum_F g_y(F)p(F))}{\sum_k \exp(\sum_F g_k(F)p(F))} \\ &= \sigma(\mathbb{E}_F g(F))\end{aligned} \tag{2}$$

Following Eqn.(2), we derive

$$\begin{aligned}&p(Y|do(F)) \\ =&\sum_z p(Z=z|F) \sum_f p(F=f)[p(Y|F=f, Z=z)] \\ =&\mathbb{E}_{Z|F}\mathbb{E}_F[p(Y|F,Z)] \\ =&\mathbb{E}_{Z|F}\mathbb{E}_F\sigma(\varphi_1(F,Z)) \\ \approx&\sigma[\mathbb{E}_F\mathbb{E}_{Z|F}\varphi_1(F,Z)].\end{aligned} \tag{3}$$

Provided that the classifier φ_1 is linear with respect to either $\mathbb{E}_F F$ or $\mathbb{E}_{Z|F} Z$, we have the further approximation of Eqn. (3) as,

$$\sigma[\mathbb{E}_F\mathbb{E}_{Z|F}\varphi_1(F,Z)] \approx \sigma[\varphi_1(\mathbb{E}_F F, \mathbb{E}_{Z|F})], \tag{4}$$

which completes the derivation.

A.2. Proof of Proposition 4.1

Proof. We have the causal chain $(\mathcal{D}^p, Y) \to F \to Z$, so that $I(\hat{Z}; Y, \mathcal{D}^p) \leq I(\hat{Z}; F)$. The left part $I(\hat{Z}; Y, \mathcal{D}^p) = I(\hat{Z}; \mathcal{D}^p) + I(\hat{Z}; Y|\mathcal{D}^p)$ according to the chain rule of mutual information. Combining both, we have

$$I(\hat{Z}; \mathcal{D}^p) \leq I(\hat{Z}; F) - I(\hat{Z}; Y|\mathcal{D}^p). \tag{5}$$

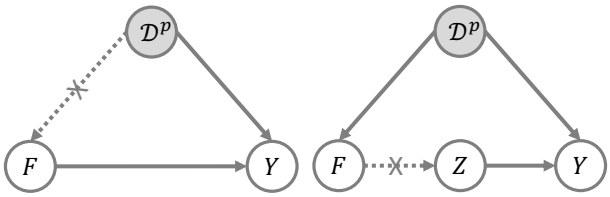
According to the definition of mutual information, we have

$$\begin{aligned}I(\hat{Z}; Y|\mathcal{D}^p) &= H(Y|\mathcal{D}^p) - H(Y|\hat{Z}, \mathcal{D}^p) \\ &= H(Y) - H(Y|\hat{Z}, \mathcal{D}^p) \\ &\geq H(Y) - H(H(Y|\hat{Z}) = I(\hat{Z}; Y),\end{aligned} \tag{6}$$

which is conditioned on the fact that the label Y is independent from the pre-training dataset $\mathcal{D}^p$. Substituting the inequality (6) into inequality (5), we finally obtain

$$I(\hat{Z}; \mathcal{D}^p) \leq I(\hat{Z}; F) - I(\hat{Z}; Y) = I(\hat{Z}; F) - I(F; Y) \tag{7}$$

which holds under the assumption that $\hat{Z}$ is sufficient for F. □



(a) Backdoor adjustment (b) Front-door adjustment

Figure 1: Two categories of deconfounding methods.

B. Background on Causal Intervention

The primary objective of causal intervention is to deconfound the confounder, *e.g.*, $\mathcal{D}^p$ in our problem. There are mainly two categories of deconfounding techniques, which we will detail below.

B.1. Backdoor adjustment

The backdoor adjustment seeks a way to block the causal link from the confounder $\mathcal{D}^p$ to the features F, as shown in Fig. 1a. Mathematically, it computes the causal effect of the features F to the predictions Y at each stratum of the confounder $\mathcal{D}^p$, *i.e.*,

$$p(Y|do(F)) = \sum_d p(Y|F, \mathcal{D}^p = d)p(\mathcal{D}^p = d), \tag{8}$$

where the $do(\cdot)$ operation denotes the estimation of the true causal effect via intervention, under the premise that the prediction of $p(Y|F)$ given each stratum d of $\mathcal{D}^p$ is not biased. Unfortunately, this backdoor adjustment method is not applicable to our problem, as the pre-trained dataset $\mathcal{D}^p$ is oftentimes too huge to access and stratify.

B.2. Front-door adjustment

Different from backdoor adjustment, front-door adjustment intervenes by introducing a mediator Z in the forward path $F \to Y$, leading to the prediction

$$p(Y|F) = \sum_z p(z|F)p(Y|Z=z). \tag{9}$$

In this case, deconfounding $\mathcal{D}^p$ requires both $F \to Z$ and $Z \to Y$ to be estimated with the true causal effect. First, $p(Z|do(F)) = p(Z|F)$ since Y works as a collider that blocks the information from F to Z, *i.e.*, $F \leftarrow \mathcal{D}^p \to Y \leftarrow$

Z. Second, similar to the stratification in back-door adjustment, the true causal effect $p(Y|do(Z=z))$ is obtained by computing at each stratum of F, *i.e.*,

$$p(Y|do(Z=z)) = \sum_{f} p(Y|z,f)p(f). \tag{10}$$

Substituting Eqn. (10) into Eqn. (9) gives the overall front-door adjustment as,

$$p(Y|do(F)) = \sum_{z} p(z|F) \sum_{f} p(Y|z,f)p(f). \tag{11}$$

C. Pseudo-code

We present the pseudo-codes of resolving rare features in Algorithm 1 and of resolving spuriously correlated features in Algorithm 2.

Algorithm 1 Rare features.

Require: Pre-trained feature extractor f; classification matrix W; queues Q; patch size PS; batch size B; iterations T; momentum m; the query parameters θ_q of f and W.

Initialize the momentum-updated model (f_k and W_k) with parameters θ_k by copying θ_q.
Randomly initialize the keys Q_y for each category y.
for iteration = 1 to T **do**
  Sample a batch of data $\{(x_i, y_i)\}_{i=1}^{B}$;
  Generate image-level features $F_i = f(x_i)$ and predictions $W(F_i)$ for each image;
  Get the confusing classes y_i' and the ground true y_i;
  Get 9 channels with $W_{y_i} - W_{y_i'}$ closest to zero and crop patches with a size of PS at the most attentive position of each selected channel;
  Obtain rare features $\{F_i^{r1}, F_i^{r2} ... F_i^{r9}\}$ from the cropped patches via feature extractor f;
  Copy keys from Q and detach.
  for i = 1 to B **do**
    Obtain positive keys F_j^r from Q_{y_i};
    Obtain negative keys F_k^r from $Q_{y \in y \neq y_i}$;
    Calculate losses for this sample: $\mathcal{L}_r^i$;
    Update queue: repeat $5-8$ using f_k and W_k to update rare features F_j^r in Q_{y_i};
  end for
  Average losses in the batch;
  loss.backward();
  Update θ_q by gradients and update θ_k via $\theta_k \leftarrow m\theta_k + (1-m)\theta_q$;
end for

Algorithm 2 Spuriously correlated features.

Require: Pre-trained feature extractor f; patch/channel attention modules with aggregation operations φ_2; classifier φ_1; batch size B; iterations T.

Randomly initialize the φ_1 and φ_2.
for iteration = 1 to T **do**
  Sample a batch of data $\{(x_i, y_i)\}_{i=1}^{B}$;
  Generate mediator $\hat{z}_i = \varphi_2(f(x_i))$ for each image;
  Obtain predictions $\varphi_1(\hat{z}_i)$ for each image;
  Calculate losses for each sample: $\mathcal{L}_s^i$;
  Average losses in the batch;
  loss.backward();
end for

D. Diagrams

D.1. Diagram of rare features

The detailed structure of resolving rare features is shown in Figure 2.

D.2. Diagram of spuriously correlated features

The detailed structure of resolving spuriously correlated features is shown in Figure 3.

E. Experimental Setup

E.1. Dataset description

Following [13, 16, 15], we evaluate our methods on eight datasets: **CUB-200-2011** [12], **Stanford-Cars** [7], **FGVC Aircraft** [9], **CIFAR10**, **CIFAR100** [8], **Vegetable** [11], **ISIC** [3] and **Caltech101** [4]. In addition, **ISIC** is a medical dataset with a highly imbalanced class distribution, which is closer to real-world applications. All datasets are obtained from the official websites agreeing with their licenses. The split of 100% sampling rates either follows previous works [13, 16, 15] except ISIC. ISIC is a closed challenge dataset that does not provide the label of the official testing set. Thus we randomly split the official training set into our training set and testing set. Besides, the split in experiments of different data sizes (*i.e.*, sampling rates 50%, 30%, and 15%) is provided by previous works [13, 16]. We will also provide detailed image lists for the data split in our codes. Table 1 reports the statistics of all datasets.

E.2. Environments and implement details

All methods are implemented in PyTorch [10] and on a computational platform with 8 Tesla V100 GPUs. Checkpoints of ResNet-50/ResNet-101 supervised pre-trained on ImageNet-1k are provided by PyTorch, and those of ResNet-50 self-supervised pre-trained on ImageNet-1k are obtained on their official implementation websites. Especially, SimCLR [2] and BYOL [5] provide the checkpoints imple-

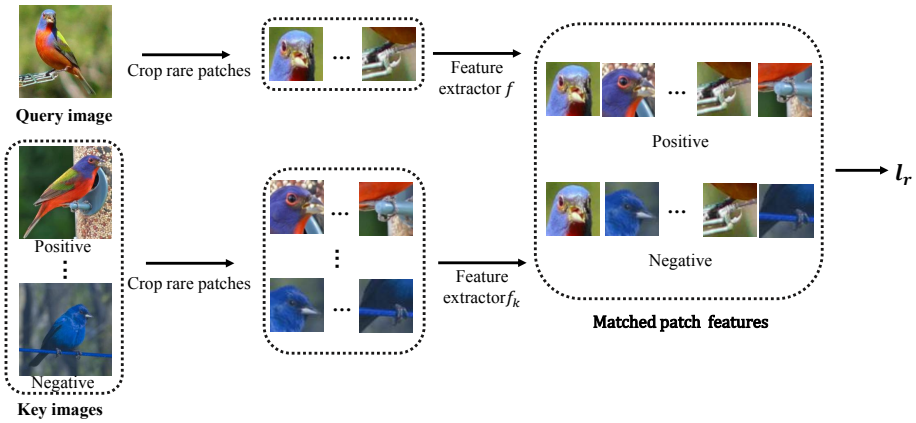


Figure 2: Illustration of resolving rare features. Given a query image and a set of key images, whose label is the same as the query image, are the attentive regions

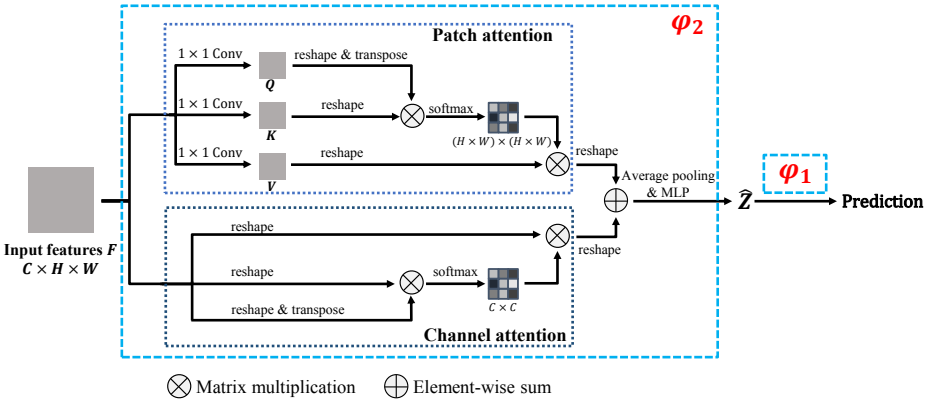


Figure 3: Illustration of resolving spuriously correlated features. Given an extracted feature map, we first use two attention modules to approximate the expectations and then aggregate them via a concatenation operation. Then, $\hat{Z}$ is obtained by an MLP, which is later used for predictions with φ_1.

Table 1: Statistics of datasets.

Datasets	Classes	Training	Testing
CUB-200-2011	200	5994	5794
Stanford-Cars	196	8144	8041
FGVC Aircraft	100	6667	3333
CIFAR10	10	50,000	10,000
CIFAR100	100	50,000	10,000
Vegetable	200	20,000	61,117
ISIC	7	5005	5010
Caltech101	102	3,060	6,084

mented in TensorFlow and we convert them into PyTorch using the codes approved by their authors. For all datasets, we follow the data augmentations used in [13, 16]: during training, images are randomly resized and cropped into 224×224 and then randomly horizon-flipped; during inference, images are resized to 256 and then center-cropped to 224×224. To be consistent with [16], the batch size is set as 48 for all datasets except CIFAR10 and CIFAR100, and all methods are optimized by stochastic gradient descent with momentum 0.9. More statistics of hyper-parameters are reported in Table 2.

F. Additional Experimental Results

F.1. Compared to the model trained from scratch

In the appendix, we further report the comparison of the model trained from scratch and Concept-tuning. We choose the models on CUB using supervised pre-trained ResNet-50. As shown in Table 3, only 2.27% images exist where the model trained from scratch makes correct predictions while

Table 2: Statistics of hyper-parameters.

Hyper-parameters	CUB	Cars	Aircraft	CIFAR10	CIFAR100	Vegetable	ISIC	Caltech101
Epochs	50	20				50		
Iterations per epoch	500			200		500		
Batch size	48			128		48		
Learning rate (LR)	0.01							
LR schedule	MultiStep	CosineAnnealing				MultiStep		
K	40							
Dimension of $\hat{Z}$	512							
Patch size	64							
Trade-off α	1.0							
Trade-off β	$5e-3$							

Table 3: Percentages of testing images of CUB, on which the model trained from scratch makes correct predictions while fine-tuning methods misclassifies.

Dataset	Vanilla fine-tuning	Bi-tuning	Concept-tuning
CUB	3.08	2.74	**2.27**

Concept-tuning misclassifies, much better than Bi-tuning.

F.2. Full table of results

Table 4 shows the experimental results using ViT/B-16 pre-trained by MAE and MoCo v3. Besides, in the main paper we report the experimental results of five methods on three datasets using ResNet-50 pre-trained by different pre-training methods. We further report other methods in Table 5.

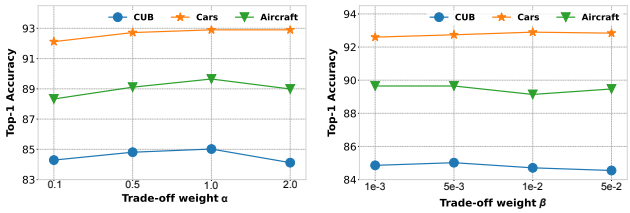
Table 4: Results on fine-tuning ViT/B-16 by MAE and MoCo v3.

Pre-training approach	Fine-tuning method	CUB	Car	Aircraft
MAE	Vanilla fine-tuning	77.80	87.35	86.98
	Bi-tuning	79.29	88.97	87.91
	Core-tuning	79.96	89.54	88.00
	Ours 2	**80.62**	**90.05**	**90.44**
MoCo v3	Vanilla fine-tuning	80.34	83.51	85.60
	Bi-tuning	83.76	87.70	88.45
	Core-tuning	81.00	89.52	89.31
	Ours 2	**84.57**	**90.91**	**90.89**

F.3. More ablation studies

Influences of the trade-off weight α. In previous experiments, we set the trade-off weights α as 1.0 for all datasets. In this section, we further analyze how this term influences performance. Larger α will more strongly pull rare features, in the meantime, will increase the risk of overfitting. The results on three datasets in Fig. 4a show clear peaks, which supports the existence of a trade-off.

Influences of the trade-off weight β. In previous experiments, we set the trade-off weights β as $5e-3$ by default for all datasets. Here, we further analyze how this term influences our methods. We try different values on three datasets, and the results are shown in Fig. 4b. Even though the best β may vary on different datasets, the default value $5e-3$ is enough to obtain good performances.



(a) (b)

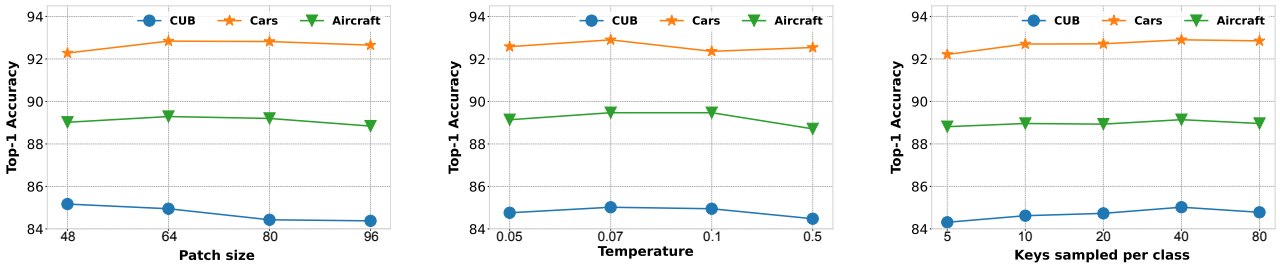
Figure 4: Analysis of α and β in our methods using supervised pre-trained ResNet-50.

Influences of the patch size. Selection of an appropriate patch size is also important in our methods and the best patch size varies from dataset to dataset. While features in larger patch sizes contain more information and are more discriminative for contrastive learning, a sufficiently large patch that occupies most regions of an object and likely fuses both rare and non-rare features decreases the effectiveness of resolving rare features. For example, as shown in Fig. 5a, the best patch size for CUB is 48 while the best one for Cars and Aircraft is 64 because the objects in CUB occupy relatively smaller regions in the images. Furthermore, when the patch size increases to 96, the performances will decrease a lot (*e.g.*, performance drops from 85.17% to 84.38% on CUB).

Influences of the temperature τ**.** In previous experiments, we follow the implementation of supervised contrastive learning [6] to set the temperature τ as 0.07. In this section, we conduct experiments to explore the influences of τ. As shown in Fig. 5b, τ as 0.07 obtains better performances. The potential reason is that smaller τ tends to punish more on hard samples [14] for generating more universal representations while reducing the tolerances of hard samples.

Table 5: Top-1 accuracy (%) on three datasets using four different pre-trained ResNet-50. $*$ denotes that methods can not converge.

Dataset	Method	Pre-trained method				
		MoCo-V2	SimCLR	SwAV	BYOL	Avg.
CUB	Vanilla fine-tuning	76.72 ± 0.21	76.51 ± 0.28	80.45 ± 0.32	81.29 ± 0.29	78.74
	L2SP	71.88 ± 0.44	64.55 ± 0.53	75.04 ± 0.38	76.72 ± 0.25	72.05
	DELTA	72.87 ± 0.38	$*$	$*$	$*$	$*$
	BSS	76.92 ± 0.29	76.75 ± 0.20	80.89 ± 0.35	81.53 ± 0.25	79.02
	Co-tuning	76.39 ± 0.22	76.35 ± 0.17	80.93 ± 0.24	81.72 ± 0.31	78.85
	REGSL	$*$	$*$	77.70 ± 0.29	79.13 ± 0.26	$*$
	Bi-tuning	79.48 ± 0.24	75.73 ± 0.25	81.72 ± 0.23	82.02 ± 0.29	79.74
	Core-tuning	77.93 ± 0.18	77.55 ± 0.15	80.60 ± 0.27	78.46 ± 0.18	78.64
	Ours 1	82.48 ± 0.14	78.18 ± 0.20	83.47 ± 0.22	83.38 ± 0.18	81.88
	Ours 2	$\mathbf{82.53 \pm 0.21}$	$\mathbf{79.81 \pm 0.23}$	$\mathbf{84.78 \pm 0.32}$	$\mathbf{84.45 \pm 0.29}$	**82.89**
Cars	Vanilla fine-tuning	88.45 ± 0.35	84.53 ± 0.12	88.17 ± 0.21	88.99 ± 0.39	87.54
	L2SP	81.58 ± 0.28	65.25 ± 0.41	76.51 ± 0.27	81.72 ± 0.31	76.26
	DELTA	82.28 ± 0.33	$*$	$*$	$*$	$*$
	BSS	88.07 ± 0.27	91.80 ± 0.34	88.07 ± 0.31	89.28 ± 0.15	89.31
	Co-tuning	88.35 ± 0.16	91.56 ± 0.25	88.53 ± 0.17	89.45 ± 0.15	89.47
	REGSL	90.96 ± 0.19	80.03 ± 0.32	78.60 ± 0.24	85.77 ± 0.23	83.84
	Bi-tuning	90.05 ± 0.15	91.75 ± 0.18	90.49 ± 0.27	90.90 ± 0.18	90.80
	Core-tuning	90.87 ± 0.23	91.78 ± 0.26	91.84 ± 0.14	91.95 ± 0.18	91.61
	Ours 1	91.02 ± 0.11	93.27 ± 0.20	93.41 ± 0.26	93.22 ± 0.15	92.73
	Ours 2	$\mathbf{91.75 \pm 0.18}$	$\mathbf{93.36 \pm 0.23}$	$\mathbf{93.79 \pm 0.32}$	$\mathbf{93.68 \pm 0.25}$	**93.15**
Aircraft	Vanilla fine-tuning	88.60 ± 0.18	87.79 ± 0.24	83.26 ± 0.17	85.03 ± 0.15	86.17
	L2SP	86.17 ± 0.25	65.25 ± 0.52	76.27 ± 0.42	80.32 ± 0.23	77.00
	DELTA	82.28 ± 0.33	$*$	$*$	$*$	$*$
	BSS	88.63 ± 0.17	88.30 ± 0.30	83.59 ± 0.15	84.64 ± 0.20	86.29
	Co-tuning	88.63 ± 0.23	87.64 ± 0.35	83.59 ± 0.24	84.52 ± 0.16	86.10
	REGSL	81.16 ± 0.21	$*$	$*$	73.90 ± 0.37	$*$
	Bi-Tuning	89.05 ± 0.16	88.69 ± 0.17	85.69 ± 0.13	87.16 ± 0.11	87.65
	Core-Tuning	89.02 ± 0.19	89.47 ± 0.21	88.66 ± 0.34	89.74 ± 0.20	89.22
	Ours 1	$\mathbf{89.65 \pm 0.18}$	90.13 ± 0.11	91.42 ± 0.36	90.82 ± 0.22	90.50
	Ours 2	89.32 ± 0.21	$\mathbf{90.85 \pm 0.17}$	$\mathbf{91.75 \pm 0.14}$	$\mathbf{91.21 \pm 0.13}$	**90.76**



(a) (b) (c)

Figure 5: Analysis of the patch size, temperature, and keys in our methods on three datasets using supervised pre-trained ResNet-50.

Influences of the number of keys. This section discusses the influences of the number of keys used in $\mathcal{L}_r$. In contrastive learning, a larger queue is beneficial [2]. However, a large number of keys in supervised contrastive learning necessarily sacrifice the stochasticity of sampling from the queue [16], especially under limited training data (*e.g.*, up to 30 training samples per class on CUB), unfavorably easing contrastive learning and leading to saturation. Our experimental results in Fig. 5c present apparent saturation, which supports the unnecessary storage of too many keys. Note that the best number of keys varies in different datasets, and our default value of 40 also performs well.

G. Additional Visualization Results

G.1. More visualization of attentive regions

In the section, We further show the regions attended by the three models on **Aircraft** in Figure 6. Consistent with the difference in predictions, the two fine-tuning methods rely more on front power-plant features.

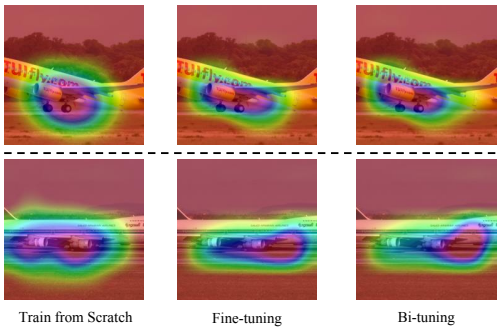


Figure 6: Exemplar attentive regions of the model trained (a) from scratch, by (b) fine-tuning and (c) bi-tuning, where only the first column predicts correctly.

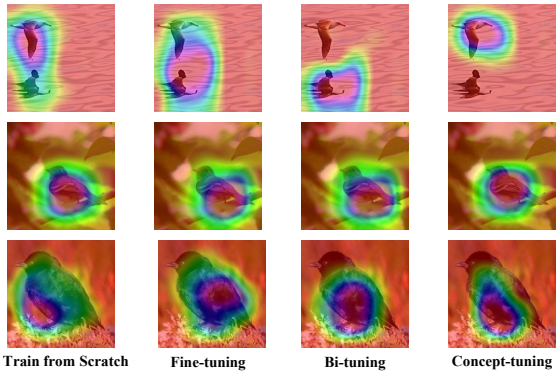


Figure 7: CAM visualization of four methods.

G.2. More visualization of Concept-tuning

To better understand the effectiveness of our methods, we provide several examples of CAMs in different methods, as shown in Fig. 7. Influenced by the pre-trained model, fine-tuning and Bi-tuning will be attracted by the pre-training features and make wrong predictions, while Concept-Tuning could resolve the negative effects and predicts correctly. For example, fine-tuning and Bi-tuning neglect the chest features as shown in the second row of Fig. 7, while Concept-tuning attend regions closer to the model trained from scratch.

References

[1] Pierre Baldi and Peter Sadowski. The dropout learning algorithm. *Artificial intelligence*, 210:78–122, 2014. 1

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607, 2020. 2, 5

[3] Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019. 2

[4] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPRW*, pages 178–178, 2004. 2

[5] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *NeurIPS*, 33, 2020. 2

[6] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *NeurIPS*, 2020. 4

[7] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCVW*, pages 554–561, 2013. 2

[8] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2

[9] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. arXiv preprint arXiv:1306.5151. 2

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035. 2019. 2

[11] Yushan Feng Saihui Hou and Zilei Wang. Vegfru: A domain-specific dataset for fine-grained visual categorization. ICCV. 2

[12] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 2

[13] Kaichao You, Zhi Kou, Mingsheng Long, and Jianmin Wang. Co-tuning for transfer learning. *NeurIPS*, 33, 2020. 2, 3

[14] Chaoning Zhang, Kang Zhang, Trung X Pham, Axi Niu, Zhinan Qiao, Chang D Yoo, and In So Kweon. Dual temperature helps contrastive learning without many negative samples: Towards understanding and simplifying moco. In *CVPR*, pages 14441–14450, 2022. 4

[15] Yifan Zhang, Bryan Hooi, Dapeng Hu, Jian Liang, and Jiashi Feng. Unleashing the power of contrastive self-supervised visual models via contrast-regularized fine-tuning. *arXiv preprint arXiv:2102.06605*, 2021. 2

[16] Jincheng Zhong, Ximei Wang, Zhi Kou, Jianmin Wang, and Mingsheng Long. Bi-tuning of pre-trained representations. *arXiv preprint arXiv:2011.06182*, 2020. 2, 3, 5