Efficient and Adversarially Robust Object Detection

Anton Liu
Dept. of Computer Science, Western University
aliu467@uwo.ca
Supervisor: Professor Roberto Solis-Oba

Abstract

The abstract goes here.

I. Introduction

Object detection, one of the fundamental tasks in computer vision, is used in numerous real-world applications by enabling machines to identify and locate objects within images or videos. Object detection algorithms aim to address two challenges in recognizing objects of interest within complex scenes: where (identifying their spatial extent with bounding boxes) and what (assigning corresponding class labels). This capability is crucial for a wide range of tasks, including pedestrian detection for autonomous driving, object tracking in video surveillance, and product recognition for inventory management. Moreover, object detection serves as a foundational building block for higher-level computer vision tasks such as scene understanding, semantic segmentation, and action recognition, facilitating deeper insights and decision-making capabilities in intelligent systems.

The emergence of deep learning revolutionized object detection, leading to significant advancements in accuracy and efficiency. Convolutional Neural Networks (CNNs) have demonstrated remarkable potential and have been the main research direction in recent years [1]. In safety-critical applications like autonomous driving, the performance of object detection systems can have severe consequences, jeopardizing human safety and privacy. The fatal accident involving an Uber self-driving car in 2018 illustrates the severe consequences that failures in object detection systems can have. While the performance of deep learning-based object detection models has significantly improved, their vulnerability to adversarial attacks poses a significant challenge to their reliability and security [2]. Adversarial attacks involve maliciously crafted perturbations to input data, which can cause deep neural networks to make incorrect predictions or fail to detect objects altogether [3]. The counter the attacks, adversarial defence is a critical area of research focused on enhancing the robustness of deep learning models against adversarial attacks. While deep learning models are typically optimized to maximize performance metrics on clean images, adversarial training aims to increase performance in adversarial scenarios [4]. Despite the importance of adversarial defence techniques, there is a noticeable discrepancy in the number of papers focusing on adversarial attacks compared to defences in the context of object detection [5]. This asymmetry highlights the need for more research efforts dedicated to adversarial defence in object detection, to mitigate the growing threat posed by adversarial attacks and enhance the robustness of deep learning models in practical settings.

Despite their recent success, deep learning methods are computationally intensive during both training and inference due to the need to process large volumes of data through multiple layers. Training requires

updating millions of parameters through backpropagation, while inference demands rapid calculations to generate predictions [6]. To handle their Full Self-Driving (FSD) processing, Tesla install high-performance computing platforms capable of 50 trillion operations per second (TOPS) in their vehicles. This custom chip enables complex CNN models to execute while maintaining the low latency necessary for safe autonomous driving. Therefore, as the models get larger and more complex, efficiency becomes an increasing concern. Efficiency can be measured in terms of reductions in computational resources, such as memory usage, processing power, and energy consumption. In this work, the efficiency of the model will be measured in inference time, which is advantageous because it directly reflects the model's performance in real-world applications, particularly those requiring real-time or low-latency processing. Inference time indicates how quickly a trained model can make predictions, which is critical for tasks like autonomous driving, video-based object detection, or any time-sensitive application. A model with shorter inference time can handle more data in a given period, improving overall throughput and user experience.

Various methods can be employed to enhance CNN efficiency, such as transfer learning, quantization, weight sharing, and pruning. Pruning is a key technique for improving efficiency and comes in two forms: unstructured pruning and structured pruning. Unstructured pruning removes individual weights that contribute little to the network's performance, leading to sparser weight matrices. While effective in reducing the number of parameters, unstructured pruning can result in irregular memory access patterns, making it harder to optimize on certain hardware. In contrast, structured pruning removes entire filters/kernels, channels, or layers, resulting in a more compact architecture. This type of pruning not only reduces the parameter count but also decreases the computational load, making it more suitable for real-time deployment on hardware like GPUs or mobile devices. Structured pruning maintains the integrity of the network structure, allowing for faster inference times while maintaining accuracy.

While adversarial robustness and efficiency have individually been explored in CNNs, limited research combines these two areas in object detection. This work aims to bridge this gap by adapting adversarial robustness techniques to efficient object detection frameworks.

II. Background

A. Object Detection

To train a CNN model for object detection, a large labeled dataset is required, where each image contains annotations for the objects present, including their class labels and bounding box coordinates. The model is trained by passing input images through a network that extracts features and learns patterns, with the help of techniques like backpropagation and gradient descent to minimize the error between the predicted and true labels. During training, the CNN learns to detect objects by adjusting its weights to improve both classification accuracy and localization precision, ultimately producing a model capable of accurately identifying and locating objects in unseen images.

1) Model: CNN models for object detection generally consist of a feature extraction backbone and a detection head. The backbone, often a pre-trained deep CNN such as ResNet or VGG, extracts high-level features from images, which are then fed into the detection head to perform localization and classification tasks. Some models, like Faster R-CNN, use a region proposal mechanism, while others like SSD (Single Shot Multibox Detector) predict bounding boxes and class labels in a single pass. YOLO (You Only Look Once) models take a unique approach by dividing the image into a grid and predicting bounding boxes and class probabilities simultaneously, making them highly efficient for real-time applications. YOLOv3,

in particular, improves upon earlier versions with better detection at multiple scales, making it an excellent model for balancing speed and accuracy in object detection tasks.

2) Dataset: Datasets play a pivotal role in advancing the field of object detection by providing annotated images for model training and evaluation. Many openly available datasets exist online, such as Pascal VOC, ImageNet, and Open Images [7]. Among these datasets, the Microsoft Common Objects in Context (MS COCO) dataset stands out as one of the most comprehensive and widely used benchmarks [8]. Its superiority stems from several factors. Firstly, COCO has a diverse range of object categories, encompassing common everyday objects across 80 distinct classes, including people, animals, vehicles, and household items. This diversity ensures that models trained on COCO generalize well to a wide array of real-world scenarios. Secondly, each image in the dataset is annotated with instance-level bounding box coordinates, and class labels for precise training and evaluation of detection algorithms. Additionally, COCO provides a large-scale dataset comprising over 200,000 images split into training, validation, and test sets, facilitating robust model training and unbiased evaluation [8].

B. Adversarial attack

Adversarial attacks are techniques used to manipulate machine learning models, especially neural networks, by introducing small, often imperceptible perturbations to inputs that cause the model to make incorrect predictions. These attacks reveal vulnerabilities in applications from autonomous driving to facial recognition and medical diagnosis. Several methods exist for generating adversarial attacks, each varying in complexity and impact. Fast Gradient Sign Method (FGSM) is a straightforward single-step attack that adjusts the input in the direction of the gradient, making it a fast way to create adversarial examples. The Carlini & Wagner (C&W) attack is a more complex, optimization-based method known for creating highly effective but subtle perturbations. Projected Gradient Descent (PGD), an iterative and robust approach, refines the FGSM method by taking multiple, bounded steps in the direction of the gradient, projecting each step to remain within a specified range of the original input. This makes PGD particularly powerful for generating stable adversarial examples, making it the chosen attack method for this work.

C. Efficient Adversarial Robustness

Many papers have attempted to tackle the problem of making CNNs robust to adversarial attacks, and many others focused on making CNNs more efficient. However, not many combined the two important topics and developed CNNs that are both efficient and adversarially robust. Vaddadi et al. showcased an efficient CNN model optimized for adversarial robustness, achieving high classification accuracy and effective resistance to adversarial samples. Another work from Wijayanto et al. explored the vulnerability of CNNs, particularly compressed models used in mobile devices, and investigated methods to enhance their robustness without sacrificing accuracy. Ye et al. presented a framework that combines adversarial training with weight pruning to achieve model compression without sacrificing robustness, addressing the challenge of maintaining both properties simultaneously. Furthermore, Gui et al. introduced an Adversarially Trained Model Compression (ATMC) framework, integrating pruning, factorization, and quantization within a unified optimization structure to achieve compact, adversarially robust models without significant accuracy loss.

1) Towards Compact and Robust Deep Neural Networks: Introduced by Sehwag et al. in 2019, the work ”Towards Compact and Robust Deep Neural Networks” evaluated the robustness of CNNs under both

structured and unstructured pruning. The authors provide a formal definition of the pruning procedure, encompassing pre-training, weight pruning, and fine-tuning, which clarifies the methodology's effectiveness in achieving compact networks without compromising performance. Empirical results demonstrate that the proposed method can maintain, on average, 93% benign accuracy (percentage of correctly classified non-modified images), 92.5% empirical robust accuracy (percentage of correctly classified adversarial examples), and 85% verifiable robust accuracy (accuracy from verifiable robust training objectives) while achieving a compression ratio of 10×.Their experimental data showed that their proposed method is effective for both pruning methods, but especially for unstructured pruning.

2) HYDRA: In a follow-up paper, HYDRA: Pruning Adversarially Robust Neural Networks, is a seminal work that has taken another step towards improving the performance of machine learning models. Sehwag et al. developed a pruning method that is aware of the robust training objective. This is achieved by formulating the pruning process as an empirical risk minimization (ERM) problem combined with a robust training objective, which is solved efficiently using Stochastic Gradient Descent (SGD). The authors also introduced importance score-based optimization, with every weight being initialized with an importance value that is proportional to pre-trained network weights. This change has been shown to help with the final performance and speed of SGD convergence over random initialization. Extensive experiments were performed by testing with CIFAR-10, SVHN, and ImageNet datasets, and with four robust training techniques: iterative adversarial training, randomized smoothing, MixTrain, and CROWN-IBP. HYDRA achieves state-of-the-art performance in both benign and robust accuracy, even at high pruning ratios (up to 99%). The method shows significant gains in robust accuracy while also improving benign accuracy compared to previous works.

The HYDRA pipeline can be described in five steps:

1) Pre-train the network: the CNN is trained on a dataset to minimize a specified loss objective, which is the ERM problem integrated with a robust training objective.
2) Initialize scores: a floating point importance score is assigned to each weight, using the scaled initialization technique.
3) Minimizing loss: weights of the CNN are frozen, but the corresponding importance scores are updated as the loss is minimized. If k weights are to be kept in the end, only top k importance weights will be used in predictions. However, all scores are updated in the backward pass.
4) Pruning: using the obtained importance scores that are now frozen, weights that are found to be less important are pruned away. If the desired pruning ratio is 90%, a binary pruning mask is created to only keep weights with top 10% score.
5) Fine-tuning: the pruned network has its parameters updated by training again with the dataset and the robust training objective. This step partially recovers the performance loss incurred during the pruning phase, and outputs the compressed network.

The paper was chosen as the foundation of this work, because of its excellent performance and extensive documentation. The performance of the pruning algorithm was strong enough to earn second place in the auto-attack robustness benchmark near the time the paper was published, and the paper has been cited 100+ time since the time of publication.

Current research on methods for obtaining efficient and adversarially robust CNNs often uses image classification as a proof of concept. Image classification is a simpler task compared to object detection, which makes classification models less complex and faster to train, test, and evaluate. This is advantageous for prototyping new techniques or testing optimization methods, however it leaves a gap in assessing the effectiveness of these methods on more complex tasks such as object detection. It presents additional

challenges with multi-object recognition, localization, and increased model architecture complexity. No current research paper has fully tested these methods on object detection, limiting our understanding of their performance and robustness in real-world applications.

III. Method

A. Adapting HYDRA to object detection

The HYDRA method was originally developed for the task of image classification, where the primary objective is to recognize and categorize the type of object present in an image. While this task is fundamental in many applications, there are scenarios where understanding the spatial arrangement of objects within an image is equally important. This need gives rise to the task of object detection, which not only identifies the objects but also localizes them by drawing bounding boxes around each detected object.

To adapt the HYDRA method for object detection, the original CNN models were replaced with YOLOv3. Weights for the model that are pre-trained on the COCO dataset are used, because the original datasets have been transitioned to COCO to include crucial localization information. To adapt a CNN model from object classification to object detection with YOLOv3, several critical modifications were implemented to accommodate the object detection framework.

First, the dataloader was restructured to read and process the COCO dataset, which contains bounding box annotations and multiple object classes within a single image, unlike classification where only a single label per image is needed. Each image's annotations were parsed to include coordinates, width, height, and class labels, with data augmentation applied to enhance robustness. The images are then each padded into a square and resized into 416x416 pixels.

Next, some output processing was added. Unlike classification, which produces a single prediction per image, YOLOv3 outputs a 2D matrix of bounding boxes, confidence scores, and class probabilities for each grid cell. This output was parsed to filter out low-confidence predictions using a confidence threshold of 0.5, followed by Non-Max Suppression (NMS) to remove redundant bounding boxes for the same object. This post-processing step ensures that the final output contains only the most probable predictions.

Following inference, detection outputs were saved in CSV format to allow for a structured evaluation. Each entry in the CSV file contained the image identifier, predicted bounding box coordinates, confidence score, and predicted class label. Finally, mAP (mean Average Precision) was calculated using the pycocotools library, an evaluation toolset that provides metrics specifically for object detection. Pycocotools compares the model's predicted bounding boxes against ground truth annotations across multiple IoU thresholds (from 0.5 to 0.95, in increments of 0.05) and averages the precision over these thresholds. This approach captures the model's performance in detecting objects of varying sizes and levels of overlap, providing a robust measure of detection accuracy for YOLOv3 in this adapted setting.

B. PGD attack

Implementing the PGD (Projected Gradient Descent) attack for YOLOv3 object detection involves generating adversarial perturbations that lead the model to misdetect or fail to detect objects. PGD, an

iterative attack method, aims to maximize the YOLOv3 model's prediction error by making small, carefully calculated perturbations to the input image. To apply PGD to YOLOv3, we start by calculating the gradient of the loss with respect to the input image. At each iteration, a small step is taken in the direction of this gradient, adding a bounded perturbation to the input image. The step size and the overall perturbation limit (ε) are hyperparameters that control the severity of the attack, with each step projected back to ensure the perturbation stays within this ε-bound. The process repeats for a fixed number of iterations or until the adversarial effect is achieved. Specifically for this work, when each pixel value is from 0 - 1, 5 steps are taken at 0.02 each, with the ε-bound as 0.05. This is a small enough perturbation that is not obvious to the human eye, but strong enough to decrease the model accuracy.

C. Implementing unstructured and structured pruning

In addition to the unstructured pruning techniques utilized in the original HYDRA method, structured pruning was implemented for a comparative analysis between the two approaches. As it can lead to more substantial reductions in model size and computational complexity while preserving the overall architecture of the network. This was inspired by the authors' previous work "Towards Compact and Robust Deep Neural Networks" where they demonstrated the effectiveness of employing both structured and unstructured pruning strategies. This comparison allows for a deeper understanding of the trade-offs associated with each pruning strategy, particularly in terms of model accuracy, robustness, and efficiency.

The structured pruning technique is implemented by focusing on reducing entire kernels/filters, as opposed to individual neurons in unstructured pruning. Specifically, for convolutional layers, structured pruning targets kernels by evaluating the importance of each kernel. The kernel importances are each calculated by averaging the importance of all the weights inside the kernel. This means that with the five-step HYDRA pipeline, changes had to be made in the pruning step where the binary pruning mask is applied at the level of kernels. An additional computation step is added to process the kernels' importance, and ensures that all the weights inside have the same value in the binary mask.

D. New initialization technique

The scaled initialization method was developed for HYDRA as a better alternative to random initialization for determining the importance of weights in a CNN. However, there are many different methods that can approximate how much each weight contributes to the final output. DeepLIFT (Deep Learning Important FeaTures) is a method that explains neural network outputs by attributing differences in output to changes in input features relative to a defined reference input. It calculates contribution scores for each feature by measuring their impact on the output difference from the reference state, using specific rules like the RevealCancel rule to enhance accuracy. This approach offers a more efficient and interpretable means of understanding neural network predictions compared to traditional gradient-based methods, providing valuable insights into feature importance and model behaviour. Therefore, DeepLIFT was used to initialize the weights' importance in this work.

The DeepLIFT implementation leverages the Captum library, a model interpretability and understanding library for PyTorch. Captum supports a variety of attribution methods, including DeepLIFT, which simplifies the integration of interpretability techniques into deep learning models. Here, DeepLIFT is used to replace steps 2 and 3 in the original HYDRA pipeline by calculating importance scores instead of minimizing a loss function directly. With DeepLIFT, each weight in the network receives an attribution score that indicates its contribution to the model's output, bypassing the limitations of gradient-based

approaches, such as issues with saturation (where gradients may vanish) or thresholding artifacts. The DeepLIFT scores are then used directly to generate the binary mask to prune weights or kernels, eliminating the need for traditional weight initialization and loss-based minimization.

IV. Experiments

In the context of object detection using the COCO dataset, mAP serves as a fundamental metric for evaluating the performance of detection models. mAP quantifies the precision-recall trade-off by calculating the average precision across different object categories. COCO dataset's mAP is typically computed at various Intersection over Union (IoU) thresholds such as 0.5, which is the threshold used in all experiments in the work. Also, a subset of 50,000 images from MS COCO is used to reduce training time.

A. Training time penalty

There is about 30 second time penalty using this initialization method over scaled initialization...

B. DeepLIFT vs Scaled initialization

Compare DeepLIFT to HYDRA initialization, mAP and time

describe why curve shape

why can hydra compress to 99%

compare to 0.05 or 5e-2, reject null hypothesis

Table I

TABLE I: p-values from ANOVA tests on mAP

Pruning ratio	0%	25%	50%	65%	75%	85%	90%	95%	99%
DeepLIFT vs. HYDRA (Unstructured)	1.03e-01	9.53e-02	1.71e-03	3.60e-04	1.57e-06	1.64e-04	2.43e-02	3.54e-02	1.86e-01
DeepLIFT vs. HYDRA (Structured)	1.48e-02	3.60e-02	1.18e-05	1.24e-05	6.91e-04	1.17e-02	8.99e-02	8.87e-03	5.80e-02
Unstructured vs. Structured (DeepLIFT)	6.52e-01	2.35e-05	3.70e-11	4.25e-11	4.20e-10	1.13e-09	5.46e-08	1.09e-01	3.99e-01
Unstructured vs. Structured (HYDRA)	9.00e-04	1.11e-05	1.17e-08	3.05e-09	4.44e-10	6.27e-07	1.80e-05	1.83e-03	2.42e-01

Figure 1

C. unstructured vs structured

Compare unstructured to structured pruning, mAP and time

TABLE II: p-values from ANOVA tests on inference time

Pruning ratio	0%	25%	50%	65%	75%	85%	90%	95%	99%
DeepLIFT vs. HYDRA (Unstructured)	1.28e-05	5.61e-02	7.16e-03	1.44e-01	3.96e-01	7.69e-02	4.37e-03	2.05e-01	6.64e-01
DeepLIFT vs. HYDRA (Structured)	5.13e-01	1.81e-06	2.80e-06	1.37e-02	4.21e-02	6.36e-02	2.06e-01	7.33e-01	2.15e-01
Unstructured vs. Structured (DeepLIFT)	1.75e-04	4.13e-19	4.31e-21	1.07e-24	2.76e-26	2.02e-22	6.17e-21	1.85e-16	1.68e-14
Unstructured vs. Structured (HYDRA)	8.70e-01	2.76e-17	1.03e-21	8.97e-26	3.45e-25	1.30e-21	3.55e-21	2.36e-20	1.56e-19

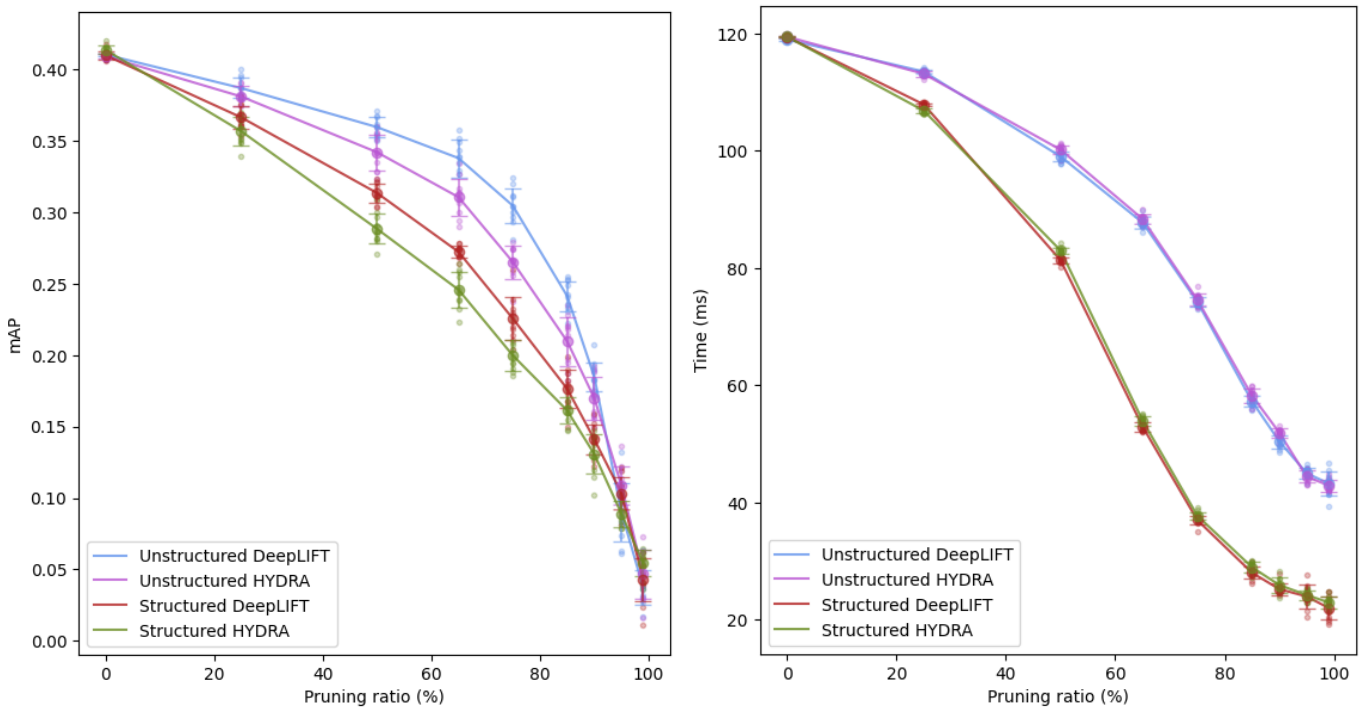

Fig. 1: Comparison of proposed approach versus HYDRA, through both unstructured and structured pruning. The mAP performance is shown on the left, and time on the right

TABLE III: p-values from ANOVA tests on mAP

Pruning ratio	0%	25%	50%	65%	75%	85%	90%	95%	99%
Greyscale vs Colour (Unstructured DeepLIFT)	1.13e-23	1.74e-15	1.06e-13	1.18e-11	2.53e-12	2.77e-10	1.75e-05	9.25e-02	5.06e-01

TABLE IV: p-values from ANOVA tests on inference time

Pruning ratio	0%	25%	50%	65%	75%	85%	90%	95%	99%
Greyscale vs Colour (Unstructured DeepLIFT)	6.11e-09	2.07e-07	1.14e-05	4.42e-04	2.08e-01	5.59e-01	2.62e-01	3.98e-02	4.82e-01

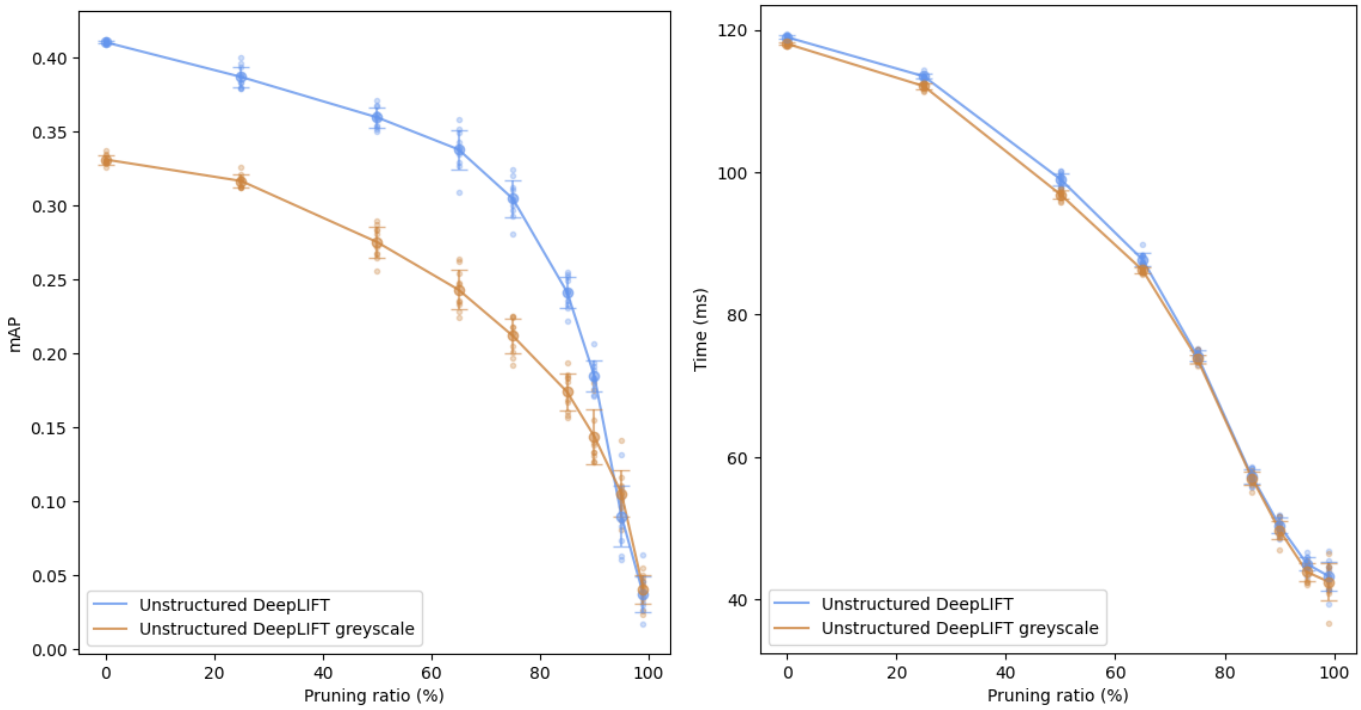

Fig. 2: Comparison of performance between color and greyscale images on the proposed method. The mAP performance is shown on the left, and time on the right

Bring up ANOVA results

Figure 1

D. effect of greyscale on performance

Figure 2

Bring up ANOVA results

E. example pictures

Subsection text here.

V. Conclusion

Conclusion here.

TABLE V: Unstructured DeepLIFT mAP

Pruning ratio	0%	25%	50%	65%	75%	85%	90%	95%	99%
run 1	0.410	0.380	0.353	0.358	0.293	0.249	0.176	0.103	0.046
run 2	0.411	0.379	0.368	0.343	0.311	0.242	0.172	0.097	0.036
run 3	0.410	0.387	0.360	0.335	0.300	0.231	0.189	0.093	0.045
run 4	0.411	0.383	0.352	0.327	0.304	0.253	0.191	0.087	0.030
run 5	0.409	0.379	0.371	0.349	0.312	0.241	0.183	0.083	0.039
run 6	0.410	0.396	0.350	0.339	0.281	0.222	0.194	0.074	0.017
run 7	0.412	0.400	0.354	0.352	0.297	0.252	0.183	0.063	0.064
run 8	0.410	0.394	0.367	0.329	0.304	0.236	0.182	0.108	0.027
run 9	0.411	0.388	0.362	0.338	0.324	0.233	0.171	0.061	0.032
run 10	0.410	0.383	0.359	0.309	0.320	0.255	0.207	0.132	0.037
max	0.412	0.400	0.371	0.358	0.324	0.255	0.207	0.132	0.064
min	0.409	0.379	0.350	0.309	0.281	0.222	0.171	0.061	0.017
avg	0.410	0.386	0.359	0.337	0.304	0.241	0.184	0.090	0.037
std dev	8.0e-04	7.1e-03	6.9e-03	1.3e-02	1.2e-02	1.0e-02	1.0e-02	2.0e-02	1.2e-02

APPENDIX A
RAW EXPERIMENT DATA

APPENDIX B

Appendix two text goes here.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, *Object detection in 20 years: A survey*, 2023. arXiv: 1905.05055 [cs.CV].

[2] H. Zhang and J. Wang, *Towards adversarially robust object detection*, 2019. arXiv: 1907.10310 [cs.CV].

[3] H. Li, G. Li, and Y. Yu, *Rosa: Robust salient object detection against adversarial attacks*, 2019. arXiv: 1905.03434 [cs.CV].

[4] J. C. Costa, T. Roxo, H. Proença, and P. R. M. Inácio, *How deep learning sees the world: A survey on adversarial attacks & defenses*, 2023. arXiv: 2305.10862 [cs.CV].

[5] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, *Recent advances in adversarial training for adversarial robustness*, 2021. arXiv: 2102.01356 [cs.LG].

[6] G. K. Erabati, N. Gonçalves, and H. Araujo, "Object detection in traffic scenarios - a comparison of traditional and deep learning approaches," Jul. 2020, pp. 225–237. DOI: 10.5121/csit.2020.100918.

[7] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, *A survey of modern deep learning based object detection models*, 2021. arXiv: 2104.11892 [cs.CV].

[8] T.-Y. Lin *et al.*, *Microsoft coco: Common objects in context*, 2015. arXiv: 1405.0312 [cs.CV].