Adversarial Learning on YOLOv8 Detection Model Using GANs

Daniel Guo
Columbia University
dg3287@columbia.edu

Gyung Hyun Je
Columbia University
gj2353@columbia.edu

***Abstract*—This project addresses the challenge of robustness in object detection models. We propose an approach to improve the robustness of such models under the assumption that they are differentiable. This approach is based on previous work improving robustness in classifiers. We then present experimental results where we improved the robustness of the YOLOv8 model for detecting sequences of digits from the Street View House Numbers (SVHN) dataset.**

Our experiments demonstrate the capacity of our approach to improve the robustness of an object detection model by training the YOLOv8 model to detect sequences of digits, generating new adversarial examples to attack the model, improving its robustness using the generated adversarial examples, and evaluating the improvement in robustness.

With successful adversarial learning, the YOLO model's performance on unseen adversarial data increased from 0.54 to 0.559 for recall, 0.561 to 0.644 for mAP50, and 0.578 to 0.659 for box precision. By demonstrating the effectiveness of adversarial training for the newest release of the YOLO model, the project proposes a novel approach to train adversarially robust object detection models.

***Index Terms*—Object detection, adversarial learning, GANs, FGSM, PGD, YOLOv8, YOLO**

I. Introduction

It is common practice to use an object detection model to read digits in various settings, including road signs and bank checks. Given the importance of accuracy and reliability of the detection model in deployment, it is critical that the model is robust against any adversarial manipulation of the images. Speed is also an important factor due to the large volume of checks being cashed each day.

We tackle this question about robustness from a few different directions. First, we will train the YOLOv8 model to detect a sequence of digits using the SVHN Dataset. Second, we will create adversarial examples using our GANs attack on our trained model. Third, we improve the robustness of our model by attacking it using new adversarial examples from FGSM, PGD, and GANs.

This report as organized as follows: we first introduce the general idea of adversarial attacks using traditional methods and GANs, and then we discuss our specific work attacking object detection models, specifically YOLOv8, and our implementation of the GANs attack.

Link to the project GitHub is found below in the reference.

This work was done for the final project of E6998-5 High Performance Machine Learning taught by Professor Parijat Dube and Professor Kaoutar El Maghraoui.

II. Contribution

The main contributions of our project are threefold:

1. Our focus on adversarial example generation for object detection is relevant to a wide range of domains where model robustness is paramount (e.g. digital check deposit for finance apps, road sign speed limit detection for autonomous vehicles).

2. YOLOv8 is the newest YOLO model (released January 2023). The project can add to people's understanding of its capabilities and limits.

3. To the best of our knowledge, GANs based attacks have only been used for classification models, and our work is the first to utilize this type of attack for object detection. We believe this work provides important insights to the efficacy of GANs attacks for non-classification domains.

III. Adversarial Attacks Background

There are numerous methods to generate adversarial examples in a white-box targeted setting for classification, as this is the most popular domain that adversarial attacks are studied. We focus on FGSM and PGD.

FGSM works as

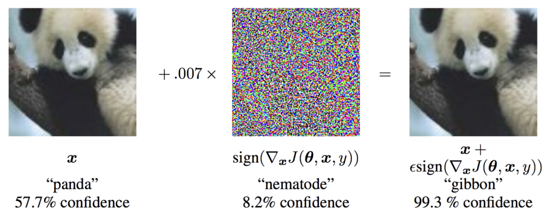
$$x' = x - \epsilon \operatorname{sgn}(\nabla L(\theta, x, y))$$

and PGD works as

$$x^{t+1} = \Pi_{\mathcal{S}}(x^t - \epsilon \operatorname{sgn}(\nabla L(\theta, x^t, y))),$$

where x is the original image, x^t is the image after t iterations, and y is the label for the targeted attack.

These attacks have demonstrated success in various classification tasks. For example, this example is a well know attack on the CIFAR dataset.


$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\operatorname{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon \operatorname{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

We will slightly modify these methods for object detection (described in Section VI Subsection A) in order to use them to evaluate our robustly trained model.

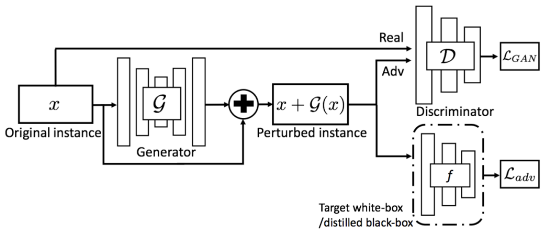
IV. Adversarial Attacks with GANs

GANs can be trained to generate perturbations on an input image, such that the attacked model (in our case the YOLOv8 model), makes a mistake. They are trained in much the same way as the traditional GANs framework but with two additions.

1. The generator receives feedback (in the form of gradients) from the attacked model to learn how to perturb the input image to make the attacked model make a mistake.

2. There is an additional hinge loss in the generator's loss to prevent the perturbation from getting too big.

The method is summarized in this figure. We leave the exact details of our implementation of GANs for object detection (different from previous methods for classification) for Section VI Subsection B.

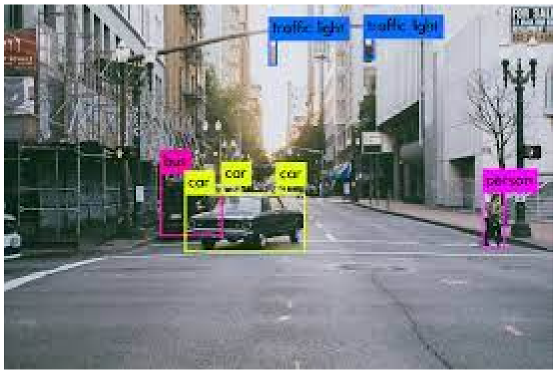


GANs have been applied very successfully as attackers in classification tasks. The referenced paper [1] achieved first place in MadryLab's MNIST Adversarial Examples Challenge outperforming other methods using FGSM, PGD, and Guassian filtering.

V. The Problem in Object Detection

Robust object detection models are important because they are used across the industry for various high-risk tasks such as banking, food delivery, autonomous vehicles, etc. Object detection is the task of (1) detecting objects in an image, (2) finding the borders of the object, and (3) classifying the object. This 3 stage process can make object detection models less robust and therefore easier to attack than purely classification models. Due to their usage in high-risk environments, a model that lacks robustness can't be safely pushed into production.

The YOLOv8 model is relatively new and the YOLO family in general is special due to their ability to be used in real-time.



The main change that we need to make in the methods discussed earlier to adapt them to the object detection domain (as opposed to purely classification problems) is the loss function that YOLO is trained on $L_{\text{YOLO}}(x, t)$. The label t comprises of the annotations of the image, including the bounding boxes and classification labels. The loss itself is much more complicated than classification, since it contains three components for (1) detection, (2) localization, and (3) classification.

In our experiments, we attack primarily the detection aspect of this loss by using an empty annotation (no objects in the image).

VI. Attack Methodology

A. Traditional Attacks

1) FGSM: In the context of YOLO, FGSM behaves as

$$x' = x - \epsilon \text{sgn}(\nabla L_{\text{YOLO}}(\theta, x, t)),$$

where x is the original image and t is the annotation we are using for a targeted attack. ϵ is a tunable parameter that determines the attack strength we give FGSM. A higher ϵ indicates we allow the perturbation to be larger, which may be more visible to the human eye.

FGSM works by calculating the gradient of the loss function with respect to the input data and then adding a perturbation in the direction of the sign of the gradient, scaled by a small epsilon value. This perturbation is calculated to minimize the loss of the model on the input data and the targeted attack label. The result is an input that is very similar to the original but has been slightly modified to cause the model to fail to detect the objects in it (if any).

2) PGD: Similarly, PGD behaves as

$$x^{j+1} = \Pi_{\mathcal{S}}(x^j - \epsilon \text{sgn}(\nabla L_{\text{YOLO}}(\theta, x^j, t))).$$

PGD is a more advanced form of the FGSM attack, though more computationally expensive due to multiple iterations. PGD is iterative attack that works by repeatedly perturbing the input data in small steps as in FGSM and projecting the result back onto a valid domain to ensure that the perturbed data remains within some allowed perturbation domain around x $\mathcal{S}$. The perturbations in each iteration are calculated using the same method as the FGSM attack but are typically applied in smaller steps. The $\Pi_{\mathcal{S}}$ indicates that after each step of FGSM we project the result back into the allowed domain $\mathcal{S}$. Typically $\mathcal{S}$ is a ball around the original input image x of radius α. Similarly to before, α is a tunable parameter which determines attack strength and detectibility to the human eye.

B. GANs Attack

There is some justification for why the GANs attack is believed to be stronger and more useful compared to just FGSM or PGD. This attack is able to take advantage of a much stronger perturbation generator using a deep neural network, and by way of the discriminator, it can create perturbed images that appear much more similar to the original image to the human eye than the two other methods.

The GANs attack comes down to just training the GANs. The output of the training process is a generator G which, given an input image x, produces a perturbation x such that $x + G(x)$ is indistinguishable from x and causes YOLO to make a mistake. In our case this mistake is YOLO failing to detect objects that exist in x. The figure in Section IV presents an overview of the training process, and we describe it in detail here.

The training process involves 3 models: (1) the generator model, (2) the discriminator model, and (3) the attacked model (YOLO for us). We describe first the general purpose of each model and then specify the optimization problem each of them solves to achieve it.

(1) Generator G: given an input image x, G attempts to produces a perturbation x such that $x+G(x)$ appears to YOLO as if it does not have any objects in it.

(2) Discriminator D: given an input image, determine if it is a true input image x or a perturbed image $x + G(x)$ given by the generator.

(3) YOLO Y: gives signal (in the form of gradients) to G for how to perturb x to make Y model make a mistake. Y itself is not trained in this framework, since we are trying to attack the model Y.

Our work differs from previous works due to our selection of Y being a YOLOv8 model. Previous works have only focused on attacking classification models, whereas we are attacking a object detection model. This is made possible due to the YOLO architecture. Traditionally, object detection models utilize some sort of windowing technique to split the image into pieces. This makes getting gradients from the loss of the detection model with respect to the input image difficult. YOLO utilizes a fully convolution network, which allows gradients to be easily propagated from the loss to the input image.

Now we present the loss functions which incentivize the above objectives (1), (2), and (3).

The first loss is the same as in the traditional GANs framework given by

$$L_{\text{GANs}} = E_x \log(D(x)) + E_x[1 - D(x + G(x))].$$

This is the standard minimax formulation, which represents the discriminator's ability to distinguish between the "real" images x and the "fake" images modified by the generator $x + G(x)$. The discriminator trains to maximize this value, while the generator trains to minimize it. This in effect causes the generator to only create perturbations $G(x)$ which are undetectable by the discriminator.

The second loss comes from the YOLO model given by

$$L_{\text{YOLO}} = E_x l_{\text{YOLO}}(x + G(x), t),$$

where t is the empty annotation. l_{YOLO} represents how far the YOLO prediction on $x + G(x)$ is from the label t. By minimizing this loss, the generator is incentived to produce a perturbation such that $x + G(x)$ makes the YOLO model fail to detect objects in x (if any).

The third and final loss comes from the desire to constrain the adversarial example $x + G(x)$ to be close to x. Thus we bound the amount of perturbation as

$$L_{\text{hinge}} = E_x[\max(0, ||G(x)||_2 - c)],$$

where c is a tunable threshold for how much perturbation we allow. Note this is different than the previously discussed thresholds for FGSM and PGD because it is not a hard threshold and it uses the L_2 ball.

With all these pieces in place, we train the generator to minimize the loss

$$L = L_{\text{YOLO}} + \alpha L_{\text{GANs}} + \beta L_{\text{hinge}},$$

where α and β are tunable parameters which control the relative importance of each loss.

To reiterate, with generator G trained in this framework (and assuming it generalizes well), we have the ability for any input image x to apply the perturbation $G(x)$ such that $x + G(x)$ causes YOLO to make mistakes while looking similar to x. This allows us to use G in the same manner as FGSM and PGD to produce adversarial examples.

VII. Data

We used the Street View House Numbers (SVHN) Dataset from the Google Street View images. The dataset contains house number images with character level bounding boxes from 0 to 9. Each image has up to five digits.

We used the cropped versions (32 x 32 x 3) of the original images, which have variable image sizes. The final data contains 5251 training images, 1004 validation images and 500 test images.

VIII. Experiment

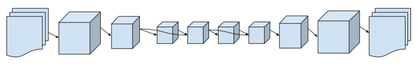
A. Setup

The overall setup of our experiments is as follows.

1. Train YOLOv8 model M on original data.
2. Train GANs G on same data to attack M.
3. Get dataset of original data + perturbed data from G.
4. Train YOLOv8 model M_r on new dataset.
5. Evaluate various attacks on M and M_r and compare attack efficacies.

B. Architecture

We forgoe describing the YOLOv8 architecture as it is not the focus of our project. The generator architecture is inspired by autoencoders and ResNet, comprising of 3 downsampling blocks (convolutions), 3 residual blocks, and 3 upsampling blocks (transpose convolutions). Rough sketch shown.

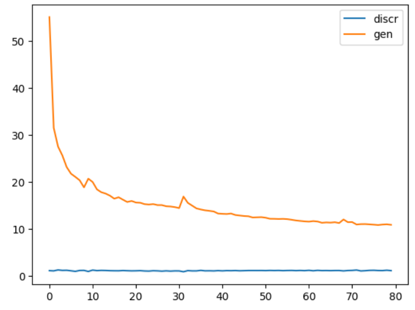


The discriminator is a typical CNN classifier with 4 convolution blocks with a sigmoid applied to the output.

C. Model hyperparameters and hardware

We trained the YOLO model on the SVHN dataset for 100 epochs, using learning rate 0.001, batch size 16, weight decay 0.0005, and Adam optimizer.

Then we trained the GANs model on a subset of SVHN dataset for 80 epochs, batch size 16, discriminator learning rate 0.01, generator learning rate 0.01, discriminator iteration 1, generator iteration 4, alpha 1, beta 10, threshold 0.3, and Adam optimizer. The trained model was used to produce GANs images. All models were run on 1 x NVIDIA Tesla P100-PCIE-16GB GPU.



Convergence of GANs during training.

D. Attack Efficacy

We created 500 adversarial images from test data with FGSM, PGD, and GANs attack methods and evaluated the YOLO model's performance on the adversarially perturbed dataset. The model performance on the adversarially perturbed dataset is found in Table 1. For FGSM and PGD with lower threshold ($\gamma = 0.01$), YOLO model has improved recall. Higher perturbation threshold was applied ($\gamma = 0.1$) to be effective on the YOLO model, after which the recall drops to 8.2% and 43.4% respectively. However, the adversarial images produced with higher threshold are less ideal for adversarial attack, since the perturbation becomes visually noticeable in the images. See the Figure 1-4 for reference.

	Recall	mAP50	Box Precision
GANs	0.54	0.561	0.578
FGSM ($\gamma = 0.1$)	0.082	0.038	0.0876
FGSM ($\gamma = 0.01$)	0.668	0.69	0.673
PGD ($\gamma = 0.1$)	0.434	0.439	0.476
PGD ($\gamma = 0.01$)	0.658	0.694	0.679
Unperturbed image	0.641	0.7	0.709

Table 1: Model recall, box precision, and mAP50 on adversarially perturbed images and original images before adversarial training.

After adversarial training with 1500 additional adversarially perturbed images in the training dataset, we saw a big improvement in model performance in the same unseen perturbed images across almost all metrics. YOLO model's recall on GANs images increased from 0.54 to 0.559, mAP50 from 0.561 to 0.644, and box precision from 0.578 to 0.659.

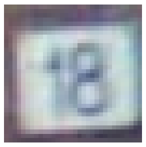


Fig. 1. Original

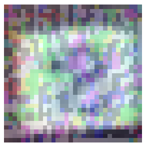


Fig. 2. FGSM threshold 0.1

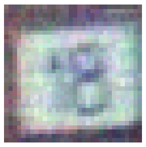


Fig. 3. PGD threshold 0.1

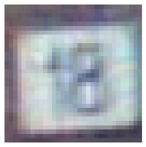


Fig. 4. GANs

	Recall	mAP50	Box Precision
GANs	0.569 +	0.644 +	0.659 +
FGSM ($\gamma = 0.1$)	0.0753 -	0.123 +	0.0421 -
FGSM ($\gamma = 0.01$)	0.670 +	0.67 -	0.681 +
PGD ($\gamma = 0.1$)	0.453 +	0.527 +	0.606 +
PGD ($\gamma = 0.01$)	0.684 +	0.714 +	0.639 -
Unperturbed image	0.676 +	0.714 +	0.66 -

Table 2: Model recall, box precision, and mAP50 on adversarially perturbed images and original images after adversarial training.

E. Training and Attack Speed

GANs has the overhead from training time, but it is very fast during inference. Methods like PGD need to run multiple iterations to achieve any efficacy. See the Table 3 for details on training time and inference time.

	Training per epoch (sec)	Image generation (sec)
GANs	18.459 15.543 (gen) 2.764 (discr) 1.951 (YOLO)	0.005
FGSM	0	0.0298
PGD	0	0.3203

Table 3: Model training time and inference time (seconds). PGD iterations used here is 10.

This can be explained by the procedure each method needs to perform during inference. With GANs, we just need to perform inference through the generator, which has a relatively small architecture. In comparison, FGSM and PGD (which has to do this once per iteration) need to forward propagate x through the YOLO model and then backpropagate the gradient from the loss to the input. Considering the size of the YOLOv8 model and the amount of computation required, the increase in time it takes for image generation from GANs to FGSM/PGD makes sense.

IX. Conclusion

In this report, we adapted a previous approach for attacking classification models using GANs to object detection models which are differentiable. We showed that using these examples to re-train the model improves its robustness. Finally, we showed through our experiments success using this method for a YOLOv8 model trained on the SVHN dataset. Our success came in the form of improved robustness of the YOLOv8 model against attacks by FGSM, PGD, and GANs. Furthermore, we gave benchmarks on the training and inference times for each of these attacks to generate new adversarial examples.

Acknowledgements

Thank you to Professor Parijat Dube and Professor Kaoutar El Maghraoui for their support with this project and for teaching this class.

References

[1] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu and D. Song, "Generating adversarial examples with adversarial networks", Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 3905-3911, 2018.

[2] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng, "Reading Digits in Natural Images with Unsupervised Feature Learning", NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011.

[3] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[4] J. I. Choi and Q. Tian, "Adversarial Attack and Defense of YOLO Detectors in Autonomous Driving Scenarios," 2022 IEEE Intelligent Vehicles Symposium (IV), Aachen, Germany, 2022, pp. 1011-1017, doi: 10.1109/IV51971.2022.9827222.

[5] Project GitHub: https://github.com/d-guo/e6998-5_final_project