Learning Where to Edit Vision Transformers

Yunqiao Yang[1]* **Long-Kai Huang**[2]† **Shengzhuang Chen**[1]
Kede Ma[1] **Ying Wei**[3]†
[1]City University of Hong Kong [2]Tencent AI Lab [3]Zhejiang University
`{yqyang.cs, szchen9-c}@my.cityu.edu.hk` `hlongkai@gmail.com`
`kede.ma@cityu.edu.hk` `ying.wei@zju.edu.cn`

Abstract

Model editing aims to data-efficiently correct predictive errors of large pre-trained models while ensuring generalization to neighboring failures and locality to minimize unintended effects on unrelated examples. While significant progress has been made in editing Transformer-based large language models, effective strategies for editing vision Transformers (ViTs) in computer vision remain largely untapped. In this paper, we take initial steps towards correcting predictive errors of ViTs, particularly those arising from subpopulation shifts. Taking a locate-then-edit approach, we first address the "where-to-edit" challenge by meta-learning a hypernetwork on CutMix-augmented data generated for editing reliability. This trained hypernetwork produces generalizable binary masks that identify a sparse subset of structured model parameters, responsive to real-world failure samples. Afterward, we solve the "how-to-edit" problem by simply fine-tuning the identified parameters using a variant of gradient descent to achieve successful edits. To validate our method, we construct an editing benchmark that introduces subpopulation shifts towards natural underrepresented images and AI-generated images, thereby revealing the limitations of pre-trained ViTs for object recognition. Our approach not only achieves superior performance on the proposed benchmark but also allows for adjustable trade-offs between generalization and locality. Our code is available at `https://github.com/hustyyq/Where-to-Edit`.

1 Introduction

In many scientific and engineering disciplines, computational models serve as approximations of complex real-world phenomena. As a consequence, they are inherently prone to predictive errors, aptly encapsulated by George Box's adage: "*All models are wrong, but some are useful*." Model editing [55, 6, 9, 40] has emerged as a promising technique to make (large) pre-trained models *more useful* by enabling targeted updates to model behavior on specific inputs or tasks in a data-efficient manner without pre-training again from scratch. An ideal model editing method should satisfy three major desiderata [9, 61]: 1) *reliability*, ensuring the model behavior is effectively updated for the current sample; 2) *generalization*, so that the changes extend to neighboring samples; and 3) *locality*, meaning the edit should have minimal impact on the model behavior on unrelated samples.

Model editing has allowed many fascinating applications, including error correction, factual knowledge update, bias mitigation, policy compliance, and personalization, though most of them have predominantly been within large language models (LLMs) [15, 1, 12] in the natural language processing (NLP) community [9, 40]. With the enormous and often inaccessible pre-training datasets and the ever-growing model sizes that make retraining computationally demanding, the need for effectively

*Part of the work was done when the author interned at Tencent AI Lab.
†Corresponding authors.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

editing computer vision (CV) models is also becoming urgent. Adapting model editing techniques from NLP to CV is non-trivial and presents unique challenges. From the data perspective, NLP deals with one-dimensional, discrete signals that are highly semantic and information-dense, whereas CV requires processing high-dimensional continuous sensor data that is spatially redundant. From the model perspective, lots of model editing methods in NLP are specially designed for LLMs with *unidirectional* (*i.e.*, autoregressive) attention, such as GPT-3 [15] and GPT-4 [1]. In contrast, CV models have primarily been based on convolutional networks [33, 52, 24], with more recent implementations using vision Transformers (ViTs) [11, 35] that otherwise employ *bidirectional* attention. These differences in data formats and model structures make targeted edits more challenging to implement in CV models, and when such edits are achieved, they often result in suboptimal performance.

In this paper, we take initial steps towards editing pre-trained ViTs for object recognition [10], aiming to correct predictive errors without the need for costly and time-consuming retraining. Specifically, we take a locate-then-edit approach, which breaks down model editing into two key subproblems: where-to-edit and how-to-edit. Moreover, we prioritize learning where to edit rather than how to edit to facilitate a simpler yet better trade-off between generalization and locality, without needing to store previously trained data.

For the where-to-edit phase, we first narrow the editing scope using a greedy search-based heuristic. Next, inspired by the proven effectiveness of meta-learning [14] in optimizing training strategies for individual samples, we meta-train a hypernetwork to generate a binary task, indicating which parameters are critical for the editing sample. To address the issue of limited data, the hypernetwork is trained solely using pseudo-samples, each comprising a natural image paired with its CutMix version [62] (see Fig. 1). The optimization objective is to align the predicted probability distribution of the CutMix sample to that of the original. By controlling the sizes of patches used in CutMix and randomly varying their locations, we simulate distribution shifts in backgrounds, contextual objects, and object attributes, creating opportunities to learn generalizable binary masks that effectively respond to real-world failures. Additionally, we apply a sparsity constraint to the binary masks, acting as an indirect, data-free regularizer to promote locality. Once the where-to-edit problem is solved, the how-to-edit phase becomes straightforward: we simply fine-tune the selected parameters using a variant of gradient descent to apply targeted edits.

To validate our method, we construct an editing benchmark that exposes the weaknesses of pre-trained ViTs by introducing two types of subpopulation shifts. The first is a natural subpopulation shift [45, 50], with underrepresented natural images of certain categories efficiently identified by the maximum discrepancy (MAD) competition [58]. The second is an artificial subpopulation shift, introduced by synthesized images from high-quality text-to-image generative models like Stable Diffusion [46].

In summary, our key contributions are as follows:

- a first-of-its-kind model editing method for pre-trained ViTs that leverages meta-learning to prioritize the where-to-edit phase;
- an editing benchmark that provides valuable resources for future model editing research in CV;
- an extensive experimental demonstration that our method achieves the best Pareto front between generalization and locality on the proposed benchmark, while offering flexible trade-offs in the how-to-edit phase.

2 Related Work

In this section, we provide a brief overview of current model editing methods in NLP and CV.

2.1 Model Editing in NLP

Memory-based Methods rely on external mechanisms, such as wrappers [41] and caches [21], to store factual updates without modifying the internal model parameters. A common theme in these studies is the use of a gating mechanism to determine whether a test sample falls within the editing scope; if so, the base model behavior is overridden. For instance, SERAC [41] and GRACE [21] employ a scope classifier as a form of hard gating, while Murty *et al.* [42] utilized a soft gating function, allowing for smoother integration. More recent approaches like IKE [63] and MeLLo [64] alter the input prompts of an LLM for knowledge update, where the gating mechanism is implicitly embedded within the LLM itself. Generally, memory-based methods offer advantages such as non-destructive updates, modularity, and suitability for continual and few-shot learning settings. However, they face scalability issues when handling a large number of edits. Additionally, the editing success heavily depends on the accuracy of the gating mechanism.

Parameter-based Methods modify the internal model parameters, which offers a more fine-grained approach to editing. These methods can roughly be categorized into two subgroups: locate-then-edit approaches and hypernetwork-based approaches. Locate-then-edit methods focus on identifying a subset of key parameters for editing. For instance, ROME [38], MEMIT [39], and MEMIT$_{\text{CSK}}$ [20] leverage causal mediation analysis (*i.e.*, representation denoising) to locate hidden states (*i.e.*, intermediate representations, not model parameters) responsible for knowledge storage. The theory of associative memory [32] is then applied to transfer the state localization results to model parameters. Recent studies [22] suggest that knowledge localization may not reliably inform successful edits. Furthermore, the very notion that knowledge can be localized may be inherently flawed, as factual information in LLMs may be encoded in a distributed manner [40]. *Single-step* integrated gradient across multiple editing samples [8, 60] is another commonly used statistic for localization. Here, we adopt a more principled meta-learning strategy to locate key parameters, using *multi-step* gradient information that more accurately captures the changes in model behavior.

Hypernetwork-based methods, such as KnowledgeEditor [9], MEND [40], and MALMEN [54], train an external network to directly generate parameter updates for the editing sample, which is represented by either feedforward feature representation [9] or backward gradient decomposition [40]. Localization techniques can be applied beforehand to restrict the functional space of the hypernetwork. Existing hypernetwork-based methods emphasize the how-to-edit aspect but treat the where-to-edit superficially, and often result in suboptimal performance, especially when adapting to CV applications. In contrast, our method prioritizes learning where to edit, achieving a better balance between generalization and locality.

2.2 Model Editing in CV

Limited research on model editing has been conducted in CV. Bau *et al.* [3] took a locate-then-edit approach to rewrite generative adversarial networks. Santurkar *et al.* [49] adapted this method for editing image classifiers based on convolutional networks by mapping the representation of the new visual concept to that of a previously learned concept. However, this approach requires prior knowledge of the new visual concept, its location within the image, and the specific target concept for correction. In practical applications, such detailed information may not always be available. In contrast, our method relaxes all these assumptions and is one of the first applied to ViTs.

3 Learning Where to Edit ViTs

In this section, we first present the preliminaries, followed by a detailed description of the proposed method for learning where to edit ViTs. The system diagram of our method is shown in Fig. 1.

3.1 Preliminaries

Problem Formulation Given a base computational model $f(\cdot;\theta) : \mathcal{X} \mapsto \mathcal{Y}$, parameterized by θ, model editing aims to modify the model behavior for specific inputs $x \in \mathcal{X}$ (or regions of the input space, $\mathcal{S} \subset \mathcal{X}$) while keeping its overall performance intact. Denote the post-edited model as $f\left(\cdot;\theta^{(e)}\right)$, where $\theta^{(e)}$ represents the updated parameter vector[3] after editing. Typically, $f\left(\cdot;\theta^{(e)}\right)$ is evaluated based on three main criteria: reliability, generalization, and locality.

- **Reliability**: For any editing sample (x, y), the edited model $f\left(x;\theta^{(e)}\right) = y$.
- **Generalization**: For any neighboring[4] sample $(x', y') \in \mathcal{N}(x, y)$, $f\left(x';\theta^{(e)}\right) = y'$, even if (x', y') is not directly used in the editing process.

[3]We slightly abuse the notation $\theta^{(e)}$ to encompass any possible modifications, including those by memory-based methods.

[4]Conceptually, in high-level vision, two images are considered neighbors, if they are semantically similar, such as belonging to the same category or subpopulation.

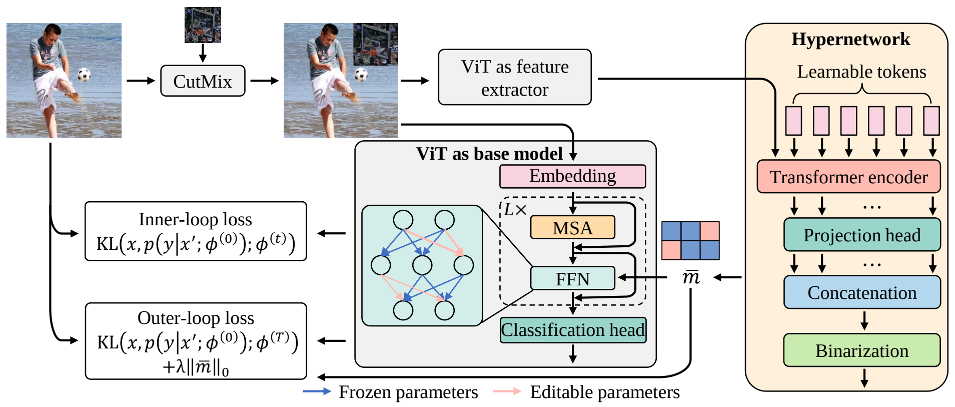


Figure 1: System diagram of the proposed model editing method.

- **Locality**: For any sample $(x', y') \notin \mathcal{N}(x, y)$, the model behavior should remain unchanged, *i.e.*, $f\left(x'; \theta^{(e)}\right) = f(x'; \theta)$.

An ideal model editing method shall ensure reliable edits while balancing generalization and locality effectively. As initial model editing attempts in CV, we limit our scope to single-example editing.

Vision Transformers A ViT [11] feature extractor, denoted by $e(\cdot; \phi)$ with parameter vector ϕ, consists of a linear embedding layer followed by L attention blocks. Each block is composed of a multiheaded self-attention (MSA) layer and a feedforward neural network (FFN). The FFN, which underpins most model editing methods, including ours, comprises two fully-connected (FC) layers: $\mathrm{FFN}(z) = \mathrm{GELU}(zW + b)W' + b'$. Here, $W \in \mathbb{R}^{N \times N_m}$ and $W' \in \mathbb{R}^{N_m \times N}$ are weight matrices, where N_m denotes the intermediate dimension. $b \in \mathbb{R}^{N_m \times 1}$ and $b' \in \mathbb{R}^{N \times 1}$ are bias terms. The activation function $\mathrm{GELU}(\cdot)$ is the Gaussian error linear unit [26].

An input image x is first partitioned into M non-overlapping, fixed-size patches, each linearly embedded in an N-dimensional feature space together with a class token `[cls]`, yielding a concatenation of patch embeddings of size $(M + 1) \times N$. These embeddings are processed through the L attention blocks for feature extraction. A linear classification head, $h(\cdot)$, maps the extracted features to a probability distribution over classes in $\mathcal{Y}$, represented as $p(y = c|x; \phi) = h_c(e(x; \phi))$, where $c \in \mathcal{Y}$. For notation simplicity, we omit the parameters in the classification head $h(\cdot)$, as they constitute only a small fraction of the total parameters and are generally frozen during model editing.

3.2 Model Editing at Training Time: Where-to-edit

The simplest way of editing a ViT is through vanilla fine-tuning, which involves updating all model parameters. However, modern ViTs have millions to billions of parameters, and fine-tuning on a single sample (x, y) can lead to overfitting, while incurring substantial computation costs. To overcome these, prior research [8, 23] first identifies a subset of key parameters, followed by editing:

$$\phi^\star = \phi + \bar{m} \odot \Delta\phi, \tag{1}$$

where $\bar{m}$ is a binary mask of the same dimension as ϕ, $\Delta\phi$ represents the parameter update, and $\odot$ is the Hadamard product.

Prevailing localization strategies in NLP rely on casual mediation analysis [38], integrated gradients [8], or pure heuristic methods [28], which may not be ideal for ViTs due to differences in data modalities and model architectures. In this work, we follow the locate-the-edit approach, and decompose model editing into two subproblems: where-to-edit (*i.e.*, computing $\bar{m}$) and how-to-edit (*i.e.*, computing $\Delta\phi$), with a focus on where-to-edit. Drawing inspiration from the demonstrated success of meta-learning [34, 14] in tailoring training strategies for individual samples, we meta-train a hypernetwork to generate the binary mask $\bar{m}$ for each editing sample.

Meta-learning [34, 14], also known as learning-to-learn, involves training models on a collection of training episodes [7] to enable effective generalization and adaptation to novel, unseen episodes. In our context, a training episode corresponds to a single editing example. We employ optimization-based meta-learning approaches [14, 44], framing where-to edit as a bi-level optimization problem. In the inner loop, key parameters, indicated by $\bar{m}$, are updated for the editing sample by optimizing a reliability loss via gradient-descent over T iterations. In the outer loop, the hypernetwork $g(\cdot;\varphi)$, parameterized by φ, is refined to generate $\bar{m}$. Mathematically, we have

$$\begin{aligned}
\min_{\varphi} \quad & \ell\left(x,y;\phi^{(T)}\right)+\lambda\|\bar{m}\|_0 \\
\text{s.t.} \quad & \bar{m}=g(x;\varphi) \\
& \Delta\phi^{(t)}=\Delta\phi^{(t-1)}-\alpha\nabla_{\phi}\ell\left(x,y;\phi^{(t-1)}\right),\, t\in\{1,2,\ldots,T\} \\
& \phi^{(t)}=\phi^{(0)}+\bar{m}\odot\Delta\phi^{(t)},\, t\in\{1,2,\ldots,T\},
\end{aligned} \tag{2}$$

where (x,y) is the editing sample. $\phi^{(T)}$ is the updated parameter after T iterations of inner-loop optimization, and $\phi^{(0)}$ denotes the pre-trained parameters of the base model as initialization. The term $\Delta\phi^{(t)}$ is the parameter update after the t-th iteration, with $\Delta\phi^{(0)}=0$. The loss function $\ell\left(x,y;\phi^{(t)}\right)$ measures the reliability of the edit. To encourage sparsity in the binary mask $\bar{m}$, we add an ℓ_0-norm term in the outer-loop objective, which acts as an indirect, data-free regularizer to encourage locality. The scalar λ controls the trade-off between the two terms. In our implementation, the hypernetwork takes the last-stage features corresponding to the `[cls]` token from the ViT feature extractor $e\left(\cdot;\phi^{(0)}\right)$ as input, *i.e.*, $\bar{m}=g\left(e\left(x;\phi^{(0)}\right);\varphi\right)$.

3.3 Optimization Challenges

Despite mathematical elegance, solving the bi-level optimization problem in (2) presents three challenges. First, meta-training the hypernetwork necessitates a sizable of high-quality editing samples, which are expensive and time-consuming to collect in practice. To address this, we generate pseudo-samples using a data augmentation technique known as CutMix [62]. Second, identifying key parameters within the entirety of the ViT presents a vast search space. This combinatorial complexity not only introduces unacceptable computational costs but also makes the localization of key parameters a challenging endeavor [36, 51]. To alleviate this, we shrink the editing scope based on a greedy search-based heuristic. Third, generating a binary mask typically involves a binarization operation in $g(\cdot;\varphi)$, which produces zero gradients almost everywhere and is thus ineffective in optimizing. To resolve this, we use a gradient-friendly approximation to binarization.

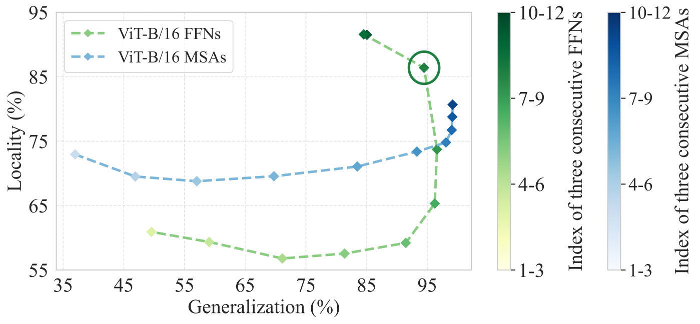
Pseudo-sample Generation We employ CutMix [62] to generate pseudo-samples for editing. Specifically, given a natural image x', we apply CutMix [62] to randomly overlay a small patch from another irrelevant image onto x', producing a pseudo-sample x. This patch-based perturbation tends to alter the predicted probability distribution, resulting in $p\left(y=c|x;\phi^{(0)}\right)\neq p\left(y=c|x';\phi^{(0)}\right)$, for $c\in\mathcal{Y}$. This motivates us to instantiate the reliability loss $\ell\left(x,y;\phi^{(t)}\right)$ in Problem (2) as the Kullback-Leibler (KL) divergence [27] between $p\left(y|x';\phi^{(0)}\right)$ and $p\left(y|x;\phi^{(t)}\right)$:

$$\ell\left(x,\left\{p\left(y|x';\phi^{(0)}\right)\right\};\phi^{(t)}\right)=\sum_{c\in\mathcal{Y}}p\left(y=c|x';\phi^{(0)}\right)\log\left(\frac{p\left(y=c|x';\phi^{(0)}\right)}{p\left(y=c|x;\phi^{(t)}\right)}\right), \tag{3}$$

where $\left\{p\left(y|x';\phi^{(0)}\right)\right\}$ is treated as the soft ground-truth label.

Editing Scope Shrinkage Previous studies [38, 40] have suggested that modifying FFNs within a Transformer is more effective for achieving successful edits [17, 18]. For example, MEND[40] focuses on editing the last three FFNs, while ROME [38] targets the middle FFNs. Here, we conduct a similar empirical investigation to identify a subset of consecutive FFNs in a ViT, by greedy search for the optimal generalization and locality trade-off. Specifically, we fine-tune ten groups of FFNs (or MSAs) in three consecutive layers [40] of a pre-trained ViT/B-16, denoted as {1-3, 2-4, . . ., 10-12}. The editing set comprises 100 predictive failures of the ViT, where `volleyball` is mistaken for `basketball` (see Fig. 2a), identified by the MAD competition [58] (see more details in Sec. 4.1). The average results across the editing set are shown in Fig. 2b, where we see that editing MSAs is not conducive to preserving locality. In contrast, editing the 8-th to 10-th FNNs tends to achieve the best trade-off, which are selected as the default layers for subsequent experiments.

(a) Editing samples.



(b) Editing results via vanilla fine-tuning.

Figure 2: The left subfigure shows representative editing examples, highlighting the predictive errors of the base ViT when predicting `volleyball` as `basketball`. The right subfigure depicts the generalization and locality trade-offs when editing different groups of FFNs or MSAs in the base ViT. It is evident that editing the 8-th to 10-th FFNs achieves the optimal Pareto front.

To further limit the output space of the hypernetwork, we employ structured tuning [8] by selecting specific rows/columns of the weight matrices in the FFNs for updating. As suggested in [8], we select the weights along the intermediate dimension N_m, which further reduces the output dimension of the hypernetwork to $N_m \times 6$ (*i.e.*, three FFNs with two FCs each).

Binarization Approximation As a special case of quantization in signal processing, binarization can be approximated to enable gradient-based training through three main approaches: straight-through estimation [4], uniform noise addition [2], and soft-to-hard annealing [30]. Here, we use a fixed parametric sigmoid function with favorable gradient behavior as the approximation:

$$\hat{m} = \text{Sigmoid}(k \times m), \tag{4}$$

where m is a continuous map computed by the hypernetwork right before binarization, and k is a hyperparameter that controls the degree to which the sigmoid curve approximates the desired binarization operation. Empirically, we set $k = 10$. We have also experimented with a soft-to-hard annealing for k, and observed comparable results. After adopting Eq. (4), we substitute $\bar{m}$ with $\hat{m}$ and replace the ℓ_0-norm with the ℓ_1-norm in Problem (2) to facilitate gradient-based optimization.

3.4 Model Editing at Test Time: How-to-edit

At test time, we solve the how-to-edit problem in a manner similar to the inner-loop optimization. The two minor differences lie in the loss function and the binarization operation.

At test time, we are provided with the editing sample x and its ground-truth label y. Therefore, the KL divergence during training reduces the cross-entropy loss during testing:

$$\ell\left(x, y; \phi^{(t)}\right) = -\sum_{c\in\mathcal{Y}} \mathbb{I}[y = c] \log\left(p\left(y = c|x; \phi^{(t)}\right)\right). \tag{5}$$

Also, we can directly employ the threshold-based binarization without approximation to obtain

$$\bar{m}_i = q(m_i) = \begin{cases} 1 & m_i \geq \rho \\ 0 & m_i < \rho, \end{cases} \tag{6}$$

where i is the positional index, and ρ is a hyperparameter that can be adjusted for different model editing applications. When ρ is set to zero, all parameters in the selected FFNs are updated with improved reliability. As ρ increases, fewer parameters are updated, which favors locality.

3.5 Hypernetwork Architecture

Similar to the ViT feature extractor $e\left(\cdot; \phi^{(0)}\right)$, the hypernetwork $g(\cdot; \varphi)$ comprises five attention blocks, an FC layer as the projection head, and a binarization operation. As shown in Fig. 1, we introduce six learnable tokens, each corresponding to an FC layer within the three selected FFNs of the base ViT. These tokens are concatenated with the image features derived from $e\left(\cdot; \phi^{(0)}\right)$ and serve as input to the hypernetwork to compute the binary mask $\bar{m}$.

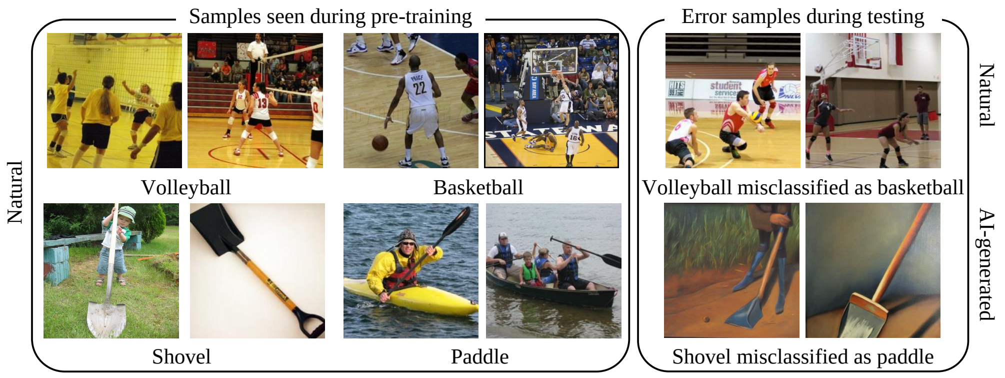

Samples seen during pre-training
Error samples during testing
Natural
Volleyball
Basketball
Volleyball misclassified as basketball
Natural
AI-generated
Shovel
Paddle
Shovel misclassified as paddle

Figure 3: Visual examples seen by the base ViT/B-16 during pre-training, contrasted with visual examples in the proposed editing benchmark as predictive errors of the base ViT/B-16.

4 Editing Benchmark with Subpopulation Shifts

In this section, we establish an editing benchmark that exposes failures of the base ViT in object recognition by introducing subpopulation shifts to underrepresented natural and AI-generated images.

4.1 Natural Image Subset

To build the natural image subset, we first compile a large dataset of unlabeled images, denoted as $\mathcal{U}$, from Flickr, by leveraging keywords relevant to the object categories in ImageNet-1k [10]. Next, we employ the MAD competition [58] to facilitate failure identification of the base ViT to be edited. Under the principle of model falsification as model comparison, MAD chooses to identify images that best distinguish two classifiers, $f(\cdot)$ and $f'(\cdot)$, by maximizing their prediction discrepancies. This can be mathematically formulated as

$$x^{(i)} = \underset{x' \in \mathcal{U} \setminus \mathcal{D}_n}{\arg\max}\, d\left(f(x'), f'(x')\right), \tag{7}$$

where $\mathcal{D}_n = \{x^{(j)}\}_{j=1}^{i-1}$ is the set of $i-1$ images that have been identified. $d(\cdot, \cdot)$ is the multi-hop distance defined over the WordNet [13] to measure prediction discrepancy at a semantic level. Intuitively, if one classifier is weaker, the identified image set $\mathcal{D}_n$ is more likely to include its predictive failures, thereby substantially reducing the human effort for failure identification. Moreover, the "ground-truth" labels for these failures can be first suggested by the stronger model and then verified by two of the authors. To leverage this intuition, we pair our base model (*i.e.*, a ViT/B-16 pre-trained on ImageNet-1k) with a stronger one (*i.e.*, the same ViT/B-16 pre-trained using CLIP [43] and fine-tuned on ImageNet), which generally exhibits better generalization to unseen data. In total, we collect $2,354$ MAD-searched natural images, which are partitioned into 16 groups, *i.e.*, $\mathcal{D}_n = \{\mathcal{S}^{(i)}\}_{i=1}^{16}$, based on the predictions by the two models. Each group is named according to the format "prediction of the stronger model"-"prediction of the base model," with the statistics and visual examples given in the Appendix.

4.2 AI-generated Image Subset

Classifiers pre-trained on natural images often struggle to generalize to AI-generated images [56, 59]. To exploit this, we construct an AI-generated image subset containing two groups of images, denoted as $\mathcal{D}_a = \{\mathcal{S}^{(i)}\}_{i=17}^{18}$. The 17-th group includes 860 images with an art style shift (*i.e.*, oil painting) generated by Textural Inversion [56], while the 18-th group comprises $1,092$ images with a lighting condition shift (*i.e.*, stage light) produced by PUG [5]. Both Textural Inversion and PUG are text-to-image generators, wherein the "ground-truth" label is embedded in the input text prompt and subsequently verified by two of the authors. Additional details of the AI-generated image subset can be found in the Appendix.

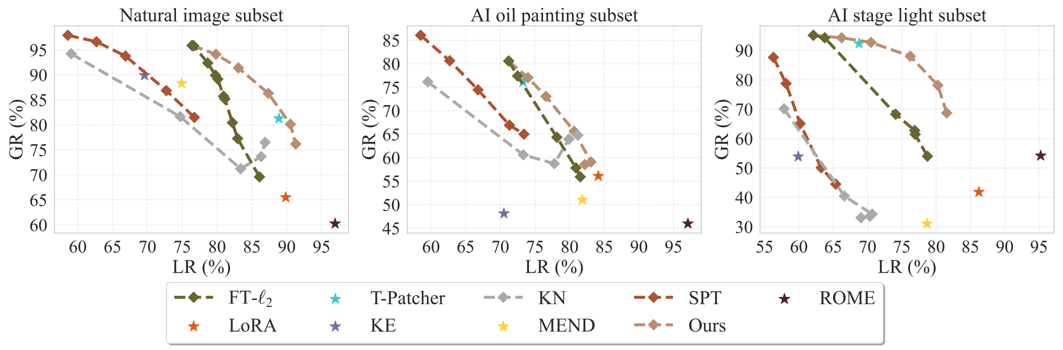


Figure 4: Editing results for ViT/B-16 on the proposed benchmark.

5 Experiments

In this section, we first describe the experimental setups and then present comparison results on the proposed editing benchmark.

5.1 Experiment Setups

Evaluation Metrics Following [29], we evaluate all model editing methods on the single-example editing task and compare their performance using three evaluation metrics. The first is the *success rate* (SR), which indicates the reliability (*i.e.*, accuracy) of the edited model $f\left(\cdot;\theta^{(e)}\right)$:

$$\mathrm{SR}(f,\mathcal{D}_r)=\frac{1}{|\mathcal{D}_r|}\sum_{(x,y)\in\mathcal{D}_r}\mathbb{I}\left[y=f\left(x;\theta^{(e)}(x,y)\right)\right], \tag{8}$$

where $\mathcal{D}_r=\mathcal{D}_n\bigcup\mathcal{D}_a$ consists of all MAD-searched and AI-generated images, and we make it explicit the dependence of the updated parameters $\theta^{(e)}$ on the editing sample (x,y). The second metric is the *generalization rate* (GR), which assesses the accuracy of the edited model on neighboring samples that fall within the editing scope:

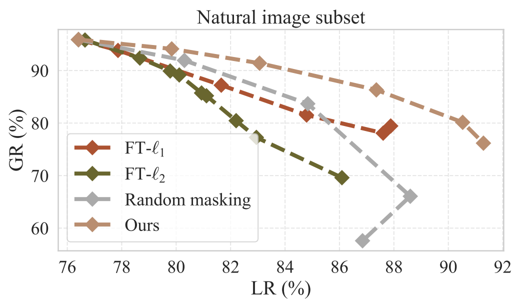
$$\mathrm{GR}(f,\mathcal{S})=\frac{1}{|\mathcal{S}|(|\mathcal{S}|-1)}\sum_{(x',y')\in\mathcal{S}}\sum_{(x,y)\in\mathcal{S}\setminus(x',y')}\mathbb{I}\left[y=f\left(x;\theta^{(e)}(x',y')\right)\right], \tag{9}$$

where $\mathcal{S}$ denotes one of the 18 groups in the proposed editing benchmark. We further average the GR values across all groups as an overall indicator of generalization. The third metric is the *locality rate* (LR), which examines whether the edited model maintains its predictions on unrelated samples outside the editing scope:

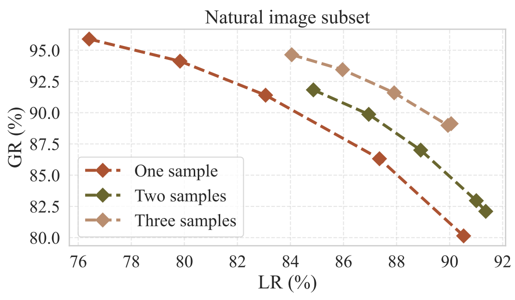
$$\mathrm{LR}(f,\mathcal{D}_r,\mathcal{D}_l)=\frac{1}{|\mathcal{D}_r||\mathcal{D}_l|}\sum_{(x',y')\in\mathcal{D}_r}\sum_{(x,y)\in\mathcal{D}_l}\mathbb{I}\left[y=f\left(x;\theta^{(e)}(x',y')\right)\right], \tag{10}$$

where $\mathcal{D}_l$ includes out-of-scope images. Using the validation set from ImageNet-1k as $\mathcal{D}_l$ does not adequately examine locality, as the majority are easy samples that lie far from the decision boundary [16]. To more closely examine the adverse effects of model editing, we have carefully curated $2,071$ images near the decision boundary of the base model from the validation sets of ImageNet-1k [47], ImageNet-R [25], and ImageNet-Sketch [57], whose predictions are more susceptible to change. Our selection criteria rely on the predicted probabilities of the pre-trained ViT/B-16 model as follows: 1) the predicted probability for the true label is the highest, and 2) the difference between the top two predicted probabilities is less than 0.05, suggesting a highly ambiguous class. We also employ the GR-LR curve to delineate the generalization and locality trade-off.

Base Models For all model editing methods, we experiment with two ViT backbones, ViT-B/16 and ViT/S-16, both pre-trained on ImageNet-21k and ImageNet-1k [53, 47].



(a) Localization effectiveness.



(b) More editing samples.

Figure 5: Ablation results of the hypernetwork for ViT/B-16.

Competing Methods We compare our method with several recent model editing approaches as follows. 1) Fine-tuning (FT) updates the 8-th to 10-th FFNs, which have been identified as the most effective layers using greedy search (see Fig. 2). 2) FT-ℓ_2 [39] incorporates ℓ_2-norm regularization during fine-tuning. 3) T-Patcher [29] adds and tunes a single neuron in the last FFN. 4) KN [8] and 5) SPT [23] select key parameters based on integrated gradient information. 6) ROME [38] is implemented to adjust the second FC layer of the last FFN by solving a constrained least squares problem. 7) LoRA [28] introduces trainable low-rank matrices to update the queries and values of all MSAs. 8) KE [9] and 9) MEND [40] employ hypernetworks to generate parameter updates for the last three FFNs. In line with previous work [40, 39], early stopping is applied when the training loss drops below 0.01 or the maximum of 100 editing steps is reached. Detailed implementations of the competing methods and additional training configurations are provided in the Appendix.

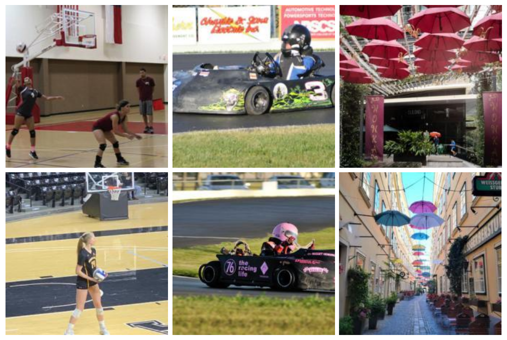
5.2 Main Results

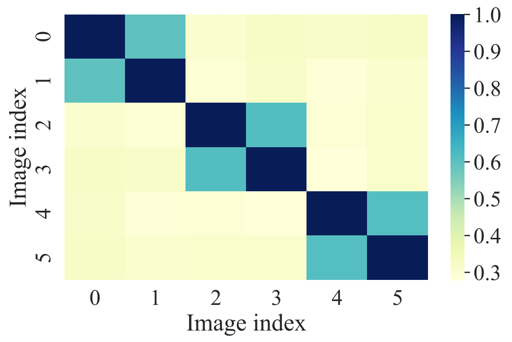
Fig. 4 shows the GR-LR curves for different editing methods applied to ViT-B/16, averaged across 18 groups in the proposed benchmark. We highlight several interesting observations. First, correcting a single predictive error is generally feasible, as evidenced by a nearly 100% SR for most methods. Second, achieving high levels of generalization and locality simultaneously proves to be a significant challenge. T-Patcher and ROME utilize previously seen data to maintain locality. Nevertheless, T-Patcher, which relies on an editing scope classifier, exhibits noticeable generalization variability across different editing samples. ROME, being specifically designed for language-based GPT [15], shows limited promise in generalizing to ViTs. LoRA manages to maintain locality because of its low-rank updates but struggles to generalize. Both KE and MEND exhibit low locality on the MAD-searched natural images and poor generalization to the AI-generated images. Third, our method achieves the new state-of-the-art without relying on previously trained data to explicitly enforce locality. Similar conclusions can be drawn for ViT-S/16, shown in the Appendix.

We then evaluate our method across different parameter sparsity levels in the three FFNs from $\{0.25, 0.50, 0.75, 0.90, 0.95\}$, corresponding to $\{12.4\%, 8.25\%, 4.13\%, 1.65\%, 0.83\%\}$ parameters of the entire model, by adjusting ρ in Eq. (6). The competing methods—FT-ℓ_2, KN, and SPT—are adjusted to comparable levels of parameter sparsity by tuning their respective hyperparameters. Note that our method reduces to FT when $\rho = 0$. The resulting GR-LR curves are shown in Fig. 4. As expected, increasing the parameter sparsity in KN, SPT, and our method improves locality at the expense of generalization. Notably, our method achieves the best Pareto front among all methods, which we believe arises from our proposed strategy of learning where to edit towards editing success.

5.3 Ablation Studies

Localization Effectiveness To substantiate that the effectiveness of our method is indeed due to the successful localization of a specific subset of key parameters, rather than merely due to sparsity, we compare the binary masks produced by our hypernetwork to random masks at the same sparsity levels, together with FT-ℓ_1 and FT-ℓ_2. As depicted in Fig. 5a, FT-ℓ_1 generally surpasses FT-ℓ_2 at various regularization levels as ℓ_1-norm is more effective in zeroing out less important parameters. Applying random masks shows effects akin to FT-ℓ_1. When the ratio of editing parameters falls below 1.65%, the performance of random masking becomes significantly inferior to our method.


(a) Six representative editing examples from three different groups.



(b) Binary mask IoU results between pairs of samples in (a), indexed in column-major order.

Figure 6: Mask specificity results.

Mask Specificity To confirm the specificity of the parameters identified by the hypernetwork for different editing samples, we compute the intersection over union (IoU) of the corresponding binary masks at the 0.95 sparsity level for samples within and outside the same groups in the natural image subset. Fig. 6b illustrates that the identified parameters demonstrate substantial overlaps for images within the same group and much lower overlaps between images from different groups. These findings support that the hypernetwork successfully pinpoints key parameters necessary to correct specific errors while effectively excluding parameters associated with other unrelated samples. This learned mask specificity allows our method to balance effectively between generalization and locality.

More Editing Samples We further evaluate our method when multiple editing samples in the same group (*i.e.*, with similar failure causes) are available. As a straightforward extension, we compute the average of the continuous masks generated from each sample, followed by binarization using Eq. (6). Fig. 5b presents the results of using one, two, and three samples for model editing. Remarkably, the editing performance improves with more editing samples, which can be attributed to more precise parameter localization as a result of the ensemble of masks.

More Ablation Studies More ablation studies (*e.g.*, the alternative pseudo-sample generation strategy, the sparsity regularization in the outer loop, the gradient step and learning rate in the inner loop, and the number of attention blocks in the hypernetwork) are in the Appendix.

6 Conclusion and Discussion

We have introduced a model editing method to correct predictive errors in ViTs. Our method prioritizes where-to-edit over how-to-edit by meta-training a hypernetwork to identify a subset of structured parameters for editing. By applying ℓ_1-norm regularization, our method promotes sparsity in the generated mask, thereby indirectly ensuring locality without needing to retrain on previously used data. Comprehensive tests on the proposed editing benchmark confirm that our method effectively corrects predictive errors in ViTs. Moreover, the introduced edits are not only reliable but also generalize well to neighboring samples, while maintaining a high rate of locality.

Our work is among the early endeavors in CV model editing, and it raises several intriguing questions for future research. First, our approach utilizes the CutMix technique [62] to generate cost-effective pseudo-samples for training, but its effectiveness has only been confirmed empirically. The reasons why the hypernetwork trained on such synthetic data achieves reasonable generalization and the identification of optimal synthetic data generation techniques remain wide open. Second, it would be beneficial to adapt our method to other vision architectures, such as convolutional networks or Swin Transformers [35], and extend its application to other vision areas like dense prediction, generative modeling, and multimodal LLMs. Third, exploring how to apply our method in a batch-editing setting represents a promising avenue. In such scenarios, the use of a decoupling trick (see more details in the Appendix) may prove essential for effectively reducing computational and memory demands.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia L Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations*, 2017.

[3] David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In *European Conference on Computer Vision*, pages 351–369, 2020.

[4] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[5] Florian Bordes, Shashank Shekhar, Mark Ibrahim, Diane Bouchacourt, Pascal Vincent, and Ari Morcos. PUG: Photorealistic and semantically controllable synthetic data for representation learning. In *Advances in Neural Information Processing Systems*, pages 8072–8081, 2023.

[6] Wray Buntine. Theory refinement on Bayesian networks. In *Conference on Uncertainty in Artificial Intelligence*, pages 52–60, 1991.

[7] Jiaxin Chen, Xiao-Ming Wu, Yanke Li, Qimai Li, Li-Ming Zhan, and Fu-lai Chung. A closer look at the training strategy for modern meta-learning. In *Advances in Neural Information Processing Systems*, pages 396–406, 2020.

[8] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained Transformers. In *Annual Meeting of the Association for Computational Linguistics*, pages 8493–8502, 2022.

[9] Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Empirical Methods in Natural Language Processing*, pages 6491–6506, 2021.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[12] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[13] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[14] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

[15] Luciano Floridi and Massimo Chiriatti. GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.

[16] Irena Gao, Gabriel Ilharco, Scott Lundberg, and Marco T Ribeiro. Adaptive testing of computer vision models. In *IEEE International Conference on Computer Vision*, pages 4003–4014, 2023.

[17] Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Empirical Methods in Natural Language Processing*, pages 30–45, 2022.

[18] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Empirical Methods in Natural Language Processing*, pages 5484–5495, 2021.

[19] Chong Guo, Michael Lee, Guillaume Leclerc, Joel Dapello, Yug Rao, Aleksander Madry, and James Dicarlo. Adversarially trained neural representations are already as robust as biological neural representations. In *International Conference on Machine Learning*, pages 8072–8081, 2022.

[20] Anshita Gupta, Debanjan Mondal, Akshay Sheshadri, Wenlong Zhao, Xiang Li, Sarah Wiegreffe, and Niket Tandon. Editing common sense in transformers. In *Empirical Methods in Natural Language Processing*, pages 8214–8232, 2023.

[21] Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with grace: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems*, pages 47934–47959, 2023.

[22] Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? Surprising differences in causality-based localization vs. knowledge editing in language models. In *Advances in Neural Information Processing Systems*, pages 17643–17668, 2023.

[23] Haoyu He, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. Sensitivity-aware visual parameter-efficient fine-tuning. In *IEEE International Conference on Computer Vision*, pages 11825–11835, 2023.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[25] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *IEEE International Conference on Computer Vision*, pages 8340–8349, 2021.

[26] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[27] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv preprint arXiv:1503.02531*, 2015.

[28] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[29] Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. In *International Conference on Learning Representations*, 2023.

[30] Jeong-Hun Jang and Ki-Sang Hong. Binarization of noisy gray-scale character images by thin line modeling. *Pattern Recognition*, 32(5):743–752, 1999.

[31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[32] Teuvo Kohonen. Correlation matrix memories. *IEEE Transactions on Computers*, 100(4):353–359, 1972.

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[34] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical vision Transformer using shifted windows. In *IEEE International Conference on Computer Vision*, pages 10012–10022, 2021.

[36] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. MetaPruning: Meta learning for automatic neural network channel pruning. In *IEEE International Conference on Computer Vision*, pages 3296–3305, 2019.

[37] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[38] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, pages 17359–17372, 2022.

[39] Kevin Meng, Arnab S Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a Transformer. In *International Conference on Learning Representations*, 2023.

[40] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022.

[41] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831, 2022.

[42] Shikhar Murty, Christopher D Manning, Scott Lundberg, and Marco T Ribeiro. Fixing model bugs with natural language patches. In *Empirical Methods in Natural Language Processing*, pages 11600–11613, 2022.

[43] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763, 2021.

[44] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pages 113–124, 2019.

[45] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *International Conference on Machine Learning*, pages 5389–5400, 2019.

[46] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[47] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.

[48] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust ImageNet models transfer better? In *Advances in Neural Information Processing Systems*, pages 3533–3545, 2020.

[49] Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. Editing a classifier by rewriting its prediction rules. In *Advances in Neural Information Processing Systems*, pages 23359–23373, 2021.

[50] Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. BREEDS: Benchmarks for sub-population shift. In *International Conference on Learning Representations*, 2021.

[51] Haopu Shang, Jia-Liang Wu, Wenjing Hong, and Chao Qian. Neural network pruning by cooperative coevolution. In *International Joint Conference on Artificial Intelligence*, pages 4814–4820, 2022.

[52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[53] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your ViT? Data, augmentation, and regularization in vision Transformers. *Transactions on Machine Learning Research*, 2022.

[54] Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language models via meta learning. In *International Conference on Learning Representations*, 2024.

[55] Sebastian Thrun and Tom M Mitchell. Learning one more thing. In *International Joint Conference on Artificial Intelligence*, pages 1217–1223, 1995.

[56] Joshua Vendrow, Saachi Jain, Logan Engstrom, and Aleksander Madry. Dataset interfaces: Diagnosing model failures using controllable counterfactual generation. *ArXiv preprint arXiv:2302.07865*, 2023.

[57] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019.

[58] Haotao Wang, Tianlong Chen, Zhangyang Wang, and Kede Ma. I am going mad: Maximum discrepancy competition for comparing classifiers adaptively. In *International Conference on Learning Representations*, 2020.

[59] Olivia Wiles, Isabela Albuquerque, and Sven Gowal. Discovering bugs in vision models using off-the-shelf image generation and captioning. In *Advances in Neural Information Processing Systems Workshop on Machine Learning Safety*, 2022.

[60] Xinwei Wu, Junzhuo Li, Minghui Xu, Weilong Dong, Shuangzhi Wu, Chao Bian, and Deyi Xiong. DEPN: Detecting and editing privacy neurons in pretrained language models. In *Empirical Methods in Natural Language Processing*, pages 2875–2886, 2023.

[61] Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. In *Empirical Methods in Natural Language Processing*, pages 10222–10240, 2023.

[62] Sangdoo Yun, Dongyoon Han, S Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *International Conference on Computer Vision*, pages 6023–6032, 2019.

[63] Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? In *Empirical Methods in Natural Language Processing*, pages 4862–4876, 2023.

[64] Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *Empirical Methods in Natural Language Processing*, pages 15686–15702, 2023.

A More Details about the Editing Benchmark

A.1 Natural Image Subset

Table A: Statistics of the natural image subset. The first column lists identifiers for each object category in ImageNet-1k. The "Class Name" in the second column is in the format as "prediction by the stronger model"-"prediction by the base model."

Group Identifier	Class Name	Sample Number
890-430	`volleyball-basketball`	123
933-923	`cheeseburger-plate`	133
470-644	`candle-matchstick`	113
900-437	`water tower-beacon`	159
609-586	`jeep-half track`	410
543-422	`dumbbell-barbell`	240
879-762	`umbrella-restaurant`	49
417-865	`balloon-toyshop`	75
573-751	`go-kart-racer`	172
880-671	`unicycle-mountain bike`	149
954-582	`banana-grocery store`	75
752-890	`racket-volleyball`	137
640-539	`manhole cover-doormat`	80
407-654	`ambulance-minibus`	155
562-975	`fountain-lakeside`	155
888-718	`viaduct-pier`	129

We divide the MAD-searched natural image subset into 16 groups, whose statistics are listed in Table A. Visual examples in each group are shown in Figs. A and B. These images are sourced from Flickr, prior to the advent of Stable Diffusion, and are licensed under creative commons.

A.2 AI-generated Image Subset

We adopt Textural Inversion [56] and PUG [5] to construct the AI-generated image subset, encompassing the oil painting and stage light shifts, respectively. The statistics are given in Table B.

Specific classes in the oil painting subset include `stingray`, `bullfrog`, `box turtle`, `garter snake`, `harvestman`, `crayfish`, `hermit crab`, `mongoose`, `rhinoceros beetle`, `weevil`, `wood rabbit`, `capuchin`, `african elephant`, `breastplate`, `drumstick`, `envelope`, `hand blower`, `shovel`, `spatula`, `syringe`, `wine bottle`, and `corn`.

Specific classes in the stage light subset include `barrel`, `cofee mug`, `washer`, `jack o lantern`, `vase`, `throne`, `soccer ball`, `basketball`, `car wheel`, `vacuum`, `birdhouse`, `laptop`, `piano`, `pool table`, `carousel`, `jellyfish`, `convertible`, `motor scooter`, `mask`, `sewing machine`, `hay`, `gasmask`, `bell pepper`, `drum`, `table lamb`, `backpack`, `chicken hen`, `tennis ball`, `safe`, `pay phone`, `cabbage`, and `pineapple`.

Visual examples of the oil painting and stage light images are shown in Fig. C and Fig. D, respectively.

A.3 Potential Dataset Filtering

Recall that the editing benchmark is designed to challenge the ViT/B-16 model. Thus, it is likely that some images might not induce predictive errors in other base models, which vary in terms of training data, model architecture, loss function, and optimization pipeline. For the ViT/S-16 model, the benchmark is subject to an additional filtering process based on its predictions. Consequently, 65% of the natural images and 100% of the AI-generated images are retained.

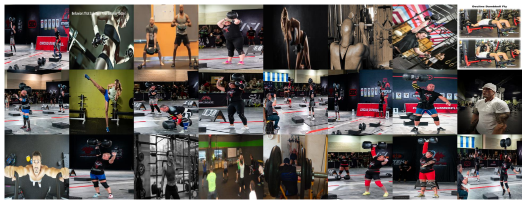
Group 890-430: `volleyball-basketball`

Group 933-923: `cheeseburger-plate`

Group 470-644:`candle-matchstick`

Group 900-437: `water tower-beacon`

Group 609-586: `jeep-half track`


Group 543-422: `dumbbell-barbell`

Group 879-762: `umbrella-restaurant`

Group 417-865: `balloon-toyshop`

Figure A: Visual examples in each group of the natural image subset. Part 1/2.

Table B: Statistics of the AI-generated image subset.

Group	Class Number	Sample Number
oil painting	22	860
stage light	32	1,092

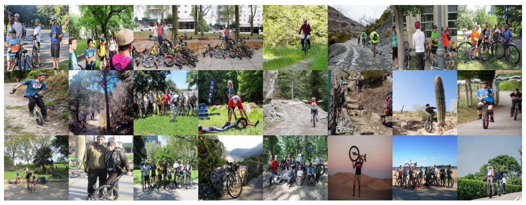
B More Experimental Details

In this section, we give more implementation details of the proposed and competing model editing methods. Algorithm 1 presents the pseudo-code of our method.

B.1 More Details of Our Method

Decoupling Trick In meta-learning, optimization of the hypernetwork entails differentiating the outer-loop loss with respect to the output of the inner loop $\phi^{(T)}$, and propagating the gradient through the inner-loop optimization to the output of the hypernetwork $\hat{m}$ (approximated by Eq. (4)), and finally to the parameters of the hypernetwork, φ. This extended chain of computation not only demands substantial computational resources but also hampers efficient optimization. To mitigate these, we decouple the pathway of hypernetwork optimization from the meta-learning gradient. Specifically, we introduce an auxiliary variable $\tilde{m}$, matching the dimensionality of $\hat{m}$, to substitute

Group 573-751: `go-kart-racer`

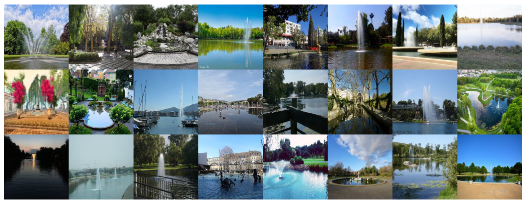

Group 880-671: `unicycle-mountain bike`

Group 954-582: `banana-grocery store`

Group 752-890: `racket-volleyball`

Group 640-539: `manhole cover-doormat`

Group 407-654: `ambulance-minibus`


Group 562-975: `fountain-lakeside`

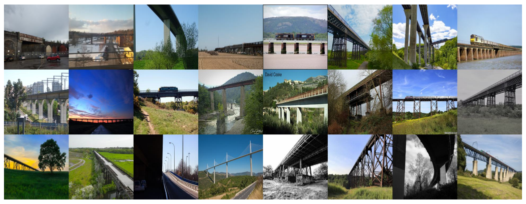

Group 888-718: `fountain-lakeside`

Figure B: Visual examples in each group of the natural image subset. Part 2/2.

for the hypernetwork's output during bi-level optimization. As a result, $\phi^{(T)}$ is now dependent on $\tilde{m}$, rather than $\hat{m}$. We first optimize the auxiliary variable:

$$\tilde{m}^\star = \arg\min_{\tilde{m}} \ell\left(x, y; \phi^{(T)}\right) + \lambda \|\tilde{m}\|_1. \tag{11}$$

Subsequently, $\tilde{m}^\star$ directs the parameter optimization of the hypernetwork using the element-wise KL divergence averaged across all positions:

$$\varphi^\star = \arg\min_{\varphi} \frac{1}{\dim(\tilde{m}^\star)} \sum_i \mathrm{KL}\left(g_i\left(e\left(x; \phi^{(0)}\right); \varphi\right), \tilde{m}_i^\star\right), \tag{12}$$

where i is the positional index and $\dim(\tilde{m}^\star) = N_m \times 6$ in our implementation.

Pseudo-sample Generation When applying CutMix, we vary the sizes of the pasted patches from 48×48 to 128×128, ensuring the preservation of the primary structural and textural details in the original images, which are 224×224 in size.

Hypernetwork Architecture We design the hypernetwork to mirror the architecture of its corresponding base model (*i.e.*, ViT/B-16 or ViT/S-16), with the same input and intermediate dimensions. Nevertheless, we reduce the number of attention blocks to five.

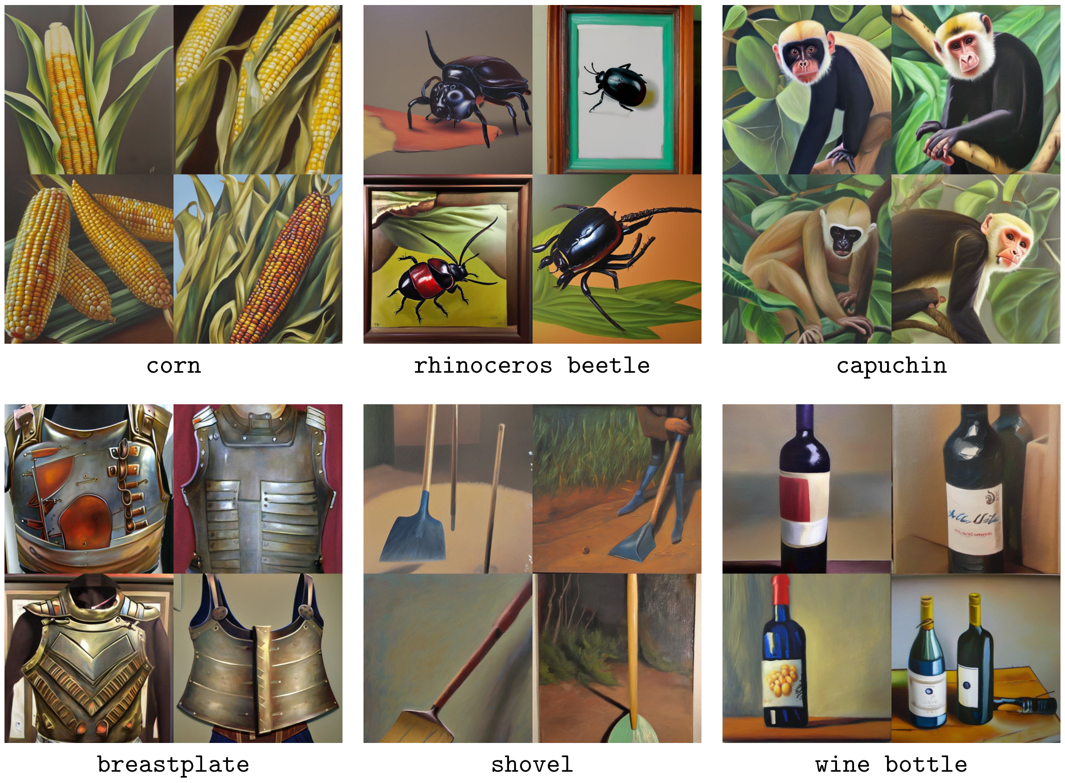


Figure C: Visual examples of the AI-generated oil painting images.

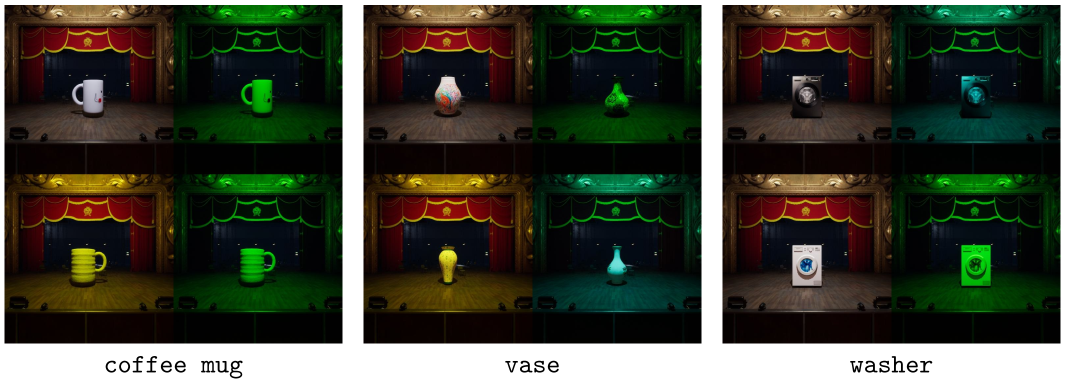


Figure D: Visual examples of the AI-generated stage light images.

Hyperparameter Configuration We set the learning rate in the inner loop as 0.001, and perform gradient descent for five steps (*i.e.*, $T = 5$). In the outer loop, we apply the Adam optimizer with a learning rate of 0.1 to optimize $\tilde{m}$ from random initialization for a total of ten steps. For the hypernetwork optimization, RMSProp[5] is utilized with a learning rate of 10^{-4}, a minibatch size of eight, and a maximum iteration number of $7,000$. Training a hypernetwork for the base ViT/B-16 takes approximately 9 hours on a single RTX A6000 GPU (48G).

[5]`https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`

Algorithm 1 Hypernetwork Meta-training via Standard Implementation / Decoupling Trick

```
Require: Hypernetwork, $g(\cdot;\varphi)$; ViT feature extractor, $e\left(\cdot;\phi^{(0)}\right)$; CutMix dataset, $\mathcal{B}$; training iteration, $\mathrm{MaxIter}$; inner-loop learning rate, $\alpha$; inner-loop step, $T$; outer-loop hypernetwork learning rate, $\beta$; outer-loop $\tilde{m}$ learning rate, $\gamma$; outer-loop step, $\mathrm{MaxOuterIter}$; trade-off parameter, $\lambda$
1: Randomly initialize $\varphi$
2: for $\mathrm{MainIter}$ = 1 to $\mathrm{MaxIter}$ do
3:     Create a CutMix dataset $\mathcal{B}$ from ImageNet-1k
4:     for $(x', x) \in \mathcal{B}$ do
5:         Calculate $\hat{m} = g\left(e\left(x;\phi^{(0)}\right);\varphi\right)$        // Approximated by Eq. (4)
6:         Set $\Delta\phi^{(0)} = 0$
7:         for $t = 1$ to $T$ do
8:             $\Delta\phi^{(t)} = \Delta\phi^{(t-1)} - \alpha\nabla_\phi\ell\left(x, p\left(y|x';\phi^{(0)}\right);\phi^{(t-1)}\right)$
9:             $\phi^{(t)} = \phi^{(0)} + \hat{m}\odot\Delta\phi^{(t)}$
10:        end for
11:        $\varphi \leftarrow \varphi - \beta\nabla_\varphi\left[\ell\left(x, p\left(y|x';\phi^{(0)}\right);\phi^{(T)}\right) + \lambda\|\hat{m}\|_1\right]$
12:        Randomly initialize $\tilde{m}$
13:        for $\mathrm{OuterIter} = 1$ to $\mathrm{MaxOuterIter}$ do
14:            for $t = 1$ to $T$ do
15:                $\Delta\phi^{(t)} = \Delta\phi^{(t-1)} - \alpha\nabla_\phi\ell\left(x, p\left(y|x';\phi^{(0)}\right)\phi^{(t-1)}\right)$
16:                $\phi^{(t)} = \phi^{(0)} + \tilde{m}\odot\Delta\phi^{(t)}$
17:            end for
18:            $\tilde{m} \leftarrow \tilde{m} - \gamma\nabla_{\tilde{m}}\left[\ell\left(x, p\left(y|x';\phi^{(0)}\right);\phi^{(T)}\right) + \lambda\|\tilde{m}\|_1\right]$
19:        end for
20:    end for
21:    $\varphi \leftarrow \varphi - \beta\nabla_\varphi\mathrm{KL}\left(\hat{m},\tilde{m}\right)$
22: end for
```

B.2 Implementation Details of Competing Methods

For methods that involve updating the base model parameters through backpropagation—including FT, FT-ℓ_2, KN [8], SPT [23], and our method—we follow [9] and adopt RMSProp as the optimizer, where the learning rate is set to 2×10^{-5} for ViT/B-16 and 10^{-4} for ViT/S-16, respectively.

T-Patcher [29] adds one neuron in the last FFN, together with a trainable multiplier initialized as 10. The new parameters are optimized using Adam with a learning rate of 5×10^{-3}.

ROME [38] employs Adam with a learning rate of 0.01 to obtain the target hidden representations of the last FFN, and then solves a constrained least squares problem to update the second FC layer.

We follow the default setting in LoRA [28], adding learnable matrices with a rank of eight. These low-rank matrices are optimized by Adam with a learning rate of 10^{-4}.

For KE [9] and MEND [9], we adhere to their training protocols to edit the six FC layers within the last three FFNs. The hypernetworks are meta-trained on editing samples sourced from ImageNet-1k to alter the base model's predictions to match the top-k randomly selected classes. The optimizer is Adam [31] with a learning rate of 10^{-5}.

C More Experimental Results

C.1 More Editing Results for ViT/B-16

In the main paper, we report the averaged editing results for ViT/B-16 across the sixteen groups in the natural image subset. Here, we further report the editing results on each group in Fig. E.

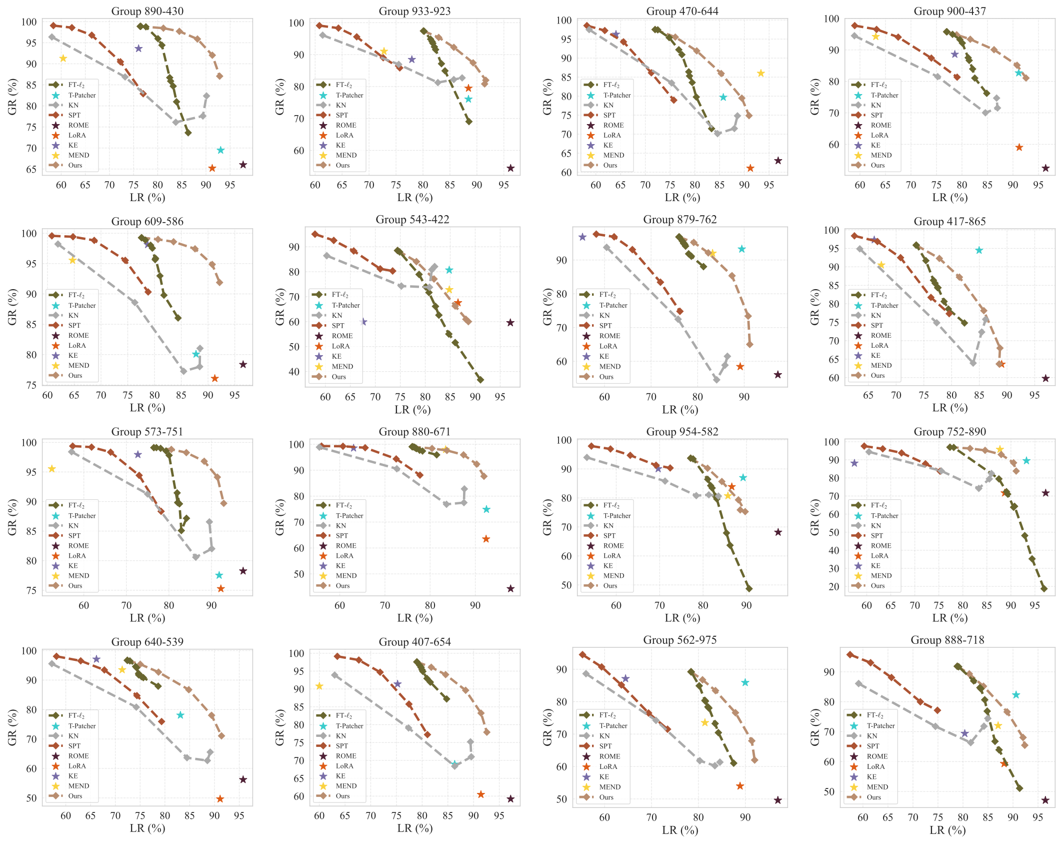


Figure E: Editing results for ViT/B-16 on the sixteen groups in the natural image subset.

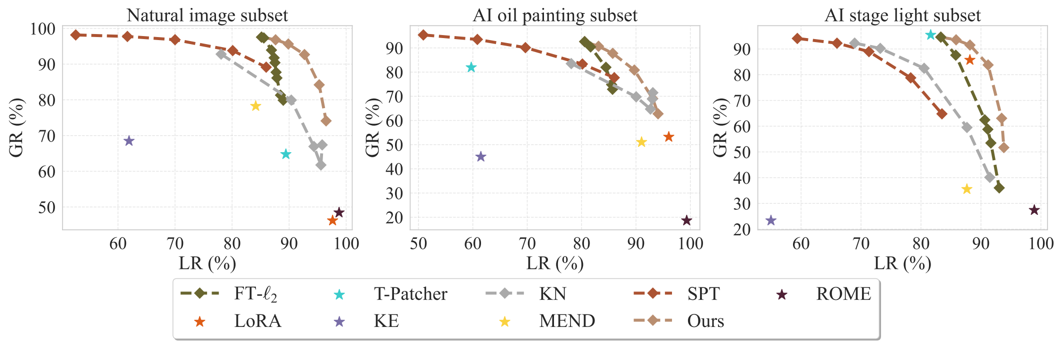


Figure F: Editing results for ViT/S-16 on the proposed benchmark.

C.2 Editing Results for ViT/S-16

Fig. F presents the editing outcomes for ViT/S-16, where our method continues to exhibit the optimal generation-locality trade-off, demonstrating its adaptability across various model architectures. Meanwhile, Fig. G presents the editing results on each group in the natural image subset.

C.3 More Analysis

We present the training curves of the hypernetwork in Fig. H. We find that the mask sparsity increases rapidly at the beginning of training from 0.0 to 0.86, which poses challenges for successful edits. As training progresses, the mask sparsity stabilizes while the KL divergence decreases. This suggests that the hypernetwork has effectively located key parameters relevant to successful edits.

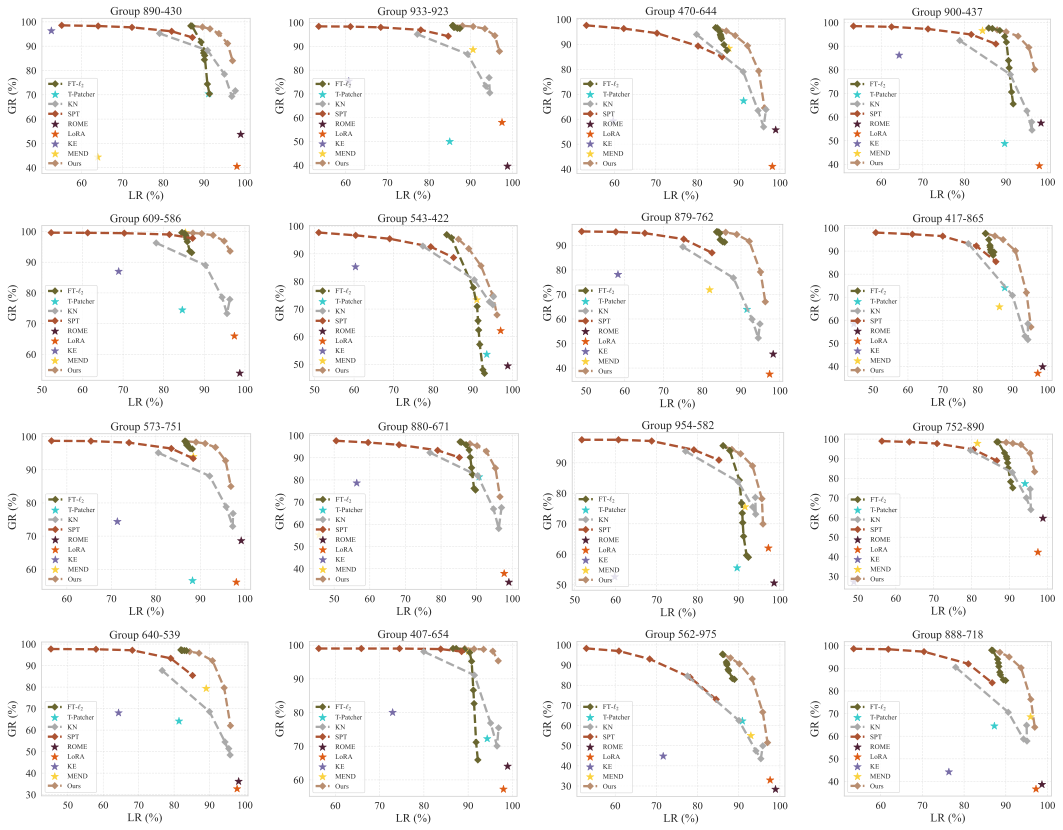


Figure G: Editing results for ViT/S-16 on the sixteen groups in the natural image subset.

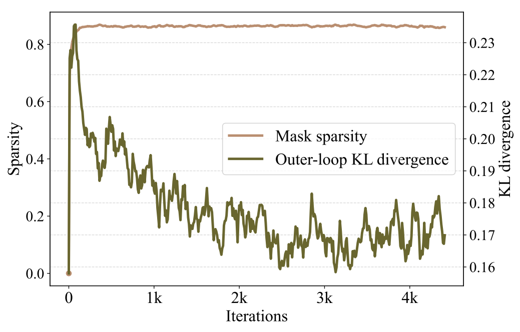


Figure H: Training curves of the hypernetwork.

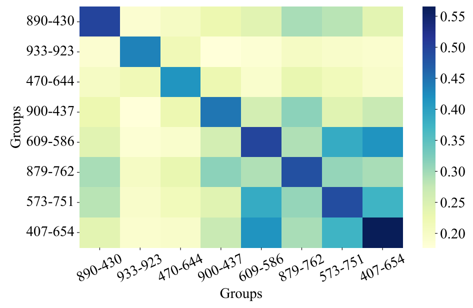


Figure I: Binary mask IoU results for samples among eight groups of the natural image subset.

C.4 Ablation Studies

Mask Specificity We further compute the averaged IoU results of the binary masks at the 0.95 sparsity level for editing samples among eight groups in the natural image subset. The results in Fig. I show that the identified parameters exhibit substantial overlaps for samples within the same group and much lower overlaps for samples from different groups.

Alternative Strategy for Pseudo-sample Generation We examine another more computationally expensive pseudo-sample generation strategy, *i.e.*, PGD [37], which has been validated to capture diverse distribution variations [19, 48]. Given a natural image x' with the label y' in the pre-training set, we apply PGD [37] on x' to obtain the pseudo-sample x with the prediction different from y'. We set the number of attack steps to 10 with a step size of $2/255$, under the feasible set of

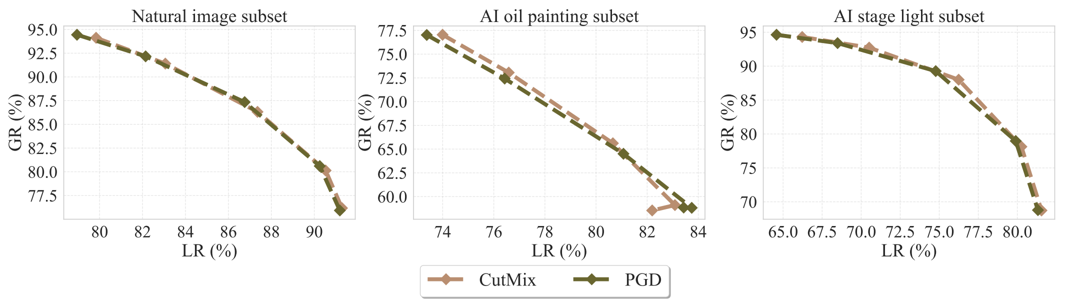


Figure J: Editing results for ViT/B-16 on the proposed benchmark, using the hypernetworks meta-trained by two different pseudo-sample generation approaches.

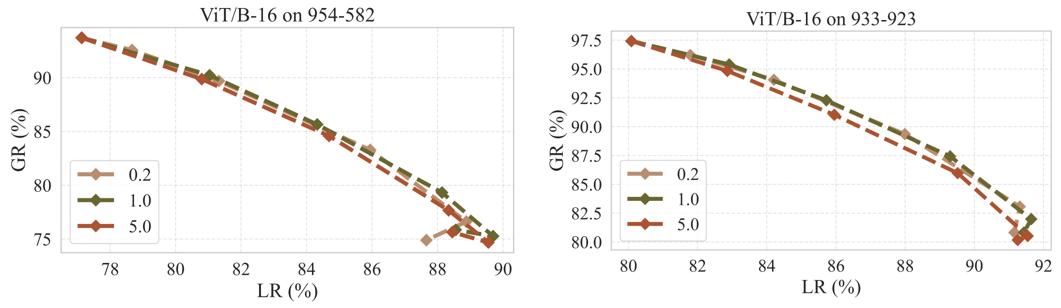


Figure K: Ablation results of the hyperparameter λ in the outer-loop optimization of Problem (2).

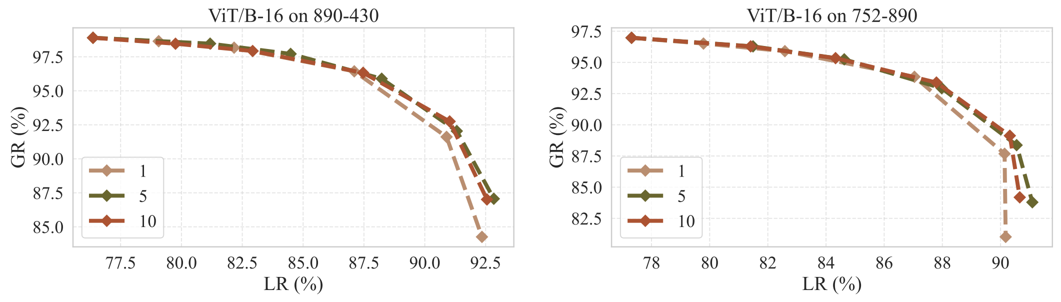


Figure L: Ablation results of the gradient step T in the inner loop.

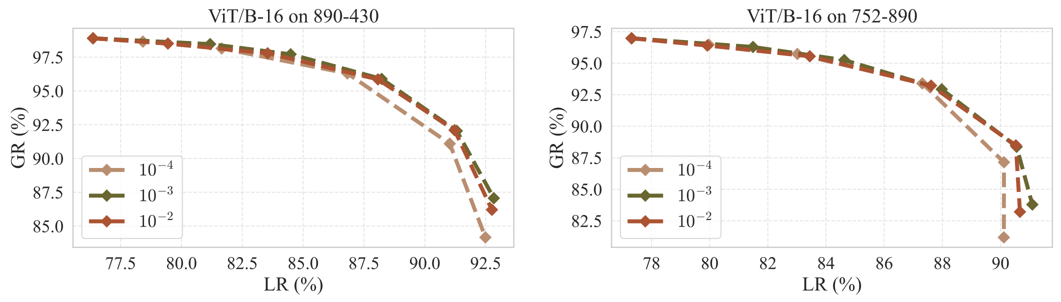


Figure M: Ablation results of the learning rate in the inner loop.

$\ell_\infty(x, x') \leq 8/255$. During training, we employ the cross-entropy loss $\ell\left(x, y'; \phi^{(t)}\right)$ to correct the prediction of x.

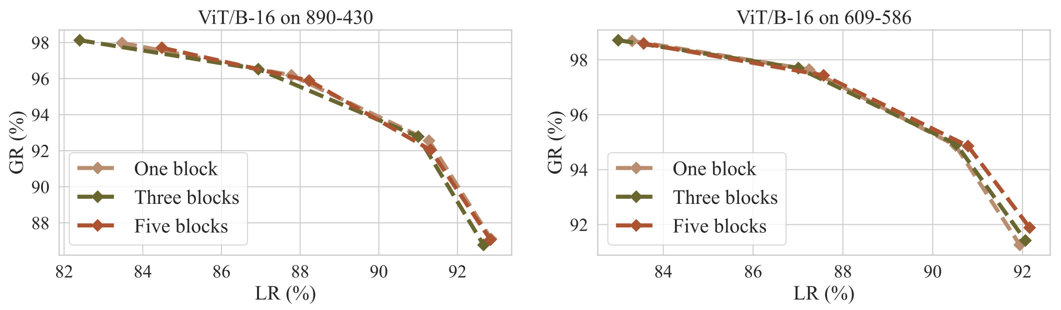


Figure N: Ablation results of the number of attention blocks in the hypernetwork.

Fig. J shows the editing results of two hypernetworks meta-trained using the two different pseudo-sample generation approaches. Remarkably, the simple CutMix rivals PGD in simulating distribution shifts, even in the two AI-generated image subsets.

Sparsity Regularization in the Outer Loop In the outer loop, we introduce a trade-off hyperparameter, λ, to balance the reliability objective with the sparsity regularizer. Here, we explore the impact of λ and observe that the sensitivity of hypernetwork to this trade-off parameter is minimal, as shown in Fig. K.

Gradient Step in the Inner Loop For the gradient step, T, in the inner loop, we test values of $\{1, 5, 10\}$. The performance of ViT/B-16 for each setting is illustrated in Fig. L, where we find that one gradient step yields slightly inferior results compared to more steps. Five and ten steps perform similarly, yet ten steps have greater training costs. Thus, we opt for five gradient steps as the default.

Learning Rate in the Inner Loop We explore the impact of the learning rate in the inner loop with values from $\{10^{-4}, 10^{-3}, 10^{-2}\}$. The editing results shown in Fig. M indicate that a lower learning rate (*i.e.*, 10^{-4}) exhibits slightly inferior performance than a larger learning rate. This may arise because a lower learning rate results in minimal updates to the base model within five gradient steps, thereby ineffective in guiding the hypernetwork training.

Number of Attention Blocks We additionally conduct ablative experiments to evaluate the impact of the number of attention blocks in the hypernetwork. We test values of $\{1, 3, 5\}$, and the editing performance for ViT/B-16 is illustrated in Fig. N, where we find that a small hypernetwork can achieve comparable performance to larger hypernetworks. Decreasing the number of attention blocks in the hypernetwork from five to three, and to one, does not incur a noticeable performance drop.

D Limitations

See the Conclusion and Discussion section in the main text.

E Broader Impact

Model editing has a broad impact by accelerating innovation in AI development through rapid iterations and refinements without extensive retraining, thus conserving resources and reducing environmental impact. The proposed method enables error correction of CV models, thereby enhancing adaptability and accessibility. We believe our method has great potential in addressing ethical concerns by mitigating biases and improving fairness in CV applications, while also increasing the robustness of CV systems against security threats like adversarial attacks.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes] ,

   Justification: Limitations are discussed in the Appendix D.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Implementation details are provided in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code is available at `https://github.com/hustyyq/Where-to-Edit`.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Details are provided in the Appendix A and B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Fine-tuning on a specific example is determinative when no additional randomly initialized parameters are introduced. In this case, there is no associated error bar.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Computer resources are provided in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: In appendix E.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer:[Yes]

    Justification: For the 2354 images scraped from the Internet, we check the images manually to ensure the safety.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The authors cite the original paper that produced the dataset used in this paper. The Internet-searched images from the website are used under the copyright, which is stated in Appendix A.1.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: The paper does not release new assets before acceptance.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [No]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.