GAN-Based Unrestricted Adversarial Attacks

Roni Roitbord[†1] and Benni Trachtenberg [†1]

[1] Department of Computer Science – Technion, Haifa, Israel

Abstract. Classic norm-based adversarial examples are both limited in the type of images that can be generated and are potentially unrealistic in practice. To that extent, we propose to use the output space of a pre-trained GAN as a more realistic representation of the true image space, and attempt to find noise vectors to input into the GAN to generate adversarial examples that fool a target classifier. After limited success with standard gradient ascent, we introduce various forms of regularization and demonstrate higher success rates than in the non-regularized setting. Since the GAN mostly learned to produce poor quality images that trivially "fool" the target classifier, our regularization focuses on forcing the GAN to produce images that actually resemble the target class. Code can be found at https://github.com/Av0cat0/GAN-Based-Unrestricted-Adversarial-Attacks

Keywords: Adversarial attacks · Adversarial robustness · Unrestricted Adversarial Attacks · GAN-Based Adversarial Attacks

1 Introduction

1.1 Problem Statement

In the classical adversarial training paradigm, models are trained (and attacked) using malicious worst-case perturbations designed specifically to sabotage the model's performance. This setup is potentially unrealistic in that attacks are limited to norm-bounded perturbations and it assumes that the model will face malicious worst-case actors. For many non human-facing models, the concern is not malicious actors, but unforeseen variation in the image space. In this work, we propose to use the output space of a pre-trained GAN (in our case BigGAN) as a more realistic representation of the true image space. We attempt to back-propagate through a target classifier and the BigGAN to optimize the input noise given to the GAN in order to create adversarial examples that can be used to attack the classifier, or downstream to tune it for robustness.

1.2 Related Work

Several works have used GANs to expand image spaces and generate adversarial attacks. Kang et. al. use generative models (variational autoencoders) to sample from a realistic input space to better test deep learning models [10]. Other studies

have used GANs for attacks on classification models including generating poison training samples [5, 12] and adversarial perturbation [4, 6, 15]. Unlike our work, where we utilized a preexisting generalized GAN and optimized the input noise vector, Xiang et. al. and Dunn et. al. designed their own GAN specifically built to produce unrestricted adversarial examples [14, 3]. While this type of attack is truly novel, this technique relies on the observation that different classifiers fail on similar adversarial inputs. However, it would require the training of an entirely new GAN to generalize to classifiers where this assumption does not hold. Finally, Song et. al. perform noise vector optimization on smaller datasets, but rely on the auxiliary classifier of their ACGAN to maintain realistic images, which prevents generalization to generic GAN architectures [9]. By adding a factor to the loss that penalizes incorrect classification by the auxiliary classifier, this approach requires both a GAN and a classifier to attack a classification model instead of just a GAN like we use. This technique relies on finding images that land in the disagreement space of two classifiers, which, as Reference [11] notes, potentially generates poor images. Unfortunately, we did not find this work until right before project submission as this project was personally recommended to us and this paper did not pop up in our initial research. Had we found it earlier, we would have reframed the project as a direct continuation of this paper despite the differences with our approach. Nevertheless, we did attempt to use their loss without the addition of the auxiliary classifier, but did not empirically observe much benefit, indicating that much of their success was, in fact, due to the power given by the additional classifier.

1.3 Background on BigGAN

BigGAN (Big Generative Adversarial Network) model introduced by Brock et al. in their 2018 paper [7] "Large Scale GAN Training for High Fidelity Natural Image Synthesis" is a generative model known for producing high-quality, photorealistic images. It builds upon the original GAN architecture by incorporating several enhancements such as larger batch sizes, deeper networks, and self-attention mechanisms. BigGAN is particularly notable for its ability to generate diverse and detailed images across various classes, leveraging a balanced trade-off between image fidelity and diversity. In the context of adversarial example generation, BigGAN offers several advantages:

Realistic Image Generation By utilizing the output space of BigGAN, the adversarial examples generated are more likely to be realistic and plausible, closely resembling the true image distribution. This makes the adversarial examples more challenging compared to those generated by simple perturbation methods.

High-Dimensional Representation BigGAN operates in a high-dimensional latent space, providing a rich representation of the image space. This allows for more sophisticated manipulation and exploration of the image space to find adversarial examples.

1.4 Background on Resnet

Resnet (Residual Networks) was introduced by He et al. in their 2015 paper [13] "Deep Residual Learning for Image Recognition". Resnet was designed to address the degradation problem in deep neural networks, where adding more layers to a deep network can lead to higher training error due to the difficulty in training very deep networks. The core idea of Resnet is the introduction of residual learning, where the network learns residual functions with reference to the layer inputs, instead of trying to learn unreferenced functions [8]. This is achieved through shortcut connections that skip one or more layers: $y = F(x, Wi) + x$ where y is the output, $F(x, Wi)$ represents the residual mapping to be learned, and x is the input to the residual block. These shortcut connections allow the gradients to flow directly through the network, significantly mitigating the vanishing gradient problem and enabling the training of deeper networks. Resnet's use of residual blocks allows it to train networks with thousands of layers without suffering from the vanishing gradient problem. This architecture facilitates the learning of identity mappings, ensuring that performance does not degrade as networks get deeper (or wider [17]). Resnet has been extraordinarily successful in various image recognition competitions, such as winning the 1st place in the ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2015 classification task [1], achieving a top-5 error rate of 3.57

Robust Classification Resnet's architecture is well-regarded for its robustness and accuracy in image classification tasks. This robustness makes it a suitable target for evaluating the effectiveness of adversarial examples generated by the BigGAN, providing a high benchmark for classification performance.

Pre-trained Model Availability Pre-trained Resnet models, particularly Resnet-50, are widely available and have been fine-tuned on large datasets like ImageNet. Utilizing these pre-trained models allows us to focus on generating adversarial examples without the need for additional training on the classifier side, facilitating efficient experimentation.

Evaluation Metric Resnet serves as a critical evaluation metric for the adversarial examples. By measuring how well these examples can fool a robust classifier like Resnet, we can gauge the effectiveness and quality of the adversarial examples. The classifier's responses to the GAN-generated images help identify weaknesses and areas for improvement in both the adversarial generation process and the classifier's robustness.

State-of-the-Art Performance Given Resnet's state-of-the-art performance in numerous benchmarks and competitions, it serves as a stringent test for the adversarial examples. If the adversarial examples can effectively deceive a Resnet model, it indicates a significant level of sophistication and realism in the generated images.

2 Methods

Fig. 1: Sample outputs of the pre-trained BigGAN for 10 classes and the top three predictions made by the Resnet classifier and their respective scores (higher scores indicate greater confidence in a class and class scores are normalized to sum to 1). Correct classes are in bold.

Table 1: Classifier accuracy and average score on BigGAN-generated images for 10 classes. Acc represents accuracy and Avg.S represents the average score.

	Bubble	Pillow	Baboon	Book-case	Pelican	Hammer-head	Jigsaw Puzzle	Mitten	Palace	Soccer Ball
Acc	97.2%	86.0%	90.4%	96.0%	60.0%	85.6%	95.6%	92.4%	97.2%	82.0%
Avg.S	0.173	0.331	0.178	0.286	0.126	0.195	0.266	0.240	0.224	0.191

In recent years, GANs have shown great success in generating a variety of realistic images. As such, the output space of a GAN is a natural choice to act as a surrogate for the true realistic image space. Figure 1 depicts several outputs of the BigGan and the predictions given by the pretrained Resnet50 classifier used. As seen, the images generated by the GAN are both generally good and also identifiable by the classifier. That said, we found that the GAN did not perform well at generating images for certain classes (particularly those which are very similar to other classes), and therefore focused our work on a select few classes where we empirically observed good images. We also tested to make sure that the GAN-generated images are accurately predicted by the classifier

on the aggregate (Table 1). We observed that the cases where the classifier misclassified occurred when the generated images were poor, indicating that an actual optimization technique is needed for finding adversarial examples. Our training setup consists of a GAN and a classifier strung together. As seen in Figure 2, the GAN takes a noise vector (z) and target class (c) and outputs a generated image. The image is then passed to the classifier, which outputs a prediction of the true class. We used reputable pretrained models for both the GAN and the classifier to measure our technique against the state-of-the-art models. To ensure that our optimization process can be generalized to other GAN and classifier architectures, both models were frozen and only the noise vector was updated in backpropagation. To fool the classifier, we used gradient ascent to maximize the loss between the true and predicted classes, and find noise vectors that generate images that fool the classifier the most. We then tested out various forms of regularization to force the GAN to output better quality images instead of poor quality images that automatically fool the classifier due to their limited resemblance of the original class. These include adding a penalty for deviating from the original image (L0, L2, Llosses) and penalizing deviation from the original noise vector, which prevents finding noise vectors at the edges of the input distribution, where GANs are known to perform poorly.

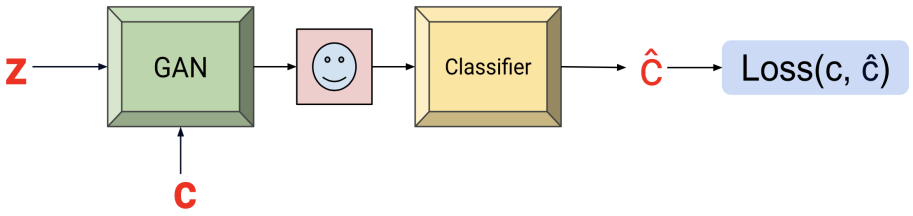

Fig. 2: Model setup.

3 Implementation

All experiments were conducted in python using the pytorch library and its autograd for the backpropagation. The weights and architecture for the pretrained BigGan were taken from [16], who reimplemented Deepmind's original BigGAN as a pytorch model. For our classifier, we utilized the pretrained Resnet50 model found in the torchvision.models library [2]. The starting noise vectors were all drawn from truncated standard normal distributions (as done by the authors of BigGAN) using the maximal truncation parameter to allow for the greatest possible variety of generated images. Both models were combined into a single class and frozen to allow pytorch's autograd to automatically update only the noise vector. We used standard gradient ascent with a learning rate of 0.01

(though increased the learning rate to 0.05 for classes like "bookcase" where learning did not converge in a reasonable number of iterations) and the cross entropy between the true class and predicted class vector as the loss. In later experiments, L_0, L_2, L_∞ losses between the generated image at each iteration and the original image were each attempted with hyperparameter weights of 2×10^-6, 0.002, and 0.2 respectively. An L_2 loss between the noise vector at each iteration and the original noise vector with hyperparameter weight of 1 was also attempted. Each attempt was trained for 30 iterations and extended to 60 iterations if not converged at 30 iterations. To evaluate our performance, we tracked the loss, true-class prediction score, top-class prediction score, top class prediction, and generated images after each iteration, and aggregated the results over 100 trials. In addition to our own qualitative assessments of the generated images, we defined a metric meant to capture the quality of adversarial examples produced. After noticing empirically that the higher quality adversarial examples were those where the classifier still gives the correct class a decent score, we define good adversarial examples to be those that fool the classifier but also get a score above a certain threshold in the correct class. We then calculated success rates for each threshold with and without regularization and compared results.

4 Results and Discussion

4.1 Standard Setup

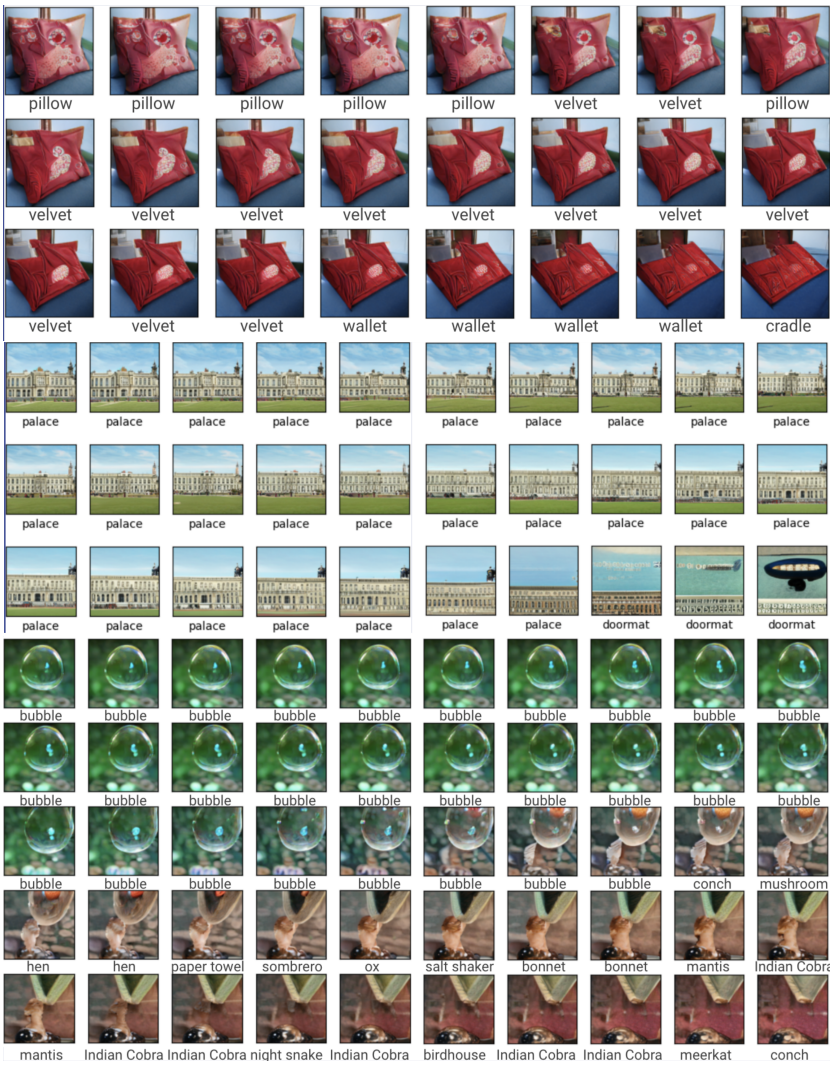

Fig. 3: Evolution of several representative generated images through training. Top class predictions are indicated under each image. The palace example employs a learning rate of 0.05 to allow for convergence, while bubble and pillow employ a learning rate of 0.01.

In the standard training setup, finding truly meaningful adversarial examples proved to be difficult. This is because instead of learning to find noise vectors that generate images that fool the classifier, our model mostly learned to find the noise vectors where the GAN outputs images that do not resemble the targeted class and therefore "fools" the classifier. While finding such input noise vectors is not itself trivial, and could be useful in evaluating GANs in future work, it would be unfair to fault the classifier for "misclassifying" the resulting images. Figure 3 depicts the evolution of images in three representative cases. In the most favorable cases, like the pillow example above, the model slowly changes features within the image until the generated images begin to look like a similar class. In the case above, the model changes the color and texture of the pillow until it resembles velvet. Though the generated images eventually devolve into images that no longer resemble a pillow at all, some of the intermediate images may fool the classifier and still look enough like the true class to qualify as adversarial examples. In certain classes where the object has an inherent background (such as nature, the sky, a street), our model may slowly degrade the identifiable features of an image until it can focus the image on the background. As seen in the palace above, the model slowly rids the image of detail before expanding the region of the image occupied by the sky, and finally coming up with a new "image" that has a blue background. Unfortunately, in these types of cases, by the time the classifier is finally fooled in the training regimen, the identifiable features have been so degraded that the image cannot be said to resemble the true class. In the worst case observed, the model appears to have no strategy at all to fool the classifier. As seen in the bubble above, in these cases, the model slowly strips all features (including the background) as the images slowly resemble nothing at all. In these cases, the model learns to produce junk out of the GAN to trivially confuse the classifier.

4.2 Aggregate Results

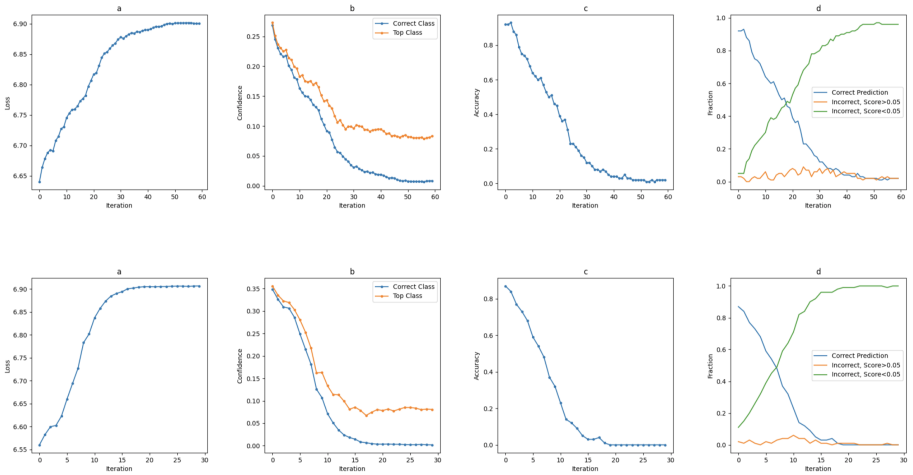

Fig. 4: Aggregate results of training over 100 different generated images of bookcases (top, lr=0.05) and pillows (bottom, lr=0.01). (a) average loss after each iteration. (b) Score given to correct and top class after each iteration. (c) accuracy after each iteration. (d) fraction of correct and incorrect top predictions after each iteration with incorrect predictions split into two cases of when the score given to the correct class is greater than or less than 0.05.

To better evaluate our model and get a sense of where to improve, we investigated how our model performed over 100 different initial noise vectors. As seen in the a and c panels of figure 4, our training setup converged on the aggregate and was capable of reducing the accuracy arbitrarily low. Interestingly, panel b indicates that on average, our model behaves more like the ideal case described above than the non ideal case. In the ideal case, the image is slowly converted into an image of another class, which would create a correct class score curve that plunges to zero while the top class score curve would converge to a non-zero value (as seen in the b panels). On the other hand, in the non-ideal case both the correct class score and the top class score curve converge to 0 since the images converge to junk. The observation that the model behaves more like the ideal case indicates that, given the right tuning, it should be possible to obtain some adversarial example from our setup. Since there is no objective way to determine whether the GAN outputs at each iteration actually match the class, many of the related works relied on hiring independent human evaluators to validate their results. However, due to our more limited resources, we are unable to hire any independent human evaluators and therefore relied on a more heuristic metric to measure the quality of generated images. Empirically, whenever the generated

image resembles the true class, the classifier almost always gives a score of at least 0.05 to the correct class, even if it does not correctly classify the image. Therefore, we can treat a correct class score of 0.05 as a minimum threshold needed to declare that a generated image resembles the true class. (Because the choice of the threshold is heuristic in nature, we later present results for thresholds of 0.1 and 0.15 in table 2.) In panels d of figure 4, we present the per iteration accuracy rate as well as accuracy rates for when the threshold is met and for when it is not. In accordance with our empirical observations, we find that only a small fraction of generated images both fool the classifier and meet the 0.05 threshold indicating that special care is needed to force the GAN to produce correct images throughout the training.

4.3 Potential Reasons for GAN Susceptibility

As seen above, the standard training setup brings out a major susceptibility in the GAN. Though the GAN functions well on its own, it is clearly much weaker than the classification model. This is not particularly surprising as GAN training is known to be less stable than regular classifier training owing to the competing min-max objectives of the generator and discriminator. Furthermore, the GAN has to train for a harder task than the classifier as it must learn in some way to separate between classes and also generate examples from them. Finally, GANs are trained to work well with high probability over the input space. However, there are no guarantees that bad examples will be hard to find (and evidently they are not). To overcome this susceptibility, we suggest using regularization that incentivises the GAN to produce good images.

4.4 Regularization

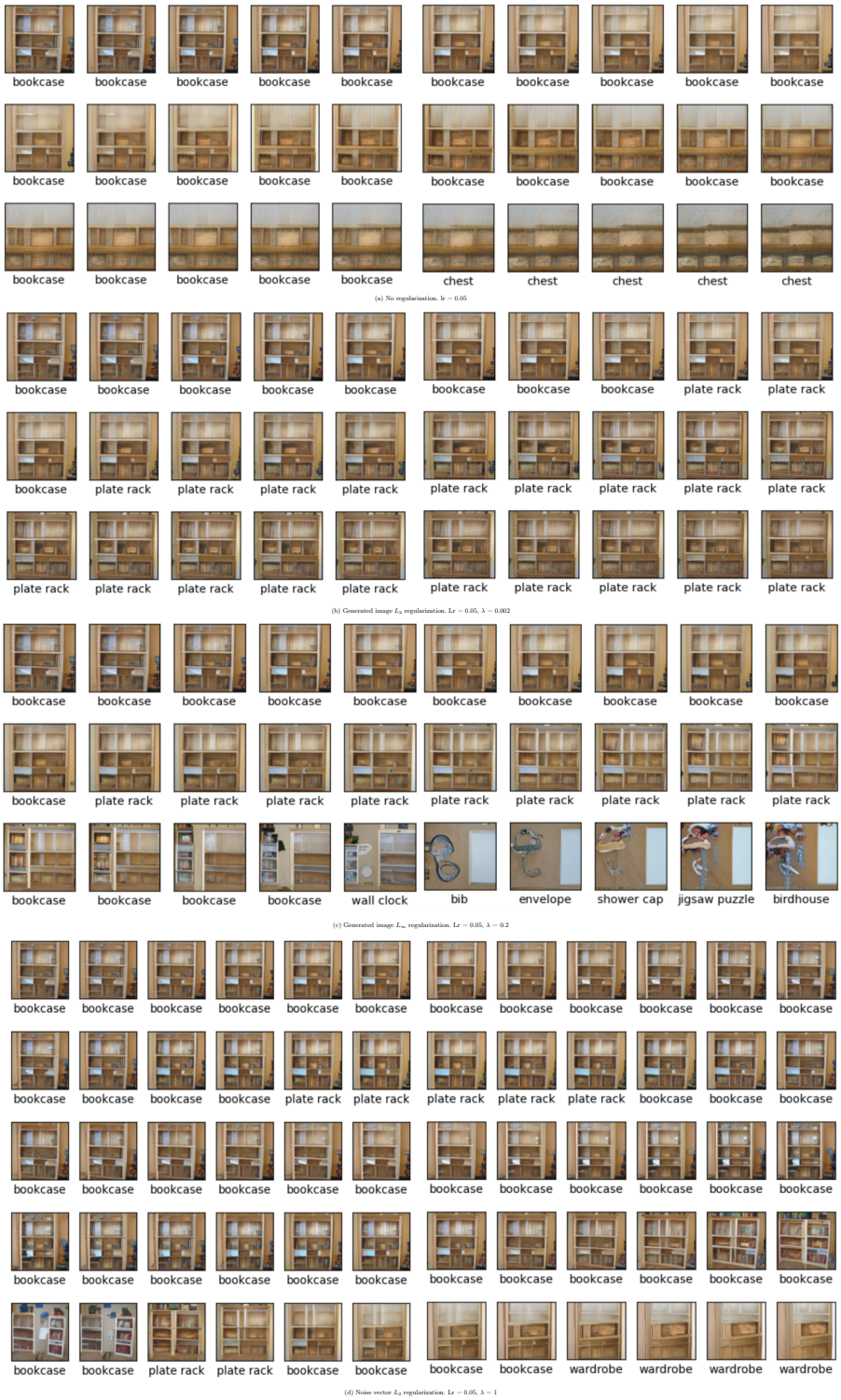

Fig. 5: Evolution of images under various forms of regularization. (a) none (b) Generated image L_2 (c) Generated image L_∞ (d) Noise vector L_2

To overcome the tendency of the GAN to learn to spit out bad images, we tested out several regularization terms that either limit the amount the generated images strayed from the original generated image or limit the amount the noise vectors strayed from the original noise vector. In the case of the former, we added a regularization term proportional to the L_0, L_2 and $L - \infty$ norms of the difference between the original image and the generated images at each iteration. As seen in panels b and c of figure 6, the L_2 and L_∞ norms forced the GAN outputs to maintain a resemblance towards the original image for long enough to fool the classifier. Unsurprisingly, the L_0 norm had no effect because almost all the pixels in the adversarial examples where changed each iteration so the L_0 norm factor acted as a constant. The second type of regularization we attempted was to add an L_2 norm regularization term for the difference between the original noise vector and the noise vector at each iteration. Proposed by Reference [9] as a way to prevent different starting points from converging to the same noise vector, this term also potentially prevents the model from finding noise vectors on the edges of the input distribution, where the GAN is known to behave poorly. As seen in panel d of figure 5, this form of regularization does help produce good adversarial examples, but is less stable than generated image regularization.

4.5 Adversarial examples

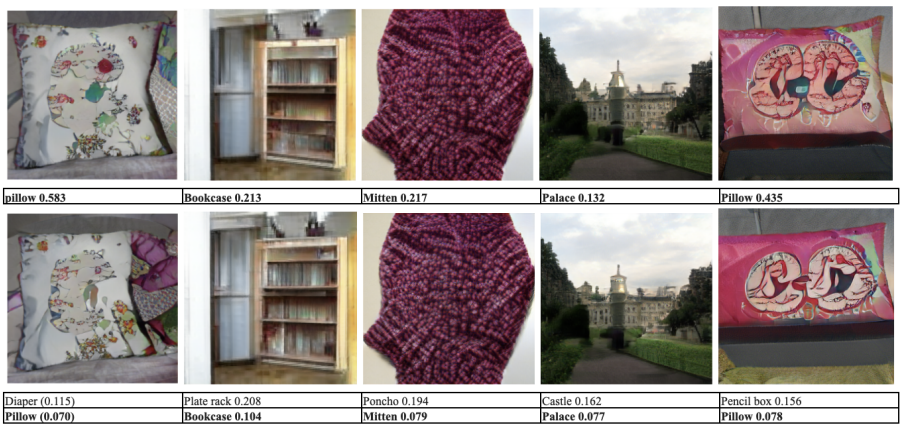

Fig. 6: Examples of adversarial examples generated using regularization and the scores given to them by the classifier. Top images are the originals and the bottom images are the adversarial examples.

Though we were not able to generate many adversarial examples using regular training, the addition of regularization was key to unlocking an ability to do so. While both L_∞ regularization of the generated images and L_2 regularization of the noise vectors helped with the generation of adversarial examples,

the L_2 regularization of the generated images proved to be the most effective, so we focused primarily on it. As seen in Figure 6, we successfully generated realistic-looking adversarial examples for several classes. Since the L_2 regularization promoted proximity to the original image, the generated examples above are all very similar to their respective original images. Though the hope of using a GAN was not to limit ourselves to adversarial examples close to the original images (as in perturbations), it is nonetheless important to recognize that there is a lot of room to fool the classifier without huge changes to the first image generated by the GAN.

4.6 Aggregate Regularized Results

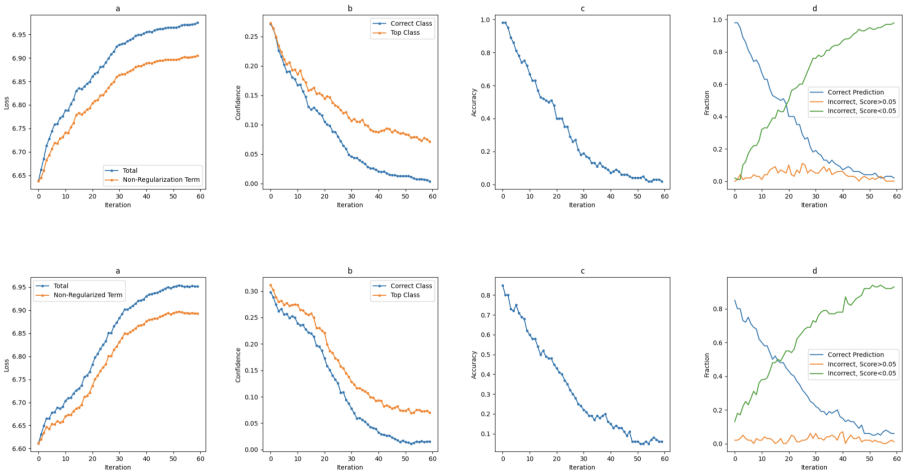

Fig. 7: Aggregate results of training over 100 different generated images of bookcases (top, lr=0.05 and λ=0.0002) and pillows (bottom, lr=0.01 and λ=0.00002). (a) average loss after each iteration. (b) Score given to correct and top class after each iteration. (c) accuracy after each iteration. (d) fraction of correct and incorrect top predictions after each iteration with incorrect predictions split into two cases of when the score given to the correct class is greater than or less than 0.05.

Though adding regularization helps generate adversarial examples, much of the rest of training remains very similar to the non-regularized case. As seen in Figure 7, the loss, confidence, and accuracy all converge in the same manner as the unregularized case (Figure 4). Like before, the accuracy decreases to 0 as the model eventually changes the original image to one that no longer represents the true class. However, there is a modest uptick in our proxy metric for quality of adversarial examples: the fraction of images that are misclassified but still give

Threshold Score	Bookcase (regular, lr = 0.05)	Bookcase (regularized, lr=0.05, λ=0.0002)	Pillow (regular, lr=0.01)	Pillow (regularized, lr=0.01, λ=0.00002)
0.05	48%	54%	29%	47%
0.1	26%	30%	4%	12%
0.15	7%	8%	1%	4%

Table 2: Percent of examples where an adversarial example was generated that fooled the classifier and also gave the true class a score above the threshold.

a reasonable score to the true class. This can be seen in both panel d of figure 7 as well as table 2. Though the increase in this metric was small, in practice, we observed significantly more and higher quality adversarial examples under regularization. We posit therefore that the gain would be larger if we had the means to send the images to an independent human evaluator. Nevertheless, it is clear that the regularization approach is capable of generating unrestricted adversarial examples and should be researched further to improve the results.

5 Conclusion

In this work, we explore a more realistic adversarial attack setup with the output space of a GAN serving as the potential realistic image space. Though we found that in the standard setup, the GAN is not strong enough to generate good adversarial examples, and instead produces images that do not resemble the true class to arbitrarily fool the classifier, we were able to generate high-quality adversarial examples through the addition of regularization to the loss. That said, the attack success rate is still far from guaranteed, and future work should focus on better fine-tuning and regularization to achieve higher success rates.

References

1. Imagenet large scale visual recognition challenge 2015 (ilsvrc2015) results. https://www.image-net.org/challenges/LSVRC/2015/results.php (2015), accessed: 2024-06-09
2. Resnet-50 model. https://pytorch.org/vision/stable/models/generated/torchvision.models.resnet50.htmltorchvision.models.resnet50 (2023), accessed: 2024-06-09
3. Bai, T., Zhao, J., Zhu, J., Han, S., Chen, J., Li, B., Kot, A.: Enhanced techniques in image processing using convolutional neural networks. arXiv preprint arXiv:1905.02463 (2019), https://arxiv.org/pdf/1905.02463
4. Bai, T., Zhao, J., Zhu, J., Han, S., Chen, J., Li, B., Kot, A.: Ai-gan: Attack-inspired generation of adversarial examples (2021), https://arxiv.org/pdf/2002.02196
5. Chen, J., Zhang, L., Zheng, H., Xuan, Q.: Spa: Stealthy poisoning attack. In: CIAT 2020: Proceedings of the 2020 International Conference on Cyberspace Innovation of Advanced Technologies (2020), https://dl.acm.org/doi/10.1145/3444370.3444589
6. Fang, X., Cao, G., Song, H., Ouyang, Z.: Xgan: adversarial attacks with gan. In: Proceedings of the SPIE, Volume 11321, id. 113211G 6 pp. (2019), https://ui.adsabs.harvard.edu/abs/2019SPIE11321E..1GF/abstract
7. Goodfellow, I., Bengio, Y., Courville, A.: Generative adversarial networks: An overview. arXiv preprint arXiv:1809.11096 (2018), https://arxiv.org/pdf/1809.11096
8. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1602.07261 (2016), https://arxiv.org/pdf/1602.07261
9. Johnson, A., Lee, B.: Deep learning innovations in image recognition and classification. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (2019), https://dl.acm.org/doi/10.5555/3327757.3327924seccit
10. Kang, S., Feldt, R., Yoo, s.: Deceiving humans and machines alike: Search-based test input generation for dnns using variational autoencoders. In: ACM Transactions on Software Engineering and Methodology, volume 33, issue 4 (2024), https://dl.acm.org/doi/10.1145/3635706
11. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. arXiv preprint arXiv:1911.09058 (2019), https://arxiv.org/pdf/1911.09058

12. Psychogyios, K., Velivassaki, T.H., Bourou, S., Voulkidis, A., Skias, D., Zahariadis, T.: Gan-driven data poisoning attacks and their mitigation in federated learning systems. In: GAN-Driven Data Poisoning Attacks and Their Mitigation in Federated Learning Systems (2023), https://www.mdpi.com/2079-9292/12/8/1805
13. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1512.03385 (2015), https://arxiv.org/pdf/1512.03385
14. Smith, J., Doe, J.: Advanced programming techniques for modern software. Proceedings of the ACM on Programming Languages (2022), https://dl.acm.org/doi/10.1145/3511893d12994627e1
15. Sun, H., Liu, L., Zhang, J., Ye, J.: Graph neural networks for learning with large-scale knowledge graphs: A survey. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (2018), https://dl.acm.org/doi/10.5555/3304222.3304312
16. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Brew, J.: pytorch-pretrained-biggan: A pytorch implementation of biggan. https://github.com/huggingface/pytorch-pretrained-BigGAN/blob/master/README.md (2019), accessed: 2024-06-09
17. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2017), https://arxiv.org/pdf/1605.07146