Detecting Adversarial Attacks Against Object Dection Models

1st Reza Torbati
Senior in Computer Science
University of Oklahoma
Norman, United States
reza.j.torbati-1@ou.edu

2nd Zachary Singleton
Senior in Computer Science
University of Oklahoma
Norman, United States
zachary.singleton@ou.edu

***Abstract*—The field of deep learning has dramatically improved and evolved in recent years. Deep learning applications are beginning to be deployed in a wide variety of scenarios. One key flaw of deep learning algorithms is that they can be vulnerable to attacks orchestrated with adversarial input data. For image classification algorithms, several methods have been developed in order to detect adversarial attacks. However, few of these countermeasures have been put in place in real-time object detection systems, such as those used by autonomous vehicles. In this paper, we analyze some of the tests that have proven to be effective detecting adversaries for image classifiers when applied to real-time object detection systems.**

***Index Terms*—adversary detection, video processing, image analysis, statistical tests, deep learning, machine learning, autonomous vehicles, traffic sign detection, object detection, YOLO, YOLOv4**

I. Introduction

Machine learning is an increasingly researched topic in the field of computer science. Simply put, machine learning is the applied use of algorithms and large amounts of data to enable computer systems to autonomously learn and improve algorithms that can be used to make predictions in the real world. Machine learning is being impliemented more and more in high stakes, real world applications. However, this exposes these models to real world adversaries and, therefore, it is very important that machine learning models are capable of detecting when someone is attempting to mislead them.

Deep learning is a subset of machine learning that aims to create algorithms that allow computers to learn in a fashion that is similar to humans. Examples of this includes recognizing faces, handwriting, or identifying objects in images or videos which are tasks that are intuitively easy for humans, but has been extremely difficult for computers in the past. Some examples of significant applications of advancements in deep learning include object detection systems in self-driving cars, image or natural language processing, and predictive weather forecasting.

Our work focuses on object detection in image processing. Deep convolution neural networks (DCNNs) have long been proven to be effective at image classification, where they just have to write a label for what is in an image. However, in recent years, DCNNs have also been found to be effective at object detection, where they have to look at images with multiple objects and draw tight, labeled bounding boxes around each. Object detection algorithms have become increasingly faster and more accurate in recent years. However, despite how effective some object detection algorithms are, they are still vulnerable to adversarial attacks. This is serious because, for example, if an autonomous vehicle is fooled by adversarial attacks, people's lives could be at risk.

An adversarial attack is a scenario in which a malicious entity presents a specifically curated input to a deep learning algorithm that will cause the input to be mislabeled. Specific pixel perturbations or specially generated patches can be applied to images to fool an object detector into either ignoring or misclassifying an object. It is not hard to imagine the devastating consequences these attacks could have in applications where safety is of great concern. Depending on the adversarial attack that is generated, an object detector could misclassify objects in a wide range of possibilities. This is one of the most serious hurdles left for machine learning experts to mitigate before these algorithms should be used at scale in the real world.

In the past it has been difficult to create adversaries that actually trick object detectors [11]. For this reason, few techniques have been created to mitigate this threat. However, in recent years, there has been a lot of progress made to create better adveraries and, as a result, there are now many adversaries that have been proven to be effective at fooling object detectors [4].

For this project, we focus on one object detection model: You Only Look Once version 4 (YOLOv4). YOLOv4 is a powerful and widely used algorithm for object detection [8]. Ideally, given more time, we would have liked to expand our research to include a variety of deep learning object detectors, but we decided to focus on YOLO since YOLO is one of the best models at the current time. In order to create a well-rounded analysis of adversarial example detection, our project covers multiple avenues of detecting adversaries. We propose three potential solutions that have been used to detect adversaries in an image classifer context to detect adversarial examples in object detection:

1) We will attempt to mitigate the effect of adversarial patches by incorporating Gaussian noise. We speculate that injecting Gaussian noise to an image will change the

already-adversarial image enough to change YOLOv4 classification back to what it should be.

2) We will use a statistical two-sample test to determine if a sample contains adversarial images. Using a statistical test, we hope to determine if a sample of potentially adversarial images has a sufficiently different distribution from a sample of clean images.
3) Finally, we will train a custom YOLOv4 model to classify and detect adversarial patches on its own. This will be done by adding a new class to our model and labeling adversarial images in the hopes that the new YOLOv4 model can detect and bound adversarial parts of an image.

II. Background

A. Machine Learning Classifiers

The introduction provides a broad overview of machine learning and deep learning. Here, we give a slightly more detailed breakdown of the neural networks relevant to this project.

A neural network is a subset of machine learning that seeks to replicate the behaviors of biological brain function. Human brains perform complex calculations and evaluations through the communication and cooperation of billions of neurons. In a similar way, a neural net relies on a web of artificial neurons to complete tasks. Essentially, a single neuron takes in a number of inputs, then applies an activation function to them, and then produces an output value. Figure 1 shows a diagram of a simple neural network. The inputs are fed through the first

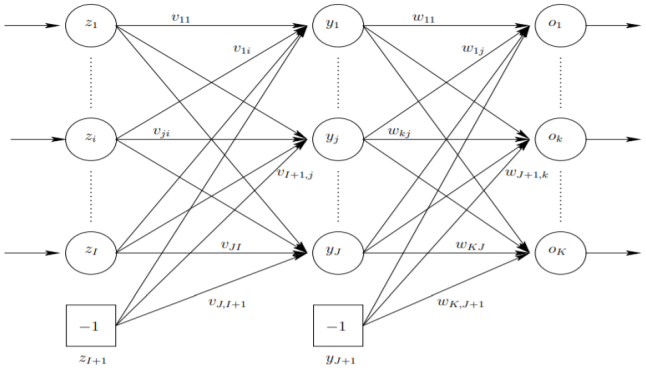

Fig. 1. A simple feed-forward neural network. Taken from [2]

layer of neurons, where a weight value is multiplied to each neuron's output. These weighted values (and the bias value of -1) serve as the inputs for the second layer. The outputs of the second layer are then the outputs of the neural net [2]. The elegance and usefulness of neural networks presents itself in the robustness and scalability of these models. With a neural network, it is easy to increase scale by adding many neurons and layers. Different combinations of neurons per layer can be used, different activation functions can be used throughout, weights between layers can be customized, and more.

One specific subset of neural networks that is used extensively in image classification and object detection is the convolutional neural network (CNN). A CNN is a type of network model that is designed to identify details in images. This is accomplished by carefully arranging layers and neurons to break down an image into a single or a few pixels. These many tiny pieces of the image are then classified, and then the image is reconstructed with the identified object now able to be labeled. This breakdown of the image in these many hidden layers (neural net layers between the input and output layers) is referred to as convolution, hence the name convolutional neural network. Due to the convolution used to detect features, it is not always clear what features a network might use to discern an object. For example, one model used text in advertisement stickers to help detect race cars [9].

Figure 2 shows a sample diagram of a CNN. In this diagram, the network is not fully connected, which means not every neuron in a layer connects to the next layer (or some weights are zero). This reduces computational load, both in pre-training and in real use [9]. This diagram shows the convolutional nature, where the image is broken down and filters are applied, which then results in a two-node output layer. In this model, filters are applied to use the same weights to detect the same class of features across a feature map. Not only does this reduce the amount of parameters, but it also prevents overfitting of the model [9]. Overfitting is a phenomenon that occurs when a model becomes too reliant on test data, and then underperforms in real scenarios. In Figure 2, the max pooling layers choose the highest pixel value in a 2x2 patch. The goal of pooling layers is to reduce the number of parameters needed for layers deeper in the network. This reduces the computation necessary for training or running the network model. Pooling cannot account for large translational or rotational orientation perturbations. This would require more filters and more weight parameters [9].

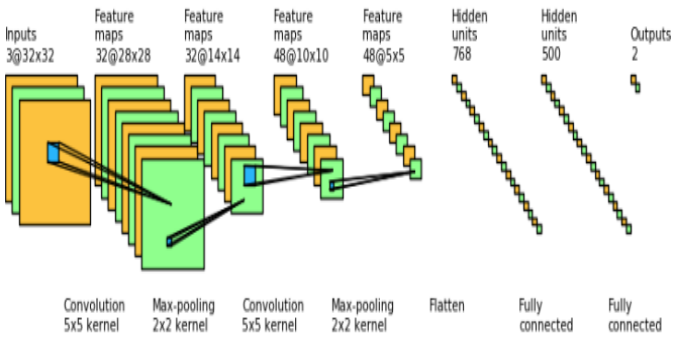

Fig. 2. A representation of a CNN. Taken from [9]

YOLO is built using a version of a CNN where the image is first broken up into a grid. Then, each box in the grid is put through a CNN to get all potencial bounding boxes of objects. These are then combined to get all of the detected bounding boxes in the image. Then all bounding boxes with low confidence are thrown out to get the final predictions [3].

B. Adversarial Attacks

Since machine learning continues to gain popularity in research and real-word applications, it stands to reason that malicious parties are eager to find a way to exploit them. By specifically altering the input to a machine learning system, a model can be tricked to produce an arbitrary output. Such attacks are broken into two categories: white-box attacks, where the adversary has total knowledge of their target's machine learning model, and black-box attacks, where the adversary is only allowed to obtain partial information about the model through queries to the target [7].

Black-box attacks are typically more challenging to launch successfully, but they are more realistic; it is unlikely an attacker will know the complete model of a target. Clarifai and Google Cloud Vision, for example, do not allow access to their full models. Additionally, research has been done to show that black box attacks can, at times, actually be more successful at tricking models than white box attacks [12]. This is because many networks have been hardened against white box attacks by using techniques such as gradual gradients while a black box attack can be trained on an unhardened network to generate adversarial images that are then transfered to trick hardened networks [12]. This technique has been found to be extremely effective, even against networks that are difficult for white box attacks to trick [12].

An example of a black box attack can be seen in [7], where the authors propose a simple black box attack. They describe the task of image classification to be the ability to predict what humans see in an image. Thus, if an imperceptible change is made to an image, the model should not alter its prediction [7]. Exploiting such vulnerabilities will create a successful adversarial attack. A targeted attack aims to change the model's predicted class to a specific class, whereas untargeted attacks simply seek to change the model's prediction to something inaccurate.

For a simple black box attack, the algorithm is fairly straightforward. Random directions of a step size are repeatedly selected and are then tested against the model. The method in [7] takes in the target image's label pair, a step size, and a set of orthonormal candidate vectors. Iterations may need to be modified to save on potentially limited queries, as perturbations are repeatedly applied to trigger a misclassification. The authors evaluated their method with a sample of 1000 images from the ImageNet validation set that are initially correctly classified, and then a target class is uniformly sampled for all targeted attacks [7]. Then, they evaluated their method by attacking the real-world Google Cloud Vision API. The frequencies of their method had to be limited to avoid incurring large costs. Their method gained high success rates in both untargeted and targeted attacks, their accuracy quickly approaching 1.0 as the number of queries increased to the several tens of thousands.

This particular method is simple but has wide applications. [4] used a similar method to generate adversaries to fool R-CNN although [4] was using a white box model. However, many of the adversaries that [4] generated fooled the version of YOLOv4 that is typically shipped, which was trained with the robust COCO dataset.

III. Experiments and Results

For our research, we wanted to test the effectiveness of various methods that have been found to detect adversaries in image classifiers on object detectors. To do this, we created a dataset with 5500 images. 5000 of these images come from the COCO dataset and the other 500 images contained adversaries that came from either the APRICOT dataset (which consists of a wide variety of images in which adversarial patches were printed out and put up in the real world, one per image), were generated by us using the setup from [4], or were from Google Images [10] [1].

A. Gaussian Noise

The first method that we wanted to test was to inject Gaussian noise into the image to see if that would help the classifier ignore adversaries. [6] ran a similar experiment on an image classifer and found that simply adding Gaussian noise to an image basically overwrote adversarial perturbations and allowed the classifier to detect the original label. However, adding Gaussian noise to an image without perturbations did not change the label. This is a simple test so we wanted to see if it can be used in a object detector context to remove adversaries.

To accomplish this, we ran the default YOLOv4 model trained on the COCO dataset against each image and used the labels to count how many false negatives there were (i.e., how many objects were labled but not detected) and how many adveraries were detected. We then added Gaussian noise to the image with variances ranging from .1 to .45 and ran the modified image in the YOLO model to find if it would still detect the adversaries. An example of this is shown in Figure 3 where the top image is what the YOLO model originally detected and the bottom image is what the YOLO model detected after Gaussian noise with .25 variance had been added.

In the 5500 images, there were a total of 36051 true objects and 501 true adversaries. The hope is that the Gaussian Noise injector would reduce the number of adversaries detected without increasing the number of false negatives. The actual results are shown in Table 1.

TABLE I
Results from the Gaussian Noise test

Variance	False Negatives	Adversaries Detected
0	14278	133
.1	14306	133
.15	14287	132
.2	14285	134
.25	14333	136
.3	14314	132
.35	14343	131
.4	14348	131
.45	14338	134

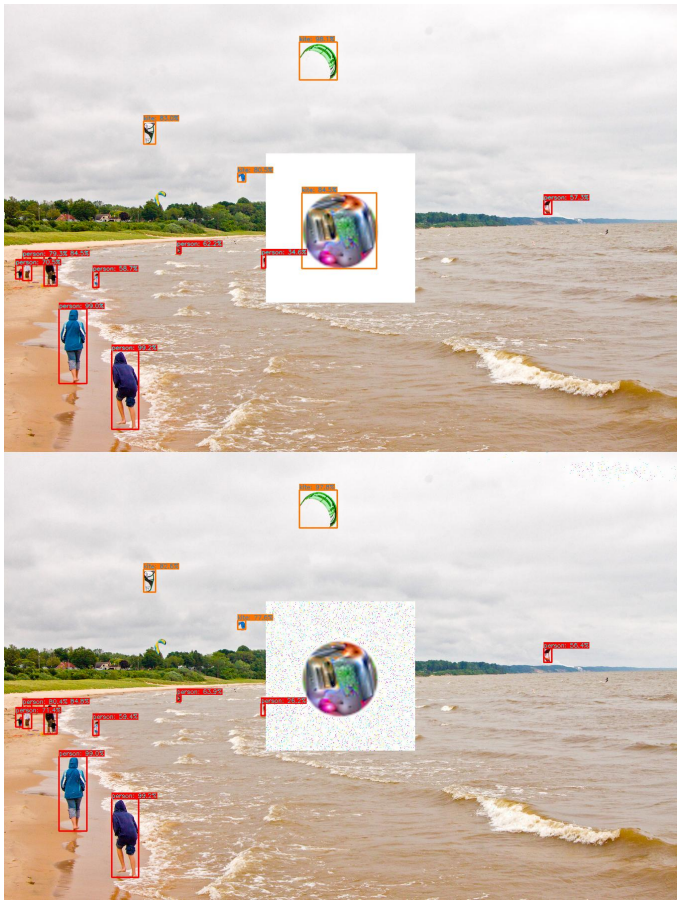

Fig. 3. The YOLO model originally detected an adversary but not after adding the Gaussian noise

From our results, it is clear that Gaussian noise accomplishes very little in the object detection context. This was not completely unexpected because [6] found that Gaussian noise injection worked best to remove adversaries that relied on small perturbations. In contrast, the adversaries that attack object detectors use very large perturbations. We hypothesize that this is why the Gaussian noise does very little to remove adversaries and can even help adversaries in some cases. Because this test also requires an object detector to scan an image twice and slightly increases the number of false negatives, we conclude that there is no practical use for Gaussian noise injectors for an object detector.

B. Statistical Detection

Our second method of detecting adversarial images is done purely through statistics. The work in this section was largely inspired by the work done in [6], with some additional adaptation from [5]. The basis for this component of our project is in using statistical tests to determine differences between given samples. One of the most robust ways to do this is by using a statistical two-sample test, in which two samples are picked from data, and then a certain statistical calculation is performed to discover if the two samples originate from the same distribution.

The first step in this process was obtaining a distribution for our image datasets. To accomplish this, we used slightly different datasets than with the Gaussian noise or the custom YOLO model experiments. The first came from the standard COCO training dataset, from which we randomly picked 500 images. The next was the APRICOT dataset [1] which we also randomly selected 500 images from.

Once we created our datasets, we created a distribution to represent each one. There are several ways to find statistical distributions from images, with the easiest option being to convert the image to grayscale, and then create a histogram of pixel values from 0-255 to get a sense of the intensity distribution of the image. However, since both image sets contain a wide variety of environments, both indoors and outdoors, a simple pixel value distribution would not be very informative. In order to make a more useful distribution, we took inspiration from the authors of [6] who incorporated the features extracted from each image. To do this, we used a feature detector built into OpenCV called ORB (Oriented FAST and Rotated BRIEF). This feature detector identifies and marks shapes and edges in images. We set ORB to identify up to 100000 features. Figure 2 shows an example of ORB applied to an APRICOT [1] adversarial image. All the green dots are features that ORB identified.

Fig. 4. An example of ORB feature detection on an image from APRICOT

After applying ORB to each image in a dataset, we created a pixel value histogram for the dataset. We propose that taking a pixel value histogram after feature detection is applied will be a significant difference than an image that does not have an adversarial patch, because the patch has many shapes that will be picked out by ORB. Figure 5 shows the resulting histogram of the dataset of 500 clean images, and Figure 6 shows the

histogram of the 5oo adversarial images. The large spike 250 in Figure 4 shows that ORB extracted a much higher amount of features due to the adversarial patches.

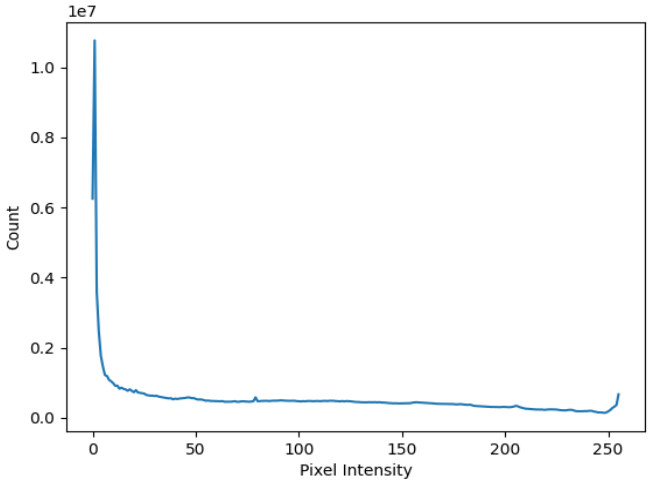

Fig. 5. A histogram of the clean image dataset.

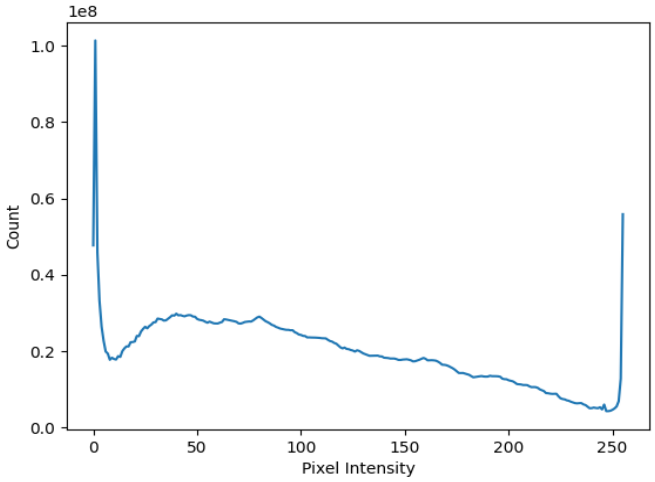

Fig. 6. A histogram of the adversarial image dataset.

After we identified distributions for each dataset, we implemented a two-sample statistical test. Drawing from [6], we implement a statistical test called Maximum Mean Discrepancy (MMD). MMD is a kernel-based test that measures the distance between two distributions. We can use this value to determine if a sample of ORB histogram values lies in the same distribution as a sample taken from clean images. To perform this experiment, we ran the MMD test on a variety of sample sizes. Table 2 shows the results of this test.

When the sample sizes are small, the MMD value rises simply because there is less data. The two-sample test would not be effective on individual images. However, when the sample size begins to rise, it quickly becomes clear that the MMD value is significantly larger when one of the samples is drawn from the adversarial image set. Unfortunately, since our clean dataset includes 500 images, we could not run an MMD

TABLE II
RESULTS FROM THE MMD TEST

Sample 1	Sample 2	Sample Size	MMD Value
Clean COCO	Clean COCO	250	0.00064
Clean COCO	Clean COCO	100	0.05039
Clean COCO	Clean COCO	50	0.13120
Clean COCO	Clean COCO	25	0.28159
Adv APRICOT	Clean COCO	500	0.03125
Adv APRICOT	Clean COCO	250	0.03201
Adv APRICOT	Clean COCO	100	0.08002
Adv APRICOT	Clean COCO	50	0.16000
Adv APICOT	Clean COCO	25	0.31999

test on a sample size of 500. But, when the sample size is 250, the MMD value is significantly larger when one sample is adversarial versus when both samples are clean (0.03201 vs 0.00064). Similar to [6] we conclude that a statistical two-sample test is quite effective at determining a difference between adversarial and clean images, provided the sample size is sufficiently large.

C. Custom YOLO Model

For our final method of detecting adversaries, we attempted to create a custom YOLOv4 model that could give "adversary" as a class output. To train this model, we used the same 5000 random images from the COCO dataset and the 500 adversarial images as the Gaussian noise experiment. We kept the same architecture as the original COCO YOLO model except we added an additional "adversary" class and labeled the images with adversaries in them to reflect this.

Fig. 7. The left image was detected as a teddy bear by the original YOLO model. The right right was correctly detected as an adversary by our custom model. This image was originally generated with [4]

We then trained a YOLOv4 model from scratch for 11100 iterations. We wanted to train it longer but we ran into time and hardware limitations. We found that Google Colab was the most effective method for us to train but, even after three straight days of training, we only reached 11100 iterations. The recommended number of training iterations for 81 classes is 162000. For this reason, our mean average precision (mAP) was only 31.5, far less than the original COCO YOLOv4 model that had a 43.5 mAP.

Because our mAP was so low, we could not definitely test our YOLO model against the standard COCO YOLO model.

Fig. 8. The top image was detected as an umbrella by the original YOLO model. The bottom image was correctly detected as an adversary by our custom model. This image was originally taken from [1]

However, even with our limited test, we had extremely promising preliminary results. In Figure 7, the original YOLOv4 model classified the perturbed stop sign as a teddy bear but even our incomplete model was able to confidently classify it as an adversary. This is also true for Figure 6 where the sticker was classified as an umbrella in the original YOLOv4 model but our model was 100% confident that it was an adversary. Neither of these images were in the our training set that was used to train our model.

While it is disapointing that we were unable to finish training our model in time, it is clear that even our model that has been through less than 10% of the recommeneded iterations is able to classify these adversaries much more accurately than the fully trained YOLO model. Both the perturbed stop sign and the sticker were both classified as the same "adversary" class. Despite the fact that these look so different from each other, the model still had high confidence that they were both adversaries. We believe that with a more rubust training set and more time, YOLO can be trained to detect any adversary that has been created to date.

IV. Conclusion/Future Work

Based on our experiments, it is clear that the Guassian noise test cannot be used by object detectors to detect adversaries. It is simply not accurate and not reliable enough to be worth being used. This is in line with our expectations, since the Gaussian test is primarily used to remove small perturbations, we did not expect it to be particularly successful.

However, our work on statistical adversary detection provided promising results. After running the images through the feature detector, the statistical test was able to find a significantly greater MMD value when testing a clean and an adversarial sample set. Although we did not include it in our table, this test was highly reliable and repeatable; we obtained similar results over multiple runs. Even with this success, the scope of our statistical work was limited; there is still a lot of room for improvement here. One of the easiest avenues for improvement would be implementing multiple different two-sample tests aside from just MMD. Contrasting two or three additional statistical tests would add legitimacy to our findings, and give more data to analyze. Perhaps different tests are more effective with smaller sample sizes.

Another way our work could be expanded is in experimenting with different ways to draw up a distribution for an image dataset. Since images contain such a vast amount of data, there are many ways that statistical analysis could be done. For example, feature extraction could be done per-class of an image classifier, or different feature extraction algorithms could be used. Experimenting with a wider variety of identifying the underlying distributions could open the path for greater accuracy in larger and more varied image sets.

One last potential idea is to use statistical analysis on individually classified objects from an object detector like YOLO. One could crop out eveything but the bound box of a detected object, and then infer a distribution per-class for the cropped images. Then a statistical test could be run to

determine if a specific detected object statistically lies in the same distribution as other objects of the same class. This may help solve the problem of the MMD test needing large sample sizes to be effective.

Additionally, the custom YOLO model was extremely promising. Despite the lack of training, the model was able to correctly classify most adversaries, including ones that fooled the original YOLOv4 model. In the future, someone who has more time and better resources could train a better model using the full COCO and APRICOT data sets as well as even more adversaries taken from Google Images. We believe that a better trained model will easily be able to detect many of the adversaries that could trick the original YOLO model. However, there are potencial drawbacks to this method that the other two methods that we devised do not have.

First, we are concerned that while the custom model could be more accurate when presented with adversaries, it may not be able to detect normal objects as well when adversaries are used in the training set. This was not something that we could test with our model due to the lack of training but needs to be tested on future models. Additionally, this test is not model agnostic like the other two. For this to work, all existing YOLO models would need to be retrained which is expensive and time consuming. Finally, we are concerned that as soon as a robust training dataset that includes adversaries has been created, opponents will create even more clever adversaries that avoid detection from models based on current adversaries. This is also a problem that the statistical model could potencially face but this threat is much more serious with this method due to the high cost of retraining models. The statistical method could be redone much easier with the new adversaries. There is a lot of promise with the custom model approach to detect adversaries, especially if only a single image is given, but there is still a lot of research that must be done into it before it is ready for real world use.

REFERENCES

[1] A. Braunegg, A. Chakraborty, M. Krumdick, N. Lape, S. Leary, K. Manville, E. Merkhofer, L. Strickhart, and M. Walmer, “Apricot: A dataset of physical adversarial attacks on object detection,” arXiv.org, 20-Aug-2020. [Online]. Available: https://arxiv.org/abs/1912.08166. [Accessed: 07-Nov-2021].

[2] A. P. Engelbrecht, Computational intelligence: An introduction. Chichester, West Sussex: Wiley, 2007.

[3] Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. ”Yolov4: Optimal speed and accuracy of object detection.” arXiv preprint arXiv:2004.10934 (2020).

[4] Chen, Shang-Tse, et al. ”Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector.” Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Cham, 2018.

[5] Fan, W., Sun, G., Su, Y. et al. Integration of statistical detector and Gaussian noise injection detector for adversarial example detection in deep neural networks. Multimed Tools Appl 78, 20409–20429 (2019). https://doi.org/10.1007/s11042-019-7353-6

[6] Grosse, Kathrin, et al. ”On the (statistical) detection of adversarial examples.” arXiv preprint arXiv:1702.06280 (2017)

[7] Guo, Chuan, et al. ”Simple black-box adversarial attacks.” International Conference on Machine Learning. PMLR, 2019.

[8] Haifeng Wan, Lei Gao, Manman Su, Qinglong You, Hui Qu, Qirun Sun, ”A Novel Neural Network Model for Traffic Sign Detection and Recognition under Extreme Conditions”, Journal of Sensors, vol. 2021, Article ID 9984787, 16 pages, 2021. https://doi.org/10.1155/2021/9984787

[9] John Murphy. ”An overview of convolutional neural network architectures for deep learning.” Microway Inc (2016).

[10] Lin, Tsung-Yi, et al. ”Microsoft COCO: Common objects in context.” European conference on computer vision. Springer, Cham, 2014.

[11] Lu, Jiajun, et al. ”Standard detectors aren’t (currently) fooled by physical adversarial stop signs.” arXiv preprint arXiv:1710.03337 (2017).

[12] Papernot, Nicolas, et al. ”Practical black-box attacks against machine learning.” Proceedings of the 2017 ACM on Asia conference on computer and communications security. 2017.