Robust Deep Neural Networks Inspired by Fuzzy Logic

Minh Le

Vrije Universiteit Amsterdam, Netherlands

m.n.le@vu.nl

Abstract

Deep neural networks have achieved impressive performance and become the de-facto standard in many tasks. However, troubling phenomena such as adversarial and fooling examples suggest that the generalization they make is flawed. I argue that among the roots of the phenomena are two geometric properties of common deep learning architectures: their distributed nature and the connectedness of their decision regions. As a remedy, I propose new architectures inspired by fuzzy logic that combine several alternative design elements. Through experiments on MNIST and CIFAR-10, the new models are shown to be more local, better at rejecting noise samples, and more robust against adversarial examples. Ablation analyses reveal behaviors on adversarial examples that cannot be explained by the linearity hypothesis but are consistent with the hypothesis that logic-inspired traits create more robust models.

1 Introduction

The success of ReLU-based neural networks in recent years originates from their *learnability* and *expressiveness*. Piecewise linearity helps them avoid gradient exploding or vanishing problems of early architectures and, in terms of expressive power, they are shown to be able to capture an exponential amount of variations [Raghu *et al.*, 2017]. However, little can be guaranteed about the *validity* of the learned representations. Recent work has pointed out two troubling phenomena: adversarial examples [Szegedy *et al.*, 2013] and fooling examples [Nguyen *et al.*, 2015]. In the context of image recognition, the former are images slightly perturbed to fool a model while being semantically the same to human eyes. The latter, fooling examples, are images that do not belong to any class and yet are classified to one with high confidence. Both the results of Szegedy et al. [2013] and Nguyen et al. [2015] demonstrate that a model that is easy to train might also easily make invalid generalization.

Much research has been devoted to explaining these phenomena [Goodfellow *et al.*, 2015; Ford *et al.*, 2019, among others]. Recently, Nguyen et al. [2018] identifies the tendency of ReLU networks to create connected classification regions as one likely source of adversarial examples.

With regards to fooling examples, the networks' distributed nature is likely to be among the causes. The activation of a ReLU unit gets stronger the *further* the input is from its decision boundary. Noise patterns are typically far from decision boundaries and therefore also likely to induce a high level of activation. A more principled approach would be to make *local* generalizations in which a model makes confident predictions within bounded regions of the input space and gradually becomes less confident further away. Even though removing the locality assumption is among motivations to develop deep architectures [Bengio and Delalleau, 2011], it will be shown here that we can have one that is both deep and local.

To design a local and disconnected deep neural network, I take inspiration from fuzzy logic. Image classification can be cast as learning a system of propositions that take pixel intensities as input and produce the likelihood of each class being present. Given the complexity of the task, it is reasonable to expect those propositions to be expressed in terms of concepts which, in turn, are expressed by sub-concepts until we reach the input, for example:

$$
\begin{aligned}
\text{class}_1 \quad &\leftarrow \quad \text{concept}_1 \vee \text{concept}_2 \\
\text{class}_2 \quad &\leftarrow \quad (\text{concept}_2 \wedge \text{concept}_4) \vee \text{concept}_3 \\
\cdots \\
\text{concept}_n \quad &\leftarrow \quad (x_{0,0} \wedge x_{0,1}) \vee (\neg x_{0,1} \wedge x_{1,1} \wedge x_{1,2})
\end{aligned}
$$

The hierarchical layout and alternating pattern of ANDs and ORs above are essential for expressiveness because they compactly express an exponential number of cases. Concepts, as specified by fuzzy propositions, can be local and disconnected in the sense that they admit a typically small subset of the input space (e.g. dogs among all visible objects) and can combine arbitrary subconcepts into one (e.g. to account for the diverse appearances of birds, from robins to ostriches to penguins). It will be shown in next sections that it is possible to incorporate such abilities into a modern neural network architecture and doing so helps alleviate fooling and adversarial examples.

The rest of paper is organized as follows: modifications to a ReLU network is proposed in Section 2; experimental procedures and results are detailed in Section 3 and 4; I will situate the research in the wider literature in Section 5 before concluding with some remarks in Section 6.
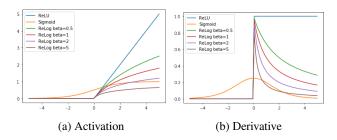
(a) Activation

(b) Derivative

Figure 1: Comparing ReLog with two popular activation functions

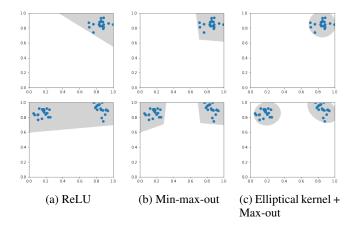(a) ReLU (b) Min-max-out (c) Elliptical kernel + Max-out

Figure 2: Different types of neurons differ in the ability to fit to the shape of the data. Blue dots represent data points and shaded regions are regions in the input space that creates positive activation in the target neuron.

2 Approximating Propositional Logic in Neural Networks

The task of emulating logic in a high-dimensional space carries with it inherent difficulty. To balance learnability and capacity, I start with ReLU-based CNNs and make amendments where they deviate from the ideals. The results are captured in this section as a series of re-considerations.

2.1 Rethinking Activation Functions

If we are to think of a neuron as representing a logic formula, constraints must be placed on its range of activation. Common sense suggests that a soft logic function ramps up its activation as more and more supporting evidence accumulates, but only to a certain point. In other words, it must exhibit a saturation region. Sigmoid achieves this by exponential decay but this has been shown to cause vanishing gradient problem.

To strike a balance between fidelity and learnability, I introduce rectified logarithmic function (Figure 1a):

$$\text{ReLog}(z; \beta) = \frac{1}{\beta} \log \left(\beta \max(z, 0) + 1 \right), \quad (1)$$

where $\beta > 0$.

As can be seen in Figure 1b, in the right half of the function, the derivative of ReLog peaks higher and stays meaningful for a much larger range compared to sigmoid. Another important property of ReLog is that it approximates ReLU when β is close to 0.

2.2 Rethinking Neurons

In the distributed representations framework, the individual neuron does not carry any particular meaning [Hinton *et al.*, 1986]. In contrast, a logical view considers the neuron as a function that returns true for a particular pattern and false for everything else.

The patterns depicted in each column of Figure 2 can be captured by logical conjunctions and disjunctions which translate into min and max in fuzzy logic [Belohlavek and Klir, 2011]. For this reason, I propose min-max-out units, an extension of maxout [Goodfellow *et al.*, 2013] in which incoming input is first passed through a min operation (representing AND) and then a max operation (representing OR):

$$\text{MinMaxOut}_{m,n}(x) = \max_{i=1}^{m} \min_{j=1}^{n} \sum_{k=1}^{p} w_{ijk} x_k \quad (2)$$

The result is a better fit as illustrated in Figure 2b. However, because the number of linear units needed to fit a point

cloud increases linearly with the number of dimensions, this solution might not scale to high-dimensional spaces. An alternative is to draw curved boundaries around the data points by applying the kernel trick:

$$z = \sum_{i=1}^{m} w'_i x_i^2 + \sum_{i=1}^{m} w_i x_i + b, \quad (3)$$

To create elliptical boundaries, the constraint $w'_i \leq 0 \ \forall i$ can be applied. Elliptical units eliminate the need for min-out but not max-out units as illustrated in Figure 2c.

2.3 Rethinking Regularization

Explicit regularization is a way to reduce overfitting. This is clear in the textbook case of fitting a polynomial curve where smaller weights can reign in curvature [Bishop, 2006]. However, it is not obvious if and how this applies to a ReLU-based neural network that is linear almost everywhere. The analysis in the previous section suggests an alternative view on regularization as margin modulation.

The distance between a ReLU unit's decision boundary and a training example in Cartesian space is:

$$d = \frac{|\sum_{i=1}^{m} w_i x_i + b|}{\sqrt{\sum_{i=1}^{m} w_i^2}}, \quad (4)$$

where $x \in \mathbb{R}^m$ is the input vector, $w \in \mathbb{R}^m$ and $b \in \mathbb{R}$ are the weights and bias of the neuron.

Similar to a support vector machines (SVM), the distance between the decision boundary and the closest example can be increased by minimizing $||w||$ [Bishop, 2006]. Notice that, different from SVM, only accepted patterns count because rejected ones lead to inactivation and zero gradient.

On the other hand, the decision boundary is removed from the coordinate origin by:

$$d_0 = \frac{|b|}{\sqrt{\sum_{i=1}^{m} w_i^2}}. \quad (5)$$

Noticing that the origin is on the same side as the accepted input pattern if $b > 0$ and on the opposite side if $b < 0$, it can

be easily shown that reducing b reduces the margin around the accepted pattern. Combining the two observations above gives us a way to control decision margin that I shall call max-fit for its ability to increase fitting to an input pattern:

$$r_{\text{max-fit}} = \frac{\gamma}{m} \sum_{i=1}^{m} |w_i| + \gamma'b \qquad (6)$$

where $\gamma, \gamma' \in \mathbb{R}^+$ govern the strength of regularization and m is the number of inputs. A similar formula can be defined for L_2 regularization.

2.4 Rethinking Training

The approximation would not be complete without the ability to reject examples that do not belong to any of the predefined classes (i.e. to compute a all-zero row in a truth table). To this end, I replace softmax with independent sigmoid activation and train directly on negative examples using binary cross-entropy (BCE) loss. I also experimented with mean square loss but found that it leads to slower convergence and lower accuracy. Because techniques to generate fooling examples are out of the scope of the current paper, I only experimented with overlaying two images from different classes. The images can be efficiently generated from training examples as a data augmentation step.

3 Experimental Design

Experiments were designed to validate the following hypotheses about a model that contains some of the proposed modifications:

H1 (learnability and expressiveness): the model can be trained to the same accuracy on a clean dataset.

H2 (locality): for natural examples, a class is predicted with high confidence which decreases as we go further away until all classes are assigned equal probabilities.

H3 (robustness): the model achieves a higher accuracy on adversarial examples compared to a ReLU network.

It is straightforward to compare models on clean and adversarial examples. To detect locality, I use a linear interpolation between a natural image and a noise pattern. The expectation is that a distributed model is strongly activated everywhere along the line while a local model is only strongly activated in the vicinity of the image. I also perform ablation analyses to detect the benefit of each proposed trait.

3.1 Datasets

MNIST [Lecun *et al.*, 1998] is a widely used dataset in image recognition. It contains 28×28 gray-scale images of hand-written Arabic digits, divided into 60,000 examples for training and 10,000 for testing. The small scale enables fast experimentation and therefore it is commonly used to evaluate proofs of concept.

CIFAR-10 [Krizhevsky, 2009] contains slightly larger (32×32) color images of 10 classes of objects in natural settings. There are 50,000 training and 10,000 testing examples. Compared to the previous dataset, CIFAR-10 is harder in both natural and adversarial settings.

3.2 Models

For MNIST, I experimented with a simple model with two convolutional layers (16 and 32 channels, both with 5×5 kernels) and a densely connected output layer. This model and its derivatives are trained using Adam with learning rate 0.001 for 20 epochs. Data augmentation methods used include resizing, erasing, and rotating randomly, and adding Gaussian noise ($\sigma = 0.3$).

For CIFAR-10, I used a scaled-down version of VGG architecture [Simonyan and Zisserman, 2014] with 8 convolutional layers. This choice helps to illustrate how a deeper architecture works while being lightweight enough for quick experimentation. All models in this line are trained using mini-batch SGD with learning rate 0.01 for 80 epochs. For data augmentation, I used random cropping, horizontal flipping, rotation, and Gaussian noise ($\sigma = 0.5$, added after normalization).

In both experiments, I start with a baseline model that uses a ReLU architecture and is trained on only natural examples using cross entropy loss. Modifications are added one by one to observe the individual effect (except for elliptical units where I also remove min-out because it becomes unnecessary while making training unstable).

3.3 Training with Quadratic and Logarithmic Units

The piecewise linearity of ReLU networks creates meaningful gradient almost everywhere therefore enables efficient training. In contrast, architectures with highly non-linear elements are very hard to train. I observe that vanilla implementations of logarithmic and elliptical nets tend to train poorly: the performance oscillates around chance level early on from the first epochs or decrease substantially midway through.

To achieve training stability, I start by training a ReLU network and gradually ramp up nonlinear elements. I also find it beneficial to detect collapse events, defined as a decrease of training accuracy by a half in an epoch. In such cases, the model is recovered to a checkpoint and trained for 5 more epochs without increasing nonlinearity.

One might notice that linear functions are simply quadratic ones with second-degree coefficients set to zero. Therefore, to train quadratic units (Section 2.2), I replace Equation 3 with:

$$z = \alpha_t \sum_{i=1}^{m} w_i' x_i^2 + \alpha_t \gamma + \sum_{i=1}^{m} w_i x_i + b, \qquad (7)$$

where t is training step, $a_t \in [0, 1]$, and $\gamma > 0$. The term $\alpha_t \gamma$ is added to the net input to compensate for a possible increase in negativity coming from quadratic terms. Because gradient only flows through active neurons, gaining new active units is a smaller concern than losing existing ones. For a network initialized with Kaiming method, I find that $\gamma = 1$ is sufficient to stabilize learning.

Similarly, ReLog activation function (Section 2.1) is a generalization of ReLU in the sense that the former approaches the latter when β tends to 0. Therefore, I compute the activation of a neuron as:

$$y = \text{ReLog}(z; \alpha_t' \beta) \qquad (8)$$

where z is the net input and α'_t grows from 0 to 1 with each training step.

Many hyperparameters are involved in the various models. For practical reasons, I only tried a few combinations that make intuitive sense and select the best one with respect to performance on natural datasets.

3.4 Attacks

The importance of evaluating against strong attacks has been highlighted more than once in the literature [Athalye *et al.*, 2018; Uesato *et al.*, 2018, for example]. I used Cleverhans, a library of standard implementations of state-of-the-art attacks. Among the algorithms offered there, I selected a battery of attacks from different categories: single-step: **FGM** [Goodfellow *et al.*, 2015]; iterative: **BIM** [Kurakin *et al.*, 2017] and **C&W** [Carlini and Wagner, 2017]; and gradient-free: **SPSA** [Uesato *et al.*, 2018].

For each combination of attack, model, and dataset, I evaluated on batches of 100 test examples, using between 15 and 100 batches depending on the speed of the attack (SPSA is the slowest, taking more than 3 hours to finish a batch). The accuracy on batches is used to calculate average performance and p-value.[1]

4 Results

In this section, I will present the results of the proposed experiments and explain how they validate my hypotheses.

4.1 Learnability and Expressiveness

The performance of my models on clean MNIST digits can be found in Table 1. It is clear that we can reach similar levels of performance with the proposed architecture under a conventional training regime. Training with negative examples hurts the performance, especially on CIFAR-10. A possible explanation is a propositional theory of CIFAR-10 might be excessively big and a more expressive form of logic might be needed. Alternatively, a neural network might need higher capacity to represent such a propositional theory.

The "Proposed arch." results show that we can train a modified architecture with elliptical units and ReLog activation function to the same or better performance compared to ReLU networks. To confirm that filters are actually quadratic, I examine the magnitude of quadratic weights. For an MNIST model, the median of quadratic weights is 1.75 and 10.76 times that of linear weights for the first and the second quadratic layers, respectively. For the CIFAR-10 model, the corresponding figures fall between 0.33 and 1.33.

4.2 Locality

Figure 3 depicts the activation pattern of the baseline model and the model with all logic-inspired traits enabled. I plotted the softmax-normalized predictions of four classes that are selected to include the highest activation for either the given example or the noise pattern.

It is abundantly clear that, whereas the ReLU model emits high probabilities for both true digits and noise, the proposed

---

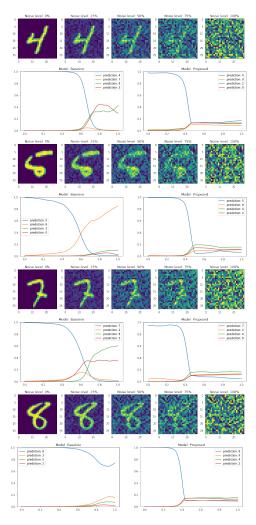[1]I use the function `ttest_ind` in SciPy package with `equal_var=False`.

Figure 3: Comparing activation patterns of ReLU and the proposed architecture on noisy images. Horizontal axis: level of noise, vertical axis: softmax activation.

model is only activated around the test instances. Hyperparameters can arguably be tuned to make the model more robust to noise but this is beyond the scope of the paper.

To evaluate the behavior quantitatively, I obtain models' predictions on 1,000 noise images (sampled uniformly in [0,1] for MNIST and from $\mathcal{N}(0,1)$ for CIFAR-10 because, for the latter, pixel intensity is normalized during preprocessing). Any prediction with higher than 50% confidence is labeled as an error. As I notice that the performance on MNIST noise pattern shows high variance, I trained and evaluated 10 models using different random seeds.

Table 1 demonstrates the high efficiency of the proposed models on rejecting noise for MNIST. The difference in predicted probability for real images and noise indicates that the models are local. Comparing the performance of the proposed architecture and negative training on a ReLU architecture, one can conclude that while negative training helps improves noise rejection, the new architecture is essential to reach high performance.

Model	Real images		Noise	
	Acc.	Prob.	Accuracy	Probability
MNIST				
Baseline	0.99	0.99	0.22 ± 0.15	0.64 ± 0.08
Proposed arch.	0.99	0.98	0.81 ± 0.35	0.35 ± 0.19
Neg. training	**1.00**	1.00	0.27 ± 0.13	0.64 ± 0.09
All mod.	0.95	0.95	$\mathbf{0.92} \pm 0.25$	0.21 ± 0.16
CIFAR-10				
Baseline	0.75	0.91	0.00	1.00
Proposed arch.	**0.77**	0.84	0.05	0.59
All mod.	0.63	0.75	**0.17**	0.64

Table 1: Performance on the task of classifying real images and noise patterns. Models: Baseline = ReLU, Proposed arch. = ReLog + Elliptical + MaxOut + MaxFit (L_1), Neg. training = BCE training + Negative examples, All mod. = Proposed architecture + Negative training. Performance on MNIST noise is reported as mean $\pm$ std as measured on 10 models trained with different random seeds.

On CIFAR-10, the new architecture increases the chance of rejecting noise and reduce the prediction confidence compared to the baseline. There is also a pronounced difference between probabilities predicted by "Proposed arch." on real images and on noise patterns, suggesting that the model is more local than a ReLU network. The negatively trained model is better at rejecting noise at the cost of a lower accuracy on true images.

4.3 Robustness on MNIST

Table 2 presents the robustness of my models against various attacks. It is clear that the proposed architectures lead to a significant increase in robustness across many types of threats. Four out of six models (Row 2, 3, 5, and 6) are more robust than the baseline on all measures. The remaining two models beat the baseline on all but one attack.

Since all models are trained on standard back-propagation, it is unlikely that the improvements come from gradient obfuscation [Athalye *et al.*, 2018]. There are other signs that this is not the case: single-step attacks (variants of FGM) and gradient-free attack (SPSA) are generally not more effective than iterative gradient-based attacks (C&W and BIM).

One of the leading hypotheses about the origins of adversarial examples states that it is the local linearity of ReLU models that makes attacks possible [Goodfellow *et al.*, 2015; Warde-farley and Goodfellow, 2018]. This hypothesis is compatible with the observation that ReLog is more robust than ReLU against adversarial perturbation. However, it is contradicted by the observation that elliptical units, despite being nonlinear everywhere, do not lead to improvement on four out of five measures when they replace min-out. It also has difficulty in explaining why max-out only has a minor impact on robustness while min-out increases performance as much as five times (from 0.03 to 0.16 for BIM).

Training with independent cross entropy loss has a mixed effect on robustness. Adversarial perturbation is an inherently local phenomenon. The effect of activation suppression in distant areas to the decision boundary around an example is
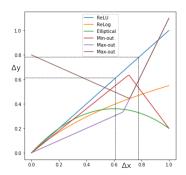

Figure 4: The change of output in response to a small perturbation of input: $\Delta_y = f(x + \Delta_x) - f(x)$ for different functions f.

a poorly-understood area and beyond the scope of the current paper.

4.4 Robustness on CIFAR-10

The improvement of robustness on CIFAR-10 is more modest (see Table 3). Modified models work the best against FGM (L_∞) and C&W attacks for which they are always better than the baseline. However, they are often only as good as the baseline for BIM and sometimes underperform the baseline for FGM (L_2). Improvement coming from ReLog is more consistent than other traits as the corresponding model outperforms the baseline in two measures and maintain statistically equivalent performance on the rest.

We again observe that min-out is superior than both maxout and elliptical units, a property cannot be explained by the linearity hypothesis. The results of BCE training mirrors what has been observed in MNIST experiments.

4.5 Discussions

Given the experimental results, a revision to the linearity hypothesis is needed. Hoffman et al. [2019] recently showed that, theoretically and empirically, smaller norms of the input-output Jacobian matrix implies higher robustness. This result explains why ReLog is consistently better than ReLU and adding elliptical units is advantageous. However, it has difficulty in explaining min-out and max-out results because adding them do not change the locally linear behavior of a ReLU neural network.

To propose a unified framework that explains all observations, let us consider the input-output finite difference instead: $\Delta_y = f(x + \Delta_x) - f(x)$ where f is a neural network, $x \in \mathbb{R}^m, y = f(x) \in \mathbb{R}^n$ are its input and output, and $\Delta_x \in \mathbb{R}^m$ is a vector of a small norm: $|\Delta_x|_p \leq \epsilon$. Figure 4 depicts a special case where $m = n = 1$ or, equivalently, where we consider only one neuron and one linear direction in the input space.

From the graph, it can be inferred that ReLog and elliptical units are inherently more robust than ReLU and the effect increases monotonically with curvature. Min-out significantly improves stability as long as the input lies close to the intersection of two hyperplanes whereas max-out can either increases or decreases stability. The inferior performance of elliptical units compared to min-out perhaps reflects a difficulty in training instead of an inherent property.

	Model	Clean	FGM $\epsilon_{L_\infty} = 0.3$	FGM $\epsilon_{L_2} = 2$	C&W	BIM $\epsilon = 0.3$	SPSA $\epsilon = 0.3$
0	Baseline	0.99	0.07	0.69	0.01	0.00	0.07
1	+ ReLog ($\beta = 2$)	0.99*	0.30*	0.84*	0.19*	0.01*	0.37*
2	+ MaxOut ($k = 4$)	0.99*	0.42*	0.86*	0.23*	0.03*	0.44*
3	+ MinOut ($k = 2$)	0.99	**0.66***	**0.92***	0.31*	**0.16***	**0.52***
4	+ Elliptical ($\alpha = 1$)	0.93*	0.58*	0.57*	**0.47***	0.02*	0.22*
5	+ MaxFit (L_1)	0.99*	0.35*	0.82*	0.19*	0.02	0.35*
6	+ BCE training	0.98*	0.56*	0.83*	0.26*	**0.16***	0.42*
7	+ Negative examples	0.96*	0.24*	0.66*	0.12*	0.00*	0.14*

Table 2: Accuracy of models on MNIST on different attacks. Elliptical units eliminate the need of min-out so I disable it from step 4 onward. BIM: iterations=5, C&W: iterations=50, SPSA: iteration=50. An asterisk (*) signifies that the result is statistically significantly different from the previous one on the same column.

	Model	Clean	FGM $\epsilon_{L_\infty} = 0.3$	FGM $\epsilon_{L_2} = 2$	C&W	BIM $\epsilon = 0.3$
0	Baseline	0.75	0.08	0.43	0.01	0.01
1	+ ReLog ($\beta = 1$)	0.75	0.16*	**0.45**	0.03*	0.01
2	+ MaxOut ($k = 4$)	0.65*	0.12*	0.31*	0.02*	0.01*
3	+ MinOut ($k = 2$)	0.70*	**0.24***	0.37*	**0.04***	0.01*
4	+ Elliptical ($\alpha = 0.5$)	0.68*	0.12*	0.35	0.02*	0.01
5	+ MaxFit (L_1)	0.76*	0.10*	0.41*	0.02	0.00*
6	+ BCE training	0.78	0.12*	0.42	**0.04***	0.01*
7	+ Negative examples	0.66	0.13	0.32*	**0.04**	**0.03***

Table 3: Accuracy of models on CIFAR-10 on different attacks. BIM: iterations=5, C&W: iterations=50. Elliptical units are applied to the first hidden layer only. An asterisk (*) signifies that the result is statistically significantly different from the previous one on the same column.

5 Related Work

Neural Network Design. Some elements of the architecture proposed here were introduced before in isolation. Training with negative examples were used in [Bromley and Denker, 1993] to encourage the rejection of "rubbish class" examples. Maxout units were designed by [Goodfellow *et al.*, 2013] and originally intended to replace rectified linear units instead of working together with them. Parallel to this research, Fan and Wang [2019] succeeded in training quadratic units. Finally, Liu et al. [2019] proposed logarithmic activation functions but their formulations are slightly different from ours and lack the smooth transition from ReLU. Different from all the work cited above, I combine various non-conventional design elements and show that they work together to make a neural network more local and robust.

Adversarial Examples. Because of practical importance, the majority of the literature is concerned with alleviating adversarial examples. Most papers focus on improving conventional architectures through regularization [Mustafa *et al.*, 2019] or injection of adversarial examples into training [Kurakin *et al.*, 2017; Madry *et al.*, 2017]. Results that link robustness to sparseness [Guo *et al.*, 2018] and margin [Croce *et al.*, 2018; Galloway *et al.*, 2018] can be considered a special case of the input-output Jacobian approach taken by Hoffman et al. [2019] which is closely related to my finite difference analysis. Taghanaki et al. [2019] show that another locality-inducing function, the radial basis functions (RBFs), can reduce the success rate of attacks, as might be expected from the analysis in the previous section.

Because experimental settings vary a lot from one paper to another, it is often hard to compare reported results across papers. To compare against Taghanaki et al. [2019], I have set up attacker models to the same settings. Their RBF models significantly outperform ones in this paper on MNIST but this result does not affect my conclusions.

Fooling Examples. Little work has been done in this area. A rare find is Ghosh et al. [2019] which uses an autoencoder with a mixture of Gaussian prior to detect and reject fooling examples.

6 Conclusions

This paper brings in ideas from fuzzy logic to improve deep neural networks. Preliminary results show that the proposed models are more well-behaving on noise patterns and more robust against adversarial examples. Analyses confirm that both the proposed architecture and training procedure contribute to performance on noise patterns while the architectural modifications improve robustness by enabling multiple disconnected regions and increasing stability in each one.

Perhaps most importantly, the current paper hints at what deep neural networks might be capable of: being both local and deep, and combining the learnability and expressiveness of connectionism and the validity of logic. Exciting lines of future research can be foreseen: expanding and generalizing the proposed architectures, improving training procedures, improving calibration on noise patterns and other stimuli, and studying the interaction between local and distributed representations.

References

[Athalye *et al.*, 2018] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *ICML 2018*, pages 436–448, 2018.

[Belohlavek and Klir, 2011] Radim Belohlavek and George J Klir. Fuzzy Logic: A Tutorial. In *Concepts and Fuzzy Logic*, pages 45–87. 2011.

[Bengio and Delalleau, 2011] Yoshua Bengio and Olivier Delalleau. On the expressive power of deep architectures. In *Algorithmic Learning Theory*, pages 18–36. Springer, 2011.

[Bishop, 2006] Christopher M Bishop. *Pattern Recognition and Machine Learning*, volume 4 of *Information science and statistics*. Springer, 2006.

[Bromley and Denker, 1993] Jane Bromley and John S. Denker. Improving Rejection Performance on Handwritten Digits by Training with "Rubbish". *Neural Computation*, 5(3):367–370, 1993.

[Carlini and Wagner, 2017] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. *Proceedings - IEEE Symposium on Security and Privacy*, pages 39–57, 2017.

[Croce *et al.*, 2018] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable Robustness of ReLU networks via Maximization of Linear Regions. 2018.

[Fan and Wang, 2019] Fenglei Fan and Ge Wang. Fuzzy logic interpretation of quadratic networks. *Neurocomputing*, sep 2019.

[Ford *et al.*, 2019] Nic Ford, Justin Gilmer, Nicolas Carlini, and Dogus Cubuk. Adversarial Examples Are a Natural Consequence of Test Error in Noise. 2019.

[Galloway *et al.*, 2018] Angus Galloway, Thomas Tanay, and Graham W. Taylor. Adversarial Training Versus Weight Decay. pages 1–15, 2018.

[Ghosh *et al.*, 2019] Partha Ghosh, Arpan Losalka, and Michael J Black. Resisting Adversarial Attacks using Gaussian Mixture Variational Autoencoders. *AAAI*, 2019.

[Goodfellow *et al.*, 2013] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout Networks. *ICML*, pages 1319–1327, 2013.

[Goodfellow *et al.*, 2015] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *ICLR*, 2015.

[Guo *et al.*, 2018] Yiwen Guo, Chao Zhang, Changshui Zhang, and Yurong Chen. Sparse DNNs with improved adversarial robustness. *NeurIPS*, pages 242–251, 2018.

[Hinton *et al.*, 1986] G E Hinton, J L Mcclelland, and D E Rumelhart. Distributed representations. In *Parallel distributed processing*, pages 77–109. 1986.

[Hoffman *et al.*, 2019] Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust Learning with Jacobian Regularization. 2019.

[Krizhevsky, 2009] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009.

[Kurakin *et al.*, 2017] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *ICLR 2017*, 2017.

[Lecun *et al.*, 1998] Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Liu *et al.*, 2019] Yang Liu, Jianpeng Zhang, Chao Gao, Jinghua Qu, and Lixin Ji. Natural-Logarithm-Rectified Activation Function in Convolutional Neural Networks. pages 1–9, 8 2019.

[Madry *et al.*, 2017] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. pages 1–28, 2017.

[Mustafa *et al.*, 2019] Aamir Mustafa, Salman Khan, Munawar Hayat, Roland Goecke, Jianbing Shen, and Ling Shao. Adversarial Defense by Restricting the Hidden Space of Deep Neural Networks. 2019.

[Nguyen *et al.*, 2015] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. *In Proceedings ofthe IEEE Conference on Computer Vision and Pattern Recognition*, page 427–436, 2015.

[Nguyen *et al.*, 2018] Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. Neural networks should be wide enough to learn disconnected decision regions. *ICML 2018*, 2018.

[Raghu *et al.*, 2017] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl Dickstein. On the expressive power of deep neural networks. In *ICML 2017*, volume 70, pages 2847–2854, 2017.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. pages 1–14, 2014.

[Szegedy *et al.*, 2013] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. pages 1–10, 2013.

[Taghanaki *et al.*, 2019] Saeid Asgari Taghanaki, Kumar Abhishek, Shekoofeh Azizi, and Ghassan Hamarneh. A Kernelized Manifold Mapping to Diminish the Effect of Adversarial Perturbations. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11340–11349, 2019.

[Uesato *et al.*, 2018] Jonathan Uesato, Brendan O'Donoghue, Aaron Van Den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *ICML 2018*, 11:7995–8007, 2018.

[Warde-farley and Goodfellow, 2018] David Warde-farley and Ian Goodfellow. Adversarial Perturbations of Deep Neural Networks. In *Perturbations, Optimization, and Statistics*. 2018.