Generative Adversarial Attacks on Image Classification

Jai Rastogi
Department of Computer Science
University of Maryland
College Park, USA
jrastogi@terpmail.umd.edu

***Abstract*—Although neural networks are widely used in real-world systems, their outputs are often unexplainable. This makes them prime targets for adversarial attacks, where imperceptible amounts of noise are added to the image to change the output in some way. We present a novel problem formulation of adversarial attacks on image classifiers that softly minimizes the noise, given that adding the noise maintains a different classification. This relaxes the hard constraint formulation from existing works. Our corresponding attack samples an action from a trained normal distribution, and uses it to reconstruct both noise and a saliency map in image space. Early experimental results on the the YOLOv11 image classifier, trained with the Soft Actor-Critic algorithm on the ImageNet dataset, showed consistent convergence into a suboptimal local minima. Despite subpar results, this still highlights some inductive biases present in the training of the classifier. Later experimental results attempted to exploit saliency maps and other features of previously successful attacks, but the additional moving parts resulted in poor results. Regardless, with more tuning, we conclude that this approach offers a promising direction in both adversarial attacks and increasing explainability in the gray areas between the structure of two discrete classes of images.**

***Index Terms*—adversarial attacks, black-box, image classification, image generation, saliency maps**

I. Overview

Deep learning models are universal function interpolators and perform well on supervised learning classification tasks, such as image classification. This expressivity comes from the nonlinear activation functions, which yield nonlinear decision boundaries in the input space. However, the train, test, and validation data to construct these models is structured to reflect real visual phenomena, and is roughly independent and identically distributed (i.i.d.). As a result, a small change to the input can significantly change the predicted output. The construction and application of this change to force incorrect results is called an adversarial attack. In images, the picture after the application of an adversarial attack looks nearly identical to the original to humans. Given the widespread use of deep learning-based software and decision-making systems, adversarial attacks pose major safety concerns in real world applications, including autonomous cars [1] and security systems [2]. Understanding these attacks is necessary to protect these real-world systems from being fooled.

In practice, adversarial attacks are black box attacks, where the victim model (model being attacked) only performs inference and only attributes like the predicted class, bounding box location, and confidence are available. Information about gradients and training data is unknown, which makes it challenging for strong attacks.

In this paper, we present a novel attack strategy. Instead of choosing n pixels, for small n, and applying fixed large magnitude changes to them, we reconstruct latent Gaussian noise into image space, with the goal of learning the minimal number and configuration of pixels with variable large magnitude to change the label. The remaining pixels should have negligible weight, forming a model of structural sparsity. By reversing the order of the optimization problem, we leave it up to the model to decide the perturbation necessary to reach and cross the nearest non-linear decision boundary between the classes in high-dimensional space. We show this approach grants valuable insights into the victim model. In particular, it exposes inductive biases learned during training, and has the potential to analyze the nonlinear decision boundaries that distinguish classes from each other in high-dimensional space.

II. Background

A. Deep Learning Models

The task of image classification is to identify the one label that best describes the contents of the entire image. Models generally apply the same Convolutional Neural Network (CNN) architecture: a backbone consisting of one or both convolutional and attention layers is applied to an input image to extract important high-level features that characterize the image, and these features pass through other layers in the network (called the classification head) to extract the probability of each class. The classifier returns the class with the highest probability. In general, for an input image $\mathbf{x} \in \mathbb{R}^{C \times W \times H}$, an image classifier f, and a set of all possible classes $Y = \mathbb{Z} \cap [0, n]$ for $n \in \mathbb{N}$, the predicted output $\hat{y}$ is given by $\hat{y} = f(x) \in Y$.

We apply our adversarial attacks to YOLOv11 [3], which is a single-stage algorithm that uses spatial attention and convolutional layers for both tasks. For image classification, the goal is to construct a minimal perturbation that changes the predicted class of the image classifier. The new class can be any other class in the dataset.

B. Embeddings

Raw image data is not suitable for neural network learning because each pixel constitutes one dimension, resulting in extremely high dimensional spaces. Instead, neural networks can be used to train embeddings, or lower-dimensional latent spaces that capture higher-level features of the image, such as shapes and object locations, while increasing computational efficiency and generalization. One neural network architecture for computing embeddings is ResNet [4]. It is pre-trained on the ImageNet [5] dataset, which has images normalized to $\mathcal{N}(\mu_{\text{img}}, \sigma_{\text{img}})$, where

$$\begin{aligned} \mu_{\text{img}} &= [0.485, 0.456, 0.406] \\ \sigma_{\text{img}} &= [0.229, 0.224, 0.225] \end{aligned} \tag{1}$$

Let $\mathbf{x}' \in \mathbb{R}^d$ denote the latent embedding of image $\mathbf{x}$, where the pixels of $\mathbf{x}$ are in $[0, 1]$ and d is the dimension of the produced latent space. Then, the latent embedding is computed as follows:

$$\mathbf{x}' = \text{ResNet}\left(\frac{\mathbf{x} - \mu_{\text{img}}}{\sigma_{\text{img}}}\right) \tag{2}$$

However, by compressing images into small abstract representations, these embeddings lose some portion of the image's information. The scope of the loss of information depends on many factors, including the size of the embedding and the degree to which the input matches the distribution of the training data. Compared to a larger dimensional latent space, a smaller embedding is more computationally efficient but may not capture enough information about the image. The weights of these neural networks are achieved by training the models on real-world data, which is assumed to be independent and identically distributed. In inference, if the input image is generated from a sufficiently different distribution than the training data, the quality of the embedding will decrease.

C. Multivariate Gaussian Distributions

The multivariate Gaussian distribution models a random vector $\mathbf{x} \in \mathbb{R}^d$ with mean $\mu \in \mathbb{R}^d$ and covariance $\Sigma \in \mathbb{R}^{d \times d}$, where Σ is symmetric and positive semi-definite. It captures linear dependencies.

A stronger form of the covariance matrix conditions (symmetry, positive definite) is satisfied using the Cholesky Decomposition:

$$\Sigma = \mathbf{L}\mathbf{L}^T + \epsilon\mathbf{I} \tag{3}$$

where $\mathbf{L}$ is a lower triangular matrix, $\mathbf{I}$ is the identity matrix, and ϵ is a small number. Representing Σ in this way verifies numerical stability and saves memory.

D. Adversarial Attacks

An adversarial attack is the process of producing and applying noise or other artifacts to images that successfully induce incorrect results from the neural network. These altered images are nearly indistinguishable from the original samples. Mathematically, as stated in [6], the general form of an adversarial attack is as follows:

$$\begin{aligned} &\arg\max_{\Delta x} P_{\text{adv}}(x_{\text{adv}}), \\ &\quad x_{\text{adv}} = x + \Delta x, \\ &\quad \text{s.t. } \|\Delta x\|_p \leq \epsilon \end{aligned} \tag{4}$$

where x_{adv} is the adversarial noise, P_{adv} is the probability that the attacked image returns a different label from the original, and x is the original image. Δx is the perturbation, with its L_p norm bounded by some small $\epsilon \in \mathbb{R}^+$.

There are two forms of adversarial attacks: white-box and black-box attacks. White-box attacks manipulate knowledge during the victim model's training process, such as altering training data and backpropagation gradients. In black box models, the victim model only performs inference, and only attributes like the predicted class and confidence values are available. While black box models are more realistic, more research so far has focused on white box models, and defending black box attacks is necessary to develop secure systems.

There are two types of black box attacks: transfer-based and query-based models. Transfer based methods involve training adversarial examples for a surrogate model and transferring it to the victim model. Since this doesn't involve the victim model, any differences in training data, model architecture, and so on between the surrogate and victim models decrease the success rate and efficiency of this approach. Query-based models repeatedly query the victim model with modified images, yielding higher success rates, but require a significant number of queries. Within query-based models, some methods attack the entire image, while others only modify a specifically chosen subset of pixels.

Early approaches to adversarial attacks on image classification visibly changed the image, using techniques with unrestricted L_p norms like rotations [7] and a selective coloring of large regions in the image [8]. These approaches were phased out in favor of attacks with restricted L_p norms, starting with OnePixel [9]. This technique used differential evolution algorithms to choose a pixel, to which a large magnitude color change would be applied. PIXLE [10] optimized query efficiency from OnePixel by using a random search. ScratchThat [11] used differential evolution to choose pixels that emulate a Bèzier curve and change them. PatchAttack [12] used policy gradient methods to choose the location where a cropped textured patch of a different class is inserted, requiring few queries but relatively large, visible alterations of the image. RLAB [13] improved on PatchAttack by using Dueling DQN and Gaussian noise perturbations, decreasing the magnitude and visibility of the attack. More recently, RFPAR [14] applied one-step REINFORCE to an image to find the N pixels it would alter, store the perturbed image with the highest reward after convergence of REINFORCE, and continue training with the perturbed image as the input. Notably, these more recent approaches bound the magnitude of their attack explicitly through the L_2 and L_∞ norms, and implicitly use the L_0 norm in the problem setup.

E. Soft Actor-Critic

Soft Actor-Critic is an off-policy algorithm that uses entropy regularization and a replay buffer $\mathbf{B}$ to optimize a stochastic policy. Given two target critic networks, it trains an actor network $\pi_\theta(a \mid s)$ and two critic networks $Q_{\phi_i}(s,a)$, as follows:

Given a batch $B \sim \mathbf{B}$:

$$y(r,s',d) = r+\gamma(1-d)\left(\min_{i=1,2} Q_{\text{targ},i}(s',a') - \alpha \log \pi_\theta(\tilde{a}' \mid s')\right) \tag{5}$$

for $\tilde{a}' \sim \pi_\theta(\cdot \mid s')$.

For $i = 1, 2$, the critic loss is as follows:

$$L(\phi_i) = \frac{1}{|B|} \sum_{(s,a,r,s',d)\in B} (Q_{\phi_i}(s,a) - y(r,s',d))^2 \tag{6}$$

The actor loss is as follows:

$$L(\theta) = \frac{1}{|B|} \sum_{s\in B} \left(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s) \mid s)\right) \tag{7}$$

where $\tilde{a}_\theta(s)$ is a differentiable sample of $\pi_\theta(\cdot \mid s)$ with respect to θ through the reparametrization trick.

III. Methods

A. Problem Formulation

The problem setup for our approach to adversarial attacks on image classification is as follows:

$$\begin{gathered} \arg\min_{\Delta x} \text{ s.t.} \\ \arg\max_{\hat{y}\in Y} P_{cls}(\hat{y} \mid x) \neq \arg\max_{\hat{y}\in Y} P_{cls}(\hat{y} \mid x_{\text{adv}}), \\ x_{\text{adv}} = x + \Delta x \end{gathered} \tag{8}$$

where $P_{\text{cls}}(\hat{y} \mid x)$ is the probability of the image classifier predicting label $\hat{y}$ for input x. The goal is to reach structured sparsity, where a dynamically allocated (but as small as possible, on the order of less than 1% of the image) set of pixels change values significantly to change the class while the other pixels remain unchanged. This differs from past work in that no hard limit is set on Δx, and the search is to minimally cross a decision boundary, as opposed to getting as far away from the original class as possible within the given boundaries.

For this optimization, we construct the following MDP, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, as follows:

$$\mathcal{S} = \{\mathbf{s} \in \mathbb{Z}^{3\times W\times H} \cap [0,1]^{3\times W\times H} \mid \arg\max_{\hat{y}\in Y} \; P(\hat{y} \mid s) \geq 0.2\} \tag{9}$$

The state space is composed of images with 3 channels (red, green, blue), width W, and height H. Each pixel value is normalized to $[0, 1]$ for numerical stability. Each image must be somewhat identifiable to the image classifier, with at least one label that has a confidence of at least 0.2. This means fully random noise is invalid.

$$\mathcal{A} = \{\mathbf{a} \in \mathbb{R}^{3\times W\times H} \cap [0,1]^{3\times W\times H}\} \tag{10}$$

The action space consists of all combinations of noise for each channel.

$$\mathcal{T}(s' \mid s, a) = \begin{cases} 1 & s' = \text{clamp}(s+a, 0, 1) \\ 0 & \text{else} \end{cases} \tag{11}$$

where

$$\text{clamp}(x, a, b) = \max(a, \min(x, b)) \tag{12}$$

The transition function deterministically adds the generated adversarial noise to the original state.

Define the old and new labels of the image as follows:

$$\hat{y}_{\text{old}} = \arg\max_{\hat{y}} P_{\text{cls}}(\hat{y} \mid s) \tag{13}$$

$$\hat{y}_{\text{new}} = \arg\max_{\hat{y}} P_{\text{cls}}(\hat{y} \mid s') \tag{14}$$

Assume the predicted label from the original image is the ground truth true label. Define the difference in confidence of this label between the original and attacked image as follows:

$$\Delta\text{conf}(s, s') = P_{\text{cls}}(\hat{y}_{\text{orig}} \mid s') - \max_{\hat{y}} P_{\text{cls}}(\hat{y} \mid s) \tag{15}$$

$$R(s,a,s',t) = \begin{cases} -5 - 2.5\Delta\text{conf}(s,s') - 0.075t & \hat{y}_{\text{old}} = \hat{y}_{\text{new}} \\ 10 - 0.075t & \text{else} \end{cases} \tag{16}$$

If the perturbation doesn't change the predicted class label, then a penalty is assigned. If $\Delta\text{conf}(s, s')$ is positive, then the perturbation increased the probability of assigning the true label to the altered image, which is the opposite of the desired goal. This increases the magnitude of the assigned negative reward. If $\Delta\text{conf}(s, s')$ is negative, the action helped but wasn't strong enough to override the original class label, so some positive reward is reallocated to that step. If the perturbation changes the predicted class label, a reward is assigned. In both cases, the reward is decremented by the step number in the episode. This is because each episode should finish as soon as possible. As a result, $\gamma = 0.5$ and the episode terminates when $\hat{y}_{\text{old}} \neq \hat{y}_{\text{new}}$.

However, reinforcement learning techniques applied to $\mathcal{M}$ will not generalize, and in many steps, will not train at all. This stems from the state and action spaces, and the transition function. Despite some loss of information, operating on a latent embedding of an image will generalize better because it contains more high-level information about the image's contents. This structure in the state embedding produces a corresponding structure in the action embedding, which can then be deconvolved to image space. Moreover, the clamping operation (12) in $\mathcal{T}$ is differentiated as follows:

$$\frac{d}{dx}\text{clamp}(x, a, b) = \begin{cases} 1 & x \in (a, b) \\ 0 & x \neq (a, b) \end{cases} \tag{17}$$

This implies that for any pixel in image space, $\mathbf{s}+\mathbf{a}$, where the clamp changes the value, the gradient is 0. This sets all backpropagation weights multiplied by that gradient to 0, or no weight updates. This depends on both the pixel values and

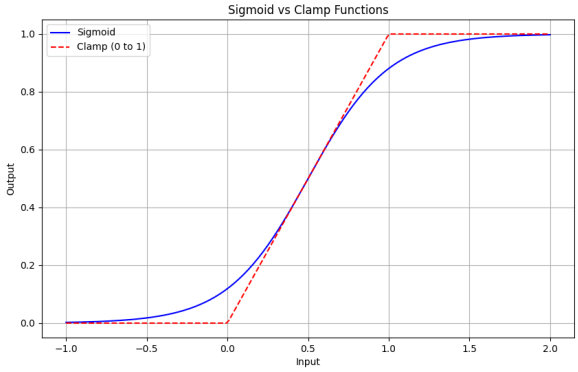

Fig. 1. Visualization of the difference between the clamp function, which has areas with zero gradient, and the softmax function, which has a nonzero gradient everywhere.

the random sampling of noise, so there would be few gradient updates in the early stages of training when the agent explores a lot.

Define a learnable MDP $\mathcal{M}' = \{\mathcal{S}', \mathcal{A}', \mathcal{T}', \mathcal{R}, \gamma\}$ as follows:

$$\mathcal{S}' = \{\text{Downsampler}\left(\text{ResNet}\left(\frac{\mathbf{s} - \mu_{\text{img}}}{\sigma_{\text{img}}}\right)\right) \mid \mathbf{s} \in \mathcal{S}\} \quad (18)$$

where the downsampler is a learnable linear layer that produces the target embedding dimension $d' \in \mathbb{Z}$, and μ_{img}, σ_{img} come from (1). To maintain the MDP formulation, we assume that every element in $\mathcal{S}'$ is a fully observable belief in the general state space $\mathcal{S}$.

$$\mathcal{A}' = \{\mathbf{a} \in \mathbb{R}^{d'}\} \quad (19)$$

This latent action space $\mathcal{A}'$ is a downsampled version of the original action space $\mathcal{A}$, with the same vector embedding dimension as $\mathcal{S}'$.

Given $\mathbf{x} \in \mathcal{S}$ such that $\mathbf{s}$ is the latent representation of $\mathbf{x}$, let:

$$\mathbf{x}' = \text{sigmoid}(4 * (\mathbf{x} + \text{decoder}(\mathbf{s} + \mathbf{a})) - 2) \quad \mathbf{x}' \in \mathcal{S} \quad (20)$$

$$T(s' \mid s, a) = \begin{cases} 1 & s' = \text{Downsampler}\left(\text{ResNet}\left(\frac{\mathbf{x}' - \mu_{\text{img}}}{\sigma_{\text{img}}}\right)\right) \\ 0 & \text{else} \end{cases} \quad (21)$$

where the sigmoid function, defined in (22), is applied element-wise.

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (22)$$

As shown in Figure 1, (20) matches (12) in the neighborhood of $x = 0.5$, and maps all pixels back to $[0, 1]$ while maintaining nonzero derivatives. While this approximation clearly has error, we assume it perfectly models the initial clamping behavior.

The decoder is a neural network used to reconstruct the noise in the dimensions of image space from the latent embedding. Specifically, it first upsamples the action $\mathbf{a} \in \mathbb{R}^{d'}$ to the dimension $\mathbb{R}^{CWH}$, and uses a series of transposed convolution, batch normalization, ReLU, and dropout layers to form an image of shape $C \times W \times H$. We also added a second network that learns a saliency map, or a one-channel image that reflects the importance of each pixel to the adversarial network.

Lastly, the reward function and discount factor are the same between $\mathcal{M}$ and $\mathcal{M}'$. We solve this proxy MDP $\mathcal{M}'$ through the Soft Actor-Critic (SAC) Algorithm [15].

B. Training Pipeline

The attack training pipeline is shown in Figure 2. For a given latent embedding $\mathbf{z} \in \mathcal{S}'$ of an image, we construct actions in this latent space, $\mathbb{R}^{d'}$, through samples of a Multivariate Gaussian distribution. The learnable parameters are the mean vector, $\hat{\mu}(\mathbf{z})$, and the lower triangular portion of the covariance matrix, $\Sigma(\mathbf{z})$. This sampled action ideally contains structured high-level features of the desired noise, and the sum of this latent action and the latent state is reconstructed to noise in image space. It is added to the original image through the differential clamp in (20), and the classifications from the original image and the adversarial image determine the reward and completion status of the step.

While our implementation solves the MDP $\mathcal{M}'$, the replay buffer in SAC contains experiences of the form (image $\mathbf{x} \in \mathcal{S}$, attacked image $\mathbf{x}' \in \mathcal{S}$, latent $\mathbf{a} \in \mathcal{A}'$, reward, completion status). This is because the latent embedding $\mathbf{s} \in \mathcal{S}'$ can always be extracted from $\mathbf{x}$. The opposite is untrue because downsampling is a many-to-one relation, removing invertibility. Since images require significant memory, we save the replay buffer in the local file system, using Zstandard [16], a lossless compression algorithm, for efficient storage.

The standard loss functions for SAC, (6) and (7), involve the latent action $\mathbf{a} \in \mathcal{A}'$, but don't backpropagate to the neural network that reconstructs the noise in image space. To train this neural network, we apply various combinations of the following loss functions to the actor loss (7):

- Since the perturbation should be minimal, the L_1 and L_2 norms on the noise in image space, and L_2 norms on the noise in the latent space are all used as loss functions, denoted $L_{1,\text{act}}$, $L_{2,\text{act}}$, and $L_{2,\text{lat}}$ respectively. This is directly optimized in the problem formulation.
- The adversarial image should look natural, implying that the pixels don't have major abrupt changes. Anisotropic Total Variation loss increases this smoothness and is defined as follows:

$$\mathcal{L}_{TV} = \frac{1}{C(H-1)W} \sum_{c=1}^{C} \sum_{i=1}^{H-1} \sum_{j=1}^{W} |I_{c,i+1,j} - I_{c,i,j}| + \frac{1}{CH(W-1)} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W-1} |I_{c,i,j+1} - I_{c,i,j}| \quad (23)$$

- Given the class of the original image, c, the classification loss, $\mathcal{L}_{\text{cls}}$ is the mean of the probability of the perturbed

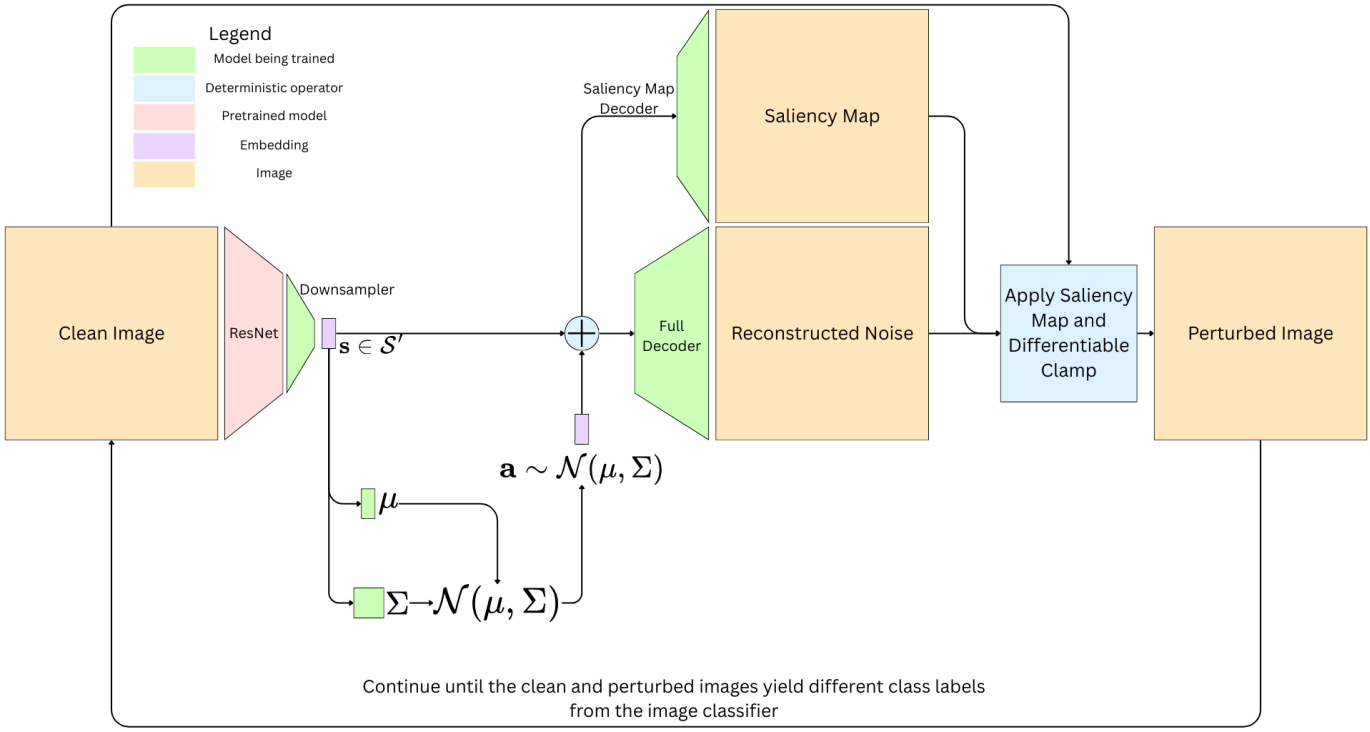

Fig. 2. Full pipeline of generating an action and applying it to the image, for each step.

image being class c. This is another direct quantity to be optimized. The classifier breaks the computation graph, so we manually reattach these results to the computation graph.

- Perceptual loss measures the L_2 norm between the latent high-level feature representations of the clean and perturbed images. This preserves semantic details about the objects and textures of the image, leading to more realistic and less blurry results. We use the Learned Perceptual Image Patch Similarity (LPIPS) [17] neural network to construct these latent embeddings because it is trained to mimic human judgment, although it returns a low loss for minor brightness shifts or deformations.
- Diversity loss encourages variability in the output. This is computed through the cosine similarity metric across actions (in both latent and image space) in the batch, denoted $\mathcal{L}_{\text{div, lat}}$ and $\mathcal{L}_{\text{div, act}}$ respectively. Similarly, the clean and perturbed images should become slightly decorrelated from each other, using the same cosine similarity logic; this loss is $\mathcal{L}_{\text{orth}}$.
- The action should not be smooth because it contradicts the goal of structured sparsity. The associated loss, $\mathcal{L}_{\text{high-freq}}$ convolves the perturbed image with the 3×3 Laplacian Kernel, which is a high-pass filter, and calculates the negative mean of convolved output.
- The perturbed image should not look darker than the original image to humans. This is quantified with perceptual luminance, or the conversion of the RGB image to grayscale, as follows:

$$\mathcal{L}_{\text{br}} = 0.299R + 0.587G + 0.114B \tag{24}$$

- In the saliency map, the importance of each pixel should be low (close to 0) or high (close to 1), enforcing the requirement of a few pixels with high magnitude and the other pixels with low or 0 magnitude. This mimics L_0 loss on the saliency map, as follows:

$$L_{0,\text{sal}} = \frac{1}{WH} \sum_{w \in W} \sum_{h \in H} \text{mask}(w,h) * (1 - \text{mask}(w,h)) \tag{25}$$

  Similarly, the L_1 loss on the saliency map, $L_{1,\text{sal}}$ encourages sparsity in a differentiable fashion. An alternative version of increasing sparsity, the area loss $\mathcal{L}_{\text{sal, area}}$, takes the sum of all values in the saliency map and calculates the mean squared error with the desired size of the perturbation.
- Since the goal involves a few large perturbations, this loss term, $\mathcal{L}_{\text{sal, large}}$ decreases the overall loss by a factor of the magnitudes of these action values. This counteracts the L_1 norm to satisfy the problem constraints.
- The remaining pixels should be small, so there is a loss term, $\mathcal{L}_{\text{sal, small}}$ to drive these pixels to 0.

IV. Results

This section presents the performance outcomes of various implementations of our proposed technique, using randomly chosen images from the validation set of ImageNet [5], resized

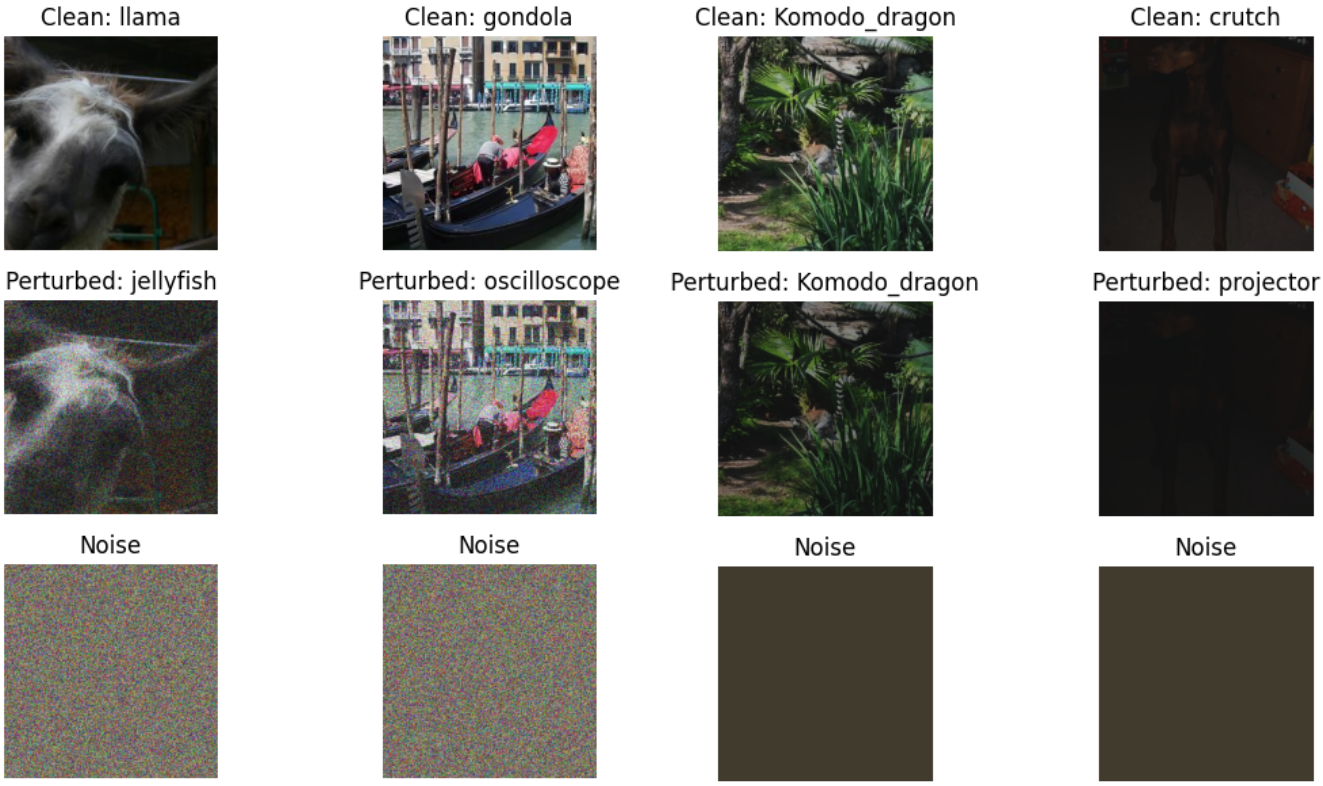

Fig. 3. Samples of random standard Gaussian noise being added to clean images, and the resulting classification.

Fig. 4. Samples of learned noise from model A being added to clean images, and the resulting classification.

and randomly cropped to $224 \times 224 \times 3$. Figure 3 displays four sample images and the predicted label from the image classifier in the first row, those same images and new labels after applying standard Gaussian noise, and a visual of the noise itself. Notably, the noise has a rough texture, making it apparent that the perturbed image is not natural. The tradeoff is that both examples are successfully misclassified after applying this noise. Similarly, Figures 4 and 5 show the same information for model A and E respectively, as trained with the hyperparameters specified in Table I, $32,000$ steps, and warm-starting training with a replay buffer of $10,000$ experiences. In figure 4, the noise is smooth and uniform across the entire image, but not all images are misclassified in one step. The second image has already had some noise applied to it, as visible from the unnatural darkening, and its perturbed version is nearly a uniform black image. In figure 5, the saliency map shows how low the active parts of the noise (bounded from 0 to 1) are, and how similar the clean and perturbed images actually are.

Table I quantifies the hyperparameters for the additional loss terms used to train the decoder part of the model, and the results on 32 thousand images from the validation set. Specifically, the L_1 and L_2 norms, and the average number of steps needed to change the class are calculated, where lower is better for all three values. The standard Gaussian noise yields the lowest number of steps necessary to change the class, but has large noise, while the learned noise yields lower noise per pixel but needs more iterations to change the label successfully.

TABLE I
HYPERPARAMETERS OF SELECTED TRAINED MODELS

Loss Term	**Models**				
	A	B	C	D	E
Spatial Saliency	No	No	No	Yes	Yes
$L_{1,\text{act}}$	2×10^{-5}	2×10^{-3}	0	1×10^{-3}	1×10^{-3}
$L_{2,\text{act}}$	1×10^{-2}	1×10^{-1}	0	0	0
$L_{2,\text{lat}}$	0	2	0	0	0
$\mathcal{L}_{\text{TV}}$	1	0	0	0	0
$\mathcal{L}_{\text{cls}}$	200	50	5	100	400
$\mathcal{L}_{\text{perc}}$	0	5	5	50	50
$\mathcal{L}_{\text{div, act}}$	0	500	0	0	0
$\mathcal{L}_{\text{div, lat}}$	0	2	0	0	0
$\mathcal{L}_{\text{orth}}$	0	0	0	0	100
$\mathcal{L}_{\text{high-freq}}$	0	0	0	0	10
$\mathcal{L}_{\text{br}}$	0	$10,000$	1000	$10,000$	$100,000$
$L_{0,\text{sal}}$	0	0	10	0	0
$L_{1,\text{sal}}$	0	0	20	1000	1000
$\mathcal{L}_{\text{sal, large}}$	0	0	50	20	20
$\mathcal{L}_{\text{sal, small}}$	0	0	10	1000	100
$\mathcal{L}_{\text{sal, area}}$	0	0	0	0.1	2×10^{-4}

Given images of size $224 \times 224 \times 3$, an L_1 norm of 37480 implies that the noise changes each pixel by 0.25 on average, where each pixel is a real value between 0 and 1. An L_1 norm of 67315.7 implies an average pixel change of 0.447.

Figures 6 and 7 present heatmaps of the top 8 most frequent

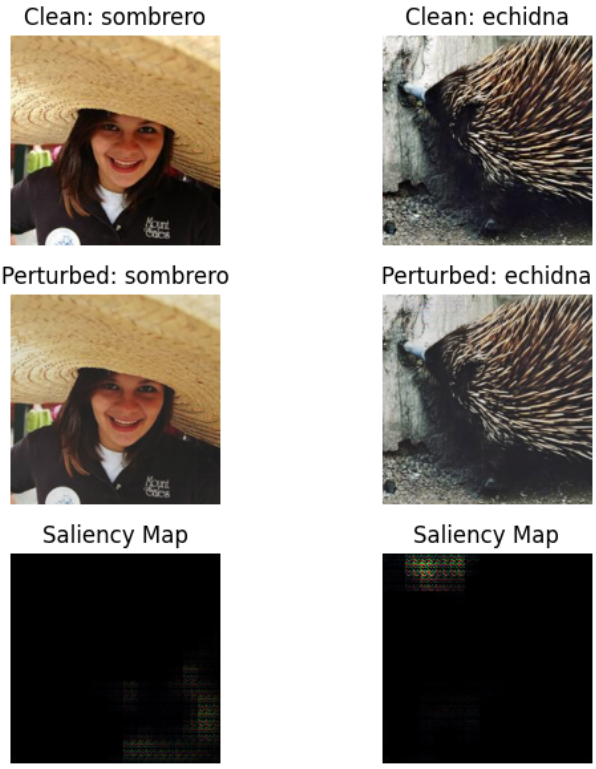

Fig. 5. Samples of learned noise from model E being added to clean images, and the resulting classification.

TABLE II
RESULTS OF SELECTED MODELS

Model	Results		
	$L_{1,\text{act}}$	$L_{2,\text{act}}$	Steps
Standard Gaussian Noise	68349.72	186.12	1.1770
A	37479.96	97.02	26.4946
B	37487.85	97.04	29.4683
C	37487.74	97.04	37.7721
D	67315.71	173.98	31.4347
E	67315.75	173.98	44.9888

class transitions after adding noise generated from a standard normal distribution and model A, respectively. Since a class change is required, the values on the principal diagonal must be 0. In 6, all classes that are not "jellyfish" transition to "jellyfish", while the transition distribution is more spread in 7.

V. DISCUSSION

The combinations of hyperparameters and loss terms applied in models A, B, and C yielded identical results: mild shading over the full image. This suggests that the actions are too correlated with the structure of the image and a mode collapse (lack in diversity among outputs) has occurred. One likely reason is that CNNs are sensitive to brightness changes. This occurs because they operate directly on pixel values, which have shadows or other inconsistencies that alter the latent feature representation, and therefore the predicted class.

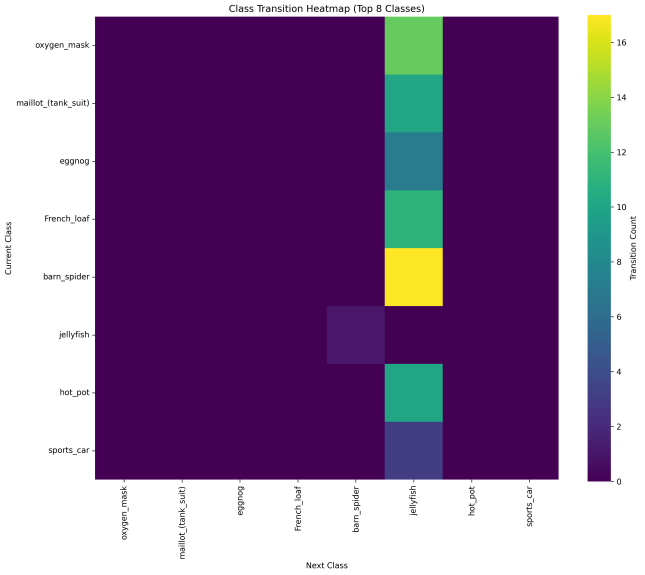

Fig. 6. Heatmap of the top 8 most frequent class transitions after adding noise from a standard Gaussian distribution.

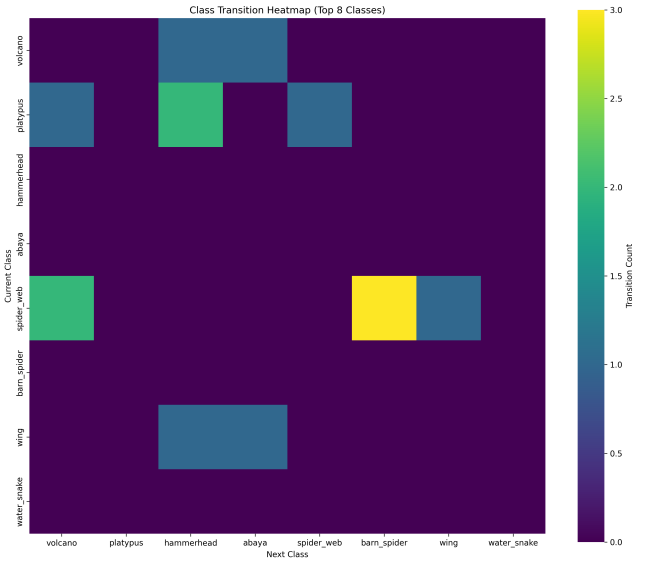

Fig. 7. Heatmap of the top 8 most frequent class transitions after adding noise from model A.

In addition, they can easily overfit to features present in all images that coincidentally have one class label. For instance, if all pictures of cars used to train the model were taken in the day, it may learn that bright outdoor lighting implies a car, and shading this lighting can change the class.

Another potential reason is that the hyperparameters for the loss terms on the decoder network were too small and brought the actor loss to a local minima. Alternatively, these losses were low-pass filters, as opposed to high-pass filters. For instance, the resulting action is very smooth. While this property helps preserve visual quality, it stops the high-pass portion of structured sparsity, which actually changes the label.

Instead, the low-pass changes need to accumulate before the label actually changes, as shown by the need for between 26 and 28 steps per episode. In contrast, since the random noise had sharp pixel perturbations, it was able to change the class very easily.

The combinations producing models D and E were our attempts to use saliency maps, high-frequency filters, and orthogonality terms between the clean and perturbed images to address these issues. However, the saliency maps in 5 appear to focus on parts of the image that visually have a low correlation with the predicted class. For the first image, the area to be changed is part of the shirt, not the sombrero, and the map highlights the background for the second image. As a result, the noise is essentially random, highlighting that more hyperparameter and loss function tuning for the decoder and saliency map is still necessary. These attempts also show that image reconstruction is extremely finicky to get working as desired.

Using the random noise generator, the most frequently applied change was to "jellyfish". This result makes sense for classes like "barn_spider", because a spider web could look similar to jellyfish tentacles, but is surprising for classes like "French_loaf". This may suggest an inductive bias that jellyfish images tend to be in very dark lighting, or an unexplainable decision boundary between jellyfish and these other classes. Moreover, using the learned noise generator from model A, the transition spread is more diverse, ranging from expected results like "spider_web" to "barn_spider" to less expected ones like "volcano" to "abaya". Interestingly, in both cases, there is minimal symmetry (class A perturbing into class B, and class B perturbing into class A). While this could be an artifact of random sampling, it can display how close various images of the same class are to different labels in feature space.

VI. Future Work

There are many extensions of this work, beyond appropriately tuning the image re-constructor components, as follows:

- We used weights from a pretrained ResNet classifier (without the classification head) to form a latent embedding from image space. This formed a static embedding, but the outputs are not necessarily the best for our desired downstream task. Instead, contrastive learning algorithms like Contrastive Unsupervised Representations for Reinforcement Learning [18] and Momentum Contrast for Unsupervised Visual Representation Learning [19] learn to keep embeddings for similar images close in latent space, and different images far from each other. The weights would be learnable parameters, optimized for performing adversarial attacks. Similarly, in the noise reconstruction, we use a pre-trained LPIPS model, which could be optimized for this purpose.
- More generally, we made assumptions in the the problem formulation to solve it as an MDP. All latent embeddings invariably fail to fully represent all information about the given state, which generalizes the proxy formulation to a POMDP. In addition, this work used a time-variant reward function, but applied techniques that assumed time-invariance. Using techniques for formulations that relax either or both assumptions can improve the results. Similarly, we assumed that our differentiable clamp function (20) perfectly matches (12). The error of this operation can be reduced by constructing an interpolating polynomial of (12), whether it be analytical or through a small neural network.
- This work focused on untargeted attacks for image classification. This technique can be extended to targeted attacks on image classification and other similar tasks, such as object detection and segmentation. This would help improve the explainability of such networks by exploring the learned decision boundaries.
- One major focus of query-based adversarial attacks is the necessary number of times the victim model runs inference on an image during training. While this work used SAC, requiring a large replay buffer to warm-start training, on-policy methods like Proximal Policy Optimization [20] and Group Relative Policy Optimization [21] can immediately start training, reducing the number of queries. Similarly, some initialization techniques for neural network parameters may start closer to the global minima than others, such as Kaiming initialization [22] or Xavier initialization [23].

References

[1] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 506–519. [Online]. Available: https://doi.org/10.1145/3052973.3053009

[2] A. J. Bose and P. Aarabi, "Adversarial attacks on face detectors using neural net based constrained optimization," *CoRR*, vol. abs/1805.12302, 2018. [Online]. Available: http://arxiv.org/abs/1805.12302

[3] G. Jocher and J. Qiu, "Ultralytics yolo11," 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014. [Online]. Available: http://arxiv.org/abs/1409.0575

[6] C. Li, H. Wang, W. Yao, and T. Jiang, "Adversarial attacks in computer vision: a survey," *Journal of Membrane Computing*, vol. 6, no. 2, p. 130–147, Apr. 2024. [Online]. Available: http://dx.doi.org/10.1007/s41965-024-00142-3

[7] L. Engstrom, D. Tsipras, L. Schmidt, and A. Madry, "A rotation and a translation suffice: Fooling cnns with simple transformations," *CoRR*, vol. abs/1712.02779, 2017. [Online]. Available: http://arxiv.org/abs/1712.02779

[8] A. S. Shamsabadi, R. Sanchez-Matilla, and A. Cavallaro, "Colorfool: Semantic adversarial colorization," *CoRR*, vol. abs/1911.10891, 2019. [Online]. Available: http://arxiv.org/abs/1911.10891

[9] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[10] J. Pomponi, S. Scardapane, and A. Uncini, "Pixle: a fast and effective black-box attack based on rearranging pixels," *CoRR*, vol. abs/2202.02236, 2022. [Online]. Available: https://arxiv.org/abs/2202.02236

[11] M. Jere, B. Hitaj, G. F. Ciocarlie, and F. Koushanfar, "Scratch that! an evolution-based adversarial attack against neural networks," *CoRR*, vol. abs/1912.02316, 2019. [Online]. Available: http://arxiv.org/abs/1912.02316

[12] C. Yang, A. Kortylewski, C. Xie, Y. Cao, and A. L. Yuille, "Patchattack: A black-box texture-based attack with reinforcement learning," *CoRR*, vol. abs/2004.05682, 2020. [Online]. Available: https://arxiv.org/abs/2004.05682

[13] S. Sarkar, S. Mousavi, A. R. Babu, V. Gundecha, S. Ghorbanpour, and A. K. Shmakov, "Measuring robustness with black-box adversarial attack using reinforcement learning," in *NeurIPS ML Safety Workshop*, 2022.

[14] D. Song, D. Ko, and J. H. Jung, "Amnesia as a catalyst for enhancing black box pixel attacks in image classification and object detection," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 87 159–87 183. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/9e770fcdb456400325c11d58b3a04d08-Paper-Conference.pdf

[15] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. Pmlr, 2018, pp. 1861–1870.

[16] Collet, Yann and Skibinski Przemyslaw, "Zstandard." [Online]. Available: https://github.com/facebook/zstd

[17] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018.

[18] A. Srinivas, M. Laskin, and P. Abbeel, "CURL: contrastive unsupervised representations for reinforcement learning," *CoRR*, vol. abs/2004.04136, 2020. [Online]. Available: https://arxiv.org/abs/2004.04136

[19] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick, "Momentum contrast for unsupervised visual representation learning," *CoRR*, vol. abs/1911.05722, 2019. [Online]. Available: http://arxiv.org/abs/1911.05722

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[21] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo, "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," 2024. [Online]. Available: https://arxiv.org/abs/2402.03300

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, vol. abs/1502.01852, 2015. [Online]. Available: http://arxiv.org/abs/1502.01852

[23] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html