Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Alexander Michael Staff

An Empirical Study on Cross-data Transference of Adversarial Attacks on Object Detectors

Master's thesis in Computer Science
Supervisor: Jingyue Li
Co-supervisor: Elizabeth Traiger
June 2021

NTNU
Norwegian University of Science and Technology

DNV

Alexander Michael Staff

An Empirical Study on Cross-data Transference of Adversarial Attacks on Object Detectors

Master's thesis in Computer Science
Supervisor: Jingyue Li
Co-supervisor: Elizabeth Traiger
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Object detectors are increasingly deployed in safety-critical scenarios, including autonomous vehicles. Recent studies have found that neural networks are fundamentally weak to adversarial attacks. Adversarial attacks on object detectors involve adding a carefully chosen perturbation to the input, which causes the object detector to make mistakes. To make sure these safety-critical systems are trustworthy, the risks of adversarial attacks must be known.

This thesis investigates adversarial attacks where the attacker does not have access to the target detector or its training set. Devising an attack in this scenario requires the attacker training their own model on data which resembles the target detector's training set as much as possible. Using their own model as a surrogate, lets the attacker generate adversarial attacks without accessing the target detector. Experiments with this type of attack will establish whether one can effectively attack the private model using public data.

To attack Darknet models with the Targeted Objectness Gradient (TOG) family of attacks, the attack framework developed by Chow *et al.* [1] was modified. The modifications made the framework output adversarial samples, Darknet models could make predictions on.

Though initial transference between the attacking and target model is low, increasing epsilon from 8 to 24 under the L_∞ distance metric strengthens transference, and reduces the target detector mean Average Precission (mAP) by about half. Transference is also studied when the datasets for the attacking and the target model intersect. Attack performance is found to be proportional with the intersection. With the stronger transference afforded by intersecting datasets, epsilon can be dropped to 16 and retain the attack performance.

Sammendrag

Objektdetektorer blir stadig mer brukt i sikkerhetskritiske scenarier, inkludert autonome kjøretøy. Nyere studier har funnet at nevrale nettverk har en grunnleggende svakhet for fiendtlig støy . Å angripe objektdetektorer med fiendtlig støy, innebærer å legge til nøye valgt støy i inputen, som får objektdetektoren til å gjøre feil. For å sikre at disse sikkerhetskritiske systemene er pålitelige, må risikoen for slike angrep være kjent.

Denne avhandlingen undersøker fiendtlig støy angrep der angriperen ikke har tilgang til objektdetektoren under angrep, eller dens treningssett. Å utarbeide et angrep i dette scenariet krever at angriperen trener sin egen modell på data som ligner målet sitt treningssett, så mye som mulig. Ved å bruke sin egen modell som surrogat, kan angriperen generere fiendtlig støy uten direkte tilgang til målet. Eksperimenter med denne typen angrep vil avgjøre om man effektivt kan angripe den private modellen ved hjelp av offentlige data.

For å angripe Darknet modeller med Targeted Objectness Gradient (TOG) familien av angrep, ble rammeverket utviklet av Chow *et al.* [1] modifisert. Modifikasjonene gjorde at rammeverket generert bilder med fiendtlig støy som Darknet - modellene kunne gjøre deteksjoner på.

Angrepsytelsen på fiendtlig støy overført mellom angreps- og målmodellen, er lav til å begynne med. Derimot, hvis en øker epsilon fra 8 til 24 under L_∞ avstandsmetrikken, styrker overføringen og reduserer måldetektoren sin mAP til omtrent halvparten. Overføring studeres også når datasettene for den angripende modellen og målmodellen har overlapp. Angrepsytelse er funnet å være proporsjonal med størrelse på overlappen. Med sterkere overføring grunnet overlapp i datasettene, kan epsilon settes ned til 16 og beholde angrepsytelsen.

Preface

This thesis concludes the author's master's degree form the Department of Computer Science at the Norwegian University of Science and Technology (NTNU). I would like to thank my supervisors, Associate professor Jingyue Li of the Department of Computer Science, and Elizabeth Traiger, Senior Researcher at DNV. Their guidance, support, and valuable feedback improved the thesis immeasurably.

From DNV I'd also like to thank Jon Arne Glomsrud, and Kristian Bertheussen Karolius for their questions, comments, and suggestions which helped shape the direction of the research.

Finally, I would like to thank my friends and family for their support throughout my studies. Producing this thesis without your support would not have been possible, especially during lockdown. A special thanks to my brother, Robin, without whom I would not have studied computer science.

Trondheim, 25 June 2021
Alexander Michael Staff

Contents

Figures

Tables

Code Listings

Acronyms

CNN Convolutional Neural Network. 3, 9

COCO Common Objects in Context. 29

Faster R-CNN Faster Regions with CNN Features. 17, 18, 21, 22, 24, 31, 33, 61

IOU Intersection Over Mean. 6, 26, 27, 33

mAP mean Average Precission. iii, v, 14, 24, 26, 27, 32, 33, 35, 36, 38, 39, 41–43, 45–47, 52, 54, 55, 57–59, 61

NMS Non Maximum Suppression. 5

RPN Region Proposal Network. 17, 21, 23, 31

SMD Singapore Maritime Dataset. xiii, 34, 35, 38, 39, 44, 45

SSD Single Shot MultiBox Detector. 4–6, 21, 24, 33, 61

TOG Targeted Objectness Gradient. iii, v, xvii, 11, 23, 31–33, 43, 46, 57–59, 61, 62

XML Extensible Markup Language. 37, 38

YOLO You Only Look Once. 4, 15–17, 19, 21, 22, 24, 26, 29–32, 35–39, 43, 46, 47, 57, 61

Glossary

black-box A context where the attacker does not have access to the internal state of the target. See detailed explanation at [2.4.5]. 1, 2, 20, 24–27, 33, 34, 43, 54, 57

COCO A large dataset compiled for object detection training and benchmarking. xiii, 15, 16, 26, 27, 30, 33, 35, 37, 39, 44, 45

Convolutional Neural Network A type of deep neural network which is well suited to analysing images. 3

CUDA The API developed to execute GPU accelerated tasks on Nvidia graphics cards. 30, 41

Darknet The neural network framework YOLO was initially developed in. iii, v, 2, 26, 30–33, 36–39, 41, 44, 46, 47, 49, 52, 57–59, 61, 62

DNV A leading certification body. Researchers from DNV co-supervised this thesis. xiii, 30, 34, 35, 44

Keras A python interface for TensorFlow. 31

TensorFLow An open-source platform for machine-learning. 30–32, 46, 47, 49, 52, 57, 59, 62

YOLOv3 The second revision of a popular real-time object detector. 6

Chapter 1

Introduction

Object detection and classification have seen a rapid improvement since the ImageNet competition in 2012. At this competition, deep convolutional networks almost halved the the error rates of the best competing approaches [2]. Such a great improvement, caused a paradigm shift in computer vision which led to powerful object detection models. Now object detection has been deployed in many different contexts, including safety-critical areas like autonomous vehicles. Unfortunately, as the rapid development continued, it was discovered that neural networks could be caused to miss-predict by generating imperceptible non-random perturbations and adding these to the input. These inputs, first described by Szegedy *et al.* [3] were called adversarial samples. It turned out that all neural networks are fundamentally susceptible to making critical errors if exposed to these adversarial samples. Attempting to make the networks more robust to these perturbations is an ongoing area of study [4]. Furthermore, it turned out that adversarial samples generated to attack a particular model were even effective on models with different architectures [5][1].

In this thesis it is not the cross-architecture transferability which is being studied, but rather the transferability between two models trained on different datasets. If an attacker decides to attack a commercially deployed autonomous vehicle, it is unlikely the attacker would have detailed information about the detector, or the training set used to train the object detector. However, if an attacker is able to inject data into the training dataset of the target detector, they could train a surrogate model on a dataset with a significant intersection with the target detector, and then generate perturbated images from those models for injections attacks. This thesis will establish whether one can effectively attack a model trained on private data, using only public data. Furthermore, we investigate whether datasets sharing images, affect the transference of the attacks. In this scenario, the attacker uses a surrogate model to generate an adversarial sample, then injects the sample directly to the target model to fool it.

To investigate whether it is possible to transfer attacks across models trained on different data, models are trained to detect the same semantic class, namely, boat, with various datasets. In addition to testing black-box attacks, the effect on

attack performance when the surrogate and target models are trained on datasets with intersections is investigated. To test this, some models are trained on datasets that share images with each other, while other models are trained on datasets that are entirely disjoint. The adversarial samples are generated using a source model trained to detect ships on one dataset, then the target model, trained to detect ships on a separate dataset, attempts to make predictions on the sample. Comparing the attack performance when the target detector is used directly to generate adversarial samples, with the attack performance when a surrogate model is used to generate the samples, demonstrates the risk of transfer attacks.

Through our investigation of black-box cross-data transfer attacks, this thesis contributes: performance results on black-box transfer attacks, as well as modifications to the Targeted Objectness Gradient attacks, which enable attacking Darknet models.

Through our investigation of the effect of dataset intersection on the transfer attacks, this thesis contributes: results pertaining to adversarial attacks on models with dataset intersections compared to models without, and several high performance single-class object detection models. This is useful to those who are deploying real-time object detectors, as they need to be aware of the risks involved.

The results of this thesis show how exposed a model is to a black-box transfer attack, as well as how the risk changes based on the intersection between the surrogate, and the target model. Perturbations generated with an epsilon of 8, under the distance metric L_∞ show poor transference between the models, the risk of such a black-box attack is quite low. However, an epsilon in the 20-30 range, results in better transference, and more effective attacks. Models trained on intersecting datasets show transference which is proportional with the size of the intersection. Performing a cross-resolution attack revealed that an attacker who does not know the resolution of the victim detector should choose a lower resolution like 416x416 to raise the probability of the attack being upsampled rather than downsampled.

The thesis begins with relevant theoretical concepts in chapter 2 and related work is presented in chapter 3. Research methodology is presented in chapter 4 and implementation is detailed in chapter 5. The results to the research questions are presented in chapter 6. Decisions made, as well as weaknesses, are discussed in chapter 7. Finally, chapter 8 concludes the thesis, and presents ideas for followup research.

Chapter 2

Background

2.1 Convolutional Neural Networks

Convolutional Neural Networks, sometimes refereed to as ConvNets, in this thesis CNNs, are the deep learning network architecture of choice for analysing images. CNNs underpin the the object detector discussed in this thesis, as well as most modern object detectors and classifiers.

Convolutional Neural Network (CNN) are well suited to any task involving images, and have been applied to classification, segmentation, and object detection. CNNs gained prominence after an impressive result in the ImageNet 2012 competition. The success in the competition made CNNs with ReLU activation functions and dropout layers standard for computer vision tasks. Though these large models would once be slow to train and make predictions with, advances in GPUs and parallelization of algorithms have led to reasonable training time, and real-time predictions [2].

Convolutional Neural Networks exploit the nature of images to produce feature maps that are refined to predictions [6].

2.2 Object Detectors

Object detection is a computer vision task which takes an image as input and gives bounding boxes with class labels as output. **The following section is adapted from my specialisation project [7].**

Object detectors are closely related to classifiers, with early object detectors simply being classifiers applied to an image multiple times using a sliding window at different scales. Another strategy is to rely on two separate neural networks. One network is tasked with feature execration which locates objects. The other network classifies the located objects. These object detectors have high precision, but are generally not fast enough for real-time tasks. To make real-time predictions, one-stage object detectors were developed. One-stage detectors predict bounding boxes and object classes at the same time. Two real-time object

detectors: Single Shot MultiBox Detector (SSD) [8], and You Only Look Once (YOLO) [9] share several strategies. At a fundamental level they are both feed-forward convolutional networks. The convolutional nature of the network allows multiple objects to be detected in parallel.

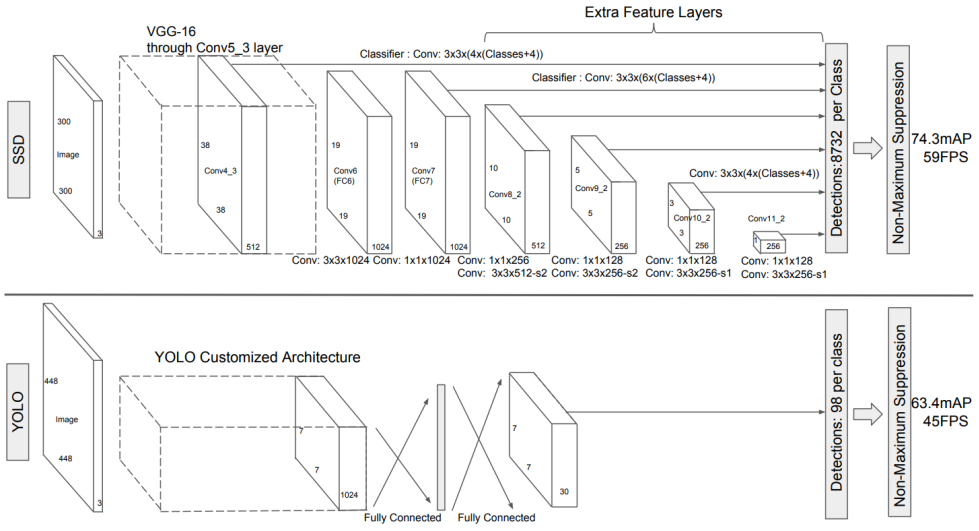


Figure 2.1: Comparison between the initial YOLO and SSD architectures[1]

2.2.1 Backbone Network

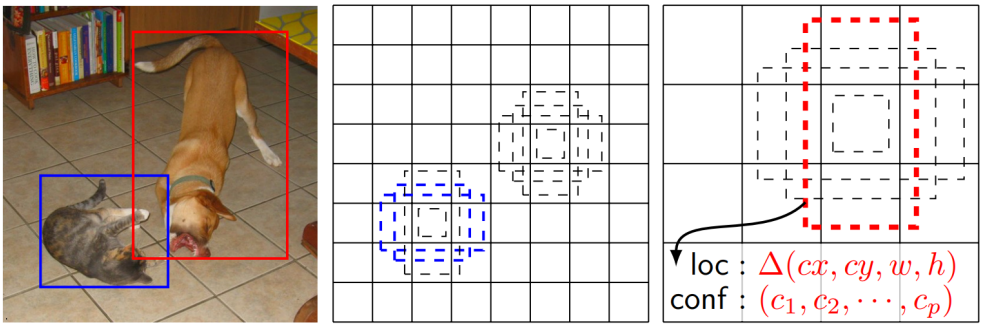
Both YOLO and SSD rely on a backbone network as part of their architecture. A backbone network is a convolutional network pre-trained as a classifier, which is used to find possible objects in the image. After the backbone architecture has been trained, the last few layers are removed. This makes the network output low resolution representations of the input image. These images, called feature maps, describe the characteristics of the input image. The low resolution of the feature maps can be viewed as a grid with activations where objects are present, if there are several objects in the image there should be multiple activations in the grid. See 2.2.

To make predictions having an indication that an object might be present is not sufficient. Object detectors need to figure out the probability a grid cell contains an object p_{obj}, they need to figure out which class the object belongs too, and they need to predict the bounding box coordinates (x, y, h ,w). This amounts to each bounding box requiring five output channels in addition to object class. Each output channels is associated with a convolutional filter. To deal with multiple objects in a grid cell, each grid cell is able predict multiple bounding boxes.

SSD does not predict p_{obj} and instead seeks to directly predict whether a class

[1] Single Shot MultiBox Detector: `https://arxiv.org/abs/1512.02325`

is present in a bounding box. To recognise when there are no objects present SSD uses a "background" class.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Figure 2.2: The SSD feature map and default boxes[2]

2.2.2 Non-max Suppression

Filtering predictions on p_{obj} lets the network discard low confidence predictions, but the network might produce several high confidence predictions for the same object. A technique called Non Maximum Suppression (NMS) is used to deal with this. NMS discards redundant predictions by selecting the highest confidence prediction for each class, then discarding any prediction of the same class which overlaps the high confidence prediction beyond a threshold.

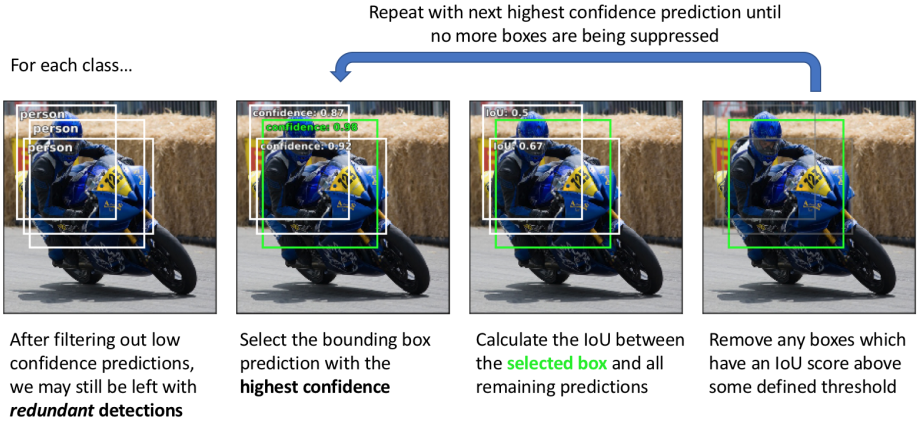


Figure 2.3: A demonstration of a non-max suppression algorithm[3]

[2]Single Shot MultiBox Detector: `https://arxiv.org/abs/1512.02325`
[3]An overview of object detection: one-stage methods: jeremyjordan.me

2.2.3 Anchor Boxes

At first YOLO would directly predict the four coordinates of the bounding boxes [10]. Other object detectors e.g: Faster R-CNN and SSD used hand picked priors, also called anchor boxes. This technique predicts an offset from the established boxes. Introducing anchor boxes simplifies the task the object detector needs to accomplish. By simplifying the problem, the network is faster and easier to train. SSD uses a set of default boxes and aspect ratios. YOLO however uses k-means clustering on the bounding boxes of the training set to discover the best anchor boxes. Since using Euclidean distance would cause larger boxes to generate more error, YOLO uses a custom distance metric. $d(box, centroid)$ refers to the difference between the centroid (found by k-mean clustering), and the bounding box.

$$d(box, centroid) = 1 - IOU(box, centroid)$$

Intersection Over Mean (IOU) also called Jaccard index is a measure of how similar two sets are, and is defined as:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

2.3 YOLOv3

YOLOv3 is a one-stage object detector built to run on the Darknet framework. Darknet is written in c and CUDA (NVIDIA's parallel computing API) which allows fast execution and GPU acceleration. Due to YOLO's impressive performance it has been implemented in several neural network frameworks. Among them: OpenCV, TensorFlow, and PyTorch. YOLOv3 uses the Darknet-53 backbone network, see: figure 2.4.

The YOLO algorithm received two iterative updates from its creators. Where YOLO predicted bounding boxes directly using fully connected layers, YOLO9000 added anchor boxes and predicted an offset from these instead. YOLOv3 brought two central improvements. The backbone network was improved and outputs at multiple scales was added [9].

A challenge to accurate object detection is detecting objects at different scales in the image. If too small an area is considered large objects wont be detected, but if only large areas are considered, recall of small objects drops significantly. SSD solves this by outputting predictions intermittently as the network downsamples the feature map. See 2.1. YOLO uses a technique called a feature pyramid. After the first prediction at the smallest scale, the feature map is upsampled and concatenated with a feature map from earlier in the network. Then this process is repeated for the final scale. This allows the larger scale predictions to benefit from fine grained information discovered in the earlier feature maps. This significantly improves performance on small or distant objects. However the output from the

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2.4: Darknet-53 is the backbone network of YOLOv3[4]

smallest scale is less robust and therefore easier to fool with adversarial examples. This is discussed by Zhao *et al.* [11]

2.4 Adversarial Examples

Adversarial examples, as defined by Szegedy *et al.* [3] are test images that have an imperceptible non-random perturbation applied to them. This perturbation causes the network to arbitrarily change its classification. The perturbation is generated by optimising the input to maximise the prediction error. The most surprising discovery was that these adversarial examples will often cause misclassifications on networks trained on different examples or have different architectures [3]. Maximising prediction error is not the only way to find adversarial examples. In fact, different adversarial attacks often develop their own loss function to find adversarial examples. For object detectors the loss function might include shifting bounding boxes or suppressing detections altogether.

Goodfellow *et al.* [12] argues that linearity is the reason these models are vulnerable to adversarial examples. Due to design effort to make networks operate linearly for as much time as possible, they are vulnerable to these attacks. This linearity is an intentional optimisation to make the model faster and easier to train. It grants stability to the model that lets it converge faster. But to be robust

[4]Redmon and Farhadi [9]

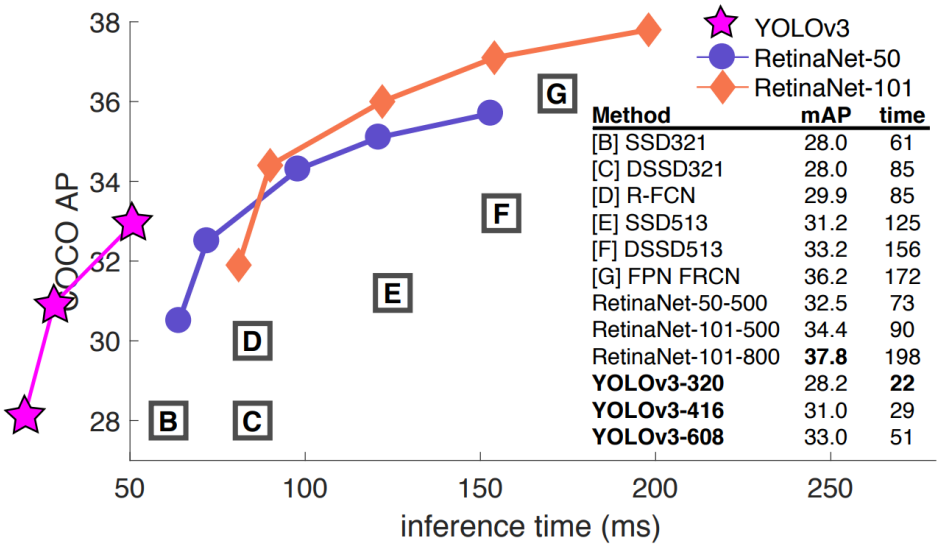


Figure 2.5: YOLO mean average precision and inference time compared to other object detectors[5]

against these attacks it needs non linear effects.

2.4.1 Fast Gradient Sign Method

Applying this linear view of the vulnerability led to the development of new ways of generating adversarial examples. The fast gradient sign method (FGSM) allows adversarial examples to be generated quickly. Fast enough that adversarial training (augmenting the training set with adversarial examples) becomes practical. See a visual representation here: figure 2.6. Let η be the adversarial perturbation with θ being the parameters of model, x is the input and y the targets of x. With $J(\theta, x, y)$ representing the cost used in training, equation 2.4.1 gives an adversarial perturbation constrained on ϵ.

$$\eta = \epsilon sign(\nabla_x J(\theta, x, y))$$

Using FGSM Goodfellow *et al.* [12] were able to cause an error rate of 99.9% on a shallow softmax classifier with $\epsilon = .25$. Softmax is defined here: equation 2.4.3. The fact that simple and cheap algorithms have such a strong effect is further evidence of the linearity interpretation of adversarial examples.

2.4.2 Generalisation of Adversarial Examples

The reason adversarial examples generated for one model often causes a different model to misclassify is also explored [12]. The effects that needed explaining were:

[5] Redmon and Farhadi [9]

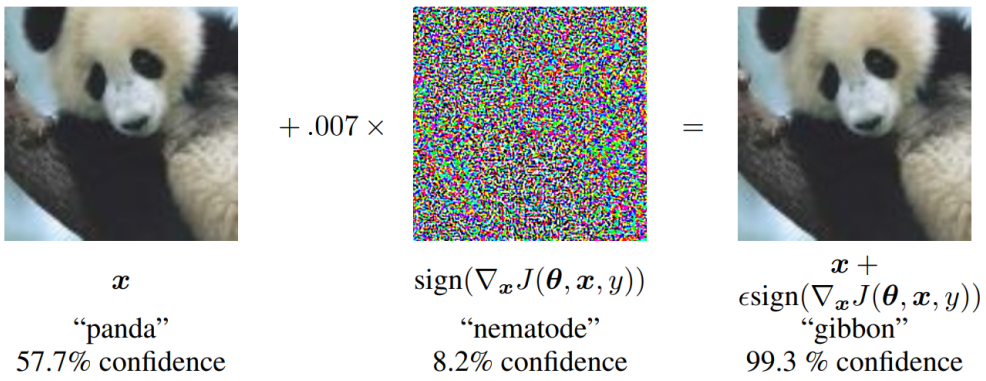


Figure 2.6: This figure demonstrates how an FGSM attack operates.[6]

- Models trained on disjoint training sets were vulnerable to the same adversarial examples [12].
- Models with different architectures were vulnerable to the same adversarial examples [12].
- Multiple classifiers would assigning the same class to adversarial examples [12].

To the linear view, adversarial examples occur in broad subspaces. Finding them only requires η having a positive dot product with the gradient of the cost function. This property is visualised in figure 2.7. Due to this, adversarial examples are abundant and different classifiers have a high probability of misclassifying the same adversarial example.

Goodfellow *et al.* [12] further argues that, with different classifiers assigning the same, wrong, class to adversarial examples, there is an indication that Convolutional Neural Network (CNN) classification models resemble a linear classifier trained on the same set. This is a consequence of the way they are trained. If one were to train a linear reference classifier on a different subset of the training data, it would learn similar weights. This gives classifiers the ability to generalise, but also makes them predict the same label for the adversarial examples.

2.4.3 Beyond FGSM

As the field responded to Goodfellow *et al.* [12], new attacks and defenses were developed. The DeepFool [14] attack was developed as an efficient way to benchmark model robustness. It also outperformed contemporary attacks in that it fooled networks with a smaller perturbation. Papernot *et al.* [15] developed defensive distillation to make networks more adversarially robust. They saw a great improvement in resisting adversarial examples, but it didn't take long before attacks were developed to defeat defensive distillation. Carlini and Wagner [4] developed new strong attacks that were able to fool models that had been defensively dis-

[6] Goodfellow *et al.* [12]

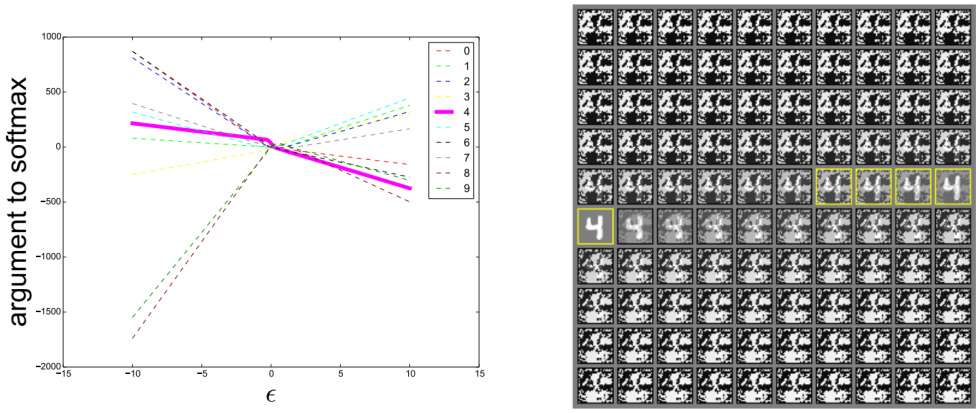


Figure 2.7: This plot demonstrates that a sufficiently large ϵ generates an adversarial example, given one moves in the correct direction. In the plot on the left we see the arguments to the softmax layer of a classifier naively trained on MNIST [13]. The correct classification is four, which only occurs on a thin manifold. The rest of the space is adversarial examples and rubbish class examples (an input a human would say does not belong to any class). The figure indicates that the arguments are piecewise linear with ϵ. On the right we see the input to the model.[7]

tilled. This demonstrated that fully securing neural networks from adversarial examples would be hard.

Defensive distillation works by splitting the training into two steps with two identical networks. When training the first network the softmax function is modified to include a constant T, called a temperature constant.

Standard softmax function:

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\Sigma_j e^{x_j}}$$

Modified with temperature constant:

$$\text{softmax}(x, T)_i = \frac{e^{x_i/T}}{\Sigma_j e^{x_j/T}}$$

Once the network is trained, it is used to generate soft training labels. In this case, that means that instead of the labels being the groundtruth, they are instead the output vector. The vector gives the predicted probability of each class. Then the distilled model is trained on the soft labels. This will train the second network to make predictions like the first. The point of this is to prevent the network from overfitting to the training data. One of the theories put forth by Szegedy *et al.* [3] was that adversarial examples exist because the model overfit to the training data and has therefore developed blind spots. This is not the explanation favoured

[7] Goodfellow *et al.* [12]

by Goodfellow *et al.* [12]. When Carlini and Wagner [4] developed attacks that could defeat defensive distillation, they suggest this is additional evidence for the linearity view of adversarial attacks.

To compare different adversarial example generation methods, an important attribute is how large the perturbation needs to be to fool the model. Distance metrics are used to define how large a perturbation is.

- L_0 measures the number of points that are changed between x and x'. For an image, this means how many pixels have been altered.
- L_2 is the Euclidean distance between x and x'. Small changes to many pixels will keep L_2 small.
- L_∞ measures the maximum change to any point. This is the distance metric used by Goodfellow *et al.* [12] in FGSM. When configuring an FGSM attacker one selects an ϵ which represents the largest change allowed. Which gives: $||x - x^{'}||_\infty = max(|x_1 - x_1^{'}|, ..., |x_n - x_n^{'}|)$

To develop an algorithm that constructs adversarial examples Carlini and Wagner [4] first defined the problem as:

$$\text{minimise } D(x, x+\delta)$$

$$\text{such that } C(x+\delta) = t$$

$$x + \delta \in [0,1]^n$$

x is fixed. So, to minimise $D(x, x+\delta)$, one must find a δ. D is distance metric, either L_0, L_2, or L_∞. Because $C(x+\delta) = t$ is highly non-linear, the problem is reformulated to allow for the use of existing optimisation algorithms. Then the best algorithm is determined empirically.

2.4.4 TOG

The Targeted Objectness Gradient (TOG) attacks, developed by Chow *et al.* [1], are a class of adversarial attacks developed specifically to target object detectors. The attacks use an iterative gradient method to force mistakes in object detectors. With detection expressed as $\mathcal{O}^*(x)$ and attack loss as $\mathcal{L}^*$, TOG can be expressed as :

$$x_{t+1}^{'} = \prod_{x,\epsilon}[x_t^{'} - \alpha_{TOG}\Gamma(\nabla_{x_t^{'}}\mathcal{L}^*(x_t^{'}, \mathcal{O}^*(x); \theta))] \tag{2.1}$$

$x_t^{'}$ is the adversarial sample at the t-th iteration, Γ is the sign function, and α_{TOG} is the attack learning rate. Further details can be found in their paper [1]. Using this TOG can find adversarial perturbations targeting different functions of object detectors. The *Untargeted* attack is successful when it causes the detector

to misdetect. Whether it fails to detect an object, misclassifies an object, or fabricates an object. The *Object-vanishing* attack seeks to suppress detections, and is successful when the detector finds no objects in the sample. The *Object-fabrication* makes the detector hallucinate multiple objects, making the prediction useless. Finally there is the *Targeted object-mislabeling* attack. This attack causes the detector to mislabel detected objects with the chosen target class, while maintaining the correct bounding boxes [1][16]. The targeted object-mislabeling attack is less relevant to this thesis, as most of the models being studied are single class models.

2.4.5 Transfer Attacks

Transfer attacks are adversarial attacks with different source and target models. Different models, in this context, have different model architectures or different training sets. Even generating the adversarial sample at a different resolution than the detection operates at can be seen as a transfer attack. Transfer attacks rely on the fact that even though two models have been trained on different datasets, if they are able to detect the same objects it can be assumed that the two models share commonalities. These can be exploited through adversarial attacks when the attacker lacks access to the target model. Normally, to generate adversarial examples one needs to be able to give the model input and read the output. If one lacks access to the model however, one technique is to train a stand-in model (the source model) to make similar predictions to the target model. Then adversarial samples generated using the source model will also have an effect on the target model [17][1].

2.5 Performance Metrics

There are several ways to measure the performance of a neural network. To describe the accuracy of a predictive model there are some basic metrics: recall, precision, and F1 score.

Symbol	Explanation
tp	True positive
fp	False positive
tn	True negative
tp	True negative
tpr	Recall
ppv	Precision
R_n	Recall at nth threshold
P_n	Precision at nth threshold
AP	Average precision
MAP	Mean average precision
Q	Number of queries

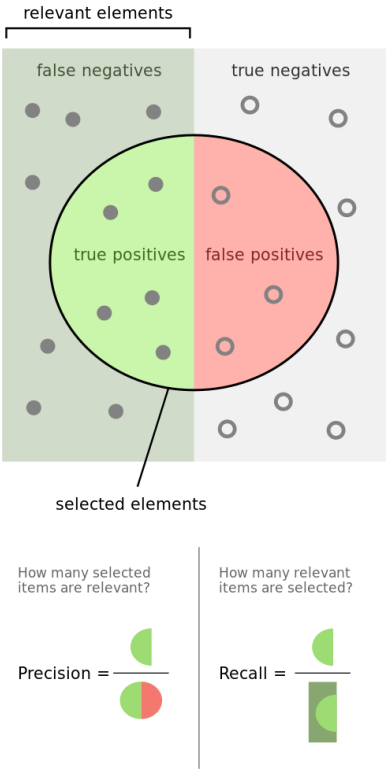


Figure 2.8: An illustration of precision and recall[8]

$$Precision = \frac{tp}{tp+fp}$$

$$Recall = \frac{tp}{tp+fn}$$

$$F_1 = 2 * \frac{ppv*tpr}{ppv+tpr} = \frac{2tp}{2tp+fp+fn}$$

These metrics are useful to understand the performance, but to get a better view, one can plot a precision-recall curve. This plot shows the tradeoff between precision and recall at different confidence thresholds. Calculating the area under the precision-recall curve gives average precision.

$$AP = \sum_n (R_n - R_{n-1}) p_n$$

Since average precision is generated for each query one can take the mean to get a single number that represents the models performance on the whole dataset.

[8]Precision and recall: `https://en.wikipedia.org/wiki/Precision_and_recall`

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q}$$

One of the reasons mAP became an important metric for object detection is that object detection challenges like PASCAL VOC [18] and COCO [19] rank object detectors by mAP. This makes it easy for researchers to compare their models with the state of the art if they know the mAP it achieves.

Chapter 3

Related Work

3.1 Towards Adversarially Robust Object Detection

Zhang and Wang [20] present a multi-task learning perspective for adversarial training. They use several experiments to establish benchmarks for different architectures and task losses. Key findings are:

- Classification and localisation task domains are largely distinct with some overlapping.
- The gradients of the two tasks are not fully aligned, which means there is possible conflict between the gradients.
- Comparing models with different task domains, the model with the domain which is the union of the classification and localisation domain showed the best overall performance.
- These results were verified on several different architectures and backbones.

3.2 Contextual Adversarial Attacks For Object Detection

Bounding boxes that include background information are a specific target of this attack. The contextual adversarial perturbation [21] attack is a fully white-box attack which uses backpropagation to maximise classification loss, regression loss and a novel context loss. As detecting ships in the ocean relies on a lot of contextual information, the sea in the background, this attack would be particularly effective against an autonomous maritime vessel.

3.3 Relevance Attack on Detectors

Chen *et al.* [22] develop a highly transferable attack together with a dataset of adversarial samples based an COCO. These adversarial samples are generated to be transferable which makes them suitable to benchmark model robustness. The developed attack, called Relevance Attack on Detectors, is tested on cross-architecture transferability by using a You Only Look Once (YOLO) model as a

surrogate model to attack other object detectors. However, these models are all trained on the COCO dataset. In this thesis the goal is to study transferability between models trained on different datasets. When studying cross-dataset transferability, the attacker can not train their surrogate detector on the same dataset as the target detector.

3.4 Cross-task Universal Perturbation

Zhang *et al.* [5] develop an attack that generates universal perturbations which transfers across task, model, and dataset. The perturbations are generated using U-Net [23] and ResNet [24], and are tested several models, including YOLO, to show transferability. The goal of this attack is to create image independent perturbations, that allow for rapidly producing a large adversarial set. In the case that there is a size mismatch between the perturbation and the target image, Zhang *et al.* [5] present two methods to correct the size discrepancy. One method is to simply resize the perturbation to fit, while the other method overlays multiple perturbations then clips the perturbation to fit the target image.

3.5 Multiple Object Tracking Attack

Jia *et al.* [25] argue that the Multi Object Tracking (MOT) system often reading the output of an object detector in an autonomous vehicle reduces the effectiveness of standard object detection adversarial attacks. Motivated by this finding they develop a novel attack that defeats MOT by hijacking it. They are able to shift tracked objects into the path of an autonomous vehicle with nearly 100% success rate by changing only 3 frames.

The goal of the attack is to suppress the initial detection of an object while fabricating a detection of the same object shifted to one side. This causes the object tracker to apply a false velocity to the object which can move an obstacle in to or out of the path of an autonomous vehicle.

The attack is more effective if the difference between the predicted velocity of the target and the induced velocity from the attack is great. The attack generates an adversarial patch and applies this to the target. The location of the patch is important to how the attack functions. In future work, Jia *et al.* [25] speculate that the adversarial patches could be printed and used in a physical attack, but this was not tested.

3.6 Attacks Analysis

This section investigates various adversarial attacks. There are several attributes which a relevant attack should posses. The attack should target object detectors, though it is feasible to modify a classifier attack to attack an object detector,

these attacks are never as effective as attacks designed to attack object detectors [20]. As our goal is to measure attack transference, the attack should apply an adversarial perturbation to the entire image. Adversarial patches, and other localised attacks, are often designed to be deployed physically, which would be a confounding factor for this thesis. Furthermore, the attack must not target the Region Proposal Network (RPN) that some detector use [26]. Since YOLO is a regression based detector and does not make use of an Region Proposal Network (RPN), these attacks will not be particularly effective. Finally, attacks that target real-time object detectors or object detectors deployed in autonomous vehicle directly, are particularly relevant. Any attack that possesses all or most, of these attributes will likely have a YOLO implementation, which makes setting up an attack pipeline much simpler.

In table 3.1 we listed adversarial attacks on object detectors, and detailed their suitability in regards to studying transference attacks on YOLO.

Attack	Real-time	Regression-based	Autonomous Vehicles	Summary
Pick-Object-Attack: The Pick-Object-Attack [27] constrains the perturbation to the targeted objects bounding boxes. The goal is to generate smaller perturbations, while maintaining attack effectiveness. Nezami *et al.* [27] argue the smaller perturbations are harder to detect by downstream systems. E.g. image captioning.	✓	✗	✗	The Pick-Object-Attack is designed to attack Faster Regions with CNN Features (Faster R-CNN) trained on the Visual Genome dataset. Though Faster R-CNN models can have real-time performance, this attack is not targeting autonomous vehicles. Additionally, fooling a downstream image captioning system is not relevant to this thesis.

Contextual Camouflage Attack: The Contextual Camouflage Attack seeks to fool object detectors by applying camouflage patterns to vehicle. The patterns are applied to a vehicle in a simulated photo realistic environment, and if the attack is successful, the object detector fails to detect the other vehicles in the scene.	✓	✓	✓	The attack checks a lot of boxes, but is ultimately not relevant to this thesis. The attack applies camouflage designs to vehicles in a simulate photo realistic environment. This would add a lot of variables if one were to gather data model transference.
Contextual Adversarial Attacks For Object Detection: Zhang *et al.* [21] develop a novel attack called Contextual Adversarial Perturbation (CAP). This attack targets the context of an object to degrade recall. The attack is designed for proposal-based detectors e.g. Faster R-CNN [26]	✓	✗	✗	Relevant to this thesis in so far as our object detector needs to rely on contextual information for detections of distant watercraft. However, the attack does not target regression based detectors, Making it unsuitable to this thesis.
Membership inference attack: Park and Kang [28] Develop a membership inference attack targeting object detectors. They train an attack model that can predict whether an image is part of the training data	✓	✓	✗	Although, inferring which images are in the training set could be relevant to training a surrogate model on a dataset with a large intersection with the target model dataset, this attack can not be used to study attack transference.

Sparse Adversarial Attack: Bao [29] Develop the Sparse Adversarial Attack which targets object detectors with adversarial patches. The attack is an evasion attack, which means it seeks to prevent the object detector from detecting anything. The attack is constrained by the l_0 distance metric, so it attempts to alter as few pixels as possible (less than 2% of the image), hence its sparse nature. They define separate loss functions for different models due to different models treating the background differently.	✓	✓	✗	As it is a patch attack, it adds extra variable to the analysis of the attack. Additionally, as it has different loss functions for different models, the results would be less generalizable.
Adversarial Attack against Multiple Object Tracking: Jia *et al.* [25] Develop an attack which is designed to defeat multi object tracking downstream from an object detector. They find that adversarial attacks targeting object detection need a success rate of at least 98% to break the visual perception pipeline.	✓	✓	✓	Though the attack is tested on YOLO, its real target is a downstream multiple object tracker. This makes it unsuitable for this thesis.

Cross-task Universal Perturbation: Zhang *et al.* [5] develop a black-box universal perturbation attack against object detectors. The perturbations are trained for each class and superimposed on each other to cause a classifier to misclassify all classes.	✓	✓	✓	Though, this attack is designed to have a high transference, Zhang *et al.* [5] operate with a different definition of black-box to this thesis. When generating adversarial samples they allow arbitrary input and reading the output in the black-box setting. In this thesis adversarial samples must be generated with no access to the model, in the black-box setting.
Discrete Cosine Transfer Attack: Kuang *et al.* [30] develop a boundary attack based on Discrete Cosine Transfer. It is a black-box attack that targets object detectors, specifically YOLOv3 and AWS Rekognition. The attack works on individual semantic objects rather than the whole image.	✓	✓	✗	Like the Cross-task Universal perturbation attack, this attack relies on arbitrary input and reading the output of the model under attack. Furthermore, it is designed explicitly to cause the model to misclassify. It does not attempt to alter bounding boxes. Most of the model used by this thesis are single class models, making this attack ineffective.

Category-wise Attack: Liao *et al.* [31] develop an attack that targets object detectors that don't rely on anchor boxes e.g. DenseBox, CornerNet and CenterNet. This is in contrast to YOLO, Single Shot MultiBox Detector (SSD) and Faster R-CNN which all use anchor boxes to localise and predict objects.	✗	✓	✗	Attack targets anchor-free models, as YOLO uses anchors, this attack is unsuitable.
Physical Adversarial Patch for Object Detection: Lee and Kolter [32] develop an adversarial patch targeting object detectors. The patch does not need to overlap with objects to suppress their detection. This results in prediction where the patch is the only object detected	✓	✓	✗	As a patch attack designed for physical deployment, it is unsuitable for this thesis.
Projected Gradient Descent on Object Detectors: Wang *et al.* [33] applies Projected gradient decent [34] to Region Proposal Network based object detectors e.g. Faster R-CNN [26]. Since the adversarial examples are generated on the total loss of Faster R-CNN, the detector will both misclassify as well as mis-localise.	✓	✗	✗	As the attack is targeting RPN based object detector, it is unsuitable to this thesis.

Adversarial Patch for Object Suppression: Wang *et al.* [35] develop an adversarial patch attack that suppresses detection when it overlaps an object. They show that the patch can successfully attack real-time object detectors through a physical patch.	✓	✓	✗	As it is a physical patch attack it is unsuitable for this thesis.
Half-Neighbor Masked Projected Gradient Descent: Zhang *et al.* [36] develop a white-box, PGD-based, attack against object detectors. Tested on YOLO and Faster R-CNN it finds a mask that lets it apply the adversarial perturbation to the pixels that matter most to the final detection. The attack is constrained by the l_0 distance metric, which results in an adversarial patch covering key areas of the image.	✓	✓	✗	The attack alters few pixels by localizing the perturbation on the key areas of the image. Otherwise, a potentially suitable attack.

Targeted Objectness Gradient Attacks: Chow *et al.* [1] develop a framework to attack and benchmark object detectors as well as a novel family of attacks. They use an iterative gradient approach to find the adversarial perturbation. The attacks work in different modes: untargeted, object vanishing, object fabrication, and targeted object-mislabeling.	✓	✓	✗	With public code and several object detector specific attacks implemented, this attack is suitable. The attack applies perturbations to the entire image using the L_∞ distance metric. Additionally, Chow *et al.* [1] test the attack in cross-resolution and cross-model scenarios, which gives easy comparison points for further transference attacks.
Lane-Keeping Assistance System Attack (LKAS): Sato *et al.* [37] develop a "dirty road" patch to attack LKAS.	✗	✗	✓	The attack targets a lane-keeping assistance system, not an object detector.

Table 3.1: Results from a literature review looking for an appropriate attack to study transference.

3.6.1 More details on TOG Attacks

The TOG [1] attacks allow for a number of different attacks. There is an untargeted attack whose success metric is any detection that is not correct. There is an object vanishing attacks which suppresses an object from being detected. The object fabrication attack adds false objects to the prediction. Targeted object-mislabeling causes the detector to predict the class of the attackers choice but keeps the bounding boxes.

The principal advantages of the Targeted Objectness Gradient (TOG), are a suite of different attacks all developed for object detectors. TOG attacks also work on both one-shot, regression-based detectors, as well as two staged, Region Proposal Network (RPN)-based ones. This makes the attacks potentially very representative. The attacks take inspiration from the universal perturbations which

have been applied to classifiers in the past [38]. This technique is similar to what Zhang *et al.* [5] did to make their attack more transferable. Chow *et al.* [1] explore black-box attacks through attack transferability. They look at cross-model transferability as well as cross-resolution transferability. Testing at different resolutions is done due to YOLO and Faster R-CNN allowing variable input resolutions. The test uncovers whether the adversarial example is robust enough to function after being resized and interpolated. They found the transferred attack was significantly weaker, but still reduced the mAP of the model by about half. Using SSD300 [8] as the source model and targeting YOLOv3, they reduced mAP from 83.43 to 56.87.

Chapter 4

Method

This chapter presents the research motivation, design and implementation.

4.1 Motivation

As classification, segmentation, and object detection has become increasingly advanced, they have seen deployment in safety critical contexts. Classification, and segmentation are deployed as part of a diagnosis tools. Object detectors are deployed as part of the vision system of autonomous vehicles. As reliance on these systems increases, it becomes important to determine the attack surface of the neural network.

When Szegedy *et al.* [3] discovered that neural networks have fundamental weaknesses, interest in adversarial attacks on classifiers grew rapidly. Attempts to make neural networks more robust to these attacks were found to be computationally expensive, and insufficient to protect against modern attacks [4]. Though attacks targeting classifiers can be modified to work on object detectors, as they essentially attack the classification portion of the detector, but they are not an ideal way to attack object detectors. Object detectors are different to classifiers in that, they are multi-task learners performing a more complex task than classifiers. Recently attacks have been developed that target object detectors directly, and therefore attack them much more successfully [1][39][17].

Chow *et al.* [1] and Wei *et al.* [17] investigate cross-architecture attacks, where the source and target architectures are different. The attack scenario this thesis explores however, is a black-box attack, where the attacker is not able to use the same dataset as the victim. This is a significant gap, as it is unlikely an attacker would have access to the dataset used to train a commercial autonomous vehicle. My study aims to find the level of exposure one can expect if an attacker tries to attack a model they have no direct access to, and do not have know important model details.

4.2 Research Questions

- RQ1. Is it possible to transfer black-box attacks across models, in particular, models trained on different data?
- RQ2. If the answer to RQ1 is yes, will datasets sharing images affect the transference of the attack?

4.3 Research Methodology

This section will explain how answers to the research questions were sought, how data was gathered and analysed, as well as describing the research paradigm of the thesis.

4.3.1 Research Strategy

To answer RQ1, a set of high performance object detection models are required. These are trained using YOLO - Darknet as this implementation produces real-time object detectors with leading performance [40]. One model is trained purely on private data to enable black-box attacks. Other models are trained on various custom datasets based on publicly available data. Then the performance of adversarial attacks using these models is measured in mAP. Adversarial samples are generated with different combinations of source and target models. The measured performance of these attacks will establish whether one can effectively attack the private model using public data.

To answer RQ2, the performance metrics of attacks between models with intersecting datasets, are compared to the metrics of models with disjoint datasets. In particular, we investigate whether there is a correlation between the relative size of the dataset intersection, and the performance of the transfer attack.

4.3.2 Data Analysis

We analyse data mainly through performance metrics from the object detection models. The primary performance metric of this thesis, mAP, is widely used due to it allowing researchers to express model quality with a single value. This allows the field to easily compare: architectures, datasets, as well as any techniques used to improve model performance. Adoption of mAP was spurred by Pascal VOC using it to rank object detectors. Following Pascal VOC, COCO also uses mAP as the official performance metric, though in COCO's case the metric is tweaked to reward precisely placed bounding boxes more than mAP at .5 Intersection Over Mean (IOU) does. In the context of object detection challenges like Pascal VOC and COCO, the challenge includes a benchmark dataset as mAP is highly dependent on the dataset. Unfortunately, the models produced for this thesis can not be benchmarked in that way as they are single class and two class detectors, not general object detectors.

In this thesis mAP is mostly used for internal comparisons. The performance on clean samples is compared to the performance on adversarial samples. Through internal comparisons, the effectiveness of the attacks can be established.

As performance is highly dependant on the dataset, each model produces multiple mAP scores. In addition to performance on clean samples, mAP is calculated for adversarial sets. Here, performance is dependent on both the initial clean sample performance, as well as the source model of the adversarial samples.

RQ1 is based on adversarial sample performance of the models compared to the clean sample performance. The primary case is the performance of the private model on samples generated by the models trained on the Singapore Maritime Dataset [41], and the COCO [19] dataset. RQ1 relies on quantitative analysis of model performance. First, models are trained, and their performance on clean samples is established. At this step, the goal is to train a model on publicly available data with similar performance to the model trained on private data. Once the performance of the models has been established on a common benchmark dataset, we generate adversarial sets using all the models and the same benchmark dataset. The benchmark dataset will be composed of private data as this data is closest to the intended domain of the private model. Measuring the performance of the models on the adversarial sets, will make it possible to compare performance on clean samples, to white-box attacks, and black-box attacks. Attack performance is inferred from the drop in mAP, between clean, and adversarial sets. The performance of transfer attacks, is inferred from the difference in mAP drop between white-box and black-box attacks.

RQ2 is based on attack performance between the various public models. As these models are trained on various compositions of public datasets, comparisons can be drawn between the intersection of the datasets and the performance of the adversarial samples. The intersection between the datasets is measured as the number of images which exist in both datasets, divided by the size of the dataset. This is similar to Intersection Over Mean (IOU), but importantly, IOU is a symmetric measure, i.e. $IOU(A,B) = IOU(B,A)$, while the measure we use reflects the difference between attacking a model trained on a subset of the model's own training set, and attacking a model trained on a superset of the model's training set.

4.3.3 Research Paradigm

The goal of this thesis is to uncover features of object detectors, while meeting quality criteria of objectivity, reliability, internal validity, and external validity. A fundamental assumption of this thesis is that comparing the performance of object detectors on clean and adversarial samples gives insight to the inner workings of the neural network. Additionally it is assumed that, these insights are based on objective observations, which can be repeated by third parties. The thesis does not explore social processes or seek to understand the social context of these systems. Instead quantitative data analysis is conducted. Therefor, this thesis is a positivist

endeavor.

Chapter 5

Implementation

This chapter details how experiments were implemented. Including, the training of models, measuring performance, generating adversarial samples, and attack the models.

5.1 Attack Scenario

The attack scenario considered in this thesis, is modeled to establish a worst-case scenario for an autonomous ferry under attack, where the attacker does not have access to the target model, or key information about the model. The attacker is able to apply adversarial noise directly to the input, but is not able to generate adversarial samples using the target model. The attacker uses the same architecture as the target model. This effectively establishes a worst-case scenario as no cross-model attack could be stronger than the attack with the same source and target architecture. Additionally, there are not many different real-time object detection architectures suitable for autonomous vehicles. YOLO is the best performing architecture on the Common Objects in Context (COCO) dataset, which makes it reasonable for an attacker to guess that a YOLO model might be used.

A common way to train a model to act as a surrogate for a transfer attack, is to train the model on the labels output from the target model instead of on the groundtruth. Though, this produces a surrogate model that has similar performance to the target model, this technique did not fit the attack strategy. If an attacker was able to read the model output form arbitrary inputs they could simply train the attack to target the model directly.

5.2 RQ1

5.2.1 Object Detection

To answer RQ1, an object detector needed to be selected. The object detector would serve as the target model, as well as the surrogate model. Using the same

architecture for the attacker and the target would isolate cross-dataset performance, and give a worst-case scenario. The primary reason for choosing the YOLO detector came from DNV. DNV's Revolt project, which this thesis is produced in association with, is investigating the challenges of developing an autonomous ferry. As part of this work, a sensor fusion suite was produced which includes a YOLO model [42]. Beyond this, YOLO is interesting because it is the highest performing real-time object detector on the COCO dataset [40]. YOLO's high performance at real-time speeds, together with implementations in several different neural network frameworks, makes it well suited to research on object detectors, and autonomous vehicles.

However, the primary implementation of YOLO is YOLO - Darknet which is written in c and CUDA. This makes it harder to do research with, as Darknet does not have the community support large neural network frameworks like TensorFLow and PyTorch have. Therefore, a large portion of research on YOLO is done on implementations in other frameworks, making models trained on Darknet harder to use in experiments. This issue was solved by converting the Darknet trained weights to weights a TensorFLow implementation could use. However, the Darknet and TensorFLow models do not make identical predictions, which causes the adversarial sample produced with the TensorFLow model to be less effective on the Darknet model. The effectiveness of an adversarial sample is heavily dependant on how a model makes predictions. Therefore, this impacts the attack performance. Though this effect is only seen in some circumstances, and changes based on the source and target model.

Another problem which came from using two different YOLO implementations, was that YOLO operates with a fixed internal resolution, and the two YOLO implementations dealt with this in different ways. The Darknet implantation resized the image to match the internal resolution before making predictions. The TensorFLow implementation, however used letterboxing which preserves the aspect ratio, before scaling the image. Images that were letterboxed and scaled were not suitable inputs to the Darknet model. The model was unable to make predictions on these images altogether. To solve this issue, the TensorFLow implementation needed to be changed to resizing the images in the same way the Darknet model does.

5.2.2 Generating Adversarial Samples

Several attributes were considered when looking for an attack to generate adversarial examples. The attack should be:

- Designed to target object detectors.
- Able to target single stage, regression based detectors like YOLO.
- Have a public implementation that enabled experimentation.
- Produce adversarial samples by adding adversarial noise to the entire image.

Although research into adversarial examples has been enthusiastic since Szegedy *et al.* [3], a large portion of the research has been focused on classifiers. Attacks

designed for classifiers can be applied to object detectors. In fact, early on in this project, an implementation of the Carlini and Wagner [4] attack was modified to attack YOLO. The problem is that object detector solve a more complex problem than classifiers, which means an attack designed for classifiers can not be optimal [1]. Furthermore, Faster RCNN is a popular object detector in research, but Faster R-CNN is commonly implemented as two networks, a region proposal network and Faster R-CNN as a detector network. Attacks targeting Faster R-CNN often target the RPN, which makes them unsuitable to attack YOLO which is a single network, single stage detector. The final attack type not under consideration are various techniques that produce adversarial patches. Patches are very relevant to autonomous vehicles, as an attacker could place patches in the environment to fool any detector that observed it. This scenario was not considered as the practicality of conducting rigorous experiments on patches placed in a marine environment seemed low. Furthermore, such experiments would be rather premature, as a suitable baseline performance for adversarial samples directly fed to the model did not exist.

These considerations led to the Targeted Objectness Gradient (TOG) [1] family of attacks. TOG was developed specifically to attack object detectors. Additionally, Chow *et al.* [1] attacked YOLO with it, even attacking YOLO with cross-model attacks. Since TOG is implemented in TensorFLow the darknet models needed to be converted. To ensure compatibility, the same TensorFLow implementation of YOLO was used to convert the models, as is used internally by TOG.

The TOG attacks are implemented in Keras/TensorFLow 1.X which means several steps are required before models trained using Darknet can be used. Darknet saves weights in a .weights file. This file is interpreted using a separate configuration file. TensorFLow saves weights in the .h5 format which also contains configuration information. To use Darknet models with TensorFLow, they need to be converted, Fortunately public repositories have code available to convert models from .weights to .h5. The conversion software was found at keras-yolov3. As this is a popular implementation of YOLO in TensorFLow, it is also used by TOG to interface with the model under attack.

However, the TensorFLow implementation of YOLO deals with the fixed internal resolution by resizing and letterboxing the image to preserve the aspect ratio. This is different to the Darknet implementation which resizes the input image without preserving the aspect ratio. Since the TensorFLow implementation squashes the input image down to 416x416 and adds black bars to preserve the aspect ratio, the result was TOG outputting small letterboxed images, which the YOLO models had trouble making detections on. To output images the target model would be able to make predictions on, the internal resolution of the source model was increased to 1920x1920. 1920 being the horizontal resolution of the images in the validation set. Since the images were no longer shrunk, a simple script could crop away the letterbox, and leave the sample with the same resolution as it had initially. Since the adversarial perturbation was applied to the entire sample, including the letterbox, the cropping script had to use a tolerance

to locate the edge of the actual sample and crop away the rest. The code is shown in listing 5.1

Code listing 5.1: The script to crop black bars from an image. source: `https://codereview.stackexchange.com/a/132934`

```
def crop_image_only_outside(img, tol=80):
    # img is 2D or 3D image data
    # tol  is tolerance
    mask = img > tol
    if img.ndim == 3:
        mask = mask.all(2)
    m, n = mask.shape
    mask0, mask1 = mask.any(0), mask.any(1)
    col_start, col_end = mask0.argmax(), n - mask0[::-1].argmax()
    row_start, row_end = mask1.argmax(), m - mask1[::-1].argmax()
    return img[row_start:row_end, col_start:col_end]
```

Unfortunately, it turned out that generating the adversarial sample with a surrogate model using a much higher resolution, then cropping the letterbox, left the adversarial sample ineffective. When the Darknet model made detections on the adversarial samples, they achieved the same performance they did on clean samples. Once this was discovered it became clear that this method would not work. To figure out how to solve this issue the code for YOLO - Darknet and TensorFLow was examined. It turned out that the reason the TensorFLow implementation used letterboxing, was because this used to be the default behavior of the Darknet implementation as well. Though it would be possible to return the Darknet implementation to the older technique, this would require that all the model were re-trained. To avoid re-training all the models, TOG was updated to add resizing functionality similar to Darknet. The Pillow library was already in use to process images, making the resizing code simple to add once it was clear that this was the best solution. The resizing code is shown in listing 5.2

Code listing 5.2: The resizing code added to TOG

```
from PIL import Image
import numpy as np

def image_resize(image, size=(832, 832)):
    new_image = image.resize((size), Image.BICUBIC)
    new_image = np.asarray(new_image)[np.newaxis, :, :, :] / 255.0

    return new_image
```

Once it was clear that adversarial samples could be generated with TOG and it would have an effect on the performance of the Darknet model, scripts were written to automate the process of generating adversarial sets. When generating adversarial samples, all models are used to generate an adversarial set. Once all the adversarial sets are produced, every model is benchmarked on each adversarial set. This produces a mAP value for each pair of models.

Cross-model attacks are not the focus of this thesis. However, there is an element of this to the results. This is due to the TOG attack not being able to directly

attack a Darknet model. Though this brings attack performance below optimal, it would require significant development effort to enable TOG to attack Darknet models directly. It is unlikely an attacker would invest this effort when they do not know for sure that they are attacking a Darknet model. Keep in mind that TOG can directly attack SSD and Faster R-CNN, making it a good choice to attack the major real-time object detectors.

5.2.3 Measuring Performance

To answer RQ1, data analysis was largely carried out by entering the output from the Darknet mAP tool into a spreadsheet and using that to group and visualise the data. The spreadsheet was used to compare model performance, calculate correlations, and make graphs.

The primary way model performance was measured, was through the testing tools implemented in Darknet. These tools calculate mAP using the validation set for the model. The validation set in Darknet is defined in a configuration file and so it is trivial to test a model on a different validation set. This enables the use of the primary benchmark set.

While the primary challenge metric COCO uses is mAP averaged across 10 intersection over union steps, this thesis reports results using a single IOU stage, $IoU = .50$. The 10 IOU steps COCO uses reward object detectors that are able to place very precise bounding boxes. This is a less interesting metric for real-time object detectors that need to make compromises for added speed. Furthermore, the smallest IOU steps demand performance beyond human which is less relevant to autonomous vehicles [9].

5.2.4 Data Sources

In order to answer RQ1, i.e. to investigate adversarial robustness, one must first have access to multiple representative object detectors. This section describes the process of configuring and training models which could be a component of a sensor fusion for autonomous ships.

Multiple datasets were compiled over the course of the research. The central goal was to obtain a dataset that would train a representative object detector for autonomous ships. Additionally, another dataset was needed to train the surrogate model which would enable black-box attacks. To demonstrate black-box attacks and answer RQ1, the two datasets should be disjoint, no images appearing in both datasets. Furthermore performance between the models should be as similar as possible. This is due to the assumption that this would demonstrate the attack transferability.

Weights trained as part of a previous thesis [43] were made available, together with the dataset used to train them. After calculating the mean Average Precission (mAP) of the model, it became clear that the weights didn't have the desired performance profile. The decision was made to train a new model. First the dataset was re-annotated to ensure consistent annotations. This dataset consisted of over

600 images taken in maritime environments. Primarily, The Trondheim fjord and Høvik which is on the Oslo fjord. The images are of different resolutions and of boats and ships of different sizes and at different distances. Most of the images are taken from the shore overlooking the Oslo fjord. The crafts passing by are largely small pleasure craft, and kayaks. They appear at medium to long distance, which means they are often quite small in the image. However, they are visible due to the contrast with the water around them. Figure 5.1 shows a sample of the images.

Figure 5.1: Sample images from the dataset shared from Grini [43]

DNV, who made co-supervisors available for this thesis, also shared dataset they had produced which could be used to train maritime object detection. It consists of around 4700 frames captured from the "Hurtigruten minute by minute" TV show. The rights to these images were licensed from NRK, and then all the images were annotated. The various watercraft Hurtigruten encounters, were placed in 30 categories. All the images were taken from the bow of the ship. These images all have the same resolution, being sampled from the camera overlooking the ships bow. They feature a wide variety of small to medium sized craft. Most of these pleasure craft sailing ahead of the ferry. There are images in different lighting conditions, and some taken during inclement weather. Figure 5.2 shows a sample of the images. Like the images from Grini [43], most craft will have water as the background.

Figure 5.2: Sample images from the dataset shared from DNV

To properly simulate a black-box attack it was necessary to train a model for the adversary. This model, called the source model acts as a surrogate to the target model. To avoid giving the adversary an advantage, this model would be trained on a dataset disjoint to the target model. Therefore, models were also trained on publicly available datasets. One of the public sources used was the Singapore Maritime Dataset [44]. The Singapore Maritime Dataset (SMD) is quite different to the data shared from Grini [43]. It consists of several videos taken in the Singapore harbour. Some videos are captured on board, some are from the shore.

For training only the visible light images were used, though there are also Near-infrared images in the dataset. This dataset is not as varied as the data from DNV as the videos are much shorter than the TV show. There is not much variety in the dataset either, mostly large freight ships. Though, there are also some tugboats, pleasure craft and kayaks. Figure 5.3 shows a sample of the images. The water craft are broken down into 7 different classes.

Figure 5.3: Sample images from SMD

The third data source used is the COCO dataset. This dataset is a large dataset used to benchmark and rank object detectors. There are annual challenges with different goals, including image segmentation and keypoints (poses). The dataset consists of over 200000 labeled images divided into 80 categories. As the name implies, a focus for this dataset is presenting the objects in context. For the images in the boat class this results in more watercraft on the beach or docked in the harbour, than is found in the other datasets. Figure 5.4 shows a sample of the images. The resolution of the images is not uniform, and the images are extremely varied in lighting composition, and environment. These attributes makes it much harder for an object detector to achieve a high mAP on COCO.

Figure 5.4: Sample images from COCO

5.2.5 Preparing Datasets to Train the Target Model

To train YOLO models on these data sources generally required some setup. YOLO needs images to be divided into two folders, one for training, and one for valid-

ation. Annotations of the correct format must be in the same folder with each annotation file sharing the name of the corresponding image, but with a .txt extension. Aside from the Grini [43] data, which was re-annotated as part of the research, each data source needed to have it's annotation format converted to the Darknet format.

The first data which was prepared for training was the Grini [43] data. In addition to the dataset weights trained on that data was shared, though these weights ended up not being used. After calculating the mean Average Precission (mAP) of the model it became clear that the weights didn't have the desired performance profile. The decision was made to train a new model. First the dataset was re-annotated to ensure consistent annotations. To annotate the images the program LabelImg [45] was used. LabelImg allows the user to define semantic classes and draw bounding boxes around these classes in images. Additionally, LabelImg supports YOLO annotations as well as Pascal VOC and Create ML. See an image of the interface in figure 5.5.

Figure 5.5: The interface for LabelImg [45]

Once the Grini data was annotated the goal became to try to augment the data as approximately 600 images is not enough for a high performance object detector. Since Grini [43] found the best performing model was trained on two categories: buildings, and boats, the decision was made to keep these two categories for the first model. As the Hurtigruten dataset consisted of about 4700 images it would be impractical to try to annotate these images, but buildings were not labeled in the Hurtigruten dataset. To solve this, images from the Hurtigruten dataset would be culled before it was merged with the Grini dataset. First, all watercraft classes were merged to a single boat class. Then, all the images with prominent buildings were removed. The final merged dataset included over 3800 images which were split 70/30 into training and validation datasets. 70/30 is the recommended spilt

for training YOLO models on Darknet and was used for all the models. The resulting dataset is quite representative of recreational boating in Norwegian fjords, but lacks the large ships one might find in busy ports. The validation portion of this dataset is used as the primary performance benchmark for this thesis.

Due to not being happy with the performance of the first model, another dataset was compiled from the Hurtigruten dataset. It used all the images from the Hurtigruten dataset, after some duplicates and other bad data was removed. Once more all watercraft classes were merged to a single boat class. This was done for three reasons. The first is that, to make comparisons between models easier, they should be as similar as possible beyond training on different data. The second is that COCO uses a single boat class, and incorporating this data in a model was one of the goals. The third reason is that it was assumed that the model would achieve higher performance if did not need to separate different watercraft from each other, only identify and locate them. The hurtigruten dataset was annotated using Extensible Markup Language (XML) which needed to be converted to YOLO annotations before the dataset could be used. The code used to convert annotations is shown in listing 5.3 and listing 5.4. Finally, since the images were split up by episode, recall that this dataset is based on videos from a TV show, a script was written to collect all the images with their corresponding annotation file, and copy them to train and validation directories with a 70/30 split. Once this was all done, the dataset consisted of about 4700 images.

Code listing 5.3: Code extract that converts multi-class xml to single-class yolo

```
def convert_annotation(dir_path, output_path, annotation_path):
    basename = os.path.basename(annotation_path)
    basename_no_ext = os.path.splitext(basename)[0]

    in_file = open(annotation_path)
    out_file = open(output_path + basename_no_ext + '.txt', 'w')
    tree = ET.parse(in_file)
    root = tree.getroot()
    size = root.find('size')
    w = int(size.find('width').text)
    h = int(size.find('height').text)

    # Modified to assign every bbox to class 0 which is 'boat'

    for obj in root.iter('object'):
        # difficult = obj.find('difficult').text
        im_cls = obj.find('name').text
        if im_cls not in classes:
            print('skipped: ', im_cls)
            continue
        cls_id = 0                  #classes.index(cls)
        xmlbox = obj.find('bndbox')
        b = (float(xmlbox.find('xmin').text),
        float(xmlbox.find('xmax').text),
        float(xmlbox.find('ymin').text),
        float(xmlbox.find('ymax').text))
        bb = convert((w,h), b)
        out_file.write(str(cls_id) + " " +
        " ".join([str(a) for a in bb]) + '\n')
```

Code listing 5.4: The conversion step called in 5.3

```
def convert(size, box):
    dw = 1./(size[0])
    dh = 1./(size[1])
    x = (box[0] + box[1])/2.0 - 1
    y = (box[2] + box[3])/2.0 - 1
    w = box[1] - box[0]
    h = box[3] - box[2]
    x = x*dw
    w = w*dw
    y = y*dh
    h = h*dh
    return (x,y,w,h)
```

5.2.6 Preparing Datasets to Train the Attackers Model

With a performant model trained on the Hurtigruten dataset, it was time to train a model on a disjoint dataset, which could act as surrogate in black-box attacks. The data source chosen was the SMD [41] as this dataset seemed to have a lot in common with the Hurtigruten dataset. It was partially collected from the bow of a ship and initially consisted of videos with consistent resolution. To train a Darknet model on SMD, it was necessary with some preprocessing. The first step was to extract individual frames from the videos and save them as images. Extracting the frames and creating corresponding annotation files was a fairly large challenge. However, the hardest parts had already been solved by Bontzorlos [41]. Modifying this code, and then using the XML to YOLO converter, as shown in listing 5.3, made the data fairly easy to use. Though one could extract every frame from the video as its own image, the decision was made to extract every fifth frame from the dataset. This was so that the final dataset would be similarly sized to the dataset used to train the target model. Additionally, successive frames are very similar and so add little to the training model. Even using every fifth frame this became an issue. Final size of the dataset was about 5100 images. Once images had been extracted, the annotations also needed to be converted to an appropriate format. This was a two stage process where the .mat files used by the videos were converted to XML, which were then converted to You Only Look Once (YOLO) annotations. Additionally, in this step all watercraft classes were merged to the single class, boat, and any other classes were discarded. This was in order to bring the surrogate model in line with the target model.

The initial surrogate model trained on the SMD had very poor performance on the benchmark dataset. It achieved a mAP of 0.39. Therefore, a new surrogate model was trained incorporating the Grini [43] data. As the primary target model now only used the Hurtigruten dataset, the Grini data was free to use to train surrogate models. Even though the Grini data was being used, only the boat labels were desired to be trained on. To solve this a python script was written which would loop through the annotations and discard all but the boat annotations. The code is shown in listing 5.5.

Code listing 5.5: The conversion step called in 5.3

```
import os

def fix_annotation():
    for filename in os.listdir(os.getcwd()):
        if filename.endswith('.txt'):
            with open(filename, "r") as f:
                lines = f.readlines()
            with open(filename, "w") as f:
                for line in lines:
                # The boat class is designated 0
                    if line.strip("\n")[0] != '1':
                        f.write(line)

os.chdir('obj')
fix_annotation()
print('obj processed')
os.chdir('../test')
fix_annotation()
print('test processed')
```

Though the model using SMD and data from Grini showed improvements, models were trained using COCO data as well to see if these would perform better on the benchmark dataset. A model was trained incorporating the COCO data some of the SMD data, and the Grini [43] data. Finally, a model was trained merging all data available. This model, trained on a superset of the data sources would be a good point of comparison for performance.

5.2.7 Training Configuration for the Target Model

YOLO operates with a fixed internal resolution which has to be a multiple of 32. In the Darknet implementation images fed to the model are resized to fit the internal resolution. The initial model was trained at 416x416, for 6000 iterations. Bochkovskiy *et al.* [40] suggests the number of training iterations should be 2000 iterations for each class, but no fewer iterations than training images, and no fewer than 6000. The default training resolution of YOLO is 608x608, but a lower training resolution allows for faster training. With the initial model to compare to, attempts were made to improve model performance. Previous work [40] suggested increasing the internal resolution would increase precision. Increasing precision would lead to an increase in mAP, the primary performance metric. Therefore, a model was trained on the same dataset at 832x832. The higher resolution model was also trained for 9000 iterations to make sure performance was as high as possible. Instead of an increase in precision, an increase in recall was observed accompanied by a decrease in precision, leading to an overall lower mAP. Studying the data, it was clear that the higher resolution model was having trouble with the building class. The data is shown in figure 5.6. Therefore a high resolution model was trained on the entirety of the Hurtigruten dataset without merging inn the Grini [43] data. This model achieved higher performance than the initial low resolution model trained on the merged dataset, making it the primary target model for this thesis. The model achieved a mAP of 72.41.

```
 detections_count = 7415, unique_truth_count = 2252
class_id = 0, name = boat, ap = 62.50%           (TP = 1443, FP = 538)
class_id = 1, name = building, ap = 56.04%       (TP = 65, FP = 29)

 for conf_thresh = 0.25, precision = 0.73, recall = 0.67, F1-score = 0.70
 for conf_thresh = 0.25, TP = 1508, FP = 567, FN = 744, average IoU = 49.97 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.592695, or 59.27 %
Total Detection Time: 58 Seconds
```

```
 detections_count = 13164, unique_truth_count = 2252
class_id = 0, name = boat, ap = 68.41%           (TP = 1619, FP = 693)
class_id = 1, name = building, ap = 40.82%       (TP = 66, FP = 94)

 for conf_thresh = 0.25, precision = 0.68, recall = 0.75, F1-score = 0.71
 for conf_thresh = 0.25, TP = 1685, FP = 787, FN = 567, average IoU = 48.09 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.546144, or 54.61 %
```

Figure 5.6: Comparison between training M1 at 416x416 (top) and 832x832 (bottom)

5.2.8 Training Configuration for the Attackers Model

As the goal was to train a surrogate model with similar performance to the target model, all subsequent models were also trained at 832x832. Some models were trained for more than 9000 iterations, as the training set was bigger. Training iterations should be at least equal to the number of images in the dataset. Therefore, increasing training iterations was important.

5.3 RQ2

To answer RQ2, the dataset intersection was calculated using a script that accessed all the datasets and compared the filenames pairwise. This returned the intersection between each pair of datasets. The code is shown in 5.6.

Code listing 5.6: The code that finds the dataset intersection.

```python
from glob import glob

def getImagesInDir(dir_path="data/test"):
    image_list = []
    files = (
        glob(dir_path + "/*.jpeg")
        + glob(dir_path + "/*.jpg")
        + glob(dir_path + "/*.png")
    )
    for filename in files:
        image_list.append(filename.split('/')[-1])

    return image_list
```

```
base_path = "/cluster/work/alexamst/"
dir_list = [
"yolo-train",
"yolo-train/pure_hurtig",
"yolo-train/smd_merged",
"boat_coco",
"super",
"improved_coco"
]

for x_directory in dir_list:
    for y_directory in dir_list:
        if x_directory == y_directory:
            continue
        dir_x = getImagesInDir(base_path + x_directory + "/obj")
        dir_y = getImagesInDir(base_path + y_directory + "/obj")
        intersect = set(dir_x).intersection(dir_y)
        print(str(round((len(intersect)/len(dir_x))*100, 2)) + "%" "␣of␣" +
        x_directory.split('/')[-1] +
        "␣intersects␣with␣" + y_directory.split('/')[-1])
```

5.4 Hardware

This section explores the different hardware setups used to train models and generate adversarial samples. Several training setups were attempted over the course of this project. As all operations: training models, calculating mAP, and generating adversarial samples involve manipulating images, a powerful graphics card is necessary.

5.4.1 Local

To begin with a local setup on a personal computer was attempted. The computer had an Nvidia GTX 970 graphics card installed. The first challenge was building Darknet from source and correctly installing CUDA drivers. As this was difficult on Windows a Linux partition was installed to do computation tasks. Though detection was fast and convenient on the local setup, training was much harder. Training takes a long time on GTX 970, and using the computer for other things while it trains a model is inconvenient. Not to mention trying to sleep in the same room while the model trains. This setup was quickly abandoned due to lacking hardware resources. With the local setup proving untenable, training in the cloud would be preferable.

5.4.2 Colaboratory

Google Colab is a service that lets one run, and collaborate on, jupyter notebooks on remote hardware for free. Though this gives free access to powerful hardware, the service is not quite designed for training large models. The service is meant to let researcher or students carry out many short computations. There is a premium

tier meant for training large models and other tasks that require powerful hardware for a longer time, but this is only launched in the US. To prevent one user from tying up too many resources, operations are terminated after 12 hours. This however is a maximum limit and the actual limit will change based on the load the service is experiencing, and how much resources one has used lately. GPU resources are especially limited. After long operations one will also be prevented from starting new GPU sessions. Under these limitations a model was trained at 416x416, but trying to train another was unsuccessful due to constant disconnections from having used to many GPU resources.

5.4.3 NTNU IDUN Computing Cluster

The final migration brought the project to the execution environment which would actually train all the models, generate the adversarial samples, and run mAP calculations. Idun [46] is a computer cluster which students at the department of computer science can apply for access to. The cluster has more than 70 nodes and 90 GPGPUs. Each node contains two Intel Xeon cores, at least 128 GB of main memory, and is connected to an Infiniband network. The cluster is accessed through SSH which lets one carry out file management, code compilation, and queuing jobs using the Slurm Workload Manager. Under training, a single Nvidia V100 GPU was used. This allowed training a new model to complete in approximately 35 hours. Generating adversarial sets with all the models would take about 7 hours, while benchmarking all the models on all the adversarial sets was roughly 3 hours.

Chapter 6

Results

This chapter presents the results of the experiments. The composition of datasets is detailed, and the performance of the models derived form the datasets is described. All the code used for generating adversarial samples can be found here: `https://github.com/alexmstaff/TOG/`

6.1 RQ1: Is It Possible To Generate Adversarial Samples With One Model and Attack a Model Trained on Different Data?

Section 6.1 details the performance of black-box attacks on YOLO using the TOG family of attacks. To begin with, datasets and their corresponding YOLO models are presented. Their initial performance on clean samples form a baseline to compare the adversarial attacks to. All mAP data is mAP .50. This is because, precisely placed bounding boxes are not a focus of this thesis. Additionally, it makes the data more comparable to other papers e.g. on object detection. A more in depth discussion of mAP can be found in chapter 2.5.

6.1.1 Dataset Composition

The data sources available were quite different. An important difference is the aspect ratio of images in the dataset. YOLO uses a fixed internal resolution and resizes the images prior to training and detection. This results in objects in the image changing appearance. When one trains the model on a dataset with the same aspect ratio as the images used to test the model, this effect disappears as the model learns how objects look when they are warped. The issue arises when detecting on images of an aspect ratio not seen in the training set. In this case recall and confidence drops.

The images shared from Grini [43] were compiled to train an object detector for the ReVolt project. Most images are 1280x720, and captured in Høvik, Norway. These images are also largely of smaller craft and kayaks. To ensure consistent

labeling, these images were re-annotated. No other data source was re-annotated as the datasets were far to big. This prevented the training of a larger two-class detector, as none of the other data sources annotated buildings.

The Hurtigruten data was shared from DNV and contains images captured from the bow of the Hurtigruten ferry as it travels up the Norwegian coast. These images are captured at a resolution of 1920x1080. The watercraft in the images are largely small to medium sized pleasure craft.

This is in contrast to the Singapore Maritime Dataset, where the images were collected in the Singapore harbour. These images are dominated by large freight ships. Due to this, models trained on the Singapore data struggle making accurate predictions on the Hurtigruten dataset and vice versa. The Hurtigruten images and the Singapore images share the same resolution, but the difference in subject is the dominant factor in this case.

The () dataset is the most diverse dataset. Both in terms of resolutions and watercraft. The images have various resolutions from 280x500 to 640x640. Additionally, watercraft appear in many different contexts, not simply on water. This is important for a multi-class detector which needs to separate between several classes. For a single-class detector designed to work in a specific context however, images more closely resembling the intended domain are desirable.

Data Source	Description	Size
Grini data	Images shared from Grini [43]. Re-annotated for this thesis, two classes: Buildings, boats.	622
Hurtigruten	Data shared from DNV. Collected from the show "Hurtigruten minute by minute". Initially separate classes for different watercraft, but these are are combined to a single "boat" class for this thesis.	3740
SMD	The Singapore Maritime Dataset [44] Consists of a set of high definition videos taken in the Singapore harbour. Initially separate classes for different watercraft, but these are combined to a single "boat" class. As the dataset consists of videos, significant preprocessing must be carried out before one can train Darknet models on the dataset. To build a set of images, every 5 frame from the videos was used.	3590
COCO	The COCO [19] dataset consists of over 200K labeled images divided into 80 classes.	5123
Total		13075

Table 6.1: Data sources with their description

Numbers in Table 6.2 only include images in the training set, images in the

validation set are excluded. Aside from D5, datasets were kept at similar sizes. SMD would be very large if every possible image was used. As well as being much larger than the other datasets, consecutive frames are very similar and would therefore not add a great deal to the model.

Dataset ID	Source Datasets	No. Images	Total Size
D1	Grini data	497	3048
	Hurtigruten	2551	
D2	COCO	3025	5602
	Grini data	517	
	SMD	2060	
D3	Grini data	517	4107
	SMD	3590	
D4	Hurtigruten	3573	3573
D5	COCO	3025	10705
	Hurtigruten	3573	
	Grini data	517	
	SMD	3590	
D6	COCO	5123	5640
	Grini data	517	

Table 6.2: Assembled datasets with their composition

In Table 6.3, the Validation column shows the performance on the validation set associated with the training set for the model. The Benchmark column however, shows the performance on the dataset used to gauge performance between models, and to generate adversarial sets. The Benchmark dataset is the validation set associated with D1. The reason M1 does not have the same performance on the validation and benchmark datasets is that in the benchmark column, the images have been resized. To generate adversarial samples the input image must match the model dimensions, but annotations are not corrected for this difference. A technique to correct the annotations was not explored due, to the mAP difference being to small to affect data analysis. M2 and M4 are highlighted, as these models are the primary models used to answer RQ1. M4 is trained on private data, and M2 is the highest performing model trained on public data. For context on the performance numbers, the model with the highest mAP .5 on the 2017 COCO bounding box challenge (the latest one held), achieved a mAP of 72.9. This number should not be directly compared with numbers in this thesis, but knowing the state-of-the-art performance of a multi-class object detector, might be helpful when comparing the performance of the models in this thesis to each other. The reason these numbers are not comparable is that they were not achieved on the same dataset. Furthermore, the performance of a single class model is not comparable to an 80 class model. Finally, the model that achieved the high score was not a real-time model.

Model ID	Dataset	Validation	Benchmark
M1	D1	47.96	45.75
M2	D2	78.28	26.32
M3	D3	87.06	10.36
M4	D4	70.8	68.29
M5	D5	85.35	74.56
M6	D6	59.84	26.39

Table 6.3: Performance of models on clean samples

6.1.2 Attack Performance

Unless otherwise stated all adversarial samples are generated with an epsilon of 8. 8 is the default value for the TOG attacks, and using this value makes the numbers comparable to the data gathered by Chow *et al.* [1]. Epsilon represents the maximum change allowed to any point using an L_∞ distance metric, see Chapter 2.4.3. For an image, epsilon represents how much one can change a single pixel with no limit on the amount of pixels changed. Since a greyscale pixel can hold the values from 0 to 255, an epsilon value of 8 means no pixel is changed more than $\frac{8}{255} = 0.031$ [4].

```
 detections_count = 11890, unique_truth_count = 10820
class_id = 0, name = boat, ap = 87.06%          (TP = 8796, FP = 481)

 for conf_thresh = 0.25, precision = 0.95, recall = 0.81, F1-score = 0.88
 for conf_thresh = 0.25, TP = 8796, FP = 481, FN = 2024, average IoU = 76.97 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.870613, or 87.06 %
```

Figure 6.1: A detailed view of the output from the Darknet mAP measurement tool

Table 6.4 shows the mAP achieved on the adversarial set generated by M2. The Target Model column shows which model made detections on the dataset. White-box attacks, where the source and target model are the same, are highlighted in grey. Source model, refers to the model used to generate the adversarial set, think, attacking model. The Clean column is the model performance on the unperturbed, but resized set. This is to ensure that the only effect being compared, is the adversarial perturbation. The Untargeted column, refers to the model's performance on images perturbed by the TOG untargeted attack. Similarly, the Vanishing column, refers to the model's performance on the TOG Vanishing attack. An explanation of these attacks can be found in chapter 2.4.4. It is apparent from this table that transference is very low. Although the white-box was very effective, M4 only took a minor hit to its mAP. An important aspect of these attacks is that beyond testing attack performance with different source and target models, there is also an element of cross-model transference to the attack. This is a consequence of generating samples using the TensorFLow - YOLO model, and detecting with

the Darknet - YOLO model. An example of the difference in the detection between the two models can be seen in Figure 6.5c. The figure shows that, although detection is similar in broad strokes, several bounding boxes are placed in different locations. Several bounding boxes that appear in the TensorFLow detection, are missing form the Darknet detection. That is an indication that the perturbation didn't fully transfer.

Table 6.5 shows the mAP achieved on the adversarial set generated by M4. In the table we see that only the untargeted attack was successful in white-box mode. Neither attack transfer to M2 effectively.

Target Model	Clean	Untargeted	Vanishing
M2	26.32	4.63	3.27
M4	68.29	61.62	63.55

Table 6.4: Performance on adversarial samples generated by M2

Target Model	Clean	Untargeted	Vanishing
M2	26.32	24.4	25.17
M4	68.29	16.94	57.61

Table 6.5: Performance on adversarial samples generated by M4

6.1.3 Attack Performance Compared to Epsilon

D4, used to train M4, has no intersection with D2, D3, or D6. This allows black-box transfer attacks, where the attacker does not have access to the dataset used to train the victim detector. Figure 6.2 shows the mAP achieved by the models on adversarial sets generated by M2 with increasing epsilon values. At the left extreme we see the clean set performance. The largest mAP change is from clean to 8 epsilon on M2. As this is the white-box attack, this is to be expected. Curiously, though mAP was not reduced to 0, increases in epsilon did not decrease performance further. Looking at the line for M4 we see a steeper slope between 8 and 32 than between 0 and 8. This indicates that the attack transference is poor while epsilon is small, but once epsilon is large enough the attack strength improves proportionally with epsilon. Attacking M4 effectively with M2 requires a quite high epsilon, but by epsilon=24 the mAP of M4 is reduced to less than half.

On the other side, attacks generated by M4 attacking M2 show worse transference. Shown in Figure 6.3. An epsilon in the 20-30 range, is quite high in comparison to the white-box attacks. However, when attacking object detectors, an epsilon of 20 is in line with other efforts [5][22]. The white-box attack stands out as mAP decreases much faster than for any other model. In contrast to the white-box attack in Figure 6.2, the mAP of M4 is reduced to 0 by the white-box attack

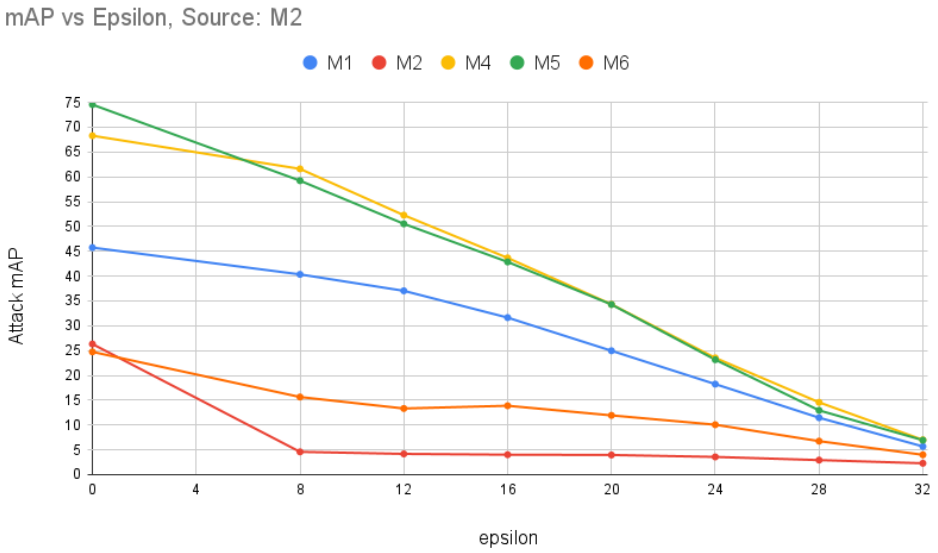


Figure 6.2: The black-box and white-box performance plotted against epsilon. The adversarial set is generated by the M2 model

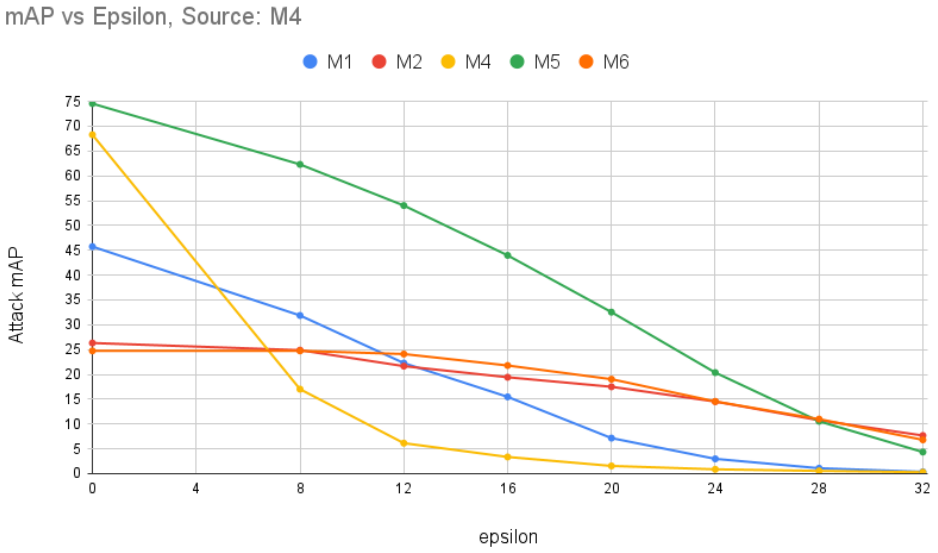
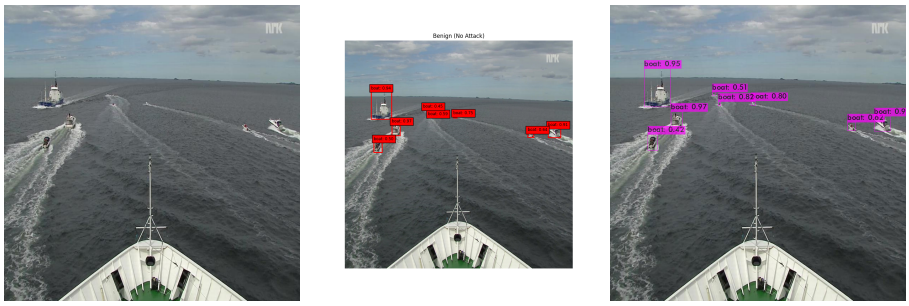


Figure 6.3: The black-box and white-box performance plotted against epsilon. The adversarial set is generated by the M4 model

Figures 6.4a, 6.5a, 6.6a, and 6.7a show how the adversarial noise is increasingly visible as epsilon is increased. Additionally, detections from the TensorFLow model is contrasted with detections from the Darknet model. In a lot of cases the predictions are quite similar, but these small differences can allow the Darknet model to make correct predictions when the TensorFLow model is fooled by the adversarial sample.

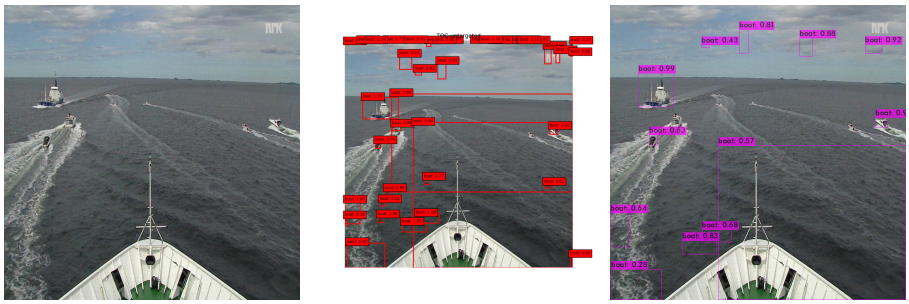


(a) Clean sample

(b) Clean sample with predictions from the TensorFlow model

(c) Clean sample with predictions from the Darknet model

Figure 6.4: The clean samples



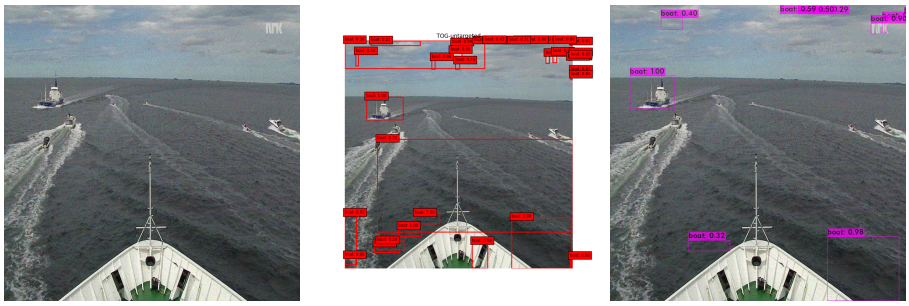
(a) Adversarial sample

(b) Adversarial sample with predictions from the TensorFlow model

(c) Adversarial sample with predictions from the Darknet model

Figure 6.5: The adversarial samples, epsilon=8

Figure 6.8 shows a direct comparison between a subtle perturbation with epsilon=8, and a much more obvious perturbation with epsilon=32. With epsilon=32 the sample no longer meets the definition of an adversarial sample by Szegedy *et al.* [3]. The original adversarial samples were a result of imperceptible non-random perturbations. However, this definition was developed when attacking classifiers, and object detectors generally require greater perturbations before the adversarial sample is effective. Additionally, there are now attacks which make no attempt at imperceptibility, like patch attacks.

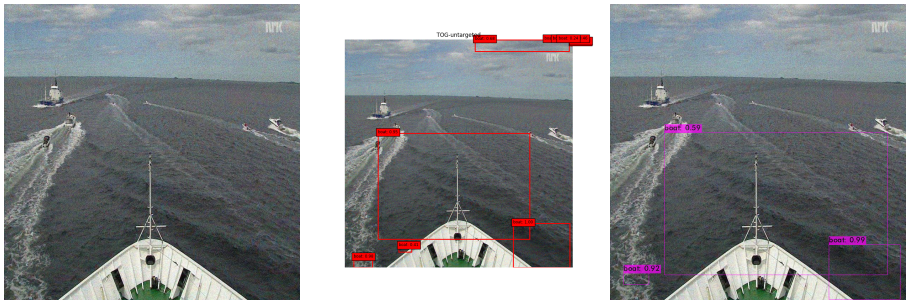


(a) Adversarial sample

(b) Adversarial sample with predictions from the TensorFlow model

(c) Adversarial sample with predictions from the Darknet model

Figure 6.6: The adversarial samples, epsilon=20



(a) Adversarial sample

(b) Adversarial sample with predictions from the TensorFlow model

(c) Adversarial sample with predictions from the Darknet model

Figure 6.7: The adversarial samples, epsilon=32

Figure 6.8: Distortion comparison of epsilon=8 and epsilon=32

6.2 RQ2: How Will Datasets Sharing Images Affect the Transference of the Attack?

6.2.1 Dataset Intersection

Table 6.6 shows the intersection between the datasets. Each row shows what percentage the intersection with the model on the column is of its total size.

$$\begin{matrix} D2 \subseteq D5 \\ \frac{D2 \cap D5}{D5} = 52\% \end{matrix} \tag{6.1}$$

Equation 6.1 follows from Table 6.6. The key finding in Figure 6.10 can be seen when comparing the slope of M4 line, to the slope observed in Figure 6.2. In Figure 6.10 the M4 slope is steeper due to the intersection between D4 and D5. This intersection allows M5 to attack M4 more effectively than M2. Using Table 6.6 we can explain the performance difference between attacks on M5 generated by M4, shown in Figure 6.3 and vice versa, shown in Figure 6.10, by the fact that while M4 is a subset of M5, images in M4 only make up about a third of M5. Figure 6.10 also shows that the white-box attack on M5 is by far the most effective white-box attack, compared to M2, and M4. This suggest that the M5 Darknet model is more similar to its TensorFLow counterpart, than M2, or M4.

	D1	D2	D3	D4	D5	D6
D1	100	16.31	16.31	65.58	81.89	16.31
D2	8.87	100	46	0	100	63.23
D3	12.1	62.75	100	0	100	12.59
D4	55.95	0	0	100	100	0
D5	23.32	52.33	38.37	33.38	100	33.09
D6	8.81	62.8	9.17	0	62.8	100

Table 6.6: The intersection between the various datasets

6.2.2 Attack Performance by Intersecting Models

This section presents attacks generated by models with dataset intersections. Models without intersections are also featured for context. Attacks on models with no intersections are highlighted in green to make comparison easier.

Since M5 is trained on a superset of the other datasets, it shows the best transference of the attacks. See Table 6.9

M1 is the only two-class model, and Table 6.7 shows poor attack transfer performance. The greatest mAP reduction besides the white-box attacks, is on M4 and M5. It roughly achieves a 10 point difference. However, the white-box attack is not particularly effective on M1 either, making the lack of transference expected.

Target Model	Clean	Untargeted	Vanishing
M1	45.75	29.16	37.67
M2	26.32	25.15	24.59
M3	10.36	9.3	8.73
M4	68.29	47.15	61.44
M5	74.56	65.99	64.77
M6	24.74	23.25	24.71

Table 6.7: Performance on adversarial samples generated by M1

Target Model	Clean	Untargeted	Vanishing
M1	45.75	40.7	41.66
M2	26.32	23.9	20.18
M3	10.36	2.13	0.94
M4	68.29	61.02	61.38
M5	74.56	64.14	59.34

Table 6.8: Performance on adversarial samples generated by M3

Target Model	Clean	Untargeted	Vanishing
M1	45.75	37.66	41.4
M2	26.32	7.91	9.71
M3	10.36	4.77	5.41
M4	68.29	52.66	60.65
M5	74.56	1.87	9.16
M6	24.74	7.95	12.35

Table 6.9: Performance on adversarial samples generated by M5

Target Model	Clean	Untargeted	Vanishing
M1	45.75	42.72	41.35
M2	26.32	15.12	10.99
M3	10.36	8.57	6.03
M4	68.29	57.8	59.8
M5	74.56	56.83	54.08
M6	24.74	1.24	0.09

Table 6.10: Performance on adversarial samples generated by M6

Plotting these values against attack performance gives the correlation between attack transference and dataset intersection, see Figure 6.9. It is apparent that a greater intersection leads to more transferable attacks. Training on the same images should make the two object detectors make similar predictions, which make them vulnerable to the same adversarial sample.

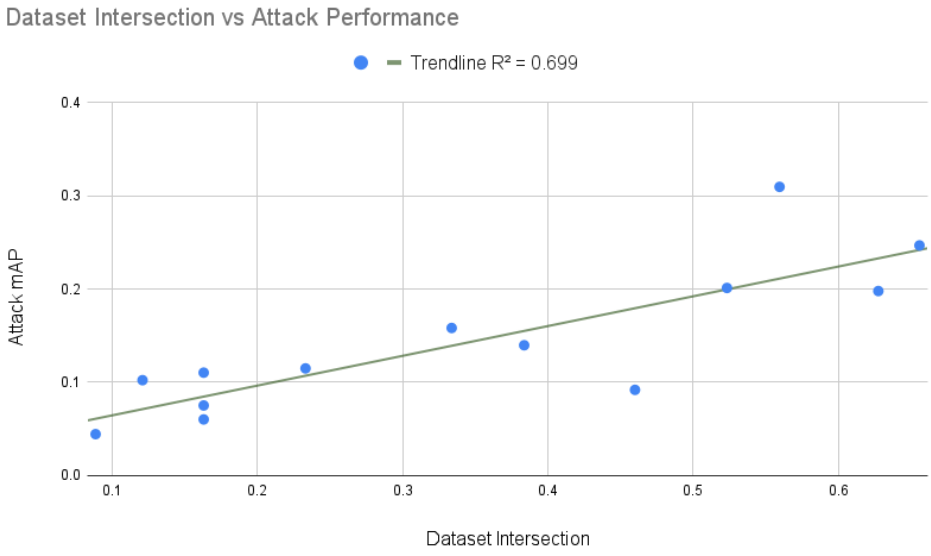


Figure 6.9: The correlation between dataset intersection and attack performance

6.2.3 Different Source and Target Resolution

The data in Table 6.11 show that a mismatch in resolution between source and target model incurs a penalty in attack strength. Table 6.12 compares the data from Table 6.11 and Table 6.9. Looking at Table 6.12 we can see that the average performance drop between same resolution and different resolution attacks, is 22.5 percentage points. This is in line with the findings from Chow *et al.* [1]. They found that downsampling an adversarial sample to a lower resolution always produced a weaker attack than upsampling the image. This implies that an attacker who does not know the resolution of the victim detector should chose a lower resolution like 416x416 to raise the probability of the attack being upsampled rather than downsampled.

Table 6.12 shows a comparison of the data in Table 6.9 with the data in Table 6.11. This shows the performance penalty incurred by downsampling the adversarial sample. The attack penalty for mismatched resolutions is smaller than the penalty for cross-dataset transference. This can be seen by comparing the black-box attacks in Table 6.4 and 6.5, with the white-box attack in Table 6.11. The black-box attacks achieve a mAP reduction of 6.67 and 1.92 respectively. The

Target Model	Clean	Untargeted	Vanishing
M1	37.96	36.47	37.29
M2	12.52	5.71	6.19
M3	4.47	2.02	2.07
M4	50.47	43.76	44.16
M5	51.25	40.06	37.53
M6	11.14	4.76	4.96

Table 6.11: Performance on adversarial samples generated by M5 at 416x416, but detected at 832x832 demonstrating the cross-resolution performance

white-box attack achieves a mAP reduction of 11.19. At small perturbations, a private dataset gives adversarial robustness, while mismatched resolutions only weakens the attack.

Target Model	Untargeted 832	Untargeted 416	Difference
M1	21.44%	3.93%	17.51
M2	72.23%	54.39%	17.84
M3	55.31%	54.81%	0.5
M4	25.85%	13.30%	12.55
M5	98.70%	21.83%	76.87
M6	67.87%	57.27%	10.6
Average			22.50

Table 6.12: Relative mAP loss for samples generated and detected at the same resolution compared to samples generated and detected at different resolutions.

6.2.4 Transfer Attack Performance Compared to Epsilon

Figure 6.10 shows that although epsilon=8 is a little too low to get a strong attack transference between some models, by epsilon=20 the model performance has dropped off sharply. This is a good indication, of how important the dataset intersection is to get stronger transference. Comparing Figure 6.3 to Figure 6.10 it is apparent that M5 generates adversarial samples that are more likely to fool M4, than M4 is to generate adversarial samples that fool M5.

The intersection between M5 and M4 allows for an epsilon reduction of almost 10, to achieve the same mAP reduction.

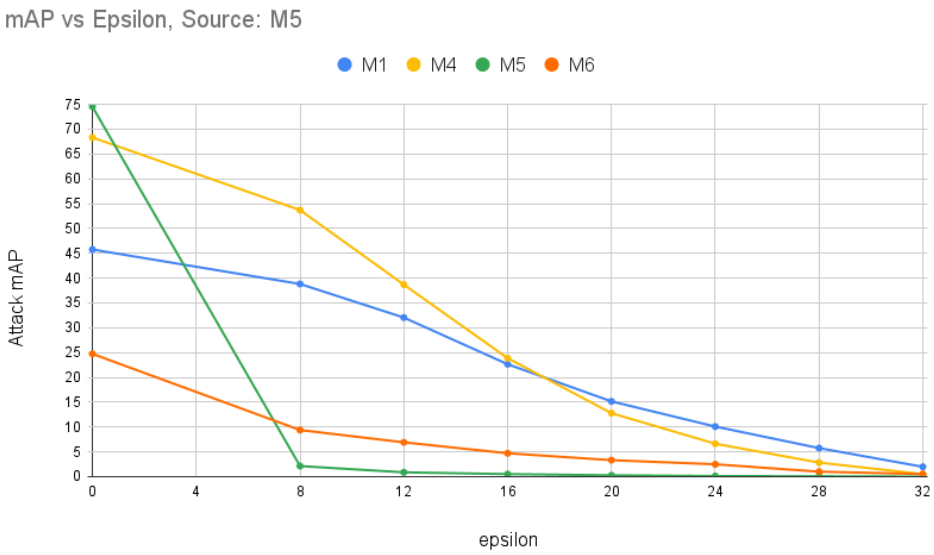


Figure 6.10: The black-box and white-box performance plotted against epsilon. The adversarial set is generated by the M5 model

Chapter 7

Discussion

7.1 RQ1

This section describes the contribution of the findings in section 6.1, as well as comparison to related work.

7.1.1 Comparison to Related Work

Though other studies have been done on transferable adversarial attacks, few target real-time object detectors. Of the studies done on transferable samples targeting object detectors [1][22][5], none operate with the same black-box model used in this thesis. They perform cross-resolution and cross-architecture transference, but not cross-dataset. Zhang *et al.* [5] do include a cross-dataset transference, but they use image independent universal perturbations. Their universal perturbations are able to reduce the mAP of their YOLO model by 30.9% at an L_∞ epsilon of 20. Our M2 model is able to reduce the mAP of M4 by 49.7% at the same L_∞ epsilon of 20.

The biggest issue encountered when attacking YOLO-Darknet is slight difference in prediction between the Darknet and TensorFLow implementations. Due to these differences attacks generated using the TensorFLow models do not have the same performance on the Darknet models. In this way the attacks can be thought of as cross-model attacks in addition to being black-box. The ideal way to avoid this would be to directly attack the Darknet models with TOG. To achieve that, one would need TOG to call the Darknet model, and correctly interpret the output. Such a setup seems achievable, but is left as future work.

Though other studies have looked at transference between YOLO detectors using different backbone networks, looking at transference between two YOLO models implemented in different frameworks was not found in other research. To attack the Darknet model with the TOG attacks, code had to written to correct for the resolution, the classes the model could detect, as well as the resizing method used by TOG.

7.1.2 Implications to Academia

Expanded knowledge about cross-dataset attacks, a rarely studied subject. Most papers study cross-architecture, or cross-resolution. Furthermore, modified TOG to allow adversarial attacks on Darknet models. The transfer attacks between datasets initially show quite low transference. However, by increasing epsilon from 8 to the mid 20s, the transference attack is able to reduce mAP by half. Though this is a much larger perturbation, it is in line with what other studies have used to achieve cross-architecture transference on object detectors [1][22][5].

7.1.3 Implications to Industry

Transfer attacks with subtle perturbations may not be an immediate threat. Though at higher epsilon values the transfer attacks become effective. These larger perturbations could be detected and pre-filtered, to avoid the detector being exposed to them [47].

7.2 RQ2

This section will discuss the findings in section 6.2, their contribution, and any new insights reached through the experiments. The results will also be compared to related work.

7.2.1 Comparison to Related Work

As cross-dataset transference on real-time object detectors is largely unexplored no other studies were found that examined the correlation between attack transference and relative dataset intersection. Paudice *et al.* [48] find that they can discover, and filter adversarial dataset poisoning attacks. However, if the samples added to the dataset have no perturbation added to them, but only serve to create an intersection between the target detector, and the attacker's surrogate detector, such techniques would be useless.

7.2.2 Implications to Academia

A correlation between relative dataset intersection and attack transference is found. To explore this further, two models were trained. One model was trained on a superset of the other model's training data. We found that the model trained on the superset, generated adversarial samples with better transference to the other model, than vice versa. This indicates that it is not simply the size of the intersection that matters, but the size of the intersection relative to the total training set. The average penalty to mAP reduction with attacking and target models at different resolutions, was found to be 22.5. To arrive at this number, adversarial samples were generated at 416x416, then detected at 832x832. Adversarial samples were generated at a lower resolution than the target resolution due to findings in Chow

et al. [1]. They found that downsampling adversarial samples always produces a weaker attack than upsampling it. This implies that an attacker who does not know the resolution of the target model, should train their surrogate model at a fairly low resolution like 416x416, to make it more likely that the sample is upsample rather than downsampled. We also found that the greater attack transference gained through intersecting dataset, allowed for effective transfer attacks at lower epsilons. Instead of epsilons of 24-28 with disjoint dataset, 16 was enough to reduce mAP to less than half, when datasets intersected.

7.2.3 Implications to Industry

The training set for an object detector deployed in an autonomous vehicle must be secured. Inclusion of public data must be carefully considered, as the public data increases attack transference, allowing effective attacks with a smaller epsilon. Comparing figure 6.2 to figure 6.10 shows that the M5 model, which was trained on a dataset with a significant intersection with M4, produces perturbations which are much more transferable than M2.

7.3 Threats to Validity

Internal validity is susceptible to selection and experimenter bias. Selecting an appropriate attack framework was long process with several blind allays. To make the selection process transparent, section 3.6 presents potentially relevant attacks, and their suitability.

To avoid experimenter bias, the data presented in section 6.1.2, and section 6.2.2 is the data output from the Darknet mAP tool. When derived data is presented, the raw data is also available together with the calculations done to arrive at the derived data. Section 5.2.2 describes how adversarial samples were generated, and section 5.2.3 describes how the performance was measured on clean and adversarial samples.

External validity is susceptible to poor generalizability and replicability of the results. This study is conducted with a single object detector, and attack framework, which does limit the generalizability of the results. However, the fact that adversarial samples transfer between dataset, architectures, even tasks, show that results on one set of detectors and attacks, are relevant to other detectors and attacks. Additionally, the scenario described in section 5.1 severally constrained the object detectors and adversarial attacks which were relevant to this study. Since the mAP score an object detection model achieves is highly dataset dependant, several datasets were compiled, and trained on.

Aiding replication is that all the code used to train models, convert the weights to TensorFLow weights, and measure mAP is public. Additionally, the pipeline used to generate adversarial samples targeting Darknet models with the TOG framework, is also public. Chapter 5 contains all the steps needed to replicate the

experimental setup. However, one of the primary datasets used in this thesis consists entirely of private data. Furthermore, the hardware required to train models, and generate adversarial samples is extremely expensive, and so only available to large companies, or universities.

Chapter 8

Conclusion

As all neural networks are vulnerable to adversarial examples, their adversarial robustness must be well understood, if they are to be deployed in safety critical contexts, like diagnosis, or autonomous vehicles. Though studies have been done on the risk of cross-architecture and cross-resolution attacks on real-time object detectors. Little work has been done on cross-dataset attacks.

This thesis studied cross-dataset transference by training Darknet models on disjoint training sets. Then used the TOG framework to generate adversarial sets. Once the adversarial sets were generated, attack performance was measured by calculating the mAP achieved on the adversarial set, and comparing this value with the mAP achieved on the clean set. We compared the attack performance on the surrogate model with the target model, and measured how well the attack transfers. At an epsilon of 8, the attack barely transferred, leaving the target model unaffected. Raising the epsilon to the mid 20s makes the transference stronger and reduces mAP on the target model by about half.

Additional models were trained on datasets with various intersection sizes. Adversarial sets were generated, and the models were benchmarked on them to find their mAP. Plotting the mAP reduction against the dataset intersection found the correlation between attack transference and dataset intersection.

By comparing attack strength on the models with intersecting datasets to the disjoint models, we found that epsilon could be lowered to 16 while retaining an effective attack when there is a dataset intersection.

We found the average penalty to attack strength associated with cross-resolution attacks. Combining this result with results from Chow *et al.* [1] implies that an attacker should always choose to generate adversarial samples at a low resolution to improve attack transference.

8.1 Future Work

The natural next step for this research is to add additional architectures to the analysis. Comparing the results on YOLO with SSD and Faster R-CNN could lead to further insights.

Developing an interface in the TOG framework which enables direct attacks on Darknet models would also be of interest. Eliminating the cross-model transference between to Darknet implementation and the TensorFLow implementation could produce clearer results.

Testing cross-dataset transference on adversarial patches would also be of interest. Adversarial patches avoid the issue of minimizing the perturbation, which could suggest that they offer strong, cross-dataset transference. Studying physical patches would make the attack even more relevant to autonomous vehicles.

Bibliography

[1] K.-H. Chow, L. Liu, M. E. Gursoy, S. Truex, W. Wei and Y. Wu, 'Understanding object detection through an adversarial lens,' in *European Symposium on Research in Computer Security*, Springer, 2020, pp. 460–481.

[2] Y. LeCun, Y. Bengio and G. Hinton, 'Deep learning,' *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus, *Intriguing properties of neural networks*, 2014. arXiv: 1312.6199 [cs.CV].

[4] N. Carlini and D. Wagner, *Towards evaluating the robustness of neural networks*, 2017. arXiv: 1608.04644 [cs.CR].

[5] Q. Zhang, Y. Zhao, Y. Wang, T. Baker, J. Zhang and J. Hu, 'Towards cross-task universal perturbation against black-box object detectors in autonomous driving,' English, *Computer Networks*, vol. 180, p. 1, Aug. 2020, Copyright - Copyright Elsevier Sequoia S.A. Oct 24, 2020; Last updated - 2021-01-11. [Online]. Available: https://search.proquest.com/scholarly-journals/towards-cross-task-universal-perturbation-against/docview/2476554302/se-2?accountid=12870.

[6] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard and L. D. Jackel, 'Handwritten digit recognition with a back-propagation network,' in *Advances in neural information processing systems*, 1990, pp. 396–404.

[7] A. M. Staff, *Adversarial robustness and real-time object detectors*, 2020.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, 'Ssd: Single shot multibox detector,' *Lecture Notes in Computer Science*, pp. 21–37, 2016, ISSN: 1611-3349. DOI: 10.1007/978-3-319-46448-0_2. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[9] J. Redmon and A. Farhadi, 'Yolov3: An incremental improvement,' *arXiv*, 2018.

[10] J. Redmon and A. Farhadi, *Yolo9000: Better, faster, stronger*, 2016. arXiv: 1612.08242 [cs.CV].

[11] Y. Zhao, H. Zhu, R. Liang, Q. Shen, S. Zhang and K. Chen, *Seeing isn't believing: Practical adversarial attack against object detectors*, 2019. arXiv: 1812.10217 [cs.CV].

[12] I. J. Goodfellow, J. Shlens and C. Szegedy, *Explaining and harnessing adversarial examples*, 2015. arXiv: 1412.6572 [stat.ML].

[13] L. Deng, 'The mnist database of handwritten digit images for machine learning research [best of the web],' *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[14] S.-M. Moosavi-Dezfooli, A. Fawzi and P. Frossard, *Deepfool: A simple and accurate method to fool deep neural networks*, 2016. arXiv: 1511.04599 [cs.LG].

[15] N. Papernot, P. McDaniel, X. Wu, S. Jha and A. Swami, *Distillation as a defense to adversarial perturbations against deep neural networks*, 2016. arXiv: 1511.04508 [cs.CR].

[16] K.-H. Chow, L. Liu, M. Loper, J. Bae, M. Emre Gursoy, S. Truex, W. Wei and Y. Wu, 'Adversarial objectness gradient attacks in real-time object detection systems,' in *IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications*, IEEE, 2020, pp. 263–272.

[17] X. Wei, S. Liang, N. Chen and X. Cao, 'Transferable adversarial attacks for image and video object detection,' in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization, Jul. 2019, pp. 954–960. DOI: 10.24963/ijcai.2019/134. [Online]. Available: https://doi.org/10.24963/ijcai.2019/134.

[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman, 'The pascal visual object classes (voc) challenge,' *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, 'Microsoft coco: Common objects in context,' in *European conference on computer vision*, Springer, 2014, pp. 740–755.

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

[21] H. Zhang, W. Zhou and H. Li, 'Contextual adversarial attacks for object detection,' eng, in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2020, pp. 1–6, ISBN: 1728113318.

[22] S. Chen, F. He, X. Huang and K. Zhang, *Relevance attack on detectors*, 2021. arXiv: 2008.06822 [cs.CV].

[23] O. Ronneberger, P. Fischer and T. Brox, 'U-net: Convolutional networks for biomedical image segmentation,' in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells and A. F. Frangi, Eds., Cham: Springer International Publishing, 2015, pp. 234–241, ISBN: 978-3-319-24574-4.

[24] K. He, X. Zhang, S. Ren and J. Sun, 'Deep residual learning for image recognition,' in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[25] Y. Jia, Y. Lu, J. Shen, Q. A. Chen, Z. Zhong and T. Wei, *Fooling detection alone is not enough: First adversarial attack against multiple object tracking*, 2019. arXiv: 1905.11026 [cs.CV].

[26] S. Ren, K. He, R. Girshick and J. Sun, 'Faster r-cnn: Towards real-time object detection with region proposal networks,' in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[27] O. M. Nezami, A. Chaturvedi, M. Dras and U. Garain, *Pick-object-attack: Type-specific adversarial attack for object detection*, 2020. arXiv: 2006.03184 [cs.CV].

[28] Y. Park and M. Kang, *Membership inference attacks against object detection models*, 2020. arXiv: 2001.04011 [cs.CV].

[29] J. Bao, *Sparse adversarial attack to object detection*, 2020. arXiv: 2012.13692 [cs.CV].

[30] X. Kuang, X. Gao, L. Wang, G. Zhao, L. Ke and Q. Zhang, 'A discrete cosine transform-based query efficient attack on black-box object detectors,' eng, *Information sciences*, vol. 546, pp. 596–607, 2021, ISSN: 0020-0255.

[31] Q. Liao, X. Wang, B. Kong, S. Lyu, Y. Yin, Q. Song and X. Wu, *Category-wise attack: Transferable adversarial examples for anchor free object detection*, 2020. arXiv: 2003.04367 [cs.CV].

[32] M. Lee and Z. Kolter, *On physical adversarial patches for object detection*, 2019. arXiv: 1906.11897 [cs.CV].

[33] Y. Wang, K. Wang, Z. Zhu and F.-Y. Wang, 'Adversarial attacks on faster r-cnn object detector,' *Neurocomputing*, vol. 382, pp. 87–95, 2020, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.11.051. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219316534.

[34] A. Madry, A. Makelov, L. Schmidt, D. Tsipras and A. Vladu, *Towards deep learning models resistant to adversarial attacks*, 2019. arXiv: 1706.06083 [stat.ML].

[35] Y. Wang, H. Lv, X. Kuang, G. Zhao, Y.-a. Tan, Q. Zhang and J. Hu, 'Towards a physical-world adversarial patch for blinding object detection models,' *Information Sciences*, 2020, ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2020.08.087. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025520308586.

[36] Y. Zhang, F. Wang and W. Ruan, *Fooling object detectors: Adversarial attacks by half-neighbor masks*, 2021. arXiv: 2101.00989 [cs.CV].

[37] T. Sato, J. Shen, N. Wang, Y. J. Jia, X. Lin and Q. A. Chen, *Security of deep learning based lane keeping system under physical-world adversarial attack*, 2020. arXiv: 2003.01782 [cs.CR].

[38] K.-H. Chow, L. Liu, M. E. Gursoy, S. Truex, W. Wei and Y. Wu, *Tog: Targeted adversarial objectness gradient attacks on real-time object detection systems*, 2020. arXiv: 2004.04320 [cs.LG].

[39] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie and A. Yuille, 'Adversarial examples for semantic segmentation and object detection,' in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1369–1378.

[40] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, *Yolov4: Optimal speed and accuracy of object detection*, 2020. arXiv: 2004.10934 [cs.CV].

[41] T. Bontzorlos, *Singapore maritime dataset frames ground truth generation and statistics*, 2019. [Online]. Available: https://github.com/tilemmpon/Singapore-Maritime-Dataset-Frames-Ground-Truth-Generation-and-Statistics.

[42] H. G. Norbye, *Real-time sensor fusion for the revolt model-scale vessel*, 2019.

[43] S. V. Grini, *Systematic training and testing of deep learning-based detection methods for vessels in camera images*, 2019.

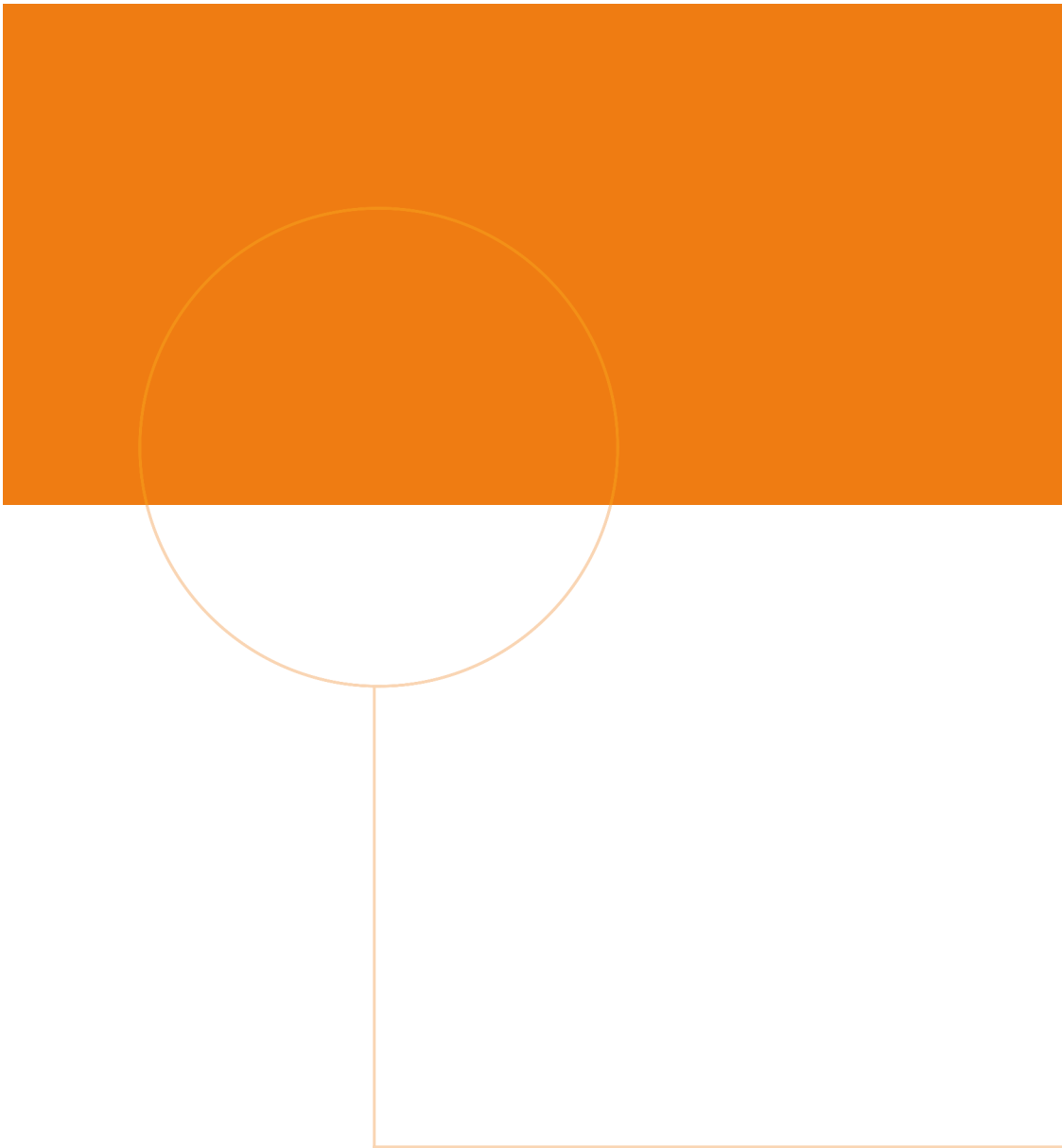
[44] D. K. Prasad, D. Rajan, L. Rachmawati, E. Rajabaly and C. Quek, *Video processing from electro-optical sensors for object detection and tracking in maritime environment: A survey*, 2016. arXiv: 1611.05842 [cs.CV].

[45] Tzutalin, *Labelimg*, 2015. [Online]. Available: https://github.com/tzutalin/labelImg.

[46] M. Själander, M. Jahre, G. Tufte and N. Reissmann, *EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure*, 2019. arXiv: 1912.05848 [cs.DC].

[47] R. Feinman, R. R. Curtin, S. Shintre and A. B. Gardner, *Detecting adversarial samples from artifacts*, 2017. arXiv: 1703.00410 [stat.ML].

[48] A. Paudice, L. Muñoz-González, A. Gyorgy and E. C. Lupu, *Detection of adversarial training examples in poisoning attacks through anomaly detection*, 2018. arXiv: 1802.03041 [stat.ML].



NTNU
Norwegian University of
Science and Technology

DNV