Sample-specific Masks for Visual Reprogramming-based Prompting

Chengyi Cai [1] Zesheng Ye [1] Lei Feng [2] Jianzhong Qi [1] Feng Liu [1]

Abstract

Visual reprogramming (VR) is a prompting technique that aims to re-purpose a pre-trained model (e.g., a classifier on ImageNet) to target tasks (e.g., medical data prediction) by learning a *small-scale pattern* added into input images instead of tuning considerable parameters within the model. The location of the pattern within input samples is usually determined by a pre-defined mask *shared across all samples*. In this paper, we show that the shared mask potentially limits VR's generalization and increases its approximation error due to the lack of sample-level adaptation. Motivated by this finding, we design a new framework for VR called *sample-specific multi-channel masks* (SMM). Specifically, SMM employs a lightweight ConvNet and patch-wise interpolation to generate sample-specific three-channel masks instead of a shared and pre-defined mask. Since we generate different masks for individual samples, SMM is theoretically shown to reduce approximation error for the target tasks compared with existing state-of-the-art VR methods. We also empirically demonstrate its performance gain on both ResNet and ViT. The success of SMM further highlights the broader applicability of VR in leveraging the latent knowledge of pre-trained models for various target tasks. Our code is available at `https://github.com/tmlr-group/SMM`.

1. Introduction

Recent studies have shown that, by taking advantage of and re-purposing well-trained/pre-trained models, one can address new tasks (i.e., *target tasks*) without training a task-specific model from scratch (Basu et al., 2023; Kossen et al., 2023; Mondal et al., 2023). In visual tasks, due to the expensive training costs even just to finetune pre-trained models, *visual reprogramming* (VR) (Neekhara et al., 2022; Wang et al., 2022; Chen et al., 2023; Tsao et al., 2024), or adversarial reprogramming (Elsayed et al., 2018; Tsai et al., 2020), has been proposed to reuse pre-trained models on target tasks. Concretely, VR is a prompting method that fixes a pre-trained model and only alters the input space by adding some learnable patterns (usually some noise) to target images. The location of the patterns to be learned is usually determined by a pre-defined binary mask that is *shared across all samples* (Elsayed et al., 2018; Yang et al., 2021; Tsai et al., 2020; Bahng et al., 2022). The key benefit of VR methods is that learning the pattern whose size is around the image size requires much less computing resource than finetuning considerable parameters within the model, posing VR as a promising research area in using pre-trained models (Chen et al., 2023; Tsao et al., 2024).

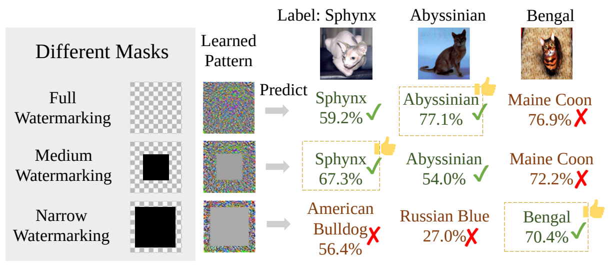


Figure 1. Drawback of shared masks over individual images. We demonstrate the use of watermarking (Wang et al., 2022), a representative VR method, to re-purpose an ImageNet-pretrained classifier for the OxfordPets dataset, with different shared masks (full, medium, and narrow) in VR. An evaluation of classification confidence across three cat images — Sphynx, Abyssinian, and Bengal — indicates a sample-specific mask preference: Sphynx with medium, Abyssinian with full, and Bengal with narrow. It shows that different masks are needed for individual images.

In this paper, we show that the shared mask often leads to poor generalization capability of VR, as demonstrated in Figures 1 and 2. In both figures, we use a representative VR method, watermarking (Bahng et al., 2022), to re-purpose an ImageNet-pretrained classifier to classify images in the OxfordPets datasets (Parkhi et al., 2012). In Figure 1, we first find that the optimal masks vary among individual images.

[1]School of Computing and Information Systems, The University of Melbourne [2]Information Systems Technology and Design Pillar, Singapore University of Technology and Design. Correspondence to: Feng Liu <fengliu.ml@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

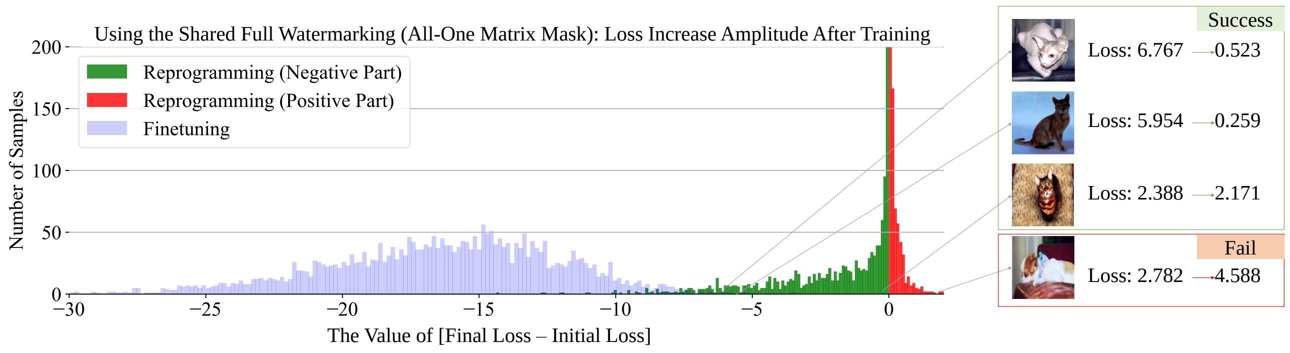


Figure 2. Drawback of shared masks in the statistical view. Optimal learning methods like finetuning usually result in loss decreases for all samples (see the blue part). But when applying the same mask in reprogramming, part of the loss changes are observed to be positive (see the red part) according to the distribution of [final loss - initial loss], which means the training loss for some samples even rises.

We apply three kinds of masks (full, medium, and narrow) in watermarking. By observing the classification confidence on three cat images: Sphynx, Abyssinian, and Bengal, we see that the medium mask is the best for Sphynx, the full mask for Abyssinian, and the narrow mask for Bengal. This suggests that different masks are needed for individual images. In Figure 2, we then find that watermarking with a single shared mask may cause the training loss of many individual samples to rise (see the red part in Figure 2). This phenomenon reveals that VR methods' learning capacity is much less than finetuning all parameters of the pre-trained model (see the blue part in Figure 2).

The examples above show a significant disadvantage of using a single shared mask for VR. This motivates our new VR framework called *sample-specific multi-channel masks* (SMM). SMM replaces the *fixed* binary mask applied in existing works with *generative* three-channel masks that can vary across different samples (shown in Figure 3).

SMM has two modules: a mask generator module and a patch-wise interpolation module. The mask generator is a lightweight *convolutional neural network* (CNN) that takes resized individual target-domain images (i.e., samples) as the input and outputs different masks for each sample. The last layer of the generator is designed to generate a three-channel mask, which allows better performance for both rich-color images (i.e., CIFAR10/100 (Krizhevsky, 2009)) and monotonous-color images (i.e., SVHN (Yuval, 2011)). Since the generated masks should be the same size as the pattern to be learned, when the size of masks is inconsistent with that of the pattern, the patch-wise interpolation module will be utilized to re-scale the generated masks to fit the pattern, facilitating the training process of the mask generator (detailed in Section 3).

To understand why SMM is effective, we theoretically analyze the approximation error of different hypothesis sets for VR. Three hypothesis sets are considered: shared pattern with a pre-defined binary mask, sample-specific patterns without masks, and our SMM. We show that SMM has a smaller approximation error (Proposition 4.3), which confirms the effectiveness of SMM.

To further substantiate the efficacy of SMM, we conduct empirical evaluations spanning 11 widely used datasets, incorporating ablation studies that discern the impact of individual SMM components. This is complemented by analysis and interpretations of the generated masks, alongside a comparative visualization of feature spaces. Notably, we demonstrate the effectiveness of SMM with both pre-trained ResNet (He et al., 2016) and ViT (Dosovitskiy et al., 2020) (Table 1 and 2), validating that SMM is compatible with commonly used classifier architectures.

Both the theoretical analysis and promising experimental results provide solid evidence that, when powered by SMM, VR can efficiently leverage knowledge within a well-trained/pre-trained model for various target tasks, shedding new light on the explanatory analysis of VR and opening avenues for future research.

2. Preliminaries and Related Works

2.1. Problem Setting of Model Reprogramming

Model reprogramming (Chen, 2022) offers an efficient transfer learning paradigm for adapting pre-trained models to resource-constrained target tasks. This paradigm re-purposes existing knowledge by strategically transforming inputs and outputs, bypassing extensive model parameter finetuning. In what follows, we will present a formal problem setting for model reprogramming.

Let $\mathcal{D}_{\mathrm{T}}$ represent the data distribution of a target task defined over $\mathcal{X}^{\mathrm{T}} \times \mathcal{Y}^{\mathrm{T}}$, where $\mathcal{X}^{\mathrm{T}} \subseteq \mathbb{R}^{d_{\mathrm{T}}}$ is the data space and

$\mathcal{Y}^{\mathrm{T}} = \{1, \dots, k_{\mathrm{T}}\}$ is the label space, and let $\{(x_i^{\mathrm{T}}, y_i^{\mathrm{T}})\}_{i=1}^{n}$ be the observations of $\mathcal{D}_{\mathrm{T}}$ (i.e., the training set in the target task). Meanwhile, we have a pre-trained model $f_{\mathrm{P}} : \mathcal{X}^{\mathrm{P}} \to \mathcal{Y}^{\mathrm{P}}$, where $\mathcal{X}^{\mathrm{P}} \subseteq \mathbb{R}^{d_{\mathrm{P}}}$ and $\mathcal{Y}^{\mathrm{P}}$ (s.t. $|\mathcal{Y}^{\mathrm{T}}| \le |\mathcal{Y}^{\mathrm{P}}|$, with the label space of the pre-trained task being larger than that of the target task) represent the data and label spaces used for training f_{P}. Then, in model reprogramming, the training objective can be formulated as

$$\min_{\theta \in \Theta, \omega \in \Omega} \frac{1}{n} \sum_{i=1}^{n} \ell(f_{\mathrm{out}}(f_{\mathrm{P}}(f_{\mathrm{in}}(x_i^{\mathrm{T}}|\theta))|\mathcal{Y}^{\mathrm{P}}_{\mathrm{sub}}, \omega), y_i^{\mathrm{T}}), \quad (1)$$

where $f_{\mathrm{in}}(.|\theta) : \mathcal{X}^{\mathrm{T}} \mapsto \mathcal{X}^{\mathrm{P}}, f_{\mathrm{out}}(.|\mathcal{Y}^{\mathrm{P}}_{\mathrm{sub}}, \omega) : \mathcal{Y}^{\mathrm{P}}_{\mathrm{sub}} \mapsto \mathcal{Y}^{\mathrm{T}}$ are the input transformation and output label mapping function with parameters $\theta \in \Theta$ and $\omega \in \Omega$, $\mathcal{Y}^{\mathrm{P}}_{\mathrm{sub}} \subseteq \mathcal{Y}^{\mathrm{P}}$ can be determined by different methods (Elsayed et al., 2018; Tsai et al., 2020; Chen et al., 2023), and $\ell : \mathcal{Y}^{\mathrm{T}} \times \mathcal{Y}^{\mathrm{T}} \mapsto \mathbb{R}^{+} \cup \{0\}$ is a loss function. Reprogramming techniques have been widely applied in visual (Elsayed et al., 2018; Tsai et al., 2020), text (Neekhara et al., 2018; Hambardzumyan et al., 2021), speech (Yang et al., 2021; 2023; Yen et al., 2023), music (Hung et al., 2023), and cross-modal tasks (Neekhara et al., 2022) in the past few years.

In the context of visual tasks, reprogramming has demonstrated potential in bio-medical measurement (Tsai et al., 2020), machine learning fairness (Zhang et al., 2022), as well as out-of-distribution detection through watermarking (Wang et al., 2022). Moving beyond application prospects, we next discuss the technical details of the specific input and output mapping functions (f_{in} and f_{out}).

2.2. Prompting and Input Visual Reprogramming

General prompting methods in visual tasks, predominantly applied to the ViT architecture (Dosovitskiy et al., 2020), introduce extra parameters to a pre-trained model for enhanced training efficiency. Prompts are flexible in their placement. For example, *visual prompt tuning* (Jia et al., 2022) positions prompts alongside image embedding before the encoder layers, while *effective and efficient visual prompt tuning* (Han et al., 2023) extends this by incorporating parameters within self-attention layers as well. *Transformer with hierarchical prompting* (Wang et al., 2023) also learns prompt tokens to represent the coarse image classes.

Meanwhile, prompting goes beyond vision foundation models to vision-language frameworks such as CLIP (Radford et al., 2021). Methods like CoOP (Zhou et al., 2022b) and CoCoOP (Zhou et al., 2022a) replace textual prompts with learnable vectors for enhanced adaptability to the target task, conditioned on input images. MaPLe (Khattak et al., 2023) further bridges vision and language by learning layer-specific mapping functions. These methods vary from each other in terms of both prompt placements and functions.

In contrast, VR provides a model-agnostic prompting technique, by adding trainable noise to the input image patterns before the forward propagation, without altering their visual essence. Originally proposed by Elsayed et al. (2018), VR has been evolving to include padding-based methods (Tsai et al., 2020; Chen et al., 2023) and watermarking that facilitate downstream target tasks (Bahng et al., 2022). AutoVP (Tsao et al., 2024) stands out with its scalable pre-padding images. A critical limitation in existing VR research is the use of *shared* noise patterns across *all target samples*, neglecting sample-level characteristics and compromising generalization. We propose SMM to manage this gap.

2.3. Output Mapping of Reprogramming

Learning-based output mapping, i.e., model f_{out}, as proposed by Chen et al. (2023), can be simplified as a one-to-one mapping from a subset of $\mathcal{Y}^{\mathrm{P}}$ to $\mathcal{Y}^{\mathrm{T}}$. Therefore, no additional parameters are required. One implementation of this mapping is *random label mapping* (Rlm), where f_{out} is a randomly assigned injective function (Elsayed et al., 2018; Chen et al., 2023), formulated as

$$f^{\mathrm{Rlm}}_{\mathrm{out}}(y|\mathcal{Y}^{\mathrm{P}}_{\mathrm{sub}}) = \mathtt{rand}(\{0, 1, ..., k^{\mathrm{T}}\}), \quad (2)$$

where $\mathtt{rand}(\{0, 1, ..., k^{\mathrm{T}}\})$ means randomly selecting one element from the set $\{0, 1, ..., k^{\mathrm{T}}\}$, and $\mathcal{Y}^{\mathrm{P}}_{\mathrm{sub}}$ is of the same size with $\mathcal{Y}^{\mathrm{T}}$ (i.e., k^{T}), randomly chosen from $\mathcal{Y}^{\mathrm{P}}$ prior to the minimization of Eq. (1). Note that, since $f^{\mathrm{Rlm}}_{\mathrm{out}}$ is injective, it ensures $f^{\mathrm{Rlm}}_{\mathrm{out}}(y_1|\mathcal{Y}^{\mathrm{P}}_{\mathrm{sub}}) \ne f^{\mathrm{Rlm}}_{\mathrm{out}}(y_2|\mathcal{Y}^{\mathrm{P}}_{\mathrm{sub}})$ for two distinct elements $y_1 \ne y_2$.

Other representative output-mapping methods determine $\mathcal{Y}^{\mathrm{P}}_{\mathrm{sub}}$ and f_{out} for different target tasks. For example, one is based on the frequency of label assignment in the pre-trained model and the target data (Tsai et al., 2020), called *frequent label mapping* (Flm). Chen et al. (2023) propose *iterative label mapping* (Ilm) that updates f_{out} in each training iteration, reflecting changes in label mapping throughout the learning of f_{in}. Detailed procedures and the pseudo-code of $f^{\mathrm{Flm}}_{\mathrm{out}}$ and $f^{\mathrm{Ilm}}_{\mathrm{out}}$ are in Appendix A.4.

3. Sample-specific Multi-channel Masks

We focus on f_{in}, while treating f_{out} as a non-parametric mapping, in line with Chen et al. (2023). We thus limit our discussion of trainable parameters to $\theta \in \Theta$ in Eq. (1). A flowchart in Appendix A.1 provides an overview of the problem structure of Input VR.

3.1. Framework of SMM

To allow both shared parameters over the whole dataset and variability among individual samples, it is intuitive to

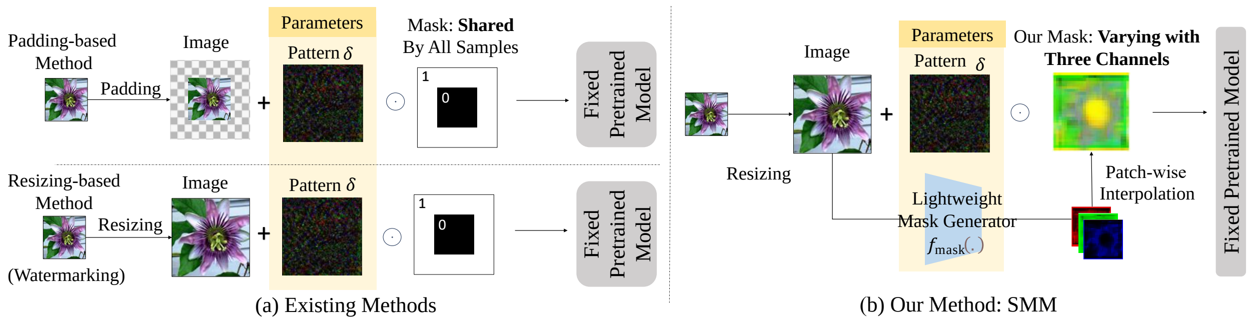


Figure 3. Comparison between (a) existing methods and (b) our method. Previous padding-based reprogramming adds zeros around the target image, while resizing-based reprogramming adjusts image dimensions to fit the required input size. Both methods use a pre-determined *shared* mask to indicate the valid location of pattern δ. Our method, on the other hand, takes a more dynamic and tailored approach. We resize each target image and apply a different three-channel mask accordingly, driven by a lightweight f_{mask} with an interpolation up-scaling module, allowing for more variability in individual samples.

consider the following VR hypothesis:

$$f_{\text{in}}(x_i|\phi,\delta) = r(x_i) + \delta \odot f_{\text{mask}}(r(x_i)|\phi), \quad (3)$$

where $r : \mathcal{X}^{\text{T}} \to \mathbb{R}^{d_{\text{P}}}$ is the resizing function, typically implemented as bilinear interpolation upsampling (Wikipedia contributors, 2023) that scales image dimension from d_{T} to d_{P}, and $r(x_i) \in \mathbb{R}^{d_{\text{P}}}$ is the resized image corresponding to x_i. The mask generation function $f_{\text{mask}} : \mathbb{R}^{d_{\text{P}}} \to \mathbb{R}^{d_{\text{P}}}$, parameterized by $\phi \in \Phi$, produces a mask indicating the noise placements for each image. We denote a trainable noise pattern added to the image by $\delta \in \mathbb{R}^{d_{\text{P}}}$. The rationale for applying this hypothesis is elaborated in Proposition 4.3 and validated in ablation studies (cf. Table 3). This casts the training objective of our SMM framework ($\theta = \{\phi, \delta\}$) to find the optimal ϕ^* and δ^* such that

$$\begin{aligned}\mathop{\arg\min}_{\phi\in\Phi,\delta\in\mathbb{R}^{d_{\text{P}}}} \ & \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}_{\text{T}}}[\ell(f_{\text{out}}(f_{\text{P}}(r(x_i)+ \\ & \delta \odot f_{\text{mask}}(r(x_i)|\phi))), y_i)].\end{aligned} \quad (4)$$

Note that δ is *shared* by all images in the dataset following Bahng et al. (2022) and Chen et al. (2023), while f_{mask} *uniquely* generates sample-specific multi-channel masks for each individual image, enabling sample-specific adaptation.

Figure 3 illustrates the workflow of SMM, as well as previous padding-based and resizing-based (i.e., watermarking) VR methods. Compared with previous works, SMM features $f_{\text{mask}}(\cdot|\phi)$, integrating a *mask generator module* and a *patch-wise interpolation module*. Concretely, SMM starts by resizing target images, followed by their processing through the mask generator to create corresponding three-channel masks. For generated masks smaller than the pattern size, the patch-wise interpolation module performs upsampling, which omits the derivation step in back-propagation and facilitates training. Afterward, the learnable pattern δ is multiplied with the mask on a pixel-wise basis and added to the image. The resulting image is fed into the *fixed* pretrained classifier. We discuss further details on the mask generator (Section 3.2), the patch-wise interpolation module (Section 3.3), and the overall learning strategy presented in Eq. (4) (Section 3.4).

3.2. Lightweight Mask Generator Module

The mask generator f_{mask} is supposed to output a mask that has the same size as the input image while prioritizing different locations for δ to allow more variability. We employ a CNN as the mask generator. This choice stems from the proficiency of CNNs in mirroring localized visual perception (He et al., 2016) with fewer parameters than most deep learning structures, e.g., multilayer perceptrons.

The input of CNN is a resized image $r(x_i)$. Applying our bespoke CNN architecture shown in Appendix A.2, the output will be a three-channel mask with dimensions $\lfloor \frac{H}{2^l} \rfloor \times \lfloor \frac{W}{2^l} \rfloor$, where H and W denote image height and width, respectively, and l denotes the number of pooling layers. The analysis of input/output sizes and parameter quantity statistics are in Appendix A.2.

3.3. Patch-wise Interpolation Module

The patch-wise interpolation module upscales CNN-generated masks from $\lfloor \frac{H}{2^l} \rfloor \times \lfloor \frac{W}{2^l} \rfloor$ back to the original size $H \times W$ per channel (it is omitted when $l = 0$). Considering the inherent consistency in adjacent image areas and the benefits of concise operations for gradient calculations, we employ a grid of $\lfloor \frac{H}{2^l} \rfloor \times \lfloor \frac{W}{2^l} \rfloor$ patches in the upsampling process, each sized $2^l \times 2^l$, ensuring the same values within each patch, with non-divisible cases mirroring the closest patches. Therefore, after obtaining the output of CNN, we

Algorithm 1 Visual Reprogramming with SMM

```
1: Input: Pre-trained model $f_{\rm P}$, loss $\ell$, label-mapping function $f_{\rm out}^{(j)}$ for iteration $j$, target domain training data $\{(x_i, y_i)\}_{i=1}^n$, maximum number of iterations $E$, learning rate $\alpha_1$ for $\delta$ and $\alpha_2$ for $\phi$
2: Output: Optimal $\delta^*, \phi^*$
3: Initialize $\phi$ randomly; set $\delta \leftarrow \{0\}^{d_{\rm P}}$
4: for $j = 1$ to $E$ do
5:     # Step1: Compute individual marks using the mask generator
       # Step2: Resize masks using the patch-wise interpolation module
       $f_{\rm in}(x_i; \delta, \phi) \leftarrow r(x_i) + \delta \odot f_{\rm mask}(r(x_i)|\phi)$, $\forall i = 1, 2, ..., n$
6:     # Compute the classification loss
       $L(\delta, \phi) \leftarrow \frac{1}{n}\sum_{i=1}^n \ell(f_{\rm out}^{(j)}(f_{\rm P}(f_{\rm in}(x_i; \delta, \phi))), y_i)$
7:     $\delta \leftarrow \delta - \alpha_1 \nabla_\delta L(\delta, \phi)$
8:     $\phi \leftarrow \phi - \alpha_2 \nabla_\phi L(\delta, \phi)$
9: end for
```

enlarge each pixel to $2^l \times 2^l$ pixels by padding the same value of a pixel to its surrounding areas within the patch.

Unlike traditional interpolation methods which may introduce complicated derivation computations, our module simplifies the training by directly assigning values. The advantage of patch-wise interpolation over traditional interpolation methods will be discussed in Appendix A.3. The effect of patch size 2^l will be discussed in Section 5.

3.4. Learning Strategy

The learning process for the shared noise pattern δ and the mask generator $f_{\rm mask}$ is shown in Algorithm 1. The parameters δ and ϕ are iteratively updated in each epoch. To mitigate the impact of initialization, δ is set to be a zero matrix before training, noted as $\{0\}^{d_{\rm P}}$.

4. Understanding Masks in Visual Reprogramming for Classification

In this section, we will demonstrate that SMM enables stronger model learning capacity than previous representative VR methods, via showing reduced approximation error in the *probably approximately correct* (PAC) learning framework (Kearns & Vazirani, 1994). We first present the definition of the approximation error in PAC learning.

Definition 4.1 (Approximation Error)**.** Consider an input space $\mathcal{X}$, a discrete label space $\mathcal{Y}$, a random variable (X, Y) whose distribution $\mathcal{D}$ is defined on $\mathcal{X} \times \mathcal{Y}$ with a joint probability density function $p(x, y)$, and a hypothesis space $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y}\}$. The approximation error of $\mathcal{F}$ on $\mathcal{D}$ is

$$\mathrm{Err}_{\mathcal{D}}^{\rm apx}(\mathcal{F}) = \inf_{f \in \mathcal{F}} \mathbb{E}_{(X,Y)\sim\mathcal{D}} \ell(f(X), Y) - R_{\mathcal{D}}^*, \quad (5)$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+ \cup \{0\}$ is a loss function, and $R_{\mathcal{D}}^*$ is the Bayes risk (Snapp & Xu, 1995) on $\mathcal{D}$ defined by

$$R_{\mathcal{D}}^* = \int_{\mathcal{X}} \Big[1 - \sup_{y \in \mathcal{Y}} \Pr(y|x)\Big] p_X(x) dx. \quad (6)$$

Here, $\Pr(y|x)$ is the posterior probability of class y conditioned on observing x, and $p_X(x) = \sum_{y \in \mathcal{Y}} p(x, y)$ is the marginal distribution of X.

The approximation error of a hypothesis space $\mathcal{F}$ measures the closeness of the minimum achievable error by $\mathcal{F}$ to the theoretical minimum error on distribution $\mathcal{D}$. In general, increasing the complexity of $\mathcal{F}$ tends to reduce the approximation error. In the following theorem, we show a connection between two approximation errors when hypothesis spaces exhibit a subset relation.

Theorem 4.2. *Given an input space $\mathcal{X}$, a discrete label space $\mathcal{Y}$, and a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, if there are two hypothesis spaces $\mathcal{F}_1 \subseteq \{f : \mathcal{X} \to \mathcal{Y}\}$ and $\mathcal{F}_2 \subseteq \{f : \mathcal{X} \to \mathcal{Y}\}$ satisfying that $\mathcal{F}_1 \subseteq \mathcal{F}_2$, then we have $\mathrm{Err}_{\mathcal{D}}^{\rm apx}(\mathcal{F}_1) \geq \mathrm{Err}_{\mathcal{D}}^{\rm apx}(\mathcal{F}_2)$.*

Theorem 4.2 (proof in Appendix B.1) shows that understanding the subset relation between two hypothesis spaces is key to deriving their connections in their approximation errors. Next, we will define two hypothesis spaces: one induced by a shared mask and the other induced by SMM.

Hypothesis Space Induced by A Shared Mask. VR methods with a shared mask (Chen, 2022; Bahng et al., 2022) assume that, for each sample x_i, the mask is a constant matrix $M \in \{0, 1\}^{d_{\rm P}}$. Thus, given a fixed pre-trained model $f_{\rm P}$ and a fixed output mapping function $f_{\rm out}$ (for simplicity, we use $f'_{\rm P}$ to represent $f_{\rm out} \circ f_{\rm P}$ in this section), the hypothesis space induced by a shared mask is

$$\mathcal{F}^{\rm shr}(f'_{\rm P}) = \{f | f(x) = f'_{\rm P}(r(x) + M \odot \delta), \forall x \in \mathcal{X}\},$$

where $\delta \in \mathbb{R}^{d_{\rm P}}$. In padding-based reprogramming methods, M is a fixed mask determined by the location of the target image (Chen, 2022). The locations where x_i is placed – usually the center of $r(x_i)$ – are denoted as $\{i : M_i = 0\}$, which are excluded from further training. The rest of the locations, denoted by $\{i : M_i = 1\}$, indicate trainable parameters δ. In watermarking-based methods (Bahng et al., 2022), x_i is up-sampled to $r(x_i)$, and $\{i : M_i = 1\}$ denotes effective locations of δ added to $r(x_i)$.

Hypothesis Space Induced by SMM. Based on Eq. (4), we can obtain the hypothesis space used in SMM:

$$\mathcal{F}^{\rm smm}(f'_{\rm P}) = \{f | f(x) = f'_{\rm P}(r(x) + f_{\rm mask}(r(x)) \odot \delta), \forall x \in \mathcal{X}\}.$$

Table 1. Performance Comparison of Different Input Reprogramming Methods on Pre-trained ResNet (Mean % ± Std %, the average results across all datasets are highlighted in grey)

PRE-TRAINED	RESNET-18 (IMAGENET-1K)					RESNET-50 (IMAGENET-1K)				
METHODS	PAD	NARROW	MEDIUM	FULL	OURS	PAD	NARROW	MEDIUM	FULL	OURS
CIFAR10	65.5 ±0.1	68.6 ±2.8	68.8 ±1.1	68.9 ±0.4	**72.8** ±0.7	76.6±0.3	77.4±0.5	77.8+0.2	79.3±0.3	**81.4**±0.6
CIFAR100	24.8±0.1	36.9±0.6	34.9±0.2	33.8±0.2	**39.4**±0.6	38.9±0.3	42.5±0.2	43.8±0.2	47.2±0.1	**49.0**±0.2
SVHN	75.2±0.2	58.5±1.1	71.1±1.0	78.3±0.3	**84.4**±2.0	75.8±0.4	59.1±1.3	71.5±0.8	79.5±0.5	**82.6**±2.0
GTSRB	52.0±1.2	46.1±1.5	56.4±1.0	76.8±0.9	**80.4**±1.2	52.5±1.4	38.9±1.3	52.6±1.3	76.5±1.3	**78.2**±1.1
FLOWERS102	27.9±0.7	22.1±0.1	22.6±0.5	23.2±0.5	**38.7**±0.7	24.6±0.6	19.9±0.6	20.9±0.6	22.6±0.1	**35.9**±0.5
DTD	**35.3**±0.9	33.1±1.3	31.7±0.5	29.0±0.7	33.6±0.4	40.5±0.5	37.8±0.7	38.4±0.2	34.7±1.3	**41.1**±1.1
UCF101	23.9±0.5	27.2±0.9	26.1±0.3	24.4±0.9	**28.7**±0.8	34.6±0.2	38.4±0.2	37.2±0.2	35.2±0.2	**38.9**±0.5
FOOD101	14.8±0.2	14.0±0.1	14.4±0.3	13.2±0.1	**17.5**±0.1	17.0±0.3	18.3±0.2	18.3±0.2	16.7±0.2	**19.8**±0.0
SUN397	13.0±0.2	15.3±0.1	14.2±0.1	13.4±0.2	**16.0**±0.3	20.3±0.2	22.0±0.1	21.5±0.1	21.1±0.1	**22.9**±0.0
EUROSAT	85.2±0.6	82.8±0.4	83.8±0.5	84.3±0.5	**92.2**±0.2	83.6±0.7	83.7±0.4	85.8±0.1	86.9±0.3	**92.0**±0.6
OXFORDPETS	65.4±0.7	73.7±0.2	71.4±0.2	70.0±0.6	**74.1**±0.4	76.2±0.6	76.4±0.3	75.6±0.3	73.4±0.3	**78.1**±0.2
AVERAGE	43.91	43.48	45.04	46.85	**52.53**	49.15	46.76	49.39	52.10	**56.35**

Note that, $f_{\text{mask}}(r(x))$ belongs to $\mathbb{R}^{d_{\text{P}}}$ instead of $\{0,1\}^{d_{\text{P}}}$ like M. Next, we analyze the relation between the approximation errors of previous VR methods and SMM.

SMM Has a Lower Approximation Error. Based on Theorem 4.2 and the two hypothesis spaces above, we have the following proposition.

Proposition 4.3. *Given a fixed pre-trained model f_{P}, a fixed output mapping function f_{out}, and the definitions of $\mathcal{F}^{\text{shr}}$ and $\mathcal{F}^{\text{smm}}$, we have $\mathcal{F}^{\text{shr}}(f'_P) \subseteq \mathcal{F}^{\text{smm}}(f'_P)$. Then, based on Theorem 4.2, we have*

$$\text{Err}^{\text{apx}}_{\mathcal{D}_{\text{T}}}(\mathcal{F}^{\text{shr}}(f'_P)) \geq \text{Err}^{\text{apx}}_{\mathcal{D}_{\text{T}}}(\mathcal{F}^{\text{smm}}(f'_P)), \tag{7}$$

where $f'_P = f_{\text{out}} \circ f_{\text{P}}$, f_{mask} used in $\mathcal{F}^{\text{smm}}(f'_P)$ is a CNN demonstrated in Section 3.2, and $\mathcal{D}_{\text{T}}$ denotes the distribution of the target task.

Proposition 4.3 (see its proof in Appendix B.2) shows that SMM achieves a lower approximation error than previous shared-mask VR methods.

Estimation Error Analysis of SMM. While a lower approximation error does not suffice to guarantee a lower excess risk, the model complexity added to $\mathcal{F}^{\text{smm}}(f'_P)$ is manageable in this VR setting, since f_{mask} introduces less than 0.2% extra parameters[1] relative to f'_{P}. Such dominance of f'_{P} suggests that the estimation error of $\mathcal{F}^{\text{smm}}(f'_P)$ does not significantly exceed that of $\mathcal{F}^{\text{shr}}(f'_P)$ and is unlikely to offset its advantage in approximation error. We also provide an empirical justification from the standpoint of over-fitting to show that the additional estimation error of $\mathcal{F}^{\text{smm}}(f'_P)$ is negligible in Appendix D.3. By comparing the disparities in training and testing performance, we demonstrate that SMM does not increase the risk of model over-fitting, implying negligible estimation error.

[1]See Table 4 for statistics on network sizes.

Excess Risk Analysis of SMM. According to excess risk decomposition[2], SMM is also expected to have a lower excess risk and, consequently, superior generalization capability compared to shared-mask VR methods.

Analysis Based on Sample-specific Patterns. Having built the concept of "sample-specific", we also investigate an alternative to the proposed SMM: directly learning a *sample-specific pattern* for each image without involving δ. The hypothesis space in this context can be expressed by

$$\mathcal{F}^{\text{sp}}(f'_{\text{P}}) = \{f | f(x) = f'_{\text{P}}(r(x) + f_{\text{mask}}(r(x))), \forall x \in \mathcal{X}\}.$$

It is easy to check that $\mathcal{F}^{\text{sp}}(f'_P) \subseteq \mathcal{F}^{\text{smm}}(f'_P)$, implying that $\text{Err}^{\text{apx}}_{\mathcal{D}_{\text{T}}}(\mathcal{F}^{\text{sp}}(f'_P)) \geq \text{Err}^{\text{apx}}_{\mathcal{D}_{\text{T}}}(\mathcal{F}^{\text{smm}}(f'_P))$ (proof in Appendix B.3). Namely, SMM has a lower approximation error compared to directly learning a sample-specific pattern.

5. Experiments

Pre-trained Models and Target Tasks. Following Chen et al. (2023), we use ResNet-18, and ResNet-50 (He et al., 2016) as the pre-trained model. Performance on pre-trained ViT-B32 (Dosovitskiy et al., 2020) is also tested. All these models are pre-trained on ImageNet-1K (Deng et al., 2009), and target tasks include CIFAR10, CIFAR100 (Krizhevsky, 2009), SVHN (Yuval, 2011), GTSRB (Houben et al., 2013), Flowers102 (Nilsback & Zisserman, 2008), DTD (Cimpoi et al., 2014), UCF101 (Soomro et al., 2012), Food101 (Bossard et al., 2014), EuroSAT (Helber et al., 2019), OxfordPets (Parkhi et al., 2012), SUN397 (Xiao et al., 2010). Moreover, StanfordCars (Krause et al., 2013), which is revealed to be unsuitable for VR, is also discussed in Appendix D.4. We follow Chen et al. (2023) to split the datasets. Detailed dataset information is included in Appendix C.

[2]The excess risk is equal to the sum of approximation error and estimation error (Lauer, 2014).

Baselines. We compare our method with both padding-based (Chen et al., 2023) and resizing-based methods (Bahng et al., 2022), including: (1) *Pad*: centering the original image and adding the noise pattern around the images, (2) *Narrow*: adding a narrow padding binary mask with a width of 28 ($\frac{1}{8}$ of the input image size) to the noise pattern that covers the whole image (watermark), (3) *Medium*: adding a mask being a quarter of the size (the width is 56) of watermarks and (4) *Full*: full watermarks that cover the whole images following Wang et al. (2022). To ensure that all the methods are fairly compared, in training the shared noise pattern, we apply the same learning rate and milestones following Chen et al. (2023), with 0.01 being the initial learning rate and 0.1 being the learning rate decay. Two hundred epochs are run in total, and the $100th$ and the $145th$ epochs are the milestones. The training details of the mask generator are included in Appendix C. Experiments are run with three seeds on a single A100 GPU and the averaged test accuracy is reported. Due to page limits, we report here only the results obtained with the output mapping $f_{\text{out}}^{\text{Ilm}}$. See Appendix D.1 for the results using $f_{\text{out}}^{\text{Rlm}}$ and $f_{\text{out}}^{\text{Flm}}$.

Results on ResNets. Table 1 reports the accuracy of ResNet-18 and ResNet-50 using VR methods with the baseline shared marks and our proposed SMM method. It can be observed that our SMM yields higher accuracy for both models on all datasets tested except for ResNet-18 on DTD. The advantage is more pronounced on the datasets where the target domains are more different from the original domain, such as SVHN, Flowers102, and EuroSAT. On SVHN, 6.1% and 3.1% improvements have been witnessed for ResNet-18 and ResNet-50, respectively, while over 10% improvement is observed on the Flowers102. On DTD, the padding-based method has better results for ResNet-18. This is likely to be due to the noisy watermarks adversely impacting the texture that needs to be classified, leading to the disadvantages of resizing-based methods. Even in this challenging setting, our SMM method leads to higher accuracy when applied on the larger pre-trained model ResNet-50.

Results on ViT. Recall that input reprogramming can be applied to diverse pre-trained classifiers, we next turn our focus on ViT. Detailed in Table 2, our comparative study with baselines reveals substantial performance gains in datasets like Flowers102 (21.8%), Food101 (15.4%), and SUN397 (7.3%). These results suggest that SMM may yield even higher performance gains for larger pre-trained models. Exceptions do exist, like on EuroSAT, where all resizing-based methods show marginal under-performance, possibly a result of over-fitting on relatively simpler datasets. On UCF101, our SMM initially lags behind other strategies like narrow or medium masking but, after choosing appropriate learning rate parameters (See Appendix C), could achieve a leading 49.9% accuracy. Overall, the experiments above show the applicability of SMM over different pre-trained models and target domains. Abnormal cases of SMM in Table 1 and Table 2 will be further discussed in Appendix D.4. Next, we report ablation and parameter study results.

Table 2. Performance Comparison of Different Input Reprogramming Methods on Pre-trained ViT (Mean %, the average results are highlighted in grey)

PRE-TRAINED	VIT-B32 (IMAGENET-1K)				
METHOD	PAD	NARROW	MEDIUM	FULL	OURS
CIFAR10	62.4	96.6	96.5	95.8	**97.4**
CIFAR100	31.6	74.4	75.3	75.0	**82.6**
SVHN	80.2	85.0	87.4	87.8	**89.7**
GTSRB	62.3	57.8	68.6	75.5	**80.5**
FLOWERS102	57.3	55.3	56.6	55.9	**79.1**
DTD	43.7	37.3	38.5	37.7	**45.6**
UCF101	33.6	44.5	**44.8**	40.9	42.6
FOOD101	37.4	47.3	48.6	49.4	**64.8**
SUN397	21.8	29.0	29.4	28.8	**36.7**
EUROSAT	**95.9**	90.9	90.9	89.1	93.5
OXFORDPETS	57.6	82.5	81.0	75.3	**83.8**
AVERAGE	53.1	63.7	65.2	64.7	**72.4**

Table 3. Ablation Studies (Mean % ± Std %, with ResNet-18 as an example, and the average results are highlighted in grey)

	ONLY δ	ONLY f_{mask}	SINGLE-CHANNEL $f_{\text{mask}}^{\text{s}}$	OURS
CIFAR10	68.9±0.4	59.0±1.6	72.6±2.6	**72.8**±0.7
CIFAR100	33.8±0.2	32.1±0.3	38.0±0.6	**39.4**±0.6
SVHN	78.3±0.3	51.1±3.1	78.4±0.2	**84.4**±2.0
GTSRB	76.8±0.9	55.7±1.2	70.7±0.8	**80.4**±1.2
FLOWERS102	23.2±0.5	32.2±0.4	30.2±0.4	**38.7**±0.7
DTD	29.0±0.7	27.2±0.5	32.7±0.5	**33.6**±0.4
UCF101	24.4±0.9	25.7±0.3	28.0±0.3	**28.7**±0.8
FOOD101	13.2±0.1	13.3±0.1	15.8±0.1	**17.5**±0.1
SUN397	13.4±0.2	10.5±0.1	15.9±0.1	**16.0**±0.3
EUROSAT	84.3±0.5	89.2±0.9	90.6±0.5	**92.2**±0.2
OXFORDPETS	70.0±0.6	72.5±0.3	73.8±0.6	**74.1**±0.4
AVERAGE	46.85	42.59	49.70	**52.53**

Impact of Masking. We first investigate the impact of different masking strategies. We take three variants against the proposed SMM into comparison: (i) Shared-pattern VR $f_{\text{in}}(x_i) = r(x_i) + \delta$, with M being an all-one matrix equal to the image dimension for maximal flexibility in δ. It defaults to the "full watermarks" baseline without using f_{mask}. (ii) Sample-specific pattern without masking $f_{\text{in}}(x_i) = r(x_i) + f_{\text{mask}}(r(x_i))$. (iii) Single-channel version of SMM $f_{\text{in}}(x_i) = r(x_i) + \delta \odot f_{\text{mask}}^{\text{s}}(r(x_i))$, averaging the penultimate-layer output of the mask generator. These variants refer to the first three columns of Table 3, respectively. They help evaluate the impact of sample specificity, masking, and multiple channels introduced by SMM in the context of input VR.

As shown in Table 3, SMM consistently stands out as the

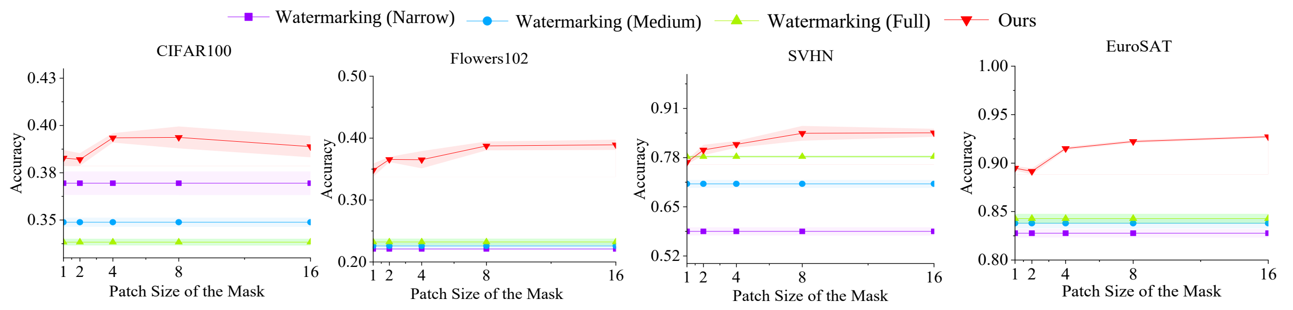


Figure 4. Comparative results of different patch sizes (2^l). ResNet-18 is used as the pre-trained model as an example.

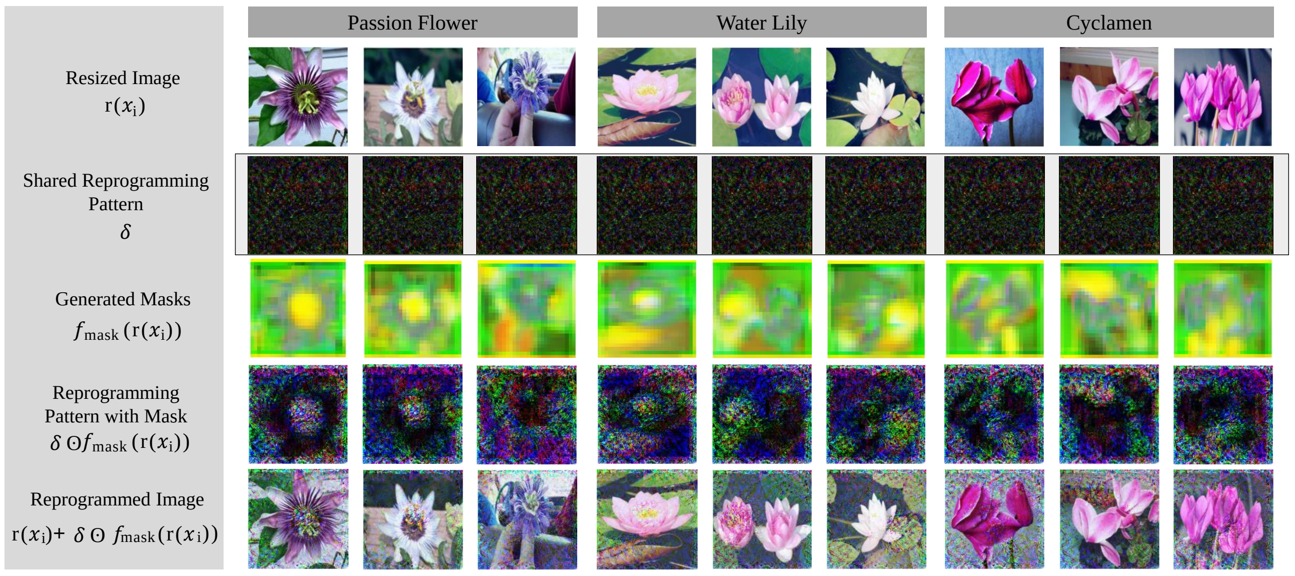
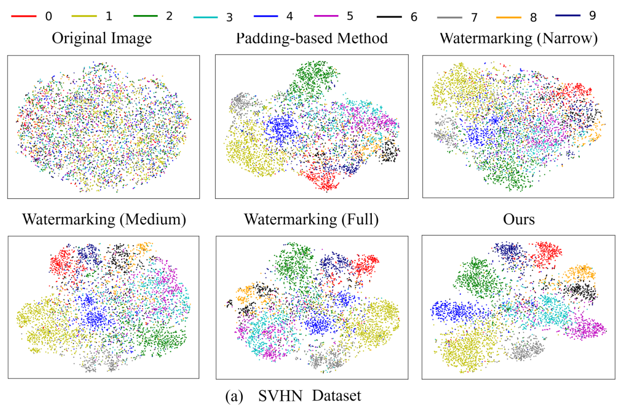

Passion Flower
Water Lily
Cyclamen
Resized Image
$\mathrm{r}(x_\mathrm{i})$
Shared Reprogramming Pattern
δ
Generated Masks
$f_\mathrm{mask}(\mathrm{r}(x_\mathrm{i}))$
Reprogramming Pattern with Mask
$\delta \odot f_\mathrm{mask}(\mathrm{r}(x_\mathrm{i}))$
Reprogrammed Image
$\mathrm{r}(x_\mathrm{i}) + \delta \odot f_\mathrm{mask}(\mathrm{r}(x_\mathrm{i}))$

Figure 5. Visual results of trained VR on the Flowers102 dataset. To show the difference in results, the original image, result image and SMM adopt histogram equalization. ResNet-18 is used as the pre-trained model as an example. Other visualization results and further analysis are included in Appendix F.

best performer on all datasets. A key observation is that only keeping shared pattern δ reduces VR effectiveness in feature-rich datasets (e.g., CIFAR10, Flowers102, and UCF101). Besides, using only f_{mask} without δ, leads to suboptimal performance on datasets with enough training data per class, including CIFAR10, SVHN, GTSRB, and SUN397. Moreover, the single-channel method is less effective, especially on datasets where images have fewer varying color palettes (e.g., GTSRB and Flowers102). Overall, we find that the shared noise in SMM boosts model performance if sufficient training data is provided, whereas the sample-specific f_{mask} enables specificity for classification tasks demanding detailed feature discrimination. Lastly, the multi-channel allows for adjusting to channel-specific priorities.

Impact of Patch Size. As an important hyperparameter in SMM, number of Max-Pooling layers, l, can vary, which means different patch sizes 2^l. Since the 5-layer mask generator neural network has at most 4 Max-Pooling layers, we examine the impact of patch sizes in $\{2^0, 2^1, 2^2, 2^3, 2^4\}$. Results are shown in Figure 4. As the patch size increases, the accuracy of the SMM increases first, followed by a plateau or decline. This suggests that overly small patches may cause over-fitting, while overly large patch sizes could result in a loss of details in SMM. We thus have set the patch size to be 8 across all datasets.

Visualization of SMM, shared patterns and output reprogrammed images. Visualization results on Flowers102 dataset is shown in Figure 5. It can be observed that when classifying passion flowers, where pedals are important for classification accuracy, the masks tend to mask out the noise



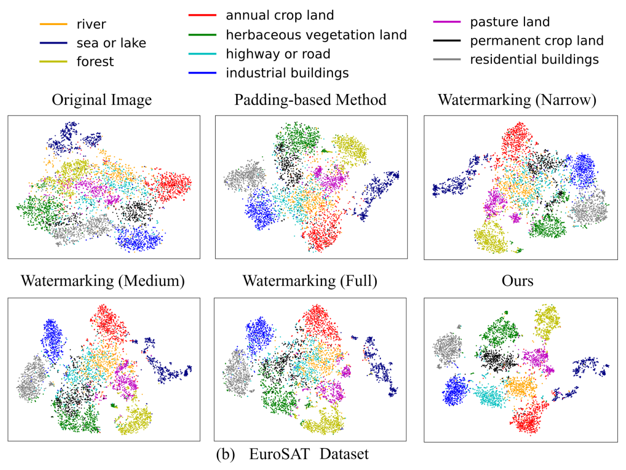


Figure 6. TSNE visualization results of the feature space on (a) SVHN and (b) EuroSAT datasets. ResNet-18 is used as the pre-trained model as an example.

pattern over the pedals, which protects useful information from being shadowed by noise. Other features such as flower pistils in passion flowers are also widely present in various similar classes such as 'oxeye', 'daisy' and 'orange dahlia', making the centers of flowers potential sources of interference in classification. Thus, for passion flowers, noise in the center of the flowers is not masked out. When classifying 'water lily', SMM will enhance the noise on interfering objects in the image. Similarly, when classifying 'cyclamen', similar stems are also commonly found in other classes such as 'gaura' and 'rose', which hinders accurate classification. Therefore, it is reasonable for SMM to introduce more noise to these interfering components. These results show that SMM is able to retain the important parts of the image and remove the interference.

Feature Space Visualization Results. Figure 6 shows the tSNE (Van der Maaten & Hinton, 2008) visualization results of the output layer feature before the label mapping layer. Before applying VR methods, the target domain's output feature space shows limited class separation. With the baseline methods, we observe enhanced but incomplete separations, where certain class pairs (such as '3, 5' and '6, 8' in SVHN, 'River' and 'highway or road' in EuroSAT) remain indistinguishable in the feature space. By applying f_{mask}, our method successfully resolves incorrectly clustered classes, underscoring the effectiveness of SMM.

Comparison with Finetuning-based Methods. In Appendix E, we compare our SMM with two prevalent finetuning approaches: finetuning fully connected layers and low-rank adaptation (Zhu et al., 2023). This comparison highlights two key benefits of input VR: (1) its efficacy in target tasks with lower-resolution images and (2) its orthogonal relationship to, yet compatibility with, finetuning methods. Additionally, Appendix E provides a comprehensive discussion on the strengths and weaknesses of Input VR in comparison to finetuning techniques.

More Experiments. The training curves are plotted and analyzed in Appendix D.2. The effectiveness of SMM when learning with different f_{out} is discussed in Appendix D.1.

6. Conclusion

In this paper, we identified significant shortcomings in the use of a shared mask across all samples in previous VR practices, notably its failure to accommodate sample diversity, leading to increased training loss of particular samples. In response, we proposed a new SMM learning framework, integrating a lightweight neural net-based mask generator to generate three-channel masks per sample, and a patch-wise interpolation module that resizes and aligns masks to model input. Both theoretical justification and experimental results validated the effectiveness of our proposed method.

Acknowledgements

CYC and FL are supported by the Australian Research Council (ARC) with grant number DE240101089, and FL is also supported by the ARC with grant number DP230101540 and the NSF&CSIRO Responsible AI program with grant number 2303037. JZQ is supported by ARC with grant number DP240101006. This research is also supported by The University of Melbourne's Research Computing Services and the Petascale Campus Initiative. We sincerely appreciate the time and dedication of the reviewers in carefully reviewing our manuscript.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Bahng, H., Jahanian, A., Sankaranarayanan, S., and Isola, P. Exploring visual prompts for adapting large-scale models. *arXiv preprint arXiv:2203.17274*, 2022.

Basu, S., Katdare, P., Sattigeri, P., Chenthamarakshan, V., Driggs-Campbell, K. R., Das, P., and Varshney, L. R. Efficient equivariant transfer learning from pretrained models. In *NeurIPS*, 2023.

Bossard, L., Guillaumin, M., and Van Gool, L. Food-101–mining discriminative components with random forests. In *ECCV*, 2014.

Chen, A., Yao, Y., Chen, P.-Y., Zhang, Y., and Liu, S. Understanding and improving visual prompting: A label-mapping perspective. In *CVPR*, 2023.

Chen, P.-Y. Model reprogramming: Resource-efficient cross-domain machine learning. *arXiv preprint arXiv:2202.10629*, 2022.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. Describing textures in the wild. In *CVPR*, 2014.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Elsayed, G. F., Goodfellow, I., and Sohl-Dickstein, J. Adversarial reprogramming of neural networks. In *ICLR*, 2018.

Hambardzumyan, K., Khachatrian, H., and May, J. Warp: Word-level adversarial reprogramming. In *ACL-IJCNLP*, 2021.

Han, C., Wang, Q., Cui, Y., Cao, Z., Wang, W., Qi, S., and Liu, D. Eˆ 2vpt: An effective and efficient approach for visual prompt tuning. *arXiv preprint arXiv:2307.13770*, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Helber, P., Bischke, B., Dengel, A., and Borth, D. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., and Igel, C. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *IJCNN*, 2013.

Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

Hung, Y.-N., Yang, C.-H. H., Chen, P.-Y., and Lerch, A. Low-resource music genre classification with cross-modal neural model reprogramming. In *ICASSP*, 2023.

Jia, M., Tang, L., Chen, B.-C., Cardie, C., Belongie, S., Hariharan, B., and Lim, S.-N. Visual prompt tuning. In *ECCV*, 2022.

Kearns, M. J. and Vazirani, U. *An introduction to computational learning theory*. MIT press, 1994.

Khattak, M. U., Rasheed, H., Maaz, M., Khan, S., and Khan, F. S. Maple: Multi-modal prompt learning. In *CVPR*, 2023.

Kossen, J., Collier, M., Mustafa, B., Wang, X., Zhai, X., Beyer, L., Steiner, A., Berent, J., Jenatton, R., and Kokiopoulou, E. Three towers: Flexible contrastive learning with pretrained image models. *arXiv preprint arXiv:2305.16999*, 2023.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3d object representations for fine-grained categorization. In *ICCV workshops*, 2013.

Krizhevsky, A. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

Lauer, F. An interactive journey into machine learning, 2014. URL https://mlweb.loria.fr/book/en/estimationapproximationerrors.html.

Mondal, A. K., Panigrahi, S. S., Kaba, S.-O., Rajeswar, S., and Ravanbakhsh, S. Equivariant adaptation of large pre-trained models. *arXiv preprint arXiv:2310.01647*, 2023.

Neekhara, P., Hussain, S., Dubnov, S., and Koushanfar, F. Adversarial reprogramming of text classification neural networks. *arXiv preprint arXiv:1809.01829*, 2018.

Neekhara, P., Hussain, S., Du, J., Dubnov, S., Koushanfar, F., and McAuley, J. Cross-modal adversarial reprogramming. In *WACV*, 2022.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*, 2008.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. Cats and dogs. In *CVPR*, 2012.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Snapp, R. and Xu, T. Estimating the bayes risk from sample data. In *NeurIPS*, 1995.

Soomro, K., Zamir, A. R., and Shah, M. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

Tsai, Y.-Y., Chen, P.-Y., and Ho, T.-Y. Transfer learning without knowing: Reprogramming black-box machine learning models with scarce data and limited resources. In *ICML*, 2020.

Tsao, H.-A., Hsiung, L., Chen, P.-Y., Liu, S., and Ho, T.-Y. AutoVP: an automated visual prompting framework and benchmark. In *ICLR*, 2024.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008.

Wang, Q., Liu, F., Zhang, Y., Zhang, J., Gong, C., Liu, T., and Han, B. Watermarking for out-of-distribution detection. *NeurIPS*, 2022.

Wang, W., Sun, Y., Li, W., and Yang, Y. Transhp: Image classification with hierarchical prompting. *arXiv preprint arXiv:2304.06385*, 2023.

Wikipedia contributors. Bilinear interpolation — Wikipedia, the free encyclopedia, 2023. URL `https://en.wikipedia.org/w/index.php?title=Bilinear_interpolation&oldid=1170546721`.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

Yang, C.-H. H., Tsai, Y.-Y., and Chen, P.-Y. Voice2series: Reprogramming acoustic models for time series classification. In *ICML*, 2021.

Yang, C.-H. H., Li, B., Zhang, Y., Chen, N., Prabhavalkar, R., Sainath, T. N., and Strohman, T. From english to more languages: Parameter-efficient model reprogramming for cross-lingual speech recognition. In *ICASSP*, 2023.

Yen, H., Ku, P.-J., Yang, C.-H. H., Hu, H., Siniscalchi, S. M., Chen, P.-Y., and Tsao, Y. Neural model reprogramming with similarity based mapping for low-resource spoken command recognition. In *INTERSPEECH*, 2023.

Yuval, N. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop*, 2011.

Zhang, G., Zhang, Y., Zhang, Y., Fan, W., Li, Q., Liu, S., and Chang, S. Fairness reprogramming. *NeurIPS*, 2022.

Zhou, K., Yang, J., Loy, C. C., and Liu, Z. Conditional prompt learning for vision-language models. In *CVPR*, 2022a.

Zhou, K., Yang, J., Loy, C. C., and Liu, Z. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 2022b.

Zhu, Y., Shen, Z., Zhao, Z., Wang, S., Wang, X., Zhao, X., Shen, D., and Wang, Q. Melo: Low-rank adaptation is better than fine-tuning for medical image diagnosis. *arXiv preprint arXiv:2311.08236*, 2023.

Appendix

A. Additional Explanation of Methods

A.1. General Procedure of Input Visual Reprogramming

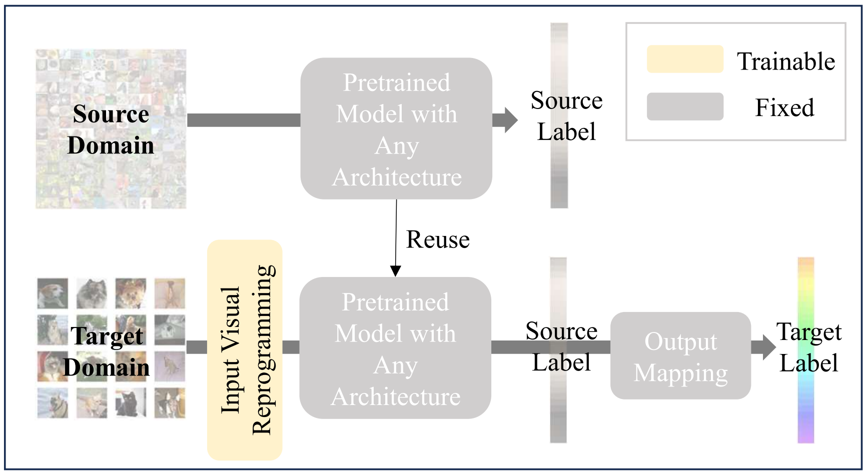

Source Domain
Pretrained Model with Any Architecture
Source Label
Trainable
Fixed
Reuse
Target Domain
Input Visual Reprogramming
Pretrained Model with Any Architecture
Source Label
Output Mapping
Target Label

Figure 7. Problem setting of input visual reprogramming. The upper part shows the source task, while the lower part shows the target task. The main focus of visual reprogramming is the trainable part marked with a yellow rectangle in the input space.

The task of VR is to reuse the fixed, well-trained model toward a target task. As shown in Figure 7, the VR module is added before the pre-trained model into the input space. To gap the difference between the source label and target label, an output mapping function without parameters is also used, taking a source label as the input and outputting a target label. Therefore, regardless of the architecture, a well-trained model on the source dataset can be transferred to the target task without editing.

A.2. Architecture of the Mask Generator and Parameter Statistics

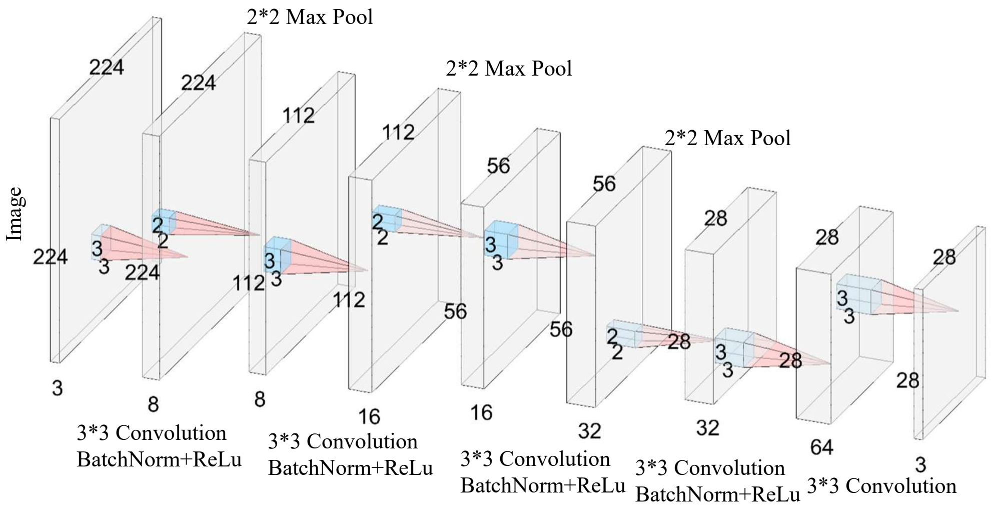


Image
2*2 Max Pool
2*2 Max Pool
2*2 Max Pool
224
224
224
224
3
8
8
112
112
112
112
16
16
56
56
56
56
32
32
28
28
28
28
64
3
28
3*3 Convolution BatchNorm+ReLu
3*3 Convolution BatchNorm+ReLu
3*3 Convolution BatchNorm+ReLu
3*3 Convolution BatchNorm+ReLu
3*3 Convolution

Figure 8. Architecture of the 5-layer mask generator designed for ResNet

Architecture of the Mask Generator. For simplicity, we only include 3×3 convolution layers and 2×2 Max-Pooling layers in the architecture. The number of channels of the last layer is set to 3 to produce a three-channel mask.

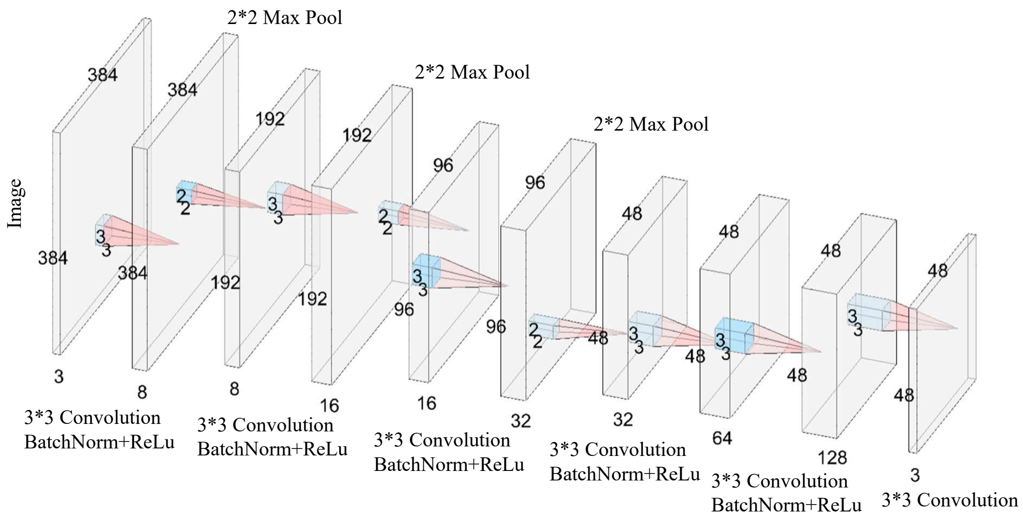


Figure 9. Architecture of the 6-layer mask generator designed for ViT

The detailed architecture of the 5-layer CNN and 6-layer CNN used in ResNet-18, ResNet-50, and ViT are shown in Figure 8 and Figure 9. Each of them contains 5 or 6 CNN layers with 3×3 kernels of padding size 1 and stride 1. Both models have 3 Max-Pooling layers.

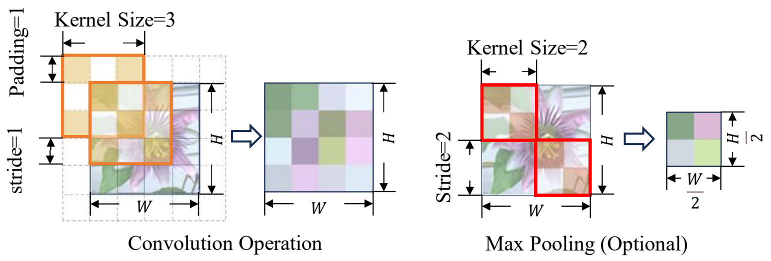


Figure 10. Changes of the image size when performing convolution and pooling operations with our stride, kernel and padding size

Discussion of Input and Output Size. To show the relationship between the sizes of the input images and the output masks, we use s, p, and k to represent the stride, padding, and kernel sizes, respectively, while H and W denote the height and the width of a certain channel. The output dimensions of the output channel after convolution or pooling are $\left\lfloor \frac{H+2p-k}{s} \right\rfloor + 1$ and $\left\lfloor \frac{W+2p-k}{s} \right\rfloor + 1$. As shown in Figure 10, when $s = 1, p = 1, k = 3$, the size of a single channel remains unchanged; when $s = 2, p = 0, k = 2$, the size of a channel is reduced by half in each dimension. In other words, by only using 3×3 convolution layers, $f_{\text{mask}}(.|\phi)$ can retain the original size of a single channel. However, if we introduce Max-Pooling layers to remove redundant information, the output size will be shrunk and another patch-wise interpolation module should be included in $f_{\text{mask}}(.|\phi)$ for resizing. Assuming that l Max-Pooling layers are used, the output size of a single channel becomes $\lfloor \frac{H}{2^l} \rfloor \times \lfloor \frac{W}{2^l} \rfloor$.

Parameter Statistics. The parameter statistics of the mask generator, f_{mask}, are summarized in Table 4. This includes a detailed breakdown of f_{mask} across different pre-trained backbone models, a relative size comparison with the watermarking reprogramming method, and the number of trainable parameters added to frozen pre-trained models by f_{mask}. From the size point of view, our mask generator is indeed lightweight and efficient: the CNN architectures contribute only 17.6% and 23.13% of the additional trainable parameters required by watermarking reprogramming. Moreover, relative to the total parameters in pre-trained models, the additional contribution of mask generators is trivial, ranging from 0.1% to 0.23% of parameters, which highlights its minimal footprint.

Table 4. Statistics of Mask Generator Parameter Size

Pre-trained	Input Image Size	f_{mask} CNN Layers	Extra Parameters of our f_{mask}	Our Extra Parameters ÷ Reprogramming Parameters	Our Extra Parameters ÷ Pre-trained Model Parameters
ResNet-18	224×224×2	5	26,499	17.60%	0.23%
ResNet-50	224×224×3	5	26,499	17.60%	0.10%
ViT-B32	384×384×3	6	102,339	23.13%	0.12%

A.3. Advantage of Patch-wise Interpolation

Table 5. Comparison of Patch-wise Interpolation and Other Interpolation Methods

		Bilinear Interpolation	Bicubic Interpolation	Ours
ResNet - 18/50	Number of Pixel Accesses (1e6)	0.602	2.408	**0.151**
	Time Per Batch (s)	0.062±0.001	0.195±0.013	**0.026±0.004**
	Require Backpropagation	Yes	Yes	**No**
ViT - B32	Number of Pixel Accesses (1e6)	1.769	7.078	**0.442**
	Time Per Batch (s)	0.165±0.009	0.486±0.026	**0.069±0.004**
	Require Backpropagation	Yes	Yes	**No**

To assess the efficiency of patch-wise interpolation, we compare it with bilinear and bicubic methods, employing the following numerical metrics for evaluation: (1) Number of Pixel Accesses: The count of times pixel values are retrieved per image during an interpolation algorithm. The fewer, the better. (2) Time Per Batch: The time cost for processing a batch of 256-sized images. The fewer, the better.

As shown in Table 5, the patch-wise interpolation module excels across all metrics. This module exclusively involves copying operations, thus avoiding floating-point calculations and avoiding backpropagation gradient computations during training. Consequently, it is more efficient.

A.4. Detailed Explanation of Ouptput Mapping Methods $f_{\text{out}}^{\text{Flm}}$ and $f_{\text{out}}^{\text{Ilm}}$

The inverse function of f_{out} regarding Flm is an injective function:

$$y_{\text{Flm}}^{\text{P}} = \arg\max_{y \in \mathcal{Y}^{\text{P}}} \Pr_{(x_i, y_i) \sim \mathcal{D}_{\text{T}}} \{ y = f_{\text{P}}(f_{\text{in}}(x_i|\theta)) | y_i = y^{\text{T}} \}, \tag{8}$$

where $y_{\text{Flm}}^{\text{P}}$ is the optimal y^{P} given the target label y^{T}, $f_{\text{P}}(f_{\text{in}}(x_i|\theta))$ is the predicted label given the input image x_i. For all images with the label y^{T}, the predicted y^{P} with the highest probability will be $y_{\text{Flm}}^{\text{P}}$ for a given y^{T}. Flm remains unchanged throughout iterations. For a specific y^{T}, Flm determines the correspondence between y^{T} and the most frequently assigned class y^{P} in $\mathcal{Y}^{\text{P}}$, utilizing the well-trained network for all target training samples of the class y^{T}, thus obtaining $f_{\text{out}}^{\text{Flm}}$, shown in Algorithm 3.

As the label mapping may change from time to time when learning f_{in}, Chen et al. (2023) proposed an *iterative label mapping* (Ilm) method that updates $f_{\text{out}}(\cdot)$ after each training iteration. Let $y_{\text{Ilm}}^{\text{P},(j)}$ be the optimal y^{P} in the jth training epoch. We have:

$$y_{\text{Ilm}}^{\text{P},(\text{j}+1)} = \arg\max_{y \in \mathcal{Y}^{\text{P}}} \Pr_{(x_i,y_i)\sim\mathcal{D}_{\text{T}}} \{y = f_{\text{P}}(f_{\text{in}}^{(j)}(x_i|\theta^{(j)}))|y_i = y^{\text{T}}\}, \tag{9}$$

where $f_{\text{in}}^{(j)}(\cdot|\theta^{(j)})$ is the parameters of the jth epoch. The output mapping function is updated after each iteration until convergence.

Algorithm 2 Computing Frequency Distribution of $[f_{\text{P}}(f_{\text{in}}(x_i|\theta)), y^{\text{T}}]$

```
1: Input: Target training set $\{(x_i^{\text{T}}, y_i^{\text{T}})\}_{i=1}^n$, given input VR $f_{\text{in}}(\cdot|\theta)$ and pre-trained model $f_{\text{P}}(\cdot)$
2: Output: Frequency distribution matrix $d \in \mathbb{Z}^{|\mathcal{Y}^{\text{P}}|\times|\mathcal{Y}^{\text{T}}|}$
3: Initialize $d \leftarrow \{0\}^{|\mathcal{Y}^{\text{P}}|\times|\mathcal{Y}^{\text{T}}|}$
4: # Compute frequency distribution $d$
5: for $i = 1...n$ do
6:     $\hat{y}_i^{\text{P}} \leftarrow f_{\text{P}}(f_{\text{in}}(x_i^{\text{T}}|\theta))$
7:     $d_{\hat{y}_i^{\text{P}},y_i^{\text{T}}} \leftarrow d_{\hat{y}_i^{\text{P}},y_i^{\text{T}}} + 1$
8: end for
```

Algorithm 3 Frequent Label Mapping ($f_{\text{out}}^{\text{Flm}}$)

```
1: Input: Label space of the pre-trained task $\mathcal{Y}^{\text{P}}$, label space of the target task $\mathcal{Y}^{\text{T}}$, target training set $\{(x_i^{\text{T}}, y_i^{\text{T}})\}_{i=1}^n$,
   given pre-trained model $f_{\text{P}}(\cdot)$
2: Output: Flm $f_{\text{out}}^{\text{Flm}} : \mathcal{Y}_{\text{sub}}^{\text{P}} \to \mathcal{Y}^{\text{T}}$
3: Initialize $f_{\text{out}}^{\text{Flm}}(\cdot) \leftarrow 0$, subset $\mathcal{Y}_{\text{sub}}^{\text{P}} \leftarrow \emptyset$ to store matched labels, initialize $f_{\text{in}}(\cdot|\theta)$ to be an identity function ($\theta \leftarrow \mathbf{0}$)
4: # Compute frequency distribution $d$
5: Use Algorithm 2 to obtain $d$
6: # Compute output mapping $f_{\text{out}}^{\text{Flm}}$
7: while size of $\mathcal{Y}_{\text{sub}}^{\text{P}}$ is not $|\mathcal{Y}^{\text{T}}|$ do
8:     Find the maximum $d_{y^{\text{P}},y^{\text{T}}}$ in $d$
9:     $\mathcal{Y}_{\text{sub}}^{\text{P}} \leftarrow \mathcal{Y}_{\text{sub}}^{\text{P}} \cup \{y^{\text{P}}\}$
10:    $f_{\text{out}}^{\text{Flm}}(y^{\text{P}}) \leftarrow y^{\text{T}}$ # Update the label mapping function
11:    $d_{y^{\text{P}},t} \leftarrow 0$ for $t = 1, 2, ..., |\mathcal{Y}^{\text{T}}|$ # Avoiding illegal assignment to the injective function
12:    $d_{s,y^{\text{T}}} \leftarrow 0$ for $s = 1, 2, ..., |\mathcal{Y}^{\text{P}}|$
13: end while
```

Ilm evolves with iterations, being an improved version of Flm. As is shown in Algorithm 4, before training the reprogramming pattern θ in each epoch, Ilm updates the one-to-one mapping from $\mathcal{Y}^{\text{P}}$ to $\mathcal{Y}^{\text{T}}$ with the training samples incorporating the current pattern, iteratively until convergence.

B. Additional Theoretical Proof

B.1. Proof of Theorem 4.2

The approximation error of $\mathcal{F}_1$ and $\mathcal{F}_2$ can be formulated as:

$$\text{Err}_{\mathcal{D}}^{\text{apx}}(\mathcal{F}_1) = \inf_{f\in\mathcal{F}_1} \mathbb{E}_{(X,Y)\sim\mathcal{D}}\ell(f(X),Y) - R_{\mathcal{D}}^*,$$
$$\text{Err}_{\mathcal{D}}^{\text{apx}}(\mathcal{F}_2) = \inf_{f\in\mathcal{F}_2} \mathbb{E}_{(X,Y)\sim\mathcal{D}}\ell(f(X),Y) - R_{\mathcal{D}}^*,$$

Straightforwardly,

$$\mathcal{F}_1 \supseteq \mathcal{F}_2 \Leftrightarrow \forall f \in \mathcal{F}_2, f \in \mathcal{F}_1$$

Algorithm 4 Iterative Label Mapping ($f_{\text{out}}^{\text{Ilm}}$)

```
1: Input: Label space of the pre-trained task $\mathcal{Y}^{\text{P}}$, label space of the target task $\mathcal{Y}^{\text{T}}$, target training set $\{(x_i^{\text{T}}, y_i^{\text{T}})\}_{i=1}^{n}$,
   given pre-trained model $f_{\text{P}}(\cdot)$, total iteration number $E$, learning rate $\alpha$
2: Output: Ilm $f_{\text{out}}^{\text{Ilm,(j)}} : \mathcal{Y}_{\text{sub}}^{\text{P}} \to \mathcal{Y}^{\text{T}}$ for iteration $j$
3: Initialize $f_{\text{out}}^{\text{Ilm,(j)}}(\cdot) \leftarrow 0$, subset $\mathcal{Y}_{\text{sub}}^{\text{P}} \leftarrow \emptyset$ to store matched labels, initialize $f_{\text{in}}(\cdot|\theta)$ to be an identity function ($\theta \leftarrow \mathbf{0}$)
4: for $j = 1...E$ do
5:   # Compute frequency distribution $d$
6:   Use Algorithm 2 to obtain $d$
7:   # Compute output mapping $f_{\text{out}}^{\text{Ilm,(j)}}$
8:   while size of $\mathcal{Y}_{\text{sub}}^{\text{P}}$ is not $|\mathcal{Y}^{\text{T}}|$ do
9:     Find the maximum $d_{y^{\text{P}},y^{\text{T}}}$ in $d$
10:    $\mathcal{Y}_{\text{sub}}^{\text{P}} \leftarrow \mathcal{Y}_{\text{sub}}^{\text{P}} \cup \{y^{\text{P}}\}$
11:    $f_{\text{out}}^{\text{Ilm,(j)}}(y^{\text{P}}) \leftarrow y^{\text{T}}$ # Update the label mapping function for iteration $j$
12:    $d_{y^{\text{P}},t} \leftarrow 0$ for $t = 1, 2, ..., |\mathcal{Y}^{\text{T}}|$ # Avoiding illegal assignment to the injective function
13:    $d_{s,y^{\text{T}}} \leftarrow 0$ for $s = 1, 2, ..., |\mathcal{Y}^{\text{P}}|$
14:  end while
15:  # Train $f_{\text{in}}(\cdot|\theta)$ for iteration $j$
16:  $\theta \leftarrow \theta - \alpha \cdot \nabla_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(f_{\text{out}}^{\text{Ilm,(j)}}(f_{\text{P}}(f_{\text{in}}(x_i^{\text{T}}|\theta))), y_i^{\text{T}})$
17: end for
```

Given $\mathcal{F}_1 \subseteq \mathcal{F}_2$, we have:

$$\begin{aligned}
&\forall f \in \mathcal{F}_1, f \in \mathcal{F}_2, \\
&\Rightarrow \inf_{f \in \mathcal{F}_1} \mathbb{E}_{(X,Y)\sim\mathcal{D}} \ell(f(X), Y) \geq \inf_{f \in \mathcal{F}_2} \mathbb{E}_{(X,Y)\sim\mathcal{D}} \ell(f(X), Y) \\
&\Rightarrow \text{Err}_{\mathcal{D}}^{\text{apx}}(\mathcal{F}_1) \geq \text{Err}_{\mathcal{D}}^{\text{apx}}(\mathcal{F}_2)
\end{aligned}$$

B.2. Proof of Proposition 4.3

We prove Proposition 4.3 as follows.

Proof. With specially designed kernel and padding sizes, the output of CNN can be reshaped to match the size of the input image. Assuming $d_{\text{P}} = H \times W \times C$, we define $M' \in \{0,1\}^{H*W*C\times 1}$ and $f'_{\text{mask}}(\cdot) \in \mathbb{R}^{H*W*C\times 1}$ as transposed flattened M and $f_{\text{mask}}(\cdot)$, respectively. As the last layer of $f'_{\text{mask}}(\cdot)$ is CNN, if the input of CNN is the resized image $r(x)$, with $x \in \mathcal{X}^{\text{T}}$ (and $r(x) \in \mathbb{R}^{d_{\text{P}}}$), we have $f'_{\text{mask}}(r(x)) = W_{\text{last}} f''_{\text{mask}}(r(x)) + b_{\text{last}}$, with b_{last} being the bias of the last layer, and W_{last} being the mapping from the flattened input of the last CNN layer (i.e., $f''_{\text{mask}}(r(x))$) to the flattened output without adding the bias, which can be derived using the parameters of the last CNN layer. With the set of any possible W_{last} being represented by $\{W_{\text{last}}\}$, and all-zero matrix being O, we have:

$$\begin{aligned}
&b_{\text{last}} \in \mathbb{R}^{H*W*C\times 1}, M' \in \{0,1\}^{H*W*C\times 1} \\
&\Rightarrow \forall M', M' \in \{b_{\text{last}} | b_{\text{last}} \in \mathbb{R}^{H*W*C\times 1}\} \qquad (10) \\
&O \in \{W_{\text{last}}\} \text{(When all weights in the last CNN layer is 0, } W_{\text{last}} \text{ is a zero matrix)} \\
&\Rightarrow f(x) = O^{H*W*C\times 1} \in \{f | f(x) = W_{\text{last}} f''_{\text{mask}}(r(x)), \forall x \in \mathcal{X}^{\text{T}}\} \qquad (11) \\
&\Rightarrow \{f | f(x) = M', \forall x \in \mathcal{X}^{\text{T}}\} \subseteq \{f | f(x) = f'_{\text{mask}}(r(x)), \forall x \in \mathcal{X}^{\text{T}}\} \text{(Given Eq. (10) and Eq. (11))} \\
&\Rightarrow \{f | f(x) = M, \forall x \in \mathcal{X}^{\text{T}}\} \subseteq \{f | f(x) = f_{\text{mask}}(r(x)), \forall x \in \mathcal{X}^{\text{T}}\} \\
&\Rightarrow \{f | f(x) = M \odot \delta, \forall x \in \mathcal{X}^{\text{T}}\} \subseteq \{f | f(x) = f_{\text{mask}}(r(x)) \odot \delta, \forall x \in \mathcal{X}^{\text{T}}\} \\
&\Rightarrow \mathcal{F}^{\text{shr}}(f'_{\text{P}}) \subseteq \mathcal{F}^{\text{smm}}(f'_{\text{P}}) \text{(since } f'_{\text{P}} \text{ is fixed)} \\
&\Rightarrow \text{Err}_{\mathcal{D}_{\text{T}}}^{\text{apx}}(\mathcal{F}^{\text{smm}}(f'_{\text{P}})) \leq \text{Err}_{\mathcal{D}_{\text{T}}}^{\text{apx}}(\mathcal{F}^{\text{shr}}(f'_{\text{P}}))
\end{aligned}$$

B.3. SMM and Sample-specific Patterns

We will then prove

Proposition B.1. *for any fixed* f'_{P}*, it holds that* $\mathcal{F}^{\mathrm{sp}}(f'_{\mathrm{P}}) \subseteq \mathcal{F}^{\mathrm{smm}}(f'_{\mathrm{P}})$*, and consequently,* $\mathrm{Err}^{\mathrm{apx}}_{\mathcal{D}_{\mathrm{T}}}(\mathcal{F}^{\mathrm{smm}}(f'_{\mathrm{P}})) \leq \mathrm{Err}^{\mathrm{apx}}_{\mathcal{D}_{\mathrm{T}}}(\mathcal{F}^{\mathrm{sp}}(f'_{\mathrm{P}}))$.

Proof. Let Δ be the set of possible δ, with all-one matrix being denoted as J, we have:

$$\begin{aligned}
&\Rightarrow J^{d_{\mathrm{P}}} \in \Delta \\
&\Rightarrow \{f | f(x) = f_{\mathrm{mask}}(r(x)) \odot J^{d_{\mathrm{P}}}, \forall x \in \mathcal{X}^{\mathrm{T}}\} \subseteq \{f | f(x) = f_{\mathrm{mask}}(r(x)) \odot \delta, \forall x \in \mathcal{X}^{\mathrm{T}}\} \\
&\Rightarrow \{f | f(x) = f_{\mathrm{mask}}(r(x)), \forall x \in \mathcal{X}^{\mathrm{T}}\} \subseteq \{f | f(x) = f_{\mathrm{mask}}(r(x)) \odot \delta, \forall x \in \mathcal{X}^{\mathrm{T}}\} \\
&\Rightarrow \mathcal{F}^{\mathrm{sp}}(f'_{\mathrm{P}}) \subseteq \mathcal{F}^{\mathrm{smm}}(f'_{\mathrm{P}}) \text{(Since } f'_{\mathrm{P}} \text{ is fixed)} \\
&\Rightarrow \mathrm{Err}^{\mathrm{apx}}_{\mathcal{D}_{\mathrm{T}}}(\mathcal{F}^{\mathrm{smm}}(f'_{\mathrm{P}})) \leq \mathrm{Err}^{\mathrm{apx}}_{\mathcal{D}_{\mathrm{T}}}(\mathcal{F}^{\mathrm{sp}}(f'_{\mathrm{P}}))
\end{aligned}$$

C. Additional Experimental Setup

Table 6. Detailed Dataset Information

DATASET	ORIGINAL IMAGE SIZE	TRAINING SET SIZE	TESTING SET SIZE	NUMBER OF CLASSES
CIFAR10	32×32	50000	10000	10
CIFAR100	32×32	50000	10000	100
SVHN	32×32	73257	26032	10
GTSRB	32×32	39209	12630	43
FLOWERS102	128×128	4093	2463	102
DTD	128×128	2820	1692	47
UCF101	128×128	7639	3783	101
FOOD101	128×128	50500	30300	101
SUN397	128×128	15888	19850	397
EUROSAT	128×128	13500	8100	10
OXFORDPETS	128×128	2944	3669	37

The 11 datasets used for the experiments are summarized in Table 6, while the corresponding training parameters are listed in Table 9. When learning the ResNet tasks, we follow the same learning strategies as Chen et al. (2023). When learning ViT-B32, we choose the initial learning rate α and the learning rate decay γ with a training parameter searching experiment, with results presented in Table 7.

Table 7. Tuning Initial Learning Rate and Learning Rate Decay Using CIFAR10 and ViT-B32 (Accucracy %)

$\gamma \vert \alpha$	0.1	0.01	0.001	0.0001
1	0.9542	0.9577	**0.9745**	0.9734
0.1	0.9516	0.9572	0.9738	0.9727

Sharing the same α and γ may not be optimal for all datasets. As shown in Table 8, on UCF101, using $\alpha = 0.001$ and $\gamma = 1$ derived from Table 7 leads to sub-optimal model performance. Nevertheless, for uniformity and fairness in this paper, we still use a single set of unified training parameters for all datasets.

Table 8. Results on UCF101 with Different Training Parameters (using ViT-B32)

	α	γ	SMM ACCURACY (%)
UNIFIED LEARNING PARAMETERS	0.001	1	42.6
SPECIFIC LEARNING PARAMETERS	0.01	0.1	49.9

Table 9. Detailed Model Training Parameter Settings of Our Mask Generator (where b, α and γ denote batch size, initial learning rate and learning rate decay, respectively)

			5-LAYER		6-LAYER	
	b	MILESTONES	α	γ	α	γ
CIFAR10	256	[0, 100, 145]	0.01	0.1	0.001	1
CIFAR100	256	[0, 100, 145]	0.01	0.1	0.001	1
SVHN	256	[0, 100, 145]	0.01	0.1	0.001	1
GTSRB	256	[0, 100, 145]	0.01	0.1	0.001	1
FLOWERS102	256	[0, 100, 145]	0.01	0.1	0.001	1
DTD	64	[0, 100, 145]	0.01	0.1	0.001	1
UCF101	256	[0, 100, 145]	0.01	0.1	0.001	1
FOOD101	256	[0, 100, 145]	0.01	0.1	0.001	1
SUN397	256	[0, 100, 145]	0.01	0.1	0.001	1
EUROSAT	256	[0, 100, 145]	0.01	0.1	0.001	1
OXFORDPETS	64	[0, 100, 145]	0.01	0.1	0.001	1

Table 10. Performance Improvement When Applying Our Input Reprogramming on Different Label Mapping Methods (the average results are highlighted in grey)

f_{out}	ITERATIVE LABEL MAPPING			FREQUENT LABEL MAPPING			RANDOM LABEL MAPPING		
	W/O OURS	W OURS	IMPROVE	W/O OURS	W OURS	IMPROVE	W/O OURS	W OURS	IMPROVE
CIFAR10	68.90%	72.80%	+3.90%	71.79%	72.75%	+0.96%	65.68%	69.71%	+4.03%
CIFAR100	33.80%	39.40%	+5.60%	29.79%	32.35%	+2.56%	16.99%	23.47%	+6.48%
SVHN	78.30%	84.40%	+6.10%	78.78%	83.73%	+4.95%	77.44%	85.37%	+7.92%
GTSRB	76.80%	80.40%	+3.60%	74.76%	80.90%	+6.14%	69.60%	82.38%	+12.79%
FLOWERS102	23.20%	38.70%	+15.50%	17.78%	32.16%	+14.37%	12.34%	37.68%	+25.33%
DTD	29.00%	33.60%	+4.60%	30.14%	34.28%	+4.14%	14.60%	19.74%	+5.14%
UCF101	24.40%	28.70%	+4.30%	22.71%	25.72%	+3.01%	9.04%	16.71%	+7.67%
FOOD101	13.20%	17.50%	+4.30%	11.58%	15.21%	+3.62%	7.15%	15.86%	+8.71%
SUN397	13.40%	16.00%	+2.60%	13.45%	15.45%	+1.99%	1.05%	3.35%	+2.29%
EUROSAT	84.30%	92.20%	+7.90%	86.00%	92.67%	+6.67%	84.49%	94.47%	+9.98%
OXFORDPETS	70.00%	74.10%	+4.10%	69.66%	72.83%	+3.16%	8.89%	16.84%	+7.96%
AVERAGE	46.85%	52.53%	+5.68%	46.04%	50.73%	+4.69%	33.39%	42.32%	+8.94%

D. Additional Experimental Results

D.1. Applying SMM with Different f_{out}

As mentioned before, and as shown in Appendix A.1, input VR is agnostic of the output label mapping method. Thus, our SMM can be applied to different output label methods other than Ilm. Experimental results are presented in Table 10.

Our method improves the performance of all output mapping methods. In most cases, the worse the output mapping method is, the more pronounced the improvement of SMM will be. When there is sufficient training data (e.g., GTSRB, SVHN, CIFAR10 and Food101), adding SMM can compensate for the worse-performing label mapping methods. With SMM, these methods also produce competitive results.

D.2. Analysis of Learning Curves

Figure 11 shows the training accuracy and loss throughout learning iterations using ResNet-18 as the pre-trained backbone. We see that our SMM yields a higher training accuracy and lower loss for most cases.

When using a more sophisticated pre-trained network, e.g., ViT, as is shown in Figure 12, the training accuracy without SMM may meet with or even exceed that of using SMM. However, this appears to be a case of over-fitting, where training accuracy is approaching 1 and test accuracy is still low without using SMM.

In general, for smaller classifiers such as ResNet-18, adding our model helps better reduce training loss and improve accuracy, while for more sophisticated classifiers such as ViT-B32 where the training accuracy is already high, adding our

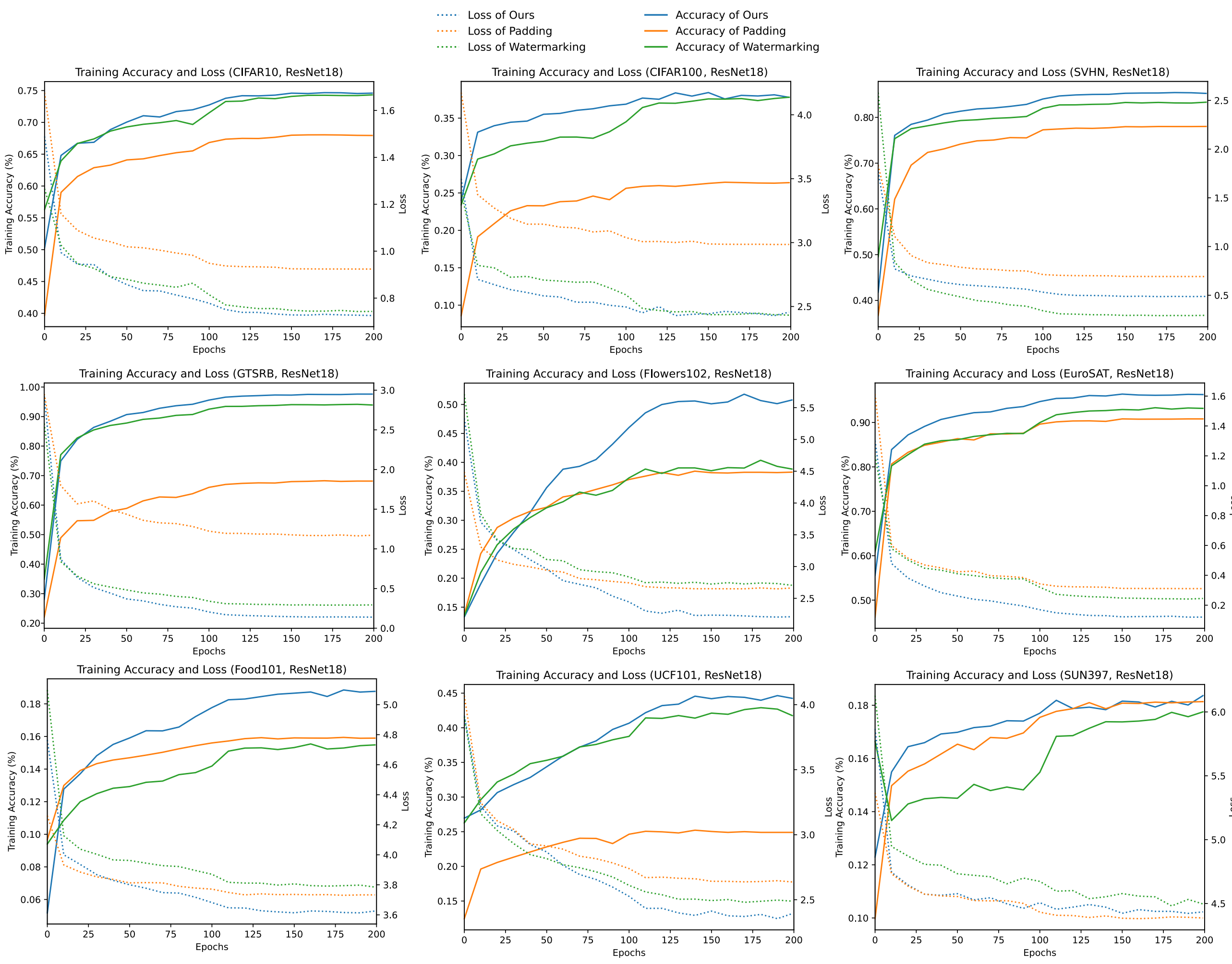


Figure 11. Training Accuracy and Loss of Different Reprogramming Methods

SMM model helps prevent over-fitting and improve the testing accuracy.

Table 11. Training and Testing Accuracy with Enlarged f_{mask} (using EuroSAT, ResNet-18)

f_{mask}	SMALL	MEDIUM (OURS)	LARGE	X-LARGE	XX-LARGE	XXX-LARGE
PARAMETERS	7203	26499	101379	396291	1566723	6230019
TRAINING ACCURACY (%)	94.9	96.2	96.4	97.3	97.7	98.1
TESTING ACCURACY (%)	91.7	92.2	92.2	93.1	93.5	93.2

D.3. More Discussion about the Estimation Error

A higher estimation error generally implies an increased risk of model over-fitting to the training data. This observation can be corroborated by comparing the disparities in training and testing performance. For instance, as depicted in Figure 12, employing a more sophisticated pre-trained network such as ViT with a mask generator f_{mask} shown in Figure 9 across some tasks like CIFAR10, SVHN, and GTSRB, the training accuracy tends towards 100% for both shared patterns $\mathcal{F}^{\text{shr}}(f'_{\text{P}})$ (i.e., 'Watermarking' in Figure 12) and SMM patterns $\mathcal{F}^{\text{smm}}(f'_{\text{P}})$ (i.e., 'Ours' in Figure 12). Despite this, $\mathcal{F}^{\text{smm}}(f'_{\text{P}})$ maintains a test accuracy that is not inferior to that of shared patterns. It suggests that our method SMM does not suffer from more significant over-fitting than shared masking, resulting in negligible potential estimation error.

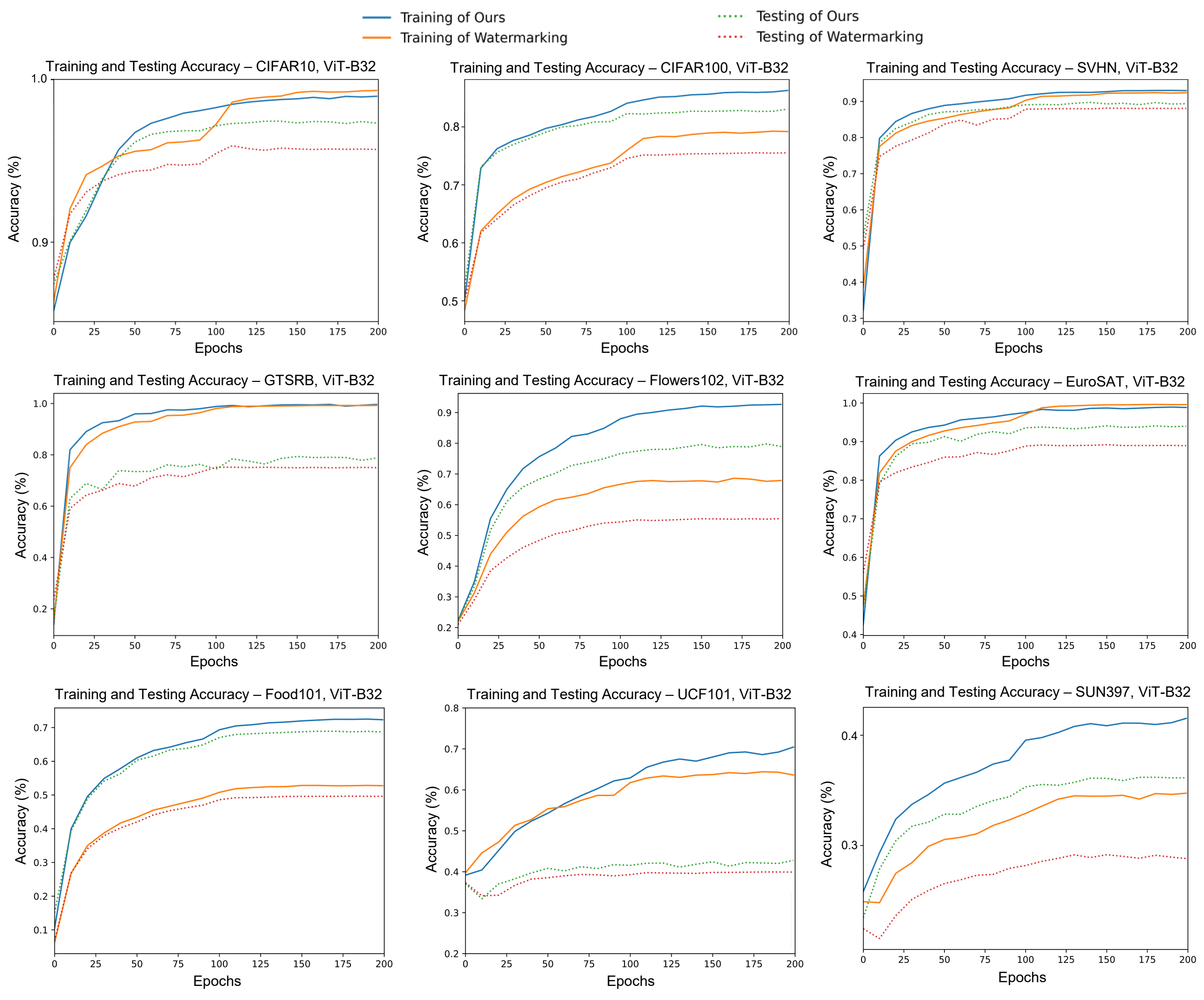


Figure 12. Training Accuracy and Testing Accuracy with and without Our Method

However, when f_{mask} is enlarged with increased number of parameters, the additional estimation error of $\mathcal{F}^{\text{smm}}(f'_{\text{P}})$ may no longer be negligible and will impact the excess risk. The relationship between the number of parameters in f_{mask} and the estimation error is influenced by various factors, including the specific target tasks, the volume of training data, the size of well-trained models, and the design of our generation model, etc. Through experiments, we will be able to estimate when the number of parameters begins to impact estimation error, potentially leading to over-fitting. For instance, in Table 11, we employ our generation model f_{mask} on the EuroSAT dataset, with ResNet-18 being the well-trained model. By progressively doubling the number of intermediate channels while maintaining the architecture of f_{mask}, we investigate how the model size affects performance.

Through the results of Table 11, we come to the following conclusions: (1) As the number of parameters continues to increase, although the training accuracy slowly increases, the test accuracy may even decrease, implying that the estimation error becomes more and more noticeable. (2) Under this situation (i.e., EuroSAT, ResNet-18), when the size of f_{mask} is close to the same order of magnitude as the well-trained model, the estimation error should not be overlooked. (3) A larger model with the best test accuracy may not be optimal because of too many parameters. Our work strikes a balance between the number of parameters and test accuracy.

D.4. Further Analysis of the Performance of SMM

More Discussion of SMM Abnormal Cases. In Section 5, we have briefly analyzed abnormal performance in Table 1 and Table 2. In this section, we will provide a more comprehensive discussion. Here, we outline detailed discussions regarding abnormal performance:

- ResNet-18, DTD: As shown in Figure 18, the DTD dataset contains a significant amount of texture features. Therefore, for relatively simple well-trained models, introducing reprogramming noise in the form of watermarking may affect the original features of the images. It can be observed that when the watermarking area is small (Narrow), the effect is better compared to when it is large (Full), and our method is also affected by this factor. However, the padding-based method preserves the original pixels of the image and only introduces reprogramming noise around them, thereby achieving relatively good results.

- ViT-B32, EuroSAT: This is because EuroSAT is one of the target tasks with the least task complexity. When using a large-scale network like ViT, the resizing-based method leads to over-fitting. As evident in the third column of the second row in Figure 12, the training accuracy is already close to 1. Therefore, in this scenario, the padding-based method yields slightly better test results compared to our method (which also belongs to resizing-based methods).

Table 12. An Ineffective Case of Input Reprogramming - StanfordCars (Mean % ± Std %)

METHOD	PAD	NARROW	MEDIUM	FULL	OURS
RESNET-18	4.5±0.1	3.6±0.1	3.6±0.1	3.4±0.1	2.9±0.2
RESNET-50	4.7±0.2	4.7±0.1	4.7±0.2	4.6±0.1	3.0±0.6
VIT-B32	4.7±0.6	7.7±0.2	8.3±0.3	5.0±0.0	4.8±0.9

SMM on An Ineffective Case of Input Reprogramming. All input visual reprogramming methods seem ineffective on fine-grained recognition tasks where subtle appearance differences should be detected. As shown in Table 12, in the classification of StanfordCars, where 196 types of cars are to be classified, the accuracy of all input VR methods is below 10 %, indicating the failure of VR methods in this fine-grained recognition tasks. Adding our SMM module will not improve performance when VR methods fail.

E. Additional Discussion about Input VR Compared with Finetuning

E.1. Advantages of VR in Dealing with Distorted Input Images

Table 13. Performance of Finetuning (LoRA) and SMM Facing Target Tasks with Different Input Image Sizes (Accyracy %, using ViT-L with a 384×384 input as the well-trained model, average results are calculated on all four tasks with 32×32 inputs and all seven tasks with 128×128 inputs)

	EXTRA PARAMETERS	CIFAR10	CIFAR100	SVHN	GTSRB	AVERAGE (32×32)	AVERAGE (128×128)
FINETUNING-LORA	0.60M	95.9	83.6	65.3	66.6	77.9	83.4
OUR SMM	0.54M	**97.4**	**87.3**	**91.0**	**84.2**	**90.0**	**83.5**

In this section, we will compare the results of our SMM with finetuning-based methods to show the advantages of input VR in dealing with distorted input images. *Low-rank adaptation* (LoRA) (Hu et al., 2021) is an efficient finetuning-based transfer method proposed based on large language models for natural language processing, which has been adapted to ViT (Zhu et al., 2023). Here, we compare SMM for ViT with LoRA for ViT, which are representative methods that belong to input VR and finetuning, respectively.

Since LoRA for ViT already includes finetuning the fully connected layers, we also incorporate it in SMM. All training settings are kept the same. We set the rank of LoRA to be six, resulting in an additional parameter number being 0.60M (without counting the fully connected layers), which will be comparable to that of input VR and SMM (being 0.54M) for fairness. ViT-Large with the input size being 384×384 is applied, and the learning rate is 0.01, running 10 epochs in total.

Therefore, for both methods, the target training samples will be resized before input. VR mainly trains parameters in the input space before well-trained models, whereas LoRA injects parameters into layers of ViT. Results are listed in Table 13.

The results of target tasks with the input size being 128×128 are similar. However, it is observed that for those target tasks with lower resolution (e.g., CIFAR10/100, SVHN, GTSRB), our SMM appears to perform better. This is likely because when a 32×32 image is resized to 384×384, it may become distorted, thus affecting the performance of target tasks. This distortion is especially noticeable on tasks with simple features, such as SVHN and GTSRB. Since VR modifies the input space, it effectively addresses this issue of significant differences in the input image sizes of pre-trained and target tasks.

E.2. Advantages of VR in Being Orthogonal to Finetuning-based Methods

Table 14. Performance of Finetuning the Fully-Connected Layers (Finetuning-FC) without or with our SMM Module (Accuracy %, using ResNet-50 as the well-trained model)

	CIFAR10	CIFAR100	SVHN	GTSRB	FLOWERS102	DTD
FINETUNING-FC	90.1	70.7	63.5	77.8	**90.9**	67.6
FINETUNING-FC + OUR SMM	**91.2**	**72.4**	**86.9**	**85.2**	**90.9**	**68.2**

	UCF101	FOOD101	SUN397	EUROSAT	OXFORDPETS	AVERAGE
FINETUNING-FC	70.8	57.6	53.5	95.7	90.4	75.3
FINETUNING-FC + OUR SMM	**72.0**	**59.6**	**57.9**	**95.8**	**90.6**	**79.2**

Since finetuning and reprogramming are orthogonal because finetuning modifies the model while reprogramming modifies the input and output spaces. Input VR can also be combined with finetuning-based methods. In this section, we will add the input VR module (i.e, using SMM as an example) to finetuning-based methods and analyze the performance gain. A widely-used method - finetuning the fully connected layer (named 'Finetuning-FC') - is employed as the baseline method. Using ResNet-50 as the well-trained model, we add our SMM input VR module to 'Finetuning-FC' to demonstrate the effectiveness of our module.

Results are shown in Tabel 14. Utilizing our module achieves an average accuracy of about 4% higher than solely finetuning the fully connected layers. Conclusively, input VR can be attached to finetuning-based methods to improve performance.

E.3. Strengths and Weaknesses of Input Reprogramming in Visual Tasks

This part includes a conclusion of the strengths and weaknesses of Input VR, compared with finetuning-based methods.

E.3.1. STRENGTHS

- The *parameter numbers* of VR tend to be *negligible* considering the size of well-trained models. Besides, the parameter numbers in VR are solely determined by the size of a single input image, independent of well-trained models, and remain fixed as the well-trained model size grows.

- VR is suitable for all well-trained models, *regardless of the architecture*, whereas finetuning-based methods are usually designed for a specific architecture (e.g., LoRA is specifically designed for ViT).

- VR improves the performance of the target task by altering the input and output space, and analyzing these changes may help *understand why* the model can also perform well in the target domain.

- By changing the input and output spaces while fixing the well-trained model, VR *avoids practical issues* such as catastrophic forgetting (i.e., the well-trained model may lose previously learned representations when being finetuned for new tasks).

- VR *can be attached to* most mainstream finetuning methods to further improve performance.

- In future research, VR could also utilize the well-trained model as a *black box*. This approach might prove useful for re-purposing models that only offer an application programming interface.

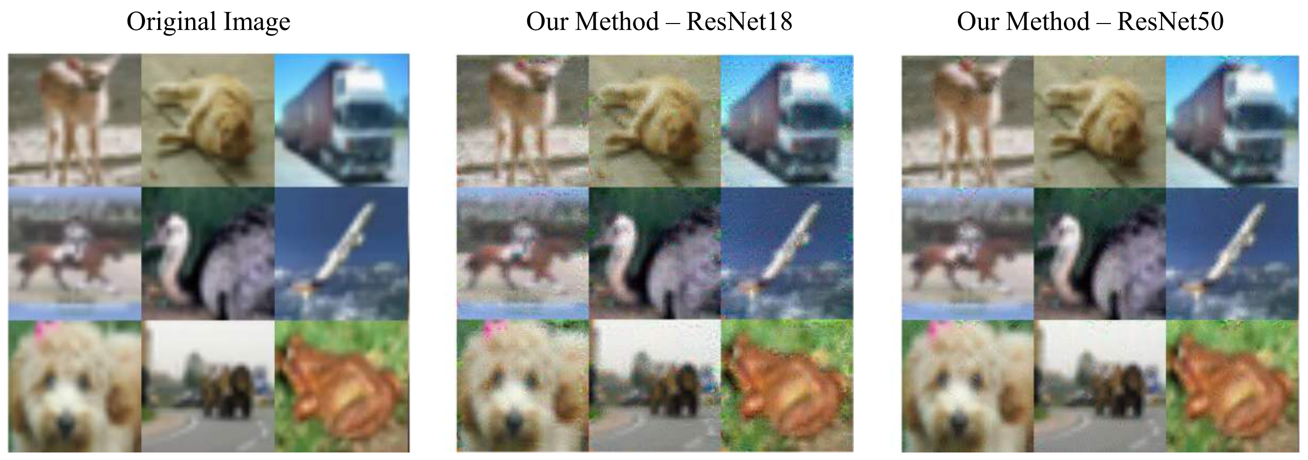


Figure 13. Original Images and Visual Reprogramming Results on CIFAR10

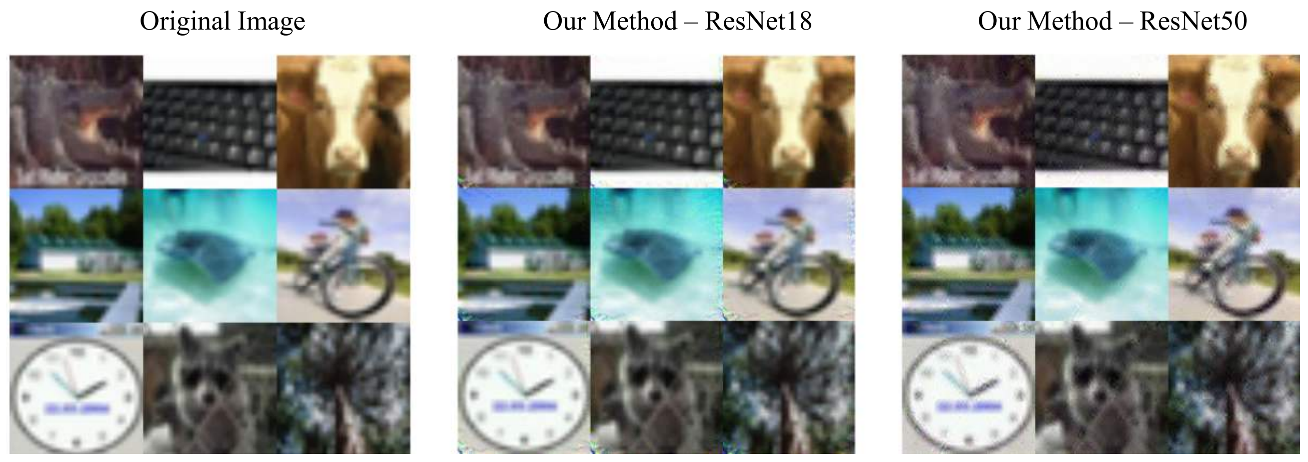


Figure 14. Original Images and Visual Reprogramming Results on CIFAR100

E.3.2. WEAKNESSES

- When target tasks are more *challenging* than the tasks well-trained models have been trained on, merely adjusting the input space may *not be sufficient* for satisfied performance. This poses a challenge for VR.
- For better performance approaching re-training or fully finetuning, integrating VR with other finetuning methods appears necessary (e.g., VR may be combined with finetuning the fully connected layer). *How to train the combined model more effectively* remains a task for future research.

F. Additional Visualization Results

Figure 13-23 show sample images of the VR results of SMM on 11 datasets. These figures show that (1) our VR method does not alter the input space heavily; it only adds noise within a limited range, which ensures that the original images remain intact; (2) the more different the target domain is (e.g., GTSRB and SVHN), the more pronounced the noise pattern will be; (3) on datasets that prefer VR to be a narrow padding-sized watermark, SMM will convergence to a similar situation, that is, the noise at the outer frame of the images is much greater than that inside the images (e.g., UCF101, Food101, OxfordPets and SUN397).

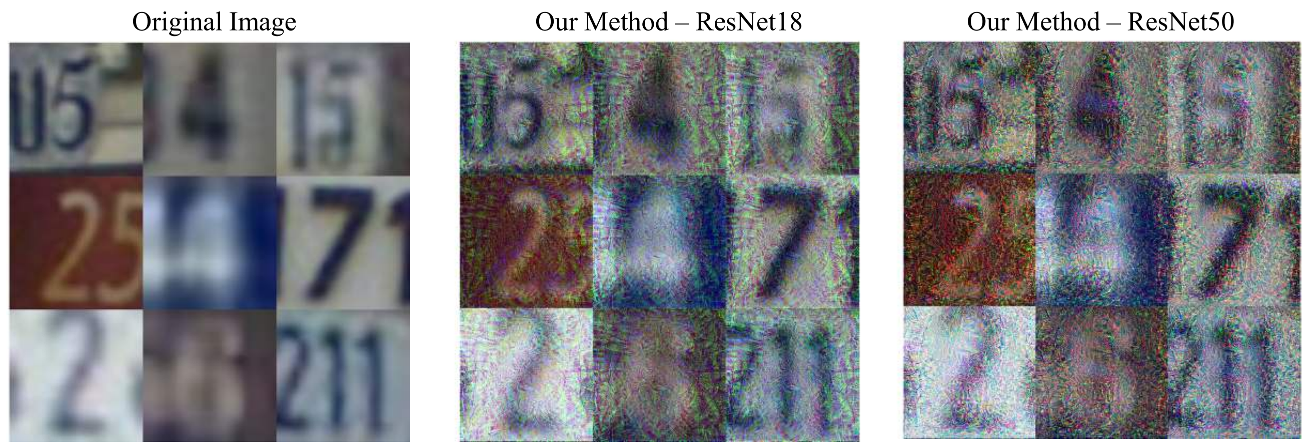


Figure 15. Original Images and Visual Reprogramming Results on SVHN

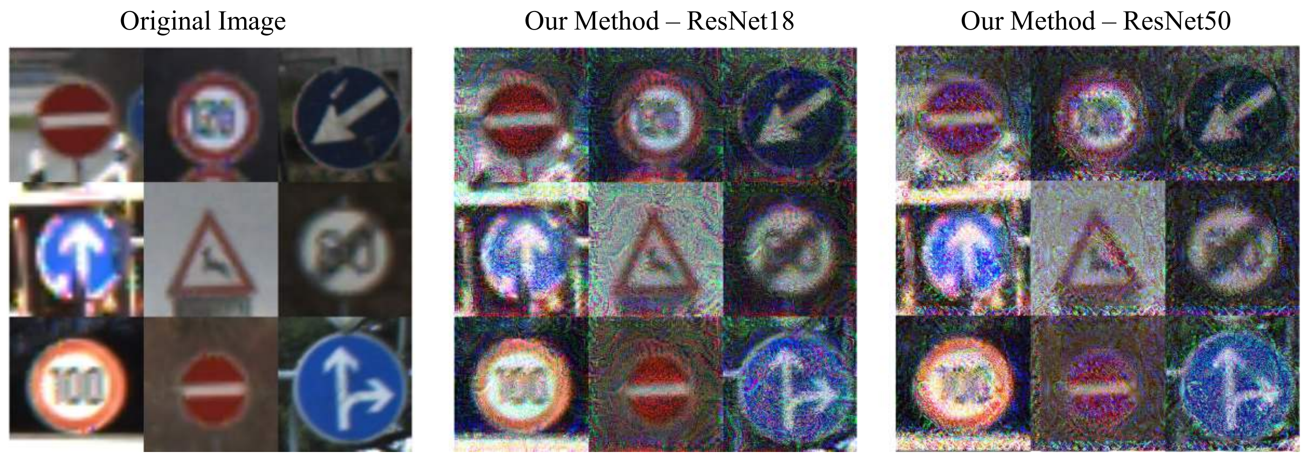


Figure 16. Original Images and Visual Reprogramming Results on GTSRB

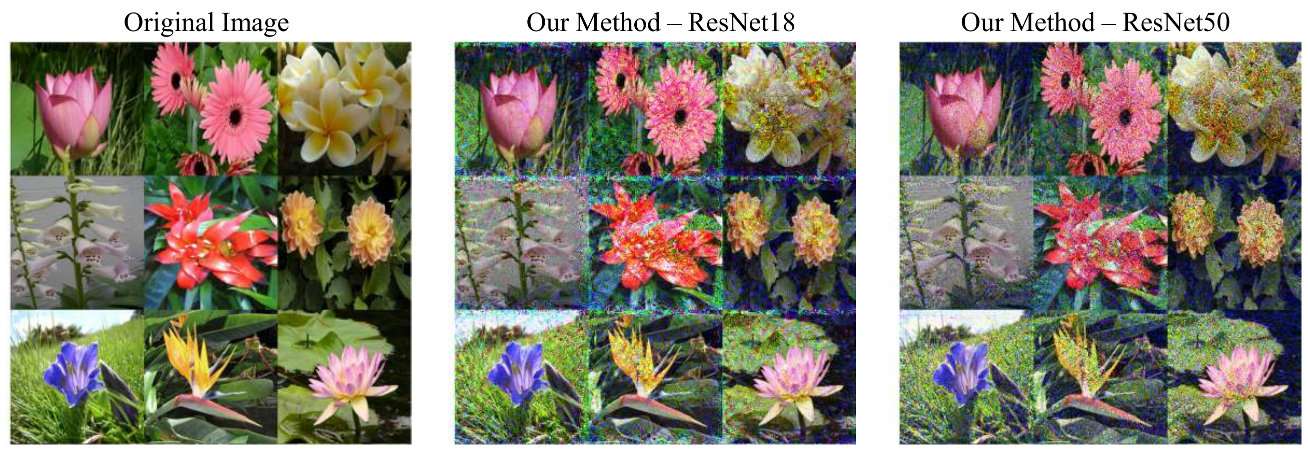


Figure 17. Original Images and Visual Reprogramming Results on Flowers102

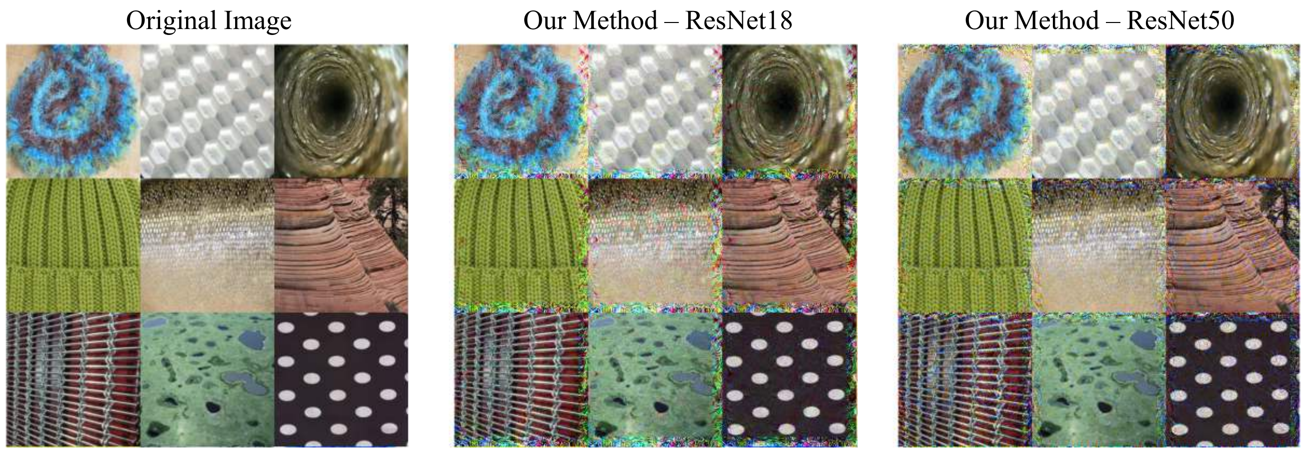


Figure 18. Original Images and Visual Reprogramming Results on DTD

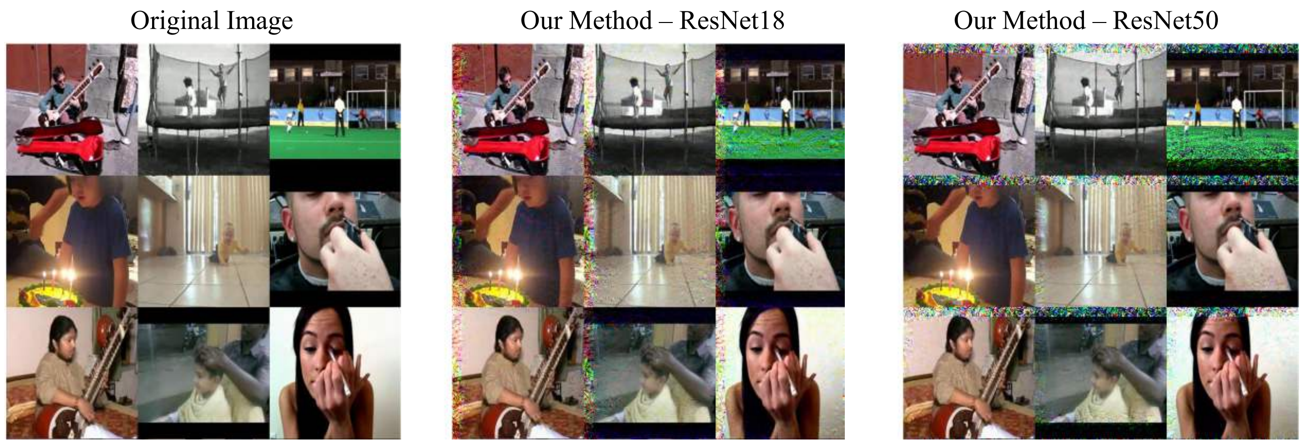


Figure 19. Original Images and Visual Reprogramming Results on UCF101

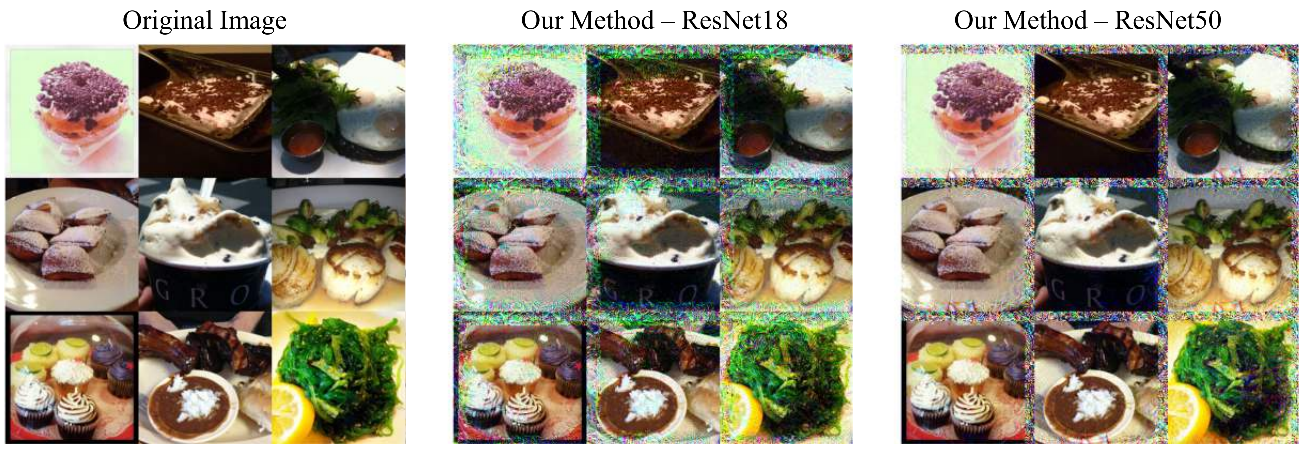


Figure 20. Original Images and Visual Reprogramming Results on Food101

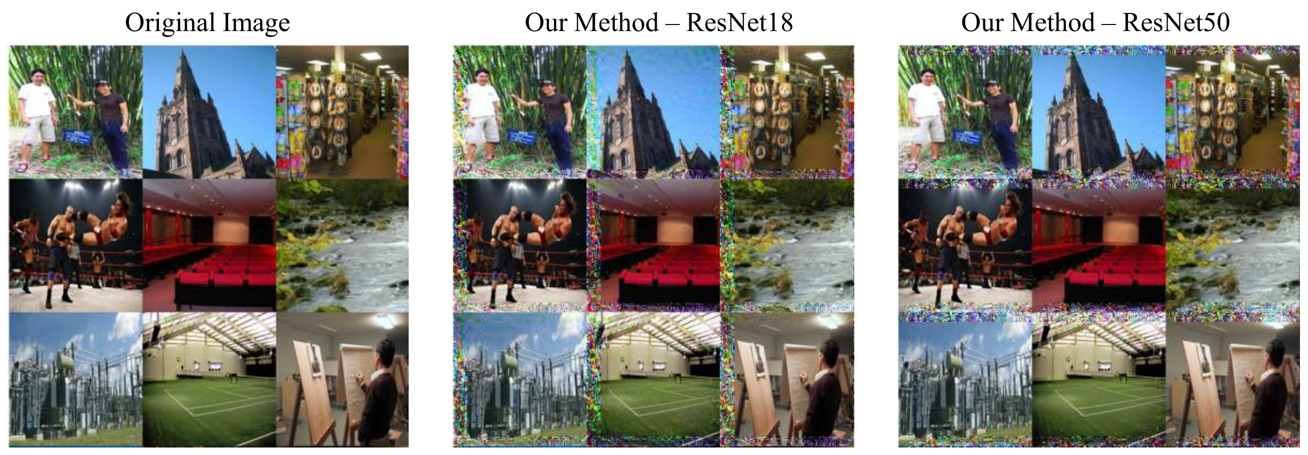


Figure 21. Original Images and Visual Reprogramming Results on SUN397

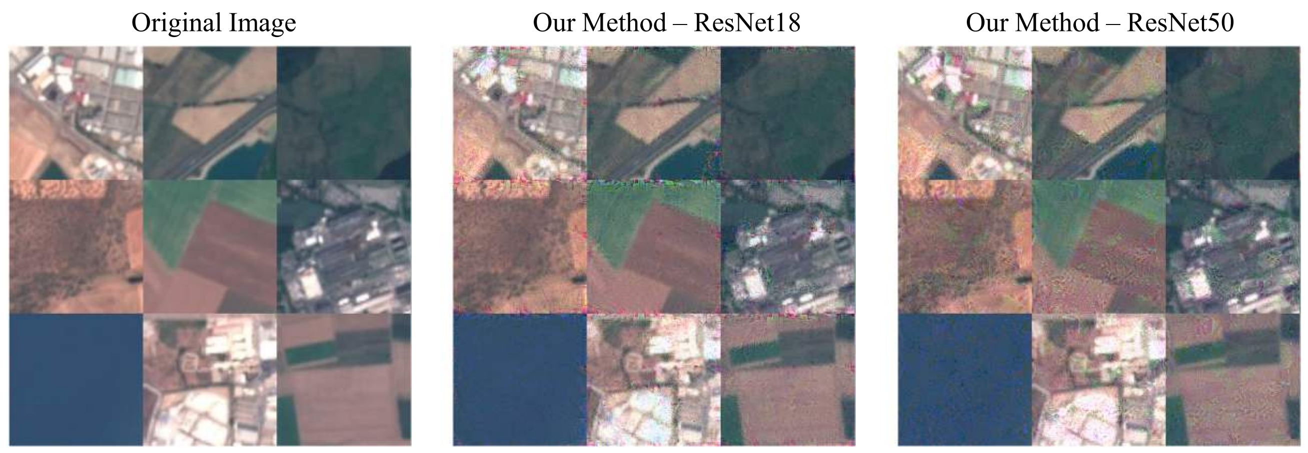


Figure 22. Original Images and Visual Reprogramming Results on EuroSAT

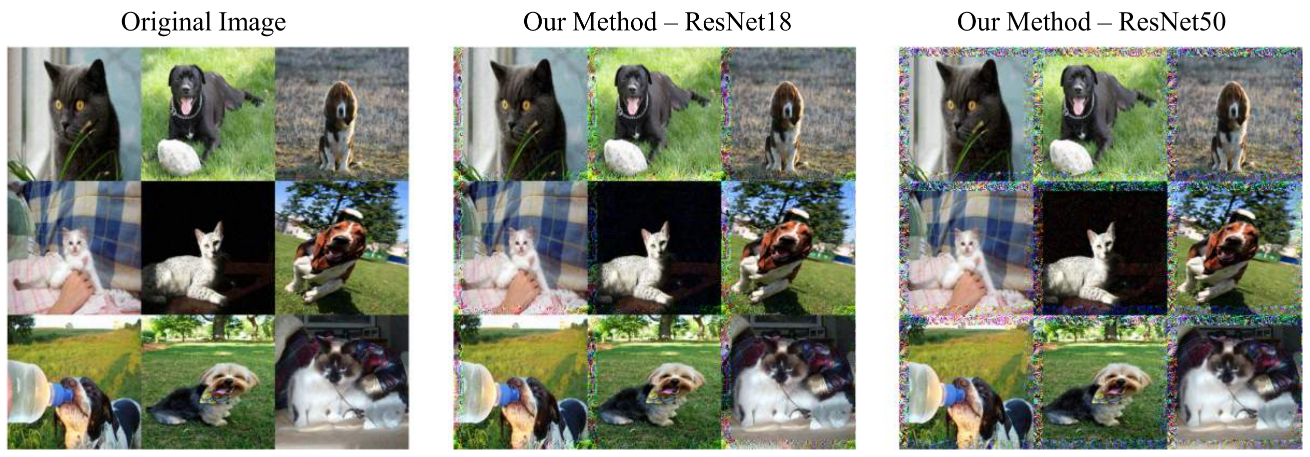


Figure 23. Original Images and Visual Reprogramming Results on OxfordPets