THESIS

UTILIZING NETWORK FEATURES TO DETECT ERRONEOUS INPUTS

Submitted by

Matthew Gorbett

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2020

Master's Committee:

Advisor: Nathaniel Blanchard

Charles W. Anderson
Emily King

ABSTRACT

UTILIZING NETWORK FEATURES TO DETECT ERRONEOUS INPUTS

Neural networks are vulnerable to a wide range of erroneous inputs such as corrupted, out-of-distribution, misclassified, and adversarial examples. Previously, separate solutions have been proposed for each of these faulty data types; however, in this work I show that the collective set of erroneous inputs can be jointly identified with a single model. Specifically, I train a linear SVM classifier to detect these four types of erroneous data using the hidden and softmax feature vectors of pre-trained neural networks. Results indicate that these faulty data types generally exhibit linearly separable activation properties from correctly processed examples. I am able to identify erroneous inputs with an AUROC of 0.973 on CIFAR10, 0.957 on Tiny ImageNet, and 0.941 on ImageNet. I experimentally validate the findings across a diverse range of datasets, domains, and pre-trained models.

ACKNOWLEDGEMENTS

I really lucked out last year when I emailed Dr. Nathaniel Blanchard out of the blue. I wanted to pursue a research specific Master's Degree; however, I was an online student struggling to find an advisor. Nate decided to take a shot on me, and a month later the stars aligned when my partner Phoebe got a job here in Fort Collins, enabling me to move here and immerse myself in the program. Although it has been a strange year with coronavirus, I've really enjoyed it. I'm very grateful for the opportunity Professor Blanchard gave me; I've learned a ton and really grown as a person.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Introduction

Humans are capable of adapting to diverse types of data in ways machine learning models cannot [16]. While the human visual system is able to generalize across varying image representations such as different Instagram filters, deep learning classifiers misbehave when presented with image corruptions [11, 21], adversarial examples [48], and previously unseen classes [3, 23, 46]. In this work, I automatically detect a broad range of inputs that will be incorrectly processed by a given neural network.

The timely, preemptive detection of these failures is essential for preventing unreliable AI action based on incorrect predictions. An automated technique that broadly identifies when a model is in error is critically important for safe AI [1]. This need is particularly relevant now, given the increasingly ubiquitous deployment of deep learning models in real-world applications such as autonomous vehicles and medical devices. The main contribution of this work is the identification that the collective set of erroneous inputs that cause failure can be jointly identified and separated from correctly processed inputs. Figure 1 shows an example of each type of erroneous input:

Figure 1: Modern neural network models have state-of-the-art performance on image tasks like object classification. However, these networks are vulnerable to erroneous inputs i.e., images that are corrupted, out-of-domain, adversarial, or misclassified. Previous research has focused on separate solutions to defending against each type of erroneous input, but I show that all erroneous inputs can be jointly identified with a linear SVM trained on a given network's activations. Images from ImageNet [49], SUN [45], and ImageNet-C [21].

image corruptions [11, 21], adversarial examples [48], previously unseen classes [3, 23, 46], and misclassifications, an inevitable part of any model since no model has perfect performance.

Given the breadth of erroneous input variations, it is unsurprising that each of these inputs has been traditionally tackled as a separate, individual problem requiring a unique, specialized solution. For example, out-of-distribution detection identifies when images are outside of a model's training (i.e. unseen classes) [10, 23, 29, 30, 46] (additional related work on erroneous input detection is discussed in Section 2). In the study of these seemingly disparate phenomena I noticed a common theme: the network's internal activation patterns were notably distinct when stimuli were correctly processed. I hypothesized that erroneous inputs could be collectively separated from correctly processed inputs with a linear SVM trained on these features.

I test my hypothesis with four pre-trained image classification models, each pre-trained on a separate dataset, and find erroneous inputs can be broadly detected as a collective group and generalize to a multitude of alternative datasets. Further, I find that the detection technique

is robust to four types of adversarial attacks. To my knowledge, this work is the first to consider the feasibility of broadly detecting all erroneous inputs. Specifically, I find that the model's internal activations in the softmax layer and final hidden layer are sufficient for automatically filtering incorrectly processed inputs, as shown in Figure 2.

In summary, this paper makes the following contributions:

- This is the first research showing that the four distinct types of erroneous inputs in Figure 1, which neural networks will fail to correctly classify, can be jointly separated from correctly classified images using the network's activations in response to the input.
- I achieve results comparable to, or better than, state-of-the-art techniques by considering the activations of both the softmax and final hidden layer of common pre-trained neural networks.
- Further, I show how this method can be used as an auxiliary technique in existing anomaly detection models for enhanced performance.

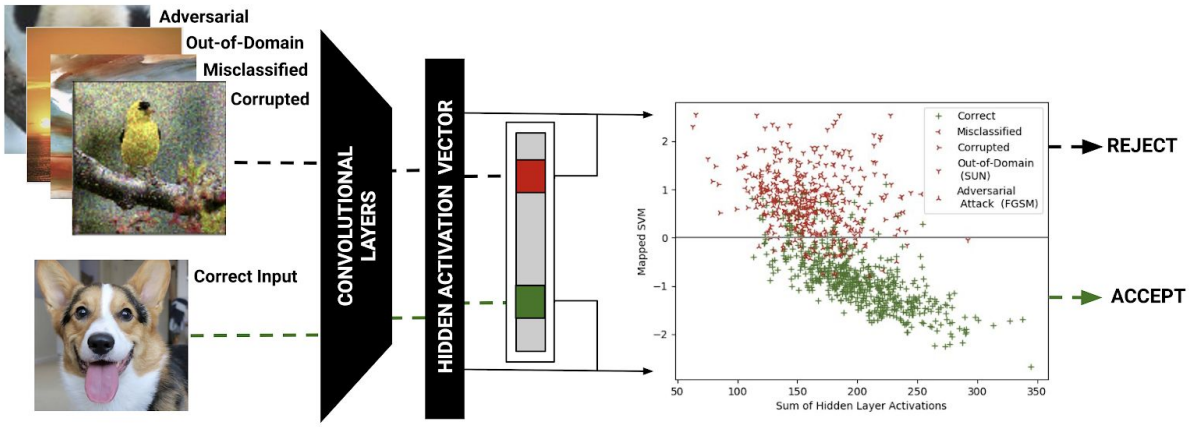


Figure 2: Erroneous inputs cause neural network's to incorrectly classify images. However, these inputs can be preemptively identified with a simple linear SVM that is trained on the network's internal activations (i.e. features) that occur in response to the image. Specifically, the SVM is trained on features from the hidden activation vector and the softmax vector of the network. Using this technique, erroneous inputs are detected with an AUROC of 0.973 on CIFAR-10, 0.957 on Tiny ImageNet, and 0.941 on ImageNet. Above is the optimal hyperplane for CIFAR10 mapping erroneous examples against correct examples.

Chapter 2 Related Work

A variety of works have studied automatically detecting inputs that will cause neural networks to fail. In this section, I discuss the four subfields that compose erroneous inputs and previously established connections between inputs.

2.1 Adversarial Attacks

Adversarial examples occur when small, meaningful changes to clean data alter a network's prediction. Szegedy et al. [48] highlighted these limitations, giving rise to the field of adversarial attacks. Since then, numerous methods have been proposed to generate adversarial attacks [6, 18, 37, 41]. Several works have attempted to defend against adversaries [32, 35]; however, the field largely conforms to a classic "white hat"/"black hat" paradigm (i.e. solutions are often temporary because new attacks are created to overcome the defensive solution [5]). To date, only a few defenses have proven effective on smaller input dimensions [33, 47]. Researchers have identified the upper bounds of adversarial robustness techniques for certain distributions [12, 17,34]. In particular, Fawzi et al. [12] suggested a strong relation between adversarial robustness and the linearity of a classifier in latent space. Gilmer et al. [17] showed that when a model misclassifies even a fraction of inputs, it can be exposed to adversarial perturbations of size $O(1/\sqrt{n})$, where 'n' is the number of input dimensions. My work finds that adversarial inputs, generated using a variety of methods (see Section 3.2.4), manifest distinct activation patterns that I use to detect when models will fail. I also consider "natural adversaries," which categorically overlap with misclassifications.

2.2 Image Corruption

Neural networks also perform poorly on corrupted examples. While adversarial images are typically malicious, image corruption is unintentional: corruptions range from image quality degradation to lighting changes. In particular, networks have shown to be susceptible to Gaussian noise, blur, pixelation, and JPEG compression [11, 21]. In real-world scenarios, like autonomous driving, models need to be robust to corruption from weather and debris [7, 36]; some research even proposed that networks should be explicitly evaluated on their robustness to corruption [42].

2.3 Out-of-Distribution Detection

An OoD example occurs when a model is presented with data outside of its training paradigm. In this work, I defined OoD examples as originating from datasets other than the dataset the baseline model was trained on, although I controlled for class ambiguity (for example, if a dataset was trained on dogs, I controlled for wolves in OoD data -- see section 3.2.2 for details). Hendrycks and Dietterich [22] established a benchmark for OoD detection by using the Maximum Softmax Probability (MSP). Results were presented on several datasets in different neural network models such as computer vision and natural language processing. Other works trained neural networks to reject out-of-distribution data with auxiliary branches [10], auxiliary datasets [23], GAN examples [29], and other train-time techniques [4, 15]. A notable cross-domain technique was proposed by Liang et al. [30], who used adversarial perturbations on input data to discriminate in-distribution from OoD examples. Bendale and Boult proposed OpenMax [3], a network layer that used Meta Recognition to estimate the

probability of an input originating from an unknown class. Specifically, for a given example, OpenMax takes the activation vector mean from each correctly predicted class to compute a Weibull distribution to determine the probability that a new example is OoD. They successfully detect out-of-distribution images and some types of adversarial attacks. Their out-of-distribution set contains fooling images labeled with high-confidence in standard softmax models, similar to my ImageNet-O set referenced section 3.2.2. Additional work by Rozsa et al. [44] proposed the adversarial algorithm LOTS as an attack against the OpenMax algorithm. The algorithm modifies an image in order to mimic the features of the penultimate layer of correct examples, and was shown to break OpenMax. I would like to note that this algorithm could theoretically break the proposed SVM model; however, I do not make any claims that the model is robust to any adversarial attack.

2.4 Misclassified Examples and Thresholding

Misclassified examples occur when a model fails to predict an instance of a class, despite being trained to predict that classand the class existing within the training domain. One technique to detect this poorly-learned data is thresholding. Thresholding has a rich history of research [8, 9]; however, I believe I am the first to consider multiple layers of the network for thresholding. Hendrycks and Dietterich [22] provide thresholding results for misclassified examples on contemporary datasets, while [14] presented results of the softmax response from a risk perspective. In recent years, thresholding techniques have been applied to detect adversarial examples [32, 40] and OoD examples [22]. Lu et al. [32] provide the approach most similar to my own. They utilize layers toward the end of deep learning models to detect adversarial examples with SVM-RBF classifiers. My work also shows that some adversarial

examples can be identified using these features; however, I achieve higher accuracy by considering the hidden layers as well as the softmax layer.

2.5 Connecting Erroneous Inputs

Some recent research has found similarities in the above areas. Hendrycks and Gimpel [22] showed that both misclassified and out-of-distribution examples can be detected by utilizing the maximum softmax probability. Ford et al. [13] proposed a similarity between the fields of adversarial and corrupted examples, showing empirical and theoretical evidence that these two fields are manifestations of the same phenomena. Another notable cross-domain technique was proposed by Liang et al. [30], who used adversarial perturbations on input data to discriminate in-distribution from OoD examples. Finally, Rozsa and Boult [43] argued that adversarial perturbations exist in open space, contrary to popular belief that they exist near training samples. The above research highlights the distinct work being done in each of these four fields, and previous work that has identified connections between some of these fields. To my knowledge, this work is the first to fully explore the link between all of these areas, and the first to detect them with a single approach.

Chapter 3 Method

I first formally define the domain of erroneous inputs (Section 3.1) and set up the experimental design (Section 3.2), enumerating the pre-trained models and datasets used in the experiments. Finally, I define the methods for training and testing erroneous input detectors (Section 3.3) and the evaluation metrics used to evaluate the results (Section 3.4).

3.1 Defining the Domain

I defined the domain in the context of detecting when input data to a visual classification model, f(x), was classified incorrectly. Incorrect classifications in the experiments encompassed out-of-distribution classes (D_{out}), adversarial examples (A), corrupted examples (C), and misclassified in-distribution data (M). M could be considered any input from in-domain dataset D_{in} where output $Y_{predicted} \neq Y_{actual}$.

In my experiments, I train a binary classifier, h(X'), to detect erroneous examples {D_{out}, A, C, M}, where X' is the activations of the penultimate and final layers of a pre-trained model for image X. In h(X'), I consider D_{out}, A, C, and M as belonging to the positive class, which were in turn classified against the correctly predicted in-domain dataset D_{in}, where $Y_{predicted} = Y_{actual}$.

To build the training and tests set for the binary classifier I use validation and test sets from the dataset under test. My training set was built such that $f(A) \neq Y_{actual}$ and $f(C) \neq Y_{actual}$ — I only used samples which the pre-trained neural network classified incorrectly. When an adversarial or corrupted example is classified correctly, I do not want to remove the example from the baseline models purview. Further, adversarial examples were generated from inputs the model originally predicted correctly, i.e. $f(X) = Y_{actual}$ and $f(X \rightarrow A) = Y_{actual}$. Finally, it is implied

that $f(D_{out})$ and f(M) would produce invalid results, so each example will be taken from these datasets.

It should be noted that I did not classify in-domain versus out-of-domain data because I consider the set of everything except D_{out} to be in-domain: $\{D_{in}, A, C, M\} \in D$ while $D_{out} \notin D$. Further, my initial experiments were set up to classify correct example sets from each of the erroneous examples sets, i.e. classifying correct examples against the set of misclassified examples.

3.2. Experimental Setup

For my experiments I used several common baseline datasets and models for image classification. The baseline models were fed various correct data and erroneous data and the activations for both the softmax output and the penultimate layer of the model were collected for use in the detection models. The baseline models were not altered in any way during experiments.

3.2.1 Baseline Datasets and Models

Below are the summaries of the pre-trained models used foreach dataset under test. For uniformity, all images passed to the pre-trained models were resized and normalized the same way.

CIFAR-10 contains 32 x 32 colored images of 10 different classes of objects [28]. The dataset has 50,000training images and 10,000 testing images. The CIFAR-10 model was trained using the ResNet50 architecture and open-sourced by [33].

Tiny ImageNet is a 200-class subset of the ImageNetdataset where images were cropped and resized to a resolution of 64 x 64. Bounding box information was used inthe image cropping

[25]. The Tiny ImageNet model is a pre-trained WideResNet [50]. The trained model was open-sourced by [21].

ImageNet consists of 1000 classes of objects [45], with varied dimensions and resolutions. My work used the validation dataset to produce examples for the detection model. I utilized the default pre-trained ResNet50 model in the Pytorch library for the ImageNet experiments.

3.2.2 Out-of-Distribution Datasets

I use several commonly used OoD datasets:

CIFAR-100 contains 32 x 32 colored images of 100 different classes of objects [28]. I filtered out classes similar to those found in CIFAR-10 to ensure all classes were truly OoD, leaving 74 classes. *CIFAR-100 Excluded classes:* Bear, Bus, Camel, Cattle, Dinosaur, Elephant, Fox, Hamster, Kangaroo, Leopard, Lion, Lizard, Mouse, Pickup Truck, Possum, Rabbit, Raccoon, Shrew, Skunk, Squirrel, Streetcar, Tank, Tiger, Tractor, Train, Wolf

SVHN I use the test set from the Street View HouseNumbers dataset [38], which contains colored numbers with class labels 0 to 9.

The Scene UNderstanding dataset (SUN) contains images of scenes with varying resolutions [49].

Places365 contains 365 classes of scenes with varying image resolutions. I use the high-resolution validation set for use in my Tiny ImageNet and ImageNet models [51].

ImageNet-O dataset was constructed from sampled images from ImageNet-22K [24]. First, images overlapping with classes in ImageNet-1K were filtered out. Next, they retained images that a ResNet50 ImageNet model classified with high softmax probability. Finally, they hand-selected a subset of high-quality images.

3.2.3 Corruption Datasets

CIFAR-10-C, Tiny ImageNet-C, ImageNet-C are image corruption datasets for the purpose of testing model robustness [21]. Each corrupted dataset included 15 common visual corruptions such as Gaussian noise, blur, and digital distortions. Each type of corruption had five levels of 'severity' for a total of 75 distinct corruptions per image. I used each corruption and severity type in the models.

3.2.4 Adversarial Attack Methods

White-box attack methods are used to generate adversarial examples. White-box methods assume the attacker has full access to the learning model.

Fast Gradient Sign Method (FGSM) was an attack introduced by Goodfellow et al. [18]. It adjusts the inputs to maximize the loss based on the back propagated gradients of the predicted value: $f(A)=X+\varepsilon sign(\nabla_x J(\theta, X, Y))$. FGSM examples were generated using=0.01.

Carlini and Wagner l_2 Attack (C & Wl2) searches for low distortion in the l_2 metric [6]. Notable in the Carlini and Wagner attacks is κ, which allows a confidence level for the adversarial example to be calibrated. This means examples can be generated that the neural network predicts incorrectly with high probability. Examples in my main experiments were generated with κ=0 — I did not search for a perturbed input which satisfied a specific confidence.

Carlini and Wagner l_∞ Attack (C & W l_∞) was a modified version of the C&W l_2 attack. It controls the l_∞ norm, i.e. the maximum perturbation applied to any pixel [6].

Projected Gradient Descent (PGD) finds the perturbation that maximizes the loss of a model on an input, and, after each iteration, it projects the perturbation onto an L_P ball of radius while also clipping values so they fall within a permitted range [33]. Carlini and Wagner l_2 , l_∞ attacks

and PGD attack examples were generated using the adversarial-robustness-toolbox with default parameters [39]. I used the PyTorch implementation of the FGSM attack.

ImageNet-A is a dataset of 'natural adversaries'. ImageNet-A was created by taking examples from the ImageNet dataset and removing examples the model predicted correctly. Then, a subset of high-quality images were hand-selected [24]. I included this dataset as an extension to the misclassified experiments.

Special considerations for adversarial examples:

- While I test the method on four different adversarial attacks, I do not claim this method to be robust to any attack. Section 5.2 looks at this further.
- I looked at the first 20 generated examples of each adversarial dataset to verify the perturbed image was realistic.

3.3. Detection Model

I employed a linear Support Vector Machine (SVM) classifier for my experiments, where examples ({x1}, y1),..., ({xn}, yn) were trained to maximize the hyperplane between the groups y=0 and y=1. Correct examples were the base class (y=0).While MSP and OE use a model-less ROC during their evaluation, I instead evaluate my detection model using five-fold cross validation. I found that linear SVMs were able to generalize better in high dimensions than other SVM kernels. To generate features {x1}...{xn} for my model, I take all values from the softmax output vector and sort them from smallest to largest. I then take the activations from the penultimate layer of the baseline model, which is a fully-connected layer. For ResNet50 pretrained models (CIFAR10 and ImageNet), the penultimate layer is 2048 values, while Tiny

ImageNet has a penultimate layer of size128. I present results in this paper with a balanced dataset: each model contains the same number of correct and erroneous examples.

Algorithm 1 SVM Training for the hidden activation + sorted softmax algorithm

Require: Pretrained object classification model, $f(\mathcal{X})$, where $f_p(\mathcal{X})$=penultimate layer of $f(\mathcal{X})$ and $f_{ss}(\mathcal{X})$=sorted softmax output of $f(\mathcal{X})$, i.e. $(f(\mathcal{X}))$.

Require: Known erroneous example set$\{\ \mathcal{D}_{in}, \mathcal{A}, \mathcal{C}, \mathcal{M}\}$ and known correct example set $\{\mathcal{C}\}$. Erroneous examples have y-label 1 in SVM model, while correct examples have y-label 0 in SVM.

Ensure: Size of correct example set is equal to size of erroneous example set: $|\{\mathcal{C}\}| = |\{\mathrm{D}_{in}, \mathcal{A}, \mathcal{C}, \mathcal{M}\}|$

1: **for i** in $\{\mathcal{C}, \mathcal{D}_{in}, \mathcal{A}, \mathcal{C}, \mathcal{M}\}$ **do**
2: $\quad features \leftarrow [p(i), ss(i)]$
3: **end for**
4: SVM Fit $\{features_i, ...features_n\}$
5: **return** SVM detection model

Figure 3: End-to-end algorithm for the linear SVM algorithm trained on the hidden and softmax features of a pretrained neural network.

3.4. Testing and Evaluation Metrics

I evaluate binary detection tasks using three metrics: area under the receiver operating characteristic curve (AUROC) and area under the precision-recall curve (AUPR), and false positive rate at N% true positive rate (FPRN). AUROC plots the true positive rate (TPR) against the false positive rate (FPR). Random classifiers score 50% while perfect classifiers achieve 100%. AUPR is another cumulative distribution function which plots the precision(True Positive)/(True Positive + False Positive) versus the recall (True Positive)/(True Positive + False Negative). For my ImageNet model I also present FPRN scores, also used by [23, 30, 31]. The FPRN calculates the false positive rate at a set True Positive Rate. I use TPR of 95%, similar to past papers.

3.5. Other Experiment Details

Dataset sizes Sizes generally ranged from 3,500 examples up to 50,000 examples. There were two exceptions to this rule: the CIFAR10 pretrained model only misclassified 475 images on the test set, and TinyImageNet FGSM attacks rendered only 1,764 bad examples. The number of correct examples generated were 9,525 for CIFAR10, 6,366 for Tiny ImageNet, and 23,350 for ImageNet. Further details in Table 1.

Table 1: Erroneous dataset sizes. Correct example set sizes described in Section 3.5.

Test Type	Dataset Size
CIFAR10	
Misclassified	475
SUN	16873
SVHN	20000
CIFAR10-C	4547
CIFAR100	7400
FGSM	5653
C & W l_∞	3501
C & W l_2	3501
PGD	3501
Tiny ImageNet	
Misclassified	3634
SUN	10000
Corrupted	18730
Places	15000
FGSM	1185
PGD	4001
C & W l_∞	4001
C & W l_2	4001
ImageNet	
Misclassified	7000
SUN	15000
Places365	15000
ImageNet-C	15015
ImageNet-O	2000
ImageNet-A	7492
FGSM	10000
C & W l_∞	3501
C & W l_2	3501
PGD	3501

Baseline detection models I test my erroneous datasets on two baselines: MSP [22] and Outlier Exposure [23]. For MSP, I use the pre-trained models described in Section 3.2.1. I use the same pre-trained models for my main experiments, the columns labeled “Linear SVM” in Tables 1 and 2. As an additional measure, I test the datasets on Outlier Exposure by using pre-trained models open sourced by the researchers on GitHub. The models were trained with a modified loss function which encourages out-of-distribution examples to fit a uniform distribution. An important note is that they expose the pretrained model to OoD examples. Further, the choice of OoD exposure was important, for instance, corrupting in-distribution examples with noise did not perform well. As a result, their models were trained with realistic OoD data from 80 Million Tiny Images dataset and ImageNet22k.

Chapter 4 Results

In this section, I present my findings: First, Section 4.1 breaks down the results by pretrained models and datasets, then, Section 4.2 presents the combined results across erroneous inputs.

4.1 Single Erroneous Dataset Results

In Table 2, I directly compared the applicability of the detection model with Maximum Softmax Probability (MSP) [22], as well as Outlier Exposure (OE) [23]. Additionally, I applied the method to existing OE models for CIFAR10 and Tiny ImageNet. Results show that the method helps both Outlier Exposure models as well as standard pretrained models. By applying the method to pre-trained models, I am able to outperform OE and MSP on 15 of 17 datasets. These results show evidence that the linear kernel of the SVM is able to generalize to new examples across validation sets. In Table 3, I look at the method for ImageNet examples in a pretrained ResNet model with weights loaded from PyTorch. My linear SVM is again trained with five-fold cross validation and compared to the baseline AUROC curve of the MSP. All of the models were tested on the same examples.

Table 2: CIFAR-10 and Tiny ImageNet experiments for MSP, Outlier Exposure, Outlier Exposure + Linear SVM (Mine), and Linear SVM (Mine). Results are presented using five-fold cross validation for each model. All error rates are near zero (less than 0.01). Best results for each row are bolded.

Test Type		MSP [22]		OE [23]		OE+Linear SVM (Ours)		Linear SVM (Ours)	
CIFAR-10	**Detail**	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
Misclassified	-	.930	.905	.927	.908	.935	.910	**.939**	**.911**
Out-of-Distribution	Sun	.928	.912	.998	.998	**1.0**	**1.0**	.989	.989
Out-of-Distribution	CIFAR100	.925	.908	.992	.991	**1.0**	**1.0**	.970	.966
Out-of-Distribution	SVHN	.954	.937	.998	.996	**1.0**	**1.0**	**1.0**	**1.0**
Corrupted	CIFAR10-C	.945	.926	.961	.959	.975	.975	**.982**	**.982**
Adversarial	FGSM	.830	.825	.998	.998	**1.0**	**1.0**	.980	.980
Adversarial	C & W l_2	.982	.973	.752	.747	.891	.891	**.991**	**.989**
Adversarial	C & W l_∞	.916	.893	.817	.808	.910	.906	**1.0**	**1.0**
Adversarial	PGD	.999	.999	.989	.996	**1.0**	**1.0**	**1.0**	**1.0**
Combined		.925	.915	.871	.908	**.973**	**.977**	.969	.970
Tiny ImageNet									
Misclassified	-	**.860**	**.834**	.846	.809	.820	.803	.847	.822
Out-of-Distribution	Sun	.876	.864	**.999**	**.999**	.999	.998	.994	.993
Out-of-Distribution	Places365	.882	.873	.993	**.999**	**.999**	.998	.992	.991
Corrupted	TinyImageNet-C	.895	.885	.996	.996	**.998**	**.997**	.994	.993
Adversarial	FGSM	.998	.999	.720	.699	.984	.983	**.999**	**.999**
Adversarial	C & W l_2	.908	.835	.823	.794	**.994**	**.994**	.967	.960
Adversarial	C & W l_∞	.855	.782	.815	.785	**.995**	**.995**	.887	.858
Adversarial	PGD	.990	.991	.743	.718	.996	.998	**1.0**	**1.0**
Combined		.886	.870	.828	.863	**.957**	**.964**	.931	.933

4.2. Joint Erroneous Example Detection

Erroneous inputs were randomly sampled from the set of all erroneous datasets to create a balanced dataset, i.e. {D_{out}, A, C, M}. In Tables 1 and 2 I add these results in the 'Combined' row. I found that, across all of the models, erroneous inputs could be detected with reasonably high performance (AUROCs between 0.941 and 0.973). A manual analysis of model failures found that examples from PGD attacks were consistently not detected by the model because their maximum softmax probabilities tended to be higher (PGD inputs had nearly 100% probability on average) than for correctly classified examples (80% MSP average for ImageNet, 86% MSP average for Tiny ImageNet). All other erroneous example sets were lower (between

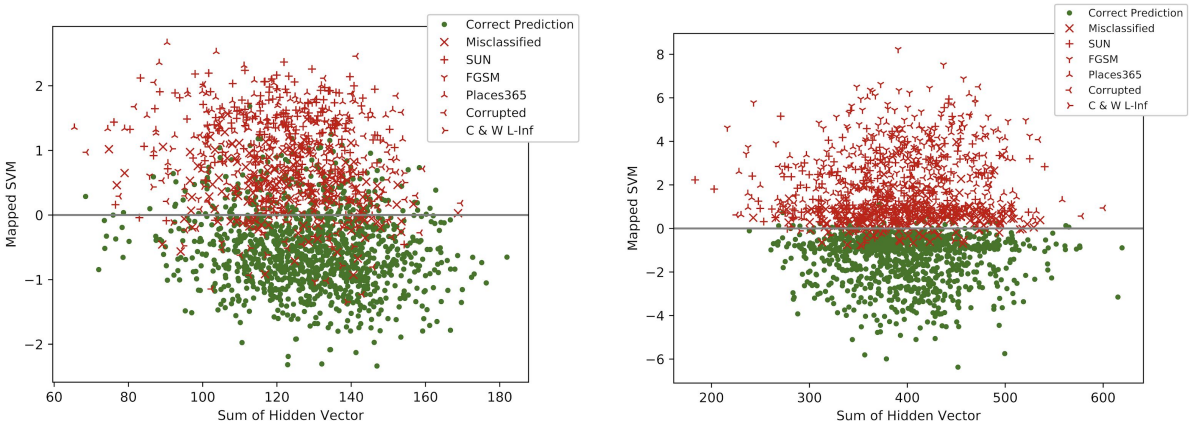
25% and 93% average). Table 4 details this further. As a result, I exclude PGD examples from the combined results. Since PGD examples are easily separable from correct examples for each of the models, a PGD detector can be implemented as a separate model.

Table 3. ImageNet experiments for MSP as well as the Linear SVM model. Up arrows indicate when a higher score is better, while down arrows indicate when a lower score is better.

ImageNet		MSP [22]			Linear SVM (Mine)		
Test Type	**Detail**	AUROC ↑	AUPR ↑	FPR (95%) ↓	AUPR ↑	AUROC ↑	FPR (95%) ↓
Misclassified	-	.853	.833	.506	**.898**	**.868**	**.395**
Misclassified	ImageNet-A	.908	.910	.423	**.940**	**.940**	**.052**
Out-of-Distribution	Sun	.830	.822	.640	**.983**	**.981**	**.021**
Out-of-Distribution	ImageNet-O	.656	.552	.617	**.897**	**.873**	**.035**
Out-of-Distribution	Places365	.846	.843	.618	**.983**	**.981**	**.015**
Corrupted	ImageNet-C	.927	.924	.326	**1.0**	**1.0**	**0.0**
Adversarial	FGSM	.949	.944	.222	**1.0**	**1.0**	**0.0**
Adversarial	C & W l_∞	.870	.802	**.320**	**.875**	**.847**	.444
Adversarial	C & W l_2	.890	.812	**.223**	**.930**	**.911**	.236
Adversarial	PGD	.990	.990	.030	**.996**	**.997**	**.001**
All		.881	.876	.472	**.941**	**.947**	**.277**

Table 4: The mean and variance of the maximum softmax probability for each correct and erroneous datasets under test.

Test Type	Detail	Mean	Variance
CIFAR10			
Correct	-	.943	.015
Misclassified	-	.637	.031
SUN	Out-of-Distribution	.642	.044
CIFAR100	Out-of-Distribution	.657	.041
CIFAR10-C	Corrupted	.556	.027
FGSM	Adversarial	.649	.03
C & W l_∞-	Adversarial	.844	.027
C & W l_2-	Adversarial	.624	.02
PGD	Adversarial	1	0
Tiny ImageNet			
Correct	-	.865	..035
Misclassified	-	.423	.041
SUN	Out-of-Distribution	.45	.058
Places	Out-of-Distribution	.431	.058
TinyImageNet-C	Corrupted	.443	.057
FGSM	Adversarial	.457	.058
PGD	Adversarial	.999	0
C & W l_∞	Adversarial	.551	.03
C & W l_2	Adversarial	.464	1.3
ImageNet			
Correct	-	.80	.053
Misclassified	-	.426	.06
SUN	Out-of-Distribution	.443	.074
Places365	Out-of-Distribution	.416	.072
ImageNet-C	Corrupted	.289	.05
ImageNet-O	Out-of-Distribution	.947	0
ImageNet-A	Misclassified	.31	.06
FGSM	Adversarial	.24	.034
C & W l_∞	Adversarial	.45	.03
C & W l_2	Adversarial	.423	.019
PGD	Adversarial	.999	0



Figures 4/5: Two graphs showing the linear separability of erroneous datasets from correctly classified examples. “0” on the y-axis represents the optimal hyperplane in a linear SVM classifier. Left: Erroneous examples from Tiny ImageNet are separated from correctly classified images using the hidden activations plus the sorted softmax features. Right: Erroneous examples from ImageNet are separated from correctly classified examples. To improve readability, I only include a subset of the correct and erroneous example sets.

Chapter 5 Further Investigation

I examined four additional scenarios in the following section: linear separation among erroneous example sets, high-softmax erroneous examples, left-out erroneous datasets, and finally corrupted examples with correct predictions. Each of the experiments show evidence of the beneficial properties of high-dimensional data, otherwise known as the 'Blessing of Dimensionality' [26]. See discussion in Section 7 for more details.

5.1. Discriminability of Erroneous Sets

Figure 3, which visualizes the separability of erroneous inputs and correctly processed inputs, indicates that the set of erroneous inputs occupy the same feature space. I investigated with an SVM, trained to separate erroneous inputs. Results, shown in Table 5, indicated that each erroneous input pair generally occupied a unique feature space, indicating that the strength of the method is the model's ability to separate correctly processed inputs from other inputs. This finding implies that the method will be robust against additional or future types of erroneous inputs.

Table 5: I test the separability of erroneous inputs on the CIFAR10 model with Maximum Softmax Probability and a Linear SVM. The linear SVM model is trained on the penultimate layer as well as the sorted softmax output, and contains 2058 features. The results indicate that each type of erroneous input manifests largely in it's own feature space. The experiments are performed with 5-fold cross validation, with error rates all below 0.015.

		MSP		Linear SVM	
Base	Second	AUROC	AUPR	AUROC	AUPR
Incorrect	FGSM	.634	.660	1.0	1.0
Corr.	Incorrect	.530	.530	1.0	1.0
FGSM	Corr.	.681	.641	.981	.983
Corr.	C-100	.538	.426	.935	.928
C-100	Incorrect	.522	.500	1.0	1.0
FGSM	C-100	.634	.612	.988	.989

5.2. High Softmax Erroneous Examples

I explored if the method was susceptible to high softmax erroneous examples. Specifically, I filter the CIFAR10 datasets to retrieve examples where the maximum softmax probability is greater than 0.99999. The filtering yields 342 erroneous examples from each type excluding PGD, which generally has a softmax closer to one than correctly classified examples. I filter out correctly predicted examples with the same method and perform an ROC measure on the MSP as well as a five-fold cross validation using a linear SVM. When tested, MSP has an AUROC of 0.513 and an AUPR of 0.532, or slightly better than random. Performing five-fold cross validation using the method, I achieve an AUROC of 0.922 and an AUPR of 0.935, with error rates of 0.026 and 0.014 respectively.

5.3. Leave One Erroneous Set Out

To verify robustness of the method, I use leave-one-out modeling, where I train a linear SVM on each erroneous set minus a single left out dataset, and then test the model on the left out dataset. I hypothesized that if the data occupies the same feature space, the model's hyperplane would be able to discriminate new erroneous sets. In the experiments, I use one dataset from each erroneous group from the CIFAR10 model: the misclassification dataset, FGSM attacks, SVHN dataset, and theCIFAR10-C dataset. Specifically, I train a linear SVM to classify correct inputs against three out of the four erroneous datasets. Then, I test the model on the fourth erroneous set. Results in Table show evidence that the model will be able to generalize to new types of bad data: the model with SVHN left out did the best, with an AUROC of .987 and AUPR of 0.988. FGSM struggled more, with an AUROC of .822 andAUPR of .817. Corrupted scored 0.952 AUROC 0.958 while incorrect examples scored 0.863/0.855.

5.4. Corrupted Examples

A last experiment I performed was separating corrupted examples with correct predictions -- $f(C)=Y_{actual}$ from erroneous examples. I used five-fold cross validation on the CIFAR10 dataset. Experiments illicit strong results, with AUROCs between 0.956 and 1 and AUPRs from 0.955 to 1. Interestingly, the dataset that performed the worst was incorrect corrupted images, with AUROC 0.956, indicating the model had most trouble discriminating similarly corrupted images.

Chapter 6 Real-World Use Case

To promote potential use of this model, I describe a simple production scenario where this algorithm could be used. An autonomous vehicle relies on a legacy machine learning algorithm f(x) to determine its course of action, where x is the video input from the vehicle's camera system. Specifically, f(x) outputs a vehicle's next course of action (e.g. turning the wheel, stopping, etc.) based on visual cues from the road ahead. The model was originally been trained to be robust against erroneous inputs by a method such as Outlier Exposure [23]; however, the model still fails in certain scenarios such as:

1) Ambiguous, fooling images that the model misclassified, such as the dog behind a bush pictured in Figure 1. A realistic example could be a child in the road ahead, obstructed by another vehicle.

2) The model receives corrupted input data, such as when the weather is poor, and the camera system has a distorted view.

3) The model receives an input which it has never seen before, such as an exotic animal.

4) New adversarial attacks on the camera software meant to fool the model.

During testing and use of the autonomous driving software, engineers have flagged various erroneous inputs where f(x) fails. These inputs are compiled into a dataset, and the engineers follow our methods to train an SVM to delineate bad examples from inputs where f(x) performs well. The resulting model can be quickly deployed to alert drivers when new erroneous inputs are present and the driver needs to take control of the vehicle. This quick corrective mechanism adds safety measures to an existing production software.

Chapter 7 Discussion

Erroneous inputs have been largely studied as distinct phenomena, with the detection of different faulty inputs requiring their own methods. My results show that these faulty inputs can be broadly detected by considering a model's internal behavior. I propose a new internal activation combination that allows for the broad detection of erroneous inputs, and establish standards for the detection of faulty data including adversarial, corrupted, out-of-distribution, and misclassified examples. A substantial benefit to this approach is that it requires no modification to the baseline model. Apart from Hendrycks and Gimpel [22], most work in the area requires some sort of network modification during training or test time. There are limitations to this approach that future research will need to address; first, the current research focuses on the image domain. It remains to be seen how well this technique will generalize to other domains (like natural language processing), but initial work by Hendrycks and Gimpel [22] found MSP, which my method directly builds on, was useful in several other domains, indicating the techniques will likely generalize. Second, the method requires erroneous detection models to be trained for individual neural models, with activation data from those models, which can be cumbersome, albeit arguably trivial when compared with the cost of a neural architecture search [27]. Third, my erroneous detection models are linear SVMs, which do fail under some conditions – for example, adversarial PGD inputs had higher maximum softmax probabilities, while other adversarial inputs had lower maximum softmax probability. Still, my models were highly successful at detecting erroneous inputs, and the simplicity of the model has important implications for real-time usage.

I believe that the positive outcomes in this paper result from the beneficial properties of high-dimensional data. Contrary to the popular 'Curse of Dimensionality' [2], which argues that

problems become more difficult in high-dimensions, Kainen coined the term ’Blessing of Dimensionality’ describing scenarios in which complex data is more beneficial [26]. Stochastic separation theorems, recently introduced by Gorban and Tyukina [20], formally established this phenomena, showing that in moderately high-dimensions we can achieve linear separability of sets with probability close to 1. Further work by the researchers presented a similar experiment to our own [19] by using LDA to discriminate anomalies from correct data on a simple dataset. These results are consistent with our experiments in the preceding sections.

A model capable of interpreting the breadth of all inputs is at best years away, and thus detection of ’bad’ data, which will cause models to fail, is essential for the safe use of any real world system. I argue that by moving towards detection of bad data in the broader application of a learning based system, we can advance the reliability of machine learning models in real world applications.

Bibliography

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mane. Concrete problems in AI safety. arXiv:1606.06565 [cs], 2016.

[2] Richard Bellman. Dynamic programming and stochastic control processes. Information and Control, 1(3):228–239,1958.

[3] Abhijit Bendale and Terrance Boult. Towards open set deep networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1563–1572, 2016.

[4] Petra Bevandic, Ivan Kreso, Marin Orsic, and Sinisa Segvic. Discriminative out-of-distribution detection for semantic segmentation. arXiv preprint arXiv:1808.07703, 2018.

[5] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pages 3–14, 2017.

[6] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. 2017 IEEE Symposium on Security and Privacy (SP), pages 39–57, 2017.

[7] Mohamed Chaabane, Ameni Trabelsi, Nathaniel Blanchard, and Ross Beveridge. Looking ahead: Anticipating pedestrians crossing with future frames prediction. In The IEEE Winter Conference on Applications of Computer Vision, pages 2297–2306, 2020.

[8] C.K. Chow. On optimum recognition error and reject trade-off. IEEE Transactions on information theory, 16(1):41–46,1970.

[9] C. K. Chow. An optimum character recognition system using decision functions. IRE Transactions on Electronic Computers, EC-6(4):247–254, 1957.

[10] Terrance DeVries and Graham W. Taylor. Learning confidence for out-of-distribution detection in neural networks. arXiv:1802.04865 [cs, stat], 2018.

[11] Samuel Dodge and Lina Karam. A study and comparison of human and deep learning recognition performance under visual distortions. 2017 26th International Conference on Computer Communication and Networks (ICCCN), pages 1–7, 2017.

[12] Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. Advances in Neural Information Processing Systems 31, page 1178–1187, 2018.

[13] Nic Ford, Justin Gilmer, Nicolas Carlini, and Dogus Cubuk. Adversarial examples are a natural consequence of test error in noise. Proceedings of the 36th International Conference on Machine Learning, 2019.

[14] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. Advances in Neural Information Processing Systems 30 (NIPS 2017), page 4878–4887, 2017.

[15] Yonatan Geifman and Ran El-Yaniv. SelectiveNet: A deep neural network with an integrated reject option. Proceedings of the 36th International Conference on Machine Learning, pages 2151–2159, 2019.

[16] Robert Geirhos, Carlos R. M. Temme, Jonas Rauber, Heiko H. Schutt, Matthias Bethge, and Felix A. Wichmann. Generalisation in humans and deep neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems 31, pages 7538–7550. Curran Associates, Inc., 2018.

[17] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S.Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. The International Conference on Learning Representations (ICLR) 2018 Workshop,2018.

[18] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. The International Conference on Learning Representations (ICLR) 2015, 2015.

[19] Alexander N. Gorban, Valery A. Makarov, and Ivan Y.Tyukin. High–dimensional brain in a high-dimensional world: Blessing of dimensionality. Entropy, 22(1):82, 2020.

[20] A. N. Gorban and I. Y. Tyukin. Stochastic separation theorems. Neural Networks, 94:255–259, 2017.

[21] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. The International Conference on Learning Representations (ICLR) 2019, 2019.

[22] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. The International Conference on Learning Representations (ICLR), 2017.

[23] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. The International Conference on Learning Representations (ICLR) 2019, 2019.

[24] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. arXiv:1907.07174 [cs, stat], 2020.

[25] Stanford Johnson. Tiny imagenet visual recognition challenge. 2015.

[26] Paul C. Kainen. Utilizing geometric anomalies of high di-mension: When complexity makes computation easier. Miroslav Ќarńy and Kevin Warwick, editors, Computer Intensive Methods in Control and Signal Processing: The Curse of Dimensionality, pages 283–294. Birkh ¨auser, 1997.

[27] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Advances in neural information processing systems., pages 211–252, 2009.

[28] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. International Conference on Learn-ing Representations (ICLR) 2018, 2018.

[29] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. International Conference on Learning Representations (ICLR) 2018, 2018.

[30] Si Liu, Risheek Garrepalli, Thomas G. Dietterich, Alan Fern,and Dan Hendrycks. Open category detection with PAC guarantees. Proceedings of the 35th International Conference on Machine Learning, 2018.

[31] Jiajun Lu, Theerasit Issaranon, and David Forsyth. SafetyNet: Detecting and rejecting adversarial examples robustly. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017.

[32] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. The International Conference on Learning Representations (ICLR) 2018, 2018.

[33] Saeed Mahloujifar, Dimitrios I. Diochnos, and MohammadMahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. In Proceedings of the AAAI Conference on ArtificialIntelligence (Vol. 33), pages 4536–4543, 2018.

[34] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. The International Conference on Learning Representations(ICLR), 2017.

[35] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos,Evgenia Rusak, Oliver Bringmann, Alexander S. Ecker, Matthias Bethge, and Wieland Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 8828–8838, 2019.

[36] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: a simple and accurate method to fool deep neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2574–2582, 2016.

[37] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011, 2011.

[38] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zant-edeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. Adversarial robustness toolbox v1.0.1. arXiv:1807.01069 [cs.LG], 2018.

[39] Tianyu Pang, Chao Du, Yinpeng Dong, and Jun Zhu. Towards robust detection of adversarial examples. In S. Bengio,H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi,and R. Garnett, editors, Advances in Neural Information Processing Systems 31, pages 4579–4589. Curran Associates, Inc., 2018.

[40] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. 2016 IEEE European Symposium on Security and Privacy (EuroSP), pages 372–387, 2015.

[41] Brandon RichardWebster, Samuel E. Anthony, and Walter J. Scheirer. Psyphy: A psychophysics driven evaluation framework for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI), 41(9), September 2019.

[42] Andras Rozsa and Terrance E. Boult. Improved adversarial robustness by reducing open space risk via tent activations. arXiv:1908.02435 [cs], 2019.

[43] Andras Rozsa, Manuel Gunther, and Terrance E. Boult. Adversarial robustness: Softmax versus openmax. Proceedings of the British Machine Vision Conference 2017, 2017.

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge.International Journal of Computer Vision (IJCV),115(3):211–252, 2015.

[45] Walter Scheirer, Anderson Rocha, Archana Sapkota, and Terrance Boult. Toward open set recognition. IEEE transactions on pattern analysis and machine intelligence, 35:1757–72, 07 2013.

[46] Lukas Schott, Jonas Rauber, Matthias Bethge, and WielandBrendel. Towards the first adversarially robust neural network model on mnist. The International Conference on Learning Representations (ICLR) 2018, page 16, 2018.

[47] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, JoanBruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. arXiv:1312.6199[cs], 2014.

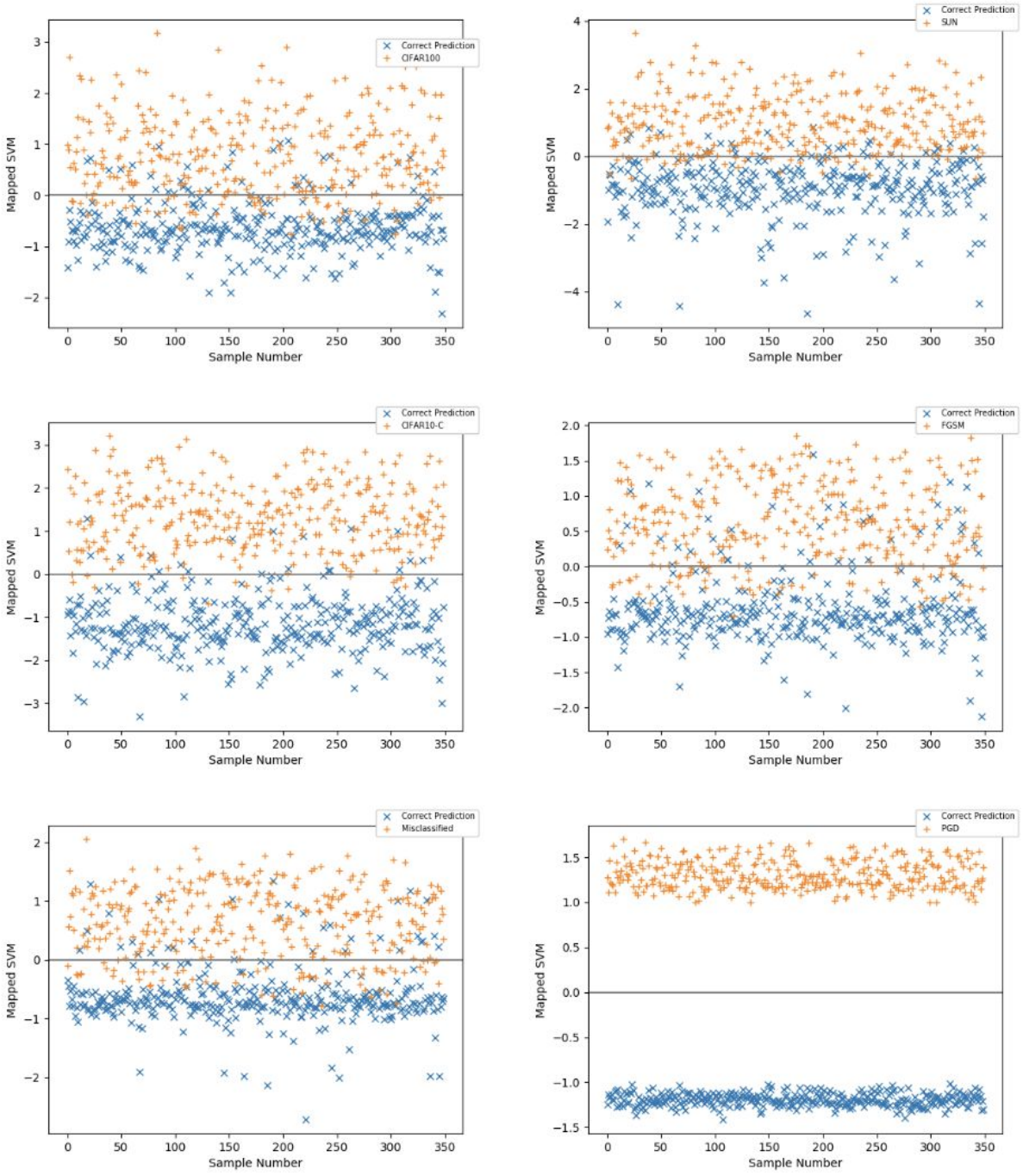
[48] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva,and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 3485–3492. IEEE, 2010.

[49] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. Proceedings of the British Machine Vision Conference 2017, pages 87.1–87.12, 2017.

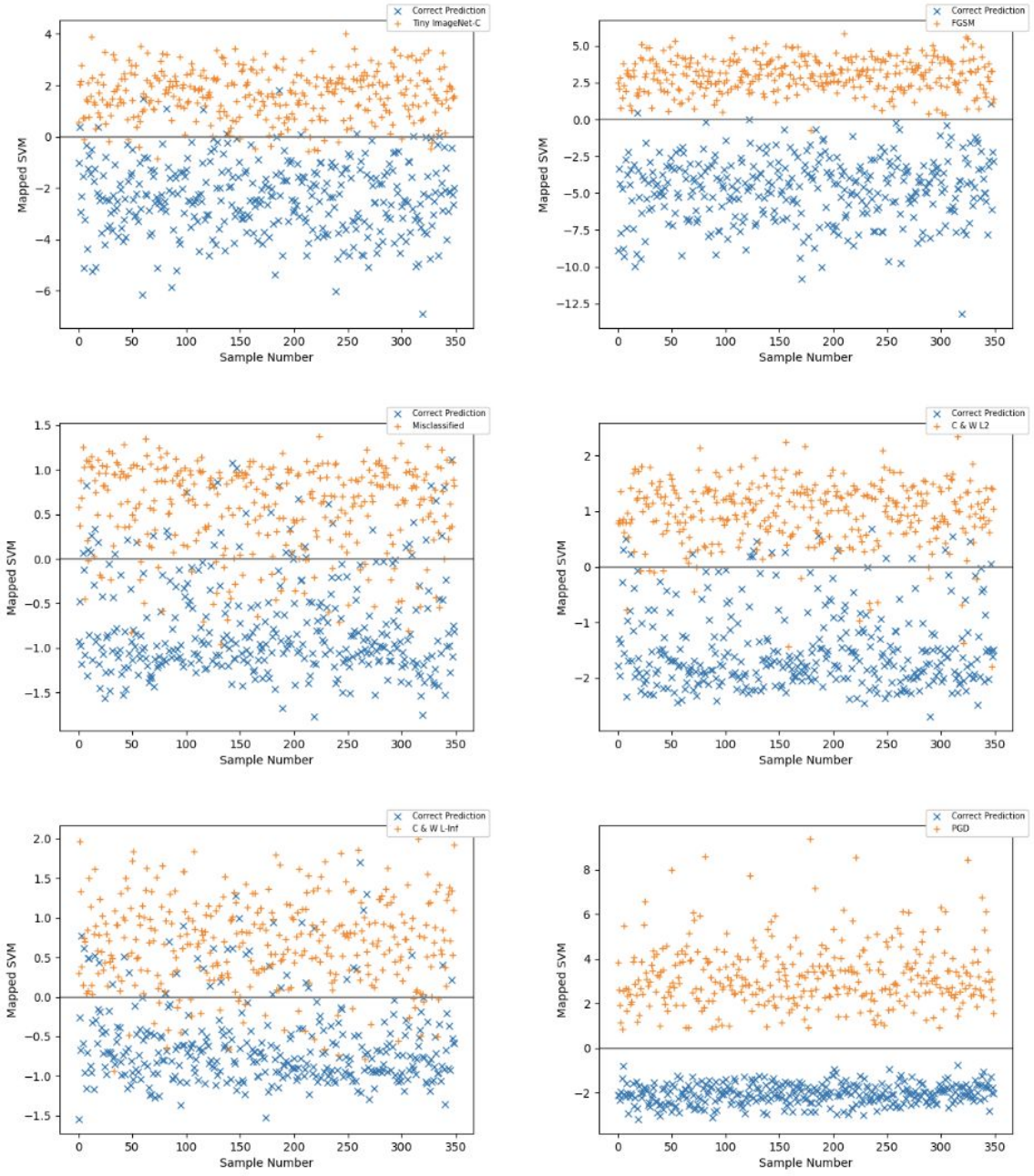
[50] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva,and Antonio Torralba. Places: A 10 million image database for scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(6):1452–1464, 2018.

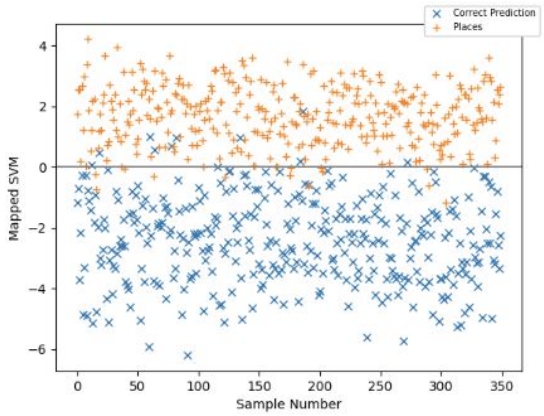
Appendix

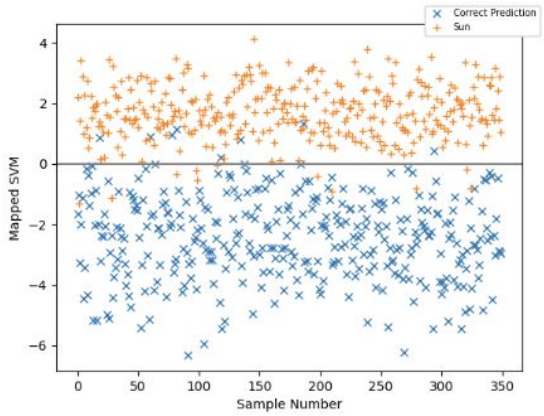
CIFAR10 Visualizations Hyperplane visualizations for the CIFAR10 dataset. Each image represents the hyperplane separating a single erroneous example set, denoted with orange “+” signs, from the correct example set, denoted with blue “x” signs.



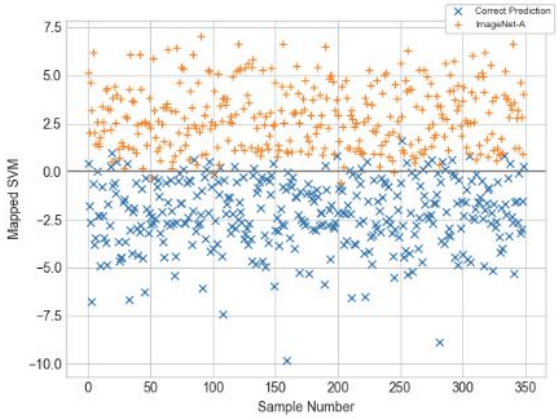
Tiny ImageNet Visualizations Hyperplane visualizations for the Tiny ImageNet dataset. Each image represents the hyperplane separating a single erroneous example set, denoted with orange “+” signs, from the correct example set, denoted with blue “x” signs.

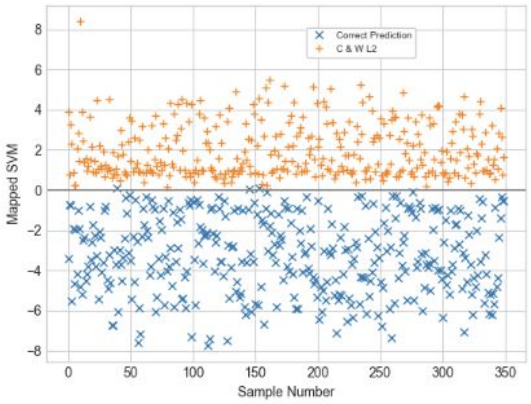


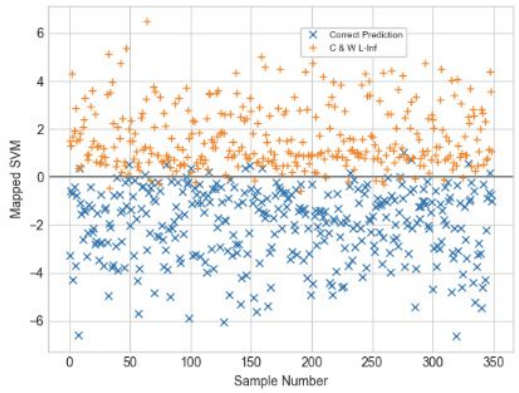


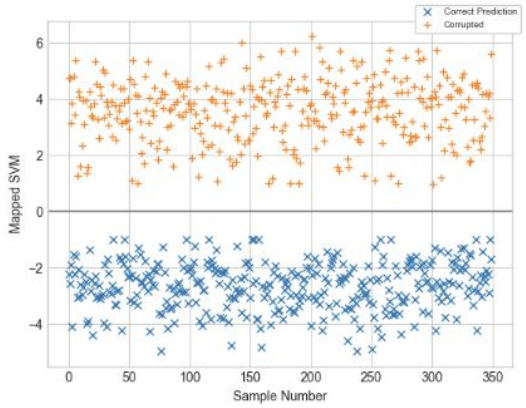


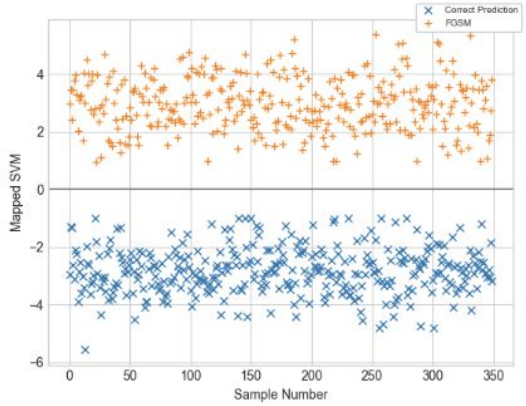
ImageNet Visualizations Hyperplane visualizations for the ImageNet dataset. Each image represents the hyperplane separating a single erroneous example set, denoted with orange "+" signs, from the correct example set, denoted with blue "x" signs.

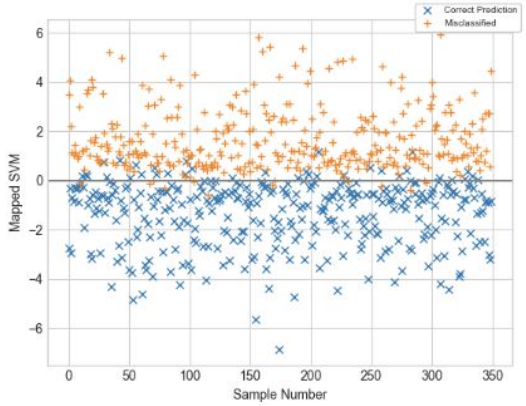


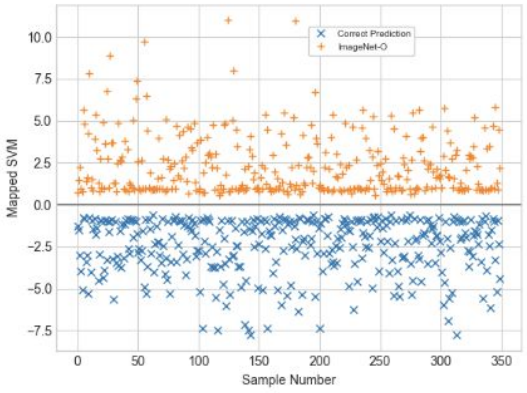


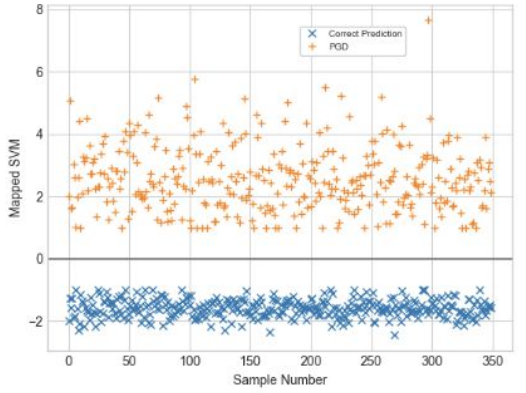

Correct Prediction
C & W L-Inf
Mapped SVM
Sample Number


Correct Prediction
Corrupted
Mapped SVM
Sample Number


Correct Prediction
FGSM
Mapped SVM
Sample Number


Correct Prediction
Misclassified
Mapped SVM
Sample Number


Correct Prediction
ImageNet-O
Mapped SVM
Sample Number


Correct Prediction
PGD
Mapped SVM
Sample Number