

# Utilizing Network Features to Detect Erroneous Inputs

Matt Gorbett  
 Colorado State University  
 matt.gorbett@colostate.edu

Nathaniel Blanchard  
 Colorado State University  
 nathaniel.blanchard@colostate.edu

## Abstract

*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 we show that a collective set of inputs with variegated data quality issues can be jointly identified with a single model. Specifically, we train a linear SVM classifier to detect four types of erroneous data using the hidden and softmax feature vectors of pre-trained neural networks. Our results indicate that these faulty data types generally exhibit linearly separable activation properties from correctly processed examples. We are able to identify erroneous inputs with an AUROC of 0.973 on CIFAR10, 0.957 on Tiny ImageNet, and 0.941 on ImageNet. We experimentally validate our findings across a diverse range of datasets, domains, and pre-trained models.*

## 1. Introduction

Humans are capable of adapting to diverse types of data in ways machine learning models cannot [15]. 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 [10, 20], adversarial examples [45], and previously unseen classes [3, 22, 44].

Figure 1 shows an example of each type of erroneous input: image corruptions [10, 20], previously unseen classes [3, 22, 44], adversarial examples [45], and misclassifications, an inevitable part of any model since no model has perfect performance. 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.

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 in-



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 we show that all erroneous inputs can be jointly identified with a linear SVM trained on a given network’s activations.

creasingly ubiquitous deployment of deep learning models in real-world applications such as autonomous vehicles and medical devices.

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) [9, 22, 27, 28, 44] (additional related work on erroneous input detection is discussed in Section 2). In our study of these seemingly disparate phenomena we noticed a common theme: the network’s internal activation patterns were notably distinct when stimuli were *correctly* processed. We hypothesized that erroneous inputs could be collectively separated from correctly processed inputs with a linear SVM trained on these features.

We test our hypothesis with three 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, we find that our detection technique is robust to four types of adversarial attacks. To our knowledge, this work is the first to consider the feasibility of broadly detecting four distinct types of erroneous inputs. Specifically, we find that the model’s internal activations in the softmax layer and final hidden layer are sufficient for automatically

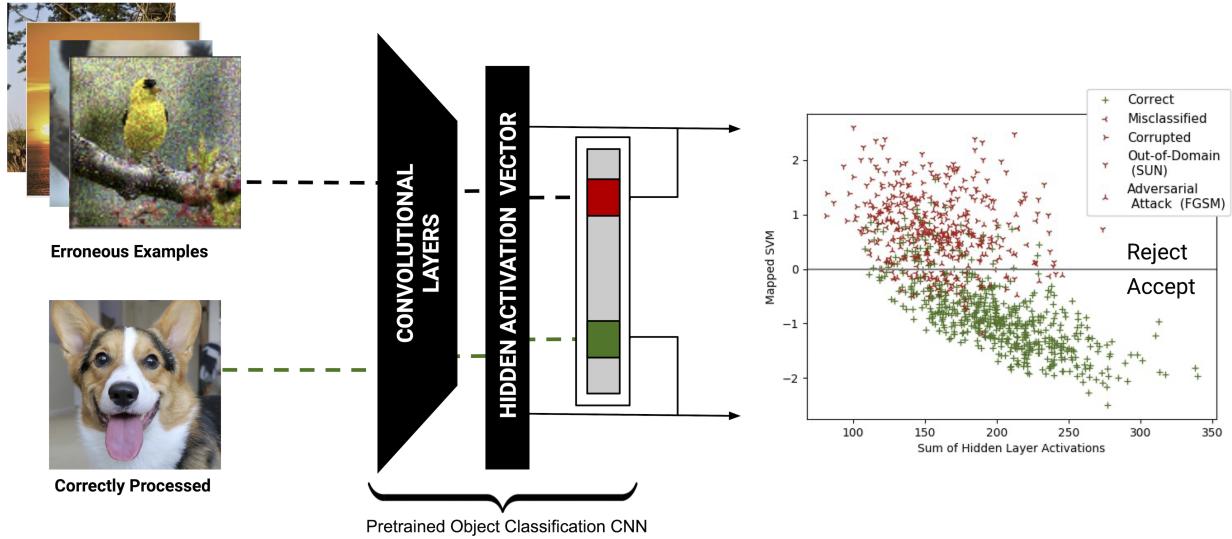


Figure 2. Erroneous inputs cause neural network’s to incorrectly classify images. However, these inputs can be preemptively identified with a simple linear SVM that is trained on the network’s internal activations (i.e., features) that occur in response to the image. Specifically, the SVM is trained 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.

filtering incorrectly processed inputs, as shown in Figure 2

In summary, we make the following contributions:

- This is the first research showing that the four distinct types of erroneous inputs (see 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.
- We achieve results comparable to state-of-the-art techniques by considering the activations of both the softmax and final hidden layer of common pre-trained neural networks.
- We show how this method can be used as an auxiliary technique in existing anomaly detection models to enhance performance.

## 2. Related Work

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

**Image Corruption** Standard neural networks perform poorly on corrupted examples. Image corruption is an unintentional side-effect of any visual system: 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 [10, 20]. In real-world scenarios, like autonomous driving, models need to be robust to corruption from weather and debris [6, 34]; some research even proposed that networks should be explicitly evaluated on their robustness to corruption [40].

**Out-of-Distribution Inputs (OoD)** An OoD example occurs when a model is presented with data outside of its training paradigm. In this work, we defined OoD examples as originating from datasets other than the dataset the baseline model was trained on, although we controlled for class ambiguity (for example, if a dataset was trained on dogs, we controlled for wolves in OoD data).

Hendrycks and Dietterich [21] established a baseline for OoD detection by thresholding 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 [9], auxiliary datasets [22], GAN examples [27], and other train-time techniques [4, 14]. A notable cross-domain technique was proposed by Liang *et al.* [28], who used adversarial perturbations on input data to discriminate in-distribution from OoD examples.

Further, Bendale and Boult proposed OpenMax [3], a network layer that uses the penultimate activation vector of neural networks and propose OpenMax, a network layer to estimate the probability of being from an unknown class using Meta Recognition. 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 our ImageNet-O set referenced section 3.2.2. Additional work by Rozsa et. al. [42] proposed an adversarial attack against OpenMax, which we describe in Section 2.

**Adversarial Attacks** Adversarial examples occur when small, meaningful changes to clean data alter a network’s prediction. Szegedy *et al.* [45] highlighted these limitations, giving rise to the field of adversarial attacks. Since then, numerous methods have been proposed to generate adversarial attacks [5, 17, 35, 39]. Several works have attempted to defend against adversaries [30, 33], 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).

Researchers have identified the upper bounds of adversarial robustness techniques for certain distributions [11, 16, 32]. In particular, Fawzi *et al.* [11] suggested a strong relation between adversarial robustness and the linearity of a classifier in latent space. Gilmer *et al.* [16] 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.

Our work finds that adversarial inputs, generated using a variety of popular methods, manifest linearly separable activation’s we can use to detect when models will fail. While we test our model on a variety of adversarial methods, attacks on the penultimate layers features will likely cause our model to fail. One example of this is LOTS [42], an algorithm designed specifically to break OoD method OpenMax. The algorithm modifies an image in order to mimick the features of the OpenMax.

**Misclassified Examples and Thresholding** Misclassified examples occur when a model fails to predict an instance of a class, despite the model being trained to predict that class. Thresholding is a common technique to detect misclassified data, and has a rich history of research [7, 8], however, we believe we are the first to consider multiple layers of the network for thresholding.

Hendrycks and Dietterich [21] provide thresholding results for misclassified examples on contemporary datasets, while [13] presented results of the softmax response from a risk perspective. In recent years, thresholding techniques have been applied to detect adversarial examples [30, 38] and OoD examples [21]. Lu *et al.* [30] provide the approach most similar to our own, utilizing layers toward the end of deep learning models to detect adversarial examples with SVM-RBF classifiers. Our work also shows that some ad-

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

**Connecting Erroneous Inputs** Recent research has found similarities in the above areas. Hendrycks and Gimpel [21] showed that both misclassified and out-of-distribution examples can be detected by utilizing the maximum softmax probability. Ford *et al.* [12] 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.* [28], who used adversarial perturbations on input data to discriminate in-distribution from OoD examples. Finally, Rozsa and Boult [41] 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 our knowledge, our work is the first to fully explore the link between all of these areas, and the first to detect them with a single approach.

### 3. Method

We 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 experiments. Finally, we define the methods for training and testing (Section 3.3), evaluation metrics (Section 3.4), and other details (Section 3.5) such as dataset sizes.

#### 3.1. Defining the Domain

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

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

To build the training and tests set for our binary classifier we use validation and test sets from the dataset under test. Training data is completely disjoint from our test data. Details in Section 3.5. Our training set was built such that

$f(\mathcal{A}) \neq \mathcal{Y}_{actual}$  and  $f(\mathcal{C}) \neq \mathcal{Y}_{actual}$  — we only used samples which the pretrained neural network classified incorrectly. When an adversarial or corrupted example is classified correctly, we 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.  $f(x) = \mathcal{Y}_{actual}$  and  $f(x \rightarrow \mathcal{A}) \neq \mathcal{Y}_{actual}$ . Finally, it is implied that  $f(\mathcal{D}_{out})$  and  $f(\mathcal{M})$  would produce invalid results, so each example will be taken from these datasets.

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

### 3.2. Experimental Setup

For our experiments we 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 our detection models. The baseline models were not altered in any way during our experiments.

#### 3.2.1 Baseline Datasets and Models

Below are the summaries of the pre-trained models used for each 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 [26]. The dataset has 50,000 training images and 10,000 testing images. Our CIFAR-10 model was trained using the ResNet50 architecture and open-sourced by [31].

**Tiny ImageNet** is a 200-class subset of the ImageNet dataset where images were cropped and resized to a resolution of 64 x 64. Bounding box information was used in the image cropping [24]. Our Tiny ImageNet model is a pre-trained WideResNet [47]. The trained model was open-sourced by [20].

**ImageNet** consists of 1000 classes of objects [43], with varied dimensions and resolutions. Our work used the validation dataset to produce examples for our detection model. We utilized the default pre-trained ResNet50 model in the Pytorch library for our ImageNet experiments.

#### 3.2.2 Out-of-Distribution Datasets

We use several commonly used OoD datasets:

**CIFAR-100** contains 32 x 32 colored images of 100 different classes of objects [26]. We filtered out classes similar to those found in CIFAR-10 to ensure all classes were truly OoD, leaving 74 classes. Details on excluded classes are in the appendix.

**SVHN** We use the test set from the Street View House Numbers dataset [36], which contains colored numbers with class labels 0 to 9.

**The Scene Understanding dataset (SUN)** contains images of scenes with varying resolutions [46].

**Places365** contains 365 classes of scenes with varying image resolutions. We use the high-resolution validation set for use in our Tiny ImageNet and ImageNet models [48].

**ImageNet-O** dataset was constructed from sampled images from ImageNet-22K [23]. First, images overlapping with classes in ImageNet-1K were filtered out. Next, they retained images that a ResNet50 ImageNet-1K 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 [20]. 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. We used each corruption and severity type in our models.

#### 3.2.4 Adversarial Attack Methods

We test our method on four different adversarial attacks, however, we do not claim this method to be robust to *any* attack or modified hyperparameters of the attacks used. Section 5 looks at this further. Further, we looked at the first 20 generated examples of each adversarial dataset to verify the perturbed images were realistic.

White-box attack methods are used to generate adversarial examples. White-box methods assume the attacker has full access to the learning model. Next we describe each attack used in our experiments.

**Fast Gradient Sign Method (FGSM)** was an attack introduced by Goodfellow et al. [7]. It adjusts the inputs to maximize the loss based on the back-propagated gradients of the predicted value:  $f(\mathcal{A}) = \mathcal{X} + \epsilon sign(\nabla_x J(\theta, \mathcal{X}, \mathcal{Y}))$ . FGSM examples were generated using  $\epsilon=0.01$ .

**Carlini and Wagner  $l_2$  Attack (C & W  $l_2$ )** searches for low distortion in the  $l_2$  metric [5]. Notable in the Carlini and Wagner attacks is  $\kappa$ , 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 our main experiments were generated with  $\kappa = 0$  — we did not search for a perturbed input which satisfied a specific confidence.

**Carlini and Wagner  $l_\infty$  Attack (C & W  $l_\infty$ )** was a modified version of the C&W  $l_2$  attack. It controls the  $l_\infty$  norm, i.e. the maximum perturbation applied to any pixel [5].

**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  $\epsilon$  while also clipping values so they fall within a permitted range [31].

Carlini and Wagner  $l_2$ ,  $l_\infty$  attacks and PGD attack examples were generated using the adversarial-robustness-toolbox with default parameters [37]. We 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 [23]. We included this dataset as an extension to our misclassified experiments.

### 3.3. SVM Model

We employed a linear Support Vector Machine (SVM) classifier for our experiments, where examples  $(\{x_1\}, y_1), \dots, (\{x_n\}, y_n)$  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 evaluation, we evaluate our detection model using five-fold cross validation. We found that linear SVM's were able to generalize better in high dimensions than other SVM kernels.

To generate features  $\{x_1\} \dots \{x_n\}$  for our SVM, we take values from the softmax output of the baseline model, sort them, and concatenate them onto the fully-connected activations from the penultimate layer. For ResNet50 pretrained models (CIFAR10 and ImageNet), the penultimate layer is 2048 values, while Tiny ImageNet has a penultimate layer of size 128.

We present results in this paper with a balanced dataset: each model contains the same number of correct and erroneous examples.

### 3.4. Testing and Evaluation Metrics

We evaluate binary detection tasks using three metrics: area under the receiver operating characteristic curve (AUROC), area under the precision-recall curve (AUPR), and false positive rate at N% true positive rate (FPR $\wedge$ ).

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

---

**Algorithm 1** SVM Training for our 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:  $|\{\mathcal{C}\}| = |\{\mathcal{D}_{in}, \mathcal{A}, \mathcal{C}, \mathcal{M}\}|$

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

---

(True Positive)/(True Positive + False Positive) versus the recall (True Positive)/(True Positive + False Negative).

For our ImageNet model we also present FPR $\wedge$  scores, also used by [22] [28] [29]. The FPR $\wedge$  calculates the false positive rate at a set True Positive Rate. We 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: our CIFAR10 pretrained model only misclassified 475 images on the test set, and TinyImageNet FGSM attacks rendered only 1,764 bad examples. Details in the appendix.

**Baseline detection models** We test our erroneous datasets on two baselines: MSP [21] and Outlier Exposure [22]. For MSP, we use the pretrained models as described in Section 3.2.1. We use the same pretrained models for our main experiments, the columns labeled "Linear SVM" in Tables 1 and 2. As an additional measure, we test the datasets on Outlier Exposure by using pretrained models open sourced by the researchers. 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, the models were trained with realistic OoD data from 80 Million Tiny Images dataset and ImageNet22k.

## 4. Results

We present our 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.

Table 1. CIFAR-10 and Tiny ImageNet experiments for MSP, Outlier Exposure, Outlier Exposure + Linear SVM (Ours), and Linear SVM (Ours). 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 [21]		OE [22]		OE+Linear SVM (Ours)		Linear SVM (Ours)	
<b>CIFAR-10</b>	<b>Detail</b>	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
Misclassified	-	.930	.905	.927	.908	.935	.910	<b>.939</b>	<b>.911</b>
Out-of-Distribution	Sun	.928	.912	.998	.998	<b>1.0</b>	<b>1.0</b>	.989	.989
Out-of-Distribution	CIFAR100	.925	.908	.992	.991	<b>1.0</b>	<b>1.0</b>	.970	.966
Out-of-Distribution	SVHN	.954	.937	.998	.996	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>
Corrupted	CIFAR10-C	.945	.926	.961	.959	.975	.975	<b>.982</b>	<b>.982</b>
Adversarial	FGSM	.830	.825	.998	.998	<b>1.0</b>	<b>1.0</b>	.980	.980
Adversarial	C & W $l_2$	.982	.973	.752	.747	.891	.891	<b>.991</b>	<b>.989</b>
Adversarial	C & W $l_\infty$	.916	.893	.817	.808	.910	.906	<b>1.0</b>	<b>1.0</b>
Adversarial	PGD	.999	.999	.989	.996	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>
<b>Combined</b>		.925	.915	.871	.908	<b>.973</b>	<b>.977</b>	.969	.970
<b>Tiny ImageNet</b>									
Misclassified	-	<b>.860</b>	<b>.834</b>	.846	.809	.820	.803	.847	.822
Out-of-Distribution	Sun	.876	.864	<b>.999</b>	<b>.999</b>	.999	.998	.994	.993
Out-of-Distribution	Places365	.882	.873	.993	<b>.999</b>	<b>.999</b>	.998	.992	.991
Corrupted	TinyImageNet-C	.895	.885	.996	.996	<b>.998</b>	<b>.997</b>	.994	.993
Adversarial	FGSM	.998	.999	.720	.699	.984	.983	<b>.999</b>	<b>.999</b>
Adversarial	C & W $l_2$	.908	.835	.823	.794	<b>.994</b>	<b>.994</b>	.967	.960
Adversarial	C & W $l_\infty$	.855	.782	.815	.785	<b>.995</b>	<b>.995</b>	.887	.858
Adversarial	PGD	.990	.991	.743	.718	.996	.998	<b>1.0</b>	<b>1.0</b>
<b>Combined</b>		.886	.870	.828	.863	<b>.957</b>	<b>.964</b>	.931	.933

#### 4.1. Single Erroneous Dataset Results

In Table 1 we compared Maximum Softmax Probability (MSP) [21] and Outlier Exposure (OE) [22] against our OE model and our linear SVM, focusing on two pretrained models: CIFAR10 and Tiny ImageNet. All results are validated with 5-fold cross validation. Results show that our methods achieve state-of-the-art results on all experiments except for misclassified inputs on the Tiny ImageNet model. By applying our method of using the hidden activations as well as the sorted softmax to linearly discriminate correct examples from erroneous examples, we are able to outperform OE and MSP on 15 of 17 datasets.

In Table 2 we present our results for ImageNet examples in a pretrained ResNet model. Our linear SVM was trained and evaluated with five-fold cross validation. In this case, we only evaluated our method against the baseline AUROC curve of the MSP. Across all of our tests, our models easily outperformed MSP.

#### 4.2. Joint Erroneous Example Detection

Section 4.1 showed how our linear SVM accurately filtered each individual erroneous input type from correctly processed inputs. In this section, we investigate the combined condition ('Combined' row in Tables 1 and 2), where

the full set of erroneous inputs are detected by a single linear model. Here, our experiments unequivocally show that, across all of our models, the full set of erroneous inputs could be detected with comparable performance to models trained to detect individual types of erroneous inputs (AUROCs between 0.941 and 0.973).

A manual analysis of model failures found we successfully identified erroneous inputs except examples from PGD attacks. For PGD, detection failed because inputs tended to have higher maximum softmax probabilities (PGD inputs had nearly 100% probability, on average) than the softmax probabilities of correctly classified examples (80% MSP average for ImageNet, 86% MSP average for Tiny ImageNet). Inversely, all other erroneous example sets had lower softmax probabilities (between 25% and 93% average). Users of this technique will need to be mindful of cases like this, since PGD examples will need their own model. However, such a model can easily separate correct examples, as shown in the PGD experiments in Tables 1 and 2.

#### 5. Further Investigation

We examined four additional scenarios in the following section: linear separation among erroneous example sets, high-softmax erroneous examples, left-out erroneous

Table 2. ImageNet pretrained model experiments for MSP as well as our Linear SVM model. Up arrows indicate when a higher score is better, while down arrows indicate when a lower score is better. OoD stands for out-of-distribution, Cor. stands for corrupted, and Adv. stands for adversarial attack. See appendix for all experimental results.

ImageNet		MSP [21]				Linear SVM (Ours)	
Test Type	Detail	AUROC $\uparrow$	AUPR $\uparrow$	FPR (95%) $\downarrow$	AUPR $\uparrow$	AUROC $\uparrow$	FPR (95%) $\downarrow$
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 $l_\infty$	.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
<b>Combined</b>		.881	.876	.472	.941	.947	.277

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*’ [25]. See discussion in Section 7 for more details.

**Discriminability of Erroneous Sets** We investigated whether erroneous datasets occupied the same feature space by training an SVM to linearly separate erroneous inputs. Our results, shown in Table 3, indicate that each erroneous set generally occupies a unique feature space in *high dimensions*. However, in lower dimensions (i.e., the single dimension MSP), the results were close to random. This underscores the importance of considering the latent vector in our experiments, and why previous works have only found links between particular pairs of erroneous input types, while we are able to jointly separate all erroneous inputs from correctly processed inputs.

**High Softmax Erroneous Examples** We explored if our method was susceptible to high softmax erroneous examples. Specifically, we filter our CIFAR10 datasets to retrieve examples where the maximum softmax probability is greater than 0.99999. Our filtering yields 342 erroneous examples from each type (we exclude PGD, as discussed in Section 4.2). MSP has an AUROC of .513 and an AUPR of .532, or nearly random. Our method achieved an AUROC of .922 ( $\pm .026$ ) and an AUPR of .935 ( $\pm .014$ ). Results positively indicate that the high-dimensional feature data performs substantially better than its single dimensional counterpart (MSP).

**Leave One Erroneous Set Out** Our experiments show we can successfully filter erroneous inputs from correctly processed ones, but there may be additional kinds of erroneous inputs that we have not tested here. To investi-

Table 3. We test the separability of erroneous inputs on our 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. Results indicate that each type of erroneous input manifests largely in its own feature space in high dimensions, but is not separable in lower dimensions. Our experiments are performed with 5-fold cross validation, with error rates all below 0.015.

BASE	SECOND	MSP		LINEAR SVM	
		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

gate how robust our method would be against a potential ‘unknown’ erroneous input, we evaluated our method in a leave-one-out context, where we train a linear SVM on each erroneous set minus a single left out erroneous input type, and then test the model on the left out type. A model capable of discriminating new erroneous example types has substantial benefits in a real-world use case.

In our experiments, we use one dataset from each erroneous group from our CIFAR10 model: the misclassification dataset, FGSM attacks, SVHN, and CIFAR10-C.

Our results indicate that our method will be able to generalize to new types of erroneous inputs, but at the cost of accuracy: most notably, unseen FGSM detection dropped to an AUROC of .822 and AUPR of .817, a substantial drop from .999 AUROC when FGSM inputs were part of train-

ing. Similiarly, unseen SVHN dropped slightly from an AUROC of 1.0 to .987 (AUPR = .988), Unseen corrupted were detected with .952 AUROC .958, compared with .998 AUROC originally. Interestingly, misclassified examples were detected at similar rate: .863 AUROC and .855 AUPR.

**Corrupted Example** Finally, we experimentally evaluated how effectively we could separate correctly processed corrupt examples ( $f(\mathcal{C}) = \mathcal{Y}_{actual}$ ) from erroneous examples. For this experiment, we used five-fold cross validation on our CIFAR-10 dataset. Experiments elicited strong results, with AUROC's between .956 and 1 and AUPR's 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. MSP, on the other hand, has an AUROC of 0.813 and 0.794 when discriminating corrupted examples with correct predictions from other erroneous predictions.

## 6. Real-World Use Case

To promote potential use of this model, we describe a simple production scenario for this algorithm:

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. 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 trained to be robust against erroneous inputs by a method such as OE [22], however, the model still fails in certain scenarios, such as: 1) Natural adversarial images that the model misclassifies, such as the dog behind a bush pictured in Figure 1. 2) The model receives corrupted input data, such as camera distortion in poor weather. 3) The model receives an input which it has never seen before, such as an exotic animal. 4) New adversarial attacks on the camera system 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.

## 7. Discussion

Erroneous inputs have been largely studied as distinct phenomena, however, our results show that these faulty inputs can be broadly detected by considering a model's in-

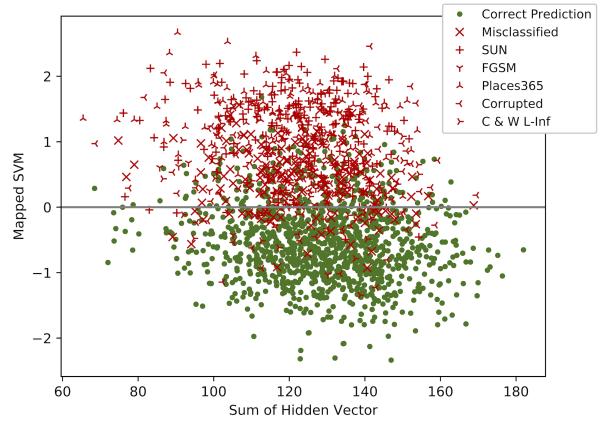


Figure 3. The hyperplane separating erroneous examples from correct examples in our Tiny ImageNet model. "0" on the y-axis represents the optimal hyperplane in a linear SVM classifier. To improve visualization, we only include a subset of the correct and erroneous example sets.

ternal behavior. We propose a new internal activation combination that allows for the broad detection of faulty data including corrupted, out-of-distribution, misclassified, and four types of adversarial attacks.

We 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 [25].

Stochastic separation theorems recently introduced by Gorban and Tyukina [19] 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 [18] 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. We 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.

## References

- [1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 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 Bevandić, Ivan Krešo, Marin Oršić, and Siniša Šegvić. Discriminative out-of-distribution detection for semantic segmentation. *arXiv preprint arXiv:1808.07703*, 2018.

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

[6] 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.

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

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

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

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] Robert Geirhos, Carlos R. M. Temme, Jonas Rauber, Heiko H. Schütt, 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.

[16] 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.

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

[18] 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.

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

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

[21] 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*.

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

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

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

[25] Paul C. Kainen. Utilizing geometric anomalies of high dimension: When complexity makes computation easier. In Miroslav Kárný and Kevin Warwick, editors, *Computer Intensive Methods in Control and Signal Processing: The Curse of Dimensionality*, pages 283–294. Birkhäuser, 1997.

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

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

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] Saeed Mahloujifar, Dimitrios I. Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning:

Evasion and poisoning attacks from concentration of measure. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33), pages 4536–4543, 2018.

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

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

[39] 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.

[40] 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.

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

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

[43] 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.

[44] 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.

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

[46] 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.

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

[48] 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.