# DDDAS Principles to Realize Optimized and Robust Deep Learning Models at the Edge: Insights from Initial Studies

Robert Canady[1], Xingyu Zhou[1], Yogesh Barve[1], Daniel Balasubramanian[1], and Aniruddha Gokhale[1]

Dept of Computer Science, Vanderbilt University, Nashville TN 37203, USA

**Abstract.** Edge computing is an attractive avenue to support low-latency applications including those that leverage deep learning (DL)-based model inferencing. Due to constraints on compute, storage and power at the edge, however, these DL models must be quantized to reduce their footprint while minimizing loss of accuracy. However, DL models and their quantized equivalents are often prone to adversarial attacks requiring them to be made robust against such attacks. The resource constraints at the edge, however, preclude any quantization and robustness design operations directly at the edge. Moreover, the changing dynamics of edge-based computations and resulting concept drifts in the models require an iterative approach to meet the needs of robust DL models at the edge. To address these challenges, this paper presents initial results on an iterative procedure involving a DDDAS feedback loop. DDDAS is used to dynamically instrument the edge-deployed, quantized DL models for data on the effectiveness of their quantization and robustness abilities, which in turn is used to drive an automated, cloud-based process that uses tools, such as Apache TVM, to generate quantized, optimized and robust DL models suitable for the edge. These models subsequently are automatically deployed at the edge using orchestration tools. Preliminary studies using this approach have shown its effectiveness in image classification and object detection applications.

**Keywords:** Deep Learning, Dynamic Data-driven System, Adversarial Machine Learning, Edge Computing, Model quantization

## 1 Introduction

Deep learning (DL) can be used to detect and segment distinct objects in an image, translate speech to text, detect fraud, etc, which has led to a number of DL-based cloud-hosted services. For several reasons, such as low latency response, conservation of bandwidth, security, privacy, environmental concerns, etc., however, applications are being designed to shift most of their computations away from the cloud and closer to the edge. Consequently, there is a need to deploy these traditional DL models on a diverse range of edge hardware from FPGAs to GPUs to ASICs [36]. Moreover, due to capacity and power constraints

of these edge resources, the footprint of these models for edge resources must be reduced – a process known as quantization. Thus, a change in model execution performance and accuracy across these different devices can be expected since these models are typically optimized for the smaller footprint and resource-constrained devices. There have been several efforts, such as Apache TVM [7], Once-for-all (OFA) [4], and knowledge distillation (KD) [14] that attempt to optimize DL models for heterogeneous platform deployment especially at the edge.

Unfortunately, current machine learning models are generally vulnerable to adversarial machine learning (AML) attacks, which are bound-limited perturbations unnoticeable to the human eye but that cause the model to misunderstand the data. While frameworks like TVM, OFA and KD work effectively under normal, non-adversarial conditions, prior studies on determining whether or not the compiler-generated smaller DL models are vulnerable to the same adversarial machine learning (AML) attacks as traditional DL models are generally lacking. Moreover, defense mechanisms against such attacks, e.g., an effective defense strategy such as adversarial training, take much longer to deploy than traditionally trained DL models.

These prohibitive costs make it very difficult to use existing AML defense techniques directly on the edge. To address this problem, we propose a novel application of the DDDAS paradigm [9] to generate, evaluate and deploy robust and optimized edge-based DL models. In our approach, the DDDAS feedback loop manifests between a powerful cloud or fog server and multiple edge-based devices. The end result is a robust and reduced-size DL model that will be deployed on the edge devices. Our novel application of the DDDAS paradigm operates as follows: An initial reduced size and robust DL model is deployed at the edge; then this edge device will periodically stream dynamically instrumented data concerning attack robustness as well as accuracy of model predictions back to the server where several larger models will check the performance and robustness of the edge-based model. Based on these performance results, adaptive retraining of the edge-based model will occur at the cloud server. Subsequently, this new model will be deployed on the edge-device after adversarial retraining and optimization.

In this paper, we lay out the general idea behind our approach and present preliminary results using this approach. In Section 2 we provide the necessary background information. This will be followed by Sections 3, 4 where we describe our approach. We present some of our initial results in Section 5. We then conclude the paper and discuss future directions in Section 6.

## 2 Background and Related Work

To make this paper self-contained, we provide background information on the use of deep learning in computer vision focusing primarily on adversarial machine learning, and on model quantization/optimization. We also describe related efforts and compare them to our proposed ideas.

### 2.1 Edge-based Computer Vision Applications

Edge Computing is the idea of moving the computation from the cloud closer to the sensors or Internet of Things (IoT). One of the benefits of this approach is that it eliminates the need to send data back to the cloud, which can be very costly depending on the type of data.

There have been several recent works, such as OpenDataCam [28], Coral-Pie [34] and DeepLite [15], that demonstrate different object detection applications at the edge and the need for edge accelerators. In Coral-Pie, the application is vehicle tracking using two Raspberry Pi's conected to a Coral USB. The authors did not use the full YOLOv3 [27] for object detection because it was too computationally expensive for the CORAL USB. OpenDataCam is an open source tool for monitoring and tracking moving objects in a live video stream. This application uses YOLOv3 on a desktop machine and recommends using YOLOv3-tiny for edge devices like Jetson Nano.

### 2.2 Adversarial Machine Learning and Defenses

Adversarial machine learning attacks on deep learning is a relatively new field with its start in machine learning models [3]. The work on adversarial evasion attacks [2] led to the seminal work on adversarial work on deep learning models [30]. The idea behind the attacks is that the image that is to be classified is perturbed enough to make the ML model misclassify but not so much that a human observer would notice. Several more efforts followed, such as FGSM [13], PGD [23] and DeepFool [25].

There have been efforts to defend against such attacks. One of the most successful defenses is a proactive method called Adversarial Training [23]. The idea is to augment the training with adversarial examples so that the model will hopefully learn smoothened decision boundaries taking into account the adversarial perturbations, and later can correctly classify the adversarial samples. There have been other defense works that utilize data augmentation [22] and/or pre-processing [33] [26] where the idea is to remove the perturbations from the image, or at least to mitigate the impact of the perturbation. A combination of these data transformations with adversarial training [32] [1] have also been proposed as a set of defense strategies.

Generally speaking, adversarial machine learning attacks and defenses have gained much research interest. A roadmap on improving and evaluating adversarial examples was given in [6], where the authors outline an approach to test the robustness of models and describe some of the usual pitfalls that can occur with defenses.

### 2.3 Deep Learning Computer Vision Attacks and Defenses

Since we focus on computer vision DL models with the goal of making them robust for the edge, we provide some background in this area. Computer vision is a very large area including many kinds of tasks like detection, classification

and segmentation. Among them, object detection is a quite fundamental one. Object detection in computer vision can be broken into two categories: Single shot and Two-stage. Two-stage object detectors like R-CNN [12] have a region proposal network in the first stage that narrows down the number of Region of Interests (ROIs), and in the second stage completes the classification and refines the bounding box. Two-stage models achieve good performances but training and inference are both expensive. Single shot detectors like SSD models [20] predict the boundary box and the class at the same time. Single shot detectors often trade accuracy (on objects too close or far away) for the inference speed.

Due to the rise in the number of applications using deep learning-based object detectors, research has focused on attacks against object detectors in a direction guided by the previous adversarial machine learning works. The purpose of these attacks can be different, ranging from causing many false objects to be detected like adding an adversarial patch [21] to not detecting any objects at all as in TOG [8].

Another recent extended attack on a tracking application was presented in [17] in which the authors try to fool the Multiple Object Tracking (MOT). They present an early work in autonomous driving to explore an attack on the complete computer vision pipeline. Since adversarial ML attacks on object detectors is still a relatively new field, the research on defense techniques is still scarce. Efforts, such as [16], have used a two-stage adversarial training algorithm to improve the robustness in safety-critical scenarios. In [35] the authors present a similar adversarial training approach with a model trained on PGD attacked data. They however only use one type of attack to augment their training data.

### 2.4 Summary of Prior Efforts and Unresolved Challenges

Much work has been done to address the challenges of adversarial machine learning and deploying models on edge-based devices, but little work has been done on evaluating the robustness of edge-based models before and after optimization and as models incur concept drift. We posit that the adversarial robustness challenges, particularly on edge hardware deployments, requires further investigations into the following questions, which formulates the need to apply the DDDAS paradigm as discussed in this paper:

1. Although research on model optimization with adversarial robustness exists, these are mostly input-dependent solutions raising the question whether such settings are too ideal without considering realistic resource limitations?
2. Past research has focused heavily on the single computation device node adversarial vulnerability raising the question whether more adaptive system-level attack evaluations can be designed?
3. The high transferability of the adversary across model settings raises the question whether models can be deployed at the edge in a resilient way to mitigate potential adversarial risks for new edge-based data-driven applications?

Given these unresolved challenges, our objective is to explore solutions from a system-level perspective. To make this system robust and resilient, we are adopting DDDAS principles because the DDDAS's dynamic and adaptive feedback loop can ensure that the ML models never start to decrease in performance nor lose their robustness.

## 3 Methodology

In this section, we present our work. First, we describe the overall system model and then discuss the approach highlighting the DDDAS loop and its components that are distributed across the edge and cloud.

### 3.1 System Model

The system consists of one or more edge devices and a cloud server. In this scenario, the edge device is assumed to run an application such as surveillance of a parking lot or aiding an augmented reality (AR) device that is giving real-time guidance to a user. It will be directly connected to a camera or any other type of sensor being used.

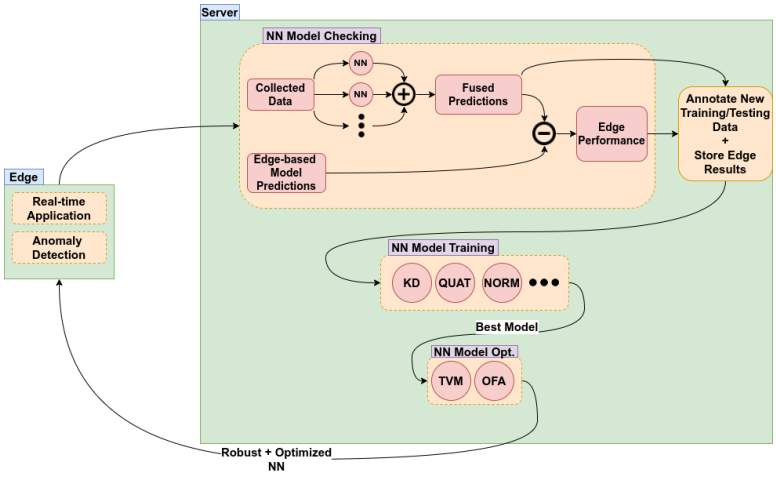


Fig. 1: System Model with the DDDAS Feedback Loop

### 3.2 Server side

As seen in Figure 1, the server is used to train the original model that is to be deployed to an edge device. This is also where multiple larger networks are deployed that have been trained using different architectures, defense methods,

etc. The goal is to cover enough 'adversarial' ground with multiple networks that a fusion of all of the models cannot be fooled by whatever perturbations are crafted.

There have been many research works on machine learning model optimizations for specific hardware, and in particular smaller edge-based devices [29]. Among them we specifically choose three critical techniques of Once-for-all (OFA) [4], Apache TVM [7] and Knowledge Distillation (KD) [14]. These three techniques involve three model phases of adversarial training, defensive model optimization and robustness-preservation model structure simplification. OFA [4] trains one network, and then uses a generalized pruning approach to obtain many smaller networks that have reduced dimensions in depth, width, kernel size, and resolution. Apache TVM [7] is a deep learning optimizer and compiler, that takes in models trained using frameworks such as Tensorflow, PyTorch, MXNet, etc. and generates code optimized to run the models on diverse hardware backends. KD [14] utilizes a teacher-student approach, where the teacher is a large model and the student is a smaller model. In our framework, the candidate models go through Apache TVM or OFA where they are optimized for the given device.

We emphasize the *dynamic data-driven* aspect by continuously checking prediction results on incoming data from the edge side in a dynamic way. The server takes in data from the edge device and passes it through multiple networks, where we then fuse those results and compare them to the edge-device model prediction. From the fused results, we create new ground truth training and validation images. These images can then either be used directly, or they can be attacked and used for adversarial training. Using terminology from Knowledge Distillation (KD) [14], while all of this is going on, an edge model is continuously fine-tuned with the newly annotated data while a student model is distilling knowledge using the fused results as the teacher model. When the edge device drops below a certain performance threshold, one of the updated models is chosen to be deployed.

### 3.3 Edge side

The edge side is where the application is actually executing. It is continuously collecting the streaming data which is passed through the quamtized and optimized network. These results are saved and checked for any potential anomalies. If there are any, then this triggers a certain amount of data and predictions to be sent back to the server. Periodically, data and predictions are sent back for model checking, where a trigger causes more data to be sent back.

We emphasize the 'dynamic data-driven' view from two aspects. First of all, the periodic prediction result checking and calibration with the server side enables the 'dynamic' model updating to guarantee consistent adversarial robustness. Secondly, the feedback information from the server side should also enable potential threat type detection and estimation(for example which $L_p$ norm attack), leading to 'dynamic' selection and execution of robust candidate models.

### 3.4 Expected Use Cases

In ongoing work we are applying these ideas to augmented reality edge applications used to provide interactive maintenance support. We expect this framework to allow users to deploy applications to the edge that can then be dynamically adapted during deployment to perform most optimally when faced with clean or adversarial data. We combine multiple ideas to obtain the most robust, optimized edge-based DL models.

## 4 Experimental Setup

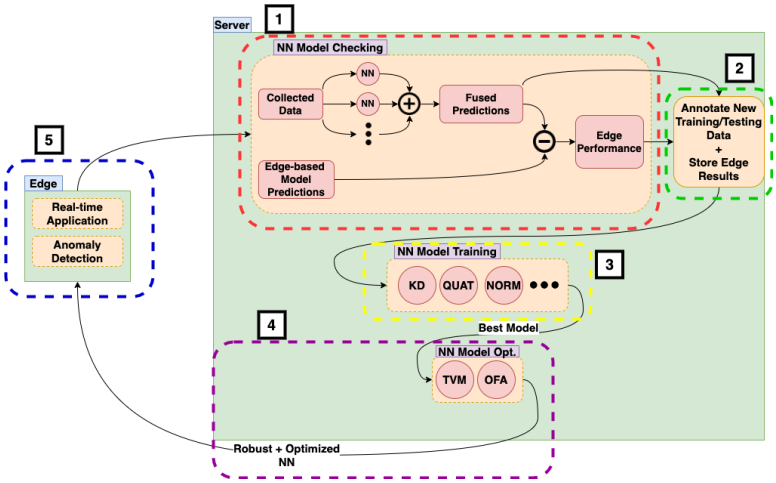


Fig. 2: System Model with the DDDAS Feedback Loop

In this section, we will explore each part of the DDDAS feedback loop and how we plan to implement and evaluate each component. To more easily visualize the distinct parts of the loop, we have highlighted the parts in Figure 2.

### 4.1 Feedback Loop Components

To begin testing the proposed DDDAS feedback loop from Figure 1, there were several experimental logistical issues that we needed to resolve. The first was determining how can we use pre-existing datasets to test out our feedback loop with 'new' data. Our initial idea was to take datasets and reduce the number of training images, and use the extra training images for the feedback loop. These could then be combined with the validation and test sets to simulate new data being presented. This data would then be used later on in the retraining phase of the loop.

We want to split the data so that there are examples from each class in the new data. We also needed to separate some data for the anomaly detection. For example, we would want an anomaly to be detected for object detection if there are too few or too many detected objects. We discovered a tool [24] that would aid us in doing this. This tool allows the user to more easily visualize and rearrange datasets for their needs.

**NN Model Checking** The first component we will discuss is how we evaluate the edge model's performance. There has been some research done on fusion and its impact on robustness, which has mostly shown that fusion does improve robustness [31]. We want to utilize this fact to evaluate the edge model.

To setup this component, we use pretrained networks, which are adversarially trained networks as well as a variety of different neural networks. The idea is to get a diverse enough ensemble of models that when fused together will not be fooled by adversarial examples, OOD data, etc. When we say fused together, we mean by utilizing decision-level fusion. Decision-level fusion is where the output probabilities from ML models trained on different modalities of data are combined in order to achieve better performance than from just one type of data. There are several different types of decision-level fusion: average ranks, naive-bayes, highest probability, generalized chernoff, and sandia probabilistic fusion. Each algorithm, makes decisions slightly differently based on the probability distributions.

We directly use the robust decisions to determine how our edge model is performing. This is done by comparing the fused results with the edge model results. We have a threshold set, that has been determined through testing, to determine when the edge model's performance has dipped enough to warrant retraining.

**Preparation of New Data and History Tracking** We discussed briefly how we plan to approach sending 'new' data through a network for evaluation and retraining. A benefit of splitting up the data more than just train and test splits, is that we have already annotated data. This way we can check how well our annotation techniques work without having to hand annotate new data.

In this component, we use the predictions from the NN model check to help us annotate the new images. We also store the results of the models to be able to compare performance going forward.

**NN Model Training/Retraining** At this point, new data has been annotated and there is a need to retrain the edge model. We utilize several different approaches to determine the best and most robust model. Several of the different training techniques we are using are our own QUAT [5] approach, normal training, knowledge distillation training, regular adversarial training, etc. We then compare each of these models to each other and send the best one to the next step of the feedback loop.

**NN Model Optimization** In this component, we optimize the trained model to be deployed on a specified edge device using preexisting frameworks such as Apache TVM. This step will optimize the model to be deployed on whatever hardware is being used. There has been some previous work showing that model quantization can actually improve adversarial robustness [37].

**Real-time Edge Application** This component is where the real-time edge application will be deployed. At the moment we are focusing solely on computer vision tasks like image classification, object detection, semantic segmentation. With one of our motivating applications being AI-assisted AR for smart maintenance, we would like to include other tasks such as natural language processing in further work.

While the application is being run, there is periodic offloading of some of the collected data as well as the model's predictions. There also is an anomaly detector running that checks the model's ouputs at runtime and checks to see if there is any anomalous behavior. An example of this for object detection would be detecting many more objects than is normal or not detecting any objects for an extended period of time.

To aid in decision-making further on in the loop, we also collect resource usage data. We see this mainly being used to help in the optimizations of the models.

### 4.2 Datasets

We want to evaluate our framework using several computer vision tasks; image classification, object detection, and semantic segmentation. To carry out the best and most informative evaluation, we wanted to select several datasets for each task.

In Table 1, there is a dataset that we collected ourselves, Car Engines. The goal for this dataset was to show a proof of concept for engine maintenance guidance using ML. This dataset was collected with help from collaborators in our research group. The dataset was then self-annotated for semantic segmentation. An example image and segmentation map from this dataset can be seen in Figure 3.

### 4.3 Models

The models we are using and their deployment level are each outlined in Table 2. We also present the model size to illustrate the differences between cloud and edge models.

| CV Task | Dataset | Train/Val/Test Images | Classes |
|---|---|---|---|
| Image Classification | CIFAR10 [18] | 60,000 | 10 |
| | CIFAR100 | 60,000 | 100/20 |
| | Imagenet [10] | 14,197,122 | 1000 |
| Object Detection | Pascal VOC [11] | 21,493 | 20 |
| | MS COCO [19] | 123,287 | 80 |
| | VisDrone [38] | 8632 | 10 |
| Semantic Segmentation | Pascal VOC | 123,287 | 20 |
| | Car Engines (Ours) | 58 | 14 |

Table 1: Description of CV Tasks and Corresponding Datasets

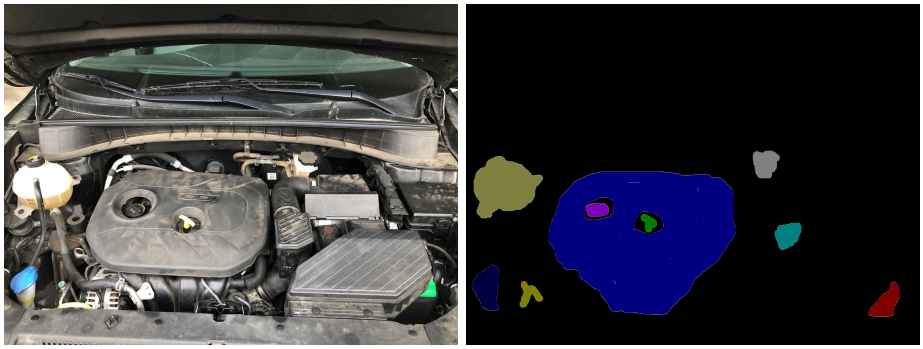

Fig. 3: Example Engine Images (L: RGB Image, R: Segmentation Map)

### 4.4 Devices

In Table 3, we list out the devices we have used/plan to use for evaluation of our feedback loop. We have also included selected resource information such as CPU cores, RAM, GPU, and typical power consumption when idle and busy.

## 5 Preliminary Studies and Challenges

Typically accuracy of the model is the only metric for image classification. Object detection and semantic segmentation, have slightly more complex metrics.

To determine how well models perform on object detection/semantic segmentation datasets like PASCAL-VOC [11], they are judged on their inference time and their mean average precision (mAP). The mAP is calculated using a metric called *Intersection over Union (IoU)*. The higher the mAP the better, but its semantics for object detection are different compared to image classification accuracy, where the classification is either correct or incorrect. In contrast, the goal of object detection is to draw bounding boxes around objects and then correctly classify the object(s). To calculate the mAP, the analyst needs the ground-truth

| CV Task | Model | Backbone | Model Size (MB) | Device Type |
|---|---|---|---|---|
| Image Classification | Resnet(18/50) | N/A | 45/98 | Cloud |
| | Densenet121 | N/A | 33 | Cloud |
| | Mobilenet v3 | N/A | 16 | Edge |
| Object Detection | YOLO v3 | Darknet | 237 | Cloud |
| | Tiny YOLO v3 | Darknet | 34 | Edge |
| | FasterRCNN | Resnet50 | 160 | Cloud |
| | FasterRCNN | Mobilenet v3 | 74 | Edge |
| Semantic Segmentation | Unet | Resnet50 | 164 | Cloud |
| | Unet | Mobilenet v3 | 43 | Edge |

Table 2: Description of CV Models

| **Deployment Level** | **Device Name** | **CPU** | **RAM** | **GPU** | **Power** |
|---|---|---|---|---|---|
| Cloud | Desktop | 12 Core | 32 GB | NVIDIA RTX 2060 | 70W/175W<br>175W/500W |
| Edge | Raspberry Pi | 4 Core | 1 GB | N/A | 1.9W/5W |
| | Jetson Nano | 4 Core | 4 GB shared | NVIDIA Maxwell | 5W/10W |
| | Jetson TX2 | 2 + 4 Core | 8 GB shared | NVIDIA Pascal | 7.5W/15W |

Table 3: Description of Devices Used

and predicted bounding box coordinates, which can then be used to calculate the IoU. The IoU is calculated as the amount the predicted bounding box overlaps with the ground-truth bounding box divided by the total area of the union of both boxes.

To determine the efficacy of the model, the analyst sets a threshold percentage for the overlap. The threshold is usually set at 0.5 per convention and because of the fact that humans can barely tell the difference between 0.3 and 0.5 IoU. For some different datasets or competitions, a different confidence threshold is used. The mAP is then calculated by drawing precision-recall curves with the IoU set at different thresholds. This is done for each class, and at this point it is just the average precision (AP). The average AP across all classes is then the mAP.

### 5.1 Image Classification

We have done extensive evaluation of image classification models. Most of this is still waiting to be publicly released, but that will all be included in the final framework. We have looked at different techniques for adversarial training as well as potential preprocessing techniques for improving adversarial robustness.

### 5.2 Object Detection

In our earlier work, we evaluated our QUAT [5] algorithm using object detection models. Through experimentation, our approach appeared to be a promising solution, but there is still room for improvement as well as more evaluation which we plan to do throughout developing this framework.

### 5.3 Semantic Segmentation

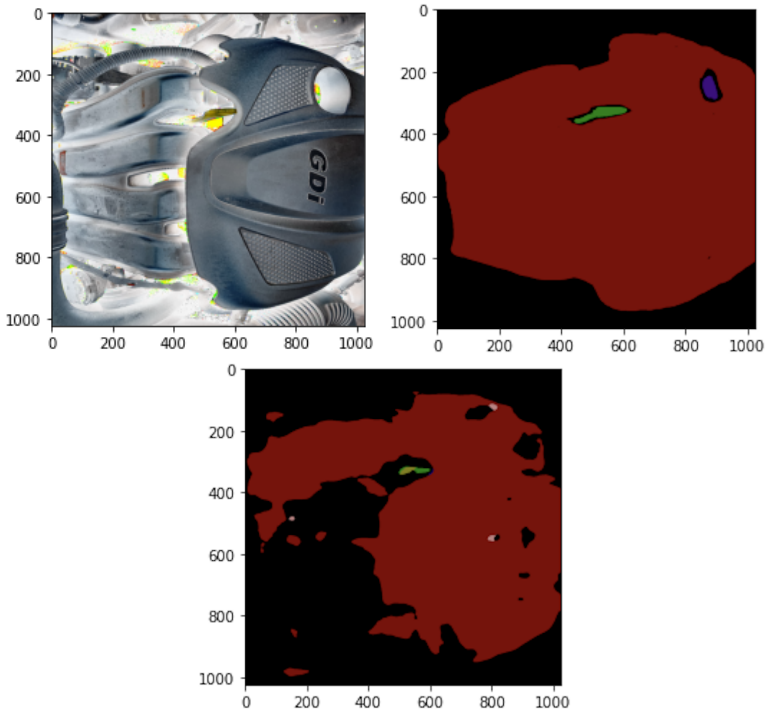


Fig. 4: Example Engine Images
(Top left: Normalized RGB, Top Right: Test seg, Bottom: Predicted seg)
(mIoU = 30.7%, Accuracy = 79.3%)

Some of our initial results on the Car Engine dataset are shown in Figure 4. While the mIoU might seem slightly low at 30.7%, it is actually not bad for a semantic segmentation task. The model was able to locate close to 3/4 of the engine as well as the dip stick. Also, the validation accuracy of 79.3% is promising showing that the model was able to classify the predicted objects reasonably well.

We believe these results can be further improved by training the model more as well as add more engine images.

## 6 Conclusions

This paper presented preliminary ideas and results on applying DDDAS principles to address the challenges of realizing adversarially robust deep learning models that are suitable for edge devices. Presently, this research has shown the effectiveness of using a DDDAS feedback loop to keep a real-time application from decreasing in performance and improve in robustness to unseen circumstances. There is still much work to be done to further explore this area of research. We plan to explore this problem by utilizing applications such as semantic segmentation and object detection that will be evaluated on a range of edge device types and application use cases, such as augmented reality-based maintenance.

## References

1. Addepalli, S., B.S., V., Baburaj, A., Sriramanan, G., Babu, R.V.: Towards achieving adversarial robustness by enforcing feature consistency across bit planes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
2. Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., Roli, F.: Evasion attacks against machine learning at test time. CoRR **abs/1708.06131** (2017), http://arxiv.org/abs/1708.06131
3. Biggio, B., Roli, F.: Wild patterns: Ten years after the rise of adversarial machine learning. CoRR **abs/1712.03141** (2017), http://arxiv.org/abs/1712.03141
4. Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S.: Once for all: Train one network and specialize it for efficient deployment. In: International Conference on Learning Representations (2020), https://arxiv.org/pdf/1908.09791.pdf
5. Canady, R., Zhou, X., Barve, Y., Balasubramanian, D., Gokhale, A.: Adversarially robust edge-based object detection for assuredly autonomous systems. In: 2022 IEEE International Conference on Assured Autonomy (ICAA). pp. 97–106 (2022). https://doi.org/10.1109/ICAA52185.2022.00021
6. Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I.J., Madry, A., Kurakin, A.: On evaluating adversarial robustness. CoRR **abs/1902.06705** (2019), http://arxiv.org/abs/1902.06705
7. Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Shen, H., Cowan, M., Wang, L., Hu, Y., Ceze, L., Guestrin, C., Krishnamurthy, A.: TVM: An automated End-to-End optimizing compiler for deep learning. In: 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). pp. 578–594. USENIX Association, Carlsbad, CA (Oct 2018), https://www.usenix.org/conference/osdi18/presentation/chen
8. Chow, K.H., Liu, L., Loper, M., Bae, J., Emre Gursoy, M., Truex, S., Wei, W., Wu, Y.: Adversarial objectness gradient attacks in real-time object detection systems. In: IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications. pp. 263–272. IEEE (2020)
9. Darema, F., Blasch, E., Ravela, S., Aved, A.: Dynamic Data Driven Applications Systems: Third International Conference, DDDAS 2020, Boston, MA, USA, October 2-4, 2020, Proceedings, vol. 12312. Springer Nature (2020)

10. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848
11. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision **88**(2), 303–338 (Jun 2010)
12. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. CoRR **abs/1311.2524** (2013), http://arxiv.org/abs/1311.2524
13. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2015)
14. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NIPS Deep Learning and Representation Learning Workshop (2015), http://arxiv.org/abs/1503.02531
15. Ho, D.H., Marri, R., Rella, S., Lee, Y.: Deeplite: Real-time deep learning framework for neighborhood analysis. In: 2019 IEEE International Conference on Big Data (Big Data). pp. 5673–5678 (2019). https://doi.org/10.1109/BigData47090.2019.9005651
16. Hu, Z., Zhong, Z.: Towards practical robustness improvement for object detection in safety-critical scenarios. In: Wang, G., Ciptadi, A., Ahmadzadeh, A. (eds.) Deployable Machine Learning for Security Defense. pp. 66–83. Springer International Publishing, Cham (2020)
17. Jia, Y., Lu, Y., Shen, J., Chen, Q.A., Chen, H., Zhong, Z., Wei, T.: Fooling detection alone is not enough: Adversarial attack against multiple object tracking. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=rJl31TNYPr
18. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
19. Lin, T., Maire, M., Belongie, S.J., Bourdev, L.D., Girshick, R.B., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. CoRR **abs/1405.0312** (2014), http://arxiv.org/abs/1405.0312
20. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., Berg, A.C.: SSD: single shot multibox detector. CoRR **abs/1512.02325** (2015), http://arxiv.org/abs/1512.02325
21. Liu, X., Yang, H., Song, L., Li, H., Chen, Y.: Dpatch: Attacking object detectors with adversarial patches. CoRR **abs/1806.02299** (2018), http://arxiv.org/abs/1806.02299
22. Lopes, R.G., Yin, D., Poole, B., Gilmer, J., Cubuk, E.D.: Improving robustness without sacrificing accuracy with patch gaussian augmentation. CoRR **abs/1906.02611** (2019), http://arxiv.org/abs/1906.02611
23. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks (2019)
24. Moore, B.E., Corso, J.J.: Fiftyone. GitHub. Note: https://github.com/voxel51/fiftyone (2020)
25. Moosavi-Dezfooli, S., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. CoRR **abs/1511.04599** (2015), http://arxiv.org/abs/1511.04599
26. Raff, E., Sylvester, J., Forsyth, S., McLean, M.: Barrage of random transforms for adversarially robust defense. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)

27. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. CoRR **abs/1804.02767** (2018), http://arxiv.org/abs/1804.02767
28. Sawadski, V.: Opendatacam, https://github.com/opendatacam/opendatacam
29. Sze, V., Chen, Y.H., Yang, T.J., Emer, J.S.: Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE **105**(12), 2295–2329 (2017)
30. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
31. Wang, S., Wu, T., Chakrabarti, A., Vorobeychik, Y.: Adversarial robustness of deep sensor fusion models. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 2387–2396 (January 2022)
32. Xie, C., Wu, Y., v. d. Maaten, L., Yuille, A.L., He, K.: Feature denoising for improving adversarial robustness. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 501–509 (2019). https://doi.org/10.1109/CVPR.2019.00059
33. Xie, C., Wang, J., Zhang, Z., Ren, Z., Yuille, A.L.: Mitigating adversarial effects through randomization. CoRR **abs/1711.01991** (2017), http://arxiv.org/abs/1711.01991
34. Xu, Z., Shah, H.S., Ramachandran, U.: Coral-pie: A geo-distributed edge-compute solution for space-time vehicle tracking. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3423211.3425686, https://doi.org/10.1145/3423211.3425686
35. Zhang, H., Wang, J.: Towards adversarially robust object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019)
36. Zhou, X., Canady, R., Bao, S., Gokhale, A.: Cost-effective hardware accelerator recommendation for edge computing. In: 3rd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 20) (2020)
37. Zhou, X., Canady, R., Li, Y., Bao, S., Barve, Y., Balasubramanian, D., Gokhale, A.: Guarding against universal adversarial perturbations in data-driven cloud/edge services. In: 2022 IEEE International Conference on Cloud Engineering (IC2E). pp. 233–244 (2022). https://doi.org/10.1109/IC2E55432.2022.00032
38. Zhu, P., Wen, L., Du, D., Bian, X., Fan, H., Hu, Q., Ling, H.: Detection and tracking meet drones challenge. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2021). https://doi.org/10.1109/TPAMI.2021.3119563