



Adversarial perturbations of physical signals

Robert L. Bassett^{1,2}  · Austin Van Dellen¹ · Anthony P. Austin²

Received: 19 March 2024 / Accepted: 29 November 2024 / Published online: 11 December 2024

This is a U.S. Government work and not under copyright protection in the US; foreign copyright protection may apply 2024

Abstract

We investigate the vulnerability of computer-vision-based signal classifiers to adversarial perturbations of their inputs, where the signals and perturbations are subject to physical constraints. We consider a scenario in which a source and interferer emit signals that propagate as waves to a detector, which attempts to classify the source by analyzing the spectrogram of the signal it receives using a pre-trained neural network. By solving PDE-constrained optimization problems, we construct interfering signals that cause the detector to misclassify the source even though the perturbations to the spectrogram of the received signal are nearly imperceptible. Though such problems can have millions of decision variables, we introduce methods to solve them efficiently. Our experiments demonstrate that one can compute effective and physically realizable adversarial perturbations for a variety of machine learning models under various physical conditions.

Keywords PDE-constrained optimization · Adversarial perturbations · Machine learning · Neural networks

1 Introduction

Advances in machine learning have produced advances in signal classification with perhaps the greatest leaps occurring in the field of computer vision. Many methods have been proposed that leverage techniques from computer vision to classify non-

✉ Robert L. Bassett
robert.bassett@nps.edu

Austin Van Dellen
austin.vandellen@nps.edu

Anthony P. Austin
anthony.austin@nps.edu

¹ Department of Operations Research, Naval Postgraduate School, 1 University Cir., Monterey, CA 93943, USA

² Department of Applied Mathematics, Naval Postgraduate School, 1 University Cir., Monterey, CA 93943, USA

visual signals by representing them as images. But despite their successes, computer vision methods have been shown to be susceptible to *adversarial perturbations*, minor modifications to an input image that cause dramatic changes in a classifier's output. The existence of adversarial perturbations demonstrates that many types of classifiers—in particular, deep neural networks—are not robust to small changes in their inputs.

In the wake of the seminal work of Szegedy et al. [1], many methods have been developed for constructing adversarial perturbations for neural network classifiers. Notable foundational examples include the fast gradient sign method of Goodfellow et al. [2], Deepfool [3], and Carlini–Wagner perturbations [4]. The zeroth-order optimization approach of Chen et al. [5] provides a way to compute adversarial perturbations in the absence of gradient information. Bassett et al. [6] construct perturbations which are less perceptible to an observer by accounting for human perception of color and texture.

All of these methods construct perturbations under the assumption that the adversary has what we call *logical access* to the classifier, i.e., the ability to modify inputs to the classifier directly. For computer vision, logical access is often an inappropriate assumption because in many applications, the classifier receives its input from a sensor, such as a camera, and an adversary has only *physical access* to the classifier via the sensor. That is, the adversary can only influence the inputs to the classifier by manipulating the sensor's physical environment. Physical access is a weaker assumption on an adversary's capabilities than logical access because constructing a perturbation that can be physically realized is more difficult than constructing a perturbation using digital artifacts that cannot be replicated in the sensor's environment.

Computer vision researchers have successfully constructed adversarial perturbations under physical access assumptions [7, 8]. In these works, the inputs to the neural networks are direct images of the physical objects being classified. In other applications of computer vision to signal classification, the relationship between the objects and their digital representations is less direct. Here, we consider applications in which univariate signals to be classified are represented as *spectrograms*, images that show how the signals' frequency content changes with time [9, Chapter 7]. Classifying signals via their spectrograms is both simple (since one can leverage off-the-shelf techniques from computer vision) and effective, having come to define the state of the art in several areas [10, 11].

Motivated by the success and popularity of spectrogram classification techniques, we investigate their susceptibility to adversarial perturbations under physical access assumptions. We model the physical access constraints using partial differential equations (PDEs), leading to PDE-constrained optimization problems for the perturbations. These problems are considerably more complicated than the optimization problems typically encountered in the adversarial perturbations literature; we show how to solve them efficiently. We also examine the effects of environmental noise, which usually has a larger impact on the kinds of signals classified by spectrogram—such as acoustic or (non-optical) electromagnetic signals—than it does for images.

2 Problem formulation

To make our discussion concrete, we consider the following scenario. An intruding submarine wishes to pass unnoticed through a region of seawater containing an acoustic detector. To accomplish this, the submarine employs an external interferer, which emits a perturbing signal to confuse the detector. The detector operates by using a neural network to process spectrograms of the acoustic signals it receives. Our aim is to select a perturbing signal that will cause the detector to misclassify a spectrogram and conclude that the submarine is not present, even though the perturbed spectrogram differs little from the unperturbed one.

Figure 1 illustrates the general setup. Our region of water is a rectangle $\Omega \subset \mathbb{R}^2$. The submarine and interferer are modeled as point sources at positions x_s and x_i , respectively, which may vary with time. The detector is a point receiver located at a fixed position x_d . As it moves, the submarine emits a signal $g(t)$. We want to choose the signal $f(t)$ emitted by the interferer to fool the detector. For simplicity, we assume that $f(0) = g(0) = 0$.

We model the propagation of sound in Ω using the linear wave equation [12, Section 5.5],

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u + q(x, t), \quad (1)$$

where u is the acoustic pressure, c is the speed of sound in water, and ∇^2 is the Laplacian. The forcing term $q(x, t)$ contains the signals generated by the submarine and the interferer, which we model as point sources, taking q to have the form

$$q(x, t) = f(t)\delta_\varepsilon(x - x_i(t)) + g(t)\delta_\varepsilon(x - x_s(t)),$$

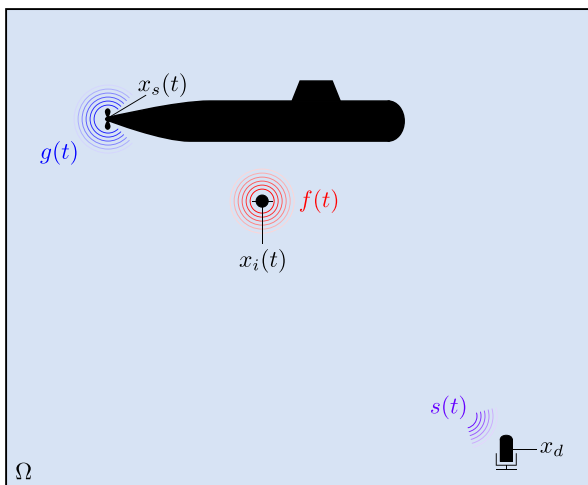


Fig. 1 Problem setup. The submarine at $x_s(t)$ emits an acoustic signal $g(t)$. The interferer at $x_i(t)$ emits a perturbing signal $f(t)$. The detector at x_d receives a signal $s(t)$, the result of f and g propagating through Ω as waves

where ε is a small parameter and

$$\delta_\varepsilon(x) = \frac{\varepsilon^2}{\pi^2(x_1^2 + \varepsilon^2)(x_2^2 + \varepsilon^2)}$$

approximates the Dirac delta [13, Section 2.5]. For simplicity, we assume that Ω is initially devoid of sound, yielding the zero initial conditions

$$u(x, 0) = 0 \quad \text{and} \quad \frac{\partial u}{\partial t}(x, 0) = 0. \quad (2)$$

Additionally, we take Ω to be a region of open water away from the seafloor and surface so that the effects of these boundaries may be neglected. To model the propagation of waves out through the boundary $\partial\Omega$ of Ω , we use a first-order approximation to an absorbing boundary condition [14] [15, Section 6.1.2],

$$\frac{\partial u}{\partial t} + c\nabla u \cdot \hat{n} = 0, \quad x \in \partial\Omega, \quad (3)$$

where \hat{n} denotes the outward-pointing unit normal vector to $\partial\Omega$.

The detector computes a spectrogram $\hat{s} = 10 \log_{10} |\mathcal{F}s|^2$ in decibels (dB), of the signal

$$s(t) = \int_{\Omega} u(x, t) \delta_\varepsilon(x - x_d) dV$$

that it receives, where \mathcal{F} is the short-time Fourier transform operator and $|\cdot|$ denotes the elementwise modulus of a complex function. The detector then passes \hat{s} through a neural network Φ that has been trained to recognize spectrograms produced by the submarine when the interferer is absent. Let L denote the loss function used to train the network; this is a function that takes $\Phi(\hat{s})$ as one argument, together with a label y that assumes the value 1 (respectively, 0) when the submarine is present (respectively, absent). To fool the detector, we want to select f to make $L(\Phi(\hat{s}), y)$ as large as possible, and to model the requirement that f be emittable by a real speaker, we constrain its frequency content to lie in a predetermined band. That is, we demand

$$\mathcal{P}\mathcal{F}f = 0,$$

where \mathcal{P} projects onto the frequencies that are disallowed.

We have thus arrived at the following optimization problem for the perturbing signal f :

$$\begin{aligned}
& \text{maximize}_{f,u,s,\hat{s}} J(f) = L(\Phi(\hat{s}), y) \\
& \text{subject to} \quad \frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u + f(t) \delta_\varepsilon(x - x_i(t)) + g(t) \delta_\varepsilon(x - x_s(t)) \\
& \quad u(x, 0) = \frac{\partial u}{\partial t}(x, 0) = 0 \\
& \quad \frac{\partial u}{\partial t} + c \nabla u \cdot \hat{n} = 0, \quad x \in \partial\Omega \\
& \quad s(t) = \int_\Omega u(x, t) \delta_\varepsilon(x - x_d) dV \\
& \quad \hat{s} = 10 \log_{10} |\mathcal{F}s|^2 \\
& \quad \mathcal{P}\mathcal{F}f = 0.
\end{aligned} \tag{4}$$

What makes this optimization problem different from those usually encountered in the literature on adversarial perturbations is the nature of the constraint. In our scenario, the perturbing signal cannot interact with the input of the neural network directly; the interaction is mediated by the physics of the surrounding environment as described by the PDE (1).

3 Discretization

To discretize (1) in space, we use a continuous Galerkin finite element method [16]. Applying standard manipulations, we arrive at the variational formulation

$$\frac{d^2}{dt^2} \int_\Omega uv dV = -c^2 \int_\Omega \nabla u \cdot \nabla v dV - c \frac{d}{dt} \int_{\partial\Omega} uv dS + \int_\Omega qv dV$$

for (1) with the boundary condition (3), where v is a test function. We partition Ω into a mesh of conforming, nonoverlapping triangular elements and approximate u and q on the mesh by continuous piecewise linear interpolants over the elements. At each time t , the interpolants are determined by the values of $u(x, t)$ and $q(x, t)$ at the mesh nodes (a P_1 Lagrange representation), which we gather into column vectors $\mathbf{u}(t)$ and $\mathbf{q}(t)$. Imposing the Galerkin condition yields the system of ODEs

$$\mathbf{M}\mathbf{u}'' = -c^2 \mathbf{K}\mathbf{u} - c\mathbf{S}\mathbf{u}' + \mathbf{M}\mathbf{q} \tag{5}$$

for \mathbf{u} , where \mathbf{M} , \mathbf{S} , and \mathbf{K} are the standard finite element mass, surface mass, and stiffness matrices.

We discretize (5) using a leapfrog scheme [17, Section 5.3]. For a single ODE $u'' = f(u, t)$, the update equation for the scheme is

$$\frac{u^{k+1} - 2u^k + u^{k-1}}{(\Delta t)^2} = f(u^k, t_k), \tag{6}$$

where Δt is a chosen time step and u^k is the approximation to the solution at time $t_k = k(\Delta t)$, $k = 0, \dots, K$. This is an explicit scheme, second-order accurate, and is nondissipative, making it well-suited to undamped wave propagation problems,

where δ_i^k and δ_s^k contain the values of $\delta_\varepsilon(x - x_i(t_k))$ and $\delta_\varepsilon(x - x_s(t_k))$ at the mesh nodes, respectively, and define the matrices

$$\mathbf{B} = (\Delta t)^2 \begin{bmatrix} 0 & & & & & \\ & 0 & & & & \\ & & \mathbf{M}\delta_i^2 & & & \\ & & & \mathbf{M}\delta_i^3 & & \\ & & & & \ddots & \\ & & & & & \mathbf{M}\delta_i^{K-1} \end{bmatrix}, \quad \mathbf{C} = (\Delta t)^2 \begin{bmatrix} 0 & & & & & \\ & 0 & & & & \\ & & \mathbf{M}\delta_s^2 & & & \\ & & & \mathbf{M}\delta_s^3 & & \\ & & & & \ddots & \\ & & & & & \mathbf{M}\delta_s^{K-1} \end{bmatrix}.$$

Gathering the values $f(t_0), \dots, f(t_K)$ and $g(t_0), \dots, g(t_K)$ into column vectors \mathbf{f} and \mathbf{g} , respectively, we combine the equations (7) for $k = 1, \dots, K - 1$ into a single block matrix equation,

$$\mathbf{A}\mathbf{u} = \mathbf{B}\mathbf{f} + \mathbf{C}\mathbf{g}.$$

We have thus completely discretized the PDE constraint in (4). Discretization of the remaining variables and constraints is more straightforward. First, we introduce a row vector \mathbf{d}^T that is a finite element discretization of the “detector” functional

$$d(u) = \int_{\Omega} u(x)\delta_\varepsilon(x - x_d) dV.$$

Then, defining

$$\mathbf{D}^T = \left[\begin{array}{c} \mathbf{d}^T \\ \mathbf{d}^T \\ \vdots \\ \mathbf{d}^T \end{array} \right] \Bigg\} K + 1 \text{ copies},$$

we have that $\mathbf{s} = \mathbf{D}^T \mathbf{u}$ contains the acoustic pressure observed by the detector at each time step. If \mathbf{F} is the matrix representing the short-time Fourier transform (see Sect. 4.3 below), then the spectrogram of the signal (in dB) at the detector is $\hat{\mathbf{s}} = 10 \log_{10} |\mathbf{F}\mathbf{s}|^2$. Finally, let \mathbf{P} be a matrix that selects entries of a spectrogram corresponding to frequencies outside the band of allowable frequencies for \mathbf{f} .

Assembling all of these objects, we are led to the following discrete version of (4):

$$\begin{aligned} & \text{maximize}_{\mathbf{f}, \mathbf{u}, \mathbf{s}, \hat{\mathbf{s}}} J(\mathbf{f}) = L(\Phi(\hat{\mathbf{s}}), y) \\ & \text{subject to} \quad \mathbf{A}\mathbf{u} = \mathbf{B}\mathbf{f} + \mathbf{C}\mathbf{g} \\ & \quad \mathbf{s} = \mathbf{D}^T \mathbf{u} \\ & \quad \hat{\mathbf{s}} = 10 \log_{10} |\mathbf{F}\mathbf{s}|^2 \\ & \quad \mathbf{P}\mathbf{f} = 0. \end{aligned} \tag{8}$$

4 Computation

Our goal is to compute local maximizers of (8). To do this, we use a first-order descent method with a projection to enforce the constraint $\mathbf{P}\mathbf{f}\mathbf{f} = 0$ [18, Section 2.D]. The primary computational difficulty is in the evaluations of the objective J and its gradient, both of which appear to require solution of the constraining PDE. In this section, we discuss how to perform these evaluations and the projection step efficiently.

4.1 Efficient evaluation of the objective

Naively, it seems that each evaluation of $J(\mathbf{f})$ requires a solution of the constraining PDE. Having selected \mathbf{f} , one first solves $\mathbf{A}\mathbf{u} = \mathbf{B}\mathbf{f} + \mathbf{C}\mathbf{g}$ for \mathbf{u} using the iteration (7). Then, one evaluates $\mathbf{s} = \mathbf{D}^T\mathbf{u}$, then $\hat{\mathbf{s}} = 10 \log_{10} |\mathbf{F}\mathbf{s}|^2$, and finally $J(\mathbf{f}) = L(\Phi(\hat{\mathbf{s}}), y)$. The solve for \mathbf{u} is by far the most expensive part of this computation, as (7) entails $K - 1$ solves with the large matrix \mathbf{A}^+ , which has dimension equal to the number of nodes in the finite element mesh. One can reduce the cost by computing and storing a factorization of \mathbf{A}^+ , but the cost typically remains significant, especially if the mesh is very fine or the number of time steps is very large.

Remarkably, we can exploit linear algebra to eliminate the need to solve for \mathbf{u} entirely. The crucial observation is that we are interested not in \mathbf{u} itself but only in

$$\mathbf{s} = \mathbf{D}^T\mathbf{u} = \mathbf{D}^T\mathbf{A}^{-1}(\mathbf{B}\mathbf{f} + \mathbf{C}\mathbf{g})$$

and that there are two ways to evaluate this product. One way evaluates $\mathbf{A}^{-1}(\mathbf{B}\mathbf{f} + \mathbf{C}\mathbf{g})$ first and then multiplies the result by \mathbf{D}^T on the left; this corresponds to the naive approach just described and requires a solve with \mathbf{A} (via (7)) for each value of \mathbf{f} . The other way evaluates $\mathbf{Y}^T = \mathbf{D}^T\mathbf{A}^{-1}$ by solving the adjoint equation

$$\mathbf{A}^T\mathbf{Y} = \mathbf{D} \tag{9}$$

and then computes

$$\mathbf{s} = \mathbf{Y}^T(\mathbf{B}\mathbf{f} + \mathbf{C}\mathbf{g}). \tag{10}$$

This latter approach has a potential advantage: because \mathbf{Y} does not depend on \mathbf{f} , we can compute it (or even $\mathbf{Y}^T\mathbf{B}$ and $\mathbf{Y}^T\mathbf{C}$) once, offline, and then use it to find \mathbf{s} for as many values of \mathbf{f} as needed in our optimization algorithm.

For this advantage to be realized, the effort required to solve (9) must be less than that required to solve all of the systems that would arise under the naive approach. Since \mathbf{D} has $K + 1$ columns, it would seem that solving (9) requires $K + 1$ solves with \mathbf{A}^T and, hence, that the alternative approach would be advantageous only if the number of optimization steps exceeds $K + 1$, under the reasonable assumption that solves with \mathbf{A} and \mathbf{A}^T are of similar cost. In fact, the situation is considerably better than this: due to the structure of \mathbf{A} and \mathbf{D} , we can solve (9) for a cost comparable to that of just a *single* solve with \mathbf{A}^T .

In more detail, partitioning \mathbf{A} as

$$\mathbf{A} = \left[\begin{array}{c|c} \mathbf{I} & \\ \hline \mathbf{I} & \\ \hline \mathbf{A}^- & \mathbf{A}^0 & \mathbf{A}^+ \\ \mathbf{A}^- & \mathbf{A}^0 & \mathbf{A}^+ \\ \vdots & \vdots & \vdots \\ \mathbf{A}^- & \mathbf{A}^0 & \mathbf{A}^+ \end{array} \right]$$

and partitioning \mathbf{D} as

$$\mathbf{D} = \left[\begin{array}{c|c} \tilde{\mathbf{D}} & \\ \hline \tilde{\mathbf{D}} & \\ \hline & \mathbf{d} \\ & \mathbf{d} \\ & \vdots \\ & \mathbf{d} \end{array} \right],$$

we have that

$$\mathbf{Y} = \mathbf{A}^{-T} \mathbf{D} = \left[\begin{array}{c|c} \tilde{\mathbf{D}} & -\tilde{\mathbf{A}}^T \hat{\mathbf{A}}^{-T} \tilde{\mathbf{D}} \\ \hline & \hat{\mathbf{A}}^{-T} \hat{\mathbf{D}} \end{array} \right].$$

The first step in computing \mathbf{Y} is to solve $\hat{\mathbf{A}}^T \hat{\mathbf{Y}} = \hat{\mathbf{D}}$ for $\hat{\mathbf{Y}} = \hat{\mathbf{A}}^{-T} \hat{\mathbf{D}}$. The matrix $\hat{\mathbf{A}}^T$ is a block upper triangular, block Toeplitz matrix with invertible diagonal blocks. The set of such matrices (for a given partitioning) forms a group under matrix multiplication (see, e.g., Horn and Johnson [19, Section 0.9.7] for the non-block case); hence, $\hat{\mathbf{A}}^{-T}$ has the same structure. The matrix $\hat{\mathbf{D}}$ is also block upper triangular (in fact, block diagonal) and block Toeplitz. The diagonal blocks of $\hat{\mathbf{D}}$ are not invertible—indeed, they are not even square—but the partitioning of $\hat{\mathbf{D}}$ conforms to the partitioning of $\hat{\mathbf{A}}^T$ and, hence, to that of $\hat{\mathbf{A}}^{-T}$. It follows that $\hat{\mathbf{Y}}$, too, is block upper triangular and block Toeplitz, being a product of two conformally-partitioned matrices with that structure:

$$\hat{\mathbf{Y}} = \begin{bmatrix} \hat{\mathbf{y}}_1 & \hat{\mathbf{y}}_2 & \hat{\mathbf{y}}_3 & \cdots & \hat{\mathbf{y}}_{K-1} \\ & \hat{\mathbf{y}}_1 & \hat{\mathbf{y}}_2 & \cdots & \hat{\mathbf{y}}_{K-2} \\ & & \ddots & \ddots & \vdots \\ & & & \hat{\mathbf{y}}_1 & \hat{\mathbf{y}}_2 \\ & & & & \hat{\mathbf{y}}_1 \end{bmatrix}.$$

We can therefore find all of $\hat{\mathbf{Y}}$ by finding just its last (block) column, and that can be accomplished by solving just one system with $\hat{\mathbf{A}}^T$ that has the last column of $\hat{\mathbf{D}}$ as the right-hand side. With $\hat{\mathbf{Y}}$ in hand, the remaining entries of \mathbf{Y} can be found by computing $-\tilde{\mathbf{A}}^T \hat{\mathbf{Y}}$, which is relatively inexpensive.

This reduction in the cost of computing \mathbf{Y} due to the structure of \mathbf{A} and \mathbf{D} is more than enough to make the alternative approach to evaluating $J(\mathbf{f})$ competitive. By using it, we were able to speed up the computations presented in Sect. 5 dramatically compared to the naive approach.

Note that this technique hinges on the assumption that both the location of the detector and the material properties of the ocean (namely, the wave speed) do not vary with time, as it is this time stationarity that results in the block Toeplitz structure of \mathbf{D} and $\hat{\mathbf{A}}$. Without this, the formulation (8) still works, but the \mathbf{d}^T , \mathbf{A}^- , \mathbf{A}^0 , and \mathbf{A}^+ matrices would need to be replaced with versions that depend on the index k of the time step. Material properties that vary merely spatially present no such obstacle.

4.2 Evaluation of gradients

To compute the gradient of J , we apply the chain rule: $\nabla J = (dJ/d\mathbf{f})^T$, where

$$\frac{dJ}{d\mathbf{f}} = \frac{\partial L}{\partial \Phi} \frac{d\Phi}{d\hat{\mathbf{s}}} \frac{d\hat{\mathbf{s}}}{ds} \frac{ds}{d\mathbf{u}} \frac{d\mathbf{u}}{d\mathbf{f}}.$$

Traditionally in PDE-constrained optimization, $dJ/d\mathbf{f}$ would be computed using the *adjoint method* [20, Section 2.3.3], sometimes called the *reduced gradient method* [21, Section 12.6]. In our context, this method first evaluates

$$\frac{dJ}{d\mathbf{u}} = \frac{\partial L}{\partial \Phi} \frac{d\Phi}{d\hat{\mathbf{s}}} \frac{d\hat{\mathbf{s}}}{ds} \frac{ds}{d\mathbf{u}}$$

using the solution \mathbf{u} to the PDE corresponding to the value of \mathbf{f} at which $\nabla J(\mathbf{f})$ is desired. Using the fact that $d\mathbf{u}/d\mathbf{f} = \mathbf{A}^{-1}\mathbf{B}$, it then evaluates

$$\frac{dJ}{d\mathbf{f}} = \frac{dJ}{d\mathbf{u}} \frac{d\mathbf{u}}{d\mathbf{f}} = \frac{dJ}{d\mathbf{u}} \mathbf{A}^{-1}\mathbf{B}$$

not by forming $\mathbf{A}^{-1}\mathbf{B}$, which is prohibitively expensive, and then multiplying on the left by $dJ/d\mathbf{u}$ but instead by evaluating $\boldsymbol{\lambda}^T = (dJ/d\mathbf{u})\mathbf{A}^{-1}$ and then computing $dJ/d\mathbf{f} = \boldsymbol{\lambda}^T\mathbf{B}$. The method derives its name from the fact that the equation that must be solved for $\boldsymbol{\lambda}$,

$$\mathbf{A}^T \boldsymbol{\lambda} = \left(\frac{dJ}{d\mathbf{u}} \right)^T,$$

involves the adjoint of \mathbf{A} . Thus, evaluating $\nabla J(\mathbf{f})$ this way requires two expensive PDE solves: one with \mathbf{A} (the original equation, to find \mathbf{u}) and one with \mathbf{A}^T (the adjoint equation, to find $\boldsymbol{\lambda}$).

By adopting the approach to evaluating J from Sect. 4.1, we can eliminate *both* of these solves, greatly reducing the cost of computing ∇J . For under that approach, by

(10), the matrix

$$\frac{d\mathbf{s}}{d\mathbf{u}} \frac{d\mathbf{u}}{d\mathbf{f}} = \frac{d\mathbf{s}}{d\mathbf{f}} = \mathbf{Y}^T \mathbf{B}$$

is already in hand, as \mathbf{Y} has been pre-computed by solving (9) offline. We evaluate $dJ/d\mathbf{f}$ via

$$\frac{dJ}{d\mathbf{f}} = \frac{\partial L}{\partial \Phi} \frac{d\Phi}{d\hat{\mathbf{s}}} \frac{d\hat{\mathbf{s}}}{ds} \mathbf{Y}^T \mathbf{B},$$

using an automatic differentiation package to evaluate $\partial L/\partial \Phi$, $d\Phi/d\hat{\mathbf{s}}$, and $d\hat{\mathbf{s}}/ds$, since they involve differentiating through the neural network Φ . As our experiments below show, this approach to evaluating ∇J is considerably more efficient than the adjoint method.

4.3 Computation of spectrograms

We compute the spectrogram $\hat{\mathbf{s}}$ of the discrete signal \mathbf{s} in the usual way via the short-time Fourier transform (STFT) [22, Section 10.3]. We select L frequencies $\omega_0, \dots, \omega_{L-1}$, divide \mathbf{s} into M overlapping time windows of length W , and evaluate

$$\hat{s}_{\ell,m} = 10 \log_{10} \left| \sum_{k=0}^{W-1} w(k) s_{mH+k} e^{-i \frac{2\pi k}{W} \omega_{\ell}} \right|^2$$

for $0 \leq \ell \leq L - 1$ and $0 \leq m \leq M - 1$, where H is the distance between the starting indices of adjacent windows (assumed to be constant) and w is the von Hann windowing function,

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

The value of $\hat{s}_{\ell,m}$ represents the amount of power present in \mathbf{s} at the ℓ th frequency during the m th time window, expressed in dB.

The STFT is a linear operation and therefore can be represented by a matrix \mathbf{F} . If we view the matrix $\hat{\mathbf{s}}$ as a vector by stacking its columns, then \mathbf{F} has dimensions $LM \times K$ and for each value of m has a copy of the $L \times W$ matrix

$$\mathbf{Q} = \begin{bmatrix} w(0)e^{-i \frac{2\pi 0}{W} \omega_0} & \dots & w(W - 1)e^{-i \frac{2\pi (W-1)}{W} \omega_0} \\ \vdots & & \vdots \\ w(0)e^{-i \frac{2\pi 0}{W} \omega_{L-1}} & \dots & w(W - 1)e^{-i \frac{2\pi (W-1)}{W} \omega_{L-1}} \end{bmatrix}$$

occupying rows mL through $(m + 1)L - 1$ and columns mH through $mH + W - 1$. Elsewhere, the entries of \mathbf{F} are zero.

4.4 Enforcing the frequency constraint

To restrict the frequency content in \mathbf{f} , we identify the frequencies among $\omega_0, \dots, \omega_L$ that we wish to disallow and construct the selection matrix \mathbf{P} so that $\tilde{\mathbf{F}} = \mathbf{P}\mathbf{F}$ contains exactly the rows of \mathbf{F} corresponding to those frequencies. (Each row of \mathbf{P} is a row of the $LM \times LM$ identity matrix.) The constraint $\mathbf{P}\mathbf{F}\mathbf{f} = 0$ then says that \mathbf{f} must belong to the nullspace $\mathcal{N}(\tilde{\mathbf{F}})$ of $\tilde{\mathbf{F}}$.

To enforce this constraint during the optimization procedure, we project the current search direction onto $\mathcal{N}(\tilde{\mathbf{F}})$. If \mathbf{f}_v denotes the guess for the optimal value of \mathbf{f} at the v th optimization step, the update equation is

$$\mathbf{f}_{v+1} = \mathbf{f}_v + \alpha \mathbf{P}_{\mathcal{N}(\tilde{\mathbf{F}})} \nabla J(\mathbf{f}_v),$$

where $\mathbf{P}_{\mathcal{N}(\tilde{\mathbf{F}})}$ is the orthogonal projector onto $\mathcal{N}(\tilde{\mathbf{F}})$ and α is the step size. We find $\mathbf{P}_{\mathcal{N}(\tilde{\mathbf{F}})}$ by noting that since \mathbf{f} is real, the constraint $\tilde{\mathbf{F}}\mathbf{f} = 0$ is equivalent to

$$\begin{bmatrix} \Re \tilde{\mathbf{F}} \\ \Im \tilde{\mathbf{F}} \end{bmatrix} \mathbf{f} = 0. \quad (11)$$

Hence, $\mathbf{P}_{\mathcal{N}(\tilde{\mathbf{F}})}$ can be formed by taking the singular value decomposition of the matrix on the left-hand side of (11) and extracting the right singular vectors corresponding to zero singular values.

5 Experimental results

We now conduct several experiments that illustrate the effectiveness of our proposed approach for computing adversarially perturbing signals under a physical access assumption.

5.1 Basic setup

We take Ω to be the rectangle $(0, 100) \times (0, 100)$, with the lengths denominated in meters. We set $c = 1525 \text{ m s}^{-1}$, which is a reasonable estimate of the speed of sound in seawater [23]. For simplicity, we fix the locations of the submarine and the interferer in time, setting $x_s(t) = (5, 75)$ and $x_i(t) = (9.75, 68.75)$ for all t . We position the detector at $x_d = (52.5, 12.5)$. (Recall that while our framework can accommodate settings in which each of these three agents is mobile, the method described in Sect. 4.1 for accelerating the evaluation of the objective requires the location of the detector to be static.) We take \mathbf{P} to be a bandpass filter for frequencies between 100 Hz and 6000 Hz.

We use the FEniCS software package to carry out our finite element computations [24, 25]. Our mesh for Ω consists of 23,270 unstructured (but nearly uniformly sized) triangular elements and has 11,836 nodes. We integrate (1) to a final time of 1.0 s using 6000 time steps, yielding a step size of $\Delta t \approx 1.6710^{-4}$ s. Our decision to use 6000 time

steps is motivated by typical block size selections in audio signal processing Lerch [26]. The diameter of the smallest cell in the mesh is approximately 0.82 m; hence, the Courant number for our simulations is $C \approx 1525 \cdot (1.67 \times 10^{-4})/0.82 \approx 0.31$. For the STFT, we use $L = 129$ uniformly-spaced frequencies between 0 Hz and 3000 Hz and a window length $W = 64$, which yields $M = \lceil 6000/64 \rceil = 94$ time windows. In the context of (8), these parameters yield decision variables of the following dimensions: $\mathbf{u} \in \mathbb{R}^{71,016,000}$, $\mathbf{f} \in \mathbb{R}^{6000}$, $\hat{\mathbf{s}} \in \mathbb{R}^{6000}$, and $\hat{\mathbf{s}} \in \mathbb{R}^{129 \times 94}$.

5.2 Spectrogram classifiers

For the detector's classifier Φ , we consider three neural network architectures from the computer vision literature: VGG-19 [27], Inception V3 [28], and GoogLeNet [29]. These models are pre-trained for image classification on the 1.2 million training images and 1000 classes from the ImageNet Large Scale Visual Recognition Challenge [30]. As described in Sect. 1, we repurpose these models for audio classification by converting audio signals to spectrograms. To customize the models for spectrogram-based signal detection, we use the standard technique of *transfer learning* [31]: we modify the dimensions of and retrain the networks' final layers specifically for this task. Transfer learning makes creating custom machine learning models accessible, as a modeler can build models for new tasks without requiring enormous computational resources or data sets. This accessibility makes models trained with transfer learning quite prevalent.

We train the classifiers to discriminate between spectrograms generated by pure sine wave signals $g(t) = \sin(\omega t)$ emitted by potential intruders, where the frequency ω is allowed to assume one of the 80 values 10, 20, ..., 800 Hz. Spectrograms corresponding to signals $g(t)$ with $\omega \leq 400$ Hz are labeled malicious (i.e., as coming from our intruding submarine); those corresponding to higher frequencies are labeled benign. To add realism, we assume that the signals received by the detector have been corrupted with noise: the detector receives the signal $s(t)$ that results from the signal $g(t)$ propagating through Ω , plus band-limited noise of the form [32, Section 14.9.3]

$$\eta(t) = \sum_{\ell=0}^{L-1} A_{\ell} \cos(\omega_{\ell} t) + B_{\ell} \sin(\omega_{\ell} t),$$

where A_{ℓ} and B_{ℓ} are independently drawn from a normal distribution with zero mean and variance σ_{ℓ}^2 , and σ_{ℓ} is proportional to the root-mean-square value of the component of $s(t)$ (the *noiseless* received signal) belonging to the STFT frequency band ω_{ℓ} . This way, the average power of the noise in each band decreases at higher frequencies, matching the physical reality that higher-frequency signals experience greater attenuation.

To generate training data, for each of the 80 allowable values of ω , we compute the signal $s(t)$ received by the detector in the absence of noise (and the interferer) by solving the wave propagation problem. We then generate 40 noisy spectrograms for each frequency by adding $A_{\ell} + B_{\ell}i$ to each entry of the STFT of the corresponding $s(t)$, with A_{ℓ} and B_{ℓ} distributed as just described. The result is a balanced training set

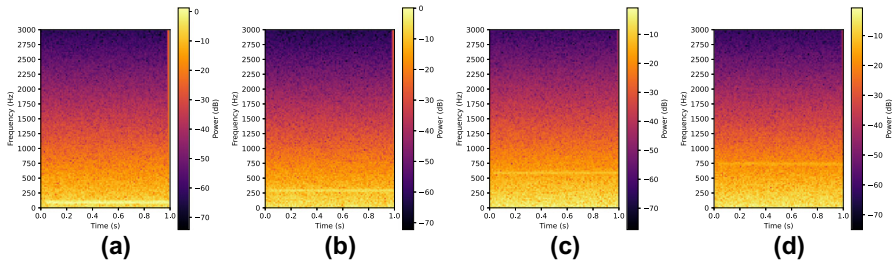


Fig. 2 Sample spectrograms from the data used to train the neural networks Φ used by the detector. **a** 100 Hz signal. **b** 300 Hz signal. **c** 600 Hz signal. **d** 750 Hz signal. Spectrograms containing signals with frequency less than or equal to 400 Hz are labeled as containing a malicious intruder. Otherwise the spectrograms are labeled benign

Table 1 Classification accuracy of each network on the testing set

Network	Accuracy without interferer (%)	Accuracy with interferer (%)
Inception V3	97.75	0.000
GoogLeNet	97.50	0.000
VGG-19	95.88	0.125

of 3200 observations, examples of which are displayed in Fig. 2. We also generate a separate validation data set for monitoring progress during training and a testing data set for assessing our method for computing perturbing signals, each consisting of 800 spectrograms, 10 for each value of ω . Finally, because the detector's classification problem is a binary decision over two classes—malicious or benign—we use a binary cross-entropy loss both to train the classifiers and as the function L in the optimization problem (4).

With the aid of the PyTorch machine learning framework [33], we train the neural networks using stochastic gradient descent, terminating the training when the loss on the validation set stops improving. The first column of Table 1 shows the accuracy of each model on the spectrograms in the testing set. Absent the interferer, all the models are excellent classifiers, attaining accuracy rates better than 95%.

5.3 Adversarial perturbations

We now test our proposed method for constructing perturbing signals $f(t)$ to be emitted by the interferer in order to fool the classifiers. For each classifier, we take each spectrogram from the testing set that it classified correctly and solve the discretized optimization problem (8) in search of a perturbing signal that causes the classifier to misclassify the observation, ideally with high confidence. Any first-order method for unconstrained problems can be used to solve (8); we use the limited-memory BFGS (L-BFGS) method as implemented by Zhu et al. [34], computing the objective and gradient as described in Sects. 4.1 and 4.2 and applying the projected gradient procedure described in Sect. 4.4. We start the algorithm with a zero initial guess and use

the automatic differentiation capabilities of PyTorch to compute derivatives involving Φ .

Among perturbing signals which succeed in inducing misclassification, small signals are preferable to large ones because they require less energy to emit and are less likely to be detected by a secondary classifier (e.g., a human observer). In the literature on adversarial perturbations, the notion of a “small” signal has been formalized using different metrics, including ℓ^2 [35], ℓ^0 [36], ℓ^∞ [37], and even some bespoke options [38]. Nevertheless, there is no consensus as to which is best. We therefore enforce the requirement that the perturbing signals be small heuristically. We limit L-BFGS to at most 100 iterations in total and enforce early stopping by checking every 10 iterations to see if misclassification been achieved. If so, we terminate the algorithm.

Our results, summarized in the second column of Table 1, demonstrate that misclassification can be achieved consistently. For both Inception V3 and GoogLeNet, we are able to induce misclassification of *every single spectrogram* that they originally classified correctly. For VGG-19, we are able to induce misclassification in all but one of the 800 observations in the testing set. Moreover, empirically, we find that many of the interfering signals have much smaller amplitude than the signals emitted by the submarine/intruder. In many cases, the interfering signals are visually indistinguishable from the background noise.

Figure 3 provides a typical example of this behavior. The figure displays both the original spectrogram and the spectrogram with the perturbing signal for a 260 Hz intruder signal. Inception V3 correctly classifies the original spectrogram as malicious with 99.0% confidence. The interfering signal results in only minor changes to the spectrogram, but the classifier’s confidence that the signal is malicious drops to 0.01%.

Figure 3c depicts the waveform $f(t)$ emitted by the interferer for this example, which has an amplitude of approximately 0.1. In these experiments, we always take the intruder signal to have unit amplitude. Thus, the interfering signal is about 10 times weaker than the intruder signal in this case, with the result that it blends easily into the background noise.

Figure 4 displays a similar example for a 70 Hz signal initially classified correctly by GoogLeNet. With the interfering signal—which is nearly 20 times weaker than the original signal—in place, the detector’s confidence that the signal is malicious drops from 95.7 to 0.01%.

5.4 Universality of computed perturbations

One limitation of our approach to constructing adversarial perturbations is that it depends on knowledge of the architecture of the neural network Φ . In many applications, Φ may be unknown or unavailable. It is thus of practical interest to understand how well the adversarial perturbations we construct generalize across neural networks. Here, we investigate this briefly by taking the perturbations computed in Sect. 5.3 and using them with neural networks other than the one used to construct them.

The results of this experiment are summarized in Table 2. Perturbations constructed for VGG-19 were the most successful at inducing misclassification in the other models, achieving a roughly 20% misclassification rate when used with Inception V3

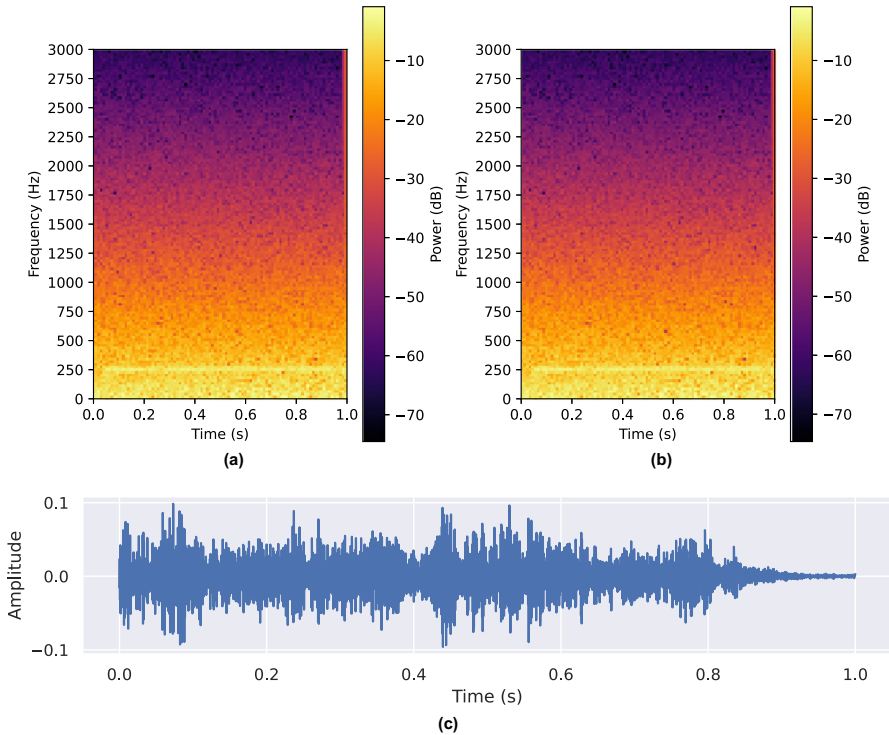


Fig. 3 The original (a) was correctly classified as malicious by Inception V3 with 99.0% confidence. When the interfering signal is added, the perturbed spectrogram (b) is misclassified by the same network as benign with 99.99% confidence. The waveform of the interfering signal is given in (c)

and GoogLeNet. Overall, the cross-model misclassification rate of these perturbations is not very high, which can be attributed to the fact that the perturbations were not constructed with cross-model generalization as a goal. Nevertheless, these misclassification rates are comparable to those obtained in similar experiments in the (non-PDE-constrained) adversarial perturbation literature [35].

The problem of constructing generalizable—more frequently termed “universal”—perturbations under logical access assumptions, is well-studied; see Zhang et al. [39] for a survey. By combining techniques for this with our approach, it may be possible to construct universal perturbations subject to physical access constraints. For instance, the algorithm of Moosavi-Dezfooli et al. [40, Algorithm 1] for constructing universal perturbations requires only the ability to repeatedly construct a perturbation for single observation and known neural network. By using the method described in Sects. 3 and 4 for this step, we can generate candidates for universal perturbations for (4).

5.5 Computational efficiency

We close with some remarks on the effectiveness of the technique described in Sect. 4.1 at reducing the time required to complete the above computations. Table 3 shows

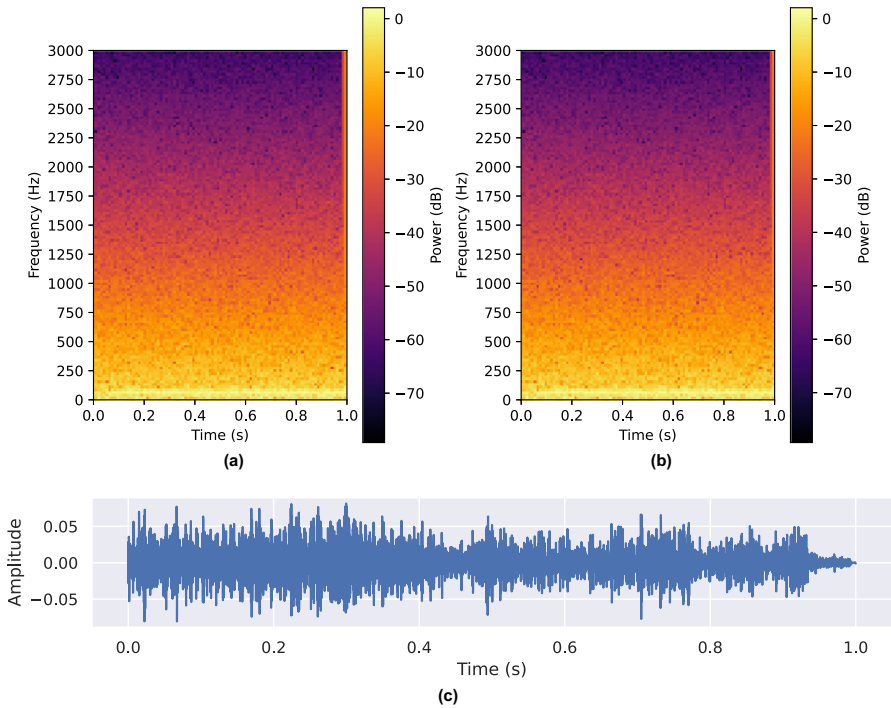


Fig. 4 The original (a) was classified as malicious by GoogLeNet with 95.7% confidence. When the interfering signal is added, the perturbed spectrogram (b) is classified by the same network as benign with 99.99% confidence. Though it is difficult to find visible differences between these spectrograms, they do exist. One example can be found by looking closely in the region corresponding to 0.4 s and 1150 Hz. The waveform of the interfering signal is given in (c)

Table 2 Percent of testing set for which a perturbation constructed for one model induces misclassification in a (potentially different) target model

		Target model		
		Inception V3 (%)	GoogLeNet (%)	VGG-19 (%)
Construction model	Inception V3	100	14.0	7.8
	GoogLeNet	18.0	100	15.3
	VGG-19	18.4	21.1	99.9

the times (averaged over 800 runs) required to evaluate the objective function and its gradient for each network architecture using both the naive approach (solving the PDE once for each objective evaluation, plus an adjoint solve for each gradient evaluation) and the method of Sect. 4.1. These times were computed using a single node of the “hamming” cluster at the High Performance Computing Center of the Naval Postgraduate School; this node is equipped with dual 8-core AMD EPYC 7F32 CPUs clocked at 3.7 GHz and 512 GB of RAM. Note that in our implementation, the gradient cannot be computed independently of the objective; hence, the times listed for

Table 3 Times for evaluating the objective function and its gradient for each network using the naive/adjoint approaches and the method of Sects. 4.1, 4.2. The speedup factor gained by using the latter is also listed

Computation	Time (s)		
	Inception V3	GoogLeNet	VGG-19
Objective (Naive)	6.8×10^0	6.7×10^0	6.7×10^0
Objective (Sect. 4.1)	7.0×10^{-2}	4.9×10^{-2}	1.0×10^{-1}
Speedup	97.1	136.7	67.0
Gradient (Adjoint method)	1.3×10^1	1.3×10^1	1.3×10^1
Gradient (Sect. 4.2)	1.5×10^{-1}	1.1×10^{-1}	1.4×10^{-1}
Speedup	86.7	118.2	92.9

the gradient computations include the time required to perform an objective evaluation too. As the table makes clear, the method of Sect. 4.1 is substantially faster than the naive approach by a factor ranging from 60x to 140x.

It is difficult to overstate the significance of these gains. The PDE problem we have considered in these proof-of-concept experiments is relatively simple: the equation is linear and two-dimensional in space; the spatial discretization is low-order and fairly coarse (only 11,000 degrees of freedom); the temporal discretization is low-order and explicit; and the horizon for integration in time is short (only 6,000 time steps). It takes only a handful of seconds to solve. Nevertheless, during optimization, the objective and its gradient may be evaluated dozens of times, and those seconds quickly add up to minutes—or even hours—to compute just one perturbing signal if the evaluations are done the naive way. Using the approach of Sect. 4.1, we must solve the PDE (or, rather, its adjoint) only *once* to compute \mathbf{Y} . Further evaluations of the objective and gradient then require only cheap matrix–vector multiplies instead of an expensive solve. As a result, we are able to compute all approximately 2400 perturbing signals in Sect. 5.3 with just a few hours of computing time.

With a more physically realistic scenario, each PDE solve may require several hours of time using specialized high-performance computing hardware. In such a case the gains realized by using the method of Sect. 4.1 would be even more substantial.

6 Conclusion

We have demonstrated that adversarial perturbations can be used to fool neural network classifiers when the adversary is limited to having only physical access to the classifiers' inputs. By considering an example from underwater acoustics, we have shown how to formulate the problem of computing such perturbations as a PDE-constrained optimization problem and described how to solve that problem efficiently.

In addition to being a novel application of PDE-constrained optimization to machine learning, our results have implications for the real-world use of neural networks. Neural networks' lack of robustness, as embodied by their extreme sensitivity to small changes in their inputs, leaves them ill-suited for settings in which those inputs can be

manipulated by a motivated, sophisticated adversary and in which misclassification has severe consequences. Many applications in security and defense fit this description. By demonstrating that indirect manipulation of a network's inputs via a physical environment suffices to elicit these shortcomings, our work broadens the range of threats that designers of, e.g., autonomous sensors should consider when choosing statistical and machine learning models for use in their systems.

This work admits many potentially interesting extensions. For instance, by replacing the wave equation with a different PDE (e.g., the Maxwell equations), one can study the adversarial manipulation of signals whose propagation is governed by physics other than those of acoustics (e.g., the physics of electromagnetics). Another promising direction, highlighted in Sect. 5.4, is the generation of so-called “universal perturbations” which can be applied regardless of the background noise and target classifier's architecture. Finally, some recent works attempt to create classifiers which are robust to adversarial perturbations [2, 41]. Though these methods do not overcome the inherent ill-conditioning of deep neural networks, they can make adversarial perturbations more challenging to find [42, 43]. Another possible direction for future work is to investigate the performance of our contributions on classifiers which include these defenses.

Acknowledgements All authors acknowledge support from ONR grants N0001421WX00142 and N0001423WX01316. We thank the High Performance Computing Center at the Naval Postgraduate School for supplying the computing resources used for our experiments. The views expressed in this document are those of the authors and do not reflect the official policy or position of the Department of Defense, the Department of the Navy, the Office of Naval Research, or the U. S. Government.

Data availability Only original datasets were analyzed in this paper. The spectrograms used to train the models and validate our contributions can be reproduced as described in Sect. 5. The neural networks for the Inception V3, GoogLeNet, and VGG-19 architectures, all of which were pretrained on training data from the 2012 Imagenet Large Scale Visual Recognition Challenge (ILSVRC), can be downloaded from PyTorch Hub [44].

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: Proceedings of the International Conference on Learning Representations (2014). <https://doi.org/10.48550/arXiv.1312.6199>
2. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Proceedings of the International Conference on Learning Representations (2015). <https://doi.org/10.48550/arXiv.1412.6572>
3. Moosavi-Dezfooli, S.-M., Fawzi, A., Frossard, P.: DeepFool: A simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2574–2582 (2016). <https://doi.org/10.1109/CVPR.2016.282>
4. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: Proceedings of the IEEE Symposium on Security and Privacy (SP), pp. 39–57 (2017). <https://doi.org/10.1109/SP.2017.49>

5. Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.-J.: ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISeC), pp. 15–26 (2017). <https://doi.org/10.1145/3128572.3140448>
6. Bassett, R., Graves, M., Reilly, P.: Color and edge-aware adversarial image perturbations. arXiv (2020) <https://doi.org/10.48550/arXiv.2008.12454>
7. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial examples in the physical world. In: Yampolskiy, R.V. (ed.) Artificial Intelligence Safety and Security, pp. 99–112. Chapman and Hall/CRC, New York (2018)
8. Sharif, M., Bhagavatula, S., Bauer, L., Reiter, M.K.: Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS), pp. 1528–1540 (2016). <https://doi.org/10.1145/2976749.2978392>
9. Cohen, L.: Time-Frequency Analysis. Prentice Hall, Upper Saddle River (1995)
10. Liu, X., Wang, X., Sahidullah, M., Patino, J., Delgado, H., Kinnunen, T., Todisco, M., Yamagishi, J., Evans, N., Nautsch, A., Lee, K.A.: ASVspoof 2021: Towards spoofed and deepfake speech detection in the wild. *IEEE/ACM Trans. Audio. Speech Language Process.* **31**, 2507–2522 (2023). <https://doi.org/10.1109/TASLP.2023.3285283>
11. Sharma, G., Umaphathy, K., Krishnan, S.: Trends in audio signal feature extraction methods. *Appl. Acoustics* **158**, 107020 (2020). <https://doi.org/10.1016/j.apacoust.2019.107020>
12. Kinsler, L.E., Frey, A.R., Coppens, A.B., Sanders, J.V.: Fundamentals of Acoustics. Wiley, New York (2000)
13. Gel'fand, I.M., Shilov, G.E.: Generalized Functions, Volume I: Properties and Operations. Academic Press, New York (1964)
14. Engquist, B., Majda, A.: Absorbing boundary conditions for the numerical simulation of waves. *Math. Comput.* **31**, 629–651 (1977). <https://doi.org/10.1090/s0025-5718-1977-0436612-4>
15. Cohen, G., Pernet, S.: Finite Element and Discontinuous Galerkin Methods for Transient Wave Equations. Springer, Dordrecht (2017)
16. Hughes, T.J.R.: The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. Dover, Mineola (2000)
17. LeVeque, R.J.: Finite Difference Methods for Ordinary and Partial Differential Equations. SIAM, Philadelphia (2007)
18. Royset, J.O., Wets, R.J.-B.: An Optimization Primer. Springer, Cham (2022)
19. Horn, R.A., Johnson, C.R.: Matrix Analysis, 2nd edn. Cambridge University Press, Cambridge, UK (2012)
20. Gunzburger, M.D.: Perspectives in Flow Control and Optimization. SIAM, Philadelphia (2003)
21. Luenberger, D.G., Ye, Y.: Linear and Nonlinear Programming. Springer, New York (2008)
22. Oppenheim, A.V., Schaffer, R.W., Buck, J.R.: Discrete-Time Signal Processing, 2nd edn. Prentice Hall, Upper Saddle River (1999)
23. Kuperman, W.A., Roux, P.: Underwater acoustics. In: Rossing, T.D. (ed.) Springer Handbook of Acoustics, pp. 149–204. Springer, New York (2007). Chap. 5. https://doi.org/10.1007/978-0-387-30425-0_5
24. Alnæs, M., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M.E., Wells, G.N.: The FEniCS Project Version 1.5. *Archive Numer. Softw.* **3**, 9–23 (2015). <https://doi.org/10.11588/ans.2015.100.20553>
25. Logg, A., Mardal, K.-A., Wells, G.N., et al.: Automated Solution of Differential Equations by the Finite Element Method. Springer, Berlin (2012)
26. Lerch, A.: An Introduction to Audio Content Analysis: Music Information Retrieval Tasks and Applications. John Wiley & Sons, Hoboken (2022)
27. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proceedings of the International Conference on Learning Representations (2015). <https://doi.org/10.48550/arXiv.1409.1556>
28. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the Inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826 (2016). <https://doi.org/10.1109/CVPR.2016.308>
29. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9 (2015). <https://doi.org/10.1109/CVPR.2015.7298594>

30. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large scale visual recognition challenge. *Int. J. Comput. Vision* **115**, 211–252 (2015). <https://doi.org/10.1007/s11263-015-0816-y>
31. Bengio, Y.: Deep learning of representations for unsupervised and transfer learning. In: Guyon, I., Dror, G., Lemaire, V., Taylor, G., Silver, D. (eds.) *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. *Proceedings of Machine Learning Research*, vol. 27, pp. 17–36 (2012). <https://proceedings.mlr.press/v27/bengio12a.html>
32. Hartmann, W.M.: Acoustic signal processing. In: Rossing, T.D. (ed.) *Springer Handbook of Acoustics*, pp. 503–530. Springer, New York (2007). Chap. 14. https://doi.org/10.1007/978-0-387-30425-0_14
33. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high-performance deep learning library. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 8026–8037 (2019). <https://pytorch.org>
34. Zhu, C., Byrd, R.H., Lu, P., Nocedal, J.: Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Software* **23**(4), 550–560 (1997). <https://doi.org/10.1145/279232.279236>
35. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint [arXiv:1312.6199](https://arxiv.org/abs/1312.6199) (2013)
36. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: *2017 IEEE Symposium on Security and Privacy (sp)*, pp. 39–57 (2017). IEEE
37. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint [arXiv:1412.6572](https://arxiv.org/abs/1412.6572) (2014)
38. Bassett, R., Graves, M., Reilly, P.: Color and edge-aware adversarial image perturbations. arXiv preprint [arXiv:2008.12454](https://arxiv.org/abs/2008.12454) (2020)
39. Zhang, C., Benz, P., Lin, C., Karjauv, A., Wu, J., Kweon, I.S.: A survey on universal adversarial attack. arXiv preprint [arXiv:2103.01498](https://arxiv.org/abs/2103.01498) (2021)
40. Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 86–94 (2017). <https://doi.org/10.1109/CVPR.2017.17>
41. Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., Madry, A.: Adversarial examples are not bugs, they are features. *Adv. Neural Inf. Process. Syst.* **32** (2019)
42. Garaev, R., Rasheed, B., Khan, A.M.: Not so robust after all: Evaluating the robustness of deep neural networks to unseen adversarial attacks. *Algorithms* **17**(4), 162 (2024)
43. Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., McDaniel, P.: Ensemble adversarial training: Attacks and defenses. arXiv preprint [arXiv:1705.07204](https://arxiv.org/abs/1705.07204) (2017)
44. PyTorch Development Team: PyTorch Hub. <https://pytorch.org/hub/>. Accessed: 2022-06-08 (2022)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.