Simulated Adversarial Testing of Face Recognition Models

Nataniel Ruiz
Boston University
nruiz9@bu.edu

Adam Kortylewski
Johns Hopkins University
akortyl1@jhu.edu

Weichao Qiu
Huawei
qiuwch@gmail.com

Cihang Xie
UC Santa Cruz
cixie@ucsc.edu

Sarah Adel Bargal
Boston University
sbargal@bu.edu

Alan Yuille*
Johns Hopkins University
ayuille1@jhu.edu

Stan Sclaroff*
Boston University
sclaroff@bu.edu

Abstract

Most machine learning models are validated and tested on fixed datasets. This can give an incomplete picture of the capabilities and weaknesses of the model. Such weaknesses can be revealed at test time in the real world. The risks involved in such failures can be loss of profits, loss of time or even loss of life in certain critical applications. In order to alleviate this issue, simulators can be controlled in a fine-grained manner using interpretable parameters to explore the semantic image manifold. In this work, we propose a framework for learning how to test machine learning algorithms using simulators in an adversarial manner in order to find weaknesses in the model before deploying it in critical scenarios. We apply this method in a face recognition setup. We show that certain weaknesses of models trained on real data can be discovered using simulated samples. Using our proposed method, we can find adversarial synthetic faces that fool contemporary face recognition models. This demonstrates the fact that these models have weaknesses that are not measured by commonly used validation datasets. We hypothesize that this type of adversarial examples are not isolated, but usually lie in connected spaces in the latent space of the simulator. We present a method to find these adversarial regions as opposed to the typical adversarial points found in the adversarial example literature.

1. Introduction

Evaluating a machine learning model can have many pitfalls. Ideally, we would like to know (1) when the model will fail (2) in which way it will fail and (3) how badly it will fail. In other words, we would like to be able to accurately estimate the model's risk on the true test data distribution as well as know what specific factors induce the model to failure. We would like to know how these failures will manifest themselves. For example, whether a face verification model will generate a false-positive or false-negative error. And finally, when this failure happens, we would like to know how confident was the incorrect decision by the model. Testing models is no longer a purely academic endeavour [60], with many high profile bad societal consequences being revealed in recent years due to insufficient testing particularly with respect to racial and gender bias in face analysis systems [5, 15, 19].

These three desiderata are very hard to achieve in practice. There are major philosophical and theoretical obstacles to achieve perfect knowledge of model failures a priori. Nevertheless, partial knowledge of model weaknesses and predictions of model failures are possible. Yet, there are still major hurdles that stand in our way.

One such hurdle is the fact that *testing data is limited*, due to the fact that it is expensive to gather and label. It is not uncommon for a model to perform well on an assigned test set and fail to generalize to specific obscure examples when it is deployed. A second important hurdle is the fact that *testing data is unruly*. There are latent factors that generate the testing data, which are hard to control or even to fully understand. For example, a known factor that is hard to control is the lighting of a scene. Most datasets have been captured without controlling for this variable, and thus present an insufficient amount of variability in this respect. Testing a model in one environment could yield perfect performance, yet fail on an environment with more lighting variability. Even if a test dataset with carefully controlled lighting were assembled, the dataset would be very expensive and time-consuming to collect and there is no guarantee that the full variability would be explored.

A way to tackle these problems is to use simulators to generate test data. Such an approach can cheaply generate a large quantity of data spanning a large spectrum. Also, sim-

*Equal senior contribution.

ulators are fully controllable and the generative parameters are known. This allows for careful exploration of situations where models fail. This includes the possibility to find intepretable factors that generate failures, to study the way these failures manifest themselves (is the model classifying a cat as a jaguar when there is green in the background?) and to examine the degrees of certainty of the model in these failure modes.

When simulating test data, we have full control over simulator parameters. Thus, we are able to explore the manifold generated by the simulator in the space of the simulator parameters. We call this manifold the *semantic image manifold*, in contrast to the *adversarial image manifold* that is explored in the traditional adversarial attack literature. A random exploration of this manifold is both inefficient and not the most informative approach. In this work we propose to ***test machine learning models using simulation in an adversarial manner*** by finding simulator parameters that generate samples that fool the model. We are inspired by the literature on adversarial examples that fool machine learning models, yet in contrast to this body of work, the adversarial examples that our simulator generates are *semantically realistic* in the sense that we are not adding low magnitude noise to an image in order to fool the model but finding semantically sensible image configurations that generate model failure. In this way, we are not investigating the well-known weakness of gradient-based models to unrealistic targeted noise but to plausible scenes that might be rare, yet mislead the model. We present a method that finds adversarial samples efficiently using a continuous policy that searches the high-dimensional space of possibilities.

A limitation of this type of work is that, in general there exists domain shift between the distribution described by the simulator and the real world distribution [7,14,20,39,54, 55]. Nevertheless, in our work we are able to show that in some situations, real model weaknesses can be found using simulated data. This gives credence to the hypothesis that, even though there is domain shift, simulated samples can be informative. Also, simulators are rapidly improving in terms of realism [11, 30, 36, 48]. This allows for greater opportunities to use these ideas in the future as simulated and real data distributions become more and more aligned.

We hypothesize that these adversarial examples are not isolated points in space, but instead are regions of this manifold. In prior work on traditional adversarial examples, optimization procedures find adversarial samples that are points in image space [6, 18, 33, 37, 49, 53]. In contrast to this body of work we propose a method to find these *adversarial regions* instead. This is valuable because ideally we would like to be able to fully describe the machine learning model's *regions of reliability*, where model predictions will tend to be correct. With this knowledge a user would be able to avoid performing inference on a model outside of its scope in order to minimize failures.

Contributions of this work are three-fold. We summarize them as follows:

- We show that weaknesses of models trained on real data can be discovered using simulated samples. We perform experiments on face recognition networks showing that we can diagnose the weakness of a model trained on biased data.
- We present a method to find adversarial simulated samples in the *semantic image manifold* by finding adversarial simulator parameters that generate such samples. We present experiments on contemporary face recognition networks showing that we can efficiently find faces that are incorrectly recognized by the network.
- We present a method to find *regions* that are adversarial, in order to locate danger zones where a model's predictions are more liable to be incorrect. To the best of our knowledge, we are the first to explore the existence of these adversarial regions in the interpretable latent space of a simulator.

2. A Framework for Simulated Adversarial Testing

Here we formalize adversarial testing using a simulator. We postulate some assumptions on the data generation process in the real and simulator world. Then we give the risks for a machine learning model and the mathematical formulation to find adversarial parameters that yield samples that fool machine learning models. We then present some parallels between our scenario and the literature on learning across domains. Finally, we describe our proposed algorithm to find such adversarial simulator parameters and adversarial samples.

Let us assume the real world data (x, y) (where x is the data and y is the label) is generated by the distribution $p(x, y|\psi)$ where ψ is a latent variable that causally controls the data generation process. For example, ψ includes the object type in the image and the angle of view of such an object, as well as all other parameters that generate the scene and image. The risk for a discriminative model f is:

$$\mathbb{E}_{\psi\sim a}[\mathbb{E}_{(x,y)\sim p(x,y|\psi)}[L(f(x), y)]], \tag{1}$$

where a is the distribution of ψ and L is the loss. We can search for ψ^* that maximizes this risk:

$$\max_{\psi\in A}[\mathbb{E}_{(x,y)\sim p(x,y|\psi)}[L(f(x), y)]] \tag{2}$$

where A is the set of all possible ψ. Let us assume that we have $\psi = (\psi_u, \psi_k)$, a decomposition of ψ into two latent variables ψ_u and ψ_k. Furthermore, let us assume that ψ_u

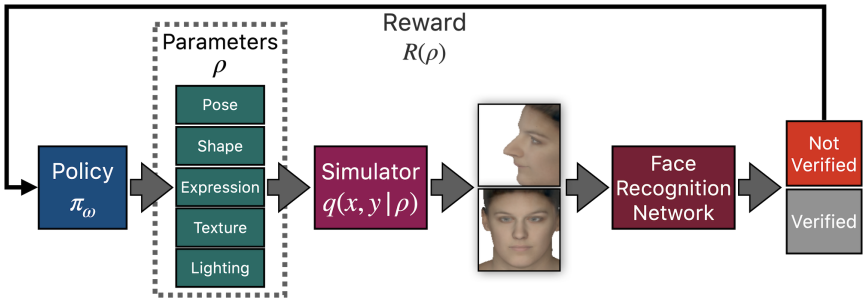

Figure 1. Our method applied to the face verification scenario. The simulator is conditioned on parameters generated by the policy. An image pair of the same identity is generated. Face verification is run on this image pair using the face recognition network that is to be diagnosed. A reward is computed based on the correct or incorrect prediction of the network and policy parameters are updated accordingly.

controls for unknown features of the image, and ψ_k controls for known features of the image such as the camera pose, or the object position with respect to the camera. We can write the average risk as:

$$\mathbb{E}_{\psi_u \sim a}[\mathbb{E}_{\psi_k \sim b}[\mathbb{E}_{(x,y)\sim p(x,y|\psi_u,\psi_k)}[L(f(x), y)]]], \quad (3)$$

where b is the distribution of ψ_k. In most scenarios, we do not have access to the real data distribution p and cannot sample from it at will. Additionally, it is very difficult to control the known latent variable ψ_k when generating data, and we do not even know what factors are hidden in the variable ψ_u, much less how to control it. Using simulated data we are able to fully control the generative process.

A simulator samples data $(x, y) \sim q(x, y|\rho)$, where q is the simulated data distribution and we have complete knowledge over the latent variable ρ. We are able to search for adversarial examples and compute estimates of the mean and worst-case risks using this simulator. For example, the parameter ρ^* that maximizes the risk is written as follows:

$$\max_{\rho \in C} [\mathbb{E}_{(x,y)\sim q(x,y|\rho)}[L(f(x), y)]] \quad (4)$$

where C is the set of all possible ρ. We can find $\hat{\rho^*}$, an estimate of ρ^*, by sampling (albeit inefficiently). In our case we are working in a less restrictive scenario since we do not try to find the global maximum ρ^*, instead we try to find any ρ where $\mathbb{E}_{(x,y)\sim q(x,y|\rho)}[L(f(x), y)]$ is above the misclassification threshold.

If we assume that the distributions p and q are similar enough we can use the knowledge gathered in simulation to understand the possibilities of failure in the real world. Essentially, this is a different kind of domain shift problem. In a traditional setting of transfer learning between domains, we are concerned about minimizing the risk on a target domain by training on a source domain. In the binary classification case, let us define a domain as a pair consisting of a distribution p on inputs $\mathcal{X}$ and a labeling function $g_p : \mathcal{X} \to [0, 1]$. We consider the *real domain* and the *simulated domain* denoted by (p, g_p) and (q, g_q) respectively.

We also introduce a *hypothesis* that is a function $h : \mathcal{X} \to \{0, 1\}$. We can write the risk of this hypothesis on p as:

$$\epsilon_p(h, g_p) = \mathbb{E}_{x\sim p}[|h(x) - g_p(x)|] \quad (5)$$

In traditional domain adaptation from simulation to reality, we seek to learn on distribution q and generalize to distribution p. We want to find a hypothesis that minimizes the risk on the target real world distribution $\epsilon_p(h, g_p)$ by training on samples from q.

In our setting, we do not train on synthetic samples. Instead we want to find a relationship between testing a hypothesis h on samples from distribution q and testing h on samples from p. There exist bound results for the risks $\epsilon_p(h, g_p)$ and $\epsilon_q(h, g_q)$ in the work of Ben-David *et al.* [4]:

$$\epsilon_p(h, g_p) < \epsilon_q(h, g_q) + d_1(q, p) + \min\{\mathbb{E}_p[|g_q(x) - g_p(x)|], \mathbb{E}_q[|g_q(x) - g_p(x)|]\}, \quad (6)$$

where d_1 is the variation divergence. The second term of the right hand side quantifies the difference between distributions q and p, and the third term of the right hand side is the difference between the labeling functions across domains, which is expected to be small.

Since this bound characterizes the cross-domain generalization error and $\epsilon_q(h, g_q)$ will usually be minimized by the learning algorithm, it is useful for studying transfer learning between domains. There are some differences in our scenario since for us h is a fixed function that has been trained on the target domain and we would like to talk about individual examples instead of overall risk over distributions. Also, the bound is proven for a binary classification problem, whereas our target scenario can be multi-class classification or regression.

Assume there exists a mapping $\tau : C \to A$, that maps the simulated latent variables to real latent variables $\psi = \tau(\rho)$. In order for adversarial examples in the simulator domain to be informative in the real domain, we want to

have a simulator such that:

$$\mathbb{P}_{(x_s,y_s)\sim q,(x_r,y_r)\sim p}[|L(x_s,y_s)-L(x_r,y_r)|<\epsilon]>\theta. \quad (7)$$

We denote $p(x_r, y_r|\tau(\rho))$ as p and $q(x_s, y_s|\rho)$ as q in the equation above for succinctness. Here ϵ is small and $\theta \in [0, 1]$ is large. This way, high-loss examples found in the semantic image manifold using simulation have a high probability of transferring to the real world. Since the simulator and real domain are different, this is a moderately strong assumption. Nevertheless, we show cases where this assumption holds in our experimental evaluations in Section 4.3.

Finding Adversarial Parameters Our task is then to find ρ such that the loss over samples generated with this latent variable is above the misclasification threshold T. One main difficulty in searching for latent variables that fulfill this condition is that in general the simulator q is non-differentiable. Thus, we turn to black-box optimization methods to search for adversarial parameters. Specifically, we use a policy gradient method [57].

We define a policy π_ω parameterized by ω that can sample simulator parameters $\rho \sim \pi_\omega(\rho)$. We train this policy to generate simulator parameters that generate samples that obtain high loss when fed to the machine learning model f. For this we define a reward R that is equal to the negative loss L and we want to find the parameters ω that maximize $J(\omega) = \mathbb{E}_{\rho\sim\pi_\omega}[R]$. Following the REINFORCE rule we obtain gradients for updating ω as

$$\nabla_\omega J(\omega) = \mathbb{E}_{\rho\sim\pi_\omega}\left[\nabla_\omega \log(\pi_\omega) R(\rho)\right] . \quad (8)$$

An unbiased, empirical estimate of the above quantity is

$$\mathcal{L}(\omega) = \frac{1}{K}\sum_{k=1}^{K} \nabla_\omega \log(\pi_\omega)\hat{A}_k , \quad (9)$$

where $\hat{A}_k = R(\omega_k) - \beta$ is the advantage estimate, β is a baseline, K is the number of different parameters ρ sampled in one policy forward pass and $R(\rho_k)$ designates the reward obtained by evaluating f on $(x_k, y_k) \sim q(x_k, y_k|\rho_k)$. We show all of the steps of our method in Algorithm 1 and we show an illustration of our method applied to the face verification scenario in Figure 1.

3. Finding Adversarial Regions

Here we describe our method to find adversarial regions. Once an adversarial simulator latent vector $\rho_{\text{adv}} \in \mathbb{R}^n$ have been found using Algorithm 1 we define a graph $G = (V, E)$. V are the vertices of the graph, obtained by discretizing the space around the adversarial point in grid with spacing ν between vertices. The edges E of the graph connect neighboring vectors, with each vector having $2n$ neighbors. We find the connected space of adversarial examples $\mathcal{R}_{\text{adv}}$ that is seeded by ρ_{adv} by following Algorithm 2.

Algorithm 1: Our adversarial testing approach using a policy gradient method.

Result: adversarial simulator parameters ρ_k and adversarial sample x_k
for *iteration=1,2,...* **do**
  Generate K simulator parameters $\rho_k \sim \pi_\omega(\rho_k)$;
  Generate K samples $(x_k, y_k) \sim q(x_k, y_k|\rho_k)$
  Test the discriminative model and obtain K losses $L(f(x_k), y_k)$
  if $\exists k \in \{1, ..., K\}; L(f(x_k), y_k) > T$ **then**
    Terminate and yield adversarial sample x_k and adversarial simulator parameters ρ_k
  end
  Compute rewards $R(\rho_k)$
  Compute the advantage estimate $\hat{A}_k = R(\rho_k) - \beta$
  Update ω via equation 9
end

In essence, our method follows the general idea of an area flooding algorithm [31, 52] with two main differences. First, that we discretize a continuous space that is n-dimensional instead of working on binary 2-dimensional image, and second, that we check for sample membership of $\mathcal{R}_{\text{adv}}$ by testing whether the model loss is higher than the adversarial threshold $L(f(x), y) > T$.

Algorithm 2: Finding connected spaces of adversarial examples.

Result: connected space of adversarial examples $\mathcal{R}_{\text{adv}}$
Data: seed adversarial simulator parameters ρ_{adv}
$\mathcal{R}_{\text{adv}} = \{\rho_{\text{adv}}\}$
Initialize a stack χ.
Push $2n$ neighbors of ρ_{adv} to χ.
for *i=1,2,...* **do**
  Pop ρ_i from χ
  Sample $(x_i, y_i) \sim q(x_i, y_i|\rho_i)$
  Test the discriminative model and obtain loss $L(f(x_k), y_k)$
  if $L(f(x_k), y_k) > T$ **then**
    $\mathcal{R}_{\text{adv}} = \mathcal{R}_{\text{adv}} \cup \{\rho_i\}$
    Push all neighbors of ρ_i that have not been visited to χ
  end
end

4. Experimental Results

4.1. Controllable Face Simulation

We use the FLAME face model [29] as a controllable face simulator with the Basel texture model [38]. FLAME uses a linear shape space trained from 3,800 3D scans of human heads and combines this linear shape space with an articulated jaw, neck, and eyeballs, pose-dependent corrective blendshapes, and additional global expression blendshapes. In this way, using shape and texture components we can generate faces with different identities. The synthetic faces that are generated in our work are new and do not mimic any existing person's features. By changing the pose and expression components we can add variability to these faces. Moreover, we have full control over the scene lighting and the head and camera pose and position. In order to render our scene we use the PyTorch3D rendering framework [41]. We extract the corresponding shape, texture and expression components from the real faces of the CASIA WebFace dataset using DECA [10].

4.2. Models, Datasets and Infrastructure

In our experiments we use the CASIA WebFace [59] dataset for training the face recognition models and the LFW [23] dataset for real-world data testing. We use a Convolutional Block Attention Module (CBAM) [58] ResNet50 with the ArcFace [8] loss as our base face recognition model. We also test our method on MobileNet [21] and CBAM-Squeeze-Excitation-ResNet [22] architectures and the CosFace [56] loss. We use a multivariate Gaussian policy $\pi(\rho) = \mathcal{N}(\mu_\pi, \sigma_\pi^2)$ where the variance is fixed $\sigma_\pi^2 = 0.05 \times I$ and μ_π is learned. For the random optimization baseline we use one Gaussian for each parameter type with standard deviation $\sigma_{\text{rs}} = \frac{w_p}{10} \times I$, where w_p is the width of the parameter domain. For the Gaussian random sampling baseline we use a standard deviation $\sigma_{\text{g}} = \frac{w_p}{2}$. We use a GeForce RTX 2080 GPU with 11GB of memory to perform all of our experiments.

4.3. Testing Weakened Models

We present a way to verify that knowledge from simulated weaknesses translates to real-world weaknesses. We weaken two networks by training on the CASIA WebFace dataset with images that exhibit a yaw parameter $[-\infty, -0.5]$ and $[0.5, +\infty]$ filtered out. We extract the yaw parameter using DECA. We call these the *Negative Yaw Filtered* (NYF) and *Positive Yaw Filtered* (PYF) datasets/networks, respectively. Both datasets have roughly the same number of samples: the *Negative Yaw Filtered* dataset has $\sim$ 440k training samples and the *Positive Yaw Filtered* dataset has $\sim$ 449k samples. We also train a *Normal* network on all of the $\sim$ 491k samples of the unfiltered CASIA WebFace dataset. We then test both the normal network and the yaw-weakened networks on simulated samples. We do this by generating two images of a same person, by fixing the shape, texture and expression parameters. The first image is a frontal image of the person. We vary the yaw component of the second image in the $[-1, 1]$ range, where -1 and 1 in the yaw component indicate a fully-profile face on the negative and positive sides, and compute the cosine similarity between the embeddings of the two images. This cosine similarity should be large given that the two images presented are of the same identity. A low cosine similarity means that the network has less confidence that the images show the same person.

We plot this in Figure 2, and observe that each yaw-weakened network makes less accurate predictions for images presenting high yaw in their respective weakness intervals. Note that all networks perform almost identically with frontal samples. Also, note that the normal network is almost always superior to the two weakened networks. This is a natural result of having 10% more training data. This plot is an average over 25 different identities that we obtain by grid-sampling the first texture and shape components over the range $[-\sigma, \sigma]$.

We compute the area between the curves for the $[-1.0, -0.5]$, $[-0.5, 0.5]$ and $[0.5, 1.0]$ intervals. We observe in Table 1 (left) that in the $[-1.0, -0.5]$ yaw range, precisely where the NYF network has been weakened, the area between the Normal-NYF curves is large and the area between the Normal-PYF curves is small. Conversely, in the $[0.5, 1.0]$ range, where PYF has been weakened, we see that the difference between the Normal-PYF curves is large and the Normal-NYF difference is smaller. Also, we observe near identical differences between Normal-NYF and Normal-PYF in the $[-0.5, 0.5]$, which is a consequence of the lesser amount of training data of NYF and PYF networks. We also compute pairwise mean differences for the different populations of Normal, NYF and PYF networks and present them in Table 1 (right). We highlight in blue the statistically significant differences. We have similar results as in Table 1 (left).

This evidence indicates that when a weakness is purposefully created in a network by filtering out key samples in the real training dataset, we can retrieve this weakness using our face simulator. This gives credence to the idea that we are able to find simulated adversarial examples in the semantic image manifold that will give us knowledge about adversarial examples in the real world.

4.4. Simulated Adversarial Testing of Face Recognition Models

In this section we evaluate adversarial testing of face recognition models for face verification. Specifically, we generate samples using the FLAME face model and use our proposed search algorithm to fool face recognition models.

↓ Models / Yaw Interval →	Area Between Curves [-1.0, -0.5]	[-0.5, 0.5]	[0.5, 1.0]
Normal:NYF	**8.69**	2.83	4.68
Normal:PYF	2.71	2.76	**8.46**

↓ Models / Yaw →	Mean Difference -1.0	0.0	1.0
Normal-NYF	0.18	0.01	0.10
Normal-PYF	0.01	0.00	0.16
NYF-PYF	-0.17	-0.01	0.06

Table 1. Quantitative differences between evaluation of the purposefully weakened *Negative Yaw Filtered* (NYF) and *Positive Yaw Filtered* (PYF) and the *Normal* on synthetic faces (bold values for emphasis). Blue values in the table on the right mean the differences are statistically significant with $p < 0.01$.

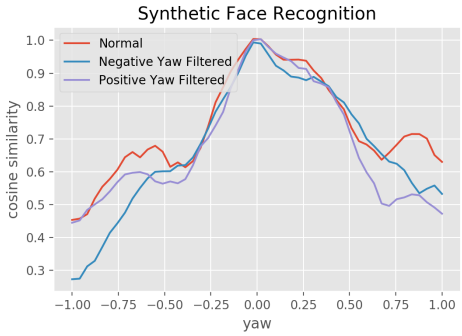

Figure 2. Recognition cosine similarity between two simulated pairs (frontal and variable yaw) of the same identity (avg. over 25 different identities). The *Negative Yaw Filtered* network exhibits less accurate predictions for highly negative yaw images than both the *Positive Yaw Filtered* and *Normal* networks. The *Positive Yaw Filtered* network exhibits less accurate predictions for highly positive yaw images than both other networks.

We train an ArcFace CBAM-ResNet50 on CASIA WebFace for 20 epochs. This network achieves a 99.1% accuracy on the LFW test set for the face verification task. The evaluation task is face verification between two synthetic images of a same person's face, one frontal and one profile image. We vary the first 15 shape parameters as well as the first 15 texture parameters for our generated identities, ranging from -2σ and 2σ where σ is the standard deviation of each parameter in question.

We propose testing the network using 100 identities obtained by random sampling these parameters following a uniform distribution. We also test the network using 100 runs of our adversarial testing algorithm (200 maximum iterations). In Table 2, we show that the random sampling testing regime achieves an accuracy of 99%, which is very close to the 99.1% real-world accuracy of the network on the LFW test set. Using adversarial testing, the network exhibits an accuracy of 36%, which is a marked drop in verification performance. We also compute the average cosine similarity between pairs, showing that adversarial testing generates highly adversarial samples (success threshold $T = 0.298$) whereas random samples are highly non-adversarial on average. In Figure 3 we show a subset of the generated samples for both the adversarial testing (above) and random sampling (below).

We perform further simulated adversarial testing experiments on several combinations of network backbones (CBAM-ResNet50, CBAM-SE-ResNet50 [22], MobileNet) and face recognition losses (ArcFace, CosFace) trained on CASIA WebFace for 20 epochs. All networks achieve accuracies in the (98.85%, 99.1%) range on the LFW test set. We vary 30 shape parameters, 30 texture parameters ranging from -2σ and 2σ where σ is the standard deviation of each parameter. We also vary the yaw pose parameter within $[-1, +1]$, corresponding to variations of $[-\pi/2, +\pi/2]$ degrees and the pitch pose parameter from $[-1/4, +1/4]$ corresponding to variations within $[-\pi/8, +\pi/8]$. Thus, in this case our algorithm has to learn 62 parameters. This is a more challenging scenario due to the larger dimensionality of the policy output.

We perform 100 runs of our adversarial testing algorithm (200 maximum iterations), 100 runs of Random Optimization using a Gaussian sampling distribution and 1,000 iterations of uniform random sampling and Gaussian random sampling. We compare these testing methods in Table 3 and we show that the networks achieve very high accuracies for both random sampling regimes and for testing using random optimization. Using adversarial testing, all networks exhibit a marked drop in verification performance. There is also a large increase in the average cosine similarity between pairs, showing that adversarial testing generates highly adversarial samples (below success thresholds $T = (0.298, 0.237, 0.292, 0.294)$ respectively), whereas other methods generate "easy" samples on average.

Further, for example, for ArcFace CBAM-ResNet50, adversarial testing achieves 51 adversarial samples over 12,587 iterations while random sampling achieves only one adversarial sample over 1,000 iterations. This makes adversarial testing 400% more sample efficient than random sampling in this specific scenario. In some of our tested scenarios and depending on the number of iterations, random sampling was not able to find any adversarial samples. This is reflected by a 100% face verification accuracy. In Figure 4, we show several successful adversarial testing runs (orange/red) and one random sampling run (green). Unsuccessful optimization attempts usually converge to low cosine similarity without becoming adversarial and remain in the high-dimensional local minima. Finally, we show an example of adversarial testing in action where all 30 shape, 30 texture and 2 pose parameters are being learned jointly in Figure 5. The algorithm finds an adversarial sample that

Table 2. CBAM-ResNet50 face verification accuracy over synthetic datasets generated by uniform random sampling or by adversarial testing (Adv. Testing). We vary the identity by varying 15 shape parameters and 15 texture parameters.

Method	Accuracy ↓	Avg. Cosine Similarity ↓
Uniform Random	99%	0.518
Adv. Testing	36%	0.263

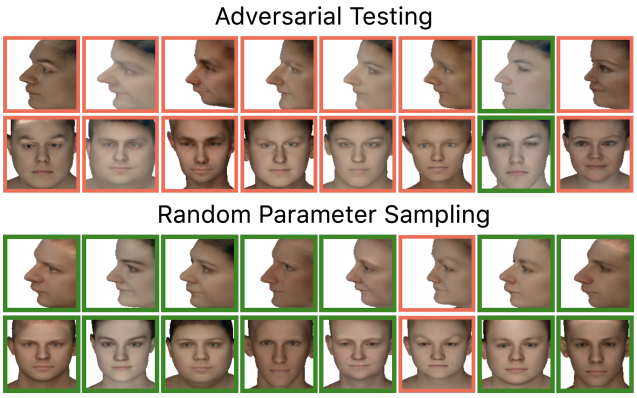

Figure 3. Face models obtained using *adversarial testing* (above) and random parameter sampling (below). A green border denotes pairs that are successfully verified as the same identity, whereas a red border denotes failed verification (model failure). We obtain adversarial samples using our *adversarial testing* method more consistently than with random parameter sampling. Some recurring features of adversarial faces are ambiguous frontal/profile features (e.g. long nose, tucked jaw), pale/dark skin colors and left/right asymmetries.

reveals model weaknesses such as vulnerability to unusual poses, exaggerated facial features and distinct skin color.

4.5. Finding Adversarial Regions of Face Recognition Models

We use our method described in Algorithm 2 to find adversarial regions in the simulator latent space for face recognition models. We do this in the face verification scenario between a frontal image with neutral expression and a profile image with an open jaw. We vary the first shape and texture parameters to find an adversarial sample, and then find the connected spaces to those seed parameters. We also grid sample both parameters in order to plot the synthetic sample surface. We show the surface of all synthetic samples (blue), along with the adversarial region (red) and the adversarial threshold plane (orange) in Figure 6.

We are successful in finding the adversarial regions when they exist. We discover a surprising fact when plotting the synthetic loss landscape (Figure 7) of all the tested networks. In this configuration with only 2 variable parameters, the only network with an adversarial region is ArcFace CBAM-ResNet50. Even though all networks have been

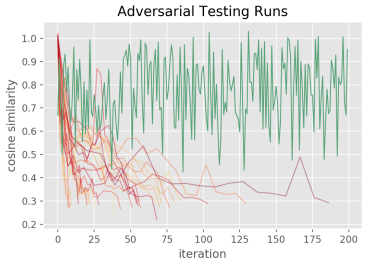

Figure 4. Cosine similarity for successful *adversarial testing* (red) and random parameter sampling (green).

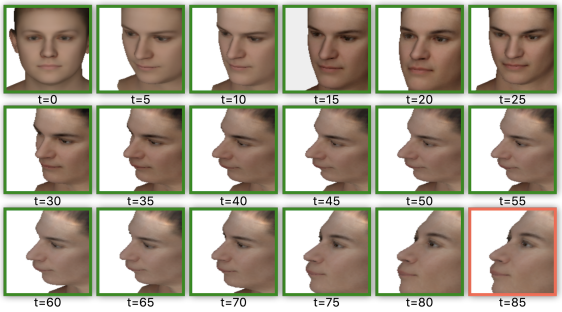

Figure 5. A sequence of generated synthetic samples undergoing *adversarial testing* (left to right, top to bottom). Our method searches through all 30 shape, 30 texture and 2 pose parameters jointly to find an adversarial face. The border line colors denote whether the face recognition network can successfully verify the pairs, with red denoting a failed verification and green denoting a successful verification.

trained in the same manner on the same dataset, the network backbone and the loss function change the loss landscape substantially. Some networks have a similar downward slope from negative shape towards positive shape, but some particularities arise in some. Strikingly, ArcFace MobileNet is the most robust of all the networks in this scenario with a landscape far above the misclassification threshold plane. The landscape shape is also completely different from the other networks.

5. Related Work

Testing computer vision models on synthetic data is not a new idea [24, 26, 27, 35, 40, 48], although there is a relative paucity of work in this area. More common are investigations on training models on synthetic data [9, 12, 16, 17, 28, 34, 42, 43, 46]. Recent works even learn to adapt the generative distribution of synthetic data in order for the model to learn better representations [2, 3, 13, 25, 32, 47] or adapt the pixels or features of the synthetic data to bridge the synthetic-to-real domain gap [7, 14, 20, 39, 54, 55]. In contrast to this body of work, we propose to search the parameter space of a simulator in order to test the model in an adversarial manner. There is very interesting work that adapts

Loss + Backbone	Uniform Random Acc. ↓	Uniform Random Avg. CS ↓	Gaussian Random Acc. ↓	Gaussian Random Avg. CS ↓	Random Opt. Acc. ↓	Random Opt. Avg. CS ↓	Adv. Testing Acc. ↓	Adv. Testing Avg. CS ↓
ArcFace CBAM-ResNet50	99.9%	0.766	99.3%	0.695	93%	0.414	49%	0.282
CosFace CBAM-ResNet50	99.9%	0.696	99.6%	0.637	86%	0.318	57%	0.281
ArcFace SE-CBAM-ResNet50	99.8%	0.738	97.7%	0.663	73%	0.348	34%	0.305
ArcFace MobileNet	100%	0.825	99.8%	0.751	96%	0.454	58%	0.372

Table 3. Comparison of different synthetic sampling techniques on different combinations of network backbones and face recognition losses. We vary 30 shape, 30 texture and 2 pose parameters.

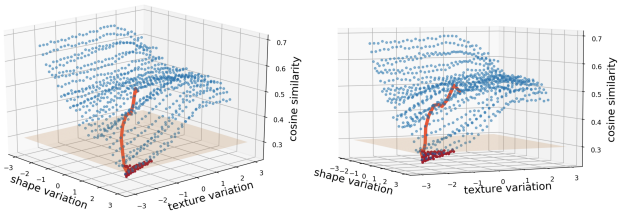

Figure 6. Our algorithm finds the adversarial region (red) in the shape-texture landscape (blue). We plot the initial learning trajectory (lighter red) that yields the seed adversarial simulator parameters. We also plot the adversarial threshold plane (orange).

generative distributions in order to test models [1, 51, 61]. In contrast to [51, 61] we test computer vision models that are trained on real data, which is a more challenging scenario since the domain shift problem has to be described and overcome. Different from [1, 51, 61] we work on the domain of face recognition instead of object classification or VQA, where we have a higher number of simulator parameters including shape, expression, texture, lighting and pose parameters. We search the parameter landscape using a continuous policy that explores all parameters simultaneously, which is important since model performance does not vary independently with each parameter (as Figure 6 shows), and discrete changes in parameter space can yield high loss changes due to gradient sharpness. A final difference with these and work on traditional adversarial attacks [6, 18, 33, 37, 44, 45, 53] is that we present a method that not only finds one isolated adversarial example, but locates regions of them. There exist methods that propose objectives that locate regions of adversarial examples [50]. In contrast, we explore the adversarial regions that lie in the latent space of a simulator instead of pixel space.

6. Conclusion

In this work we propose to test machine learning models by searching for semantically realistic adversarial examples using a simulator. We present a framework for simulated adversarial testing, as well as a method to find simulated adversarial examples. Finally, we present a method to find connected spaces of adversarial examples in the semantic space of latent variables and evaluate our methods on contemporary face recognition networks using a face simulator.

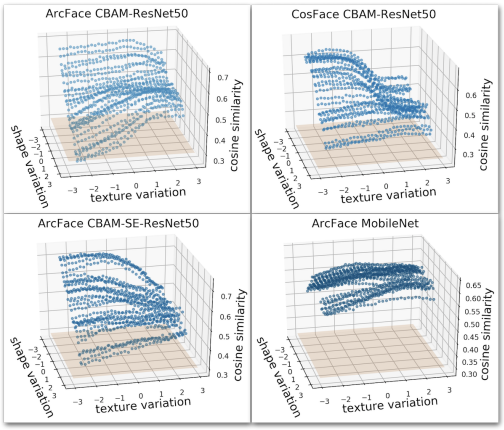

Figure 7. Landscape comparisons for different network backbones and losses. Networks trained on CASIA WebFace.

We find that face recognition networks that have real world weaknesses due to biased training sets with respect to pose can be analyzed using controllable simulated faces and these weaknesses can be discerned. We also find that contemporary face recognition networks are fooled by specific combinations of simulated face shapes and textures. Some recurring features of adversarial faces are ambiguous frontal/profile features (e.g. long nose, tucked jaw), pale/dark skin colors and left/right asymmetries. When such a network is tested using adversarial testing, it's accuracy plummets compared to random testing or testing on a real-world test set such as LFW. We show evidence that these adversarial examples are not isolated, but part of connected spaces of adversarial examples in the manifold of semantically plausible images. We also show that network loss landscapes can vary significantly depending on the network architecture and loss used, even though the training dataset is fixed. Even so, adversarial testing finds adversarial samples for all networks effectively. We will investigate this phenomenon in future work. Finally, we have an in-depth discussion of the limitations and potential negative impact of our work in the supplementary material.

Acknowledgments This work was supported in part by grants ONR N00014-21-1-2812 and NIH R01 EY029700 to Alan Yuille and a gift grant from Open Philanthropy to Cihang Xie.

References

[1] Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4845–4854, 2019. 8

[2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. 7

[3] Sara Beery, Yang Liu, Dan Morris, Jim Piavis, Ashish Kapoor, Neel Joshi, Markus Meister, and Pietro Perona. Synthetic examples improve generalization for rare classes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020. 7

[4] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010. 3

[5] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91, 2018. 1

[6] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017. 2, 8

[7] Yi-Hsin Chen, Wei-Yu Chen, Yu-Ting Chen, Bo-Cheng Tsai, Yu-Chiang Frank Wang, and Min Sun. No more discrimination: Cross city adaptation of road scene segmenters. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 7

[8] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019. 5

[9] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 7

[10] Yao Feng, Haiwen Feng, Michael J. Black, and Timo Bolkart. Learning an animatable detailed 3D face model from in-the-wild images. *ACM Transactions on Graphics (ToG), Proc. SIGGRAPH*, 40(4):88:1–88:13, Aug. 2021. 5

[11] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. *arXiv preprint arXiv:2012.03065*, 2020. 2

[12] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016. 7

[13] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, S. M. Ali Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. In *ICML*, 2018. 7

[14] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, Jan. 2016. 2, 7

[15] R. V. Garcia, L. Wandzik, L. Grabner, and J. Krueger. The harms of demographic bias in deep face recognition research. In *2019 International Conference on Biometrics (ICB)*, pages 1–6, 2019. 1

[16] Baris Gecer, Binod Bhattarai, Josef Kittler, and Tae-Kyun Kim. Semi-supervised adversarial learning to generate photorealistic face images of new identities from 3d morphable model. In *Proceedings of the European conference on computer vision (ECCV)*, pages 217–234, 2018. 7

[17] Baris Gecer, Alexandros Lattas, Stylianos Ploumpis, Jiankang Deng, Athanasios Papaioannou, Stylianos Moschoglou, and Stefanos Zafeiriou. Synthesizing coupled 3d face modalities by trunk-branch generative adversarial networks. In *European conference on computer vision*, pages 415–433. Springer, 2020. 7

[18] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc. ICLR*, 2015. 2, 8

[19] Patrick J Grother, Mei L Ngan, and Kayee K Hanaoka. Face recognition vendor test part 3: Demographic effects. 2019. 1

[20] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. PMLR, 2018. 2, 7

[21] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 5

[22] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 5, 6

[23] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007. 5

[24] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017. 7

[25] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 7

[26] Adam Kortylewski, Bernhard Egger, Andreas Schneider, Thomas Gerig, Andreas Morel-Forster, and Thomas Vetter. Empirically analyzing the effect of dataset biases on deep face recognition systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2093–2102, 2018. 7

[27] A. Kortylewski, B. Egger, A. Schneider, T. Gerig, A. Morel-Forster, and T. Vetter. Analyzing and reducing the damage of dataset bias to face recognition with synthetic data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2261–2268, 2019. 7

[28] Adam Kortylewski, Andreas Schneider, Thomas Gerig, Bernhard Egger, Andreas Morel-Forster, and Thomas Vetter. Training deep face recognition systems with synthetic data. *arXiv preprint arXiv:1802.05891*, 2018. 7

[29] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):194:1–194:17, 2017. 5

[30] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, and Zhaoyang Lv. Neural 3d video synthesis. *arXiv preprint arXiv:2103.02597*, 2021. 2

[31] Henry Lieberman. How to color in a coloring book. *SIGGRAPH Comput. Graph.*, 12(3):111–116, Aug. 1978. 4

[32] Gilles Louppe and Kyle Cranmer. Adversarial variational optimization of non-differentiable simulators. *arXiv preprint arXiv:1707.07113*, 2017. 7

[33] 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. 2, 8

[34] Richard T Marriott, Sami Romdhani, and Liming Chen. A 3d gan for improved large-pose facial recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13445–13455, 2021. 7

[35] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016. 7

[36] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020. 2

[37] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. ACM, 2017. 2, 8

[38] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *2009 sixth IEEE international conference on advanced video and signal based surveillance*, pages 296–301. Ieee, 2009. 5

[39] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1406–1415, 2019. 2, 7

[40] Nicolas Pinto, James J DiCarlo, and David D Cox. Establishing good benchmarks and baselines for face recognition. In *Workshop on Faces In'Real-Life'Images: Detection, Alignment, and Recognition*, 2008. 7

[41] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 5

[42] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, pages 102–118. Springer, 2016. 7

[43] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016. 7

[44] Nataniel Ruiz, Sarah Adel Bargal, and Stan Sclaroff. Disrupting deepfakes: Adversarial attacks against conditional image translation networks and facial manipulation systems. In *European Conference on Computer Vision*, pages 236–251. Springer, 2020. 8

[45] Nataniel Ruiz, Sarah Adel Bargal, and Stan Sclaroff. Protecting against image translation deepfakes by leaking universal perturbations from black-box neural networks. *arXiv preprint arXiv:2006.06493*, 2020. 8

[46] Nataniel Ruiz, Eunji Chong, and James M. Rehg. Fine-grained head pose estimation without keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018. 7

[47] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. Learning to simulate. In *International Conference on Learning Representations*, 2018. 7

[48] Nataniel Ruiz, Barry-John Theobald, Anurag Ranjan, Ahmed Hussein Abdelaziz, and Nicholas Apostoloff. Morphgan: One-shot face synthesis gan for detecting recognition bias. In *32nd British Machine Vision Conference 2021, BMVC 2021, Virtual Event, UK*, 2021. 2, 7

[49] Hadi Salman, Andrew Ilyas, Logan Engstrom, Sai Vemprala, Aleksander Madry, and Ashish Kapoor. Unadversarial examples: Designing objects for robust vision. *arXiv preprint arXiv:2012.12235*, 2020. 2

[50] Hadi Salman, Jerry Li, Ilya P Razenshteyn, Pengchuan Zhang, Huan Zhang, Sébastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *NeurIPS*, 2019. 8

[51] Michelle Shu, Chenxi Liu, Weichao Qiu, and Alan Yuille. Identifying model weakness with adversarial examiner. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11998–12006, 2020. 8

[52] Alvy Ray Smith. Tint fill. *SIGGRAPH Comput. Graph.*, 13(2):276–283, Aug. 1979. 4

[53] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *In Proc. ICLR*, 2014. 2, 8

[54] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 7

[55] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017. 2, 7

[56] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5265–5274, 2018. 5

[57] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 4

[58] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 5

[59] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014. 5

[60] Alan L Yuille and Chenxi Liu. Deep nets: What have they ever done for vision? *International Journal of Computer Vision*, 129(3):781–802, 2021. 1

[61] Xiaohui Zeng, Chenxi Liu, Yu-Siang Wang, Weichao Qiu, Lingxi Xie, Yu-Wing Tai, Chi-Keung Tang, and Alan L Yuille. Adversarial attacks beyond the image space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4302–4311, 2019. 8