A Geometric Perspective on the Transferability of Adversarial Directions

Zachary Charles University of Wisconsin–Madison
Harrison Rosenberg University of Wisconsin–Madison
Dimitris Papailiopoulos University of Wisconsin–Madison

Abstract

State-of-the-art machine learning models frequently misclassify inputs that have been perturbed in an adversarial manner. Adversarial perturbations generated for a given input and a specific classifier often seem to be effective on other inputs and even different classifiers. In other words, adversarial perturbations seem to transfer between different inputs, models, and even different neural network architectures. In this work, we show that in the context of linear classifiers and two-layer ReLU networks, there provably exist directions that give rise to adversarial perturbations for many classifiers and data points simultaneously. We show that these "transferable adversarial directions" are guaranteed to exist for linear separators of a given set, and will exist with high probability for linear classifiers trained on independent sets drawn from the same distribution. We extend our results to large classes of two-layer ReLU networks. We further show that adversarial directions for ReLU networks transfer to linear classifiers while the reverse need not hold, suggesting that adversarial perturbations for more complex models are more likely to transfer to other classifiers. We validate our findings empirically, even for deeper ReLU networks.

1 Introduction

Many popular machine learning models, including deep neural networks, have been shown to be vulnerable to adversarial attacks (Szegedy et al., 2013; Goodfellow et al., 2014; Nguyen et al., 2015; Moosavi Dezfooli et al., 2016). Small adversarial perturbations of image data can cause the model to significantly misclassify the data, even though the perturbed image may seem unchanged to the human eye. These perturbations are highly structured, as neural networks have been shown to be robust to random perturbations (Liu et al., 2016; Fawzi et al., 2016). While there has been a significant amount of work on designing adversarial attacks (Grosse et al., 2016; Moosavi Dezfooli et al., 2016; Mopuri et al., 2017; Hendrik Metzen et al., 2017; Papernot et al., 2016) and defenses against these attacks (Madry et al., 2017; Sinha et al., 2017), theoretical properties of these adversarial perturbations are not fully understood. As shown by Athalye et al. (2018), state-of-the-art defenses are often beaten in short order by newly designed attacks. A better theoretical understanding of these adversarial examples could lead to a better understanding of why attack and defense strategies perform well in certain situations and badly in others.

One phenomenon that has been repeatedly observed in the literature is that adversarial perturbations often *transfer* to other data points. For instance, Moosavi-Dezfooli et al. (2017a) show that adversarial perturbations for a given input often work as an adversarial direction for many other inputs on the same neural networks. Such adversarial perturbations are often referred to as "universal" adversarial perturbations. Even worse, adversarial perturbations often transfer between classifiers (Papernot et al., 2016; Liu et al., 2016). This seems to hold even if the classifiers have different architectures Su et al. (2018) or are trained on different subsets of the training data (Szegedy et al., 2013).

This transferability has led to more effective adversarial attacks. Narodytska and Kasiviswanathan (2016) show that even if an adversary only has black-box access to a neural network, they can still fool it relatively often. Papernot et al. (2016) construct adversarial examples with only black-box access to a neural network by generating adversarial examples on a substitute network designed to emulate the first. Recent work has attempted to make machine learning systems more secure by "blocking" transferability (Hosseini et al., 2017),

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

to only limited success.

Unfortunately, adversarial perturbations are not yet fully understood, especially on a theoretical level. Moosavi-Dezfooli et al. (2017b) show that adversarial perturbations exist if certain geometric conditions hold, while Fawzi et al. (2018) study the robustness of linear and quadratic classifiers to adversarial examples. Tramèr et al. (2017) show empirically that subspaces corresponding to the span of adversarial examples often have large intersection, demonstrating the capacity of adversarial examples to transfer. Still, it is not fully understood why and which adversarial perturbations transfer between classifiers, even for relatively simple classifiers.

Our Contributions: In this work, we show that for linear classifiers and certain two-layer ReLU networks, classifiers trained on similar data sets can often be targeted by similar adversarial perturbations. Moreover, the geometry of the decision boundary causes these perturbations to be adversarial for most of the training data. We do this by analyzing directions which lead to adversarial perturbations, so-called "adversarial directions." While many adversarial directions do not transfer to other classifiers and data, we explicitly construct adversarial directions which will transfer with high probability.

In Section 3, we show that for linear classifiers, the max-margin SVM can be used to construct adversarial directions which transfer to other data points and to other linear classifiers. We also show that for all linear separators of a given dataset, there are universal adversarial perturbations (that is, perturbations which are adversarial for all points with a given label) whose norm depends only on the max-margin classifier. We extend our results to soft-margin linear classifiers trained on independent data sets, as well as multi-class linear classifiers. In Section 5, we consider adversarial perturbations for ReLU networks. We use a geometric analysis to show that there exist adversarial directions for certain two-layer ReLU networks which transfer to all other such ReLU networks and all the training data of a given class. In Section 6 we show that while these adversarial direction for ReLU networks transfer to linear classifiers, the reverse need not hold. In Section 7, we augment our theory with an empirical study showing that adversarial perturbations often transfer within a classifier, and even between distinct classifiers.

2 Overview

Notation: In the following we denote vectors in bold script, and functions, sets, and scalars in standard script. Matrices are denoted by capital letters. For a vector $\boldsymbol{w}$, w_i denotes its ith element.

Suppose we have some data space $\mathcal{X} \subseteq \mathbb{R}^d$ with label space $\mathcal{Y}$. For any set $S \subseteq \mathcal{X} \times \mathcal{Y}$ and $a, b \in \mathcal{Y}$, let $S[a] = \{(\boldsymbol{x}, y) \in S : y = a\}$ and $S[a, b] = \{(\boldsymbol{x}, y) \in S : y \in \{a, b\}\}$. A classifier is a function $h : \mathcal{X} \to \mathcal{Y}$. We say that h is S-accurate if for all $(\boldsymbol{x}, y) \in S$, $h(\boldsymbol{x}) = y$. We are interested in when we can perturb $\boldsymbol{x} \in \mathcal{X}$ to produce $\boldsymbol{x}'$ such that $h(\boldsymbol{x}') \neq h(\boldsymbol{x})$. This leads to the following definition.

Definition. *We say that $\boldsymbol{v}$ is an adversarial direction for h at $(\boldsymbol{x}, y)$ if $h(\boldsymbol{x}) = y$ and there is some $c > 0$ such that $h(\boldsymbol{x} + c\boldsymbol{v}) \neq y$.*

The vector $c\boldsymbol{v}$ is referred to as an *adversarial perturbation*, while the vector $\boldsymbol{x}' = \boldsymbol{x} + c\boldsymbol{v}$ is referred to as an *adversarial example*. We will show that for well-known classes of classifiers, there are directions that are adversarial for many classifiers on large subsets of $\mathcal{X}$. In other words, the adversarial direction is "transferable" both to other classifiers and to other data points.

The fact that adversarial directions transfer is not immediately obvious or even true for all models. For example, consider Figure 1. For any linear classifier, the most natural adversarial direction is the normal vector to the decision boundary. If we consider just the linear classifier h_1, this direction is given by $\boldsymbol{v}_1$. However, $\boldsymbol{v}_1$ is not an adversarial direction for h_2, as moving in that direction does not change any labels. Similarly, $\boldsymbol{v}_2$ is adversarial for h_2 but not for h_1. In short, not all adversarial directions transfer.

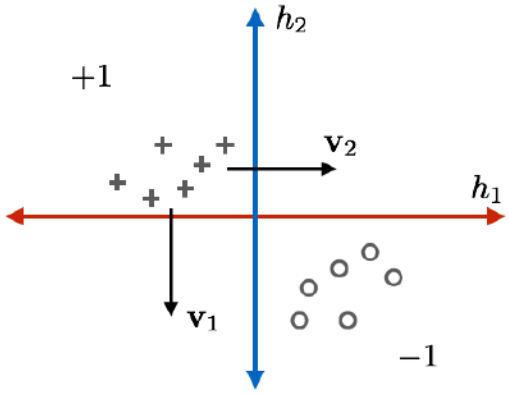


Figure 1: The decision boundary of two linear classifiers, h_1 (red) and h_2 (blue). The data with label $+1$ are marked by pluses, and those with label -1 are marked by circles. For the $+1$ data, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ give adversarial directions for h_1 and h_2, respectively.

We instead show that the *max-margin classifier* for a set S gives us an adversarial direction which transfers to all classifiers that linearly separate S, and extend this to the case where we train two linear classifiers h_1, h_2 on S_1, S_2 sampled independently from some distribution $\mathcal{D}$. We derive the following theorem.

Theorem 1. *Suppose linear classifiers h_1, h_2 are trained on sets $S^{(1)}, S^{(2)}$ of size n sampled independently from $\mathcal{D}$, and that each h_j correctly classifies $\Theta(n)$ of $S^{(j)}$ correctly. With high probability, there is a vector w and a set $S \subseteq S^{(1)} \cup S^{(2)}$ such that w is an adversarial direction for both h_1 and h_2, on S and $|S \cap S^{(j)}| = \Theta(n)$ for $j = 1, 2$.*

This is an informal version of Theorem 7. Intuitively, as long as we train linear classifiers on similar datasets, there is an adversarial direction which transfers between the linear classifiers and works for most of the data. This occurs even if h_1, h_2 are not the max-margin SVM on $S^{(1)}, S^{(2)}$. We derive a version of this for multi-class linear classifiers in Theorem 9.

For neural networks, the study of adversarial directions becomes more difficult. For a linear classifier, an adversarial direction for some data point is also adversarial for other data points with the same label. This need not hold for neural networks. Consider Figure 2. There are two nonlinear decision boundaries for classifiers h_1, h_2. Note that v_1 is adversarial for h_1 at x_1 and for the other points with $+1$ label. However, v_1 does not transfer to the $+1$ instances for h_2. Furthermore, v_2 is an adversarial direction for h_2 at x_2, but does not transfer to all other points with label -1 for h_2, nor does it transfer to any -1 labeled point for h_1.

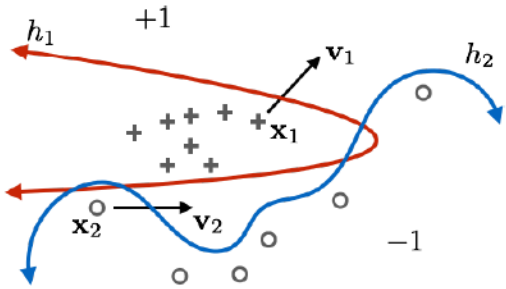


Figure 2: Two nonlinear decision boundaries, h_1 (red) and h_2 (blue). The instances with label $+1$ are marked by pluses, those with label -1 are marked by circles. An adversarial direction for h_1 at x_1 is denoted by v_1, while an adversarial direction for h_2 at x_2 is denoted by v_2.

With arbitrarily complicated decision boundaries, the study of adversarial directions becomes difficult. If we impose structure on our neural networks, we can begin to understand them. We show that for large classes of two-layer ReLU networks, there are transferable adversarial directions. The following theorem is an informal version of Theorem 15.

Theorem 2. *Fix a set S and consider the set of two-layer ReLU network with nonnegative weights on the final layer that correctly classify S. There is a vector v_S that is adversarial for all such networks at any point $(x, y) \in S$ such that $y = -1$.*

We also study whether adversarial directions transfer between linear classifiers and ReLUs. In Section 6 we show that while the adversarial direction for ReLU networks in Theorem 2 transfers to linear classifiers, the adversarial direction for linear classifiers in Theorem 1 does not necessarily transfer to ReLUs. In general, it seems that adversarial directions are more likely to transfer to a classifier with a less complicated decision boundary than with a more complicated one. We give supporting empirical evidence for this phenomenon in Section 7.

3 Linear Classifiers

Hard-margin Classifiers: We first focus on the case of hard-margin linear classifiers trained on the same training set S. By hard-margin, we mean that the classifier should correctly classify every point in S. We will show that the max-margin SVM classifier on S gives a transferable adversarial perturbation between the linear classifiers. We will relate the size of the perturbation needed to the support vector coefficients. We will use this as a stepping stone to consider both soft-margin linear classifiers and training data sampled from some underlying distribution.

Suppose we have data $\mathcal{X} \subseteq \mathbb{R}^d$ with labels $\{\pm 1\}$. An linear classifier h is given by $h(x) = \text{sign}(\langle w, x \rangle + b)$ for some $w \in \mathbb{R}^d, b \in \mathbb{R}$. Note that $h(x)$ is invariant under scaling (w, b) by the same positive constant, so we restrict to $\|w\|_2 = 1$. We let $\mathcal{H}$ denote the set of such linear classifiers.

Suppose we have some set S of labeled examples $\{(x, y)\}_{i=1}^N$ and that h is S-accurate. We will let $\mathcal{H}_S := \{h \in \mathcal{H} \mid h \text{ is } S\text{-accurate}\}$. The *margin* of $h \in \mathcal{H}_S$ is defined as $\gamma(h) = \min_{(x,y) \in S} y(\langle w, x \rangle + b)$.

We say that S is *linearly separable* if $\mathcal{H}_S \neq \emptyset$. In this case, standard theory shows that there is an SVM classifier $h^* \in \mathcal{H}_S$ with maximum margin given by $h^* = \text{sign}(\langle w^*, x \rangle + b^*)$. We will let γ^* denote its margin. We will use the following lemma about max-margin classifiers on S. This follows from Theorem 15.8 of Shalev-Shwartz and Ben-David (2014) and the discussion thereafter, combined with strong duality.

Lemma 3. *There are $\alpha_i \geq 0$ such that (1) $w^* = \sum_{i=1}^N \alpha_i y_i x_i$ and (2) $\sum_{i=1}^N \alpha_i y_i = 0$.*

We will use this to show that the max-margin classifier gives us a transferable adversarial direction.

Theorem 4. *For all $h \in \mathcal{H}_S$ and $(x, y) \in S$, $-yw^*$ is an adversarial direction for h at x.*

A slight modification of the proof shows the following.

Theorem 5. *Fix $h \in \mathcal{H}_S$ with margin γ_h. Fix (x, y) such that $y(\langle w, x \rangle + b) > 0$. Then $h(x - cyw^*) \neq h(x)$*

for all c satisfying

$$c > \frac{y(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b)}{\gamma_h \sum_{i=1}^{N} \alpha_i}.$$

Fix a linear classifier h. If we take $(\boldsymbol{x}, y)$ to be the point in S with minimum margin for h, this implies that as long as $c > (\sum_{i=1}^{N} \alpha_i)^{-1}$, we will have $h(\boldsymbol{x} - cy\boldsymbol{w}^*) \neq h(\boldsymbol{x})$. Thus, for this c, we are guaranteed that perturbing the data by $cy\boldsymbol{w}^*$ will cause h to misclassify some $\boldsymbol{x}$. This implies the following corollary.

Corollary 6. *For any $h \in \mathcal{H}_S$, there is some $(\boldsymbol{x}, y) \in S$ such that for all $c > (\sum_{i=1}^{N} \alpha_i)^{-1}$, $h(\boldsymbol{x} - cy\boldsymbol{w}^*) \neq h(\boldsymbol{x})$.*

Thus, for any set S, there is a universal constant c_S such that for any h that linearly separates S and any $(\boldsymbol{x}, y) \in S$, $-c_S y \boldsymbol{w}^*$ is an adversarial perturbation for h at $\boldsymbol{x}$.

Soft-margin Classifiers: The results above have two fundamental limitations. First, they only concern hard-margin linear classifiers. Second, they assume that the classifiers are trained on the same dataset S. Here, we extend the results above to soft-margin linear classifiers trained on samples drawn independently from some distribution. We will show that if we train two separate soft-margin linear classifiers on independently drawn training sets, then with high probability, there will be a single adversarial direction for both classifiers that affects most of the training examples in both sets.

Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y} \subseteq \mathbb{R}^d \times \{\pm 1\}$. We assume that with probability 1 over $\mathcal{D}$, $\|\boldsymbol{x}\|_2 \leq R$. Let $S^{(1)}, S^{(2)}$ be training sets of size n drawn independently from $\mathcal{D}$. Suppose we have two linear classifiers $\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}$ trained on $S^{(1)}$ and $S^{(2)}$ in some way. For notational simplicity, we assume no bias terms, though the same result can be derived in this setting. Thus, the label assigned to $\boldsymbol{x} \in \mathcal{X}$ by $\boldsymbol{w}^{(j)}$ is $h_j(\boldsymbol{x}) = \text{sign}(\langle \boldsymbol{w}^{(j)}, \boldsymbol{x} \rangle)$. We make no assumptions about the training process itself. We only make three relatively minor assumptions on the output of the training process.

Assumption A1. *With probability 1 over the training procedure, $\|\boldsymbol{w}^{(j)}\|_2 \leq B$.*

Assumption A2. *The probability that $\boldsymbol{w}^{(1)}$ correctly classifies any point in $S^{(1)}$ is independent from the probability $\boldsymbol{w}^{(2)}$ correctly classifies any point in $S^{(2)}$, and vice-versa.*

Assumption A3. *There are at least $(1-\rho)n$ for $\rho \in [0, 1)$ points $(\boldsymbol{x}, y) \in S^{(j)}$ such that $y\langle \boldsymbol{w}^{(j)}, \boldsymbol{x} \rangle \geq 1$.*

A1 is a straightforward assumption that states that the output of the training method cannot be of unbounded size, while **A2** essentially states that the training procedures are independent. **A3** simply says that h_j is correct and *confident* about its prediction on at least a $(1-\rho)$ fraction of the dataset. Under these assumptions, we have the following theorem about adversarial directions for h_1 and h_2.

Theorem 7. *Under the assumptions above, with probability at least $1 - 4n^{-2}$, there is a set $S \subseteq S^{(1)} \cup S^{(2)}$ with max-margin SVM $\boldsymbol{w}_S^*$ such that for $j \in \{1, 2\}$,*

1. *$|S \cap S^{(j)}| \geq (1-2\rho)n - 2BR\sqrt{n} - 4\sqrt{n \ln(n)}$.*
2. *For all $(\boldsymbol{x}, y) \in S$, $-y\boldsymbol{w}_S^*$ is an adversarial direction for h_j at $\boldsymbol{x}$.*

Note that for $\rho < \frac{1}{2}$ and $BR = o(\sqrt{n})$, $|S \cap S^{(j)}| = \Theta(n)$. Thus, there is a large subset of $S^{(1)} \cup S^{(2)}$ that both h_1 and h_2 are accurate on, but for which there is an adversarial direction $\boldsymbol{v}$ that transfers among all points in S and both classifiers.

To prove this, we first use Rademacher complexity bounds to show there is a large subset $S \subseteq S^{(1)} \cup S^{(2)}$ that both h_1 and h_2 classify correctly. Once we know that such an S exists, we can use our hard-margin results to find an adversarial direction $\boldsymbol{v}$ that is adversarial for both classifiers on most of the union of the training sets. The details are in Appendix **??**.

4 Multi-class Linear Classifiers

Hard-margin Classifiers: Suppose now that we have data $\mathcal{X} \subseteq \mathbb{R}^d$ with K potential labels. We will assume these labels are $\{1, 2, \ldots, K\}$. We now consider multi-class classification. One standard way to extend linear classifiers to multi-class classification is to use a *one-versus-rest* approach. With this approach, we train K binary linear classifiers. We train the k-th linear classifier on a version of S with modified labels, where labels of k are replaced with $+1$ and the remaining labels are replaced with -1. To classify $\boldsymbol{x}$, we evaluate it on all K linear classifiers and return the class maximizing its output value.

Formally, we consider classifiers of the form $h(\boldsymbol{x}) = g(W\boldsymbol{x} + \boldsymbol{b})$ where $W \in \mathbb{R}^{K \times d}, \boldsymbol{b} \in \mathbb{R}^K$ and $g(\boldsymbol{z}) = \text{argmax}_{1 \leq i \leq K} z_i$. Intuitively, $g(\boldsymbol{z})$ returns the index i of the largest entry of $\boldsymbol{z}$. We let $\mathcal{M}$ denote the set of such classifiers.

This setup generalizes standard techniques such as soft-max layers. The soft-max function $s : \mathbb{R}^d \to \mathbb{R}^d$ turns any vector in to a probability vector, but satisfies $g(s(\boldsymbol{z})) = g(\boldsymbol{z})$. Therefore, our setup includes the soft-max function and any classifier h that returns the index maximizing the vector $W\boldsymbol{x} + \boldsymbol{b}$. Let S denote some set

of instances with labels in $\{1, \dots, K\}$. We will let $\mathcal{M}_S$ denote the set of S-accurate classifiers.

Note. *A linear classifier is S-accurate if $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$ is positive for all positive instances and negative for the rest. For the multi-class case, S-accuracy does not require positivity of any classifier. Suppose $\boldsymbol{x}$ has label k, and let $\boldsymbol{z} = W\boldsymbol{x} + \boldsymbol{b}$. To classify $\boldsymbol{x}$ accurately we do not require $z_j < 0$ for $j \neq k$. In fact, all of $\boldsymbol{z}$ can be positive as long as the largest entry is z_k.*

Recall that for $k \in \{1, \dots, K\}$, we define $S[k] = \{(\boldsymbol{x}, y) \in S \mid y = k\}$. We will use max-margin classifiers to construct transferable adversarial directions for $S[k]$, as in the following theorem.

Theorem 8. *Fix $k \in \{1, \dots, K\}$. There is a vector $\boldsymbol{v}_k^*$ such that for all $h \in \mathcal{M}_S$ and for all $(\boldsymbol{x}, y) \in S[k]$, $\boldsymbol{v}_k^*$ is an adversarial direction for h at $\boldsymbol{x}$.*

A proof is given in the appendix. In fact, the theorem actually shows a slightly stronger statement. Namely, that for distinct $k, l \in \{1, \dots, K\}$, there is a vector $\boldsymbol{v}_{k,l}^*$ that is an adversarial direction for all $h \in \mathcal{M}_S$ and for all $(\boldsymbol{x}, y) \in S[k, l]$, where $S[k, l]$ was defined as $\{(\boldsymbol{x}, y) \in S \mid y \in \{k, l\}\}$.

Note that it is not necessarily true that for sufficiently large c, for any $\boldsymbol{x} \in S[k]$, $h(\boldsymbol{x} - c\boldsymbol{w}^*) = m$. The direction $-\boldsymbol{w}^*$ may also increase the output associated to some other label. Since we know that $(W(\boldsymbol{x} - c\boldsymbol{w}^*) + \boldsymbol{b})_k < (W(\boldsymbol{x} - c\boldsymbol{w}^*) + \boldsymbol{b})_l$ for sufficiently large c, this is sufficient to show that $h(\boldsymbol{x} - c\boldsymbol{w}^*) \neq k = h(\boldsymbol{x})$.

Soft-margin Classifiers: As in Section 3, we can extend our results to soft-margin classifiers trained on independently drawn sets. Suppose that we have some distribution $\mathcal{D}$ on $\mathcal{X} \times \{1, \dots, K\}$ such that with probability 1 over $\mathcal{D}$, $\|\boldsymbol{x}\|_2 \leq R$. Let $S^{(1)}, S^{(2)}$ be training sets drawn independently from $\mathcal{D}$. We have two models $W^{(1)}, W^{(2)}$ trained on $S^{(1)}$ and $S^{(2)}$ in some way again with no bias terms. Let $\boldsymbol{w}_i^{(j)}$ denote the transpose of the i-th row of $W^{(j)}$. The label assigned to $\boldsymbol{x} \in \mathcal{X}$ by $W^{(j)}$ is given by $h_j(\boldsymbol{x}) = \max_{1 \leq i \leq K} \langle \boldsymbol{w}_i^{(j)}, \boldsymbol{x} \rangle$.

As in Section 3, we do not make any explicit assumptions on the training process, only a few minor assumptions on the output models. Fix some $k \in \{1, \dots, K\}$. Recall that for any set $S \subseteq \mathbb{R}^d \times \{1, \dots, K\}$, $S[k]$ denotes the subset of S with label k and $S[k, l]$ denotes the subset with label k or l. We assume the following three properties hold for $j \in \{1, 2\}$.

Assumption B1. *With probability 1 over the training procedure, for $i \in \{1, \dots, K\}$, $\|\boldsymbol{w}_i^{(j)}\|_2 \leq B$.*

Assumption B2. *The probability that $W^{(1)}$ correctly classifies any point in $S^{(1)}$ is independent from $S^{(2)}$ and $W^{(2)}$, and vice-versa.*

Assumption B3. *There is a label l such that at least a $(1 - \rho)$ fraction of points $(\boldsymbol{x}, y) \in S^{(j)}[k, l]$ satisfy $h_j(\boldsymbol{x}) = y$ and $|\langle \boldsymbol{w}_k^{(j)}, \boldsymbol{x} \rangle - \langle \boldsymbol{w}_l^{(j)}, \boldsymbol{x} \rangle| > 1$.*

B1 and **B2** are analogues of **A1** and **A2** in the linear classifier case and are relatively straightforward assumptions. **B3** is a kind of multi-class generalization of **A3**. It says that our classifier is approximately accurate on a fraction of samples with label k or l, and moreover, that for such samples, $W^{(j)}$ is much more confident about the label k than the label l. In other words, $W^{(j)}$ does not conflate the labels k and l.

For simplicity we assume that $|S^{(1)}[k, l]| = |S^{(2)}[k, l]| = n$. In general, we can derive an analogous result to Theorem 9 below based on the ratio of $|S^{(1)}[k, l]|$ to $|S^{(2)}[k, l]|$. Note that as we sample more from $\mathcal{D}$, with high probability $|S^{(1)}[k, l]|/|S^{(2)}[k, l]|$ will be close to 1. We then have the following theorem, whose proof we leave to the appendix.

Theorem 9. *Fix $k \in \{1, \dots, K\}$ satisfying the above assumptions. With probability at least $1 - 4n^{-2}$, there is a vector $\boldsymbol{v}$ and a set $S \subseteq S^{(1)}[k, l] \cup S^{(2)}[k, l]$ such that for $j \in \{1, 2\}$,*

1. *$|S \cap S^{(j)}[k, l]| \geq (1 - 2\rho)n - 4BR\sqrt{n} - 4\sqrt{n \ln(n)}$.*
2. *For all $(\boldsymbol{x}, y) \in S$ such that $h_j(\boldsymbol{x}) = y$, $-\boldsymbol{v}$ is an adversarial direction for h_j if $y = k$, and $\boldsymbol{v}$ is an adversarial direction for h_j if $y = l$.*

Note. *Condition 2 of Theorem 9 is slightly weaker than condition 2 of Theorem 7. While the adversarial direction still transfers, we do not require h_1 to correctly classify $S \cap S^{(2)}[k, l]$. This slight weakening comes about by reducing the multi-class linear classification case to the single-class linear classification case.*

5 Two-layer ReLU Networks

In this section we show that for certain classes of two-layer ReLU networks and any data set S, there is an adversarial direction for all networks in this class that separate S. In other words, if we restrict to certain classes of two-layer ReLU networks, then fitting to a data set S forces the network to be susceptible in specific directions.

Let σ denote the ReLU function. For $z \in \mathbb{R}$, $\sigma(z) = \max\{0, z\}$. For $\boldsymbol{z} \in \mathbb{R}^n$, we abuse notation to let $\sigma(\boldsymbol{z})$ denote the vector with $\sigma(\boldsymbol{z})_i = \sigma(z_i)$. A two-layer *ReLU network* with hidden layer of width L is a function $f : \mathbb{R}^d \to \mathbb{R}^k$ of the form $f(x) = V\sigma(W\boldsymbol{x} + \boldsymbol{b})$.

For simplicity, we will only consider binary classification with labels $\{\pm 1\}$. Therefore, we restrict to the setting where $k = 1$, so that there is only one output unit.

In this case, a ReLU network is of the form $f(x) = \boldsymbol{v}^T\sigma(W\boldsymbol{x} + \boldsymbol{b})$ for some $\boldsymbol{v} \in \mathbb{R}^L$. A *ReLU classifier* is a function $h : \mathbb{R}^d \to \{\pm 1\}$ of the form $h(\boldsymbol{x}) = g(f(\boldsymbol{x}))$ where f is a ReLU network and $g : \mathbb{R} \to \{\pm 1\}$ is some decision rule. We will often consider g that are indicator functions of the form $g(z) = 1$ for $z \in A$ and -1 otherwise. We denote such a function by $\mathbb{J}_A(z)$.

We first consider ReLU networks $\mathcal{F}$ where $\boldsymbol{v}$ is the all-ones vector and ReLU classifiers $\mathcal{R}$ consisting of ReLU networks in $\mathcal{F}$ combined with the classification rule g where $g(x) = 1$ if $x > 0$ and -1 otherwise. Formally, we define:

$$\mathcal{F} := \left\{ f : \mathbb{R}^d \to \mathbb{R} \;\middle|\; f(\boldsymbol{x}) = \sum_{i=1}^{L} \sigma(\boldsymbol{w}_i^T\boldsymbol{x} + b_i) \right\}.$$

$$\mathcal{R} := \left\{ h : \mathbb{R}^d \to \{\pm 1\} \;\middle|\; \begin{array}{l} h(\boldsymbol{x}) = g(f(\boldsymbol{x})), \\ g(z) = \mathbb{J}_{(0,\infty)}(z) \end{array} \right\}.$$

Intuitively, such ReLU classifiers compute L functions of the form $\sigma(\boldsymbol{w}_i^T\boldsymbol{x} + b_i)$ and then return 1 if at least one of these is positive.

Let $S \subseteq \mathbb{R}^d \times \{\pm 1\}$ denote a set of labeled data. We assume that S contains at least one example with each label. We let $\mathcal{R}_S$ denote the set of S-accurate classifiers in $\mathcal{R}$. We then have the following theorem that shows the existence of transferable adversarial directions for such classifiers.

Theorem 10. *There is some $\boldsymbol{v}$ such that for all $h \in \mathcal{R}_S$ and $(\boldsymbol{x}, -1)$ such that $h(\boldsymbol{x}) = -1$, $\boldsymbol{v}$ is an adversarial direction for h at $\boldsymbol{x}$.*

Note that the theorem implies more than $\boldsymbol{v}$ being adversarial for h on S. In particular, it applies to any $(\boldsymbol{x}, -1)$ such that $h(\boldsymbol{x}) = -1$. Thus, $\boldsymbol{v}$ may be adversarial for more than just instances in the training set.

We will derive this theorem as a consequence of a more general theorem. Suppose we have two possible labels, α and β. Consider all functions $h : \mathbb{R}^d \to \{\alpha, \beta\}$. Suppose we have some set $S \subseteq \mathbb{R}^d \times \{\alpha, \beta\}$. We will analyze h such that $h^{-1}(\alpha)$ is convex (note that $h^{-1}(\beta)$ is not necessarily convex). We define

$$\mathcal{C} := \{h \mid h \text{ is continuous, } h^{-1}(\alpha) \text{ is convex}\}.$$

We denote the set of S-accurate h in $\mathcal{C}$ by $\mathcal{C}_S$. We then have the following theorem, which we prove in the appendix.

Theorem 11. *There is some $\boldsymbol{v}$ such that for all $h \in \mathcal{C}_S$ and $(\boldsymbol{x}, \alpha)$ such that $h(\boldsymbol{x}) = \alpha$, $\boldsymbol{v}$ is an adversarial direction for h at $\boldsymbol{x}$.*

The requirement that $h \in \mathcal{C}_S$ can be weakened. Fix $\boldsymbol{x}_\alpha \in S[\alpha]$ and $\boldsymbol{x}_\beta \in S[\beta]$. Let $\mathcal{T}$ denote the set of continuous functions $h : \mathbb{R}^d \to \{\alpha, \beta\}$ such that $h(\boldsymbol{x}_\alpha) = \alpha, h(\boldsymbol{x}_\beta) = \beta$, and there is some convex set C_h containing $\boldsymbol{x}_\alpha$ such that for all $\boldsymbol{x} \in C_h^c$, $h(\boldsymbol{x}) = \beta$. A similar proof then shows the following result.

Theorem 12. *Fix $\boldsymbol{x}_\alpha \in S[\alpha], \boldsymbol{x}_\beta \in S[\beta]$. There is some $\boldsymbol{v} \in \mathbb{R}^d$ such that for all $h \in \mathcal{T}$ and for all $(\boldsymbol{x}, \alpha) \in C_h$ such that $h(\boldsymbol{x}) = \alpha$, $\boldsymbol{v}$ is an adversarial direction for h at $\boldsymbol{x}$.*

To apply Theorem 11 to $\mathcal{R}_S$, it suffices to show the following lemma. We do so in the appendix.

Lemma 13. *For $h \in \mathcal{R}$, $h^{-1}(-1)$ is convex.*

We can extend $\mathcal{R}$ to include a broader class of ReLU classifiers. Define

$$\mathcal{F}' := \left\{ f : \mathbb{R}^d \to \mathbb{R} \;\middle|\; \begin{array}{l} f(\boldsymbol{x}) = \boldsymbol{v}^T\sigma(W\boldsymbol{x} + \boldsymbol{b}), \\ v_i \geq 0, \ \forall i \end{array} \right\}.$$

$$\mathcal{G} := \left\{ g : \mathbb{R} \to \{\pm 1\} \;\middle|\; \begin{array}{l} g(z) = \mathbb{J}_{(a,\infty)}(z), \ a > 0 \\ \text{or } g(z) = \mathbb{J}_{[a,\infty)}(z), \ a > 0 \end{array} \right\}.$$

We can view $\mathcal{F}'$ as the set of two-layer ReLU neural networks whose output layer has all nonnegative weights. We then define

$$\mathcal{R}' := \left\{ h : \mathbb{R}^d \to \{\pm 1\} \;\middle|\; \begin{array}{l} h(\boldsymbol{x}) = g(f(\boldsymbol{x})), \\ f \in \mathcal{F}', \ g \in \mathcal{G} \end{array} \right\}.$$

In order to apply Theorem 11, we would need to show that one of the decision regions of any $h \in \mathcal{R}'$ is convex. More formally, we have the following lemma.

Lemma 14. *For $h \in \mathcal{R}'$, $h^{-1}(-1)$ is convex.*

A proof is given in the appendix. Combining Theorem 11 and Lemma 14, we derive the following.

Theorem 15. *There is some $\boldsymbol{v}$ such that for all $h \in \mathcal{R}'_S$ and $(\boldsymbol{x}, -1)$ such that $h(\boldsymbol{x}) = -1$, $\boldsymbol{v}$ is an adversarial direction for h at $\boldsymbol{x}$.*

We can also extend this result to a class of ReLU classifiers $\mathcal{R}''$ that are similar to $\mathcal{R}$. Here, we consider the set of ReLU networks

$$\mathcal{F}'' := \left\{ f : \mathbb{R}^d \to \mathbb{R}^L \;\middle|\; f(\boldsymbol{x}) = \sigma(W\boldsymbol{x} + \boldsymbol{b}) \right\}$$

where $W \in \mathbb{R}^{L \times d}$ so that the output is potentially vector-valued and σ is the ReLU function applied entry-wise. Let $\phi : \mathbb{R}^L \to \mathbb{R}$ where $\phi(\boldsymbol{z}) = 1$ if all entries of $\boldsymbol{z}$ are positive and -1 otherwise. We then define

$$\mathcal{R}'' := \left\{ h : \mathbb{R}^d \to \{\pm 1\} \;\middle|\; \begin{array}{l} h(\boldsymbol{x}) = \phi(f(\boldsymbol{x})), \\ f \in \mathcal{F}'' \end{array} \right\}$$

Since $\phi(\boldsymbol{z}) = 1$ if all the entries of $\boldsymbol{x}$ are positive and -1 otherwise, $h^{-1}(1)$ is the intersection of L open half-spaces corresponding to the pair $(W, \boldsymbol{b})$. Therefore, $h^{-1}(1)$ is convex. Applying Theorem 11, we then get the following theorem concerning the set $\mathcal{R}''_S$ of S-accurate classifiers in $\mathcal{R}''$.

Theorem 16. *There is some v such that for all $h \in \mathcal{R}''_S$ and $(x, 1)$ such that $h(x) = 1$, v is an adversarial direction for h at x.*

6 Transferability Between Linear Classifiers and ReLU Networks

In this section we analyze whether the adversarial directions above transfer between classifiers of different types. We will determine if and when adversarial directions for linear classifiers in Section 3 transfer to the ReLU classifiers in Section 5 and vice-versa.

Suppose we have some training set $S \subseteq \mathbb{R}^d \times \{\pm 1\}$. We wish to consider applying both linear classifiers and ReLU classifiers to S. Recall that above, we defined $\mathcal{H}$ to be the set of linear classifiers on $\mathbb{R}^d$. We also defined $\mathcal{R}'$ as the set of two-layer ReLU classifiers of the form $h(x) = g(f(x))$ where $g(z) = \mathbb{J}_{[a,\infty)}(z)$ or $g(z) = \mathbb{J}_{(a,\infty)}(z)$ and f is of the form $f(x) = v^T\sigma(Wx + b)$ with $v \geq 0$. Recall that for a set A, $\mathbb{J}_A(z)$ is 1 if $z \in A$ and -1 otherwise.

We will let $\mathcal{H}_S, \mathcal{R}'_S$ denote the set of S-accurate classifiers in each of these sets. In Section 3, we constructed adversarial directions that applied to all $h \in \mathcal{H}_S$ and all $(x, y) \in S$, while in Section 5, we constructed adversarial directions for all $h \in \mathcal{R}'_S$ and $(x, y) \in S[-1]$, that is, examples with label -1. We will show that the adversarial direction for $\mathcal{R}'_S$ constructed in Theorem 15 transfers to $\mathcal{H}_S$, while the adversarial direction for $\mathcal{H}_S$ constructed in Theorem 4 need not transfer to $\mathcal{R}'_S$.

To see that the adversarial direction v for $\mathcal{R}'_S$ transfers to $\mathcal{H}_S$, recall that this direction was derived from Theorem 11. In particular, all that is necessary for v to transfer to a classifier h is that $h^{-1}(-1)$ is convex and that $\mathcal{H}_S$ is non-empty. Since linear classifiers divide the data space in to half-spaces, we know that for any $h \in \mathcal{H}$, $h^{-1}(-1)$ and $h^{-1}(+1)$ are both convex. Therefore, we get the following corollary.

Corollary 17. *Suppose S is linearly separable. Then there exists $v \in \mathbb{R}^d$ such that for all $h \in \mathcal{R}'_S \cup \mathcal{H}_S$ and $x \in h^{-1}(-1)$, v is an adversarial direction for h at x.*

The adversarial direction for $\mathcal{H}_S$ constructed in Theorem 4 does not necessarily transfer to $\mathcal{R}'_S$. A graphical explanation of this is given in Figure 3. As stated in Corollary 17, the ReLU adversarial direction v_2 constructed in the proof of Theorem 15 transfers to the linear classifier classifier h_1. However, the linear classifier adversarial direction v_1 constructed in the proof of Theorem 4 does not serve as an adversarial direction for the -1 labeled points for h_2. The vector v_2 is constructed in the proof of Theorem 11 in Appendix **??**, and can be any vector v_2 in the direction of $x_\beta - x_\alpha$ for any $x_\beta \in S[1], x_\alpha \in S[-1]$.

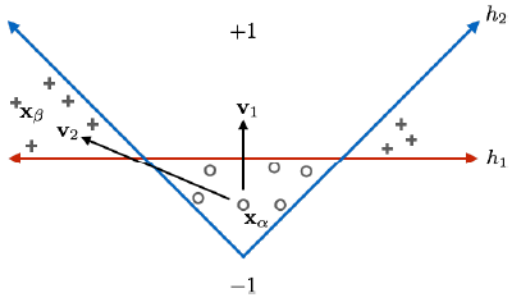


Figure 3: The decision boundary of a linear classifier h_1 and a two-layer ReLU classifier h_2, in red and blue, respectively. The instances with label 1 are marked by pluses, those with label -1 are marked by circles. The vector v_1 is the adversarial direction for h_1 from Theorem 4, and v_2 is the ReLU adversarial direction Theorem 15 pointing from x_α to x_β.

This intuitive idea formalized in Appendix **??**. There, we explicitly construct a linearly separable dataset and a ReLU network that correctly classifies the data but where the max-margin classifier does not transfer to the ReLU classifier.

7 Experiments

In this section we empirically study how well adversarial perturbations transfer between classifiers. As demonstrated by Moosavi-Dezfooli et al. (2017a), Narodytska and Kasiviswanathan (2016), and Papernot et al. (2016), adversarial examples often transfer to other classifiers. Here, we show empirically that adversarial perturbations for ReLU networks often transfer to other ReLU networks and SVMs. Similarly, adversarial perturbations for SVMs trained on different data sets often transfer among one another. Our experiments indicate that adversarial examples generated from SVMs transfer less often to ReLU networks.

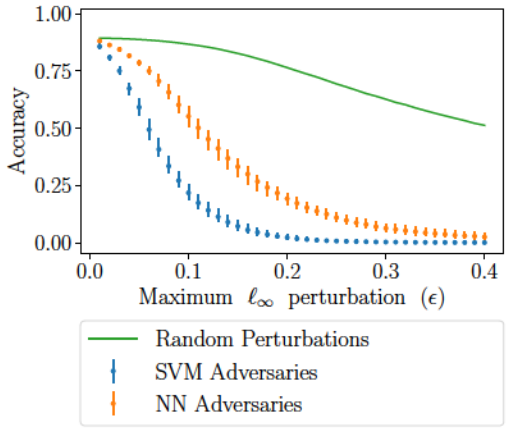
Setup: We use the MNIST dataset to perform our experiments. MNIST consists of handwritten numbers represented as vectors $x \in \mathbb{R}^{28\times 28}$ with labels in $\{0, 1, \ldots, 9\}$. We use Pytorch (Paszke et al., 2017) to train 50 moderately-sized neural networks with the same architecture as the PyTorch MNIST examples. Each neural network is trained on 10000 random images from the MNIST training set. We use scikit-Learn LinearSVC (Pedregosa et al., 2011; Fan et al., 2008) to train 50 support vector machines, each on 10000 random images from the MNIST training set.

To generate adversarial examples, we use the projected gradient method of Madry et al. (2017). Define $\mathcal{B}(x, \epsilon) = \{x' : \|x - x'\|_\infty \leq \epsilon\}$. To find an adversarial $x' \in \mathcal{B}(x, \epsilon)$, this method uses projected gradient ascent with step-size γ to maximize the loss function

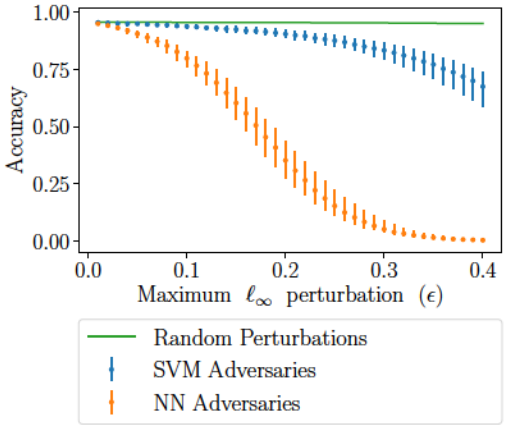
J. Thus, we move our iterates $x^{(t)}$ in the direction of $\gamma \nabla_{x^{(t)}}(J)$ and then project on to $\mathcal{B}(x, \epsilon)$.

We perform 40 iterations with $\gamma = .01$ to produce adversarial examples in $\mathcal{B}(x, \epsilon)$ for ϵ in $\{0.01, 0.02, \ldots, 0.40\}$. For each classifier and ϵ, we construct 1000 adversarial examples by applying FGSM to 1000 randomly selected examples from the MNIST test set. For each ϵ, this gives us 50000 SVM adversarial examples 50000 neural network adversarial examples. We also generate 50000 random perturbations for a given ϵ.

For each classifier, we find its overall accuracy on the random perturbations, SVM adversarial examples, and neural network adversarial examples. We plot the average accuracy (over all classifiers of a given type) in Figure 4. We also plot error bars corresponding to the minimum and maximum total accuracy of any single classifier on each type of adversarial examples.



(a) Average accuracy over all trained SVMs on various adversarial examples.



(b) Average accuracy over all trained neural networks on various adversarial examples.

Figure 4: Average classifier accuracy after applying perturbations of varying ℓ_∞ norm. Perturbations are generated randomly, by SVM adversaries, and by neural network (NN) adversaries.

As previously demonstrated in other works (see work by Liu et al. (2016) and Fawzi et al. (2016)), both SVMs and neural networks are relatively robust to random perturbations. Consider Figure 4a. This shows that SVMs are often susceptible to adversarial examples generated from other SVMs trained on different subsets of the data. This supports our result that SVMs trained independently from some distribution are jointly susceptible to some adversarial direction. Moreover, SVMs are still somewhat susceptible to adversarial examples generated from neural networks. While SVM adversaries are more effective than NN adversaries at fooling other SVMs, the neural network adversarial examples still do much better than random perturbations.

Similarly, neural network adversarial examples transfer to other neural networks, as shown in Figure 4b. Despite being trained on different subsets of the data, these neural networks are often susceptible to similar perturbations. This stands in stark contrast to the resilience of neural networks to random perturbations. This figure also supports our theory from Section 6 stating that adversarial examples generated from SVMs need not transfer to neural networks. SVM adversaries only do slightly better than random perturbations at fooling neural networks. This aligns with the empirically observed phenomenon that classifiers with more capacity seem to be more resistant to adversarial attacks (Goodfellow et al., 2014).

8 Conclusion

In this paper, we theoretically investigate the transferability of adversarial directions on linear classifiers and two-layer ReLU networks. Our results show that classifiers that accurately classify similar datasets are often jointly susceptible to some adversarial direction. This holds even without assumptions on the training procedure. Moreover, we show that while adversarial perturbations for ReLU networks transfer to linear classifiers, the reverse need not hold. We confirm all of our findings experimentally on MNIST.

Adversarial examples and transferability are still not fully understood. While we focus on adversarial directions, a natural extension would be to constrain the norm of the perturbation allowed. Second, our results are derived only for linear classifiers, and could be studied from a kernel perspective as well. Finally, our results above show that ReLU adversarial examples often transfer to linear classifiers. Is this phenomenon more general? Do adversarial examples classifiers with higher effective capacity tend to transfer to those with lower effective capacity? Answers to these questions could help inform future work in designing adversary-resistant classifiers.

References

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems*, pages 1632–1640, 2016.

Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning*, 107(3):481–508, 2018.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435*, 2016.

Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. Universal adversarial perturbations against semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2755–2764, 2017.

Hossein Hosseini, Yize Chen, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Blocking transferability of adversarial examples in black-box learning systems. *arXiv preprint arXiv:1703.04318*, 2017.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *arXiv preprint*, 2017a.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. Analysis of universal adversarial perturbations. *arXiv preprint arXiv:1705.09554*, 2017b.

Konda Reddy Mopuri, Utsav Garg, and R Venkatesh Babu. Fast feature fool: A data independent approach to universal adversarial perturbations. *arXiv preprint arXiv:1707.05572*, 2017.

Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. *arXiv preprint arXiv:1612.06299*, 2016.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness with principled adversarial training. *arXiv preprint arXiv:1710.10571*, 2017.

Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy? – a comprehensive study on the robustness of 18 deep image classification models. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.