Uncertainty Quantification and Deep Ensembles

(Seminar: Intriguing properties of neural networks)

Hepsiba Komati
Technische Universität Dortmund
Dortmund, Germany
hepsiba.komati@tu-dortmund.de

ABSTRACT

Models using deep neural networks are quite expressive. Although their expressiveness is what makes them successful, it also leads them to learn puzzling solutions that could have counter intuitive characteristics. We describe two such features in this paper.

The first property is semantic information in neurons, the assumption is that the neurons separate features however we find that there is no distinction between individual neurons according to various methods of unit analysis

Second property is in adversarial attacks the classifiers are robust, however we conclude that adversarial examples do exist when we introduce perturbation to the original image which leads to network prediction error. These perturbations are not random learning artifacts; instead, they might lead multiple networks that were trained on different subsets of the dataset to incorrectly categorize the same input.

1 INTRODUCTION

Deep neural networks are powerful learning models that have demonstrated impressive performance across visual and speech recognition problems.[3] The multiple layers of deep learning models are parallel to one another and have non-linear relationships. The main property of deep learning is that it is able to identify and extract the features automatically through back propagation. However, we are unaware of and incapable of what is occurring inside the model. As a result, it is exceedingly challenging to interpret the model and it can also have counter intuitive properties. In this paper, we will discuss two such counter intuitive properties in the deep neural network. [4]

The first property is the semantic information of individual neuron. In other words, what kind of features do these neurons react to or activate upon. This is done by finding and inspecting the images in the data sets which maximizes that particular neuron that is particularly beneficial for extracting semantic information.In later section we see that the activation value for any particular layer $\phi(x)$ are identical to the coordinates of in terms of semantics which means the majority of the semantic information is contained in the total space of activation rather than in the individual units.The individual units of the vector representations are unlikely to contain semantic information because they are well-defined up to a rotation of the space.

The second one is related to stability of the neural network with respect to small variations in their inputs (perturbation). Adversarial examples are specialized inputs designed to confuse a neural network and cause it to misclassify a particular input. These inputs are undetectable to the human eye, but they prevent the network from correctly identifying the image's contents. So, when the attacker adds small perturbations (distortions) to the original image, which results in the model labelling this image differently with high confidence with increasing networks prediction error. The process of adding these perturbations leads to adversarial attacks. Another network that was trained on a different portion of the dataset may incorrectly categorize the same input as a result of the same perturbation.

It's assumed that in adversarial examples the classifiers are fairly robust. When the adversarial examples are generated on one particular model and transferred to different models or if the models are trained on subsets of data and tested on different subsets of data the examples are transferred well. If we utilize one neural network to create a set of adversarial examples, we discover that these examples are still statistically challenging for a second neural network, even if it was trained with a different set of hyper parameters. We discover that deep neural networks effectively learn input-output mappings by back-propagation have non intuitive characteristics that are discontinuous.

2 FRAMEWORK

Though out the paper we refer unit as a single neuron,input image by $x \in R^m$ and the activation value for any particular layer is represented as $\phi(x)$. Firstly, we evaluate the characteristics of the original image $\phi(x)$ later look for blind spots. Here, we perform multiple experiments on numerous networks and three different datasets.

- Below architecture is used for MNIST dataset
  - A simple neural network with one or more hidden layers and a SoftMax function in the final layer.This network is referred as "FC".
  - A classification model is trained on the top of the autoencoder. We refer to this as "AE".
- The ImageNet dataset referred as AlexNet [1]
- 10M image samples from YouTube, which is unsupervised trained network with over 1 billion learnable parameters. We refer to it as "QuocNet" [2]

Later in few experiments the dataset MNIST is divided into disjoint datasets P_1 and P_2 and regularization parameter is considered as λ

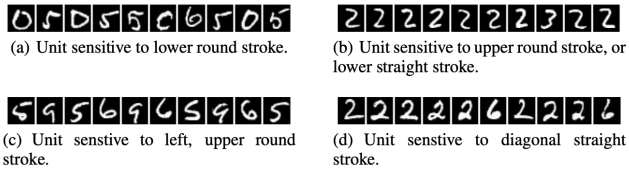
3 UNIT LEVEL INSPECTION $\phi(x)$

Using the fundamental matplotlib package and customized coding, we can employee feature extraction in classic computer vision approaches. For instance, we can extract a certain color intensity

Seminar: Intriguing properties of neural networks,

or line direction. But a deep learning model comprises plenty of layers and units. Therefore, until retrieved individually, we cannot tell which layer is recognizing what the feature and technique is.

Here, we specify the feature vector space from which we select set of images. A natural basis vector (a basis vector with amplitude in just one direction) and a random basis vector are the two types of feature vectors we will now utilize (which has random direction). These tests are run on the test dataset of the MNIST dataset (pictures that the model had not previously seen), and the results are listed below.



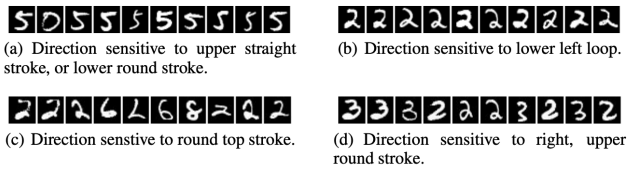
(a) Unit sensitive to lower round stroke.

(b) Unit sensitive to upper round stroke, or lower straight stroke.

(c) Unit senstive to left, upper round stroke.

(d) Unit senstive to diagonal straight stroke.

Figure 1: Maximum stimulation in the natural basis direction



(a) Direction sensitive to upper straight stroke, or lower round stroke.

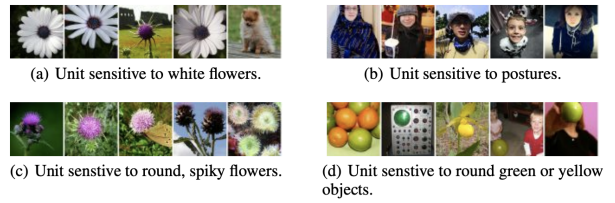
(b) Direction sensitive to lower left loop.

(c) Direction senstive to round top stroke.

(d) Direction sensitive to right, upper round stroke.

Figure 2: Maximum stimulation in the random basis direction

Repeating the experiment with the AlexNet dataset,to find out if both vector spaces can incorporate semantic information into their individual units. The outcomes are listed below.



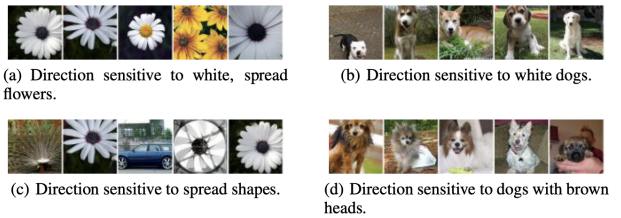
(a) Unit sensitive to white flowers.

(b) Unit sensitive to postures.

(c) Unit senstive to round, spiky flowers.

(d) Unit senstive to round green or yellow objects.

Figure 3: maximum stimulation in natural basis direction



(a) Direction sensitive to white, spread flowers.

(b) Direction sensitive to white dogs.

(c) Direction sensitive to spread shapes.

(d) Direction sensitive to dogs with brown heads.

Figure 4: maximum stimulation in random basis direction

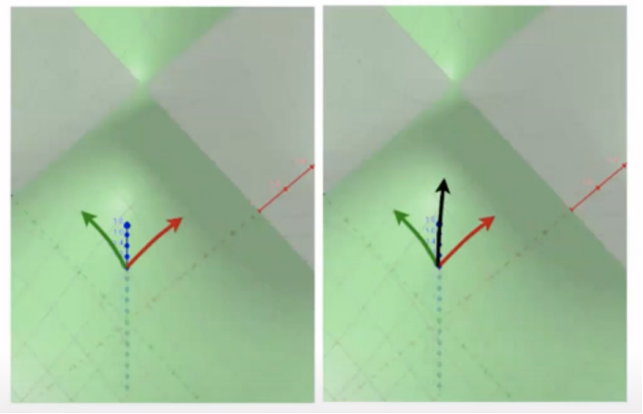
The findings above show that both of these strategies perform quite similarly to one another. They are able to locate the necessary items. Mathematically the method mentioned above is formally known as visual inspection of image $x^{'}$ which satisfy

$$x^{'} = \underset{x \in \text{image set}}{\arg\max} \langle \phi(x), e_i \rangle$$

However, analysis shows that any random direction generates semantic features that are understandable. That are also semantically connected, more technically, that maximize the activation's in a random direction.

$$x^{'} = \underset{x \in \text{image set}}{\arg\max} \langle \phi(x), v \rangle$$

This suggests that when examining properties, a natural basis is not preferable to a random basis properties of $\phi(x)$.Although this analysis gave some insights of ϕ to generate in-variance on particular subset of input ,however it doesn't explain the behavior of its domain.



From a mathematical standpoint, the individual units are incomplete. If we take Red to be unit 1 and Green to be unit 2 in the image above, then by focusing on only one of those units, we can obtain images that may be close to local maxima. The combination of both units, however, would be the most practical local maximum; the key characteristic would be conforming to the black arrow. Although it is possible to have local maxima, the network's size makes it unlikely that they will line with the axis. In view of the fact that we achieve the same outcomes when we maximize one unit as opposed to several, semantic information may be distributed throughout the units rather than being unique to any one of them. So, using this approach to analyze individual units is not recommended.

3.1 Blind spot in neural network

This earlier mentioned method helps us find what a particular unit is doing in the deep neural network. However, it receives very little support when we think of a genuinely deep network. To ascertain which properties are being captured, we can alternatively utilize a trained model on datasets that have previously been categorized. The deep neural networks' nature is nonlinear. The produced output

is largely incoherent with the photos used as input. Each layer has a unique nonlinear activation function with a unique value.

This phenomenon is called non - local generalization prior. When viewed from a distance, images that are categorized in one domain will not produce the same results (zooming the image). Adversarial examples are those images created by minor adjustments that the trained models are unable to recognize. These cases are difficult for the model to detect and have a low probability. This is used by many of the current computer vision frameworks to increase the model's effectiveness and speed of convergence. This method is known as data augmentation.

This can be compared to hard-negative mining, where the goal is to identify the set of cases that the model gives low probability to but that actually deserve high likelihood. The training set is retrained using the augmented data with additional priotization. We have proposed a optimization method similar to hard - negative mining.

4 ATTACK ALGORITHM

L-BFGS (limited memory Broyden Fletcher Goldfarb Shannno optimizing algorithm) is a slow, targeted and a white box algorithm.White box is a model whose inner workings, logic, and coding are transparent and whose decision-making process is consequently comprehensible. When the perturbation is introduced to a clean image x and sent to the model $f : R^m \longrightarrow \{1...K\}$, the result is class $l \in \{1....K\}$, which is not the original label. In an adversarial scenario utilizing box-constrained L-BFGS we work to change the label in a way that minimizes the perturbation. We must ensure that the original image and any disturbance always fall between 0 and 1.

$$1.\ f(x+r) = l$$

$$2.\ x + r \in [0,1]^m$$

The minimizer r could not be the only one, there could be several of these perturbations for arbitrarily chosen minimizer by $D(x, l)$. Due of this, it becomes quite difficult to solve the problem. Instead, combining two constraints to approximate the solution.

$$Minimizec|r| + \text{loss}_f(x+r, l) \text{ subject to } x + r \in [0,1]^m$$

One common loss function to use is cross-entropy. Line search is performed to find the constant c > 0 that yields an adversarial example of minimum distance: in other words, we repeatedly solve this optimization problem for multiple values of c, adaptively updating c using bisection search or any other method for one-dimensional optimization. This algorithm is employed in a number of models to demonstrate the existence of adversarial examples.

5 IMPLEMENTING THE ALGORITHM AND EXPERIMENTING THE RESULTS

The minimum distortion function in the model has the following intriguing properties

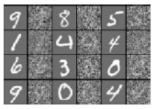
1. We were able to produce adversarial examples that are misclassified by the original model for each sample for all three of the aforementioned datasets and samples.[1]
2. Cross-Model Generalization: Networks trained from beginning with various hyper-parameters will misclassify a sizable portion of samples (number of layers, regularization or initial weights). In other words, even a small change to the original model causes incorrect results.
3. Generalization across training sets: Networks trained from scratch on a discontinuous training set will misclassify a sizable portion of samples. In other words, if the training dataset is changed, the model will produce incorrect classification results.

This explains how generalization of the model is improved by training the images on adversarial data. We have successfully trained a two-layer network with 100 layers on the input layer and 100 layers on the hidden layer using the MNIST dataset. Typically, we don't think of the output layer as a network layer. By using a pool of adversarial data in place of the existing data, we were able to obtain a test error of just 1.2%. We have use weight decay but not dropout.

For comparison, a similar-sized dataset trained solely with weight decay results in an error rate of 1.6%. By carefully dropping out, it can be raised to 1.3%. This demonstrates how using hostile examples reduces error. Adding adversarial data to higher order layers has been found to have better results than doing so with lower order layers. The examples of these created adversarial instances for the MNIST dataset are shown below.

(a) Even columns: adversarial examples for a linear (FC) classifier (stddev=0.06)

(b) Even columns: adversarial examples for a 200-200-10 sigmoid network (stddev=0.063)



(c) Randomly distorted samples by Gaussian noise with stddev=1. Accuracy: 51%.

Figure 5: Adversarial examples for MNIST compared with randomly distorted examples

The original images are shown in odd columns. The accuracy of the adversarial examples created for the particular model is 0%. As the adversarial instances are never correctly identified, the randomly deformed examples are barely recognizable yet are nevertheless correctly classified in half of the cases. Even columns correlate to their warped equivalents, whereas odd columns belong to the original images.

Model Name	Description	Training error	Test error	Av. min. distortion
softmax1	Softmax with $\lambda = 10^{-4}$	6.7%	7.4%	0.062
softmax2	Softmax with $\lambda = 10^{-2}$	10%	9.4%	0.1
softmax3	Softmax with $\lambda = 1$	21.2%	20%	0.14
N100-100-10	Sigmoid network $\lambda = 10^{-5}, 10^{-5}, 10^{-6}$	0%	1.64%	0.058
N200-200-10	Sigmoid network $\lambda = 10^{-5}, 10^{-5}, 10^{-6}$	0%	1.54%	0.065
AE400-10	Autoencoder with softmax $\lambda = 10^{-6}$	0.57%	1.9%	0.086

Figure 6: Tests of the generalization of adversarial instances on MNIST

The first three models are linear models with different weight decay parameters that operate at the pixel level. The last three models are straightforward linear models (SoftMax) devoid of any secret layers. One of them has a learning parameter of 1 which is incredibly high. Simple sigmoidal neural networks with two

hidden layers and an output layer make up two of these models. The final layer consists of 400 nodes with a SoftMax classifier and a single layer sparse autoencoder with sigmoidal activation function. From the below table we can notice that the larger the lambda the larger the perturbations and that increases the average minimum distortion. On the hand larger the lambda means less accuracy in clean images,that is the test error keeps increasing. The following formula calculates distortion by using the differences between the original and altered images as the numerator.

$$\sqrt{\frac{\sum (x_i^{'} - x_i)^2}{n}}$$

In the same experiment, we generated a set of adversarial examples and added them to the training set to measure the instances of misclassified images. We transfer these adversarial examples to different models. To support these let us consider the six models from the table the final column displays the least distortion necessary to obtain 100% accuracy across the training set. Adversarial images on same model turned to be 100% which is consistent across all the models. When the adversarial models are sent to different models sometimes adversarial examples do transfer but it is not always consistent. So, we can draw the conclusion that adversarial are more difficult to find for models with various hyper parameters. However, although the autoencoder model performs better, it is still not competitive enough. The results of Cross-model generalization performed on MNIST dataset are given below.

	softmax1	softmax2	softmax3	N100-100-10	N200-200-10	AE400-10	Av. distortion
softmax with $\lambda = 10^{-4}$	100%	11.7%	22.7%	2%	3.9%	2.7%	0.062
softmax with $\lambda = 10^{-2}$	87.1%	100%	35.2%	35.9%	27.3%	9.8%	0.1
softmax with $\lambda = 1$	71.9%	76.2%	100%	48.1%	47%	34.4%	0.14
N100-100-10	28.9%	13.7%	21.1%	100%	6.6%	2%	0.058
N200-200-10	38.2%	14%	23.8%	20.3%	100%	2.7%	0.065
AE400-10	23.4%	16%	24.8%	9.4%	6.6%	100%	0.086
Gaussian noise, stddev=0.1	5.0%	10.1%	18.3%	0%	0%	0.8%	0.1
Gaussian noise, stddev=0.3	15.6%	11.3%	22.7%	5%	4.3%	3.1%	0.3

Figure 7: Cross-model generalization of adversarial examples

Now we divid the total MNIST training set of 60000 images into two subsets P_1 and P_2 each with a size of 3000, and trained three non-convolutional networks with sigmoid activation on them: Two FC100-100-10 and FC123-456-10, on P1, and FC100-100-10 on P2. This allowed us to study the final property, the cross-training set generalization. To examine the overall impact of simultaneously modifying the hyperparameters and the training sets, we trained two networks for P1. The hyperparameters for the models FC100-100-10 and FC100-100-10 are the same; they are both 100-100-10 networks, however FC123-456-10 has a different number of hidden units. In this experiment, the test set rather than the training set is being distorted. The following figure 8 lists the model's fundamental details.

After generating adversarial examples with 100% error rates and minimal distortion for the test set, we feed them on to the training set. The results of the experiments are given in figure 9.

The intriguing conclusion is that adversarial data are even difficult for model trained on disjoint dataset.But the model is considerably better than others.

Model	Error on P_1	Error on P_2	Error on Test	Min Av. Distortion
M_1: 100-100-10 trained on P_1	0%	2.4%	2%	0.062
$M_1^{'}$: 123-456-10 trained on P_1	0%	2.5%	2.1%	0.059
M_2: 100-100-10 trained on P_2	2.3%	0%	2.1%	0.058

Figure 8: Models trained to study cross-training-set generalization of the generated adversarial examples

	M_1	$M_1^{'}$	M_2
Distorted for M_1 (av. stddev=0.062)	100%	26.2%	5.9%
Distorted for $M_1^{'}$ (av. stddev=0.059)	6.25%	100%	5.1%
Distorted for M_2 (av. stddev=0.058)	8.2%	8.2%	100%
Gaussian noise with stddev=0.06	2.2%	2.6%	2.4%
Distorted for M_1 amplified to stddev=0.1	100%	98%	43%
Distorted for $M_1^{'}$ amplified to stddev=0.1	96%	100%	22%
Distorted for M_2 amplified to stddev=0.1	27%	50%	100%
Gaussian noise with stddev=0.1	2.6%	2.8%	2.7%

Figure 9: Cross-training-set generalization error rate for the set of adversarial examples generated for different models.

6 ADVERSARIAL ATTACK MATHEMATICAL BASIS

Consider input x and weight matrix W and let ϕ denote the output of particular layer.When we send the input image x with parameter W for layer one the resulted output would be $\phi_1(x)$ and this output is sent to second layer which results as $\phi_2(x)$ and so on. With K layers, the neural network's output would be as specified in the equation for$\phi(x)$. The Lipschitz constant is used to explain why neural networks are unstable. We compute the difference in output at a certain layer of the neural network for a specific picture x and perturbation r. When given a clean and perturbed image, the output difference is always constrained by L times r, where L is the Lipschitz constant and r is the perturbation. Each layer's Lipschitz constants combine to form the Lipschitz constant.We generalize the formula to account for all k layers, and the results are shown below.

$$\phi(x) = \phi_k(\phi_{k-1}(...\phi_1(x; W_1); W_2)...; W_k)$$

$$\forall x, r, ||\phi_k(x; W_k) - \phi_k(x + r; W_k|| \leq L_k||r||$$

$$||\phi(x) - \phi(x + r)|| \leq L||r||, \text{with} L = \prod_{k=1}^{K} L_k$$

With x and y serving as the axis,the purple line in Figure 10 represents the function f. If a function is lipschitz continuous, the slope of the function is a constant.The slope that is constrained by the lipschitz constant at points x and $x + r$.When we plot these two lines we obtain the graph as shown in Figure 10.The upper bound would be positive and the lower bound would be negative.The largest slope will comprehend the Lipschitz constant.

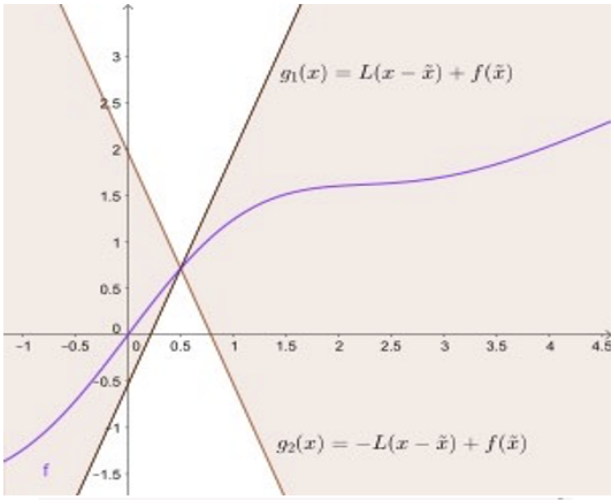


Figure 10: Lipschitz Continuity

If W denotes a generic 4-tensor convolutional layer with stride Δ ,

$$Wx = \left\{ \sum_{c=1}^{C} x_c \star w_{c,d}(n_1\nabla, n_2\nabla); d = 1..., D \right\},$$

where x_c denotes the c input feature image, then we can verify that its operator norm as

$$||W|| = \sup_{\xi \in [0, N\Delta^{-1})^2} ||A(\xi)||$$

where $A(w)$ is the matrix whose rows are

$$\forall\, d = 1 \ldots D,\ A(\xi)d = \left(\Delta^{-2}\widehat{wc, d}(\xi + l \cdot N \cdot \Delta^{-1}); c = 1 \ldots C, l = (0 \ldots \Delta - 1)^2\right)$$

Layer	Size	Lower bound	Upper bound
Conv. 1	$3 \times 11 \times 11 \times 96$	0.1	20
Conv. 2	$96 \times 5 \times 5 \times 256$	0.06	10
Conv. 3	$256 \times 3 \times 3 \times 384$	0.01	7
Conv. 4	$384 \times 3 \times 3 \times 384$	$5 \cdot 10^{-5}$	7.3
Conv. 5	$384 \times 3 \times 3 \times 256$	0.03	11
FC. 1	9216×4096	0.09	3.12
FC. 2	4096×4096	10^{-4}	4
FC. 3	4096×1000	0.25	4

Table 5: Frame Bounds of each convolutional layer

Above table shows the upper and lower bounds computed from the ImageNet deep convolutional network.When a neural network is created, the Lipschitz constant is tested and calculated at each layer. First layer's Lipschitz constant is 2.75, second layer's is 10, and third layer's is 7. The Lipschitz constant is the product of all constants when we have to compute it for the entire network. The instabilities, with an upper bound of 2.75, are observed in the network's very early phases, according to the conclusion. The upper bound just establishes that there are instabilities rather than conclusively demonstrating the existence of adversarial situations. However, by lowering the upper bound, the stability's can be reduced.

7 STRENGTHS

- Good explanation for high level understanding
  - Relating to core understanding of DNN which helps to apply the knowledge
- Simple unit experiment design for property 1
  - Strong refutation of method
- Clear quantitative results
- Wide-spread for the attacks
  - Transferability for classification networks for critical systems
- Useful mathematical analysis for further research
  - Potential to reduce attack effectiveness

8 WEAKNESSES

- Two Proposed intriguing properties seem to be disjoint,both the properties don't connect to each other well.
- Random combination of units is not well explained, as in if it's totally random or if it follows any specific distribution.
- Weak experiments supporting semantic information property.
- Adversarial input generation algorithm is not detailed.
- Results from adversarial training are insignificant,that is when adversarial samples are sent back to the network along with the original clean images to check if it learns better,however these results seems insignificant.It's just an improvement of 0.1% from the paper.
- little evidence for extremely low probability of adversarial inputs.

9 THOUGHTS

- Issue of rugged decision boundary is not addressed
  - Adversarial training does not seem to be very effective
- Can we use the same optimization to retrieve original label?
- If the number of classes is small,will the problem be less severe

10 SUMMARY

This study comes to the conclusion that adversarial attacks raise concerns about robustness of the classification network. Adversarial attacks can be applied to many models even if there is no data overlap when doing so,the attacks do transfer to different models and different networks,however they are not always consistent.The existence of attacks has a mathematical foundation to back up.The first section's conclusion stated that the units share semantic information. Individual unit analysis is challenging. Therefore in this research we have seen that the deep neural networks have both counter intuitive qualities and the semantic meaning of both their discontinuities and individual units. Additionally, we discovered that data including hostile negatives provides relatively little likelihood. The likelihood of its occurrence may be rather low. As a

result, the model might not be able to apply to such data. But in order to understand the true functioning behavior for this, further investigation is needed.

REFERENCES

[1] Krizhevsky Alex, Sutskever Ilya, and Hinton Geoff. 2012. In Advances in Neural Information Processing Systems. (2012).

[2] Jia Deng, Wei Dong, Richard Socher, Li Li-Jia, Li-kai, and Li Fei-Feir. 2009. In Computer Vision and Pattern Recognition. (2009).

[3] Hinton Geoffrey, Deng Li, Yu Dong, Dahl George, Mohamed Abdel, rahman, Jaitly Navdeep, Senior Andrew, Vanhoucke Vincent, Nguyen Patrick, Sainath Tara, and Kingsbury Brian. 2012. Deep neural networks for acoustic modeling in speech recognition:. (2012).

[4] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. (12 2013).