The Effect of Error Correcting Output Codes and Learnable Output Codes on Adversarial Robustness of Deep Neural Networks

A project report submitted in partial fulfillment of the requirements for the award of

Dual Degree (BTech + Mtech) in Electrical Engineering

Submitted by: **Farhad Asim Ashfaq Ahmad Shaikh**
EE16B149

Under the supervision of
Guide: **Dr. Avhishek Chatterjee**

Department of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
Chennai, Tamil Nadu - 600036

JULY 2021

Department of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
Chennai, Tamil Nadu 600036

CANDIDATE'S DECLARATION

I, **Farhad Asim Ashfaq Ahmad Shaikh, Roll No: EE16B149** , student of Dual Degree in Electrical Engineering, hereby declare that the project dissertation titled "**The Effect of Error Correcting Output Codes and Learnable Output Codes on Adversarial Robustness of Deep Neural Networks**" which is submitted by me to the Department of Electrical Engineering, Indian institute of Technology Madras, Chennai, Tamil Nadu 600036 in partial fulfillment of the requirement for the award of Dual Degree (Btech + Mtech) in Electrical Engineering is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship or other similar title or recognition.

PLACE: Udgir — Farhad Asim Ashfaq Ahmad Shaikh

DATE: 24/07/2021 — EE16B149

Department of Electrical Engineering

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

Chennai, Tamil Nadu 600036

CERTIFICATE

I hereby certify that the Project Dissertation titled "**The Effect of Error Correcting Output Codes and Learnable Output Codes on Adversarial Robustness of Deep Neural Networks**" which is submitted by **Farhad Asim Ashfaq Ahmad Shaikh, Roll No - EE16B146**, student of Dual Degree (Electrical Engineering), Indian Institute of Technology Madras, Chennai, Tamil Nadu - 600036, in partial fulfillment of the requirement for the award of the Dual Degree (B.Tech + M.Tech) in Electrical Engineering, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

PLACE: Chennai

DATE: 24/07/2021

GUIDE: Dr. Avhishek Chatterjee

Department of Electrical Engineering

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

Chennai, Tamil Nadu 600036

ACKNOWLEDGEMENT

I wish to express my sincerest gratitude to Professor Dr Avhishek Chatterjee for his continuous guidance and mentorship that he provided me during the project. He showed me the path to achieve my targets by explaining all the tasks to be done and explained to me the importance of this project as well as its research relevance. He was always ready to help me and clear my doubts regarding any hurdles in this project. Without his constant support and motivation, this project would not have been successful.

PLACE: Udgir

DATE: 24/07/2021

Farhad Asim Ashfaq Ahmad Shaikh

EE16B149

ABSTRACT

Over the past few years, Deep Neural Networks (DNNs) have been shown to achieve state-of-the-art performance on several applications such as computer vision, natural language processing. However, DNN models are susceptible to adversarial attacks where minor modifications to the original inputs can lead to significantly different output distributions and incorrect predictions. A recent work [1] shows that using ideas from the principles of error correction in coding theory improves the robustness of DNNs to adversarial attacks. In this project, we first implement the various ideas proposed in [1] viz. (i) using a sigmoid decoder instead of softmax activation (ii) using error correcting output codes for class representations (iii) improving independence in the output bits through model ensembling. Next, we investigate the idea of learning the output codes instead of a fixed set of output codes. We propose a framework consisting of an input encoder and a label encoder and jointly learn their parameters to minimize the cross-entropy loss. We furthermore implement regularization on learnable output codes which produces similar codes to error correcting output codes but are learned at the time training. We show that our proposed framework outperforms the baseline Softmax and Logistic models on both normal and adversarial inputs in the MNIST and CIFAR-10 datasets. Incomparison to the Tanh model with hadamard code, while our proposed framework performs better on normal inputs, its performance is lower on the adversarial inputs.

Contents

1. INTRODUCTION

Deep neural networks (DNNs) achieve state-of-the-art performance on image classification, speech recognition, and game-playing, among many other applications. However, they are also vulnerable to adversarial examples, inputs with carefully chosen perturbations that are misclassified despite containing no semantic changes . Often, these perturbations are "small" in some sense, e.g. some Lp norm. From a scientific perspective, the existence of adversarial examples demonstrates that machine learning models that achieve superhuman performance on benign, "naturally occurring" data sets in fact possess potentially dangerous failure modes. The existence of these failure modes threatens the reliable deployment of machine learning in automation of tasks. Many defenses have been proposed to make DNNs more robust to adversarial examples; virtually all have been shown to have serious limitations. Adversarial defenses can be broadly classified by which part of the learning pipeline they aim to protect. i) manifold-based which projects the input into a different space ii) quantization-based, which alters input data resolution or encoding iii) randomization-based, in which portions of the input and/or hidden layer activations are randomized or zeroed out. Model based approach seek to defend either by training (augmented) with adversarial examples or by seeking to model properties of inputs or hidden layers and detect adversarial examples with small probability under natural data distribution. We make an attempt towards adversarial robustness by drawing inspiration from coding theory, the branch of information theory which studies the design of codes to ensure reliable delivery of a digital signal over a noisy channel. Coding theory formalizes the idea that in order to minimize the probability of signal error, codewords should be

well-separated from one another. We find that encoding outputs using such codes, as opposed to one-hot encoding significantly improved the probability estimates and increased the robustness to adversarial examples. While implementing the error correcting output codes which are well-separated, we do not really know whether maximizing the separation give us the best performances. In other words,we do not know whether Hamming distance is the optimal criteria for class representation. Hence, we investigate the idea of learning output codes instead of fixed output codes apriori. We propose a framework consisting of an input encoder and a label encoder and jointly learn their parameters to minimize the cross-entropy loss. We discuss our experimental results on the MNIST and CIFAR-10 datasets.

2. MODEL FRAMEWORK

We denote C, the $M \times N$ matrix of codewords, where M denotes the number of classes and N denotes the length of codeword. The kth row of $\mathbf{C}$, $\mathbf{C}_k$, is the desired output of DNN when input is from class K. For one-hot encoding $C = I_M$, the identity matrix of order M. We will consider codes with $N = M$ and $N > M$.

2.1 ERROR CORRECTING OUTPUT CODES

We firstly describe the model framework which uses fixed error correcting output codes for class representation.

2.1.1 SOFTMAX ACTIVATION

The softmax maps a vector $\mathbf{z}$ in $\mathbb{R}^M$ into $(M-1)$-dimensional probability simplex. We will denote the M-dimensional vector of softmax activation by

ψ. The kth softmax activation is given by

$$p_\psi(k) = \frac{exp(z_k)}{\sum_{i=1}^{M} exp(z_i)} \tag{1}$$

2.1.2 SIGMOID ACTIVATION

As proposed by [1], the idea is to map model logits to the elements of a codeword and assign probability to class k as proportional to how positively correlated the model output is to $\mathbf{C}_k$

$$p_\sigma(k) = \frac{max(\sigma(\mathbf{z}) \cdot \mathbf{C}_k, 0)}{\sum_{i=1}^{M}(max(\sigma(\mathbf{z}) \cdot \mathbf{C}_i, 0))} \tag{2}$$

where $\sigma(\mathbf{z})$ and $\mathbf{C}_k$ are vectors of length N. The possible choices of σ can be tanh function taking values in (-1,1). When $\mathbf{C}$ takes values in (0,1) logistic function is appropriate to use. The function max() is used when $\mathbf{C}$ takes values in (-1,1) and the operator is used to avoid negative probabilities.

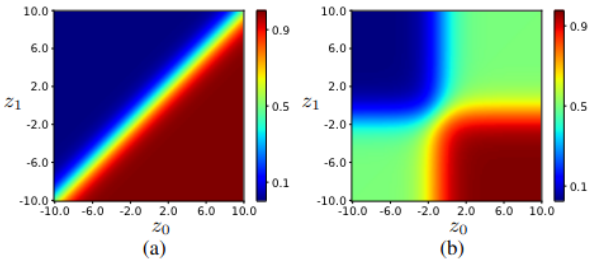

Figure 1: Probability of class 0 as a function of logits, for the (a) softmax activation and (b) sigmoid decoding scheme.

It shows that p_σ allocates non-trivial volume in logit space to uncertainty. Hence, p_σ effectively shrinks the attack surface available to an attacker seeking to craft adversarial examples.

2.1.3 HAMMING DISTANCE

The hamming distance between any two binary codewords x and y, denoted d(x,y) is $|x-y|_0$, where $|\cdot|_0$ denotes the L_0 norm. The hamming distance of codebook **C** is defined as

$$d = min\{d(x,y) : \mathbf{x}, \mathbf{y} \in \mathbf{C}, \mathbf{x} \neq \mathbf{y}\} \tag{3}$$

The standard one-hot coding scheme has a Hamming distance of only 2. This means that if the adversary can sufficiently alter even a single logit, an error may occur. Ideally, we want the classifier to be robust to changes to multiple logits. Consider the $M = N = 32$ case where each of 32 classes is represented by a 32-bit codeword (meaning that the DNN has 32 outputs versus 2 for the case in Figure 1). Figure 2 shows the probability of class 0 as a function of a 3 dimensional slice of the logits (z_{29}, z_{30}, z_{31}), where the fixed logits are set to be consistent with class 0

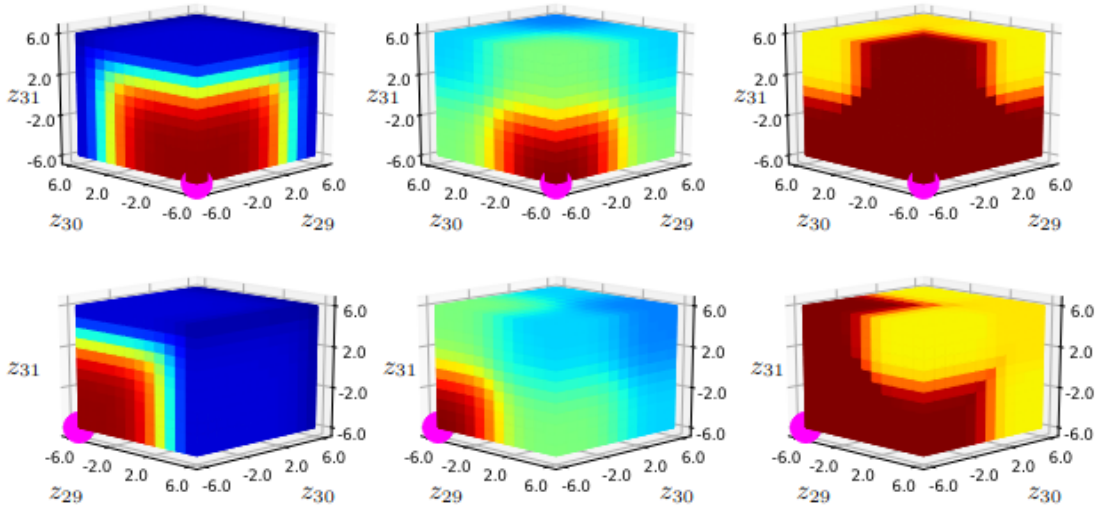

Figure 2: Probability of class 0 as a function of logits, for different choices

of output activation and code, for a 32 class multi-classification problem: (leftmost column) softmax (Eq 1) with $\mathbf{C} = \mathbf{I}_{32}$, (middle column) sigmoid decoder (Eq 2) with logistic activation and $\mathbf{C} = \mathbf{I}_{32}$, (rightmost column) sigmoid decoder (Eq 2) with tanh activation and $\mathbf{C} = \mathbf{H}_{32}$, a Hadamard code. 29 logit values are fixed and remaining logits (here denoted z_{29}, z_{30}, z_{31}) are allowed to vary. The magenta circle shown has probability > 0.999 of being in class 0.

Note how the softmax decoder has a very small region corresponding to uncertainty (i.e., probability near 0.5); as the logits vary, the model rapidly transitions from assigning probability ≈ 1 to class 0 to assigning probability ≈ 0. In contrast, the logistic decoder assigns far more volume to uncertainty. The Hadamard code based decoder is even more robust; it still assigns large probability to class 0 despite large changes to multiple logits.

2.1.4 CODE DESIGN

The work that has followed on ECOC (Error Correcting Output Codes) focused on potential gains in generalization of multi-class settings over one-hot coding. Several methods exist in order to create the good ECOC which focus on achieving a large Hamming distance between codes. We state a theorem which is used to select a near optimal choice of $\mathbf{C}$.

Theorem 1 *(Plotkin's Bound). For a $M \times N$ coding matrix $\boldsymbol{C}$, $d \leq \left\lfloor \frac{N}{2} \frac{M}{M-1} \right\rfloor$*

Theorem states an upper bound on the Hamming distance of $\mathbf{C}$. For large M and even N, the bound approaches $\frac{N}{2}$ which can be achieved if Hadamard matrix is chosen. We use the notation $\mathbf{H}_P$ to denote a $P \times P$ Hadamard matrix. If there are more codewords P available than actual classes(M), we use first M rows of $\mathbf{H}_P$.

2.1.5 BIT INDEPENDENCE

In a typical DNN, a singe network outputs all the bits of the output code in its final layer. The errors in the individual bits that are made up by a DNN or an ensemble method are often correlated and input causing an error in one particular output often correlates to errors in other outputs. Such correlations reduce the effective Hamming distance between codewords since the dependent error process means that multiple bit flips are likely to co-occur. Therefore, promoting diversity across the constituent learners is crucial and is generally a priority in ensemble-based methods. For ECOCs, each ensemble member j solves a different classification problem (specified by the jth column of $\mathbf{C}$). Thus we find that it is sufficient to simply train an ensemble of networks, where each member outputs $B \leq N$ bits (neurons) of the output code.

2.2 LEARNABLE OUTPUT CODES

Previously we assumed a predetermined set of codewords for class representations. We describe our framework to jointly learn the class representation and the model parameters.

2.2.1 LABEL ENCODER

The framework described previously implemented the error correcting output codes which are well-separated and hence used Hadamard codes and we do not really know whether maximizing the separation give us the best performances. The label encoder takes the one-hot representation of a class-label and maps it to dense distributed vector representations. Mathematically, given a one-hot representation of a class label y, the label encoder $\mathbf{L}_\theta(.)$ converts y into a d-dimensional vector representation $\mathbf{L}_\theta(y)$. The

objective of the label encoder is to learn the implicit structural similarities between the classes in the dataset. For example, in the CIFAR-10 dataset, we expect the class representation of automobiles and trucks to be more similar to each other than automobiles and horses.

2.2.2 INPUT ENCODER

The input encoder converts an input image into dense vector representation. Mathematically, given an input $\mathbf{X}$, the input encoder $\mathbf{I}_\phi(.)$ converts $\mathbf{X}$ into a d-dimensional vector representation $\mathbf{I}_\phi(\mathbf{X})$. The role of input encoder is to extract relevant features from a given image that will be useful for its classification.

We use the previously proposed idea to calculate the output probabilities. The probability function now becomes,

$$p_\sigma(y|\mathbf{X},\theta,\phi) = \frac{max(\mathbf{I}_\phi(\mathbf{X})\cdot\mathbf{L}_\theta(y),0)}{\sum_{i=1}^{M} max(\mathbf{I}_\phi(\mathbf{X})\cdot\mathbf{L}_\theta(i),0)} \tag{4}$$

The overall loss function now becomes,

$$L(y,\mathbf{X},\theta,\phi) = -(yp_\sigma(y|\mathbf{X},\theta,\phi) + (1-y)(1-p_\sigma(y|\mathbf{X},\theta,\phi))) \tag{5}$$

We mention the training procedure to jointly learn the parameters of the label encoderθ and input encoder ϕ

Algorithm 1: Training procedure to jointly learn label encoder θ and input encoder ϕ

for *number of training iterations* **do**

  for *k steps* **do**

    sample minibatch of m examples $\{(\mathbf{x}^{(1)}, y^{(1)}) \ldots (\mathbf{x}^{(m)}, y^{(m)})\}$ from training. data.

    Update the input encoder's parameters using its stochastic gradient:

$$-\nabla_{\phi}\Big(yp_{\sigma}(y|\mathbf{X}, \theta, \phi) + (1-y)(1-p_{\sigma}(y|\mathbf{X}, \theta, \phi))\Big)$$

  end

  for *j steps* **do**

    Sample minibatch of n examples $\{(\mathbf{x}^{(1)}, y^{(1)}) \ldots (\mathbf{x}^{(n)}, y^{(n)})\}$ from training. data.

    Update the label encoder's parameters using its stochastic gradient:

$$-\nabla_{\theta}\Big(yp_{\sigma}(y|\mathbf{X}, \theta, \phi) + (1-y)(1-p_{\sigma}(y|\mathbf{X}, \theta, \phi))\Big)$$

  end

end

2.2.3 REGULARIZATION

We now add regularization to the existing label encoder which aims to minimize the Euclidean distance between $\mathbf{L}(y)$ and $\mathbf{H}(y)$, where $\mathbf{L}(.)$ is the label encoder, y is one-hot representation of class label and $\mathbf{H}(y)$ is the corresponding Hadamard code vector representing that class and the distance is given as:

$$d(\mathbf{L}(y), \mathbf{H}(y)) = \|\mathbf{L}(y) - \mathbf{H}(y)\| \tag{6}$$

This regularization will let the codes to be able to learn the implicit structural similarities between different classes while being close to hadamard codes in Euclidean space. The loss function for label encoder changes as follows:

$$L'(y, \mathbf{X}, \theta, \phi) = -(yp_\sigma(y|\mathbf{X}, \theta, \phi) + (1-y)(1-p_\sigma(y|\mathbf{X}, \theta, \phi))) + \lambda \frac{\sum_{i=1}^{M} d(\mathbf{L}(i), \mathbf{H}(i))}{M}$$

λ is the regularization hyperparameter. The training procedure changes as follows:

Algorithm 2: Training procedure to jointly learn label encoder θ and input encoder ϕ

---

for *number of training iterations* **do**

  for *k steps* **do**

    sample minibatch of m examples $\{(\mathbf{x}^{(1)}, y^{(1)}) \dots (\mathbf{x}^{(m)}, y^{(m)})\}$ from training. data.

    Update the input encoder's parameters using its stochastic gradient:

$$-\nabla_\phi \Big(yp_\sigma(y|\mathbf{X}, \theta, \phi) + (1-y)(1-p_\sigma(y|\mathbf{X}, \theta, \phi)) \Big)$$

  end

  for *j steps* **do**

    Sample minibatch of n examples $\{(\mathbf{x}^{(1)}, y^{(1)}) \dots (\mathbf{x}^{(n)}, y^{(n)})\}$ from training. data.

    Update the label encoder's parameters using its stochastic gradient:

$$-\nabla_\theta \left(yp_\sigma(y|\mathbf{X}, \theta, \phi) + (1-y)(1-p_\sigma(y|\mathbf{X}, \theta, \phi)) - \lambda \frac{\sum_{i=1}^{M} d(\mathbf{L}(i), \mathbf{H}(i))}{M} \right)$$

  end

end

---

3. EXPERIMENTS

We conduct experiments with a series of models which vary the choice of code C, the length of the codes N, and the activation function applied to the logits. Our training and adversarial attack procedures are standard. Table 1 summarizes the various models used.

Model	Architecture	Code	Activation
Softmax	Standard	$\mathbf{I}_{10}$	softmax
Logistic	Standard	$\mathbf{I}_{10}$	logistic
Tanh16	Standard	$\mathbf{H}_{16}$	tanh
LogisticEns10	Ensemble	$\mathbf{I}_{10}$	logistic
TanhEns16	Ensemble	$\mathbf{H}_{16}$	tanh
TanhEns32	Ensemble	$\mathbf{H}_{32}$	tanh
Learnable16	Standard	Learnable	tanh
Learnable16($\lambda = 0.3$)	Standard	Learnable	tanh

Table 1: Characterization of various models that were tested

“Standard” refers to a standard convolutional architecture with a dense fully connected output layer while ”ensemble” refers to the setup described in Section 2.1.5 . λ for learnable desribes the value of regularization hyperparameter used. Table 2 shows the results of our experiment on MNIST and Table 3 shows the results of our experiment on CIFAR-10. All models are trained for E epochs using the Adam optimizer and a learning rate of 2e-4, where E = 150 for MNIST dataset and E = 400 for CIFAR-10 dataset. To prevent overfitting , we add zero-mean gaussian noise with standard deviation 0.3 for MNIST and 0.032 for CIFAR-10. We also use standard data augmentation (flipping, rotating and shifting images). For softmax models, we use the standard cross-entropy loss. For logistic models, we use the binary cross-entropy loss on a per output neuron basis. For tanh models, we which implement fixed hadamard codes we use (SVM) hinge loss on a

per output basis. The column 2 shows the accuracy of test sets. The remaining columns show results on various attacks; all such results are in the white-box setting (adversary has full access to the entire model). Columns 3 and 4 show results for the projected gradient descent (PGD, $\epsilon = 0.3$) and Carlini-Wagner (CW) attacks [2], respectively, These columns show the fraction of adversarially crafted inputs which the model correctly classifies, i.e., examples which fail to be truly adversarial. Column 5 contains results of the "blind spot attack" [3], which first scales images by a constant α close to 1 before applying the Carlini Wagner attack. Column 6 shows the fraction of random inputs for which the model's maximum class probability is smaller than 0.9; here, a random input is one where each pixel is independently and uniformly chosen in (0,1). Column 7 shows the accuracy on test inputs where each pixel is independently corrupted by additive uniform noise in $[-\gamma, \gamma]$, where $\gamma = 1$ for MNIST and 0.1 (CIFAR-10) and clipped to lie within the valid input range, e.g.(0,1).

4. CONCLUSION

The approach to improve model robustness is centered around three core ideas. i) Moving from softmax to sigmoid decoding so that non-trivial volume of the Euclidean logit space is allocated. In crafting convincing adversarial, perturbation, the adversary must now guard against landing in such regions.i.e the attack surface is smaller. ii) Changing the codewords from $\underline{I}_M$ to one with larger Hamming distance so that the Euclidean distance in logit space between any two regions of high probability for any given class becomes larger. The adversary's perturbation has to be larger in magnitude to attain same level of confidence.

Model	Benign	PGD	CW	BSA $\alpha = 0.8$	Rand	Uniform
Softmax	0.9905	0.09	0.53	0.18	0.24	0.71
Logistic	0.9911	0.09	0.56	0.21	0.68	0.82
Tanh16	0.9916	0.34	0.79	0.28	0.64	0.79
LogisticEns10	0.9913	0.31	0.84	0.49	0.91	0.77
TanhEns16	0.9922	0.87	0.99	0.97	0.99	0.84
TanhEns32	0.9930	0.88	0.99	0.99	0.99	0.86
Learnable16	0.9924	0.34	0.70	0.19	0.69	0.80
Learnable16($\lambda = 0.3$)	0.9929	0.26	0.72	0.20	0.67	0.81

Table 2: Accuracies of models trained on MNIST against various attacks

Model	Benign	PGD	CW	BSA $\alpha = 0.8$	Rand	Uniform
Softmax	0.866	0.08	0.11	0.07	0.41	0.78
Logistic	0.859	0.10	0.13	0.10	0.47	0.80
Tanh16	0.862	0.34	0.16	0.09	0.72	0.84
LogisticEns10	0.868	0.08	0.15	0.13	0.48	0.84
TanhEns16	0.884	0.55	0.75	0.77	0.96	0.85
TanhEns32	0.891	0.59	0.77	0.78	0.97	0.87
Learnable16	0.871	0.13	0.17	0.13	0.56	0.82
Learnable16($\lambda = 0.3$)	0.877	0.10	0.16	0.14	0.60	0.82

Table 3: Accuracies of models trained on CIFAR-10 against various attacks

iii) Learning output bits with multiple disjoint networks, we reduce correlations between outputs. Such correlations are implicitly capitalized on by common attack algorithms. We then propose a framework to learn the output codes instead of using a fixed set of output codes. Our framework consists of an input encoder and a label encoder and we jointly learn their parameters to minimize the cross-entropy loss. On the MNIST and CIFAR-10 datasets, we show that our proposed framework outperforms the baseline softmax and Logistic models on both normal and adversarial inputs. In comparison to the Tanh model with hadamard code, while our proposed frame-

work performs better on normal inputs, its performance is lower on the adversarial inputs. We further added regularization which tends to minimize the distances between the output codes and Hadamard codes. The results were similar to learned output codes with no regularization. One important avenue to study further is to consider datasets of larger input dimensionality, such as ImageNet. It may be possible that in very high input dimensions, adversarial perturbations exist that can still surmount the larger Hamming distances afforded by ECOCs.

References

[1] Gunjan Verma and Ananthram Swami. Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks. In H. Wallach, H. Larochelle,*Advances in Neural Information Processing Systems* 32, pages 8646-8656. Curran Associates, Inc., 2019.

[2] N.carlini and D.Wagner, "Towards evaluating the robustness of neural networks." in 2017 *IEEE symposium on Security and Privacy*, 2017, pp.39-57

[3] H.Zhang, H. chen, Z. song, D. Boning, I. S. Dhillon and C.J. Hsieh,"The limitations of adversarial training and the blind-spot attack,"*arXiv preprint arXiv:1901.04684*, 2019.

[4] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572,2015

[5] Christian Szegedy, Wojciech Zaremba, Ilya Suskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neutral networks. *CoRR*, abs/1312.6199, 2014.