



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

COMPUTATION OF ADVERSARIAL MANIPULATIONS UNDER PHYSICAL ACCESS

by

Austin J. Van Dellen

June 2023

Thesis Advisor:
Second Reader:

Robert L. Bassett
Johannes O. Royset

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE June 2023		3. REPORT TYPE AND DATES COVERED Master's thesis
4. TITLE AND SUBTITLE COMPUTATION OF ADVERSARIAL MANIPULATIONS UNDER PHYSICAL ACCESS				5. FUNDING NUMBERS
6. AUTHOR(S) Austin J. Van Dellen				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000				8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A				10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.				12b. DISTRIBUTION CODE A
13. ABSTRACT (maximum 200 words) Neural networks are notoriously susceptible to adversarial perturbations, where small changes to an input cause drastic changes to the network's predictions. Previous work on adversarial perturbations assumes that an adversary can directly manipulate inputs to the neural network. We shift this assumption to one where the adversary can modify inputs only indirectly by manipulating an ambient physical environment. This assumption on an adversary's access to network inputs more realistically models the threat adversarial perturbations pose in many DOD applications. For concreteness, we focus on an application to autonomous sensing for passive sonar. In this setting, the ambient environment is measured with a physical sensor and then passed to a neural network for analysis. We investigate three state-of-the-art neural networks employed in this manner, each of which are able to correctly identify adversarial presence in at least 95% of the cases examined. Assuming the role of the adversary, we find perturbations to the physical environment that drop success rates to less than 50% for the same three networks. In so doing, we demonstrate that the lack of robustness in neural networks can be exploited to compromise physical sensors.				
14. SUBJECT TERMS adversarial perturbations, neural networks, autonomous classifiers				15. NUMBER OF PAGES 65
				16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified		18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified		19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified
				20. LIMITATION OF ABSTRACT UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**COMPUTATION OF ADVERSARIAL MANIPULATIONS
UNDER PHYSICAL ACCESS**

Austin J. Van Dellen
Captain, United States Marine Corps
BS, United States Naval Academy, 2017

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

**NAVAL POSTGRADUATE SCHOOL
June 2023**

Approved by: Robert L. Bassett
Advisor

Johannes O. Royset
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Neural networks are notoriously susceptible to adversarial perturbations, where small changes to an input cause drastic changes to the network's predictions. Previous work on adversarial perturbations assumes that an adversary can directly manipulate inputs to the neural network. We shift this assumption to one where the adversary can modify inputs only indirectly by manipulating an ambient physical environment. This assumption on an adversary's access to network inputs more realistically models the threat adversarial perturbations pose in many DOD applications. For concreteness, we focus on an application to autonomous sensing for passive sonar. In this setting, the ambient environment is measured with a physical sensor and then passed to a neural network for analysis. We investigate three state-of-the-art neural networks employed in this manner, each of which are able to correctly identify adversarial presence in at least 95% of the cases examined. Assuming the role of the adversary, we find perturbations to the physical environment that drop success rates to less than 50% for the same three networks. In so doing, we demonstrate that the lack of robustness in neural networks can be exploited to compromise physical sensors.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1	Introduction and Literature Review	1
1.1	General Problem Context	2
1.2	Narrower Research Area	2
1.3	Need for Further Research.	7
1.4	Summary of Approach	9
2	Methodology	13
2.1	Modeling the Acoustic Environment.	13
2.2	Problem Formulation.	15
2.3	Gradient Computation and Interferer Output Constraints	18
3	Experiment and Results	27
3.1	Experimental Framework	27
3.2	Defender’s Neural Network	29
3.3	Perturbation Generation.	35
3.4	Interferer Impact	37
4	Conclusions and Future Research	41
4.1	Conclusions	41
4.2	Future Research.	41
	List of References	43
	Initial Distribution List	47

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

Figure 1.1	Adversarial examples as shown in Szegedy et al. (2013)	5
Figure 1.2	Adversarial example via FGSM	6
Figure 1.3	3D perturbation under physical access assumption	9
Figure 1.4	Example of spectrogram produced from a signal	11
Figure 3.1	Visual depiction of state space	29
Figure 3.2	Frequency ranges of common oceanic activity	31
Figure 3.3	Log-linear regression of RMS as a function of frequency	32
Figure 3.4	Adding noise to spectrograms	33
Figure 3.5	Example of perturbed spectrogram at 310 Hz	38
Figure 3.6	Example of perturbed spectrogram at 390 Hz	39

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

Table 3.1	Experimental bounds and constant parameters	28
Table 3.2	Location and movement of adversarial agents	28
Table 3.3	Intruder and non-hostile frequencies	30
Table 3.4	Model accuracies	35
Table 3.5	Experimental results	37

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AI	Artificial Intelligence
CNN	Convolutional Neural Network
DOD	Department of Defense
FEM	Finite Element Method
FGSM	Fast Gradient Sign Method
L-BFGS	Limited-memory Broyden-Fletcher-Goldfarb-Shanno
PSD	Power Spectral Density
RGM	Reduced Gradient Method

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

In recent years, one of the most prominent developments within artificial intelligence has been the use of neural networks for image classification. While these neural networks demonstrate a remarkable capacity to identify images, it is well documented that they are also susceptible to adversarial perturbations, where small changes to an input cause drastic changes to the network's predictions. The preponderance of literature on this topic, however, assumes that an adversary has unhindered access to neural network inputs. From a defense perspective, a more plausible case is one in which the adversary can interfere with inputs only indirectly by manipulating a physical environment in which it operates. This research focuses on such a scenario, specifically examining the case of a passive sonar sensor, known as the Defender, monitoring an acoustic environment for adversary presence. An adversarial underwater vessel, referred to as the Intruder, moves through this environment while emitting an acoustic signal. A second adversarial underwater vessel, the Interferer, also moves through the environment, seeking to emit its own acoustic signal such that the Defender does *not* detect the presence of the Intruder.

We model this problem under the assumption that the Defender uses a neural network to map the received acoustic signal to a confidence of intruder presence. Specifically, the signal is converted to a spectrogram via a short-time Fourier transform and processed as an image. The neural network is trained using a variety of spectrograms, each labeled as either "Intruder" or "Non-Hostile," with the assumption that a particular set of frequencies is associated with adversarial signatures.

The Interferer seeks to emit an acoustic signal which, when processed in conjunction with all other noise in the environment, results in a spectrogram which the neural network classifies incorrectly. The primary challenges in producing this signal come from the physically constrained nature of the problem. To successfully perturb a spectrogram produced by the Defender, the Interferer needs to determine how its acoustic output impacts the Defender's confidence that the spectrogram contains an Intruder output. This requires the Interferer to model the acoustic environment in an efficient and scalable manner, emphasizing the need for a generalizable approach applicable in a wide range of settings as opposed to a narrowly focused case study. Furthermore, the Interferer may need to constrain the frequencies of

its emission to match the physical constraints of its speaker. Despite these constraints, we are able to produce signals which prevent the Defender's neural network from detecting the Intruder.

We find that the lack of robustness in neural networks *can* be exploited to compromise physical sensors. State of the art CNNs see high success rates when classifying Intruder presence *absent* Interferer outputs. Specifically, the Defender correctly classifies Intruder presence in 98%, 99%, and 95% of cases examined using three separate state-of-the-art neural networks. Assuming the role of the Interferer, we are able to find acoustic outputs which modify the environment in a way that the Defender is often unable to detect the presence of the Intruder. With Interferer output included, the Defender's success rate drops to 50%, 43%, and 0% for the same three neural networks. Frequently, the acoustic output of the Interferer is much smaller than the output of the Intruder, and the resulting spectrograms produced by the Defender often look nearly identical to those with no Interferer signal. Moreover, the successful perturbations were found using only a single optimization method with hand tuned parameters. While we note that the Interferer may be able to drive success rates lower through a more exhaustive use of optimization methods or more deliberate choice of tuning parameters, our work clearly demonstrates that the lack of robustness in neural networks can be exploited to compromise physical sensors.

CHAPTER 1:

Introduction and Literature Review

Using the Manhattan Project as a model, the United States must undertake and win the artificial intelligence race by leading in the invention and deployment of AI.

— 2020 House Armed Services Committee

Artificial intelligence (AI) has revolutionized many fields by automating complex tasks previously thought to require human intelligence. Machine learning models lie at the heart of AI, and are designed to make predictions, identify patterns, or perform particular tasks based on data they are trained on. In recent years, certain machine learning models have shown a remarkable capacity for a wide variety of tasks such as image recognition, data analysis, and language processing. However, this performance is often accompanied by heightened complexity, which brings with it an inherent lack of robustness.

Researchers have shown that many machine learning models are susceptible to deception or manipulation through subtle alterations to input data. This thesis provides a survey of the both the methodology and numerical experiments found in Bassett et al. (2023), in which we manipulate model inputs in an under-investigated context. We present an environment in which an adversary is able to modify inputs only *indirectly* through manipulation of an acoustic environment. In this setting, we seek to determine whether the lack of robustness can be exploited to compromise physical sensors.

This chapter serves as an introduction to the problem as well as a review of relevant literature on the topic. We discuss the general context of our work, the narrower research area, and the need for further research to motivate our specific research question. We then outline our approach to answering this question before transitioning to Chapter 2.

1.1 General Problem Context

The Department of Defense (DOD) is in an artificial intelligence race. It is widely accepted across the DOD that AI, defined as “the ability of machines to perform tasks that normally require human intelligence” (Allen 2020), is changing the character of war (Berger 2019). Accordingly, the United States must be aware of developments within AI, determine how best to leverage them, and move quickly to deploy them. This position is well-substantiated in key planning documents from recent history. The 2018 National Defense Strategy stated that the “Department of Defense will invest broadly in the military application of autonomy, artificial intelligence, and machine learning” (Mattis 2018). The 2022 National Defense Strategy reiterated that the DOD intends to enact institutional reforms to quickly integrate AI developments into operations. (Austin III 2022). Individual service Departments reflect similar priorities. One example comes from the 38th Commandant’s Planning Guidance for the United States Marine Corps, published in 2019. It states that the Marine Corps “must leverage unmanned technologies and artificial intelligence to enhance our situational awareness, lethality, and expeditionary potential” (Berger 2019).

The DOD is well aware that AI is poised to make a broad variety of contributions if employed properly. However, as we race to leverage these emerging capabilities, we must be cognizant of the risks they pose.

1.2 Narrower Research Area

In recent years, one of the most prominent developments within AI has been the use of statistical models for image classification. These models associate images with categories based on the features of those images. The most successful version of these models is the neural network, which in some cases performs image classification at a higher accuracy rate than humans (He et al. 2015).

A neural network is a type of statistical model initially inspired by the form and function of the human brain, composed of interconnected nodes, or neurons (Goodfellow et al. 2016). Each neuron applies a linear transformation to a series of inputs, applies an “activation function” to the result of this transformation, and passes this result to other neurons. Typically, the activation function is non-linear, which allows for more complex relationships between inputs and outputs than linear models. Generally, neural networks consist of any number of

layers of neurons, the first of which is typically referred to as the “input layer,” the last of which is referred to as the “output layer.” In the case of a grayscale image, it is common for neural networks to assign one neuron per pixel to capture its color intensity numerically. For colored images, neural networks typically have three neurons per pixel to capture each of the colors red, green, and blue. The output layer has one or more neurons expressing the likelihood that a given input belongs to each of the categories comprising the data set. All layers in between the input and output layers are referred to as hidden layers, and networks with more than one hidden layer are referred to as “deep” networks.

A convolutional neural network (CNN) is a particular type of neural network designed to take images and other grid-like data as inputs. CNNs contain one or more layers which employ a linear operation called a convolution in place of general matrix multiplication, perform a “pooling” operation which retains information provided by the convolution while reducing the spatial dimensions, and apply an activation function before passing the output to the next layer (Goodfellow et al. 2016). See chapter 9 of Goodfellow et al. (2016) for a more complete introduction to CNNs.

At least as early as 2015, state-of-the-art CNNs have surpassed human performance in particular image classification contexts (He et al. 2015). The potential of superhuman performance at near instantaneous speeds has obvious implications for a plethora of DOD activities, most notably intelligence, surveillance, and reconnaissance operations. Proper understanding of the capabilities and limitations of these systems, however, is crucial to proper employment.

Despite resounding success in many image classification applications, deep neural networks suffer from a lack of robustness to adversarial perturbations. An adversarial perturbation is a modification to an input of a neural network which results in incorrect classification by that network. An adversarial example is the modified input that results from the perturbation. The intended effects of these examples vary. Papernot et al. (2015) and Barreno et al. (2010) provide a taxonomy of the different types of attacks. In some instances, an adversary may seek to introduce broad ambiguity by reducing a network’s confidence in its output classifications. In other contexts, an adversary might construct a hyper-specific perturbation such that the example receives one of thousands of potential classifications with high confidence.

In the case of image classification, Szegedy et al. (2013) develop a methodology for computing adversarial perturbations using a box-constrained Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm. BFGS is a popular quasi-Newton algorithm used to solve optimization problems (see, e.g., Section 1.I of Royset and Wets [2022]). L-BFGS employs BFGS, but uses less computer memory than the original method as it restarts the algorithm after a specified number of iterations.

Carlini and Wagner (2017) concisely describe the problem solved by Szegedy et al. in Problem (1.1). In this formulation, F denotes a fixed neural network, x is an image in $\mathbb{R}^n$ and x' is the adversarial example generated by applying a perturbation to x . Taking $F(x')$ generates a probability vector indicating the network's level of confidence regarding the classification of image x' across the different categories within the dataset. The problem is then expressed as

$$\begin{aligned} \min_{x'} \quad & \|x - x'\|_2^2 \\ \text{s.t.} \quad & \operatorname{argmax} F(x') = y \\ & x' \in [0, 1]^n. \end{aligned} \tag{1.1}$$

Here, the objective function chooses a perturbed image x' such that the dissimilarity between x and x' as measured by L_2 distance is minimized. The argmax function selects the index of the highest value from the probability vector generated by $F(x')$, and the label y corresponding to that index is given to the perturbed image x' . In this context, the variable y is a different label than the one assigned to the unperturbed image x .

The nonlinear constraint in Problem (1.1) makes it a difficult problem to solve, so Szegedy et al. solve the problem as

$$\begin{aligned} \min_{x'} \quad & c \|x - x'\|_2^2 + \operatorname{loss}_{F,y}(x') \\ \text{s.t.} \quad & x' \in [0, 1]^n \end{aligned} \tag{1.2}$$

instead. In this formulation of the problem, the objective function consists of two components. The first component, $c \|x - x'\|_2^2$, scales the Euclidean distance between x and x' by the constant c . The second component, $\operatorname{loss}_{F,y}(x')$, represents the loss incurred by assigning label y to image x' . We note that this is smallest when the label y is assigned to x' , and

further that this component is generalizable to any choice in loss function used to train F . Szegedy et al. (2013) then perform a line search to find c such that the Euclidean distance between x and x' is minimized and the label y is assigned to x' .

While the reader may see Carlini and Wagner (2016) or Szegedy et al. (2013) for more details, we note here that Szegedy et al. not only draw attention to the existence of adversarial perturbations, but put forth an efficient method to compute them. Figure 1.1 illustrates a handful of adversarial examples produced with this method that were generated for AlexNet, a neural network consisting of 500,000 neurons and 60 million parameters (Krizhevsky et al. 2012). At its introduction in 2012, AlexNet outperformed all previous state of the art models by a wide margin, and is still highly regarded in the field of image classification (Krizhevsky et al. 2012).

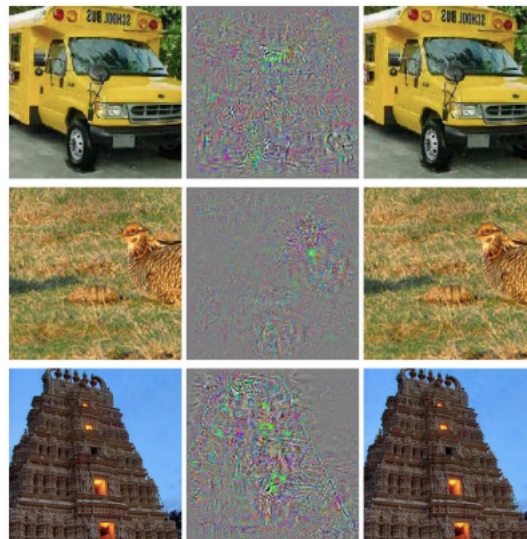


Figure 1.1. The left column contains images which were correctly classified by AlexNet (Krizhevsky et al. 2012). The middle column contains the scaled modification to the image on the left resulting in the image on the right. In each case, AlexNet labeled the image in the right column as a particular type of ostrich. In the context of (1.2), the image on the left is x , the image on the right is x' , and the modification in the middle represents $x - x'$. Source: Szegedy et al. (2013).

Following the work of Szegedy et al., many techniques have been developed for computing adversarial perturbations. Goodfellow et al. introduce the fast gradient sign method (FGSM),

a quick way to using a linear approximation to a loss function to construct perturbations (Goodfellow et al. 2014). Given neural network parameters θ , network input x , network label y , scalar ε , and loss function L used to train the neural network, FGSM computes the perturbation η as

$$\eta = \varepsilon \operatorname{sign}(\nabla_x L(\theta, x, y)). \quad (1.3)$$

By computing a linear approximation near the current values of θ , FGSM efficiently produces gradients that can be employed in a variety of optimization algorithms (Goodfellow et al. 2014). Though this method is not necessarily designed to produce an output similar to the input (Carlini and Wagner 2016), a small choice of ε can achieve this effect (see Figure 1.2).

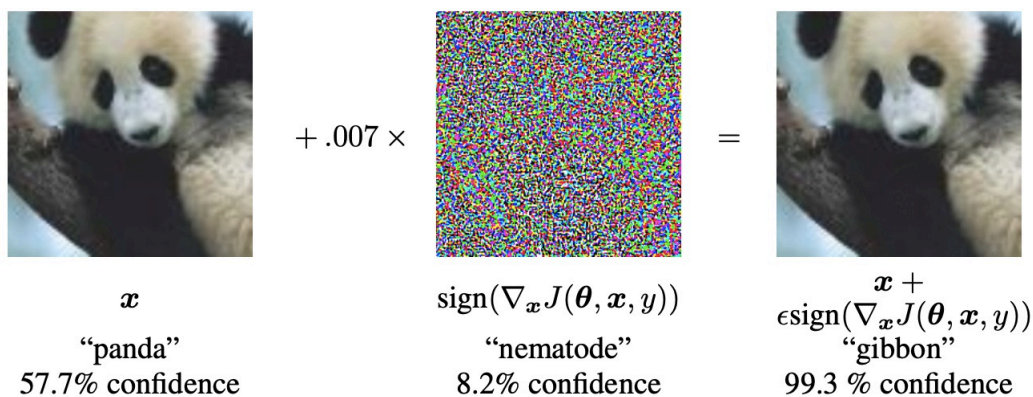


Figure 1.2. An adversarial example produced using FGSM. Note that the resulting image appears similar to the original due to the small choice in ϵ . Source: Goodfellow et al. (2014).

DeepFool, introduced by Moosavi-Dezfooli et al. provides an efficient means of producing adversarial examples which are closer to the original images by L_2 distance than the method put forth by Szegedy et al. (Carlini and Wagner 2016). DeepFool iteratively constructs hyperplanes estimating the decision boundaries of the neural network, then moves towards the nearest decision boundary (Moosavi-Dezfooli et al. 2016). For brevity, we refer the reader to the original publication for more detail (Moosavi-Dezfooli et al. 2016), but note that DeepFool provides an efficient way that improves upon the method put forth by Szegedy

et al. Carlini and Wagner (2016) introduce a method resulting in examples which are even closer to the original input image by optimizing the perturbation with respect to L_0 , L_2 , or L_∞ norms (Carlini and Wagner 2016). SuperDeepFool, a method designed in the context of both the L_2 norm minimization in the Carlini and Wagner approach and the geometric principles underlying the original DeepFool, outperforms both of these methods in both success rates and L_2 norm metrics (Abdollahpourroostam et al. 2023).

In addition to these generalizable methods for computing adversarial perturbations, effective methods for specialized contexts have been presented. For instances in which gradient information cannot be obtained, (Chen et al. 2017) construct perturbations using zeroth order optimization, which involves querying a model to indirectly estimate gradient information. GenAttack uses a gradient-free approach which proves more successful than zeroth order optimization with far fewer queries (Alzantot et al. 2018). Bassett and Graves (2020) show how to compute adversarial examples which are minimally observable to the human eye by accounting for the way it perceives texture and color. In the context of medical imaging analysis, Finlayson et al. (2018) and Dong et al. (2023) demonstrate ways in which adversarial technique can be employed in a high-stakes setting.

In short, there are proven techniques exist to construct adversarial perturbations in varying contexts and goals. Szegedy et al. provide some of the first successful methods to fool a neural network, and FGSM provides a quicker and highly intuitive way to do the same. DeepFool uses a somewhat more elaborate scheme which results in examples closer to the original input, while Carlini and Wagner reduce the difference between original and perturbed inputs further. SuperDeepFool improves upon both of these methods further still in terms of success rates and L_2 minimization. Zeroth order optimization, GenAttack, and color and edge-aware perturbations provide just a few examples of techniques in specialized domains. These and other approaches have proven to be remarkably effective. In these approaches, however, and indeed much of the literature on the topic, there is a critical assumption in place: An adversary has unhindered access to the inputs of the neural network.

1.3 Need for Further Research

The methods for constructing adversarial perturbations discussed thus far assume that an adversary seeking to manipulate an input can do so however they would like. In these

examples, the inputs are frequently modified (or left unperturbed) on a pixel-by-pixel basis. We refer to this unhindered ability as *logical access*, where an adversary has complete access to the data streamed and stored by the neural network. If an adversary were to have logical access to a neural network employed by the DOD, the compromise of the system on which the neural network resides would likely be of more concern than the misclassifications themselves. Accordingly, studying the vulnerabilities of neural networks under this assumption is pessimistic and in some ways irrelevant from a DOD perspective. We contrast the concept of logical access to that of *physical access*, in which an adversary can interfere with neural networks only *indirectly* by manipulating a physical environment in which the adversary and the neural network reside.

While the preponderance of the literature on adversarial perturbations employs a logical access assumption, some work has been done under that of physical access. Evtimov et al. (2017) show how to apply black and white stickers to street signs so that state of the art neural networks consistently classify them incorrectly from a variety of angles, distances, and environmental conditions. In a similar application, Brown et al. (2017) construct a physical patch which, when included in images of typical objects, cause misclassification in highly regarded image classification networks. Athalye et al. (2017) use 3D printers to create objects whose 2D images are consistently misclassified, such as the 3D printed turtle seen in Figure 1.3. Collectively, these examples serve as a proof of concept that neural networks used for conventional image recognition have a susceptibility to physically constrained adversarial perturbations.



■ Classified as “turtle”
 ■ Classified as “rifle”
 ■ Classified as “other”

Figure 1.3. An example of adversarial examples constructed under the assumption of physical access. Here, various 2D images of a 3D printed turtle are misclassified as a rifle by a neural network. Adapted from Athalye et al. (2017).

Using the literature referenced thus far as a starting point, we examine the vulnerability of neural networks employed with physical sensors in a more plausible DOD setting. In such a setting, a neural network may be used to classify digital signals collected from sensors within a physical environment. While this signal may be a traditional image captured by a camera, there is a wide range of signals suitable for processing by a neural network. In many of these cases, the pathway from adversarial perturbation to neural network input may be more complex than in the traditional image recognition case. For example, even in the 3D perturbations in Figure 1.3, mapping the choices in color of the physical object to pixel values in the resulting 2D image is relatively straightforward. However, when measuring physical properties such as pressure, sound, or magnetic currents, mapping a physical disturbance to the captured signal can be more challenging. In this defense-centric context, we seek to answer the following research question:

Can the lack of robustness in neural networks be exploited to compromise physical sensors?

1.4 Summary of Approach

To address our research question, we introduce the following problem setting:

Consider a passive sonar sensor, referred to as the Defender, monitoring an underwater acoustic environment for the presence of adversarial activity. An adversarial underwater vessel, referred to as the Intruder, moves through this environment while emitting an acoustic signal. A second adversarial underwater vessel also moves through the environment, seeking to emit its own acoustic signal such that the Defender does *not* detect the presence of the Intruder. We refer to this agent as the Interferer. A plausible example is a submarine employing an unmanned underwater vehicle to mask its presence, which could hold operational and even strategic level implications if done successfully. In turn, the Defender uses the signal received, or some transformation of it, as an input to a predictive model, which maps the input to $[0,1]$. This number is the Defender’s confidence that the environment contains the Intruder.

To make the problem concrete, we consider a setting in which the Defender transforms the received acoustic signal to a spectrogram, then passes it to a neural network for classification. A spectrogram is a visual representation of the power of a signal contained in various frequencies over time. In our problem context, the signal received by the Defender is transformed to a spectrogram through a Short-time Fourier Transform (STFT). For a given discretized input signal s , the STFT of s captures the intensity of frequency ω at time τ . Both τ and ω are binned based on the choice of window and hop length parameters, explained below. The STFT is defined as

$$\text{STFT}_s[\tau, \omega] = \sum_{k=0}^{K-1} w(k)s(\tau H + k)e^{-j\frac{2\pi}{K}\omega k}. \quad (1.4)$$

The input signal s is divided into overlapping frames of length K , also referred to as window length. The variable H represents the hop length, which dictates the shift between starting indices of adjacent windows. For example, given a hop length of 64, the first bin τ would begin at s_0 , the second bin at s_{64} , the third at s_{128} , and so forth. The Discrete Fourier Transform (DFT) $e^{-j\frac{2\pi}{K}\omega k}$ expresses the signal contained in a particular window in the frequency domain, where j denotes the imaginary unit (Leiber et al. 2022). Since the windowed signals are not necessarily periodic, a “window” or “tapering” function is used to taper the edges of that window to zero. We employ the commonly used Hann window

function, defined by

$$w(k) = \frac{1}{2} - \frac{1}{2} \cos\left(\frac{2\pi k}{K-1}\right). \quad (1.5)$$

For each element (τ, ω) , the STFT of s contains both a real and an imaginary part. The spectrogram itself is defined as the squared modulus of the STFT (applied elementwise), and represents the power distribution of the signal (Płotka et al. 2020). Figure 1.4 gives an example of an input signal s at 250 Hz and the spectrogram obtained via its STFT ¹.

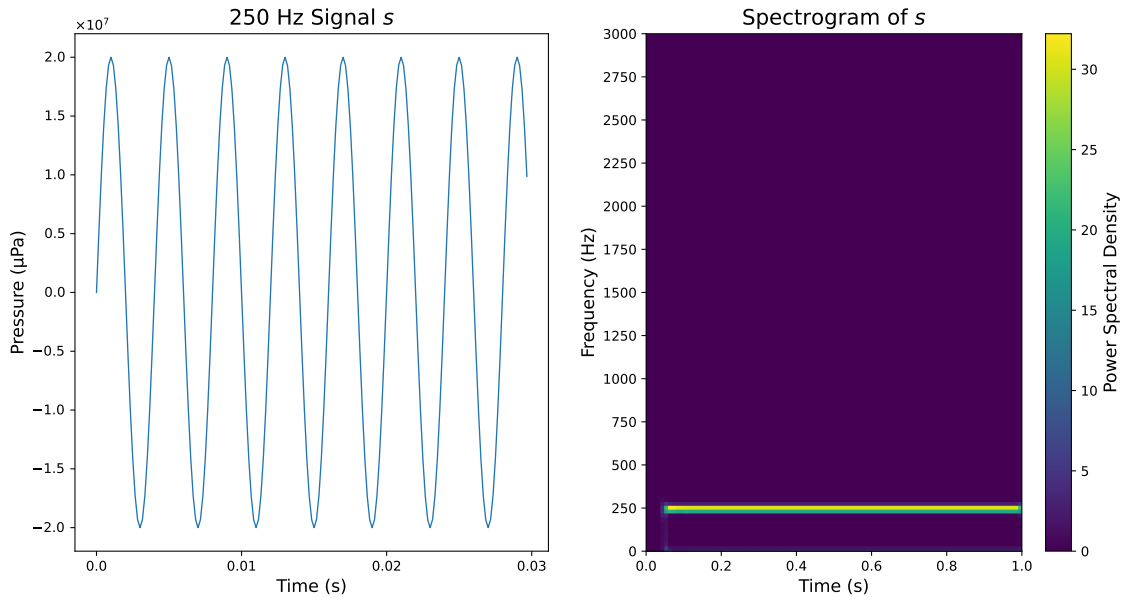


Figure 1.4. Left: Pressure resulting from a 250 Hz signal s . We examine a small subset (0.03 seconds) of the time sample captured in the figure on the right for better visualization. Right: Spectrogram of a pure 250 Hz signal. Note that this is obtained by squaring the modulus of each (τ, ω) of the STFT of s .

Spectrograms and related representations of signal frequency spectrums are used in practice in various audio processing techniques (Sharma et al. 2020). The coupling of spectrograms and neural networks is used in practical applications including speech recognition, music

¹Hertz (Hz) is defined as cycles per second. In our context, 250 Hz implies that water molecules in the environment vibrate back and forth 250 times per second. Phrased slightly differently, a 250 Hz signal would result in 250 waves passing a stationary point in one second.

detection, and even electrocardiogram classification (Leiber et al. 2022). Choo et al. use spectrograms and neural networks in tandem to improve on underwater targeting via active sonar employment, an application which has obvious DOD applications (Choo et al. 2022).

A Brief Note on Units

While our problem context is generalizable for several different choices in units, we clarify the units we use in our application. We measure Interferer and Intruder outputs f and g , the acoustic environment w , and the signal s received by the Defender in terms of pressure. In particular, we use micro Pascals, denoted μPa . We use the Pascal as it is the standard unit of measurement used in practice, but we note that the same approach we outline in Chapter 2 holds for any unit of pressure. The resultant spectrograms measure the power spectral density (PSD) of the signals. This is distinct from the measure of decibels relative to full scale (denoted dBFS) sometimes seen in spectrograms, which involves taking a logarithmic transform of the PSD we present.

CHAPTER 2: Methodology

In this chapter, we discuss how we model and solve the problem introduced in Chapter 1, Section 4. First, we discuss our approach to capturing the physical properties of our environment and clarify the behavior of the agents within it. This leads us to our problem formulation, presented in the second section. The third section highlights key components of our approach to solve this problem.

2.1 Modeling the Acoustic Environment

In this section, we outline the method we use to model the acoustic environment. We start with the homogenous wave equation, which captures physical properties of how sound propagates through water. We then introduce the acoustic output of the Interferer and Intruder via the inhomogeneous wave equation. Next, we summarize our method of discretizing the environment, and end by reinforcing key assumptions regarding the behavior of the agents within it.

2.1.1 Wave Equations

The homogenous wave equation models the behavior of waves in a given medium without the presence of any external forces. In two dimensions, the homogenous wave equation is given by (see, e.g., Section 6.2 of Jensen et al. [2011])

$$\frac{\partial^2 u}{\partial t^2} - c^2 \Delta u = 0, \quad x \in \mathbb{R}^2. \quad (2.1)$$

Here, u is a function of time t and spatial coordinates x_1 and x_2 , and represents the displacement of a wave given a constant wave speed c . The first term represents the acceleration of the wave through the medium. Δ is the Laplacian operator, meaning Δu is the sum of the second partial spatial derivatives and can be written as

$$\Delta u = \frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2}. \quad (2.2)$$

This term represents the curvature of the function u in space, capturing how wave displacement changes spatially. When multiplied with the square of the wave speed c and incorporated with the acceleration term, the homogenous wave equation captures the propagation of waves in a medium. We introduce the acoustic output of the Intruder in the inhomogeneous wave equation

$$\frac{\partial^2 u}{\partial t^2} - c^2 \Delta u = g(t) \delta(x - x(t)_{\text{intruder}}). \quad (2.3)$$

Here, $g(t)$ captures the acoustic output of the Intruder at each time step $t \in T$, δ is the Dirac delta function, and $x(t)_{\text{intruder}}$ is the location of the Intruder at time t . In concert, the terms on the right-hand side of the equation model the localized emission of the Intruder, while the left-hand side captures the propagation of the resulting waves. Finally, we introduce $f(t) \delta(x - x(t)_{\text{interferer}})$ to the right-hand side, which captures the acoustic output and location of the Interferer in the same manner $g(t)$ and $x(t)_{\text{intruder}}$ do for the Intruder. This gives us the inhomogeneous wave equation

$$\frac{\partial^2 u}{\partial t^2} - c^2 \Delta u = f(t) \delta(x - x(t)_{\text{interferer}}) + g(t) \delta(x - x(t)_{\text{intruder}}). \quad (2.4)$$

We discretize the inhomogeneous wave equation via the finite element method (FEM). Broadly speaking, FEM is a means of approximating a solution to a partial differential equation by a) discretizing the domain into finite elements, then b) using a set of functions to approximate the solution at points within each element (see, e.g., Section 7.4 of Jensen et al. [2011]). We use Lagrangian basis functions of degree 1, meaning the value at points within each element of our state space are approximated using piecewise linear functions. We adopt the same discretization used by Hyde (2023), to which we refer the reader for details. The resulting approximation of Equation (2.4) is given by the linear system

$$Aw = Bf + Cg. \quad (2.5)$$

This approximation represents the relationship between the state space w , the Interferer acoustic output f , and the Intruder acoustic output g . We introduce Ψ as the set of discretized points in space, and T as the set of discretized points in time. The state space w is a vector of

length $|\Psi||T|$ capturing the acoustic pressure field at each point in space and time. Interferer and Intruder outputs f and g are vectors of length $|T|$, each containing the acoustic output of their respective sources. The matrix A has size $(|\Psi||T|) \times (|\Psi||T|)$, and represents the linear functions used in FEM to approximate the solution to the continuous function u for all points in space Ψ and time T . The matrices B and C are of size $(|\Psi||T|) \times |T|$, and describe how the acoustic outputs of the Interferer and Intruder influence the state space w .

In concert, the matrices A , B , and C capture the dynamics of the inhomogeneous wave equation to approximate w given Interferer and Intruder outputs f and g . We note that the construction of matrix A incorporates an absorbing boundary condition (ABC). An ABC is an artificially imposed condition used to minimize the reflection of waves at the boundary of a simulation (see, e.g., Chapter 6 of Jensen et al. [2011]). In our case, we use an ABC to avoid incorporating reflections off the boundary of our simulation. We assume the acoustic waves we model in our experiments would propagate beyond the experimental boundaries rather than reflect off of them and back into the state space. Details regarding absorbing boundary conditions in acoustics in general can be found in chapter 6 of Jensen et al. (2011), and the method of encoding an ABC into A is discussed in detail in Hyde (2023).

2.1.2 Agent Behavior

We allude to the physical behavior of the agents in this setting in Section 1.5, and reinforce these key physical assumptions here. First, the Defender is static and passive throughout the simulation. This mirrors the problem context as introduced, and examination of Equation (2.5) confirms there is no term to explicitly incorporate the acoustic output of the Defender. Second, the Interferer and Intruder are moving throughout the simulation, making $x(t)_{\text{interferer}}$ and $x(t)_{\text{intruder}}$ necessary in Equation (2.4). As the Interferer, we assume that we know the location, speed, and direction of movement of both agents throughout the simulation, and incorporate this spatial data into the B and C matrices. Additionally, we assume that the Interferer knows the Intruder output g and the location of the Defender.

2.2 Problem Formulation

Throughout our experiments, we assume the role of the Interferer. Our objective is to find an acoustic output f such that when f is added to the acoustic environment, the Defender

does *not* perceive the presence of the Intruder. In this section, we formulate the optimization problem used to find this acoustic output. Indices, data, decision variables, and formulation are presented first, followed by a brief discussion of each formulation component.

Index [$\sim$ cardinality]:

$t \in T$: Discretized time index [~ 6000]

$\psi \in \Psi$: Spatial index resulting from FEM discretization of 2D space [$\sim 11, 836$]

Given Data [units]:

L : Defender's loss function

Φ : Defender's neural network

$y \in \{0, 1\}$: Spectrogram label, where $y = 1$ if an Intruder is present, $y = 0$ otherwise

$A \in \mathbb{R}^{(|\Psi||T|) \times (|\Psi||T|)}$: Matrix from Equation (2.5) operating on state space

$B \in \mathbb{R}^{(|\Psi||T|) \times |T|}$: Matrix from Equation (2.5) operating on Interferer's output

$C \in \mathbb{R}^{(|\Psi||T|) \times |T|}$: Matrix from Equation (2.5) operating on Intruder's output

g_t : Intruder's acoustic output vector for each time step $t \in T$ [pressure]

$D \in \mathbb{R}^{|T| \times (|\Psi||T|)}$: Matrix extracting subset of state space at Defender's location

F : Short-time Fourier Transform operator

Decision Variables [units]:

f_t : Interferer's acoustic output vector for each time step $t \in T$ [pressure]

$w_{\psi,t}$: Acoustic pressure field vector for each location $\psi \in \Psi$, time $t \in T$ [pressure]

s_t : A vector of the signal received by Defender for each time step $t \in T$ [pressure]

$\hat{s}$: Spectrogram of signal at Defender's location

Formulation:

$$\max_{f,w,s,\hat{s}} L(\Phi(\hat{s}), y) \quad (2.6a)$$

$$\text{s.t.} \quad Aw = Bf + Cg \quad (2.6b)$$

$$Dw = s \quad (2.6c)$$

$$|Fs|^2 = \hat{s} \quad (2.6d)$$

Objective Function

The objective function (2.6a) is the loss function used to train the Defender's neural network. In our case, we use binary cross entropy loss with a logit transformation, but other loss functions could be used in a multilabel classification or regression context. The spectrogram $\hat{s}$ is input into the Defender's neural network Φ for classification. The expression $\Phi(\hat{s})$ returns the network's predicted probability of Intruder presence, while y represents the true presence ($y = 1$) or absence ($y = 0$) of an Intruder. $L(\Phi(\hat{s}), y)$ then returns the Defender's loss for a particular spectrogram $\hat{s}$ and true label y .

Since we assume the role of the Interferer, we seek to induce misclassification by *maximizing* the Defender's loss function. Note that the state space w , the subset of the state space at the Defender's location s , and the spectrogram $\hat{s}$ are all decision variables in the objective function. However, they are each uniquely determined by f , the Interferer output. This highlights the nature of the problem we seek to solve, namely that the input we seek to perturb, $\hat{s}$, can be modified only *indirectly* by modifying the state space w in which the agents reside.

Finite Element Method Discretization of Inhomogeneous Wave Equation

The FEM discretization of the Inhomogeneous wave equation outlined in Equation (2.5) is given by constraint (2.6b). It is used to model the acoustic environment w based on Interferer and Intruder outputs f and g , respectively. See Section 2.1.1 for more details.

Signal Received by the Defender

Given the state space w , we are concerned only with the subset of w that the Defender measures. The matrix D in constraint (2.6c) is a $|T| \times (|\Psi||T|)$ matrix such that the matrix multiply Dw extract the Defender's measurement from the state space w . We refer to the signal received by the Defender as s_t , which represents the acoustic pressure at the Defender's location at time t . This vector is used by the Defender to compute a spectrogram.

Short-time Fourier Transform

For details of the Short-time Fourier Transform operator F in constraint (2.6d) and the information captured by a spectrogram, see Section 1.5. We remind the reader that Fs produces a complex number for each time bin τ and frequency bin ω of the STFT, while the spectrogram $\hat{s}$ is found by taking the square of the modulus for each (τ, ω) (Leiber et al. 2022).

2.3 Gradient Computation and Interferer Output Constraints

In this section we outline key components of our approach to solving Problem 2.6. The first two subsections discuss related but distinct methods for finding the gradient of the Defender's loss function L with respect to the Interferer output f —a key component to solving our problem. We include both methods to illustrate different ways in which an adversary may leverage knowledge of the physical environment and the structure of neural networks to compromise physical sensors. In the third subsection, we discuss how and why we constrain the Interferer output f to a particular range of frequencies.

2.3.1 Reduced Gradient Method

We use the Python programming language to model our environment, and a software package PyTorch to model the neural networks in particular. We discuss PyTorch in greater detail in Chapter 3, but note here that PyTorch provides a wide range of capabilities for working with neural networks. One capability is back-propagation. Back-propagation is a technique that iteratively modifies the weights of a neural network to reduce the discrepancy between its inputs and desired outputs. At the heart of back-propagation is the chain rule, which is used to determine the gradient of the loss function with respect to specified inputs

(Goodfellow et al. 2016). This gradient is used to iteratively update the weights of the neural network in an effort to reduce the value of the loss function. For the original algorithm, see Rumelhart et al. (1986), and for a more practitioner-focused discussion, see section 6.5 of Goodfellow et al. (2016).

The significance of the PyTorch back-propagation capability here is that we use it to compute the gradient of the Defender's loss function L with respect to the acoustic state space w directly. We can express the objective function from Problem (2.6) in terms of w , since

$$L(\Phi(\hat{s}), y) = L(\Phi(|Fs|^2), y) = L(\Phi(|FDw|^2), y), \quad (2.7)$$

and denote this representation of the objective function as $\gamma(w)$. This allows us to express our problem in the form

$$\begin{aligned} \min_{f, w} \quad & \gamma(w) \\ \text{s.t.} \quad & Aw = Bf + Cg. \end{aligned} \quad (2.8)$$

While we can use Pytorch's back-propagation capability to compute $\nabla_w \gamma(w)$, we ultimately want to compute the gradient of $\gamma(w)$ with respect to our control variable f rather than w . In other words, we seek to find $\nabla_f \gamma(w)$. Solving for w in terms of f , we have

$$w = A^{-1} (Bf + Cg), \quad (2.9)$$

allowing us to express the objective as

$$\gamma \left(A^{-1} (Bf + Cg) \right). \quad (2.10)$$

Taking the gradient with respect to f , we have

$$\begin{aligned} & (A^{-1}B)^{\top} \nabla_w \gamma \left(A^{-1} (Bf + Cg) \right) \\ &= B^{\top} A^{-\top} \nabla_w \gamma \left(A^{-1} (Bf + Cg) \right). \end{aligned} \quad (2.11)$$

While this approach is analytically straightforward, the size of A is a critical limitation. To

illustrate, consider a dense 100×100 discretized space Ψ over a period of 100 time steps T . Assuming each entry of A is stored as a double precision floating point number (8 bytes) and recalling that A has dimensions $((|\Psi||T|) \times (|\Psi||T|))$, A would be a nearly 8 terabyte matrix in this simple example. Adding granularity in time and/or space increases A to even more unmanageable sizes.

Though we want to avoid inverting A directly as in Equation (??), we note that we are able to solve linear systems in A . A has a block upper triangular structure (Hyde 2023), which greatly reduces memory requirements and improves the efficiency of solving linear systems. This structure of A enables us to implement the reduced gradient method (RGM) as explained in Luenberger and Ye (2015). This method involves calculating $\nabla_f \gamma(f) = B^\top A^{-\top} \nabla_w \gamma(A^{-1}(Bf + Cg))$ (see, e.g., Section 12.6 of Luenberger (1984)) in the following manner:

1. Solve for w in $Aw = Bf + Cg$.
2. Solve for λ in $A^\top \lambda + \nabla_w \gamma(w) = 0$.
3. Return $-B^\top \lambda$. This is $\nabla_f \gamma(f)$.

Solving for w in (1) yields

$$w = A^{-1}(Bf + Cg) \quad (2.12)$$

as before, and substituting into (2) yields

$$\lambda = A^{-\top} \nabla_w \gamma(A^{-1}(Bf + Cg)). \quad (2.13)$$

Substituting the above into (3), we have

$$B^\top A^{-\top} \nabla_w \gamma(A^{-1}(Bf + Cg)), \quad (2.14)$$

which matches the analytically derived gradient $\nabla_f \gamma(f)$. The significance of this method is that by solving (1), (2), and (3) directly, we are able to obtain $\nabla_f \gamma(f)$ without performing operations that require storing the entirety of A in memory. In concert with the back-propagation capability provided by PyTorch, this method provides a computationally tractable way to obtain the gradient of the Defender's loss function L with respect to

Interferer output f .

2.3.2 Shortcut Method

In this section, we discuss a technique which reduces the number of operations necessary to obtain a spectrogram $\hat{s}$ given Interferer and Intruder outputs f and g . We refer to this as the “shortcut method” as it takes advantage of particular characteristics of our problem to avoid computing the entire state space w for given f and g . To produce a spectrogram $\hat{s}$ for given f and g according to mathematical program 2.6, we take the following steps:

1. Solve for w in $Aw = Bf + Cg$.
2. Solve for s in $Dw = s$.
3. Convert to s to $\hat{s}$ by taking $|Fs|^2 = \hat{s}$, where F is the STFT operator.

We note that (1) takes $f, g \in \mathbb{R}^{|T|}$ and calculates $w \in \mathbb{R}^{|\Psi||T|}$, while (2) takes $w \in \mathbb{R}^{|\Psi||T|}$ and calculates $s \in \mathbb{R}^{|T|}$. Diagrammatically, we have

$$f, g \in \mathbb{R}^{|T|} \xrightarrow{1} w \in \mathbb{R}^{|\Psi||T|} \xrightarrow{2} s \in \mathbb{R}^{|T|}, \quad (2.15)$$

where the arrows represent the operations of (1) and (2). Instead, we use a method which eliminates the need to compute w in its entirety, but rather takes $f, g \in \mathbb{R}^{|T|}$ directly to $s \in \mathbb{R}^{|T|}$. We call this the shortcut method:

$$f, g \in \mathbb{R}^{|T|} \xrightarrow{\text{shortcut method}} s \in \mathbb{R}^{|T|}. \quad (2.16)$$

The shortcut method requires a handful of straightforward operations. Solving for w as in RGM, we have:

$$Aw = Bf + Cg \Rightarrow w = A^{-1}(Bf + Cg). \quad (2.17)$$

Substituting this expression of w into (2), we have

$$Dw = D(A^{-1}(Bf + Cg)) \Rightarrow Dw = DA^{-1}Bf + DA^{-1}Cg. \quad (2.18)$$

As discussed before, however, directly inverting A is infeasible. Instead we introduce a new

matrix Y defined by

$$Y = DA^{-1}. \quad (2.19)$$

We solve for this matrix Y by first right multiplying the nonsingular A matrix on both sides of this equation to yield

$$Y = DA^{-1} \Rightarrow YA = D, \quad (2.20)$$

then transposing both sides of the equation to yield

$$Y = DA^{-1} \Rightarrow A^T Y^T = D^T. \quad (2.21)$$

This gives us a system of linear equations which allows us to solve for Y^T (and thus Y) without having to invert A . Substituting Y back into our expression of Dw in Equation (2.18), we have

$$Dw = YBf + YCg \Rightarrow s = YBf + YCg. \quad (2.22)$$

We see that the resulting matrices YB and YC , both $\in \mathbb{R}^{|T| \times |T|}$, are used to take $f, g \in \mathbb{R}^{|T|}$ directly to $s \in \mathbb{R}^{|T|}$. In practice, we compute YB and YC based on the location of the agents and the Intruder output g offline during the initialization of an experiment and store them for future use.

The benefits of the shortcut method are twofold. First, we are able to obtain spectrograms at the Defender's location in an efficient manner in terms of both memory and time. Second, the relative simplicity of this method enables us to compute the gradient of the Defender's loss function with respect to the Interferer output f via back-propagation in PyTorch. The combined impact of these two factors is a more than 1000 times improvement in runtime for experiments conducted in the next chapter.

2.3.3 Projected Gradient Descent

Both RGM and the shortcut method allow us to compute the gradient of the Defender's loss function L with respect to Interferer output f . In both cases, however, f itself remains

entirely unconstrained. In practice, we may wish to constrain f to a particular range of frequencies. We may wish, for example, to emit an output mimicking frequencies associated with wildlife or other atmospheric noise, commercial shipping or recreational vessels, or even other adversarial actors. Additionally, even if abnormally high or low frequencies do not alert the Defender to the presence of the Intruder, such activity may draw the attention of other physical sensors or signal processing personnel.

Furthermore, we need to restrict the Interferer emission to a range of frequencies the Interferer's speaker is *capable* of emitting. Left unconstrained, the most successful candidates for Interferer outputs obtained in the next chapter lie in the range of 0 to 10 Hz, a range most typically associated with seismic activity (Lamb et al. 2021). Even if we were willing to generate such a low frequency, we might be incapable of doing so if those frequencies fell outside the range of frequencies the speaker can produce.

To constrain our Interferer output, we ensure that particular entries of the spectrogram of f are zero. To do so, we recall that the spectrogram of f is computed via the STFT

$$\text{STFT}_f[\tau, \omega] = \sum_{k=0}^{K-1} w(k) f(\tau H + k) e^{-j \frac{2\pi}{K} \omega k}. \quad (2.23)$$

We define $\mathcal{T}$ as the set of all time bins τ and Ω as the set of all frequency bins ω produced by the STFT, and encode it in a matrix F such that

$$\text{STFT}_f[\tau, \omega] = F f. \quad (2.24)$$

F has dimensions $(|\Omega| |\mathcal{T}|) \times (T)$, with each row corresponding to the index of a time bin τ and frequency bin ω .

The structure of the matrix F is crucial to the discussion that follows. Denote by Q the $|\Omega| \times K$ matrix which performs the DFT on a single frame, defined by

$$Q = \begin{pmatrix} w(0)e^{-j\frac{2\pi 0}{K}\omega_1} & w(1)e^{-j\frac{2\pi 1}{K}\omega_1} & w(2)e^{-j\frac{2\pi 2}{K}\omega_1} & \dots & w(K-1)e^{-j\frac{2\pi(K-1)}{K}\omega_1} \\ w(0)e^{-j\frac{2\pi 0}{K}\omega_2} & w(1)e^{-j\frac{2\pi 1}{K}\omega_2} & w(2)e^{-j\frac{2\pi 2}{K}\omega_2} & \dots & w(K-1)e^{-j\frac{2\pi(K-1)}{K}\omega_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w(0)e^{-j\frac{2\pi 0}{K}\omega_{|\Omega|}} & w(1)e^{-j\frac{2\pi 1}{K}\omega_{|\Omega|}} & w(2)e^{-j\frac{2\pi 2}{K}\omega_{|\Omega|}} & \dots & w(K-1)e^{-j\frac{2\pi(K-1)}{K}\omega_{|\Omega|}} \end{pmatrix}. \quad (2.25)$$

The STFT matrix F can be written in block matrix structure using only the matrix Q , where each block row comprises $|\Omega|$ rows, contains a single Q matrix, and is otherwise zero. The structure of F is such that each block row contains a zero matrix of size $|\Omega| \times H(i-1)$ preceding the Q matrix, then the Q matrix, and a zero matrix filling the remaining $|\mathcal{T}| - Hi$ columns following the Q matrix.

In the same way, we can construct a matrix $F_{\tilde{\Omega}}$ such that $F_{\tilde{\Omega}}f = 0$ constrains frequencies in some disallowed set of frequencies $\tilde{\Omega} \subseteq \Omega$ to be zero. Take $Q_{\tilde{\Omega}}$ to be the rows of Q corresponding to frequencies in $\tilde{\Omega}$. Build $F_{\tilde{\Omega}}$ from $Q_{\tilde{\Omega}}$ analogously to how F is built from Q by staggering the appearance of the $Q_{\tilde{\Omega}}$ matrix by H columns in each row block. We then add the constraint

$$F_{\tilde{\Omega}}f = 0. \quad (2.26)$$

to the mathematical program 2.6.

$F_{\tilde{\Omega}}$ includes only the rows of F that correspond to any frequency bin ω containing frequencies in $\tilde{\Omega}$. To facilitate computation over the real numbers, we reformulate this constraint using a real-valued matrix Z . The matrix Z is defined as

$$Z = \begin{bmatrix} \Re(F_{\tilde{\Omega}}) \\ \Im(F_{\tilde{\Omega}}) \end{bmatrix}, \quad (2.27)$$

where $\Re$ and $\Im$ are the operators that extract the real and imaginary parts, respectively, from a matrix. We see that the linear constraint

$$Zf = 0 \quad (2.28)$$

is equivalent to Equation (2.26), and adopt it as our means to constrain the Interferer output f .

A natural way to proceed while meeting the constraint in Equation (2.28) is projected gradient descent (see, e.g., Section 2.D of Royset and Wets [2022]). Given an initial feasible point $f^{(0)}$ and step size α , we set $\nu = 0$ and iteratively compute

$$f^{(\nu+1)} = f^{(\nu)} + \alpha P_{(Z^\perp)} d \quad (2.29)$$

for each iteration ν until a chosen set of stopping criteria is met. Typical stopping conditions include some combination of maximum iterations and tolerance for objective function improvement. Here, d is defined as $\nabla L(f^{(\nu)})$, the gradient of the objective function at $f^{(\nu)}$. We observe that there are many methods used in practice to determine suitable choices for α , and both RGM and the shortcut methods discussed in Sections 2.3.1 and 2.3.2 can compute d . The final component needed is $P_{(Z^\perp)}$, a matrix projecting onto the null space of Z . Similarly, to $P_{(Z^\perp)}$ denoting a projection onto the null space of Z , we denote by P_Z a projection matrix onto the column space of Z .

To construct $P_{(Z^\perp)}$, a common and intuitive approach would be to take the identity matrix minus the projection onto the column space of Z^T , defined as

$$I - Z^T (ZZ^T)^{-1} Z. \quad (2.30)$$

However, since Z is not full row rank, ZZ^T is not invertible. To address this issue, we seek some matrix M with full row rank such that

$$Mx = 0 \iff Zx = 0, \quad (2.31)$$

and take $I - P_{(M^T)}$ as our projector. To do so, we take the singular value decomposition of M as $U\Sigma V^T$. We examine the singular values in Σ , and take V_z as the columns from V for which the associated entries of Σ are zero. We then take our projector onto the null space of M as

$$I - P_{(M^T)} = V_z V_z^T. \quad (2.32)$$

By definition of M , we finally take our projector onto the null space of Z as

$$P_{(Z^\perp)} = I - P_{(Z^T)} = V_z V_z^T. \quad (2.33)$$

Equivalently, we note that one could take

$$P_{(Z^\perp)} = I - V_{nz} V_{nz}^T, \quad (2.34)$$

where V_{nz} are the columns V for which the associated entries of Σ are *nonzero*. Similarly, one might construct a projector onto the null space of M by taking the QR factorization of M^T , extracting the rows of Q corresponding to the nonzero rows of R , and taking

$$P_{(Z^\perp)} = Q^T Q. \quad (2.35)$$

For our implementation, we use the projector defined in Equation (2.32). This enables us to employ projected gradient descent, which keeps the underlying optimization formulation as simple as possible. The simplicity of this technique results in higher computational efficiency than directly incorporating the constraint in Equation (2.28) into our formulation, and a more scalable approach for use in practice.

CHAPTER 3:

Experiment and Results

In this chapter, we describe our experiments and their results. In the first two sections, we present the framework in which our experiments take place and discuss our approach to train the Defender’s neural network. Next, we describe our implementation of the methodology captured in Chapter 2 and highlight their results. We conclude with a brief synopsis of our findings and propose directions for further work in this area.

3.1 Experimental Framework

3.1.1 Software

We use the Python programming language for our experiments, capitalizing on the capabilities provided by the FEniCS Project and PyTorch package. The FEniCS Project is an open source, collaborative effort aimed to improve mathematical modeling with a focus on partial differential equations in particular (Logg and Wells 2010). At the heart of FEniCS is the DOLFIN library, which uses the finite element method to solve partial differential equations (Logg and Wells 2010). Using DOLFIN, we take the same approach as Hyde (2023) to solve the linear approximation of the inhomogeneous wave equation found in (2.5). This allows us to model the state space w according to Interferer and Intruder outputs f and g .

For the neural networking components of our experimentation, we harness the PyTorch package. PyTorch provides a wide range of capabilities making it well-suited for our application, two of which are worth particular mention. First, PyTorch enables the use of state of the art, pretrained neural networks by the Defender. These networks embody the cutting edge of AI in our setting, and incorporating them enhances the realism of our scenario. Second, PyTorch provides automatic differentiation, which is critical given the indirect nature of the Interferer perturbations.

3.1.2 Experimental Parameters

For clarity, we provide the exact parameters used in our experiments. Table 3.1 gives the spatial and temporal bounds, wave speed, and Defender location. We remind the reader that the continuous spatial and temporal parameters captured here are discretized as outlined in Section 2.2. We conduct our experiments using the first quadrant of a Cartesian plane, in which coordinates given by (x_1, x_2) denote meters of displacement from the origin. Table 3.2 gives the location and movement of the Intruder and Interferer throughout our experiments, captured visually in Figure 3.1. We note that the methodology presented in Chapter 2 generalizes to any choice of parameters in Tables 3.1 and 3.2.

Table 3.1. Experimental bounds and constant parameters

Parameter	Value
Time Horizon	1 second
Spatial Region	100 m $\times$ 100 m
Wave Speed	1525 m/s
Defender Location (x_1, x_2)	(52.5, 12.5)

Table 3.2. Location and movement of adversarial agents

Agent	Location (x_1, x_2)		Speed
	Initial	Final	
Intruder	(5, 75)	(15, 75)	10 m/s
Interferer	(9.75, 68.75)	(19.75, 68.75)	10 m/s

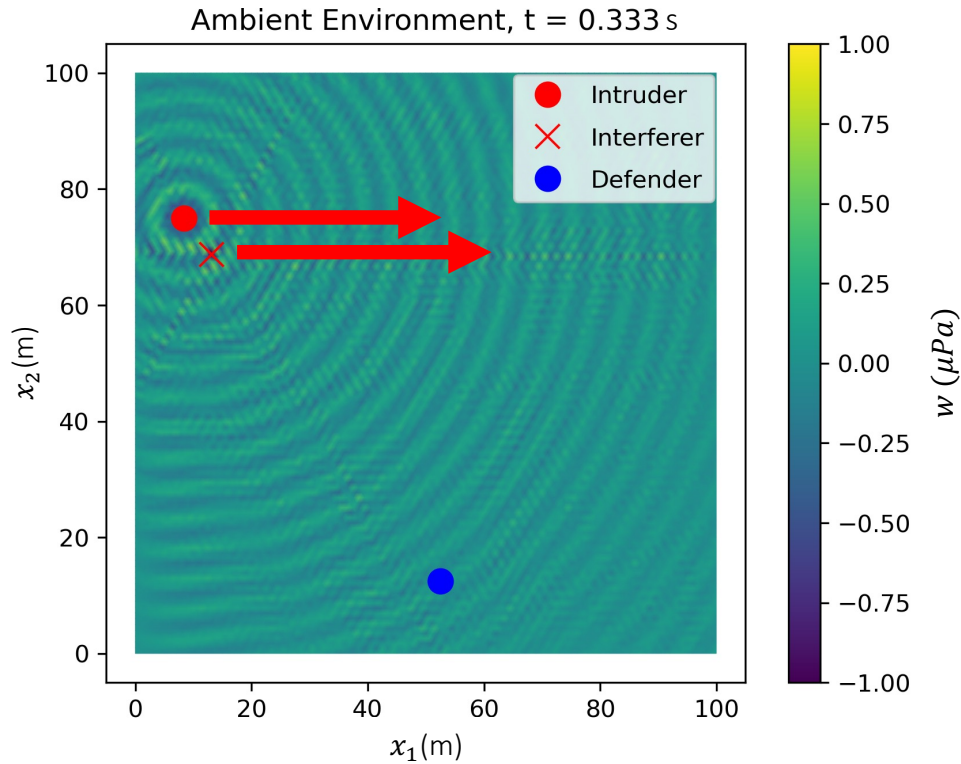


Figure 3.1. Using the DOLFIN library to model the physical environment. The state space w is simulated for a $100\text{m} \times 100\text{m}$ region based on Interferer output f and Intruder output g . In this still frame, the more prominent rings result from the Intruder output g , while the smaller agitations result from Interferer output f . The arrows represent the direction of movement of these agents.

3.2 Defender's Neural Network

3.2.1 Data Generation

To obtain training data for the Defender's neural network, we generate spectrograms from Intruder or non-hostile outputs *absent* Interferer output, and subsequently introduce random noise. There are three primary advantages to this approach. First, we guarantee generated data that is sufficient in both quantity and quality to adequately train the network. Second, we ensure that neither the Defender's neural network nor our attempts to deceive it are unduly influenced by variables outside our control. Third, we are able to compare the Defender's

success with and without Interferer outputs in an otherwise identical environment. We note that true physical experiments guarantee none of these advantages, and further that in practice, one could easily substitute empirically obtained spectrograms for the dataset described here.

Intruder vs Non-Hostile Spectrograms

A critical choice in data generation is the delineation of Intruder outputs from non-hostile outputs. While this distinction can be made in many different ways, we assume that acoustic outputs of 400 Hz or fewer belong to the Intruder while acoustic outputs greater than 400 Hz but less than 800 Hz are non-hostile. This range of frequencies is typically associated with shipping and industrial activity as well as oceanic wildlife (Wenz 1962), which provides us with a plausible domain in which non-hostile actors are readily found and adversarial agents may seek to hide. A graphical depiction of the typical frequency ranges associated with oceanic noise sources is presented in Figure 3.2. To provide a concrete framework, we adopt the specific sets of Intruder and non-hostile frequencies listed in 3.3. Crucially, we note that modifying the range and labels of frequencies used to generate our data has no demonstrable impact on our ability to generate perturbations.

Table 3.3. Intruder and non-hostile frequencies

Label	Frequencies
Intruder	{10, 20, ..., 400} Hz
Non-Hostile	{410, 420, ..., 800} Hz

For each of the 40 frequencies contained in Table 3.3, we generate spectrograms in the following manner. First, we assign each frequency to a source sharing the same location and behavior as the Intruder captured in Table 3.2 and depicted in 3.1. This ensures that differences between spectrograms containing Intruder and non-hostile output are the product of differences in frequency vice location. In all cases, we configure the source output as a sine wave with amplitude $20 \mu Pa$ and populate s , the signal received by the Defender.

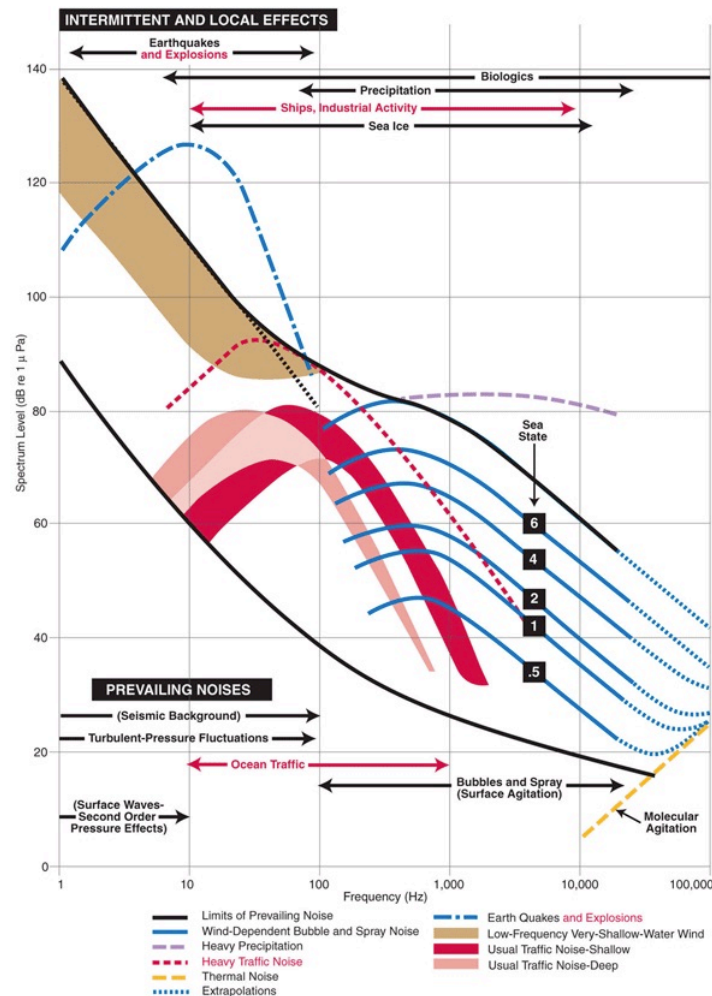


Figure 3.2. Frequency and intensity ranges of common acoustic sources in the ocean. The range of frequencies we use in training is commonly associated with shipping, industrial, and oceanic activity. Adapted from Wenz (1962))

Finally, the spectrogram $\hat{s}$ is obtained via the STFT of s as outlined in 1.4 ². Figure 3.4 shows two examples of these spectrograms both prior to incorporating random noise.

Adding Noise

In practice, the signal s captures not only the source's acoustic output, but some level of environmental noise and/or sensor distortion. To capture this dynamic, we introduce complex

²We employ the PyTorch defaults for window length (256) and hop length (64) in conjunction with the Hann window function. This resulting spectrogram has 94 time bins and 129 frequency bins.

Gaussian noise to the STFT of s ³. Two primary considerations arise when incorporating this noise. First, it is common knowledge in the field of underwater acoustics that high-frequency outputs tend to attenuate more rapidly than lower-frequency outputs. To add an appropriate amount of noise to each spectrogram, we model the standard deviation of the noise added to the STFT of s as a function of the source frequency from which it is produced. To derive this function, we fit a log-linear regression to the root mean square (RMS) of the received signal in relation to the source frequency, as illustrated in Figure 3.3. Second, we need to determine how *much* noise to add. By introducing a coefficient of 40 to the standard deviation, we achieve a spectrogram that contains a significant amount of random noise while the underlying signal is still visible in the spectrogram. Figure 3.4 shows examples of noise added to Intruder and non-hostile spectrograms in this manner.

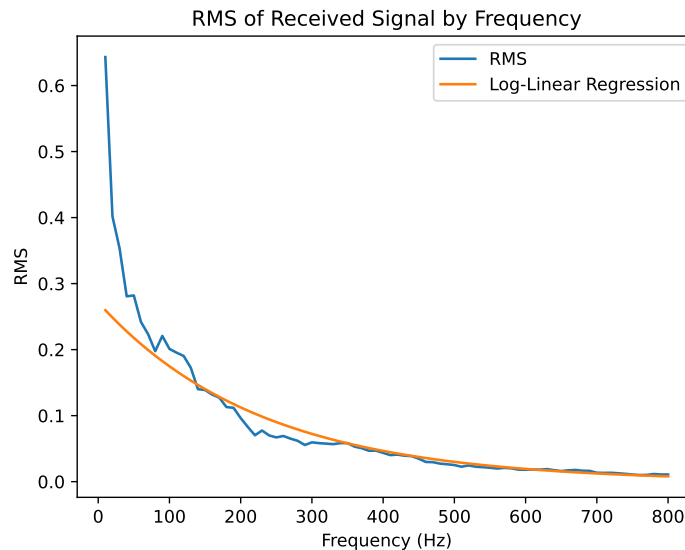


Figure 3.3. The RMS of the received signal s at the Defender's location as a function of source frequency is depicted here in blue. We take the log-linear regression of the source frequency (depicted in orange) as the standard deviation to add noise to each resulting spectrogram.

³Adding Gaussian noise directly to the state space w or s has an equivalent effect, but adding noise to the STFT of s offers advantages in computational efficiency.

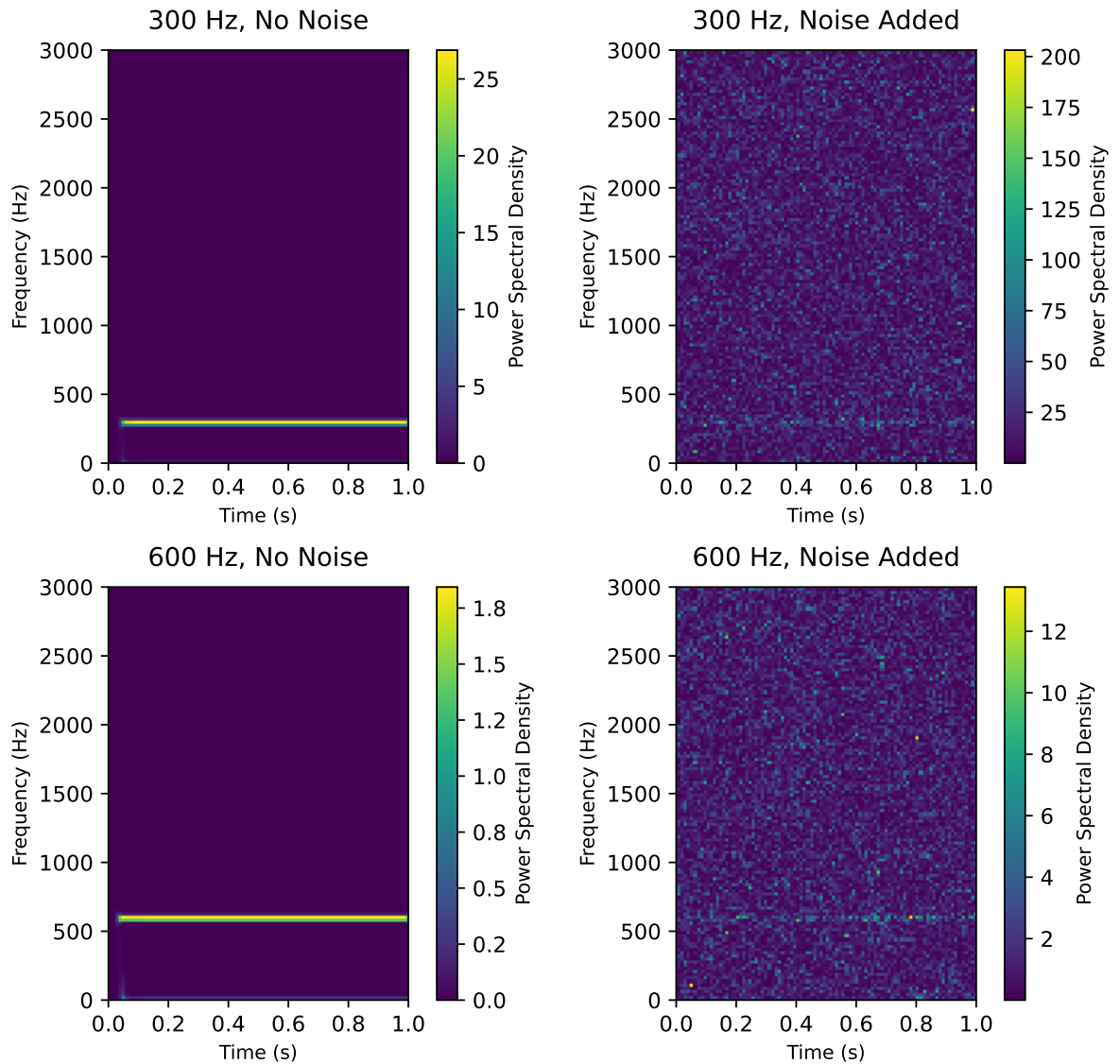


Figure 3.4. On the left are spectrograms capturing acoustic outputs of an Intruder (top) or a non-hostile agent (bottom) only. The spectrograms on the right result from adding complex Gaussian noise, and are typical examples of the data we use for transfer learning.

Final Dataset

The final issue to attend to in terms of data generation is determining how much to generate. For each of the frequencies listed in Table 3.3, we add random noise to produce a total of

40 unique spectrograms. As the length of our experiment is 1 second, this is the equivalent of a 40 second time sample for each frequency. In the aggregate, this results in a dataset of 3,200 spectrograms, evenly split between Intruder and non-hostile. Though augmentation of this dataset is straightforward, we observe that the neural networks trained on this dataset exceed a 95% accuracy rate on the test set. Further, augmentation of the dataset beyond 3,200 spectrograms had little impact on the Defender's success rates with or without Interferer presence.

3.2.2 Transfer Learning

Transfer learning is a method often used in practice which aims to transfer knowledge from one domain to another (Zhuang et al. 2019). The intent is to capitalize on information contained in a pre-trained neural network under the premise that the network is suitable for the task at hand. Typically, it involves taking a pre-trained neural network, exposing that neural network to different data than it was trained on, and fine tuning that neural network to the new data by modifying only a subset of the neural network parameters. This practice is highly effective in the context of image classification, and we refer the reader to (Zhuang et al. 2019) for a comprehensive survey of the topic.

We conduct transfer learning on three different state of the art neural networks. Inception Version 3 (V3) (Szegedy et al. 2015), GoogLeNet (Szegedy et al. 2014), and VGG-19 (Simonyan and Zisserman 2014) boast impressive classification rates on ImageNet, the hallmark dataset for image classification (Deng et al. 2009). ImageNet consists of millions of images belonging to thousands of categories, and one of the benchmark evaluation criteria for a neural network is its top-5 classification rate. This metric captures how frequently the true label of an image resides within the top 5 predicted labels of a neural network. Inception V3, GoogLeNet, and VGG-19 have among the best top-5 classification rates in the world at 3.4% (Szegedy et al. 2015), 6.6% (Szegedy et al. 2014), and 7.3% (Simonyan and Zisserman 2014), respectively. These networks exemplify the cutting edge of AI with potential for DOD application in our setting.

While the architectures of Inception V3, GoogLeNet, and VGG-19 are rather complicated and can be found in the original works, we modify only the final fully connected layer (or layers) of these models while holding all other parameters fixed. This allows us to leverage

the important information encoded in the convolutional layers of these networks while fine-tuning them to the data described in Chapter 3, Section 2.1. To test the success of the resultant models, we evaluate them on a test set of spectrograms consisting of 10 spectrograms for each frequency used in training. This 800 spectrogram data set is generated in the same manner as the training set, but with distinct random noise such that the training set described in Chapter 3, Section 2.1 is disjoint from the testing set. To evaluate the robustness of the networks, we repeat this procedure using scaling coefficients slightly above and below the coefficient used in training. These values, and the resulting accuracies of the models, are captured in Table 3.4. The accuracies achieved at higher and lower levels of noise than the models were trained on showcase their effectiveness reinforce their potential.

Table 3.4. Model Accuracies at Varying Noise Levels

Model	Accuracy with Added Noise (%)				
	20	30	40	50	60
Inception V3	87	95	99	96	92
GoogLeNet	81	92	99	95	90
VGG-19	82	94	93	89	85

3.3 Perturbation Generation

To answer our research question, we seek to compute perturbations instances in which the Defender correctly identifies Intruder presence. In the context of our experiment, we search for elements of our test set for which the neural network returns a confidence of Intruder presence of greater than 0.5 *and* the underlying source frequency is 400 Hz or less. Numerical precision limitations result in some instances meeting this criteria being reported with a confidence of 1. In these cases, the binary cross entropy loss function we use in transfer learning

$$\text{BCE}(y, p) = -[y \log(p) + (1 - y) \log(1 - p)], \quad (3.1)$$

,where $y \in \{0, 1\}$ represents the true label of an input and p the models confidence in that label, becomes undefined since $\log(0)$ is undefined. Consequently, we are unable to take a gradient of the loss function. However, in our model architectures, we use the sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3.2)$$

to map the output value x of our neural network to a confidence of Intruder presence. We note that the sigmoid function maps any real number x to a value strictly between 0 and 1. Since we are unable to obtain a confidence level of 1 analytically and cannot take the gradient in such an instance, we omit these instances from consideration.

For instances in which the Defender's neural network correctly identifies Intruder presence, but with a probability less than 1, we employ projected gradient descent using a backtracking line search algorithm. To do so, we recall that given an initial feasible point $f^{(0)}$ and step size α , we set $k = 0$ and iteratively compute

$$f^{(k+1)} = f^{(k)} + \alpha P_{(Z^\perp)} d \quad (3.3)$$

for each iteration k until a chosen set of stopping criteria is met. The strong Wolfe line search is used to choose α at each iteration which satisfies conditions of sufficient decrease in objective function value and curvature (see Condition 6.3.3 in Dennis, Jr. and Schnabel [1996]). Together, these conditions serve to avoid large step sizes that slowly lead to convergence while ensuring improvement in objective function value (Dennis Jr and Schnabel 1996).

To compute the gradient d , defined as $\nabla L(f^{(k)})$, we can employ either RGM as outlined in Chapter 2, Section 3.1 or the shortcut method as outlined in Chapter 2, Section 3.2. For scalability, we employ the shortcut method as it is much more computationally efficient. Finally, we construct a projector $P_{(Z^\perp)}$ using the method described in Chapter 2, Section 3.3, which restricts the frequency of the Interferer output f to greater than 500 Hz. This restriction ensures the Interferer's acoustic output falls well outside the range of frequencies associated with the Intruder.

We iterate the projected gradient descent algorithm, checking the Defender's reported confidence of Intruder presence at each iteration until any of the following stopping criteria

are met:

1. Defender's reported confidence of Intruder presence is < 0.5
2. Maximum number of iterations
3. $\|\nabla L(f^{(k)})\| < \rho$, where ρ captures the minimally acceptable magnitude of $\|\nabla L(f^{(k)})\|$

In the event that Condition (1) is met, we have succeeded as the Interferer. Crucially, however, if Condition (1) is not met, it does *not* imply that there is no solution. We reiterate that we use hand tuned parameters in our line search algorithm, and further note that our approach employs only one optimization method. A more methodical choice of tuning parameters and/or an expanded use of optimization methods could result in additional instances of success on behalf of the Interferer. Nonetheless, our results allow us to answer our research question in the context of the physical environment we examine.

3.4 Interferer Impact

In each of the the neural networks we examine, we find an Interferer output which prevents the Defender from detecting Intruder presence in at least 50% of the cases meeting the criteria in Section 3.3. Table 3.5 gives the Defender success rates with and without Interferer presence. In many cases, we are able to achieve these results with Interferer outputs that are much smaller in magnitude than the Intruder *and* restricted to a particular set of frequencies. In these cases, the perturbed spectrograms look remarkably similar to the unperturbed spectrograms, as seen in Figures 3.6 and 3.5.

Table 3.5. Defender's Success Rate with and without Interferer Output

Model	Defender Success Rate	
	Intruder Only	Intruder and Interferer
Inception V3	94	43
GoogLeNet	92	50
VGG-19	89	0

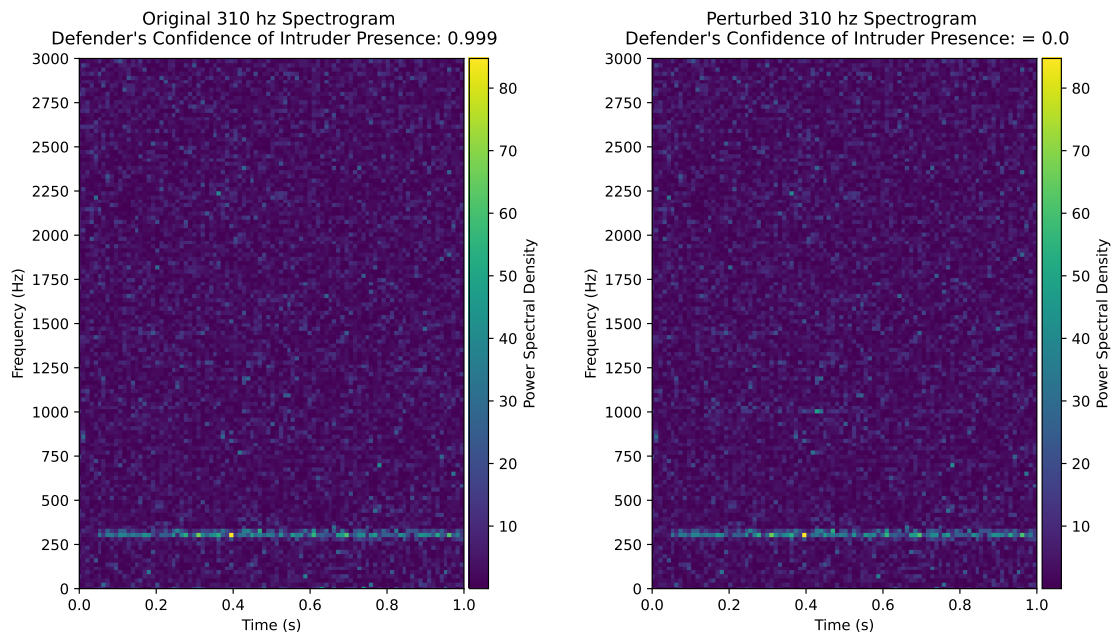


Figure 3.5. Spectrograms generated from an environment containing an Intruder with an acoustic output of 310 Hz. Without Interferer presence, the Defender's neural network returns a confidence of 0.999 of Intruder presence. However, incorporating the Interferer output in addition to the Intruder output, the Defender returns a confidence of 0 (rounded down due to numerical precision limitations) of Intruder presence. A slight difference between these spectrograms can be seen at about 1000 Hz, the frequency of our Interferer output.

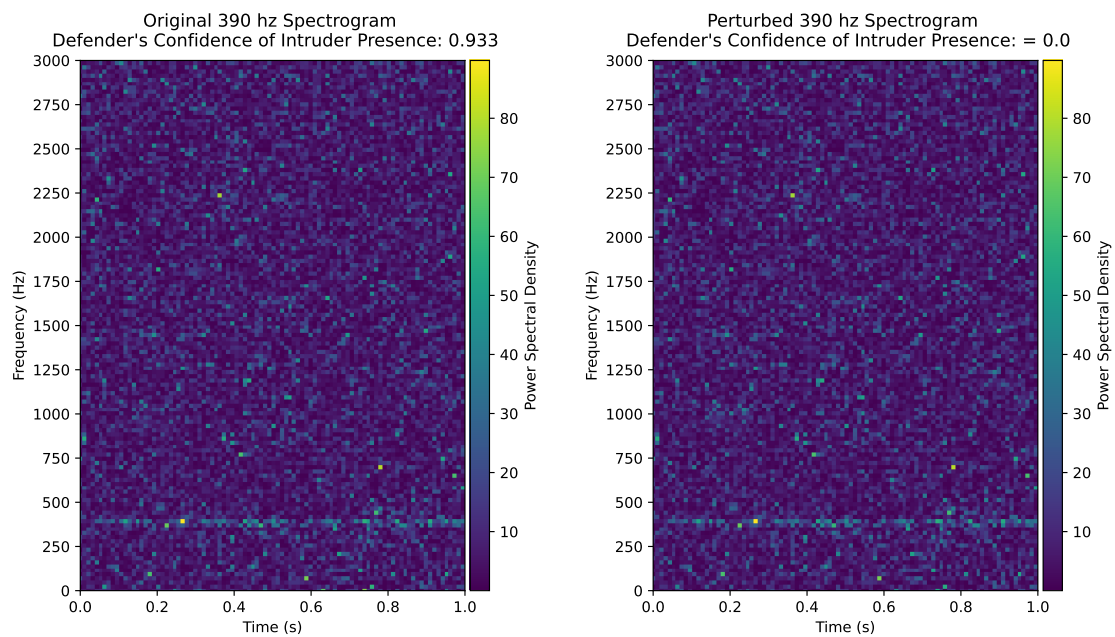


Figure 3.6. An example similar to Figure 3.5, with a less noticeable difference between the original and perturbed spectrograms.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:

Conclusions and Future Research

4.1 Conclusions

As we race to leverage the emerging capabilities of AI throughout the DOD, we must be cognizant of the risks they pose. This research has highlighted one such risk in an under-investigated area. We proposed a highly plausible context in which an adversary seeks to deceive a neural network employed in conjunction with a physical sensor. In contrast to the preponderance of literature on the subject, the adversary can only perturb neural network inputs *indirectly* by manipulating the physical environment in which it operates. After modeling an underwater acoustic environment, we successfully manipulate this underwater environment to prevent the sensor and neural network from detecting the adversary. Our research serves as an example of how state of the art neural networks can be deceived in a physical setting by integrating knowledge of the environment with established mathematical techniques. In short, we demonstrate that the lack of robustness in neural networks *can* be exploited to compromise physical sensors.

4.2 Future Research

There are many avenues to improve our proposed methodology and expand the context and relevance of this work. Within the existing problem scope and assumptions, we could increase the effectiveness of Interferer outputs with respect to two different metrics. First, we can increase how *often* we succeed. We employ only projected gradient descent with a backtracking line search and hand tuned parameters. While this suffices to answer our research question, it is likely that we could do succeed more consistently. Since we are able to compute the gradient of the Defender's loss function with respect to Interferer inputs, a broader range of optimization methods could be employed to lower the Defender's success rate via Inception V3, GoogLeNet, and other state of the art neural networks. Second, we can limit the *perceptibility* of our Interferer outputs in the perturbed spectrograms. Though Figures 3.5 and 3.6 show perturbed spectrograms that closely resemble the originals, we employ no direct mechanism to this end. The approach taken by Carlini and Wagner (2016)

referenced in Section 1.2 could limit the discrepancy between original and perturbed inputs as measured by a particular norm.

We may also consider modifying the assumptions of our problem. The novelty of this thesis is that we have replaced the assumption of logical access with that of physical access. While this more realistically captures an adversary's ability to manipulate neural network inputs in physical settings, the realism of the scenario could be further enhanced by limiting or removing knowledge of the Defender's network. Restrictions of this sort are commonly referred to as grey-box (limited knowledge) or a black-box (no knowledge) settings, and are applicable in the DOD for obvious reasons. In a black-box setting, Papernot et al. (2016) put forth a method to fool state of the art neural networks through the use of substitute models and synthetically generated data. Xie et al. (2018) and Kurakin et al. (2016) discuss similar methods for generating perturbations which are capable of fooling many different models simultaneously. We could explore ways to integrate these techniques with ours to improve the capabilities of the Interferer.

The suggested improvements to our current methodology and the introduction of a black box setting provide only a few examples of potential advancements in this area. The DOD relies on a wide variety of sensors in numerous contexts beyond the one presented here, many of which generate outputs suitable for integration with neural networks. As we seek to win the AI race, it is clear that further research in this area is crucial to understand the risks and vulnerabilities we will encounter.

List of References

- Abdollahpourroostam A, Abroshan M, Moosavi-Dezfooli SM (2023) Revisiting DeepFool: generalization and improvement. *arXiv*, arXiv:2303.12481, <https://doi.org/10.48550/arXiv.2303.12481>.
- Allen G (2020) Understanding AI technology. Technical report, Joint Artificial Intelligence Center (JAIC), The Pentagon, United States, <https://www.ai.mil/docs/Understanding%20AI%20Technology>.
- Alzantot M, Sharma Y, Chakraborty S, Zhang H, Hsieh CJ, Srivastava M (2018) GenAttack: Practical black-box attacks with gradient-free optimization. *arXiv* abs/1805.11090, <http://arxiv.org/abs/1805.11090>.
- Athalye A, Engstrom L, Ilyas A, Kwok K (2017) Synthesizing robust adversarial examples. *arXiv* abs/1707.07397, <http://arxiv.org/abs/1707.07397>.
- Austin III LJ (2022) *2022 National Defense Strategy*. Washington, DC, <https://media.defense.gov/2022/Oct/27/2003103845/-1/-1/1/2022-NATIONAL-DEFENSE-STRATEGY-NPR-MDR.PDF>.
- Barreno M, Nelson B, Joseph AD, Tygar JD (2010) The security of machine learning. *Machine Learning* 81:121–148, <https://doi.org/https://doi.org/10.1007/s10994-010-5188-5>.
- Bassett R, Graves M (2020) Color and edge-aware adversarial image perturbations. *arXiv* abs/2008.12454, <https://arxiv.org/abs/2008.12454>.
- Bassett R, Van Dellen A, Austin A (2023) Adversarial perturbations of physical signals. *Forthcoming*.
- Berger D (2019) *38th Commandant's Planning Guidance*. Marine Corps Publications Electronic Library, Quantico, VA, <https://www.marines.mil/News/Publications/MCPEL/Electronic-Library-Display/Article/1907265/38th-commandants-planning-guidance-cpg/>.
- Brown TB, Mané D, Roy A, Abadi M, Gilmer J (2017) Adversarial patch. *arXiv* abs/1712.09665, <http://arxiv.org/abs/1712.09665>.
- Carlini N, Wagner DA (2016) Towards evaluating the robustness of neural networks. *arXiv* abs/1608.04644, <http://arxiv.org/abs/1608.04644>.

- Chen PY, Zhang H, Sharma Y, Yi J, Hsieh CJ (2017) ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (ACM)*, <https://doi.org/10.1145/3128572.3140448>, New York, NY, USA.
- Choo Y, Lee K, Hong W, Byun SH, Yang H (2022) Active underwater target detection using a shallow neural network with spectrogram-based temporal variation features. *IEEE Journal of Oceanic Engineering*, <https://doi.org/10.1109/JOE.2022.3164513>.
- Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255, <https://doi.org/10.1109/CVPR.2009.5206848>, Miami Beach, FL.
- Dennis Jr JE, Schnabel RB (1996) *Numerical methods for unconstrained optimization and nonlinear equations* (Society for Industrial and Applied Mathematics), <https://doi.org/10.1137/1.9781611971200>.
- Dong J, Chen J, Xie X, Lai J, Chen H (2023) Adversarial Attack and Defense for Medical Image Analysis: Methods and Applications. *arXiv* arXiv:2303.14133, <https://doi.org/10.48550/arXiv.2303.14133>.
- Evtimov I, Eykholt K, Fernandes E, Kohno T, Li B, Prakash A, Rahmati A, Song D (2017) Robust physical-world attacks on machine learning models. *arXiv* abs/1707.08945, <http://arxiv.org/abs/1707.08945>.
- Finlayson SG, Kohane IS, Beam AL (2018) Adversarial attacks against medical deep learning systems. *arXiv* abs/1804.05296, <http://arxiv.org/abs/1804.05296>.
- Goodfellow I, Bengio Y, Courville A (2016) *Deep Learning* (MIT Press), <http://www.deeplearningbook.org>.
- Goodfellow IJ, Shlens J, Szegedy C (2014) Explaining and harnessing adversarial examples. *arXiv* arXiv:1412.6572, <https://doi.org/10.48550/arXiv.1412.6572>.
- He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv* abs/1502.01852, <http://arxiv.org/abs/1502.01852>.
- Hyde DE (2023) Active stealth in submerged environments: A numerical investigation. Master's thesis, Dept. of Math., NPS, Monterey, CA, USA.
- Jensen FB, Kuperman WA, Porter MB, Schmidt H (2011) *Computational Ocean Acoustics*, 2nd ed. (Springer, New York, NY), <https://doi.org/10.1007/978-1-4419-8678-8>.

- Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Pereira F, Burges C, Bottou L, Weinberger K, eds., *Advances in Neural Information Processing Systems*, volume 25 (Curran Associates, Inc.), https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf, Lake Tahoe, NV.
- Kurakin A, Goodfellow IJ, Bengio S (2016) Adversarial machine learning at scale. *arXiv abs/1611.01236*, <http://arxiv.org/abs/1611.01236>.
- Lamb OD, Lees JM, Malin PE, Saarno T (2021) Audible acoustics from low-magnitude fluid-induced earthquakes in finland. *Scientific reports* 11(1):19206.
- Leiber M, Barrau A, Marnissi Y, Abboud D (2022) A differentiable short-time fourier transform with respect to the window length. *2022 30th European Signal Processing Conference (EUSIPCO)*, 1392–1396, <https://doi.org/10.23919/EUSIPCO55093.2022.9909963>, Belgrade, Serbia.
- Logg A, Wells GN (2010) Dolfin: Automated finite element computing. *ACM Transactions on Mathematical Software* 37, <https://doi.org/10.1145/1731022.1731030>.
- Luenberger DG, Ye Y (2015) *Linear and Nonlinear Programming* (Springer, New York, NY), <https://doi.org/https://doi.org/10.1007/978-0-387-74503-9>.
- Mattis J (2018) Summary of the 2018 National Defense Strategy of the United States of America. Technical report, Department of Defense, Washington, United States, <https://dod.defense.gov/Portals/1/Documents/pubs/2018-National-Defense-Strategy-Summary.pdf>.
- Moosavi-Dezfooli SM, Fawzi A, Frossard P (2016) Deepfool: A simple and accurate method to fool deep neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV.
- Papernot N, McDaniel PD, Goodfellow IJ, Jha S, Celik ZB, Swami A (2016) Practical black-box attacks against deep learning systems using adversarial examples. *arXiv abs/1602.02697*, <http://arxiv.org/abs/1602.02697>.
- Papernot N, McDaniel PD, Jha S, Fredrikson M, Celik ZB, Swami A (2015) The limitations of deep learning in adversarial settings. *arXiv abs/1511.07528*, <http://arxiv.org/abs/1511.07528>.
- Płotka M, Abratkiewicz K, Malanowski M, Samczyński P, Kulpa K (2020) The use of the reassignment technique in the time-frequency analysis applied in vhf-based passive forward scattering radar. *Sensors* 20(12), <https://doi.org/10.3390/s20123434>.

- Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. *Nature* 323(6088):533–536, <https://doi.org/https://doi.org/10.1038/323533a0>.
- Sharma G, Umapathy K, Krishnan S (2020) Trends in audio signal feature extraction methods. *Applied Acoustics* 158:107020 (01), <https://doi.org/10.1016/j.apacoust.2019.107020>.
- Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. *arXiv* arXiv:1409.1556, <https://doi.org/10.48550/arXiv.1409.1556>.
- Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2014) Going deeper with convolutions. *arXiv* abs/1409.4842, <http://arxiv.org/abs/1409.4842>.
- Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2015) Rethinking the inception architecture for computer vision. *arXiv* abs/1512.00567, <http://arxiv.org/abs/1512.00567>.
- Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2013) Intriguing properties of neural networks. *arXiv* arXiv:1312.6199, <https://doi.org/10.48550/arXiv.1312.6199>.
- Wenz G (1962) Acoustic ambient noise in the ocean: Spectra and sources. *The Journal of the Acoustical Society of America* 34(12):1936–1956, <https://doi.org/https://doi.org/10.1121/1.1909155>.
- Xie C, Zhang Z, Wang J, Zhou Y, Ren Z, Yuille AL (2018) Improving transferability of adversarial examples with input diversity. *arXiv* abs/1803.06978, <http://arxiv.org/abs/1803.06978>.
- Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H, He Q (2019) A comprehensive survey on transfer learning. *arXiv* abs/1911.02685, <http://arxiv.org/abs/1911.02685>.

Initial Distribution List

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California



DUDLEY KNOX LIBRARY

NAVAL POSTGRADUATE SCHOOL

WWW.NPS.EDU

WHERE SCIENCE MEETS THE ART OF WARFARE