Improving the Efficiency of Robust Generative Classifiers

Alan Rosenthal

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-68
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-68.html

May 13, 2021

Acknowledgement

I would like to thank Prof. David Wagner for his mentorship throughout this program as my advisor. I am also grateful to have worked with An Ju, whose insights were invaluable for this project. Finally, I want to thank my family for their continual guidance and support every step of the way.

Improving the Efficiency of Robust Generative Classifiers

by Alan Rosenthal

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor David Wagner
Research Advisor

5/12/2021

(Date)

* * * * * * *

Professor Dawn Song
Second Reader

5/6/2021

(Date)

Improving the Efficiency of Robust Generative Classifiers

by

Alan Rosenthal

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor David Wagner, Chair
Professor Dawn Song

Spring 2021

Abstract

Improving the Efficiency of Robust Generative Classifiers

by

Alan Rosenthal

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor David Wagner, Chair

The phenomenon of adversarial examples in neural networks has spurred the development of robust classification methods that are immune to these vulnerabilities. Classifiers using generative models, including Analysis by Synthesis (ABS) introduced by Schott et al. and its extension E-ABS by Ju et al., have achieved state-of-the-art robust accuracy on several benchmark datasets like SVHN and MNIST. Their inference time complexity, however, scales linearly with the number of classes in the data, limiting their practicality. We evaluate two approaches to speed up ABS-style models and inference: first, a hierarchical decision tree framework that achieves accuracy nearly on par with E-ABS in logarithmic time, and second, a scheme to classify latent vectors based on a prior distribution in constant time. We also provide an algorithm to search over decision tree structures, which yields significant improvements in accuracy over naive arrangements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Neural Networks and Adversarial Robustness

Deep neural networks have made sweeping advances in a variety of tasks [21, 16]. However, they suffer from a phenomenon known as *adversarial examples*: minuscule perturbations added to inputs that can significantly change a network's behavior on those inputs [38, 34]. This work considers classification, the most frequently discussed setting for adversarial examples, in which an adversarial perturbation causes the classifier network to predict the wrong class, often with high confidence. Adversarial examples highlight the qualitative differences between neural networks and humans in information processing. Moreover, they undermine the reliability and trustworthiness of neural networks that are deployed in real-world applications.

Researchers have developed a variety of "attacks" to find adversarial perturbations [13, 24, 29]. Given a classifier and an input, most attacks have the constraint that the perturbation to the input is L_p bounded by a certain radius in pixel space.

Likewise, many "defenses" have been proposed that attempt to classify accurately without such vulnerabilities [26, 4]. Many of these defenses were subsequently defeated by stronger attacks [39, 2, 3], underscoring the difficulty of empirically evaluating robustness. Theoretically-guaranteed defenses [23, 14] are especially valued, but have lower accuracy than methods with only empirical assurances, such as adversarial training [26, 4].

1.2 ABS

Schott et al. introduced an adversarially robust classifier based on generative models called Analysis by Synthesis (ABS) [36]. ABS empirically achieved state-of-the art robust accuracy on MNIST [22] under arbitrary L_p-bounded perturbations. Additionally, adversarial examples found for ABS are more effective at deceiving humans than those found for other defense methods [12].

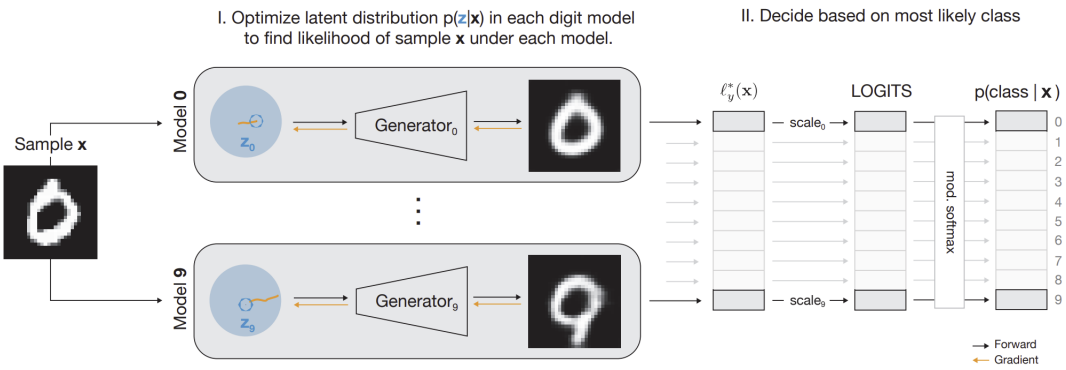

Figure 1.1: Module structure of ABS. Image taken from Schott et al. [36]

For a dataset with K classes, ABS uses K generative models, one per class to learn the conditional distribution of that class. Given an input at inference time, each of the K generative models estimates the likelihood of the input under that class, and the most likely choice of class is returned as the classification prediction.

A Brief Overview of VAEs

Each of the K classes in ABS is modeled by a class-conditional Variational Autoencoder (VAE) [19, 11]. See Figure 1.1. Consider a single class $y \in \{1, \ldots, K\}$. According to the VAE model:

- The distribution of the latent vector Z is $\mathcal{N}(0, I)$, independently of class.
- Given a latent vector Z and a class y, the distribution of data points X is a multivariate Gaussian with mean $dec_y(Z)$ and diagonal covariance $\tau^2 I$, where τ^2 is a hyperparameter and $dec_y(Z)$ is the output of class y's VAE decoder on Z.

At inference time, our prediction is the most likely class y given the input X:

$$\begin{aligned} \arg\max_{1\le y\le K} p(y|X) &= \arg\max_{1\le y\le K} p(X|y)\ p(y) && \text{(Bayes' Rule)} \\ &= \arg\max_{1\le y\le K} p(X|y) && \text{(assuming uniform prior on classes)} \end{aligned}$$

Since $p(X|y) = \int p_y(X|Z)\ p(Z)\ \mathrm{d}z$ generally is intractable to estimate, we use the evidence lower bound (ELBO) [8, 19], denoted ℓ_y^q:

$$\begin{aligned} \log p(X|y) &\ge \mathbb{E}_{Z\sim q}[\log p_y(X|Z)] + D_{KL}\big(q(Z)\ ||\ p(Z)\big) \\ &= \ell_y^q \end{aligned}$$

Here, the variational distribution $q(Z)$ is $\mathcal{N}(\mu, \sigma^2 I)$, where σ^2 is a hyperparameter and the mean μ is free to vary. [1] Since the ELBO can be written in terms of μ, this parameter can be optimized to maximize the ELBO. [2] Let $\ell_y^* = \max_q \ell_y^q$ be the best ELBO to $\log p(X|y)$. By performing the optimization once for each of the K classes, we arrive at the classification output $\arg\max_{1 \le y \le K} \ell_y^*$.

The optimization procedure begins by sampling many latent vectors from the prior, then choosing the one with highest ELBO as the starting point for optimization.

Schott et al. use this framework to derive a lower bound of the robustness of ABS, which depends on the learned class-conditional data distributions. Their result relies on the strong assumption that the optimization procedure finds a global optimum. This may be largely true for simple datasets like MNIST, where the latent space can have low dimensionality, but becomes less certain when the data representations have more dimensions.

1.3 E-ABS

Ju et al. [18] introduced E-ABS, which modifies some of the model structure and training regime of ABS to tackle more complex datasets than MNIST. See Figure 1.2. E-ABS achieved state-of-the-art robust accuracy on SVHN [31] and a 10-class traffic sign dataset. Their modifications were:

- Use Adversarial Autoencoders (AAEs) [27] instead of VAEs as the class-conditional generative models, since VAEs have difficulty matching the imposed latent prior distribution [35] and providing accurate likelihood estimates [6]. In an AAE, a separate discriminator network learns to distinguish vectors sampled from the true latent prior from latent vectors produced by the encoder. The discriminator's loss term replaces the KL divergence in the ELBO.

- Use a single encoder and discriminator, which are shared by all K classes.

- At inference time, optimize the component variances of the variational distribution $q(Z)$ in addition to the mean. In contrast, ABS held the variational distribution's covariance constant during optimization. Also, the encoder's output is considered in addition to the initial samples as a possible starting point for optimization.

- Introduce a discriminative loss during training, treating the values $(\ell_1^*, \ldots, \ell_K^*)$ as logits and using a cross-entropy loss with respect to the true class.

[1] During training, the variational distribution $q(Z)$ is parametrized by a mean and diagonal covariance, where the mean and component variances are output by a neural network encoder.

[2] Since $p(Z)$ is also multivariate Gaussian, there is a closed-form expression for $D_{KL}\big(q(Z) \,||\, p(Z)\big)$ in terms of μ. With the Gaussian reparametrization trick [19], an estimate for the term $\mathbb{E}_{Z \sim q}[\log p_y(X|Z)]$ in terms of μ can found by sampling. If $q(Z)$ concentrates around the latent vectors that reconstruct X well, then the sampling is efficient.

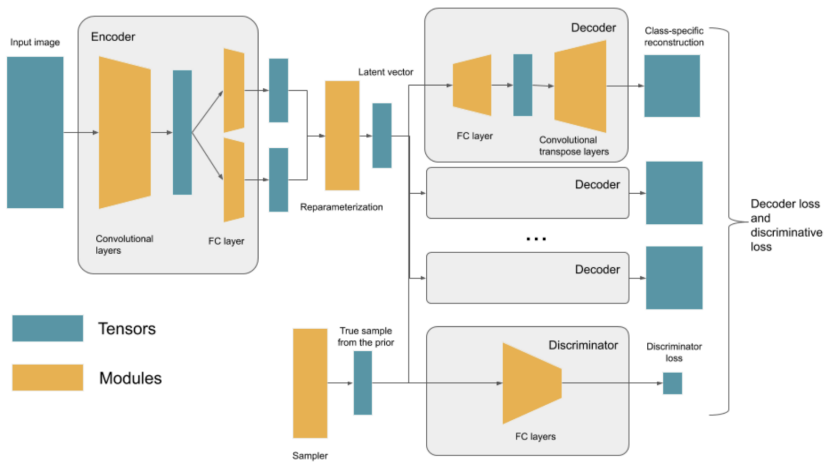

Figure 1.2: Module structure of E-ABS. Image taken from Ju et al. [18]

Ju et al. [18] also derive a similar lower bound of the robustness of E-ABS as Schott et al. [36] derive for ABS, with similar caveats.

1.4 Challenges

Despite good performance on simpler datasets, ABS-style models have two major shortcomings.

1. Time complexity of inference is linear in the number of classes (assuming each optimization costs the same) because each class performs its own class-conditional optimization to find an optimal variational distribution.

2. Achieving high accuracy on complex data distributions such as CIFAR-10 [20] remains challenging.

Our contributions address only the first issue, of time complexity during inference. The computational load is manageable for 10-class datasets like MNIST and SVHN, but would become impractical for any application with many categories, like the 1000-class ImageNet classification dataset [9]. With this motivation, we use a decision tree in which each decision node is a binary E-ABS classifier to classify in logarithmic time (Chapter 2). We also describe a method to use a generative model to classify an input's latent vector in constant time (Chapter 3).

Chapter 2

Hierarchical Binary Classification

2.1 Method

Vanilla ABS-style models require K optimizations to be performed at inference time, one optimization per class. Using a hierarchical decision tree structure, $O(\log K)$ optimizations are needed to classify an input. The model is defined as follows.

Let C_1, C_2 be a partition of the classes $\{1, \ldots, K\}$. Use an E-ABS model to classify inputs as belonging to C_1 or C_2. For each subset of classes C_1, C_2, partition C_i into C_{i1}, C_{i2} and use an E-ABS model to classify inputs into C_{i1}, C_{i2}. Repeat until all subsets are of size 1. A binary decision tree structure emerges from this construction, with $K-1$ binary classification nodes. See Figure 2.1 for an example of the tree structure with a naive partitioning of the labels $\{0, 1, \ldots, 9\}$. As long as the class partitions are as equally-sized as possible, i.e. differing in size by at most one, any path from the root to a leaf travels through either $\lfloor \log_2 K \rfloor$ or $\lceil \log_2 K \rceil$ decision nodes. Each decision node performs 2 optimizations since it is

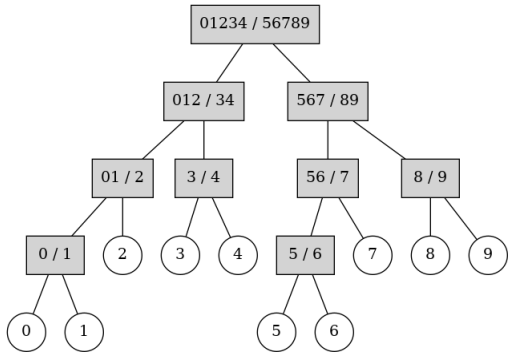

Figure 2.1: Naive splits for hierarchical binary classification trees on 10 labels.

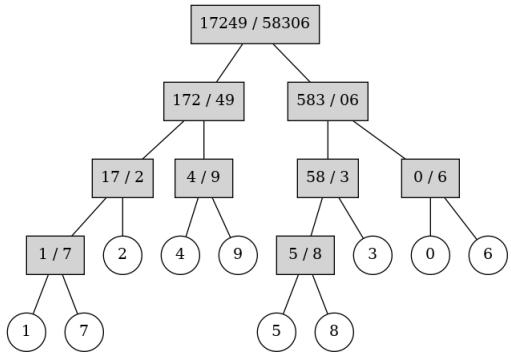

Figure 2.2: Manually designed splits for hierarchical binary classification trees on 10 labels.

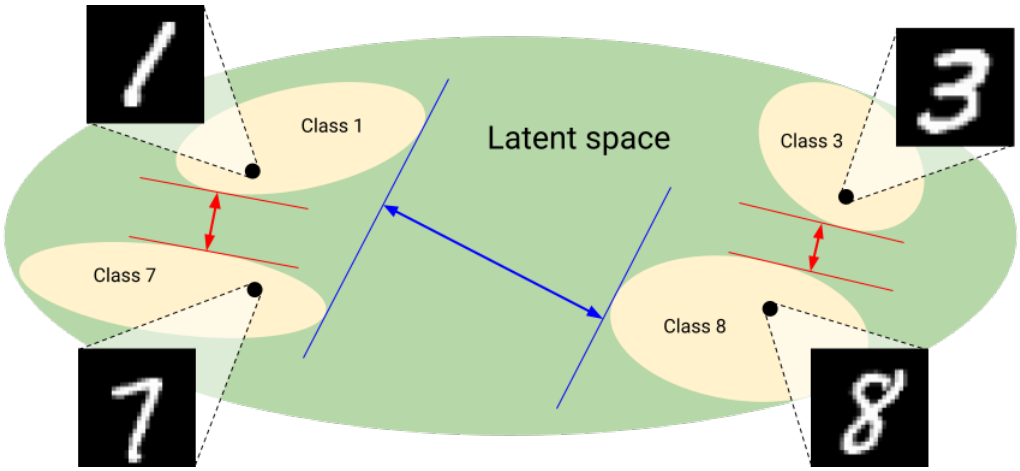

Figure 2.3: A simplified view of several classes' latent distributions for an unconditional generative model on the entire MNIST dataset.

2-class E-ABS, so at most $2\lceil \log_2 K \rceil$ optimizations are performed at inference time.

Similar constructions to use binary classifiers for multi-class classification have been discussed in the literature before [28, 37, 25], including in conjunction with neural methods [40]. In this work, E-ABS classifiers are used at each node to confer the robustness advantages of E-ABS models.

2.2 Choosing Binary Splits

The question of choosing which classes are grouped together at each binary decision node remains. We first identify two sources of classification difficulty in the tree-based model:

1. If many classes are partitioned into a binary split, the distribution of each binary class is the union of several classes' distributions. Since the generative classifier must model a more complex distribution, difficulty increases. Thus, in general, shallow nodes will have more difficulty than deep nodes.

   For example, consider the nodes 01234 / 56789 and 0 / 1 in Figure 2.1. The two generative models in 01234 / 56789 each learn the union of five different class distributions, while those in 0 / 1 each only learn one.

2. We hypothesize that there are different degrees of inherent similarity between the classes, and deciding between similar classes is more difficult than deciding between dissimilar ones.

   For example, the shape of the digit 1 could be said to be more similar to 7 than 8. See Figure 2.3 for a visualization, with the assumption that similar classes' latent

Dataset	Depth	Naive	Manual (vs. Naive)	Greedy (vs. Naive)
SVHN	1	5.181	5.235 (+0.053)	**5.250 (+0.069)**
	2	5.382	5.408 (+0.025)	**5.412 (+0.030)**
	3	**5.720**	5.591 (−0.128)	5.551 (−0.169)
	4	**5.670**	5.633 (−0.037)	5.528 (−0.142)
MNIST	1	5.314	**5.516 (+0.202)**	**5.516 (+0.202)**
	2	5.700	5.796 (+0.097)	**5.836 (+0.136)**
	3	**6.409**	5.745 (−0.664)	5.901 (−0.508)
	4	**6.799**	6.047 (−0.752)	5.574 (−1.225)

Table 2.1: Average d_{C_1,C_2} by depth for different label splitting regimes. For each depth, the value reported is the average of d_{C_1,C_2} over all the nodes at that depth. There is one node at depth 1 (the root), two nodes at depth 2, four nodes at depth 3, and two nodes at depth 4. In Figures 2.1, 2.2, 2.4, and 2.5, each node is drawn as $\boxed{C_1 \ / \ C_2}$.

distributions are closer together than dissimilar classes'. If the four depicted classes were to be paired and a binary E-ABS classifier trained to classify one pair versus the other, then the split $\boxed{17 \ / \ 38}$ would likely have higher accuracy than $\boxed{13 \ / \ 78}$ because the former distributions are farther apart (blue) than the latter ones (red).

To partially alleviate the first issue, we increase the latent dimensionality of the binary classifiers at depth 1 or 2.

A Principle for Choosing Class Partitions

To further counterbalance the asymmetry arising from classes being grouped together, we combine similar classes in the shallower decision nodes, thereby deferring to deeper nodes the harder tasks of discriminating between similar classes. Later, we show experimentally that such arrangements perform better than those in which dissimilar classes are grouped together and difficulty concentrates near the root node.

We now address the task of deciding what the class partitions will be in the decision tree. One option is manual design; for the digits $\{0, 1, \ldots, 9\}$, we manually specify the splits to follow visual intuition about the shapes of digits, arriving at the structure in Figure 2.2.

An Algorithm to Partition Classes

Motivated by the latent space intuition in Figure 2.3, we train an unconditional AAE [27] model on the entire dataset and compute the latent vectors of every point in the training set. For a data point x and a subset of classes $C \subseteq \{1, \ldots, K\}$, we define the distance from x to C as the Euclidean distance from the latent vector of x to the closest latent vector of

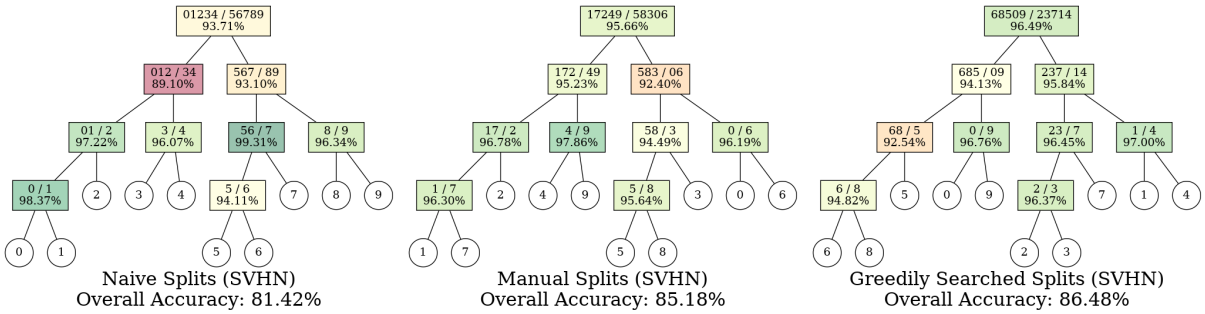

Figure 2.4: SVHN performance. Color scale shared across the three trees, with red for low accuracy and green for high accuracy.

Dataset	Tree (Naive)	Tree (Manual)	Tree (Greedy)	Vanilla E-ABS
SVHN	81.42	85.18	86.48	**90.78**
MNIST	98.32	98.37	98.37	**99.16**

Table 2.2: Percent accuracy of hierarchical decision tree classifiers and vanilla E-ABS.

any point in C. Denote this quantity $m_{x,C}$. For two disjoint subsets of classes C_1, C_2, we define the one-way distance $d_{C_1 \to C_2}$ from C_1 to C_2 as the average value of m_{x,C_2} over all x in C_1. Finally, let $d_{C_1,C_2} = \frac{1}{2}(d_{C_1 \to C_2} + d_{C_2 \to C_1})$ for symmetry. These values can be computed quickly by caching the values $m_{x,\{y\}}$ for each $y \in \{1, \ldots, K\}$ for each point x in the dataset.

Next, examine all splits of the labels $\{1, \ldots, K\}$ into balanced binary partitions and greedily choose the split (C_1, C_2) that maximizes d_{C_1,C_2}. This groups similar classes together for shallow nodes, under the assumption that class similarity is inversely related to this distance measurement. For each set of labels C_1, C_2, again greedily choose the split (C_{i1}, C_{i2}) that maximizes $d_{C_{i1},C_{i2}}$. Repeat until all subsets are of size one. The greedily chosen splits for SVHN and MNIST can be seen in the rightmost trees in Figures 2.4 and 2.5, respectively.[1]

Table 2.1 confirms that, in shallow nodes, average latent-vector distances d_{C_1,C_2} are larger for manual and greedily searched splits compared to the naive ordering. The trend reverses in deeper nodes. Under the assumption that d_{C_1,C_2} is inversely related to similarity, the manual and greedily searched splits are indeed better than the naive ones.

2.3 Results

On SVHN, manually specifying label splits for the decision tree increases overall accuracy compared to naive splits, and greedily chosen splits give the highest accuracy. Furthermore,

[1]The manual and greedy splits happen to be similar for MNIST. In particular, the first split is identical.

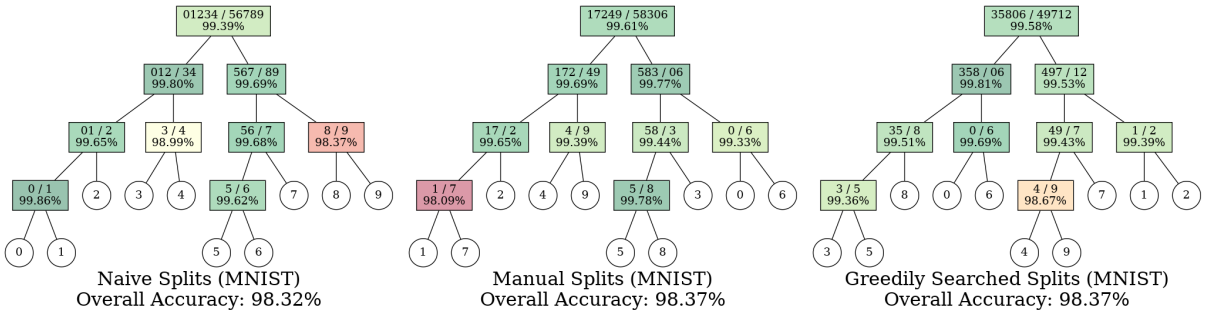

Figure 2.5: MNIST performance. Color scale shared across the three trees, with red for low accuracy and green for high accuracy.

the naive-split tree has worse performance on shallower nodes and better performance on deep nodes, in contrast to the other two, whose nodes have more uniform performance throughout. These observations support the hypothesis that combining similar classes in shallower decision nodes balances against the naturally higher difficulty of these nodes, which is due to the union of several classes' distributions.

However, vanilla E-ABS trained directly on the datasets outperformed all three decision tree models. In Table 2.2, we report the overall accuracy of all the models under consideration. Figures 2.4 and 2.5 display more detailed performance information from the decision trees. The accuracy values reported at each node are computed based only on data points that were correctly classified by the parent node. Relative to vanilla E-ABS, the decision tree model has lower inference time at the cost of having worse accuracy.

The same trends occured on MNIST, but the choice of label splits did not have as large an impact as on SVHN.

A Note on Complexity

While vanilla E-ABS outperforms the decision tree model in accuracy, the tree has better time complexity during inference: $O(\log K)$ for the tree vs. $O(K)$ for E-ABS. However, if computation is freely parallelizable, then vanilla E-ABS takes constant time (each optimization can be performed independently, in parallel). In this case, the decision tree still takes $O(\log K)$ since parallelization only reduces inference time by a constant factor of $\frac{1}{2}$ (each node has 2 optimizations that can be parallelized, but each binary decision must be made serially).

Another aspect to consider is training time. For simplicity, let us first assume that the training time for an E-ABS model is proportional to the number of classes. Then to train a single E-ABS model on K classes requires $O(K)$ time, but training the tree model takes $O(K \log K)$ since there are $O(2^d)$ trees at depth d, each has $O(K/2^d)$ classes, and depth ranges from 1 to $\lceil \log_2 K \rceil$.

Label Subsets S_i	Label 0	1	2	3	4	5	6	7	8	9
{0 1 2 3 9}	1	1	1	1	0	0	0	0	0	1
{0 1 4 5 8}	1	1	0	0	1	1	0	0	1	0
{0 2 4 6 8}	1	0	1	0	1	0	1	0	1	0
{5 6 7 8 9}	0	0	0	0	0	1	1	1	1	1

Figure 2.6: An example of subsets S_i of the labels $\{0, 1, \ldots, 9\}$ and the binary encoding induced by them. Each column is the bit string of the corresponding class.

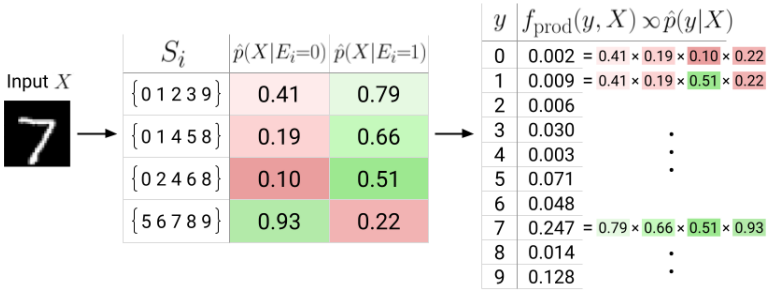

Figure 2.7: An example of the inference process using the encoding described in Figure 2.6. The bit string for label 7 is $e(7) = 0001$ since $e_1(7) = e_2(7) = e_3(7) = 0$ and $e_4(7) = 1$. The coloring reflects the relative difference between values in each row.

Finally, while the inference-time gains of the decision tree model would be clear for a dataset like ImageNet, where 1000 classes means 2000 optimizations for E-ABS and 20 optimizations for the tree, the question of choosing high-performance splits would become an issue. The exact greedy search procedure described earlier to maximize d_{C_1,C_2} would be intractable because the number of permutations grows factorially in K, so a different algorithm would be necessary. To this end, Wan et al. [40] explores inducing a class hierarchy using agglomerative clustering on pre-trained network weights.

2.4 Further Directions

The hierarchical binary classifier described in this work is far from the only way to use binary classifiers to do multi-class classification [25], and we defer exploration of these directions to future work. Allwein et al. [1] propose a code-matrix framework that unifies several schemes, including the one presented here. In that work, the correct sequence of binary decisions in the tree for a single class forms a bit string representation of that class.

This perspective suggests another possible approach using E-ABS. For a K-class classification task, consider taking $m := \lceil \log_2 K \rceil$ different subsets S_i of the labels such that $|S_i| = \frac{K}{2}$ for all $i = 1, \ldots, m$, and no two labels $y_1 \neq y_2$ have $y_1 \in S_i \iff y_2 \in S_i$ holding for every i. Let $\mathbf{1}_{\{A\}}$ denote the indicator function for event A. For each i and label y, let $e_i(y) := \mathbf{1}_{\{y \in S_i\}}$, and train an E-ABS classifier f_i to predict e_i. The bit string representation for class y is the vector $e(y) := \big(e_1(y), \ldots, e_m(y)\big)$. See Figure 2.6.

At inference time, consider an input X and its true class Y, both random variables. For each i let $E_i = \mathbf{1}_{\{Y \in S_i\}}$, and let $E := \big(E_1, \ldots, E_m\big)$. Since $Y = y$ if and only if $E = e(y)$, classifying Y is equivalent to classifying the vector E, i.e. $p(Y = y \mid X) = p(E = e(y) \mid X)$. We wish to find $\arg\max_y p(e(y) \mid X) = \arg\max_y \prod_i p(E_i = e_i(y) \mid X)$. By Bayes' rule

and the condition that $|S_i| = \frac{K}{2}$, we have $p(E_i \mid X) \propto p(X \mid E_i)$. Since the f_i are E-ABS models, we have access to the likelihood estimates $\hat{p}(X \mid E_i = 0)$ and $\hat{p}(X \mid E_i = 1)$. Then if we let $f_{\text{prod}}(y, X) = \prod_i \hat{p}(X \mid E_i = e_i(y))$, we have that $\hat{p}(y|X) \propto f_{\text{prod}}(y, X)$ and the proportionality does not depend on y. Thus, the classification result is $\arg\max_y p(e(y) \mid X) = \arg\max_y f_{\text{prod}}(y, X)$. See Figure 2.7.

This scheme would specifically exploit the generative nature of E-ABS, since the logits of standard discrimnative classifiers are known to be poor estimators of likelihood. The downside of such a model is that each binary classifier has a difficult task since each binary class is the union of half of the K classes. Conversely, $O(\log K)$ complexity is preserved for inference, and unlike the tree model, computation is parallelizable.

Further extending the interpretation of binary classification as encodings, Dietterich et al. [10] consider a scheme where the bit string encodings of each class are specially chosen to be an error-correcting code such as a Hamming code [15]. In such a model, the likelihood estimates $p(E \mid X)$ would be unnecessary; instead, the class y with the nearest (by Hamming distance) bit-encoding $e(y)$ may be used. Thus, the classification decision of the model would be robust to errors by a certain number of the binary classifiers.

Chapter 3

Classifying Latent Vectors

A crucial aspect of the robustness of ABS-style models is their inference procedure, which avoids depending solely on the output of any feedforward neural network layers that could introduce vulnerability. Instead, using optimization, they search the latent space for latent vectors that reconstruct the input well. ABS-style inference requires K optimizations for a K-class classifier. The decision tree model presented in the previous section reduces the computation from $O(K)$ to $O(\log K)$.

We now propose a different model where only one optimization is needed for inference instead of K, further reducing inference complexity to $O(1)$. A single latent variable model learns the entire dataset, with a single decoder and latent space shared by all classes. At inference time, it searches the latent space and classifies the optimal vector based on the structure of the latent space.

3.1 Model Structure and Training

Let X be a data point, $y \in \{1, \ldots, K\}$ a class, and Z a latent vector. Consider a latent-variable model in which all classes share the same latent space according to a specified prior. Then the class-conditional distribution of the latent vector $Z|y$ depends on y. Also let the likelihood of the data be conditionally independent of the class given the latent vector, i.e. $p(X|Z, y) = p(X|Z)$, to reflect that the model has a single class-agnostic decoder. At inference time, latent vectors are sampled from the prior and the best vector is the starting point for optimization, as in ABS. The optimal latent vector Z minimizes the reconstruction loss with respect to the input plus a regularization term that encourages staying in-distribution. Due to the prior imposed on the latent space, the class-conditional likelihood $p(Z|y)$ can be found immediately for any class y, and the class maximizing this value is the prediction.

For this work, the class-conditional latent distributions $Z|y$ are multivariate Gaussian, all with the same scaled identity covariance, and with mean μ_y dependent on y. For K classes, the μ_y are chosen in such a way that their mean is zero and the geometric relation of each class to each other is symmetric. Our arrangement requires that the latent dimensionality

is at least the number of classes and is as follows:

$$(\mu_y)_i = \begin{cases} -\frac{1}{K}, & i \neq y \text{ and } i \leq K \\ 1 - \frac{1}{K}, & i = y \\ 0, & i > K \end{cases}$$

We use an AAE [27] as the latent variable model, with different latent-space discriminators for each class to account for the different class-conditional latent distributions. As in E-ABS, the training objective includes a discriminative loss term: a cross-entropy loss with respect to the true class where the logits are the log class-conditional latent likelihoods $\big(\log p(Z|y=1), \ldots, \log p(Z|y=K)\big)$.

3.2 Regularization Term

For inference, a regularization term is required in addition to the reconstruction loss so that the optimization does not travel far outside the latent prior, since deep generative models are known to give poor likelihood estimates on out-of-distribution samples [30, 5, 17]. At the same time, when the latent vector is being optimized, this term should not inhibit it from switching classes during optimization, i.e. traveling to a region where a different class is most likely under the prior. That way, optimization will be able to correct for starting points that begin in the wrong class-region.

Several methods can be considered for this purpose:

- Find the class y that is most likely for the current value of the latent vector, and use the loss term from that class's latent discriminator model (or KL divergence).
- Normalize the class-conditional latent likelihoods $\big(p(Z|y=1), \ldots, p(Z|y=K)\big)$ to be a discrete probability distribution over the K classes, denoting the current latent vector as Z. Use the discrete entropy of this distribution.
- A KL divergence between a latent variational distribution and the mixture-of-Gaussians latent prior, such as would be used in a VAE, is unsuitable here for several reasons. First, there is no closed-form expression for the KL divergence between Gaussian mixtures. Second, if the variational distribution is a single Gaussian, the KL-minimizing parameters will have high covariance, defeating the purpose of the variational distribution to concentrate on the latent region that decodes the input well.

The first method will keep the latent vector close to the prior, but will make it difficult for the optimization to switch classes. In the region where a single class is most likely, the gradient will pull the latent vector toward the mean of that distribution. At the boundaries, the gradient discontinuously changes direction to point to the new most-likely class.

The second method incentivizes being close to one of the latent prior means, since increasing the likelihood of one class reduces the entropy of the discrete distribution. The

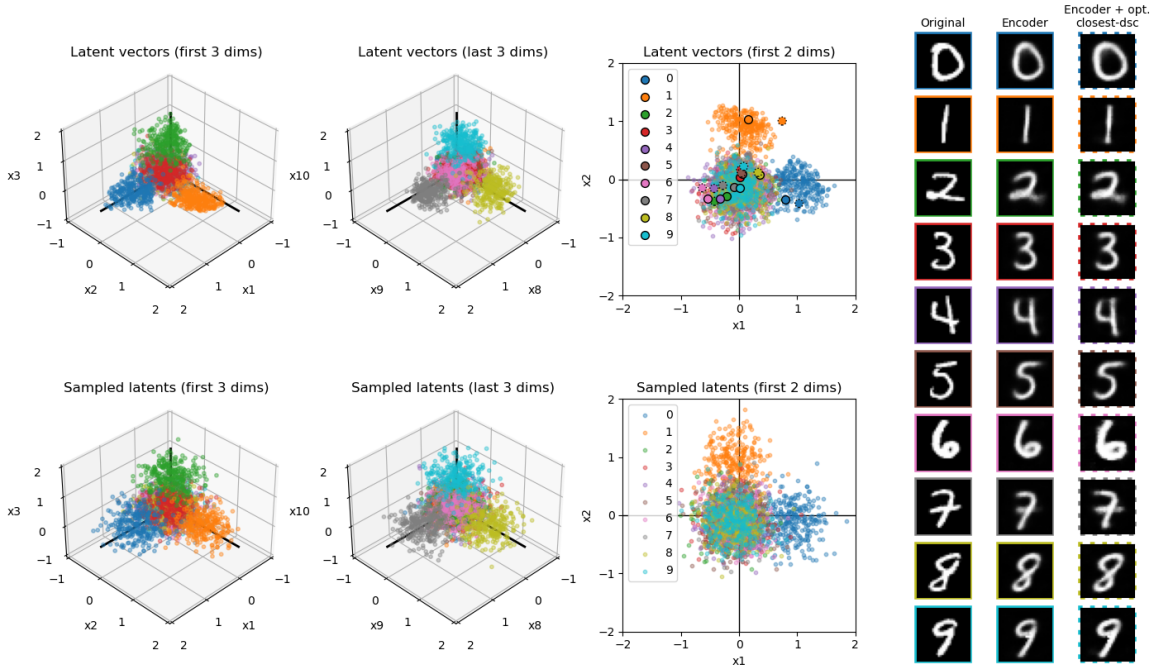

Figure 3.1: Behavior of class-conditional AAE on MNIST. During optimization, the loss from the closest class's discriminator is used as regularization. Latent vectors returned by encoder are plotted. The reconstructions of ten input points using the encoder's outputted latent vector as well as a latent vector determined by optimization are shown at right. Solid-border circles in the third-column plots are encoder outputs, dashed-border circles are latents returned by optimization.

gradient is also continuous at the class boundaries. One downside, however, is that the entropy is bounded, meaning it only has a limited capacity to penalize. Moreover, when the latent is not particularly close to any class, the gradient of entropy is relatively low, further weakening its ability to regularize.

3.3 Results

All experiments were conducted on MNIST. Figures 3.1 and 3.2 demonstrate the behavior of the model when the closest class' discriminator loss ("closest-dsc") and entropy, respectively, are used as regularization. The learned latent distributions largely match the Gaussian priors, although not perfectly, such as for class 1 (orange). Moreover, optimization tends to stray outside the prior distributions, leading to potential problems with out-of-distribution

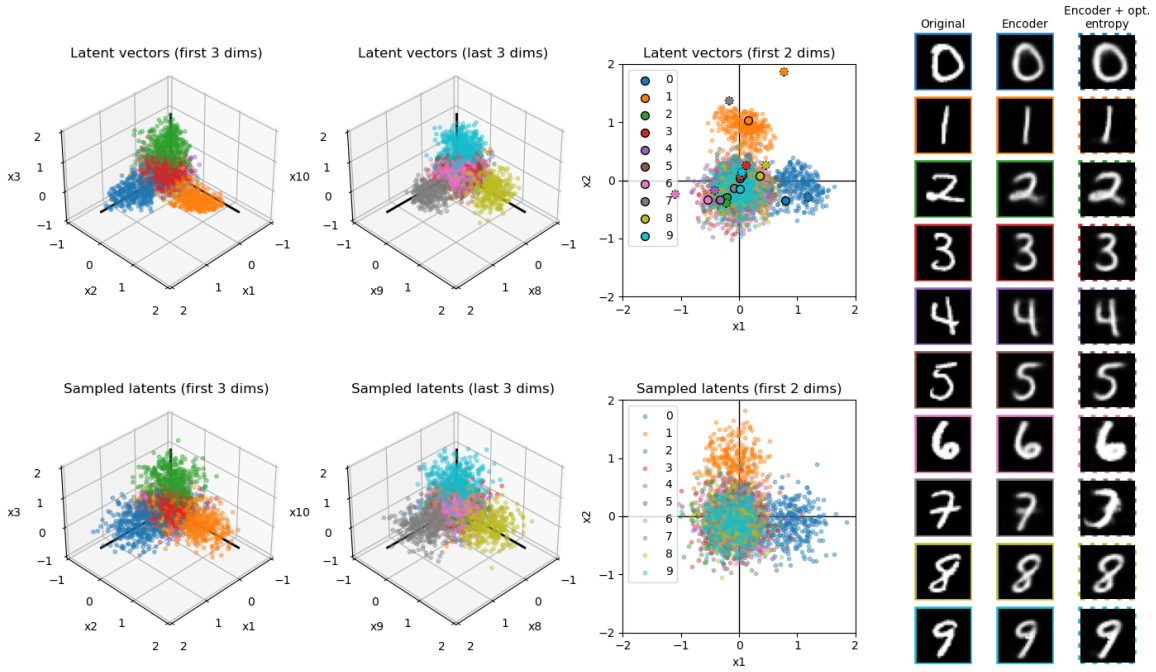

Figure 3.2: Same as Figure 3.1 but using entropy as regularization during optimization.

Regularization	Enc	Enc+Sample	Enc+Sample+Opt	Opt Switches Class
Closest Disc. Loss	**97.42**	94.53	88.43	10.51%
Entropy	**97.42**	93.88	83.10	15.32%

Table 3.1: Percent accuracy of class-conditional AAE on MNIST test data and percent of optimizations that switch class. Description of each column:
Enc (encoder): Classify encoder output directly.
Enc+Sample: Classify the point minimizing weighted sum of reconstruction loss and regularization term over the samples and the encoder's output.
Enc+Sample+Opt: Classify the latent resulting from optimization starting from the point found by Enc+Sample.
Opt Switches Class: Number of MNIST test points for which the most likely classes of the optimization starting and ending points differed, divided by the size of the MNIST test dataset.

Regularization	True Prior	Enc	Enc+Sample	Enc+Sample+Opt
Closest Disc. Loss	−4.87	−3.78	−4.75	−8.96
Entropy	−4.87	−3.78	−4.34	−8.36

Table 3.2: Average log likelihoods of latent vectors for class-conditional AAE on MNIST test data. Low values roughly indicate that the latent vectors tend to be far away from the prior distribution. Thus, the encoder produces latents that are close to the prior, while optimization travels far away. Log likelihood is taken with respect to the most likely class distribution, i.e. if class 4 gives the highest likelihood for z, then $\log p(z \mid y = 4)$ would be used. Log likelihood is calculated for every test point this way and averaged over the whole dataset. The True Prior column is slightly different, using the same log likelihood calculation but averaging over samples drawn from the true mixture-of-Gaussians latent prior instead of test points. This column illustrates what the log likelihood would "ideally" look like under the true prior.

behavior. The entropy regularization term suffers from this issue more severely than the closest-dsc term, affirming the intuition that the entropy term does not send a strong enough regularization signal except possibly when the latent vector is already close to one of the prior means. Contrary to expectations, Table 3.1 shows that using the closest-dsc term still allows the optimization to switch classes, though slightly less so than the entropy term.

The accuracy of this method is low. As detailed in Table 3.1, simply classifying the encoder's output results in the highest accuracy of 97.42%, although this is still significantly lower than the performance of E-ABS. Choosing the point that minimizes a weighted sum of the reconstruction loss and the regularization term and classifying it directly results in lower accuracy. Finally, further optimizing that point results in the lowest accuracy of all. Using optimization causes accuracy to drop 6.10% when closest-dsc is used and 10.78% when entropy is used. Most of the cases in which optimization switches classes are therefore switching from a correct starting point to an incorrect ending point.

Key Difficulties

The optimization procedure fails to work properly, as demonstrated by the fact that accuracy drops when it is used. We identify several reasons:

- It is too easy for the optimization to veer away from the prior distribution. One reason for this is that the regularization terms are ineffective at keeping the optimization within the prior, especially the entropy term due to its boundedness. Another reason is that the model produces good reconstructions for latents that are outside the prior distribution, due to the difficulty in controlling the model's behavior outside the prior. The discriminative loss term, which attempts to use outlier exposure [17] to address

this problem, is evidently not sufficient. Table 3.2 shows that optimization moves far away from the prior distribution.

- Whereas in E-ABS each class has its own latent space, here all classes share the same space, so it is more critical that the learned boundaries adhere closely to the prior than in E-ABS.

Future Work

Based on the analysis in the previous section, it is imperative for the latent prior to be learned correctly. While the difficulty of this task for deep generative models is well-recognized [30, 5, 17], some developments specifically modify model architectures to address this problem, such as two-stage VAE models [7]. Incorporating such a model is likely necessary to achieve good performance with the concept presented here.

Chapter 4

Discussion

In this work, we demonstrate that the costly inference time complexity of ABS-like models can be addressed, albeit with some possible concessions to final accuracy. Nevertheless, the challenge of generalizing to more complex datasets with ABS remains. To this end, ABS-style frameworks must as a baseline ensure two conditions are met:

1. The generative architectures used must be expressive enough to model highly complex datasets.

2. The sample-and-optimize inference procedure must search the latent space effectively with a tractable amount of computation.

To the first point, recent advances in generative modeling have led to powerful models [32, 41, 33] that can model highly complex datasets such as CIFAR-10 and ImageNet, though care must be taken to handle out-of-distribution likelihood estimation [30, 5, 17]. To the second point, while the improvements in this report address the issue of efficiency, they do so under the assumption that a single optimization takes a constant amount of time and has high accuracy. When considering challenging datasets, these assumptions must be revisited.

4.1 Considerations for Scaling ABS-style Inference

The theoretical certificate of robustness for ABS hinges on finding a global optimum with optimization [36, 18], and more generally, inference is unlikely to perform well if the latent space cannot be searched comprehensively. The robust latent search procedure of ABS inference, based on pixel-space distance, protects the model from adversarial examples that might exist in any feed-forward neural layers applied to the input directly. Non-reliance on neural outputs shifts the burden of finding a rare latent region that reconstructs the input onto the initial sampling and optimization.

Since the pixel-space reconstruction loss is a poor indicator of perceptual similarity, its gradient signal will likely be ineffective at guiding the optimization. Existing approaches to

creating perceptually-aligned features, from which distances may be calculated, rely on feed-forward neural network components. While these components may be able to be hardened using existing robustness approaches like adversarial training [26], the additional advantages of the ABS structure become questionable over simply using those approaches on their own.

Alternatively, pixel-space reconstruction loss may be sufficient if the optimization starting point is already close to an optimal region. Thus, taking many samples to get better initial coverage may compensate for less informative loss signals. To cover the latent space with a certain density, however, the requisite number of samples increases exponentially in the dimensionality of the latent space. Complex datasets require large latent spaces due to their high intrinsic complexity.

Independently of the computational concerns addressed in this work, the task of scaling ABS-style inference to complex datasets remains an open problem. Advances toward this goal, combined with the strides already being taken in generative modeling and the speedups presented here, would overcome the barriers for generative classifiers as a promising and practical alternative to feed-forward architectures in robust classification.

Bibliography

[1] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers". In: *J. Mach. Learn. Res.* 1 (Sept. 2001), pp. 113–141. ISSN: 1532-4435. DOI: 10.1162/15324430152733133. URL: https://doi.org/10.1162/15324430152733133.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 274–283. URL: http://proceedings.mlr.press/v80/athalye18a.html.

[3] Nicholas Carlini and David A. Wagner. "Towards Evaluating the Robustness of Neural Networks". In: *2017 IEEE Symposium on Security and Privacy (SP)* (2017), pp. 39–57.

[4] Nicholas Carlini et al. "On Evaluating Adversarial Robustness". In: *ArXiv* abs/1902.06705 (2019).

[5] Hyunsun Choi, Eric Jang, and Alexander A. Alemi. "WAIC, but Why? Generative Ensembles for Robust Anomaly Detection". In: *arXiv e-prints*, arXiv:1810.01392 (Oct. 2018), arXiv:1810.01392. arXiv: 1810.01392 [stat.ML].

[6] B. Dai and D. Wipf. "Diagnosing and Enhancing VAE Models". In: *ArXiv* abs/1903.05789 (2019).

[7] B. Dai and D. Wipf. "Diagnosing and Enhancing VAE Models". In: *ArXiv* abs/1903.05789 (2019).

[8] Peter Dayan et al. "The helmholtz machine". In: *Neural computation* 7.5 (Sept. 1995), pp. 889–904.

[9] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[10] Thomas G. Dietterich and Ghulum Bakiri. "Solving Multiclass Learning Problems via Error-Correcting Output Codes". In: *J. Artif. Int. Res.* 2.1 (Jan. 1995), pp. 263–286. ISSN: 1076-9757.

[11] Carl Doersch. "Tutorial on Variational Autoencoders". In: *ArXiv* abs/1606.05908 (2016).

[12] Tal Golan, Prashant C. Raju, and Nikolaus Kriegeskorte. "Controversial stimuli: pitting neural networks against each other as models of human recognition". In: *CoRR* abs/1911.09288 (2019). arXiv: 1911.09288. URL: http://arxiv.org/abs/1911.09288.

[13] I. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *CoRR* abs/1412.6572 (2015).

[14] Sven Gowal et al. *On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models*. Oct. 2018.

[15] R. W. Hamming. "Error detecting and error correcting codes". In: *The Bell System Technical Journal* 29.2 (1950), pp. 147–160. DOI: 10.1002/j.1538-7305.1950.tb00463.x.

[16] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[17] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. *Deep Anomaly Detection with Outlier Exposure*. Dec. 2018.

[18] An Ju and David A. Wagner. "E-ABS: Extending the Analysis-By-Synthesis Robust Classification Model to More Complex Image Domains". In: *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security* (2020).

[19] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: 1312.6114 [stat.ML].

[20] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Neural Information Processing Systems* 25 (Jan. 2012). DOI: 10.1145/3065386.

[22] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[23] Mathias Lécuyer et al. "Certified Robustness to Adversarial Examples with Differential Privacy". In: *2019 IEEE Symposium on Security and Privacy (SP)* (2019), pp. 656–672.

[24] Y. Liu et al. "Delving into Transferable Adversarial Examples and Black-box Attacks". In: *ArXiv* abs/1611.02770 (2017).

[25] Ana Carolina Lorena, A. Carvalho, and João Gama. "A review on the combination of binary classifiers in multiclass problems". In: *Artificial Intelligence Review* 30 (2009), pp. 19–37.

[26] A. Madry et al. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *ArXiv* abs/1706.06083 (2018).

[27] Alireza Makhzani et al. "Adversarial Autoencoders". In: *ArXiv* abs/1511.05644 (2015).

[28] Eddy Mayoraz and Miguel Moreira. "On the Decomposition of Polychotomies into Dichotomies". In: *Proceedings of the Fourteenth International Conference on Machine Learning*. ICML '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 219–226. ISBN: 1558604863.

[29] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and P. Frossard. "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 2574–2582.

[30] Eric T. Nalisnick et al. "Do Deep Generative Models Know What They Don't Know?" In: *ArXiv* abs/1810.09136 (2019).

[31] Yuval Netzer et al. "Reading Digits in Natural Images with Unsupervised Feature Learning". In: 2011.

[32] Aäron van den Oord, Oriol Vinyals, and K. Kavukcuoglu. "Neural Discrete Representation Learning". In: *NIPS*. 2017.

[33] Aäron van den Oord et al. "Conditional Image Generation with PixelCNN Decoders". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 4797–4805. ISBN: 9781510838819.

[34] Nicolas Papernot, P. McDaniel, and I. Goodfellow. "Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples". In: *ArXiv* abs/1605.07277 (2016).

[35] Mihaela Rosca, Balaji Lakshminarayanan, and S. Mohamed. "Distribution Matching in Variational Inference". In: *ArXiv* abs/1802.06847 (2018).

[36] Lukas Schott et al. *Towards the first adversarially robust neural network model on MNIST*. 2018. arXiv: 1805.09190 [cs.CV].

[37] F. Schwenker. "Hierarchical support vector machines for multi-class pattern recognition". In: *KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No.00TH8516)*. Vol. 2. 2000, 561–565 vol.2. DOI: 10.1109/KES.2000.884111.

[38] Christian Szegedy et al. *Intriguing properties of neural networks*. 2014. arXiv: 1312.6199 [cs.CV].

[39] Jonathan Uesato et al. "Adversarial Risk and the Dangers of Evaluating Against Weak Attacks". In: *ArXiv* abs/1802.05666 (2018).

[40] Alvin Wan et al. *NBDT: Neural-Backed Decision Trees*. 2021. arXiv: 2004.00221 [cs.CV].

[41] Han Zhang et al. "Self-Attention Generative Adversarial Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 7354–7363. URL: `http://proceedings.mlr.press/v97/zhang19d.html`.