OMG-ATTACK: Self-Supervised On-Manifold Generation of Transferable Evasion Attacks

Ofir Bar Tal
Tel Aviv University
ofirbartal@mail.tau.ac.il

Adi Haviv
Tel Aviv University
adi.haviv@cs.tau.ac.il

Amit H. Bermano
Tel Aviv University
amit.bermano@gmail.com

Abstract

Evasion Attacks (EA) are used to test the robustness of trained neural networks by distorting input data to misguide the model into incorrect classifications. Creating these attacks is a challenging task, especially with the ever increasing complexity of models and datasets. In this work, we introduce a self-supervised, computationally economical method for generating adversarial examples, designed for the unseen black-box setting. Adapting techniques from representation learning, our method generates on-manifold EAs that are encouraged to resemble the data distribution. These attacks are comparable in effectiveness compared to the state-of-the-art when attacking the model trained on, but are significantly more effective when attacking unseen models, as the attacks are more related to the data rather than the model itself. Our experiments consistently demonstrate the method is effective across various models, unseen data categories, and even defended models, suggesting a significant role for on-manifold EAs when targeting unseen models.

1. Introduction

Evasion attacks are a fundamental tool for model robustness validation [13, 14, 54]. As neural networks become increasingly ubiquitous, analyzing trained models for weaknesses by finding EAs becomes more and more crucial. Despite their value, EAs are still disregarded during typical model development. This is because EAs are challenging to generate, especially as models become larger [9, 72, 28] and datasets more elaborate [41, 66, 44].

In terms of computational load, some approaches use optimization to generate individual attacks, which is infeasible for large-scale training [6, 49]. Methods that produce EAs more efficiently are typically white-box based, where the attack generator leverages full knowledge and access to the target model, including its architecture, parameters, weights, and gradients [26, 47]. Again, for real-life scenar-
At the need to access the target model altogether. To realize the method, our first insight is to turn to representation learning and data augmentations. In the representation learning setting, data points are embedded as spread away from each other as possible, regardless of labeling, while being agnostic to predetermined or learned augmentations [3, 56]. These augmentations can be considered evasion attacks generated to strengthen the representation during training. For example, the ViewMaker framework, an efficient, simple-to-implement, self-supervised generative approach for enhancing a representation learning process through automatic augmentation, was proposed by Tamkin et al. [63]. In essence, the method predicts a perturbation of fixed strength that is most likely to change the representation of the input image. Using this framework, instead of learning a representation encoder, we learn augmentations that are designed to attack our pretrained target model. This results in a self-supervised, efficient, EAs generator.

Next, in order to promote transferability, our second insight is on-manifold generation. Intuitively, since on-manifold attacks are closer to the source distribution, they are attacks that are conceptually, affected more by the data distribution itself, rather than the target model and its parameters. Hence, such attacks are effective for any model that is trained on the same data. In OMG-ATTACK, we encourage the EAs to remain on the data manifold, producing attacks that are closer to the data distribution and, in practice, seem more natural, as shown in Figure 2. We do this by adding a discriminator to the aforementioned ViewMaker framework [63]. This forces our attack generator to produce images that are indistinguishable from the source distribution in the eyes of the discriminator, hence producing on-manifold EAs. Indeed, we empirically show that on-manifold examples are transferable to unseen models, and even to new classes within the same dataset, that the OMG-ATTACK model has not seen during training.

Through a series of experiments, we show that on-manifold attacks (Figure 1) are highly transferable, demonstrating robustness to cross-model evaluation on several datasets. Compared to established baselines in off- and on-manifold EAs generation, we especially show promising attack success rates in the black-box settings on unseen defended and un-defended models. Additionally, we show superior performance for classes unseen during training for all of the evaluated data.

Overall, these experiments show that on-manifold EAs are highly transferable and, in conjunction with a self-supervised training process, offer a viable solution for real-life use cases.

2. Related Work

Adversarial Examples Early adversarial example generation techniques, such as L-BFGS [62], FGSM [26], and BIM & ICML [39] attacks, primarily focused on a white-box context, where the attacker has full model insight including architecture, parameters, gradients, and training data. This led to the development of gradient-based methods, a current research focus.

Prominent white-box methods include FSGM [26], its iterative variants [19, 69], and I-FGSM by Madry et al. [47], using the Projected Gradient Decent (PGD) method. While most attacks constrained their $L_2$ or $L_{\infty}$ norms to maintain clean image visuals, others [51, 60] opted to generate perturbations via the network’s forward gradient and saliency map. Our methodology aligns closely with the semi-white-box or grey-box attacks [71, 74, 67, 68]. In these methods, a generative model that has been trained (e.g., [23]) is used to construct adversarial perturbations. This introduces an innovative approach where full access to the model is only required during training. During the inference phase, there’s no need for access to the target model since the generator is already trained. In contrast to earlier works, our model leverages self-supervision in a realistic setting and ensures that the perturbations follow an approximated manifold.

Black-box attacks present the most challenging scenario. These are further divided into score-based settings, where the attacker has access to the prediction distribution or confidence scores [35, 45], and more stringent decision-based settings, where the attack model solely depends on the prediction’s label [64, 5, 17, 17]. Query-based attacks are dominant in decision-based approaches. They estimate the gradient using zeroth-order optimization methods when the loss values can be queried, as proposed by Chen et al. [11].

Although our work can be applied in a black-box setting during testing, it adheres to a stricter real-world setting where the attacker is decision-based and does not incorporate additional parameters or training steps, as in the subtitle transfer model approaches.
On-Manifold Adversarial Examples  The topic of generative adversarial examples has seen considerable interest [10, 1, 73]. Of particular novelty is the generation of adversarial examples that adhere to the data manifold, a contrast to traditional adversarial attacks known to deviate from it [57]. Gilmer et al. [22] provided initial evidence of the existence of on-manifold adversarial examples, and subsequent research has underscored the theoretical advantages of these examples in enhancing model robustness [53, 61].

Goodfellow et al. [25] suggested that optimally balancing the generator’s predictive capabilities and the discriminator’s enforcement of on-manifold constraints requires the generator to mimic the original data distribution. Inspired by this, Li et al. [40] proposed manifold projection methods using generative models. Stutz et al. [59] introduced the concept of per-class on-manifold adversarial examples, though its application is limited in the absence of class labels. Several studies have leveraged generative models, such as Generative Adversarial Networks (GAN) [23] and Variational Auto-Encoders (VAE) [70], to generate on-manifold examples by introducing adversarial alterations to the embedding space [48, 52, 42]. Others, including AdvGan [71] and AdvGAN++ [48], ensure perturbations remain on-manifold by imposing constraints, while some approaches rely on predefined semantic attributes [35].

Our method also uses a GAN-like structure akin to AdvGAN but differentiates itself through the application of the more modern viewmaker backbone, yielding better empirical results (§4.2). Finally, IDAA [75] underscored the potential of on-manifold adversarial examples in achieving superior performance. In our research, we corroborate these findings through comprehensive evaluations, demonstrating that adherence to the data manifold can significantly boost the transferability of models to unseen models without necessitating additional training.

Self-supervised Adversarial Attacks  Self-supervised learning, a methodology that involves training a model without predefined labels, has demonstrated significant potential across various computer vision tasks [21, 50, 12, 29]. There are several paradigms that have been adopted in the realm of self-supervised learning. Instance Discrimination, as exemplified in DINO [9], BYOL [27], and SimCLR [37], makes use of common image processing augmentations to generate varied renditions of the same image. The objective is for models to create comparable representations or activations for these diverse versions. Masked Prediction has found widespread use across fields such as Natural Language Processing [16, 43], Computer Vision [2], and more. Transformation Prediction is another paradigm centered around predicting image rotations [21]. Lastly, Clustering methods have also been utilized in this space [7].

The study of adversarial examples in representation learning has led to intriguing findings, with recent research indicating that the generation of adversarial examples can bolster the performance of self-supervised representation learning frameworks. The work demonstrated in CLAE [32], for instance, established that generating adversarial examples could improve the representation learning process. Similarly, the authors of the ViewMaker network [63] proposed that adversarial examples could be effectively used as a substitute for predefined expert augmentations.

In our study, we rely on and adapt the foundational model proposed in [63] to facilitate the generation of self-supervised adversarial attacks.

3. The OMG-ATTACK Model

We now introduce our method of generating on-manifold non-targeted adversarial examples. In Figure 3 we show our pipeline for learning and generating adversarial augmentations that are on-manifold examples.

3.1. Problem Definition

Let $X \subset I^{n \times n}$ be the image dataset of dimensions $n \times n$. Suppose that $(x_i, y_i)$ is the $i$-th instance within the training set, which is comprised of an image $x_i \in X$, generated according to some unknown distribution $x_i \sim P_{data}$, and $y_i \in Y$ the corresponding true class labels. The learning system aims to learn a classifier $f : X \rightarrow Y$ from the domain $X$ to the set of classification outputs $Y$, where $|Y|$ denotes the number of possible classification outputs. Given an instance $x \in X$, the goal of a regular adversary is to generate adversarial example $x_A$, which is misclassified $f(x_A) \neq y$ (i.e. untargeted attack).

In addition, the definition of a Manifold $M_y \subseteq I^{n \times n}$ is

$$M_y = \{ x \in I^{n \times n} : C_y(x) = \text{True} \}$$

where $C_y$ is the perfect binary classifier that decides if an image is of class $y$ or not. Thus $M := \bigcup_y M_y$ is the union of all classes manifolds in the dataset.

Therefore we further augment the regular adversary definition to include the on-manifold constraint. We aim to generate an adversarial example $x_A$, which is misclassified $f(x_A) \neq y$ and if part of the dataset manifold $x_A \in M$.

3.2. Generator

We use a dense prediction network as a generative model, inspired by the viewmaker networks [63]. The architecture of the model consists of an encoder-decoder structure, with residual blocks that incorporate random noise in the form of extra channels.

Given an input image $x \in X$, the generator $G$ produces a perturbation $G(x, \eta)$ based on the input and a random noise vector $\eta \sim \mathcal{N}(0, 1)$. The resulting perturbation is of the same dimensions and shape as the input image.
Discriminator's architecture is built upon the principles of learning the data manifold. Our original view maker framework.

### 3.3. Discriminator

The discriminator, within our framework, is tasked with the crucial responsibility of learning the data manifold. Our discriminator's architecture is built upon the principles of the PatchGAN discriminator [33], which primarily penalizes structural inconsistencies at the scale of local image patches. It aims to classify the authenticity of each patch in an image, categorizing them as either 'real' or 'fake'.

The PatchGAN discriminator operates convolutionally across the image, averaging all responses from individual patches to generate its final output. This approach essentially models the image as a Markov Random Field, assuming independence between pixels separated by more than a patch diameter.

Let $I$ be an input image and $D$ be our PatchGAN discriminator. The output from $D$ is a score matrix $D(I) \in \mathbb{R}^{n \times n}$, where each element $D(I)_{i,j}$ corresponds to the score attributed to a patch of size $p \times p$ centered at pixel $(i,j)$ in the input image $I$. The binary cross entropy loss (BCE) is utilized for each patch prediction:

$$BCE(p, y) = y \log(p) + (1 - y) \log(1 - p)$$

Penalize if a fake image is considered real, and vice versa ("real" label is 1):

$$l_{real} = BCE(D(x_{fake}), 1), l_{fake} = BCE(D(x_{real}), 0)$$

The final discriminator loss is the average of all patch losses:

$$L_{Disc} = \frac{1}{2} \left( \frac{1}{n \times n} \sum l_{real} + \frac{1}{n \times n} \sum l_{fake} \right)$$

### 3.4. Target Model

Throughout the learning and training processes, our methodology retains full access to the target model. Notwithstanding, our distinctively crafted generator has the capacity to execute an attack during the inference stage, irrespective of the absence of the target model’s loss. This aligns with the "semi-whitebox" setting as detailed in [71]. Opting not to utilize the logits from the target model, we...
leverage its penultimate representations instead. This strategic choice could feasibly extend the applicability of our framework to tasks beyond classification.

3.5. Training via Self-Supervision

Our pipeline uniquely operates on the learned representations rather than the classification layer, thereby obviating the need for labeled datasets. The training process for our pipeline is entirely self-supervised, leveraging two independent objectives to cultivate the generator’s ability to manufacture adversarial, yet on-manifold, perturbations.

**Contrastive Loss** Instead of meddling with the predictions of the target model, our focus is on perturbing the learned representations. To accomplish this, we adopt a contrastive loss framework, a common technique in self-supervised and representation learning [37, 8].

The objective here is to construct analogous representations for augmentations derived from the same source. Each input is associated with a positive example, while all other data points in the batch are regarded as negative examples. The loss is assembled based on the data points’ similarity, with the optimal loss being obtained when the positive pair’s similarity is 1 and the similarity with all other inputs is 0.

The contrastive loss, for \( N \) groups of mutually exclusive positive pairs, is expressed as shown in the following:

\[
L_{Pairs} = \frac{1}{2N} \sum_{k=1}^{N} l(2k-1, 2k) + l(2k, 2k-1)
\]

\[
l(i, j) = -\log \frac{\exp (s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp (s_{i,k}/\tau)}
\]

and \( s_{a,b} \) is the cosine similarity of the representations of examples \( a \) and \( b \), and \( \tau \) is a temperature parameter.

For all inputs in the batch, we use the generator to produce adversarial examples \( X_{adv} = X + G(X, \eta) \). Examples generated from the same image are called positives, while all other examples in the batch (originals and generated) are considered to be negatives.

We compute this contrastive loss for three groups of positive pairs permutations: \((X, X_{adv1})\), \((X, X_{adv2})\), and \((X_{adv1}, X_{adv2})\). This results in three losses:

\[
L_{Contrastive} = L_{Pairs}(X, X_{adv1}) + L_{Pairs}(X, X_{adv2}) + L_{Pairs}(X_{adv1}, X_{adv2})
\]

Applying \( L_{Pairs} \) between the two generated examples \((X_{adv1}, X_{adv2})\) encourages the model to learn to produce diverse augmentations and avoid settling on a deterministic specific perturbation per input.

**On-Manifold Loss** Our approach for generating on-manifold, in-distribution augmentations involves incorporating an auxiliary loss function that leverages a discriminator model. Previous studies indicate that discriminators can efficiently learn the underlying manifold of a training dataset [24, 34].

To integrate this into our methodology, we introduce an on-manifold loss as an adversarial objective for our generative network. The generative network aspires to generate augmentations that the discriminator deems part of the original dataset. As the discriminator is trained, it learns from its misclassifications and continually refines its comprehension of the dataset’s underlying manifold.

Therefore, the on-manifold loss is derived from the discriminator’s loss. It marks the areas where the generator needs to improve upon the fake patches detected by the discriminator: \( L_{On-Manifold} = BCE(D(x_{fake}), 0) \).

**Total Loss** The overall loss employed to train the generator is a combination of both the adversarial iteration of the contrastive loss and the on-manifold loss, which is the adversarial iteration of the discriminator objective: \( L_{total} = \alpha L_{On-Manifold} - \beta L_{Contrastive} \).

Detailed information on the models’ architectures and hyperparameters utilized in our study is provided in the supplementary material.

4. Experiments

The goal of our experiments is to assess the efficacy of our model in generating adversarial examples. To that end, we evaluate our method by conducting experiments in various settings, including the white-box setting, transferability to black-box models and architectures, including defended target models, and transferability of attacks to unseen data classes at train time.

4.1. Experimental Setup

**Datasets & Models** We evaluate our pipeline on three datasets from different domains, using target models of various scales and architectures for the classification task. The datasets used are presented in Table 1.

For each dataset, we train OMG-ATTACK (comprising a discriminator and generator) against a pre-trained model on the classification task of that particular dataset. For the MNIST dataset, we use the same target network architecture based on a Convolutional Neural Network (CNN) as used by Carlini & Wagner [6] (denoted as MNIST-CNN), which has 1.2M parameters. For the German Traffic Sign Recognition Benchmark (GTSRB), we use a CNN with three Spatial Transformers [20], which achieves over 99% accuracy on this dataset (denoted as TS-CNN-STN) with 855K parameters. Lastly, for the Caltech Birds (CUB-200)
dataset, we use the popular ResNet-18 model [30] with 11.3M parameters (denoted as CUB-ResNet18).

As these models were used in training, we evaluate attacks on them under white-box settings, where we have full access to the model parameters.

Under the black-box settings, we use additional models to examine transferability. We leverage the OMG-ATTACK models trained under the white-box setting on each dataset separately and generate attacks for other unseen models. For MNIST, we use two models proposed by [65] (denoted as MNIST-TR1 and MNIST-TR2) as well as a larger-scale ResNet-18 trained on MNIST. For GTSRB, we validate against a larger-scale ResNet-50. Lastly, for the CUB-200 dataset, we evaluate against a larger-scale ResNet-50 and WideResNet-50.

Further details about the models, including the hyperparameters used, are provided in the supplementary material.

**Baselines** To comprehensively evaluate the effectiveness of our proposed model, we draw comparisons with an array of adversarial attack methods that span from simple, yet effective strategies to more complex and robust methods. This range of baselines allows us to benchmark our model against both traditional and state-of-the-art techniques.

1. **Fast Gradient Sign Method (FGSM) [26]:** The method utilizes the gradients of the loss function with respect to the input data to create adversarial perturbations.

2. **Projected Gradient Descent (PGD) [46]:** Building upon FGSM, PGD introduces iterative attacks to adversarial research. Unlike FGSM, PGD applies the perturbations multiple times within an epsilon ball, offering more potent adversarial examples with enhanced robustness.

3. **Momentum Iterative Fast Gradient Sign Method (MI-FGSM) [18]:** This advanced adversarial attack method infuses momentum into the iterative FGSM process. The momentum term allows the method to maintain the direction of gradient update across iterations, thereby enhancing both the effectiveness and robustness of the generated adversarial examples.

4. **Variance-Tuning Momentum Iterative Fast Gradient Sign Method (VMI-FGSM) [69]:** As an extension to MI-FGSM, VMI-FGSM introduces a variance term to dynamically adjust the step size throughout the iterative process. This adaptive approach yields highly potent examples, thus placing VMI-FGSM among the most powerful adversarial attack methods to date.

5. **AdvGAN [71]:** A generative adversarial network-based method for creating adversarial examples. It employs a similar domain of operation as our proposed model, thus providing a pertinent comparison.

**Evaluation Metrics** We use the Attack Success Rate (ASR) to evaluate the effectiveness of our adversarial augmentations. ASR is defined as the fraction of adversarial examples that successfully cause the target model to change its prediction. Formally,

\[
ASR = \frac{|\{X_{adv} | f(X) \neq f(X_{adv})\}|}{|\{X_{adv}\}|}
\]

where \(X_{adv}\) is an adversarial example generated from \(X\). ASR is typically expressed as a percentage, with higher values indicating a more effective attack.

**Budget Selection** Assuring comparability of the budgets across different methods is challenging, given the disparate \(L_p\) norms utilized by these methods. Thus, we conducted an examination where we studied the influence of augmentations under identical budgets across various methods. For each dataset, we used the same budget across all methods evaluated: 0.3 for the MNIST dataset, 0.015 for the GTSRB dataset, and 0.025 for the CUB-200 dataset.

For the selected budget values, we found that the difference in the intensity of the augmentations (i.e., \(P = 1/(mn) \sum (I(x, y))^2\) for each pixel \((x, y)\) in the image \(I\)) is relatively small, less than 10% among the methods. We discuss this challenge further in §6, and additional information on the budget analysis and selection process can be found in §5.

**Base-line White-box Setting** We compare the ASR performance of the trained models in the white-box settings against the different baselines and present the results in Table 2. As can be seen, the performance of our method is comparable to other approaches in this setting.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Resolution</th>
<th>#Classes</th>
<th>Data Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>28x28</td>
<td>10</td>
<td>Digits</td>
<td>60K</td>
</tr>
<tr>
<td>GTSRB</td>
<td>32x32</td>
<td>43</td>
<td>Traffic</td>
<td>40K</td>
</tr>
<tr>
<td>CUB-200</td>
<td>224x224</td>
<td>200</td>
<td>Birds</td>
<td>12K</td>
</tr>
</tbody>
</table>

Table 1. An overview of the datasets used in our experiments. MNIST [15] is a dataset of handwritten digits. GTSRB (German Traffic Sign Recognition Benchmark) [58] is a dataset consisting of images of traffic signs. CUB-200 [31] is the Caltech Birds dataset that includes images of 200 different bird species.
Table 2. White-box setting comparison. The Attack Success Rates (ASR) is reported, where all models are allocated a similar resource limit (i.e., budget). Without transferability, performance is comparable between models.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>PGD</th>
<th>FGSM</th>
<th>MI-FGSM</th>
<th>VMI-FGSM</th>
<th>AdvGAN</th>
<th>OMG-ATTACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>MNIST-CNN</td>
<td>99.87</td>
<td>87.48</td>
<td>99.86</td>
<td>99.87</td>
<td>89.27</td>
<td>99.37</td>
</tr>
<tr>
<td>GTSRB</td>
<td>TS-CNN-STN</td>
<td>33.53</td>
<td>19.71</td>
<td>33.83</td>
<td>33.91</td>
<td>20.25</td>
<td>67.48</td>
</tr>
<tr>
<td>CUB-200</td>
<td>CUB-ResNet18</td>
<td>94.27</td>
<td>92.85</td>
<td>93.65</td>
<td>93.89</td>
<td>75.09</td>
<td>46.97</td>
</tr>
</tbody>
</table>

Table 3. Transferability to defended models. The Success Rates (ASR) is reported for attacks generated using undefended versions on models, but test on the models defended using adversarial training with FGSM.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Def. Model</th>
<th>PGD</th>
<th>FGSM</th>
<th>MI-FGSM</th>
<th>VMI-FGSM</th>
<th>AdvGAN</th>
<th>OMG-ATTACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>MNIST-CNN</td>
<td>6.25</td>
<td>0.32</td>
<td>3.62</td>
<td>3.14</td>
<td>8.41</td>
<td>99.33</td>
</tr>
<tr>
<td>GTSRB</td>
<td>TS-CNN-STN</td>
<td>16.75</td>
<td>8.71</td>
<td>17.41</td>
<td>17.65</td>
<td>6.99</td>
<td>64.58</td>
</tr>
<tr>
<td>CUB-200</td>
<td>CUB-ResNet18</td>
<td>12.65</td>
<td>14.58</td>
<td>19.35</td>
<td>25.43</td>
<td>4.75</td>
<td>38.94</td>
</tr>
</tbody>
</table>

Table 4. Transferability to different architectures and scales. The attacks, originally generated for the white-box settings, are tested for their transferability to more robust black-box target models. The presented results are Attack Success Rate (ASR) percentages.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>PGD</th>
<th>FGSM</th>
<th>MI-FGSM</th>
<th>VMI-FGSM</th>
<th>AdvGAN</th>
<th>OMG-ATTACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>MNIST-TR1</td>
<td>76.98</td>
<td>55.84</td>
<td>80.65</td>
<td>84.15</td>
<td>82.64</td>
<td>99.29</td>
</tr>
<tr>
<td></td>
<td>MNIST-TR2</td>
<td>60.59</td>
<td>85.06</td>
<td>88.12</td>
<td>90.27</td>
<td>72.35</td>
<td>99.56</td>
</tr>
<tr>
<td></td>
<td>ResNet-18</td>
<td>70.91</td>
<td>56.43</td>
<td>72.49</td>
<td>73.60</td>
<td>72.32</td>
<td>89.65</td>
</tr>
<tr>
<td>GTSRB</td>
<td>ResNet-50</td>
<td>10.74</td>
<td>8.70</td>
<td>11.65</td>
<td>11.76</td>
<td>9.01</td>
<td>58.89</td>
</tr>
<tr>
<td>CUB-200</td>
<td>ResNet-50</td>
<td>31.12</td>
<td>37.90</td>
<td>45.84</td>
<td>54.20</td>
<td>17.63</td>
<td>48.33</td>
</tr>
<tr>
<td></td>
<td>WideResNet-50</td>
<td>25.14</td>
<td>33.42</td>
<td>38.67</td>
<td>46.02</td>
<td>19.93</td>
<td>46.84</td>
</tr>
</tbody>
</table>

4.2. Transferability to Unseen Models

These experiments have been conducted in a black-box setting, where the target model’s parameters, logits, and architecture are inaccessible.

**Defended Models** These subsequent experiments aimed to evaluate the robustness of the adversarial examples (originally generated for the undefended model) in manipulating the predictions of a defended model. The defensive strategy deployed in these experiments involved training the original model using a mixture of original data and adversarial examples produced by the FGSM method on the static undefended model. Our approach outperformed other strategies in all the scenarios, demonstrating that the defensive method had a substantially reduced impact on our adversarial examples. The results are presented in Table 3.

**Transferability to Architectures and Greater Scales** The black-box transfer experiment was designed to examine the efficacy of our generated adversarial examples when tested against unseen and more sophisticated models. The goal was to ascertain whether these adversarial examples specifically undermine the unique target model they were trained on or whether they target challenging, under-represented areas in the data manifold that pose inherent classification challenges for any model.

To evaluate this, we devised a simple methodology that applies our data augmentations and assesses the performance of different models, independently trained from the OMG-ATTACK model, on these augmentations. The results are tabulated in Table 4, where we observe that our augmentations achieve an ASR higher than the baseline by 10%-40%. The high ASR values suggest that the tested models encounter difficulties in accurately classifying the augmented data and do not significantly outperform the original white-box target model. This suggests that the adversarial examples are generalizable to unseen models. Notably, all tested models were more sophisticated, possessing a larger parameter space, supporting our hypothesis that the adversarial examples are located in less represented regions of the dataset’s manifold.

4.3. Transferability to Unseen Data

Another key advantage of our system is the ability to generate adversarial augmentations for classes that have not been seen by the target model. To evaluate this capability, we conduct an experiment where the target model is trained on only 80% of the classes in a dataset. Subsequently, OMG-ATTACK is trained model, and utilized to generate augmentations for the entire dataset, including the previously unseen 20%. We can see that our method consis-
**Table 5.** Domain transfer from 80% to 20%. The 'Train' column represents the ASR percentage on the truncated dataset, 'Seen' shows the ASR percentage on the untruncated dataset but only on the subset with seen classes, and 'Unseen' is the ASR percentage on the subset with unseen classes.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ADVGAN</th>
<th>OMG-ATTACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>81.03</td>
<td>46.49</td>
</tr>
<tr>
<td>GTSRB</td>
<td>20.55</td>
<td>60.61</td>
</tr>
<tr>
<td>CUB-200</td>
<td>31.11</td>
<td>70.09</td>
</tr>
</tbody>
</table>

The results are reported in Table 5, and depicted in Figure 4.

5. **Budget Analysis**

The budget determines how much of the image can be altered, or how much of the signal must be preserved, and its magnitude greatly influences the model’s performance. The higher the budget, more successful attacks (higher ASR) can be anticipated, but at the cost deviation from the original images and their content.

In the inset figure, we illustrate how varying the budget size influences the attack performance, particularly in terms of the ASR.

Generally, a larger budget tends to produce adversarial examples that are more challenging for the model to classify correctly.

This observation aligns with expectations since a larger budget provides the generator with greater flexibility to drastically alter the original image.

6. **Limitations**

The primary limitation of our study revolves around the issue of budget selection. Our approach deviates from the majority of comparable baselines due to the unique normalization function we employed. While most standard baselines—including but not limited to AdvGAN—leverage the $L_\infty$ norm for budget allocation, our method uniquely utilizes the $L_1$ norm. Despite our concerted efforts to ensure similar budget allocations across methods for fair comparison, we acknowledge that the comparison might not be entirely balanced due to the disparate norms used.

A further limitation is that certain adversarial examples generated by our method, particularly within the MNIST dataset, lead to modifications that are noticeably perceptible to human observers. This challenges their classification as genuine adversarial examples, suggesting that a more rigorous budget constraint might be necessary for these types of datasets. This observation echoes the findings of previous research that has shown perceptible variations in MNIST dataset images, even when the $L_p$ norm of their differences is below thresholds utilized in earlier works [55]. The evidence, therefore, suggests that these changes can still significantly alter human perception of the images.

In conclusion, while our method has shown promising results, these limitations highlight the need for further exploration in the area of budget allocation and the generation of truly imperceptible adversarial examples.

7. **Conclusions**

Adversarial augmentations serve as an insightful method for uncovering weaknesses in machine learning models. By directing adversarial perturbations to adhere to the data manifold, we can design data-dependent attacks that have transferability to various models.

In this study, we proposed a self-supervised framework for learning and creating such attacks. Our approach draws upon adversarial training, guiding the generator to produce perturbations that challenge the encoder while also remaining within the bounds of the manifold.

Our experiments illustrate the potential of our method in generating transferable adversarial examples, providing a closer look at the visual semantics these augmentations carry. Our findings highlight the need to account for data-dependent attacks in developing and evaluating machine-learning models. Overall, our work adds to the ongoing discussions about adversarial attacks and their effects on machine-learning models. We hope our insights encourage further research in this field, leading toward machine learning models that are more resilient to adversarial perturbations.
References


3704

[28] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.


