Portland State University

PDXScholar

Dissertations and Theses

Dissertations and Theses

Summer 8-8-2019

Sensory Relevance Models

Walt Woods

Portland State University

Follow this and additional works at: https://pdxscholar.library.pdx.edu/open_access_etds

Part of the [Computer Sciences Commons](#)

Let us know how access to this document benefits you.

Recommended Citation

Woods, Walt, "Sensory Relevance Models" (2019). *Dissertations and Theses*. Paper 5123.

<https://doi.org/10.15760/etd.7002>

This Dissertation is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.

Sensory Relevance Models

by

Walt Woods

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Electrical and Computer Engineering

Dissertation Committee:
Christof Teuscher, Chair
Dan Hammerstrom
Gerardo Lafferriere
Bart Massey

Portland State University
2019

© 2019 Walt Woods

Abstract

This dissertation concerns methods for improving the reliability and quality of explanations for decisions based on *Neural Networks* (NNs). NNs are increasingly part of state-of-the-art solutions for a broad range of fields, including biomedical, logistics, user-recommendation engines, defense, and self-driving vehicles. While NNs form the backbone of these solutions, they are often viewed as “black box” solutions, meaning the only output offered is a final decision, with no insight into how or why that particular decision was made. For high-stakes fields, such as biomedical, where lives are at risk, it is often more important to be able to explain a decision such that the underlying assumptions might be verified.

Prior methods of explaining NN decisions from images have been proposed, and fall into one of two categories: post-hoc analyses and attention networks. Post-hoc analyses, such as Grad-CAM, look at gradient information within the network to identify which regions of an image had the greatest effect on the final decision. Attention networks consist of structural changes to the network, which produce a mask through which the image is filtered before subsequent processing. The result is a heatmap highlighting regions which have the greatest effect on the final decision. This dissertation identifies two flaws with these approaches. First, these methods of explanation change wildly when the network is exposed to adversarial examples. When an imperceptible change to the input results in a significant change in the explanation, how reliable is the explanation? Second, these methods all produce a heatmap, which arguably does not have the definition required to truly understand which features are important. An algorithm that can draw a circle around a cat does not necessarily know that it is looking at a cat; it only recognizes the existence of a salient object.

To address these flaws, this dissertation explores *Sensory Relevance Models* (SRMs), methods of explanation which utilize the full richness of the sensory domain. Initially motivated by a study of sparsity, several incarnations of SRMs were evaluated for their ability to resist adversarial examples and provide a more informative explanation than a heatmap.

The first SRM formulation resulted from a study of network bisections, where NNs were split into a pre-processing step (the SRM) and a classifying step. The result of the pre-processing step would be made very sparse before being passed to the classifier. Visualizing the sparse, intermediate computation would potentially have yielded a heatmap-like explanation, with the potential for more textured explanations being formed off of the myriad features comprising each spatial location of the SRM’s output. Two methods of achieving network bisection using auxiliary losses were devised, and both were successful in generating a sparse, intermediate representation which could be interpreted by a human observer. However, even a network bisection SRM which used only 26 % of the input image did not result in decreased adversarial attack magnitude. Without solving the adversarial attack issue, any explanation based on the network bisection SRM would be as fragile as previously proposed methods.

That led to the theory of *Adversarial Explanations* (AE). Rather than trying to produce an explanation in spite of adversarial examples, it made sense to work with them. For images, adversarial examples result in full-color, high-definition output. If they could be leveraged for explanations, they would solve both of the flaws identified with previous explanation techniques. Through new mathematical techniques, such as a stochastic Lipschitz constraint, and designing new mechanisms for NNs, such as the Half-Huber Rectified Linear Unit, AE were very

successful. On ILSVRC 2012, a dataset of 1,281,167 images of size 224×224 comprising 1,000 different classes, the techniques for AE resulted in NNs $2.4 \times$ more resistant to adversarial attacks than the previous state-of-the-art, while retaining the same accuracy on clean data and using a smaller network. Explanations generated using AE possessed very discernible features, with a more obvious interpretation when compared to heatmap-based explanations. As AE works with the non-linearities of NNs rather than against them, the explanations are relevant for a much larger neighborhood of inputs. Furthermore, it was demonstrated that the new adversarial examples produced by AE could be annotated and fed back into the training process, yielding further improved adversarial resistance through a Human-In-The-Loop pipeline.

Altogether, this dissertation demonstrates significant advancements in the field of machine learning, particularly for explaining the decisions of NNs. At the time of publication, AE is an unparalleled technique, producing more reliable, higher-quality explanations for image classification decisions than were previously available. The modifications presented also demonstrate ways in which adversarial attacks might be mitigated, improving the security of NNs. It is my hope that this work provides a basis for future work in the realms of both adversarial resistance and explainable NNs, making algorithms more reliable for industry fields where accountability matters, such as biomedical or autonomous vehicles.

Acknowledgements

I am very grateful for the support and advice of my advisor, Christof Teuscher. Thank you to Jack H. Chen, Mohammad Taha, Kelsey Scharnhorst, Patrick Sheridan, and Jens Bürger for helpful discussions and collaborations which contributed to this work. Thank you to A. Madry (from [41, 63]) and J. Cohen (from [11]) for helpful discussions and clarifications about their work. I would also like to thank FuR for assisting in the collection of photos used in examples throughout this work. This work was supported in part by the National Science Foundation under award # 1028378 and by the Defense Advanced Research Projects Agency (DARPA) under award # HR0011-13-2-0015. The views expressed are those of the author(s) and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

Table of Contents

Abstract	i
Acknowledgements	iv
Table of Contents	v
List of Tables	viii
List of Figures	ix
Glossary	xix
1 Introduction	1
1.1 Contributions	8
1.1.1 Primary Contributions	8
1.1.2 Auxiliary Contributions	10
2 Locally Competitive Algorithm	13
2.1 LCA	13
2.2 LCA and Deep Learning	14
2.3 Supervised LCA	15
2.4 Extending the Supervised LCA to SRMs via Network Bisection . . .	18
3 Sensory Relevance Models via Network Bisection(s)	21
3.1 Related Work	22
3.2 Theory	24

3.2.1	Why Bisection?	25
3.2.2	Required Traits	28
3.3	Implementations and Results	32
3.3.1	Activity Adjustment	32
3.3.2	Gradient Integration Method	39
3.4	Discussion	46
4	Sensory Relevance Models via Adversarial Explanations	49
4.1	Related Work	51
4.1.1	Adversarial Attacks	51
4.1.2	Explanation Methods	56
4.1.3	Human-in-the-Loop (HITL)	59
4.1.4	Lipschitz Continuity	61
4.2	Methods	62
4.2.1	Adversarial Attack Generation and Evaluation	63
4.2.2	Defense via Lipschitz Continuity	69
4.2.3	Gradient Minimization as Weight Regularization	73
4.2.4	Half-Huber Rectified Linear Unit (HHRReLU)	74
4.2.5	Output Zeroing	75
4.2.6	Adaptive ψ	77
4.2.7	Adversarial and Noisy Training	77
4.2.8	Human-in-the-Loop (HITL)	79
4.2.9	Architectures and Datasets	80
4.2.10	Code Availability	84
4.3	Results	84
4.3.1	CIFAR-10 Experiments	84

4.3.2	ImageNet Experiments	104
4.3.3	COCO Experiments	107
4.4	Combination with Network Bisection SRMs	107
4.5	Oregon Health & Science University MEP-LINCS Report	109
4.6	Discussion	109
5	Conclusion	112
	References	116
A	Appendix Selected Papers	127
A.1	Adversarial Explanations for Understanding Image Classification Decisions and Improved Neural Network Robustness	127
A.2	Fast and Accurate Sparse Coding of Visual Stimuli with a Simple, Ultra-Low-Energy Spiking Architecture	128

List of Tables

4.1	Effect of Modifications on CIFAR-10	93
4.2	Effect of Modifications on CIFAR-10, Part II	94
4.3	Effect of Adaptiveness on CIFAR-10	95
4.4	Effect of Adaptiveness on CIFAR-10, Part II	96
4.5	Effect of Modifications on ImageNet and COCO	105

List of Figures

- 1.1 Reproduction of Figure 3 from [28]. They proposed a very compelling, multi-level attention scheme which highlights salient objects wonderfully on the CIFAR-10 dataset. Furthermore, their scheme shows how the network’s attention is distributed at multiple levels within the network. Nonetheless, these highlights beg the question: within the regions marked salient, what qualities actually affect the network’s output? 3
- 1.2 State-of-the-art methods of explaining an NN’s decision based on some input (a) rely on the generation of some heatmap (b), generated here via Grad-CAM [58]. In cases where the object’s shape is non-discriminatory, this provides little insight into the decision. Adversarial examples produce much richer output, and can be leveraged to accentuate the NN’s decision (c), making salient features clearly visible when compared with the original image. Alternatively, some desired output may be encouraged to demonstrate features salient to that desired output (d). This technique also demonstrates training deficiencies - clearly, the COCO dataset contains few images of burgers from a top-down perspective, and a pile of ingredients is a strong indicator of sandwich-ness. Unless otherwise specified, all images in this paper were taken by the authors to avoid copyright issues. 6

2.1	Supervised extension to the LCA vs its original, unsupervised formulation.	17
3.1	SRMs via network bisection sought to provide an intermediate, compressed representation of input signals that would be useful for human-in-the-loop debugging of data processing algorithms.	25
3.2	Network bisection SRMs require a high level of sparsity, which may be achieved through the ReLU operations already present in many NNs. The NN's output at any given layer may be made more or less sparse simply by moving the entire decision surface one direction or another through an auxiliary loss (Eq. (3.14)).	33
3.3	Example comparison on CIFAR-10 task between an SRM network with a 90% sparsity target and a traditional, non-SRM network. Each batch consisted of 64 training examples.	37
3.4	Example inputs (left), SRM activations before clipping (left-middle), SRM activations after clipping (middle), and actual regions of input used for final classification (right-middle). The right-most image is the same visualization as the SRM outputs applied to a traditional NN that does not use point 3 of Section 3.2.2.	38
3.5	Saliency map given an input for both (a) original input, (b) random perturbation, and (c) adversarial perturbation. Note that the saliency maps change drastically based on minor changes to the input.	40

3.6 Architecture of network, using GIM modules to produce an SRM.

At top is the architecture for a traditional ResNet - note that the “GIM” modules do not exist in a traditional network, and are underlined in red to note this. The inputs to an NN are typically already zero-mean and unit-variance, hence the lack of a batch normalization around the first GIM module. Within the Pre-block, the GIM is inserted between the two steps of the batch normalization operation (normalization and the subsequent affine operations, which are added to support a wider variety of output distributions). Within the GIM, a detached version of the information into the module is sent to a statistics network, which in turn produces a spatial mask by which features are multiplied, as per Section 3.3.2. The loss function used to train the statistics network is specified in Section 3.3.2. 41

3.7 Example GIM activations on several images. The full-color images are the inputs; the following four columns are GIM activations before the first, second, and third super blocks, and finally before the global averaging layer. Only the third column of GIM activations were used to affect the network; the others were passive observers. Black indicates a GIM output of -1 , red indicates a GIM output of just below zero, and white indicates a spatial location which was passed on to the final classification result, with a brighter white being closer to 1. The three rows illustrate three different networks: one with $\lambda = 0.9$, another with $\lambda = 0.99$, and a third with both $\lambda = 0.99$ and the learning rate for the GIM module set to 0.01 times the learning rate of the parent network, allowing for more gradual learning of relevance statistics. Note that the final column of each of these is a flat color, indicating no meaningful statistics were captured at that level. 46

4.1	While adversarial examples are considered a nuisance by most, they have the potential to provide reliable explanations with the same richness of information as the original input. For example, when an NN trained at finding lung nodules in radiographs needs investigation (a, b), an attack may be targeted at a desired new network output—such as changing a nodule classification to a non-nodule classification (c) or emphasizing the nodule (d)—to produce a new image which is minimally changed but produces the desired output. By comparing these inputs and looking at the differences, a human operator can identify relevant features in the input with greater fidelity than current methods.	50
4.2	Compared to the state-of-the-art, our method achieved higher accuracy while being tolerant of greater adversarial perturbations on both ImageNet (Fig. 4.2a) and CIFAR-10 (Fig. 4.2b). By measuring the ARA, a model’s resistance to adversarial attacks is taken into consideration alongside its ability to make predictions better than the naive baseline (the hatched region). See Section 4.1.1 for additional details. * Madry <i>et al.</i> used a network that was 10× as wide as a traditional ResNet-110, and also trained against an L_{inf} adversary rather than an L_2 adversary [41]. † This curve came from personal communication with A. Madry on a standard ResNet-50; see Section 4.1.1 for details.	54

4.3	A demonstration of LIME on an NN diagnosing lung nodules in the JSRT dataset. LIME’s algorithm consists of taking an input (a), dividing it up into super-pixels (b), and then using linear approximations to determine which subset of super-pixels most significantly affects the network’s classification (c). How well does this explain the decision?	58
4.4	Selvaraju <i>et al.</i> ’s Grad-CAM [58] processes an input (a) and uses back-propagation (b) to produce localized gradient information which can be presented as a heat map of salient regions (c). How well does this explain the decision?	59
4.5	An adversarial example corresponding to the same network and input as Figs. 4.3 and 4.4. Here, the resulting adversarial input (a) is the sum of the original input (b) and a crafted noise term scaled by a small coefficient (c). The result is an imperceptible change in input leading to a completely different NN output. Also worth noting is that the location of the noise in (c) does not correlate well with either the LIME or Grad-CAM explanations.	60
4.6	In contrast to Fig. 4.5, our preliminary technique of increasing adversarial distance concentrates the perturbations needed to change the network’s output. The RMSE (c) between the original input (a) and the adversarial attack (b) is also much greater for a smaller change in network output.	63
4.7	Example of emphasizing a lung nodule via the same method of adversarial attack. Same network as Fig. 4.6.	64

4.8	The previously mentioned attack ARA metric depends on the relative confidence between the correct class and the second-most-confident class. For related classes such as “car” and “truck,” the distance between these two classes may not increase through training. However, by measuring the distance from the correct class back to a fixed baseline, as is done with the BTR ARA metric, improvements in feature recognition may be measured regardless of the presence of related classes.	67
4.9	Different explanation techniques using $\rho = 0.075$. (a) The original image. (b) A positive explanation for the donut class; we note alignment of the added “hole” with a wrinkle in the original sandwich bun. (c) A negative explanation for the donut class resulted in the removal of the round shape of the sandwich. (d) A positive explanation for the true class, sandwich, results in exposed contents (peppers or tomatoes), and the beginnings of lettuce. (e) A negative explanation for the sandwich class reveals homogenization of the bun’s texture, and further rounding out of the overall shape. (f) Positive explanation for a completely unrelated class, horse; legs were clearly added, and the textured area in the upper-left of the image is appropriated as a saddle.	68

4.10	Training an NN on samples from a dataset (left) specify desired behavior at the data points, but does not describe behavior in between those data points, allowing the decision surface to take an arbitrary shape. Adding some noise (second from left) can somewhat improve the behavior between dataset samples, but is statistically unlikely to improve the worst behaviors. Adversarial training (second from right) deliberately attempts to improve the worst performing points, leading to a smoother decision surface. Adding a stochastic Lipschitz loss, as Eq. (4.9), further smooths behavior between data points.	78
4.11	Prototype HITL interface. Whereas adversarial training simply generates adversarial examples and trains them to be recognized as the original class, our adversarial examples can be made of sufficient quality to merit human intervention as to whether or not the class of the image has changed.	80
4.12	Plot of all CIFAR-10 experiments run with a ResNet-44; accuracy on clean data versus attack ARA (as per Section 4.2.1). ● N1 indicates an unmodified ResNet-44, ● N2 indicates a network trained with only adversarial training, ● N3 indicates a network which combines adversarial training and our regularization, and ● N4 indicates an experiment using only our regularization. Dotted experiments are also used in Fig. 4.13.	86

- 4.13 Input (left) and adversarial examples of different CIFAR-10 classes, generated for NNs with different levels of adversarial resistance using the different $g(\cdot)$ functions from Section 4.2.1, as indicated by the heading at the top of each column. The relative accuracy and attack RMSEs may be compared in Fig. 4.12; from top to bottom, these correspond to the experiments denoted by ● N1, ● N2, ● N3, and ● N4 dots. $g_{explain-}$ was generated by emphasizing the “cat” category. 92
- 4.14 The $L_{2,min}$ method of adversarial generation adds stability to adversarial training, but can suffer from degeneracy, where multiple training examples of the same class override a lone neighbor of a different class. Here, $L_{2,min}$ adversarial training would result in “Class A” being overshadowed by the two “Class B” instances. Since $L_{2,min}$ stops at the border of a misclassification, the class with fewer local members only successfully trains a very narrow region. As per Paragraph 4.3.1.15.2, this may be fixed with HHAT. 101
- 4.15 A side-by-side comparison of the $g_{explain,+}$ explanations from a network with the adversarial robustness modifications (top) versus with both adversarial robustness modification and the GIM SRM (bottom) shows that filtering out some of the intermediate activity of the network resulted in little change to the explanations. 108

4.16 Excerpt from the MEP-LINCS report delivered to OHSU. In the gray box are example images, unaltered, which were from the group treated with cxcl12|beta. Below that are three rows of images, each column being different views of the same image. Row 1 contains the adversarial example for cxcl12|beta using $g_{explain,+}$. Row 2 contains the unaltered base image, and the ligand with which it was originally treated. Row 3 shows the difference between rows 1 and 2. 110

Glossary

Adversarial Attack / Adversarial Example: When a neural network is trained, it generates the correct output for some set of inputs. An adversarial attack is a modified version of an input for which the network would have been correct, but for the modified version, the network produces incorrect output. The perturbation from the clean input to the adversarial input is typically imperceptible for state-of-the-art networks. Since the perturbation is difficult for a human observer to detect, it may be used as an attack which makes the network fail in a way a human observer would not. More information may be found in Chapter 1.

AE: *Adversarial Explanation.* The most prominent contribution of this work, an adversarial explanation is an explanation which demonstrates features salient to the network’s classification in the same domain as the network input, and is produced via the same method as adversarial attacks. AEs are described in Chapter 4.

ARA: *Accuracy-Robustness Area.* On a graph with axes of allowable adversarial perturbation RMSE and classifier accuracy, the ARA is the area between a classifier’s curve and the curve for a naive classifier. Since a non-naive classifier will always be susceptible to adversarial attacks of some magnitude, the ARA measures the decline in a classifier’s performance as progressively larger attacks are allowed. See Section 4.1.1 for additional information.

Attack ARA: ARA applied specifically to accuracy; that is, the “accuracy” axis used for the RMSE classification should be based on a top-1 classification accuracy.

See Section 4.1.1 for more information.

BTR: *Better Than Random.* For two classes which are closely related, the attack ARA is an unreliable measurement since the two classes are likely to gain confidence in tandem. BTR is therefore used to measure the difference between a class' confidence and a random confidence value, as opposed to attack ARA, which measures the difference between the top two classes. See Section 4.2.1.2 for more information.

BTR ARA: The ARA metric computed with the accuracy axis replaced by the percentage of inputs where the correct class has greater than random confidence. See Section 4.2.1.2 for more information.

Explanation: In the context of this work, an explanation is a means of understanding the decision made by an NN, ideally in a reliable way which requires minimal training to interpret. See Chapter 1 and Section 4.1.2 for comparisons amongst different state-of-the-art NN explanation methods.

GIM: *Gradient Integration Module.* A method for achieving a network bisection SRM by tracking gradient statistics in a secondary network. See Section 3.3.2 for more information.

HHRReLU: *Half-Huber Rectified Linear Unit.* The methods outlined in Chapter 4 optimize the first derivative of the network's output. A traditional ReLU function has a continuous value and a discontinuous first derivative, making optimization difficult. This may be eased by using a one-sided Huber function, which has both a continuous value and first derivative. See Section 4.2.4 for more information.

HITL: *Human-In-The-Loop*. Most supervised NNs are trained using inputs which have been annotated by human annotators. To improve the NN’s training, the data for more inputs must be gathered and these new inputs must then be labeled. HITL methods in this work refer to methods of improving a network’s training without requiring new data, instead leveraging communication between algorithms using the NN (such as AE) and a human observer. See Section 4.2.8 for more information.

LCA: *Locally Competitive Algorithm*. A sparse coding algorithm which minimizes the number of non-zero coefficients while simultaneously maximizing the fidelity of an input reconstruction based on the resulting sparse code. See Section 2.1 for more information.

ML: *Machine Learning*. Computer algorithms which learn without explicit programming. Rather than describing the input to output transformation explicitly, that transformation is learned based on examples of the desired function.

Network Bisection SRM: An SRM realized by bisecting an NN, and inserting an auxiliary loss at that point to enforce sparsity. Additional requirements and information is laid out in Chapter 3.

NN: *Neural Network*. An ML algorithm which combines layers of artificial “neurons,” which weight input signals, with non-linear activation functions. The result is a highly-parametrized model which can replicate a wide variety of functions.

ReLU: *Rectified Linear Unit*. An NN activation function which truncates negative values to zero, and leaves positive values unchanged.

RMSE: *Root-Mean-Squared Error.* A measurement of difference between two vectors, which is calculated by taking the square root of the mean of the squared difference between the elements in the vectors. Equal to the Euclidean distance between the two vectors normalized by the square root of the number of elements in the vectors.

SGD: *Stochastic Gradient Descent.* A method for training NNs which involves the repeated estimation of the network’s gradient with respect to some loss function, and then moving weights away from that gradient in an attempt to minimize the value of the loss function.

SRM: *Sensory Relevance Model.* A method of explaining an NN’s output which utilizes the same domain as the network’s inputs. By using the exact same format as the original input, a human observer may leverage their prior experience with the problem domain to analyze the NN’s decision in its original context, using changes between the explanation and the original input to infer relevance. This is in contrast to approaches which super-impose metadata on a sensory image, which may often be misinterpreted due to unclear relationships between the metadata and the original sensory domain. Two classes of SRM were explored: the network bisection SRM in Chapter 3, and adversarial explanations in Chapter 4.

Introduction

Deep learning is a fast-advancing technique for dealing with multidimensional input signals, including images and videos. The top performers for challenges such as ImageNet have been dominated by deep learning solutions [22], and the broad applicability of the technique has garnered usage across a swath of fields, including biomedical [24], logistics [40], user-recommendation engines [13], and defense [1]. Part of this popularity is owed to the genericism and simplistic efficiency of the approach: input signals are passed to highly-parallelizable computational layers that transform the data multiple times before producing output signals. The set of transformations possible in a sufficiently deep and wide network is infinite due to the repeated, non-linear activation functions. For this reason, deep *Neural Networks* (NN) have been more effective on complicated problems than straightforward Bayes' approaches or other traditional *Machine-Learning* (ML) techniques. These output signals are in practice iteratively adjusted to minimize an arbitrary objective function via the popular backpropagation algorithm. This approach is extremely flexible: either the network's output may be adjusted to approach some target signal, or the network's output may optimize some task-specific heuristic.

To date, NN-based approaches have all been "black box" approaches, which impose no requirement for conveying intuition on the internals of the network. Many different internal architectures exist based on intuitive or beneficial properties, such as convolution or pooling layers, but the final interactions of these components are unrestricted and impossible to intuit. Furthermore, these networks tend to get

stuck in local minima that exist on the training set but not on test examples from the larger problem. This condition is known as over-fitting, and it can limit the ability of existing neural networks to achieve good performance on cases outside of the training set. An example of this phenomenon can be seen when training a network to classify a photo as containing an animal or not. Without efforts to prevent it, many instances of the network will learn to classify a blurry background as an animal, and a crisp background as not an animal, an artifact of the focal length for different types of photography [34]. For human operators, this “black box” quality makes debugging the systems - or improving them - a virtually impossible task.

Overall, the goal of this dissertation was to find a more reliable and informative way to explain the decisions made by neural networks. Here, “explain” means a method which demonstrates the reasoning behind a network’s output to a person, in such a manner that does not require the person to have expert training to interpret the explanation. A method of explanation should also have high fidelity to the model: that is, it should be unambiguous how the input would need to change to get a different result.

Current methods for debugging NNs are unreliable. The state-of-the-art for post-hoc network analysis involves investigating the internal layers via saliency maps [21, 59], or looking at the receptive fields to which internal nodes respond [4, 39, 47, 54, 58, 75]. Methods of modifying an NN’s architecture such that an “attention map” is both generated and used to filter inputs have also been proposed [12, 25, 28, 64]. In addition to generating compelling explanations, attention methods have been shown to improve network accuracy. However, as this dissertation establishes, all of these existing methods share the same two flaws when considering their value for explaining network behavior.

Figure 1.1: Reproduction of Figure 3 from [28]. They proposed a very compelling, multi-level attention scheme which highlights salient objects wonderfully on the CIFAR-10 dataset. Furthermore, their scheme shows how the network’s attention is distributed at multiple levels within the network. Nonetheless, these highlights beg the question: within the regions marked salient, what qualities actually affect the network’s output?

The first flaw is demonstrated in this dissertation via adversarial attacks (also known as adversarial examples). Adversarial attacks are extremely small perturbations which completely change the network’s output; they are termed an “attack” because they may be used maliciously, for instance to make a turtle appear as a rifle [2]. Adversarial attacks against networks compatible with existing explanation methods have little to no correlation with the explanations generated by those methods. If an explanation were reliable, then the perturbations which change the network’s output should be related to that explanation. This is not the case for existing methods, drawing into question the importance of the regions highlighted by these explanation methods (Section 4.1.2).

The second flaw concerns the limited merit of heatmaps. Attention methods, such as Jetley *et al.*'s [28], produce extremely compelling masks of salient objects in the input (Fig. 1.1). However, while these do arguably show the region which most affected the network's output (ignoring the first flaw of adversarial examples for the moment), they do not address the qualities of those regions that affect the output. In the third row of Fig. 1.1, for instance, who is to say that the network's output was "deer" and not "dog?" Both classes would have roughly the same salient outline. From a human point of view, it would be easy to justify the explanation algorithm as including the antlers, but the algorithm itself highlighted a large region, the antlers being only a small portion of the overall explanation. Thus, the extent of the explanation's benefit is to show that the salient object was considered, nothing about how or why it was considered. Existing explanation methods all focus on the generation of a heatmap, and all heatmap-based methods share this flaw (Section 3.4).

This dissertation proposes *Sensory Relevance Models* (SRMs) as a means of addressing the previously mentioned flaws, while also bringing deep learning closer to a form that can be reliably understood and manipulated by human operators. SRM refers to methods of explanation which utilize the same domain as the network's sensory inputs. By using the exact same format as the original input, a human observer may leverage their prior experience with the problem domain to analyze the NN's decision in its original context, using changes between the explanation and the original input to infer relevance.

For the first formulation of SRMs, network bisection, only the first of the above flaws were known. Adversarial examples affected any known method of explaining

an NN’s decision, even attention networks. From the very existence of imperceptible adversarial examples, it may be inferred that NNs rely on very high gain factors from the input. Existing explanation methods were either post-hoc or attention-based, which multiplied all inputs by a small, but non-zero, coefficient. Attention-based examples were deemed a promising direction, and it was reasoned that filtering out spatial elements at different levels absolutely - multiplying by zero - would eliminate some of the input elements needed for adversarial attacks and focus the remaining attack into coherent features.

Counter-intuitively, explorations of the original SRM proposal resulted in adversarial examples with equivalent or even smaller perturbations. While investigating their spatial statistics yielded interesting salient regions at different levels in the network (Section 3.3.2), they utterly failed at the original goal of producing a reliable explanation mechanism. It seemed necessary to solve the mystery of why adversarial examples were not improved by this first iteration of SRMs, which utilized fewer input elements. Merely by the zeroing of some portion of inputs (up to 74 % on CIFAR-10), one might expect that the result would be adversarial examples of a larger magnitude. This was not the case. While investigating this phenomena, I also came to realize the second of the above flaws: heatmap-based methods do not possess the necessary richness to explain failure cases.

The failure of that first SRM led to a much more informative success: adversarial explanations (Chapter 4). The realization of the second flaw with existing explanation methods necessitated a new method which would not rely on a heatmap. The original flaw necessitated an approach that worked alongside adversarial examples, or resulted in adversarial examples that aligned with the explanation. Consider the nature of a minimum-perturbation adversarial example:

Figure 1.2: State-of-the-art methods of explaining an NN’s decision based on some input (a) rely on the generation of some heatmap (b), generated here via Grad-CAM [58]. In cases where the object’s shape is non-discriminatory, this provides little insight into the decision. Adversarial examples produce much richer output, and can be leveraged to accentuate the NN’s decision (c), making salient features clearly visible when compared with the original image. Alternatively, some desired output may be encouraged to demonstrate features salient to that desired output (d). This technique also demonstrates training deficiencies - clearly, the COCO dataset contains few images of burgers from a top-down perspective, and a pile of ingredients is a strong indicator of sandwich-ness. Unless otherwise specified, all images in this paper were taken by the authors to avoid copyright issues.

it demonstrates the nearest decision boundary. That is, if adversarial examples themselves could consist of more significant perturbations, they would function as

definitive explanations. Adversarial examples have the capacity to illustrate the closest boundary, similarly to how other machine learning methods, such as trees, can be inspected for the closest boundary. To the best of my knowledge, this has not been proposed nor realized previously. The resulting work proposes a variety of techniques for enhancing the adversarial robustness of an NN. The work also demonstrates the efficacy of these techniques, not only surpassing the previous state-of-the-art defense against adversarial attacks by $2.4\times$, but also producing rich, full-color explanations that clearly illustrate the features considered salient by the network. The principle is demonstrated in Fig. 1.2. Compared to a state-of-the-art heatmap method, Grad-CAM [58], the adversarial explanations much more clearly demonstrate salient features and the reasoning behind the decision, and rely less on the human observer’s intuition to fill in missing information. This work was packaged into the attached paper, Appendix A.1.

Two new SRM techniques were proposed and analyzed in this dissertation. Network bisection is a novel form of attention which completely obscures parts of the input and is learned through an auxiliary loss, not directly through the classification loss. *Adversarial Explanations* (AE) are a novel group of techniques for stabilizing an NN such that adversarial attacks against that network are larger in magnitude and consist of coherent features.

The network bisection SRM formulation offered no benefits over other state-of-the-art work in its family. Newer techniques, such as the aforementioned Li *et al.* [35] or Jetley *et al.* [28] works on attention, have more pronounced benefits in terms of accuracy and HITL interaction when compared to the network bisection SRM. This dissertation also attempted a combination of the network bisection SRM formulation and the AE work. The results of the AE work were not enhanced

by the old SRM formulation, and were superior in every quantifiable and qualitative capacity.

The new AE techniques for robust networks and reliable explanations are unparalleled at this time. They accomplish the same goals as the originally proposed SRM: increased adversarial resistance and a capacity for HITL interactions, as detailed in Chapter 4. Furthermore, they surpass the original expectations of this dissertation, as a fundamentally new way of investigating the operation of NNs. In contrast with previously proposed heatmap-based methods, including the network bisection SRM, the AE techniques provide higher-definition and more reliable feedback. As an added bonus, they demonstrate that classification networks may be used for generation, a machine learning niche previously occupied predominantly by *Generative Adversarial Networks* (GANs) and *Variational Autoencoders* (VAEs).

1.1 Contributions

This work contains evidence of my work since the completion of my master’s degree in February 2016 [70], including:

1.1.1 Primary Contributions

- Proposed network bisection SRMs, a form of saliency maps that are integrated into the network (Chapter 3).
- Implemented network bisection SRMs via activity adjustment (Section 3.3.1) and gradient integration modules (Section 3.3.2). Demonstrated that SRMs can produce effective saliency maps which reveal key regions of the input

and work in a hierarchy, but that these shed little light on the final decision (Section 3.3).

- Established that network bisection SRMs do not increase resistance to adversarial attacks, due to increased variance in remaining pixels (Section 3.4).
- Created NN modifications and training strategies aimed at producing high-quality adversarial explanations, which improve on the state-of-the-art resistance to adversarial examples by $2.4\times$, and conducted a comprehensive ablation study of each modification (Chapter 4, [72], under peer review with Nature Machine Learning as in Appendix A.1).
- Improved adversarial training without other improvements by 17 %, in terms of both ability to resist adversarial attacks and produce explanations, via the $L_{2,min}$ adversary (Paragraph 4.3.1.15.2 and Appendix A.1, [72]).
- Showed that adversarial examples could be used to produce reliable explanations with salient features (Chapter 4, [72]).
- Introduced the *Better Than Random* (BTR) *Accuracy-Robustness Area* (ARA) quantitative metric, which correlated well with the quality of explanations produced (Section 4.2.1.2, [72]).
- Utilized adversarial examples as part of a Human-in-the-Loop workflow for enhancing network robustness against adversarial attacks (Chapter 4, [72]).
- Produced a report for *Oregon Health & Science University* (OHSU)’s MEP-LINCS data on the effects of different ligand treatments on cell cultures in-vitro, based on the adversarial explanation methodology (Section 4.5).

1.1.2 Auxiliary Contributions

- Extended my M.S. thesis work, the *Simple Spiking Locally Competitive Algorithm* (SSLCA) [70], to include inhibition and an analysis of its efficacy when combined with modern deep learning techniques (Chapter 2; included as Appendix A.2, [69]).
- Proposed and evaluated four different architectures for performing vector matrix multiplication, a vital function for NNs, using memristors [68].
- Extended the training method for Rozell *et al.*'s *Locally Competitive Algorithm* (LCA) to combine supervised and unsupervised information, giving it better task-oriented effectiveness while still focusing on preserving the information contained in its inputs (Section 2.3).
- Partnered with UCLA to identify spoken digits, using an AgI atomic switch network as a computational reservoir (submitted). They made and tested the device, I formulated the experiments and analyzed the resulting data for computational ability.
- Applied the LCA to radiation detection as part of the *Defense Threat Reduction Agency* (DTRA) program.
- Implemented a state-of-the-art deep learning classification pipeline, applied to many different problems include CIFAR-10, the Stanford Tower dataset, and OpenAI's Gym's CarRacing environment.
- Published 8 peer-reviewed papers, including:
 - W. Woods and C. Teuscher, "Fast and Accurate Sparse Coding of Visual Stimuli with a Simple, Ultra-Low-Energy Spiking Architecture," IEEE

Transactions on Neural Networks and Learning Systems, pp. 1-15, 2018 [69] (Appendix A.2).

- W. Woods *et al.*, “Approximate Vector Matrix Multiplication Implementations for Neuromorphic Applications using Memristive Crossbars,” IEEE NANOARCH, 2017 [68] (winner of the conference’s Best Paper award).
 - K. Scharnhorst *et al.*, “Non-Temporal Logic Performance of an Atomic Switch Network,” IEEE NANOARCH, 2017 [57].
 - M. Taha *et al.*, “Approximate in-memory Hamming distance calculation with a memristive associative memory,” IEEE NANOARCH, 2016 [62].
 - W. Woods, “The Design of a Simple, Spiking Sparse Coding Algorithm for Memristive Hardware,” Portland State University Master’s Thesis, 2016 [70].
 - W. Woods *et al.*, “Synaptic Weight States in a Locally Competitive Algorithm for Neuromorphic Memristive Hardware,” IEEE Transactions on Nanotechnology, 2015 [71].
 - W. Woods *et al.*, “Memristor panic - A survey of different device models in crossbar architectures,” IEEE NANOARCH, 2015 [73].
 - W. Woods *et al.*, “On the Influence of Synaptic Weight States in a Locally Competitive Algorithm for Memristive Hardware,” IEEE NANOARCH, 2014 (winner of the conference’s Best Student Paper award) [67].
- Published 6 non-peer-reviewed proceedings, including:
 - W. Woods, J. H. Chen, and C. Teuscher, “Reliable Explanations via

- Adversarial Examples on Robust Networks,” Portland State University Student Research Symposium, 2019.
- U. Khan, W. Woods, and C. Teuscher, “Exploring and Expanding the One-Pixel Attack,” Undergraduate Research & Mentoring Program, no. 34, 2019.
 - J. H. Chen, W. Woods, and C. Teuscher, “Explanation Methods for Neural Networks,” Portland State University Student Research Symposium, 2019.
 - W. Woods, J. H. Chen, and C. Teuscher, “Explaining the Conclusions of Neural Networks,” Early Detection Conference at Oregon Health and Sciences University, 2018.
 - J. H. Chen and W. Woods, “Generating Adversarial Attacks for Sparse Neural Networks,” Undergraduate Research & Mentoring Program, no. 31, 2018.
 - J. Meng, C. Teuscher, and W. Woods, “Radiation Source Localization by using Backpropagation Neural Network,” Portland State University Student Research Symposium, 2018.

Locally Competitive Algorithm

The foundation of the SRM began via extensions to Rozell *et al.*'s LCA [55]. Sparsity was seen as an important factor for a useful explanation, consistent with insight into human explanations [45]. As a concept, the SRM never relied on the LCA, but the intrinsic sparsity properties of the LCA lent it as an excellent starting point for discussing the bisection of networks to produce intermediate output with semantic significance.

2.1 LCA

Our work on Rozell *et al.*'s LCA has led to a deep understanding of sparsity in ML [68, 69, 71]. Briefly, the LCA is a solution to the sparse coding problem: given an $M \times N$ dictionary Φ consisting of N examples of an input space of dimension M , how might an arbitrary input $s(t)$ be best represented as a linear combination of the columns of Φ ? That is, an estimate of the input, $\hat{s}(t)$, can be built via some coefficients $a(t)$ as $\hat{s}(t) = \Phi a(t)$. Solving the sparse coding problem involves devising some algorithm $q(\cdot)$ that maps $a(t) = q(s(t))$. The sparse coding problem encapsulates the trade-offs between minimizing the number of non-zero coefficients in $a(t)$ and the fidelity of $\hat{s}(t)$ compared to the actual input $s(t)$. Formally, the sparse coding problem may be posed as minimizing the following energy equation:

$$E(t) = \frac{1}{2} \|s(t) - \hat{s}(t)\|^2 + \sum_m \lambda C(a_m(t)), \quad (2.1)$$

where $\lambda C(\cdot)$ is a penalty term that, when large, emphasizes a small number of active coefficients, and when small, emphasizes fidelity in the reconstruction. Equation (2.1) is minimized by the LCA [55].

The minimized energy equation, Eq. (2.1), demonstrates that sparse coding has a tuning knob, λ , which can be used to trade between higher coefficient sparsity or greater fidelity to the input. It has been theorized that sparse coding is an integral part of biological computation processes, due largely to the evidence of sparse coding activity in the mammalian cortex [42, 76]. As a result, many hardware implementations of the LCA have sprung up [7, 29, 30, 48, 50, 52, 53], including my own [69].

2.2 LCA and Deep Learning

Part of my attached paper on an LCA hardware implementation (Appendix A.2, [69]) involved comparing the accuracy of deep learning models trained both on raw data and on LCA-encoded data. Despite biological backing for sparsity, the LCA always reduced the accuracy of the deep learning model. While the LCA solves the sparse coding problem, it relies on a dictionary that is trained on example inputs using unsupervised training. This unsupervised training typically derives from Oja’s rule [46], which updates elements participating in the reconstruction such that subsequent reconstructions will be more accurate:

$$\Phi_{t+1} = \Phi_t + \mu(s(t) - \hat{s}(t))a^T. \quad (2.2)$$

While the LCA solves the sparse coding problem, it traditionally relies on a

dictionary trained via this method, potentially discarding information which would be helpful for classification in favor of information which decreases the L_2 distance between the original input and the reconstruction.

Regardless of lost accuracy, the potential benefits of the LCA became clear through this work. The LCA identifies prominent features in its input, and greatly reduces the non-zero dimensionality of its output. One issue with investigating NNs is that, in a typical convolutional layer, far too many elements in the convolution are non-zero to visualize what the network is considering. With an LCA, the number of non-zero elements is so low that they can be visualized. We discuss adapting the LCA as the base of the SRM, while simultaneously addressing the loss in accuracy, in Section 2.3.

2.3 Supervised LCA

The LCA traditionally uses unsupervised learning. As its operation is defined entirely through reconstructing its input, unsupervised learning is a natural fit. However, unsupervised operation has its trade-offs. While unsupervised approaches tend to learn quickly, they never perform as well as task-specific (supervised) approaches. Rather than making all of their parameters available for the final objective, they are instead attempting to solve a completely different task: re-representing their input. On the other hand, supervised approaches attempt to transform their input for a single purpose, with each transformation bringing the data in the network closer to the desired output. Supervised learning is required to solve the overall information processing problem. This is where SRMs come in as a middle ground: we implemented a system that cleans up the input in a task-specific manner, while also preserving and compressing qualities of the input

to make the results more sensible to manipulate.

To consider using an LCA with backpropagation, we must be able to define the output of the network in terms of its inputs. This is due to the derivatives required for backpropagation to function. However, the LCA's operation is traditionally defined in reverse, where the input may be reconstructed as a linear combination of dictionary elements. Manipulating this expression using the Gramian leads to:

$$\begin{aligned}\mathbf{s}(t) &\approx \mathbf{\Phi}(t)\mathbf{a}(t), \\ [\mathbf{\Phi}^T(t)\mathbf{\Phi}(t)]^{-1} \mathbf{\Phi}^T(t)\mathbf{s}(t) &\approx \mathbf{a}(t).\end{aligned}\tag{2.3}$$

Taking the true derivative of Eq. (2.3) with respect to each weight in $\mathbf{\Phi}(t)$ would be cumbersome and expensive due to the inverse, which may either be unstable or not exist if all dictionary elements are not sufficiently independent of each other. Instead, we replace the Gramian with its diagonal approximate, propagating the scalar derivative of each element in turn:

$$\begin{aligned}[\mathbf{\Phi}^T(t)\mathbf{\Phi}(t)]^{-1} \mathbf{\Phi}^T(t)\mathbf{s}(t) &\approx \mathbf{a}(t), \\ \|\mathbf{\Phi}_i(t)\|_2^{-2} \mathbf{\Phi}_i^T(t)\mathbf{s}(t) &\approx \mathbf{a}_i(t), \\ -\sum_j 2\mathbf{\Phi}_{j,i}(t)\partial\mathbf{\Phi}_{j,i}(t) \frac{\mathbf{\Phi}_i^T(t)\mathbf{s}(t)}{\|\mathbf{\Phi}_i(t)\|_2^4} + \|\mathbf{\Phi}_i(t)\|_2^{-2} \partial\mathbf{\Phi}_i^T(t)\mathbf{s}(t) &\approx \partial\mathbf{a}_i(t),\end{aligned}\tag{2.4}$$

To show the efficacy of this approach, we have compared the traditional, unsupervised method of training the LCA with this novel, supervised method of training an LCA. Each network compared consisted of a single LCA layer with a

Figure 2.1: Supervised extension to the LCA vs its original, unsupervised formulation.

dictionary represented by 50 neurons followed by a single perceptron layer with 10 neurons for classification on the MNIST digit database. Results are shown in Fig. 2.1. Note that the traditional, unsupervised approach quickly reaches its maximum performance, and then hovers at that level or even loses some ability to classify the input due to its emphasis on reconstructing the input well rather than classifying the input well.

On its own, the supervised gradient is incapable of training the LCA. A purely supervised approach breaks down due to the conflict between the supervised loss we derived above and the sparse coding equation intrinsically minimized by the LCA (Eq. (2.1)). The incompatibility of these two terms causes the columns of Φ to become degenerate, leading to a collapse of the network’s functionality. However, combining the supervised gradient with the unsupervised gradient results in a more stable and higher-accuracy network configuration. This indicates that the LCA, which reconstructs an input stimulus based on a limited number of prominent features, can be adjusted to reconstruct the input stimulus based on task-specific

features. The two loss functions together eliminate the reduced accuracy of using a sparse coding layer while retaining its sparsity properties.

2.4 Extending the Supervised LCA to SRMs via Network Bisection

The supervised version of the LCA was functionally a bisected network with two parts: one responsible for generating a task-oriented sparse code of the input, and another for converting the resulting code into a decision regarding the input. This was the underlying theory of SRMs as originally proposed: suppress irrelevant aspects of the input, forwarding only those that would affect the prescribed task. Furthermore, since we derived a task-oriented gradient, we could use that gradient to manipulate the LCA independently of the overall network. This was the key implementation idea behind manipulating the SRM: it is feasible to descend both the loss function for the overall task as well as an auxiliary loss function applied only to the intermediate output from the network, which might be defined by a human operator. In this way, the sparse output of the SRM, which might be implemented via an LCA on top of other processing layers, could both be used to investigate the operation of the network at an intermediate step and to guide its processing.

The first proposed method for achieving an SRM therefore involved a potentially iterative procedure for bisecting the network into two pieces: one which generated a sparse, intermediate representation, and the second which generated a classification based off of that intermediate representation. While the first formulation for realizing SRMs had many similarities to attention-based neural networks [12,28,64], it was unique for two reasons: first, the mask was learned through backpropagation via an auxiliary loss separate from the classification loss; second,

the mask generated by the proposed SRM would exist in a transformed feature space congruent with the input space but potentially scaled or richer in features. This would allow for a much sparser output than typically seen with attention-based neural networks. In the context of adversarial examples, zeroing much of the input would render high gain on those masked regions irrelevant. Elegantly, this approach was motivated by the knowledge that mammalian brains explicitly process saliency as part of processing visual scenes [27]. It was also proposed that the intermediate region could be “painted” by a human operator, a form of *Human-in-the-Loop* (HITL) learning. This HITL aspect was validated by recent, promising work with redirecting the salient region in an attention model based on segmentation annotations [35]. That work, by Li *et al.*, was made public on Feb. 27th, 2018, four days after the original proposal for this dissertation.

Our prior work with LCAs also points to a larger and more significant general problem: how sparse can the input to a classifier be to still achieve a given level of performance? This question is addressed in Section 3.4, though the answer depends heavily on the problem being solved. To intuit the relation between sparsity and ease of explanation, consider the problem of dictionary definitions. While many definitions could be substantially longer to capture more connotations and usages, and thus be slightly more accurate, a shorter definition is easier to understand. Similarly, I wanted to generalize this idea to all two-stage information processing pipelines. I observe that constricting the information shared between these two pipelines to be as sparse as possible, while also preserving locality of the sparse output, will generate a relevance mapping where the few non-zero elements indicate significant information for the task implemented by the pipeline. Functionally, this is similar to an autoencoder, but differs in a few ways: the dimension of the

middle layer does not need to be smaller to be constricting, as we use a sparsity loss to achieve a similar effect, and the second half of the network is not trying to reconstruct the input, but is rather trying to accomplish a different task.

Sensory Relevance Models via Network Bisection(s)

The general task of information or signal processing can be modeled as a function that converts some input space $\mathbf{X}$ to some different output space $\mathbf{Y}$. This input-output mapping is loosely defined on purpose for most methods so that they may have a broad applicability. For NNs, this generality has come at the cost of being unable to intuit how the mapping function works.

Adequately understanding the internal operation of NNs, or the larger toolbox of ML solutions in general, has recently become a primary difficulty of using them [14]. Complications arising from the internal flexibility of NNs are epitomized in adversarial examples: inputs with minor perturbations that completely change the output of the network [2, 10, 15, 61, 63]. These reports all support that NNs solve an underconstrained problem: many possible solutions to the training data exist on manifolds which are distorted toward imperceptibility in standard visual space. As such, overall accuracy on a sample dataset of an overall problem has become less important than an ability to reason about the scope and function of ML techniques. There have been both procedural and architectural attempts at understanding the behavior of NNs as well as making them more resistant to adversarial examples; Section 3.1 details these.

I add SRMs via Network Bisection to the field of explanation techniques. In this case, the $\mathbf{X} \rightarrow \mathbf{Y}$ transformation is split up into a two-stage process, such that an intermediate result is created, the SRM denoted as $\mathcal{S}(\mathbf{X})$. Since the network’s transformation is now defined as $\mathcal{S}(\mathbf{X}) \rightarrow \mathbf{Y}$, an intuitive and reliable

method of visualizing the SRM output would allow for unambiguous confirmation of the aspects being considered by the NN. This work sought to answer: if all information transmitted to the classifier were visualizable in a robust manner, would that visualization be useful for debugging the network? Network bisection is detailed throughout the rest of this chapter.

3.1 Related Work

Procedural attempts at understanding NNs primarily fall under the category of either saliency maps or investigating receptive fields. Saliency maps are produced by running back-propagation all the way to the input layer, and looking at which pixels are most influential on the network’s output [59]. This approach has proven useful for demonstrating what parts of the input are most important to a neural network. Saliency maps have also been extended to aid in image segmentation [21, 59]. By combining saliency maps from multiple, convolved classifications of a source image, saliency maps have even been used to track an object across time by updating the prior of the object’s position according to computed saliency maps [21]. Studies of receptive fields have been primarily limited to investigating the input patterns to which neurons at different levels in an NN respond [39, 47, 75]. By looking at these patterns, researchers have hoped to gain an intuition of the internal functions of the networks.

However, as will be demonstrated further in Chapter 4, these post-hoc methods of explaining an NN’s intuitions suffer from incompatibility with adversarial examples, perturbations of a small enough magnitude that they are invisible to a human observer. The adversarial modifications do not correlate with the previously mentioned methods of explanation, and as such calls such explanations into

question.

Training modifications to make NNs more robust to adversarial inputs have been proposed. A. Madry *et al.* have published several works on adversarial training, integrating adversarial examples into the training process to make a network more robust [41, 63]. Columbia University’s DeepXplore project is another project that utilizes adversarial training; in their work, Pei *et al.* consider the automatic generation of adversarial examples, and using these generated examples to increase the training pool of the network [51]. Doing so forces the network to become a better generalizer as adversarial examples are actively trained away. However, this approach takes a significant amount of time, and while it is guaranteed to improve a network’s resistance to adversarial examples, it remains very difficult to prove an absence of adversarial examples. Likely, the neighborhood of possible adversarial attacks exceeds the bounds tested by DeepXplore. One might imagine adversarial attacks defined specifically closer to the boundaries of the network’s decisions than a project like DeepXplore generates. Realistically, there are limits on how much training a DeepXplore-like system can utilize, due to time and resource constraints.

A prior architectural extension of network design to make NNs more understandable is attention models [12, 64]. Where saliency maps demonstrate the most-used pixels, attention models use a gating technique which involves multiplying the inputs by a learned mask to shape how inputs are forwarded to the classifying portion of the network. Viewing the mask of the attention model shows which inputs were weighted more heavily for a given input. My proposed network bisection differs from attention models because network bisection transforms the input while keeping it local, a much richer approach than the input gating approach used by traditional attention models (attention models from the last year

have also incorporated this quality, e.g. [28]). Additionally, the network bisection technique uses a method separate from classification loss to enforce sparsity in the attentive layer (Section 3.3).

Outside of attempts to understand NNs and harden them against adversarial examples, there are two developmentally significant seeds for the network bisection SRM: one is the our work on Rozell *et al.*'s LCA [55], and the other is GANs [14]. The LCA was covered in Chapter 2.

GANs, introduced by Goodfellow *et al.* in 2014, rely on competition between two NNs attempting to optimize different loss functions: a generator, which generates outputs similar to some reference input, and a discriminator, which takes inputs from the generator and real inputs and attempts to discern which input was real and which was generated [14]. As the discriminator learns to better recognize real inputs, the generator must improve at generating fakes of the input. As the generator learns to create better fake inputs, the discriminator must improve at telling the two apart. Together, these networks compute a loss function that would be impossible to design holistically. The network bisection SRM can be thought of similarly: one network, the SRM, has the goal of transforming its input into a maximally sparse output, while the second network, a classifier, must perform classification based on the limited information passed by the SRM. While computing an optimal SRM directly would be impossible, computing it in tandem with a corresponding output classifier makes the problem we are trying to solve feasible.

3.2 Theory

I proposed network bisection as a means of applying more structure to the $\mathbf{X} \rightarrow \mathbf{Y}$ transformation, such that an intermediate result, the SRM denoted as $\mathcal{S}(\mathbf{X})$, would

Figure 3.1: SRMs via network bisection sought to provide an intermediate, compressed representation of input signals that would be useful for human-in-the-loop debugging of data processing algorithms.

provide better insight into the mechanisms of the overall transformation. The reasoning for bisection, and the necessary qualities of that bisection, are detailed in this section.

3.2.1 Why Bisection?

The proposed architecture is illustrated in Fig. 3.1. Similar to the GANs developed by Goodfellow *et al.* [14], the SRM combines two separate machine learning methods solving different, but related, problems. GANs are implemented in terms of a generator, which transforms noise into a believable image, and a discriminator, which takes one real image and one from the generator, and attempts to discern which is the real image. A problem utilizing a network bisection SRM, on the other hand, is implemented in terms of the SRM, which attempts to forward as little information as possible, and a classifier, which attempts to score highly on a given task based only on the input from the SRM. The composition of these two networks solves the original information processing task.

Bisection SRMs necessarily can only express a subset of the operations available with current state-of-the-art deep NNs. The SRM technique requires restricting

some of the free dimensions of the NN; given enough input data (to prevent overfitting and ensure generalization), an unrestricted network would almost certainly outperform the SRM formulation. On the other hand, convolutional layers are also less expressive than fully-connected NNs, yet they have led to significantly more accurate and robust networks [32].

Unrestricted networks are also known to be highly vulnerable to adversarial attacks. Consider the recent work on ArXiv from MIT’s LabSix group [2]. This group managed to 3D-print objects that Google’s Inception network would misclassify. For example, they 3D-printed a turtle model that always registered as a rifle when passed to Google’s classifier, regardless of orientation [2]. This is a significant problem: traditional neural networks cannot be used in production for critical applications with such glaring flaws.

Restricting the input space of NNs has shown some promise at reducing the effectiveness of adversarial attacks [19]. This was the SRM’s first potential benefit over a traditional NN. The traditional NN might be learn to be more accurate with enough data but should also be significantly easier to manipulate. By restricting the intermediate space through sparsity, a bisection SRM would eliminate several vectors of reasoning that would prevent it from generalizing well. In contrast to attention models, where NNs may still utilize high gain on elements which are scaled as unimportant, the bisection SRM would completely mask out elements, giving them zero influence on the final network decision.

The second proposed unique benefit of the SRM arose from its construction as two separate pieces. Consider how people learn to solve problems and generalize well. Studies on student learning have investigated the differences between how experts and novices approach problems [60]. These studies tend to conclude that

experts develop a repertoire of “schemas” which can be applied to filter larger problems into a manageable (and efficient to solve) set of smaller problems [60]. The overall result, in the context of humans, is that a novice with the proper tools can solve the same problems as an expert, but far less efficiently and with greater cognitive stress [60]. A more recent study compares learning between students given feedback in the form of worked examples or tutoring versus basic correctness feedback [43]. Similarly, McLaren *et al.* conclude that providing worked examples as opposed to only correctness feedback greatly improved the efficiency of learning, but not the overall accuracy of the material learned [43].

Applied to information processing tasks, this implies that an efficient means of teaching generalizable solutions might require access to worked solutions. While part of what makes NNs so widely applicable is their genericism, and the fact that designing a system only requires constructing input-output pairs and a network architecture, this is also likely what is holding back further understanding and development of the function space realizable by NNs. By formulating the SRM as a local, sparse transformation of the input, it is guaranteed that the output could be interpreted by a human operator. This was the fundamental requirement of our system: that a human expert be able to see, given some input, which parts of that input were used to make the final classification. As this middle layer would be visualizable, tools could be developed to manipulate the SRM layer directly, changing the system’s idea of which parts of a given input are relevant, and adjusting the classification layer accordingly to rely on only the input information that is most significant.

Analogous to specifying the middle steps of an information processing task,

this could have enabled a new type of feedback unexplored by previous NN architectures. Prior efforts to counter-act adversarial examples have been limited to generating adversarial examples and training the network to classify them correctly [41, 51], or restricting the input space directly, without concern to how this might affect the overall task [19]. Bisection SRMs would allow a human operator to directly point out which part of the input space is the most important, significantly reducing the possibility of the network honing in on the wrong aspect of the input, and improving its ability to generalize.

3.2.2 Required Traits

The implementation of the bisection SRM relied on defining two sub-problems of the greater information processing problem. A formal definition of the overall task of our system relied on some desired function $\mathbf{Y} = f(\mathbf{X})$, where $\mathbf{Y}$ is the M -dimensional output space of the problem, and $\mathbf{X}$ is the N -dimensional input space. Our task is to find an approximate $\hat{\mathbf{Y}} = \hat{f}(\mathbf{X})$ that minimizes some loss $L[\mathbf{Y}, \hat{\mathbf{Y}}]$. The SRM arose by defining $\hat{f}(\mathbf{X}) = \mathcal{C}(\mathcal{S}(\mathbf{X}))$, where $\mathcal{C}$ is the final classifier that shapes the output of the SRM, $\mathcal{S}$. We then could define loss functions for each of these:

$$\begin{aligned} L_{\mathcal{S}}[\mathcal{S}(\mathbf{X})] &= g\left(\lambda C(\mathcal{S}(\mathbf{X})), L[\mathbf{Y}, \hat{\mathbf{Y}}]\right), \\ L_{\mathcal{C}}[\mathcal{C}(\mathcal{S}(\mathbf{X}))] &= L[\mathbf{Y}, \hat{\mathbf{Y}}], \end{aligned} \tag{3.1}$$

where $\lambda C(\cdot)$ has the same meaning as in Eq. (2.1) of the LCA, a sparsity penalty term, and $g(\cdot)$ is a function to combine the sparsity and classifier output losses (often a weighted sum). Note that, if a desired SRM output $\hat{\mathcal{S}}(\mathbf{x})$ is specified, and

$L[\mathcal{S}(\mathbf{X}), \hat{\mathcal{S}}(\mathbf{X})]$ is defined as the *Mean-Square-Error* (MSE) between the norms per unit-space of $\mathbf{X}$ of the two, then both stages of the pipeline may have their respective losses minimized. This would enable a human operator to specify an ideal SRM output and simultaneously converge the SRM to recognize that region as relevant and converge $\mathcal{C}$ to perform the desired task using only that information. This exact approach was shown to be practical by Li *et al.*, who used the Grad-CAM algorithm as a basis rather than network bisection [35].

Ignoring the other requirements of an SRM for a moment, we earlier demonstrated a combined loss function on $\mathcal{S}(\mathbf{X})$ of $g(a, b) = a + b$ in Fig. 2.1. Indeed, what was derived in that work could be considered a proto-SRM. However, that model possessed neither the locality nor the correct form of sparsity required for a true SRM. To address this gap, we apply three restrictions to $\mathcal{S}(\mathbf{X})$:

1. The output must be in a similar space to $\mathbf{X}$.
2. Each element of the output must rely primarily on inputs local to its spatial point.
3. Sparsity of the output alone is an insufficient restriction; sparsity must be evaluated in the original space of $\mathbf{X}$.

Point 1 refers to the similarity between the output space of $\mathbf{X}$ and $\mathcal{S}(\mathbf{X})$. One of the primary purposes of the network bisection SRM was to make it easier to interpret what an ML technique is deeming relevant. In this light, the SRM could be thought of as a saliency map with an extra dimension containing transformed values that are fed into the final classifier. As such, the ability to re-cast the output to the original space to render overlays is imperative. An important note, if the input is a $32 \times 32 \times 3$ *Red-Green-Blue* (RGB) image, then $16 \times 16 \times 3$, $32 \times 32 \times 32$,

and $8 \times 32 \times 3$ would all be valid shapes for $\mathcal{S}(\mathbf{X})$. The input space may be re-scaled, even non-uniformly, and its depth may be infinitely extended, so long as the SRM implements point 2, that output elements rely primarily on inputs local to their spatial points. Point 1 is trivial to implement: simply define the SRM’s output in a way that retains the space of its input.

Point 2 gives the output of the SRM meaning. Presenting an image of a similar space is meaningless if calculations are non-local. Without a locality restriction, activity at the top of the input image might indicate that the bottom of the image has all of the important features, which is not a useful transformation when it comes to explaining relevance. The full scope of implementations for this point is an open question; the trivial solution with neural networks, and the one explored in this dissertation, was to define the SRM purely in terms of a limited number of convolutions. For example, an NN that has two convolutional layers, each with a kernel width of 5, would result in a locality of $5 + 4 = 9$ around each input element. By never inserting a dense layer, locality is preserved. Some problems may benefit from more eccentric means of implementing this restriction. Consider a race car driving on a track. The inputs to the network controlling the car are a top-down image of the race car and its immediate surroundings, and the car’s current velocity. When considering to brake, the velocity would be incredibly important in determining whether or not the surroundings are relevant. There are at least two ways to address this with an SRM while satisfying the second restriction: have the SRM separately consider the relevance of velocity and visually the objects that the race car might collide with; or assume that the velocity is relevant, and discard that element from the output space, focusing on the image. Either would produce

a valid SRM, and each would highlight different features. Not giving the image-filtering part of the SRM access to velocity would force it to always be aware of potential collisions, which might be easier for a human operator when assigning relevance. On the other hand, including speed would allow a human operator to check whether or not the race car is properly avoiding obstacles when going fast, and when going slow, is driving in the right direction on the track. Locality is a flexible topic. It is a requirement for network bisection SRMs to provide insight, but different expressions of the locality requirement will lead to advantages and disadvantages specific to each problem domain. Locality might even be generally modeled as another loss term, minimizing a distance heuristic between the SRM’s output and its reliance on different spatial points of $\mathbf{X}$.

Point 3, that sparsity on the similar space must be evaluated on the original space, comes from the fact that the SRM’s output consists of multiple channels, similar to how an RGB image has multiple color channels at each spatial pixel. Functionally, due to the locality property of SRMs, there must be some aspect similar to convolution in any SRM. As a result, different relevant aspects of the input at each spatial point in $\mathbf{X}$ must be passed in a channel perpendicular to the spatial dimensions of $\mathbf{X}$. Consider that visualizing the SRM will consist of displaying the norm of the perpendicular space at each spatial point in $\mathbf{X}$. Were sparsity defined in terms of both the space of the input and this perpendicular space, it would be very possible that each spatial point of $\mathbf{X}$ would have one corresponding output active in $\mathcal{S}(\mathbf{X})$. Rendering this as an overlay of $\mathbf{X}$ would be almost impossible to display to a human operator in a meaningful way. For this to be useful, sparsity must be defined in terms of $\mathbf{X}$ ’s space. This was implemented by crafting $\lambda C(\cdot)$ to evaluate sparsity with respect to the norms at each spatial

point in $\mathbf{X}$.

The requirements of the network bisection SRM were therefore well-defined. The implementation of this theory is discussed in Section 3.3.

3.3 Implementations and Results

As per Section 3.2.2, a bisection SRM requires restrictions on the output from the SRM layer. I experimented with two varieties of achieving sparsity: one by adjusting the activity levels in a standard network, and another by using secondary networks at each bisection to integrate gradients and determine which spatial locations were important.

3.3.1 Activity Adjustment

Multi-layer neural networks, including the ResNet [16], often utilize the *Rectified Linear Unit* (ReLU) as a non-linearity. Conveniently for the definition of SRMs, the ReLU transmits no information when its input argument has a value of less than zero, illustrated in Fig. 3.2. Thus, one viable means of implementing an SRM as specified in Section 3.2.2 is to simply add a loss at some desired bisection point immediately before a ReLU operation, moving the decision surface up or down such that the non-zero elements that will come out of the ReLU will fit the sparsity requirements of an SRM (Fig. 3.2). The loss would be applied prior to the ReLU so that the network could be encouraged to produce more outputs when the SRM loss resulted in too sparse of an output.

Figure 3.2: Network bisection SRMs require a high level of sparsity, which may be achieved through the ReLU operations already present in many NNs. The NN’s output at any given layer may be made more or less sparse simply by moving the entire decision surface one direction or another through an auxiliary loss (Eq. (3.14)).

3.3.1.1 Methodology

ResNets are divided into several super blocks, all but the first being preceded by a batch normalization layer, an activation layer, and an average pooling layer. In a CIFAR10 network, where inputs are 32×32 , there are 3 super blocks, and the final classification result is calculated from a globally averaged pool across the resulting 8×8 feature map [16]. The beginnings of these super blocks, following the batch normalization and preceding the activation and average pooling layers, are natural points to bisect a network in an effort to understand its function.

To prevent degeneracy, where the SRM over-emphasizes sparsity and passes no information to the classifier, the sparsity loss and task-specific losses must be balanced against one another. One method of achieving this balance is by introduction a term for sparsity homeostasis, where a specific average amount of activation in $\mathcal{S}(\mathbf{X})$ is targeted. Targeting an average amount of activity is a technique frequently used in unsupervised learning problems [69], and makes the sparsity problem more concerned with forwarding the correct information than determining how much information to forward. By adjusting the portion of space in $\mathbf{X}$ that has a non-zero norm across $\mathcal{S}(\mathbf{X})$, I was also able to target different

quantities of the input image being highlighted by the explanation.

To hit such a homeostasis point the SRM loss needed to be balanced against the classification loss. It seemed sensible to begin by using a PI controller designed to target an average activity level in the SRM’s output. Here, activity level refers to the percentage of input pixels which could have affected the final classification decision. In a traditional ResNet, basically every convolution uses a 3×3 feature map, meaning each spatial element is allowed to pull in information from its neighbors. To achieve point 2 of Section 3.2.2, I modified the ResNet such that most convolutions only used a 1×1 filter, and 3×3 filters were allowed only prior to the final output from each block. These modifications allow for the neighborhood of input pixels which might affect a pixel at the SRM level to be precisely calculated. The total active input area may then be calculated. For example, if two 3×3 convolutional kernels are allowed prior to the SRM layer, then each SRM pixel is affected by a 5×5 patch of the original input. If one pixel had activity at the SRM level in a 10×10 image, then the final activity level would be 25 %. In practice, these activity levels were calculated by dilating the SRM layer by its neighborhood size, clipping to the original image’s dimensions, and then measuring the percentage passed.

Through the PI controller, the sparsifying loss would only be applied with significant magnitude when the average-case input resulted in more of the input’s spatial domain being utilized than desired, and otherwise the sparsifying loss would be decreased such that the classification loss could optimize the final objective. In order to utilize as much of the network as possible, the target sparsity was adjusted from 100 % (utilizing the full image) to a target percentage throughout training, using a cosine annealing schedule. To prevent domain issues, the inhibition or

excitation of elements in the SRM were clamped by two constants, α and β . These served to prevent elements that were frequently inhibited from becoming too negative and ceasing to participate in the network's training. Altogether, the loss for the SRM was:

$$r(t), \text{ the accumulated sparsity error,} \quad (3.2)$$

$$r(0) = 0, \quad (3.3)$$

$$c, \text{ the number of features in the SRM layer,} \quad (3.4)$$

$$y, \text{ the height of the SRM layer,} \quad (3.5)$$

$$x, \text{ the width of the SRM layer,} \quad (3.6)$$

$$s(t), \text{ the output of the SRM, a } c \times y \times x \text{ matrix,} \quad (3.7)$$

$$a_{target}, \text{ the target percentage of the input passed,} \quad (3.8)$$

$$a_{real}(t), \text{ the actual percentage of the input passed,} \quad (3.9)$$

$$a_{real}(t) = \left(\sum \text{dilate}(\max(s(t) \text{ across } c) > 0) \right) / (yx), \quad (3.10)$$

$$\alpha = -0.54, \quad (3.11)$$

$$\beta = 0.1, \quad (3.12)$$

$$\omega(t) = e^{-4.2+0.2|r(t)|}, \quad (3.13)$$

$$L_{srm}(t) = \omega(t) \sum \begin{cases} \max(s(t) - \alpha, 0)^2, & \text{if } a_{real}(t) > a_{target} \text{ and } r(t) > 0 \\ \max(\beta - s(t), 0)^2, & \text{if } a_{real}(t) < a_{target} \text{ and } r(t) < 0, \\ 0. & \text{otherwise} \end{cases} \quad (3.14)$$

$$r(t+1) = r(t) + a_{real}(t) - a_{target}. \quad (3.15)$$

Above, (t) denotes a time index. This loss decreases activity only for input samples which violate the sparsity constraint and only when the overall network needs to be more sparse, and is capable of increasing activity when the network becomes overly sparse. The exponential factor on the error integration, $\omega(t)$, keeps the loss function responsive to sudden changes in the network. Note that all constants were optimized for validation accuracy via a distributed search utilizing a swarm of hill climbers.

3.3.1.2 Results

A basic SRM consisting of several convolutional layers leading to the SRM output leading to several more convolutional layers, a global averaging layer, and finally a fully-connected layer was built using PyTorch [49] and applied to the CIFAR-10 dataset [31]. Batch normalization was used after each convolutional layer’s activation, which was chosen as ReLU. The test networks had $573k$ parameters in total. The SRM’s requirements from Section 3.2.2 were realized 1. by zero-padding the convolution output to preserve the same dimensionality as the input, 2. by composing the SRM as a set of convolutions, and 3. as the above auxiliary loss, $L_{srm}(t)$, applied to the SRM output that emphasized sparsity on a dilation of the spatial elements forwarded by the SRM to the second half of the NN.

Shown in Fig. 3.3 are the error rates through training on CIFAR-10 for a network with and without the SRM loss component, targeting a final activity level of 95 %. The addition of the SRM loss consistently improved results on CIFAR-10, boosting the error rate down to 8.0 % with the SRM loss compared to a 8.2 % error rate without the SRM loss, a 2.4 % reduction in error rate. Note also that the training error for the SRM is higher, but the test error is lower - indicating

Figure 3.3: Example comparison on CIFAR-10 task between an SRM network with a 90 % sparsity target and a traditional, non-SRM network. Each batch consisted of 64 training examples.

better generalization properties.

While small, this improvement was non-trivial, as approaching an error rate of 0 % becomes increasingly more difficult. An improvement in testing accuracy showed that restricting the expressible space as specified in Section 3.2 encourages the NN to learn the right kind of generalizations. This result was with a rudimentary SRM; improving the regularization quality of the SRM for a more significant improvement in accuracy is one goal of this dissertation.

A separate experiment was targeted at 26 % final sparsity. The SRM's output, shown in Fig. 3.4, clearly shows that the network learned to consider only the relevant portions of the input. This figure was generated with different sparsity settings than Fig. 3.3, and resulted in doubled error rates, but showed the feasibility of interpretable results via an SRM. The images second-from-the-right show that only features relevant to the classification of the picture were transformed and

Figure 3.4: Example inputs (left), SRM activations before clipping (left-middle), SRM activations after clipping (middle), and actual regions of input used for final classification (right-middle). The right-most image is the same visualization as the SRM outputs applied to a traditional NN that does not use point 3 of Section 3.2.2.

passed to the classification network. In the elk example, the horns and torso were enough to confidently identify the animal, whereas with the fixed-wing aircraft, the wing, its landing gear, and the curvature of the fuselage were all deemed significant to classification. A traditional network rendered with this technique shows activity everywhere, and the magnitudes are difficult to interpret.

While these initial results were promising, they did not provide resistance to adversarial examples. Somewhat counter-intuitively, even though only 26% of the input space was used in the final classification, adversarial attacks against this network were of a similar or smaller magnitude as attacks against a traditional, non-SRM network. Furthermore, multiple bisections with this method were somewhat obtuse as the target activity level at each region would fix that layer’s behavior. The goal of SRMs was to produce explainable networks, not to constrain them unnecessarily. As such, another technique for producing SRMs out of network bisections was explored.

3.3.2 Gradient Integration Method

As mentioned in Section 3.1, gradient information has been used in the past as a means of understanding the operation of NNs in the context of individual input examples. While no information on the neighborhood for which gradient information remains valid is demonstrated by these methods, they do produce a linear approximation of the most relevant parts of each input. This information is very noisy, however; see Fig. 3.5. Even small, random changes in the input or network can change the saliency map significantly. Adversarial attacks change the saliency map even more significantly. This indicates that gradient information at the network’s input is not a stable indicator of the calculations performed by the network. Given how non-linear NNs are, this is not terribly surprising. Nonetheless, the relevance of gradient information to a network’s calculations is undeniable.

This section of my work focuses on providing an SRM via a relevancy filtering mechanism. Gradient information cannot be ignored as a potential source of relevancy information, and as such this section revolves around exploring ways to utilize that information that both would not impede the network’s ability to perform inference and would provide some intuition.

3.3.2.1 Methodology

This work concerns an SRM achieved via network bisection by inserting *Gradient Integration Modules* (GIMs) as shown in Fig. 3.6. We start with the pre-activation version of ResNet-44, from [17]. The batch normalization operation is typically defined in two parts: a normalizing step and a set of affine transformations which give flexibility to the output distribution [26]. We insert the GIM module between these two stages of the batch normalization operation. Each GIM contains a

Figure 3.5: Saliency map given an input for both (a) original input, (b) random perturbation, and (c) adversarial perturbation. Note that the saliency maps change drastically based on minor changes to the input.

secondary, approximating network that maps feature vectors to a statistic relating whether or not those vectors are relevant to the overall network’s loss function. This approximating network is detached from the primary network - that is, its gradients are not backpropagated beyond the input to the approximation network. Detaching the statistics network allows the primary network to focus solely on its loss function, as it would without the SRM modifications. The approximating network’s output is used to set channels deemed irrelevant to zero, which thanks to the normalizing part of the batch normalization module, is the expected value. After this gating, the learnable scale and bias are applied, followed by a ReLU. The only difference between this arrangement and the original pre-activation ResNet-44 is that there is a chance for each piece of information to be reset to the batch

Figure 3.6: Architecture of network, using GIM modules to produce an SRM. At top is the architecture for a traditional ResNet - note that the “GIM” modules do not exist in a traditional network, and are underlined in red to note this. The inputs to an NN are typically already zero-mean and unit-variance, hence the lack of a batch normalization around the first GIM module. Within the Pre-block, the GIM is inserted between the two steps of the batch normalization operation (normalization and the subsequent affine operations, which are added to support a wider variety of output distributions). Within the GIM, a detached version of the information into the module is sent to a statistics network, which in turn produces a spatial mask by which features are multiplied, as per Section 3.3.2. The loss function used to train the statistics network is specified in Section 3.3.2.

normalization module’s learnable bias value. The modification by the GIM is thus strictly a binary operation applied at each spatial point - all channels at each point are either preserved or dropped. Whether or not the drop occurs depends on statistics unavailable to the NN’s backpropagation, and as such cannot be amplified directly in a way that leads to complex behavior. Rather, the NN must configure itself to work around these drops in order to achieve good accuracy with the increased sparsity.

As mentioned previously, gradient information is unreliable. Even with an identical input, minor changes to the network will produce significant changes in the gradient information. We address this by considering the vector of information at each spatial point in the input, which produces a smoothing effect due to the

number of gradients included, and also by storing this information through an integrator rather than trying to store it precisely via an L2-norm.

The traditional method of storing non-classification data in an NN is via the L2-norm, otherwise known as the *Mean-Squared Error* (MSE) loss function. The MSE is typically preferred because, given enough gradient descent steps, the final solution is stable. That is, there is a point at which the sum of gradients across all training examples will be zero, even if several training examples have identical inputs and different outputs:

$$\begin{aligned} L &= (y - \hat{y})^2, \\ \frac{\partial L}{\partial \hat{y}} &= -2(y - \hat{y}), \\ x_0 \rightarrow y = 1, x_1 \rightarrow y = 0, x_2 \rightarrow y = 0, \\ 0 &= \sum_{x_i} \frac{\partial L}{\partial \hat{y}}. \end{aligned}$$

On the other hand, an integrator stores the mode, not the mean, and as such is unstable:

$$\begin{aligned} L &= \hat{y}y, \\ \frac{\partial L}{\partial \hat{y}} &= y, \\ 0 &\neq \sum_{x_i} \frac{\partial L}{\partial \hat{y}}. \end{aligned}$$

Ordinarily, this instability is undesirable. However, integrators are good at filtering out high-frequency noise. For unreliable information, where attempting

to store a stable, converged value is nearly impossible, the integrator is a reasonable choice.

Assume that the input to the GIM at each spatial point is a vector $\vec{x}$ composed of N channels. When considering if that information vector is necessary for classification, we want to measure the dangers of reassigning $\vec{x} = \vec{0}$. Then, the natural way to store gradient information in the SRM as a relevancy statistic y for a corresponding spatial point with input $\vec{x}$ and a corresponding gradient g_i might be:

$$L_y = y \sum_{i=1}^N -x_i g_i.$$

However, there are several issues with this. For one, g_i is a linear approximation of a highly non-linear system; relative to the magnitude of x_i , it might change drastically. Second, the gradient will in turn affect subsequent values of x_i , as SGD progresses. Thus, rather than x_i , we use its sign:

$$L_y = y \sum_{i=1}^N -\text{sgn}(x_i) g_i.$$

In practice, one additional alteration is beneficial. SGD is used with such high learning rates that supervised NNs often overshoot their target values. This leads to chaotic interactions amongst parameters such that the expected value of each gradient is almost zero. While the above integrator equation would eventually filter out this high-frequency information, it is easiest for us to rectify the

signal immediately using a ramp function $r(\cdot)$, and only consider the positive contributions of each spatial point. Additionally, to keep some negative values, we introduce a constant c :

$$L_y = y \left(\left[\sum_{i=1}^N r(-\text{sgn}(x_i)g_i) \right] - c \right), \quad (3.16)$$

$$c = \frac{\lambda}{Y} \sum_{j=1}^Y \sum_{i=1}^N r(-\text{sgn}(x_{i,j})g_{i,j}), \quad (3.17)$$

where Y is the number of spatial points with statistics being calculated. That is, c is based on the entire image input, whereas L_y is based predominantly on information at the spatial point being considered for relevancy.

This loss equation implements an integrator that approaches $\inf$ when the expected contribution to the primary network loss is greater than c , and approaches $-\inf$ when the expected contribution is less than c . During backpropagation, while training our networks, we have no need to consider only local values. As such, we can set c as some constant λ times the average contribution across the entire image. This guarantees that only uniquely beneficial information approaches $\inf$, while information that is less than average approaches $-\inf$. The constant λ allows us to control the sparsity of the final stored values. To enforce stability, we clamp the loss if $y > 1$ and the loss function would cause y to increase further, or if $y < -1$ and the loss function would cause y to decrease. Values of the stored statistic y greater than or equal to 0 were transmitted to the second half of the network, while values of y less than zero resulting in the corresponding information vector being reset to $\vec{0}$.

3.3.2.2 Results

A ResNet-44 [16] was used for these trials, again using PyTorch [49] and applied to the CIFAR-10 dataset [31]. The requirements for a useful network bisection SRM, from Section 3.2.2, were satisfied the same as in Section 3.3.1, except for point 3. Point 3, the requirement of sparsity in the spatial domain of the input, was not guaranteed unless a sufficient number of spatial points at the SRM layer were less than zero as described in the methodology section. Multiple different values of λ from Eq. (3.17) were tried, with similar results. Note that $\lambda > 1$ would be unstable, and any questionably-useful spatial locations would default to not being transmitted to the classifier half of the network.

Since the statistics network for the GIM is detached from the standard network, one nicety of the GIM approach is that measurements may be taken without affecting the network if the zeroing of feature vectors is disabled. For these experiments, the GIM module replaced the batch normalization preceding all 3 super blocks and the final, global average pooling layer. However, only the GIM module leading into the 3rd super block was allowed to filter statistically irrelevant spatial locations in the SRM.

Using $\lambda = 0.9$, a ResNet-44 implementation which scored 92.2% accuracy instead scored 94.0% accuracy with the GIM filtering. This improvement is consistent with other attention literature on similarly-sized networks on CIFAR-10 [28, 65].

The spatial regions of the input which registered as significant with the GIMs are demonstrated in Fig. 3.7. Note that either increasing λ or decreasing the learning rate of the GIM resulted in a sparser intermediate representation, with $\lambda = 0.99$ scoring an accuracy of 93.6% and the combination of $\lambda = 0.99$ with a

Figure 3.7: Example GIM activations on several images. The full-color images are the inputs; the following four columns are GIM activations before the first, second, and third super blocks, and finally before the global averaging layer. Only the third column of GIM activations were used to affect the network; the others were passive observers. Black indicates a GIM output of -1 , red indicates a GIM output of just below zero, and white indicates a spatial location which was passed on to the final classification result, with a brighter white being closer to 1. The three rows illustrate three different networks: one with $\lambda = 0.9$, another with $\lambda = 0.99$, and a third with both $\lambda = 0.99$ and the learning rate for the GIM module set to 0.01 times the learning rate of the parent network, allowing for more gradual learning of relevance statistics. Note that the final column of each of these is a flat color, indicating no meaningful statistics were captured at that level.

learning rate of 0.01 times the base network learning rate scoring an accuracy of 93.4%. While the statistics do, by definition, show significant patterns at each level in the network, it is impossible to express with certainty why these patterns were considered significant, or what would need to change in the input image to change the final classification result.

3.4 Discussion

While the above formulations of a network bisection SRM worked well enough to appear useful at first inspection, the results were discouraging in several ways.

Adversarial examples against these networks - even with the very sparse network from Fig. 3.4 - were of comparable or lesser *Root-Mean-Square Error* (RMSE) as adversarial attacks against images on a traditional network with the $L_{srm}(t)$

loss. Even though the majority of the input image was unused in the final classification, adversarial attacks against the network resulting in similar-magnitude perturbations on the unmasked input elements, resulting in a lower RMSE overall. In other words, while the second half of the network was making decisions based on objectively fewer input elements, it seemed that those input elements in turn demonstrated a greater effect on the output of the network.

A second disappointment of this approach was that when the network misclassified an image, the saliency maps at the SRM level were not necessarily informative. The highlighted regions often denoted the whole salient object, even though the resulting classification was incorrect. The introduction of this dissertation referred to Jetley *et al.*'s work on attention models [28], shown in Fig. 1.1. Similar to the network bisection SRM presented in this dissertation, Jetley *et al.*'s model could convincingly highlight salient regions, yet still produced classification errors. The SRM could theoretically be used to craft more informative saliency maps based on which features were forwarded, rather than the binarization of whether or not any features were forwarded, but that would also be vulnerable to adversarial examples.

A third disappointment of this approach was that hierarchies were not very informative. While the GIM approach was capable of producing hierarchies (Section 3.3.2), the produced hierarchies did not seem very useful with regards to diagnosing the network's flaws or reasoning. These hierarchies were more or less consistent with either Li *et al.*'s [35] or Jetley *et al.*'s [28] works. However, in light of adversarial examples, the utility of any of such explanations is questionable.

The failures of network bisection SRMs were thus apparent at this point. Compared to previous explanation methods, they shared the same lack of resistance to

adversarial examples, and ultimately were still a heatmap-based method of explanation, which was identified as having limited utility in explaining why a region was important.

Section 2.4 raised the question of how accuracy might degrade as the input to the classifier became more sparse. As with attention models, a mild restriction on the classifier’s input improved performance. However, the model which only sent 26 % of the input to the classifier resulted in a classification accuracy of 75 %, down from 92 %. Therefore, it is safe to assume that this shall remain a general problem, which each dataset having its own curve of sparsity versus achievable accuracy.

Sensory Relevance Models via Adversarial Explanations

Despite classifying based on a small number of pixels from the input image, network bisection SRMs were susceptible to adversarial attacks of an RMSE comparable to those against traditional networks. Furthermore, they shared the flaws of other methods of explaining NNs, as heatmaps only highlight a region without indicating the significance of that region. Compare this to software for editing grammar: only highlighting a piece tells you nothing about why it was highlighted, leaving the user to guesswork (here because the word “piece” is awkward). All grammar editing solutions offer at least mouse-over help explaining the highlight. Explanation methods for NNs, which utilize only heatmaps, are missing this vital information. Figure 1.2 demonstrates the ambiguity of applying a heatmap-based explanation to a classification problem.

To elucidate the failures of traditional explanation methods for NNs, and the failures of network bisection SRMs, it was necessary to devise a scheme which achieved the original goals of the SRM. One of the failings of heatmap-based methods was that all of them were necessarily produced via linearizations of a highly non-linear network. The network bisection SRMs did not possess this particular issue, but the neighborhood of validity for the network’s decision around any input image was tiny, demonstrated by adversarial examples. It was determined in Chapter 3 that reducing the portion of the input used to make a decision did not increase the magnitude of adversarial examples. Therefore, a successful means of explaining NNs would require further analysis of adversarial examples.

Figure 4.1: While adversarial examples are considered a nuisance by most, they have the potential to provide reliable explanations with the same richness of information as the original input. For example, when an NN trained at finding lung nodules in radiographs needs investigation (a, b), an attack may be targeted at a desired new network output—such as changing a nodule classification to a non-nodule classification (c) or emphasizing the nodule (d)—to produce a new image which is minimally changed but produces the desired output. By comparing these inputs and looking at the differences, a human operator can identify relevant features in the input with greater fidelity than current methods.

At this point, it became clear that adversarial examples themselves might provide useful explanations. Adversarial examples are functionally equivalent to pushing the input toward or away from decision boundaries. This is illustrated in Fig. 4.1.

This line of work was successful, and advanced the field of explanation methods for neural networks in a meaningful way. As a result, adversarial explanations were detailed in the attached work, Appendix A.1. Much of this chapter is taken from that paper, inserted here to provide the main text of this dissertation with a complete and contained narrative.

4.1 Related Work

Three branches of ML inquiry led to this work: adversarial attacks, explanation techniques, and *Human-In-The-Loop* (HITL) methods. A mathematical branch, Lipschitz continuity, was also important in developing this work. These four related areas are discussed in Sections 4.1.1 to 4.1.4.

4.1.1 Adversarial Attacks

The struggle between adversarial attacks and methods of resisting them is perhaps best illustrated by the saddle point formulation proposed by Madry *et al.* [41] (notation adapted to be consistent throughout the current paper):

$$\min \mathbb{E}_{(x,t) \sim D} \left[\max_{\delta \in S} L(\theta, x + \delta, t) \right]. \quad (4.1)$$

That is, find the network parameterized as θ which, according to dataset D , produces the best approximation of some target t when the worst-case noise δ constrained by an allowed attack space S is added to an input x . This formulation illustrates the difficulty of working against a high-quality adversary, which is relatively unrestricted in its exploitation of the network’s properties around x .

Methods of generating adversarial attacks approximate the inner maximization problem from Eq. (4.1). Carlini *et al.* [10] compared several different methods of generating attacks, including Goodfellow *et al.*’s *Fast Gradient Sign Method* (FGSM) [15] and their own *Projected Gradient Descent* (PGD) [10]. Carlini *et al.*

[10] showed that attacks were transferable between networks, regardless of network architecture. One interpretation of this would be that NNs emphasize high-frequency signals within the data over low-frequency signals, biasing them toward changes which are imperceptible in the domain of the input. Recent work by Tsipras *et al.* [63] argued that this might be due to the natural tendency of high-accuracy classifiers to exploit small differences as a means of greedily leveraging available information.

Methods of providing robustness against adversarial attacks approximate the outer minimization problem from Eq. (4.1). To our knowledge, state-of-the-art methods of resisting adversarial attacks currently revolve around either adversarial training [41, 51, 63] or randomized smoothing [11].

Madry *et al.* [41] investigated using both the FGSM and PGD methods of generating adversarial examples, and the effects of using these methods to train networks, a technique called adversarial training. Note that, under adversarial training, a network is consistently trained based on its worst performance point in the neighborhood of each input. While training with adversarial examples was shown to somewhat reduce the transferability of attacks, they found that attacks still produced a significant reduction in NN accuracy. They also argued that any defense mechanism shown to be robust against PGD would be robust against other first-order attacks [41]. That group later expanded their theories on adversarial training in work by Tsipras *et al.* [63], demonstrating salient features materializing in adversarial examples with large limits on allowed attacks. These attacks used perturbation magnitudes which greatly surpassed the threshold at which the classifier’s accuracy would necessarily change, but for the first time demonstrated that classifiers could potentially be used for generation.

Cohen *et al.* [11] recently improved on a body of work called randomized smoothing, a provable method of inducing L_2 robustness based on evaluating a smoothed version of a network trained with Gaussian noise. Unlike techniques such as Madry *et al.*'s [41] or our work in Section 4.2, Cohen *et al.*'s method allows for a certifiable calculation of an adversarial resistance bound. That is, there may or may not be attacks against networks that exist, but are difficult to find with PGD. If these attacks exist, Cohen *et al.*'s method provides a high level of confidence that the smoothed network would also protect against them, even though these attacks cannot currently be generated. The price of this certainty comes with somewhat inflated processing time: predicting requires about 100 evaluations for each input to properly compute the smoothing function [11].

Other defensive techniques have been proposed but were either inadequately tested or shown to be broken. An approach which denoised inputs won the NIPS 2017 adversarial robustness challenge [33, 36], which was successful but defended against a static set of attacks against a standard network rather than against attacks specific to the defended network. A number of stochastic and non-differentiable defenses have been proposed and subsequently shown to be vulnerable to attacks which take these qualities into account [3].

We note that a roundup of best-practices for ensuring that new defenses are effective was recently authored by Carlini *et al.* [9]. In the context of the current work, we've complied with many of their recommendations, excepting non-gradient based attacks and an investigation of attack transferability. As our proposed techniques only affect network regularization (Section 4.2) or make gradients less obfuscated (Section 4.2.4), sticking to a gradient-based PGD-variant attack seemed sufficient. Attack transferability was not investigated as the proposed models have

Figure 4.2: Compared to the state-of-the-art, our method achieved higher accuracy while being tolerant of greater adversarial perturbations on both ImageNet (Fig. 4.2a) and CIFAR-10 (Fig. 4.2b). By measuring the ARA, a model’s resistance to adversarial attacks is taken into consideration alongside its ability to make predictions better than the naive baseline (the hatched region). See Section 4.1.1 for additional details.

* Madry *et al.* used a network that was $10\times$ as wide as a traditional ResNet-110, and also trained against an L_{inf} adversary rather than an L_2 adversary [41].

† This curve came from personal communication with A. Madry on a standard ResNet-50; see Section 4.1.1 for details.

identical architectures and processing, and therefore, for a given input, increasing the required attack perturbation magnitude necessitates that an attack would not transfer.

For comparing adversarial defense techniques, the current work used accuracy-versus-attack-magnitude plots, such as Fig. 4.2. This type of plot shows how an individual classifier’s accuracy would fall as the allowable attack space S increased. For consistency across datasets, regardless of their input dimensions, we used the

Root-Mean-Squared Error (RMSE) for the shown attack distances, which is equal to the L_2 norm of the perturbation divided by the square root of the number of elements in the perturbation. The RMSE is a more natural choice as it is scaled such that an RMSE of 0 means no change and an RMSE of 1 means the change between an all-black and all-white image, regardless of size. Figures shown in this work have scaled e.g. ϵ and other values reported in other works to the RMSE scale. Cohen *et al.* [11] presented this plot for L_2 attacks against both ImageNet and CIFAR-10 results, and Madry *et al.* [41] presented this plot for L_2 attacks against a CIFAR-10 classifier with $10\times$ the normal number of filters and trained against L_{inf} adversarial examples. Personal communication with A. Madry yielded the additional curve on Fig. 4.2, which was for a standard CIFAR-10 ResNet-50 trained against L_2 attacks with $\epsilon = 0.009$; we independently trained a similar network using their methods and achieved similar results, and show improved results for adversarial training with a slightly different adversary (Section 4.3.1.15).

To compare these curves using a single number, we’ve used the area between the curves of a naive classifier and the classifier in question, a metric we’ve termed the classifier’s *Accuracy-Robustness Area* (ARA). The ARA is illustrated in Fig. 4.2b, where the shaded area for each classifier is the area computed for the ARA. We found that, on CIFAR-10, a standard ResNet-44 had an ARA of 0.0013, extrapolating numbers from Madry *et al.*’s best L_2 resistant network (from personal communication) yielded an ARA of 0.0124, and Cohen *et al.*’s numbers yield an ARA of 0.0065. A reproduction of Madry *et al.*’s best L_2 resistant network, but as a ResNet-44 instead of ResNet-50 and using our algorithm for evaluating ARA in Section 4.2.1.1, yielded an ARA of 0.0104. On ImageNet 2012, a standard ResNet-18 had an ARA of 0.0004 and Cohen *et al.* used their method to demonstrate an

ARA of 0.0022.

4.1.2 Explanation Methods

Inadequate understanding of the internal operation of NNs, or the larger toolbox of ML solutions in general, has recently come under focus as a primary difficulty of using them [4, 34, 44, 54, 58, 59].

Preliminary attempts at addressing this problem involved looking at saliency maps computed via backpropagation to see which input pixels had the largest effect on the classification [21, 59] or looking at the receptive fields to which internal nodes respond [4, 39, 47, 75]. However, these techniques often produce very noisy images that are difficult to interpret.

Works such as Ribeiro *et al.*'s LIME [54] or Selvaraju *et al.*'s Grad-CAM [58] proposed improvements over raw saliency maps. These works highlighted the difficulty of identifying the reason behind incorrect classifications, and proposed their own solutions to the problem. LIME presented a generalized method suitable for both image and non-image inputs that boils down to set theory: if part of the input were masked, would the overall classification get better or worse? By noting which parts of the input make the most significant difference, the LIME algorithm derives a linear classifier which approximates the non-linear NN, and the linear, approximating classifier is then used to produce a mask for the input that highlights salient regions [54]. On the other hand, the Grad-CAM algorithm harnesses backpropagation directly to derive an expression for localizing the most salient regions [58]. Its innovation came from measuring image region contribution at a layer closer to the classifying end of the NN than the input. Both of these ultimately used linear techniques to describe a non-linear NN.

While these methods produced reasonable explanations for the examples provided in their papers [54, 58], these methods provide little to no guarantee about their validity. To evaluate their validity, we considered the relation between explanations and adversarial examples: if a method explaining an NN’s decision were accurate, then an adversarial example should primarily change the regions of the image marked salient by the explainer.

Consider the lung nodule dataset published by the *Japanese Society of Radiological Technology* (JSRT). This dataset consists of 247 chest X-rays, 154 of which have a single, annotated nodule, each of which show up as dark, solitary shadows. Chest X-rays are notoriously difficult to read, and so the JSRT dataset is evaluated by nominating 5 candidate points which might be proximal to a tumor. An algorithm’s output is considered correct if a tumor center is within 2.5cm, or 143px, of any such candidate point. Scores are thus presented as sensitivity given at most 5 false positives per X-ray. We trained a network on this problem which scored 66% sensitivity by this rubric and applied LIME and Grad-CAM to the NN, yielding Fig. 4.3 and Fig. 4.4 respectively. Considering the output of LIME for this classifier, Fig. 4.3, we see that the produced explanation neither makes intuitive sense nor instills confidence in the classifier. For the same network, Grad-CAM produces a smoother, more intuitive explanation (Fig. 4.4). One could reasonably infer from this explanation that the network is paying attention to the high-contrast borders of the nodule, a reasonable approach for the problem. However, this human-oriented interpretation of Grad-CAM’s output would be very inaccurate; when compared with the adversarial example generated for the same NN in Fig. 4.5, there is no correlation in overlap between the adversarial perturbations and the salient region demarked by Grad-CAM. As such, it is hard to

Figure 4.3: A demonstration of LIME on an NN diagnosing lung nodules in the JSRT dataset. LIME’s algorithm consists of taking an input (a), dividing it up into super-pixels (b), and then using linear approximations to determine which subset of super-pixels most significantly affects the network’s classification (c). How well does this explain the decision?

confidently state that the explanations yielded by LIME or Grad-CAM represent dominant factors contributing to the NN’s output.

Attention models involve modifying the NN’s architecture to both enhance overall NN performance and provide an attention mask which may be viewed as an explanation mechanism [12, 64]. Where saliency maps demonstrate the most-used pixels, attention models use a gating technique which involves multiplying the inputs by a learned mask to shape how inputs are forwarded to the classifying portion of the network. Viewing the mask of the attention model shows which inputs were weighted more heavily for a given input. However, attention models are still vulnerable to adversarial attacks which exploit only a small portion of the masked region.

Also noteworthy is Bau *et al.*’s “Network Dissection” work [4]. By cross-referencing the activation map of convolutional neurons in an NN with object annotations, an IoU was computed on a per-neuron basis. They reasoned that neurons relating to objects via a large IoU were responsible for detecting that type of object or texture. However, the reported IoUs are quite small, with the

Figure 4.4: Selvaraju *et al.*’s Grad-CAM [58] processes an input (a) and uses back-propagation (b) to produce localized gradient information which can be presented as a heat map of salient regions (c). How well does this explain the decision?

majority of reported values being below 0.2—higher than coincidence, but lower than an authoritative explanation would merit. Bau *et al.* also made no mention of how adversarial examples relate to their work. The approach is worth continued research, but is not yet an end-all means of explaining NNs. More recently, Bau *et al.* have applied their dissection methods to *Generative Adversarial Networks* (GANs), and demonstrated that omitting neurons with high IoUs in GANs can predictably modify the generated images [5]. Nonetheless, that is a generative, rather than explanatory, function.

4.1.3 Human-in-the-Loop (HITL)

While heat maps and regions of interest are poor indicators of a robust explanation, they are not wholly irrelevant. Li *et al.*’s “Tell Me Where To Look” [35] demonstrates the merit of using annotated salient regions—a straightforward method of providing HITL functionality—as part of weakly-supervised training for NNs. They introduced an algorithm which extends Grad-CAM’s heat map to be differentiable, and then used gradient descent so that the highlighted region approaches

Figure 4.5: An adversarial example corresponding to the same network and input as Figs. 4.3 and 4.4. Here, the resulting adversarial input (a) is the sum of the original input (b) and a crafted noise term scaled by a small coefficient (c). The result is an imperceptible change in input leading to a completely different NN output. Also worth noting is that the location of the noise in (c) does not correlate well with either the LIME or Grad-CAM explanations.

that of annotated segmentations on a classification dataset. This method could be considered a proof-of-concept for HITL applications based on NN explanations, where each new human annotation ultimately becomes part of the training set for the algorithm. For segmentation, their algorithm resulted in an impressive leap from a mean *Intersection-over-Union* (IoU) of 0.555 with prior methods to 0.621 with theirs [35]. Additionally, for 10,000 classification training examples, they only used segmentation masks for 1,464 of the examples [35], illustrating that even partial annotations provided some benefit.

Ribeiro *et al.* [54] also used LIME in a HITL context by providing operators, who were unfamiliar with machine learning, explanations on how a classifier got to a decision. They first created classifiers that identified whether the topic from a text document was "Atheism" or "Christianity," from a dataset of 20 newsgroups. Explanations were created based on which words led to the classifier's answer via LIME and these explanations were shown to crowdsourced human operators via

Amazon Mechanical Turk to mark words that seemed irrelevant to the task. Two algorithms were used to pick words to show to the operators: explanations randomly picked and explanations deemed important via LIME. The classifier was then retrained with the features deemed by operators to be irrelevant removed, and the improved classifier was once again sent to human operators for additional word removal. The real world accuracy of the classifier improved from a baseline of approximately 57.4% to approximately 69.7% with randomly picked explanations shown to the operators and approximately 73.2% using explanations deemed important after 3 rounds of interaction. Importantly, this experiment demonstrated that HITL processes could be used to significantly improve classifiers after few iterations.

We explored an alternative HITL pipeline to improve the adversarial resistance within an NN, instead of improving accuracy. Our process is outlined in Section 4.2.8.

4.1.4 Lipschitz Continuity

Prior work has been performed on Lipschitz continuity, particularly with analyzing the stability of NNs [6, 66]. Briefly, Lipschitz continuity is the bounding of a function’s value, such that the function’s value is not allowed to change between two points more than a constant value times the distance between those points. This is often approximated as a bounding of the derivative. We differ from prior work in this area as we consider the entirety of the NN as subject to Lipschitz constraints, rather than each layer, and demonstrate this technique as providing viable resistance against adversarial examples. We also propose several other modifications to NNs, which complement enforcing Lipschitz continuity.

4.2 Methods

Adversarial examples in state-of-the-art networks, as in Fig. 4.5, do little to illustrate the inner workings of the NN for which they are generated. However, the potential exists for adversarial examples to be a very powerful, non-linear method of explanation. Tsipras *et al.* [63] demonstrated that non-minimal adversarial examples contained salient features when networks were adversarially trained. Here, non-minimal means adversarial examples which have not been optimized for minimal perturbation distance, but only for maximal loss on the objective function. The current work considers whether decision boundaries may be pushed out even further, such that minimal adversarial examples at class boundaries might demonstrate the removal of features salient to the original classification. While previous methods of explaining NNs rely on linearization techniques, adversarial examples make full use of the NN’s non-linearities. With targeted attacks, the boundary located could signify different salient aspects of the input stimulus, as in Fig. 4.1, if the decision manifold of the network were sufficiently congruent with the visual manifold. A practical demonstration of this theory may be seen through examples on the JSRT dataset in Figs. 4.6 and 4.7; these were implemented using the techniques described throughout this section.

We will first discuss our method of generating adversarial attacks in Section 4.2.1. Following that, the methods used to train robust NNs may be found in Sections 4.2.2 to 4.2.8, and finally a discussion of the datasets used in this work and the NN architectures chosen for those datasets is found in Section 4.2.9.

Figure 4.6: In contrast to Fig. 4.5, our preliminary technique of increasing adversarial distance concentrates the perturbations needed to change the network’s output. The RMSE (c) between the original input (a) and the adversarial attack (b) is also much greater for a smaller change in network output.

4.2.1 Adversarial Attack Generation and Evaluation

Adversarial attacks were conducted with two separate goals within this paper: Section 4.2.1.1 contains the methodology for adversarial attacks aimed at reducing the classification accuracy of a network, and Section 4.2.1.2 contains the methodology for adversarial attacks aimed at producing classification explanations.

4.2.1.1 Adversarial Attacks on Accuracy

Untargeted adversarial attack generation for the evaluation of models followed Algorithm 1; this was a variant of Carlini *et al.* [10]. Rather than pursuing both target loss maximization and L_2 error minimization simultaneously, we found that alternating between these two to traverse some restriction on the adversarial example’s network output allowed for better automatic balancing between the two errors, resulting in smaller perturbation magnitudes. In contrast to the attacks presented by Carlini *et al.* [10], the algorithm presented will not begin a magnitude refinement before the target classification error is reached. The threshold at

Figure 4.7: Example of emphasizing a lung nodule via the same method of adversarial attack. Same network as Fig. 4.6.

which Algorithm 1 switches between minimizing the correct class’ post-softmax prediction s_t and minimizing the attack magnitude is defined by $g(s, t)$.

We present two choices of $g(s, t)$ for the current work. The first, $g_{adv}(s, t)$, was the well-known adversarial attack metric used by all prior work in this field [11, 63], and denotes the boundary at which top-1 accuracy would decrease:

$$g_{adv}(s, t) = \begin{cases} 1 & \text{if } s_t - \max_j(s_j; j \neq t) < 0, \\ 0 & \text{otherwise.} \end{cases} \quad (4.2)$$

This was the $g(s, t)$ used to produce Fig. 4.2. When ARA values are reported for a model, we evaluated random validation or testing images until we had 1,000 which were correctly classified. We then made a list of the RMSEs below which each image would retain the correct classification, minimized as per Algorithm 1. This list was extended with 0s for each image evaluated which was initially incorrectly classified: if a model scored 70% classification accuracy on unmodified images, we would have a final RMSE list of about 1,429 in length, 1,000 of which were

Algorithm 1: Process used to generate adversarial examples.

Input: N , the number of attack-optimizing steps; $f(\cdot)$, the NN; x , the network input; t , the true class of the input; $o(\cdot)$, an optimizing method such as SGD with momentum; $g(s, t)$, a goal function returning true if the network outputs from the attack are suitably different from the true class t ; η , a balancing term between categorical loss and MSE loss.

Output: δ_{best} , the adversarial noise which satisfies the goal $g(\cdot)$ and has minimal vector length.

```

1 begin
2    $\delta \leftarrow \vec{0}$ 
3    $M_{best} \leftarrow \text{inf}$ 
4   for  $n \in [0, \dots, N - 1]$  do
5      $\hat{x} \leftarrow c(x + \delta)$  //  $c(\cdot)$  clips elements of its argument to a valid
        input range, e.g.  $[0, 1]$ 
6      $y \leftarrow f(\hat{x})$ 
7      $s \leftarrow \text{softmax}(y)$ 
8     if  $g(s, t)$  then
9        $\Delta\delta \leftarrow 2\delta$  //  $L_2$  loss for magnitude
10      if  $\|\delta\|_2 < M_{best}$  then
11         $M_{best} \leftarrow \|\delta\|_2$ 
12         $\delta_{best} \leftarrow \delta$ 
13      else
14         $\Delta\delta \leftarrow \partial s_t / \partial (x + \delta)$ 
15         $\Delta\delta \leftarrow \eta \frac{\Delta\delta}{\|\Delta\delta\|_2}$  // Fixed gradient magnitude
16       $\delta \leftarrow o(\delta, \Delta\delta)$  // Apply optimizer step

```

non-zero. This list was then evaluated for accuracy at different levels of RMSE, as seen in Fig. 4.2, and the area above the naive baseline was taken to produce the attack ARA metric.

In the context of Algorithm 1, we used $N = 450$, $o(\cdot)$ was a *Stochastic Gradient Descent* (SGD) optimizer with a learning rate of 0.01 and momentum 0.9, and $\eta = 0.55$. Examples of our attack against a regular JSRT network can be seen in Fig. 4.5, and against a regular CIFAR-10 ResNet-44 network in Fig. 4.13.

4.2.1.2 Adversarial Attacks as Explanations

In the context of explanations, however, we found the $g_{adv}(s, t)$ metric to be lacking. For example, consider two closely related classes from CIFAR-10: automobile and truck. These classes are often confused for one another, leading to a decrease in the magnitude of untargeted attacks for members of either class. With respect to the network’s ability to tell these two apart, g_{adv} remains a good metric. However, as a classifier learns to distinguish these related classes from the other unrelated classes, the s_j corresponding to these related classes might rise in tandem. The described phenomenon is illustrated in Fig. 4.8. This situation would indicate that the network possesses a greater capacity for deciding what is “automobile” or “truck” compared to the remaining classes, but the attack magnitude would not decrease as these two classes would still be easily confused. Since the confusion between these two classes is built into the problem, g_{adv} hits a ceiling beyond which an attack magnitude based on the g_{adv} metric cannot be improved. As such, we also considered *Better Than Random* (BTR) as a measure of the classifier’s knowledge of class-specific features. The BTR magnitudes were defined based on the distance between the classifier’s prediction and a prediction at which the true label’s valuation matches that of a random classifier. As shown in Fig. 4.8, the BTR quantity continues to increase even as related classes both become more confident predictions. Thus, g_{btr} (where V is the number of classes in the prediction) is defined as:

$$g_{btr}(s, t) = \begin{cases} 1 & \text{if } s_t < \frac{1}{V}, \\ 0 & \text{otherwise.} \end{cases} \quad (4.3)$$

Figure 4.8: The previously mentioned attack ARA metric depends on the relative confidence between the correct class and the second-most-confident class. For related classes such as “car” and “truck,” the distance between these two classes may not increase through training. However, by measuring the distance from the correct class back to a fixed baseline, as is done with the BTR ARA metric, improvements in feature recognition may be measured regardless of the presence of related classes.

We note that the numerical stability of BTR is guaranteed as resetting all pre-softmax outputs to 0 achieves the required condition. Note also that we deliberately chose a truly random classifier, and not a naive classifier, for unbalanced datasets. When calculating BTR ARA metrics from a population of adversarial examples created using g_{btr} , a naive classifier was still used as the baseline for the area calculation.

For actually generating explanations, it was most useful to follow $\partial s_t / \partial (x + \delta)$ up to a boundary RMSE, at which point RMSE would be minimized, a tick-tock method similar to Algorithm 1, but substituting a slightly different boundary criteria:

$$g_{explain+}(\delta; \rho) = \begin{cases} 1 & \text{if } \|\delta\| > \rho, \\ 0 & \text{otherwise.} \end{cases} \quad (4.4)$$

We note that $g_{explain-}$ is also possible, by modifying Algorithm 1 to maximize

Figure 4.9: Different explanation techniques using $\rho = 0.075$. (a) The original image. (b) A positive explanation for the donut class; we note alignment of the added “hole” with a wrinkle in the original sandwich bun. (c) A negative explanation for the donut class resulted in the removal of the round shape of the sandwich. (d) A positive explanation for the true class, sandwich, results in exposed contents (peppers or tomatoes), and the beginnings of lettuce. (e) A negative explanation for the sandwich class reveals homogenization of the bun’s texture, and further rounding out of the overall shape. (f) Positive explanation for a completely unrelated class, horse; legs were clearly added, and the textured area in the upper-left of the image is appropriated as a saddle.

the selected class loss rather than minimizing it. These techniques are demonstrated in Fig. 4.9.

4.2.1.3 Example ARA Metrics

For a traditional ImageNet ResNet-18, we measured an attack ARA of 0.0004 and a BTR ARA of 0.0013. For a CIFAR-10 ResNet-44, we measured an attack ARA of 0.0013 and a BTR ARA of 0.0014. Further discussion of these metrics in the

context of adversarial examples may be found in Section 4.3.1.

4.2.2 Defense via Lipschitz Continuity

An integral part of many white-box attacks, including Algorithm 1, involves following the gradient of some loss. The rate at which the output of the network might be changed is likewise dependent on that gradient. To see how this might affect classification networks, consider the softmax operation, here denoted as $s(\cdot)$, applied to the output of an NN, y :

$$s(y) = \frac{e^y}{\sum_{i=1}^V e^{y_i}}. \quad (4.5)$$

In the 1,000-class ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC 2012) challenge, there are 1,000 classes [56]. Assuming 999 of those classes have an NN output of $y_i = 0$, then a value for the remaining class of $y_j = 10$ corresponds to a confidence in class j of 95.7%. For a confidence of 4.3%, that value need only fall to $y_j = 3.8$. In reality, an adversarial attack also has the option of increasing the confidence of classes $i \neq j$ to reduce confidence of class j . If $y_j = 3.8$, and another $y_k = 6.2$, $k \neq j$, then the confidence of class j falls to 2.9% and class k skyrockets to 32.1%. In other words, instability on the output values will quickly overwhelm the softmax operation. If we assume locally-linear behavior of the network, this instability may be modeled by looking at the expected change in the network's output given some gradient information. Using $E_i[\cdot]$ to denote an expectation conditioned on i , N as the number of input elements, Δ to signify an actual value change, and ∂ to signify a variable's partial:

$$E_i [|\Delta y_i|] \approx E_i \left[\left| \sum_{j=1}^N \Delta x_j \frac{\partial y_i}{\partial x_j} \right| \right], \quad (4.6)$$

$$\approx \sum_{j=1}^N E_{i,j} \left[\left| \Delta x_j \frac{\partial y_i}{\partial x_j} \right| \right], \quad (4.7)$$

These quantities are neither independent nor equivalent,

but we will simplify them as such:

$$E [|\Delta y_i|] \lesssim N E [|\Delta x_j|] E \left[\left| \frac{\partial y_i}{\partial x_j} \right| \right]. \quad (4.8)$$

Equation (4.8) provides a loose guideline for targeting different values of $|\partial y_i / \partial x|$. In fact, as a network becomes more non-linear, Eq. (4.8) becomes less accurate.

To see how effective the guideline given by Eq. (4.8) was in practice, we built a ResNet-18 and trained it on ILSVRC 2012 training data, detailed in Section 4.2.9.2. Leveraging PyTorch’s automated differential engine, we collected gradients for one of the NN’s outputs, before the softmax, with respect to each of the 150,528 input elements ($224 \cdot 224 \cdot 3$). The mean absolute value of the computed derivatives then resulted in an aggregate number which summarized the network’s volatility in the original input space. For our ResNet-18, this value worked out to 0.051 /input. Interestingly, the mean of the maximum absolute derivative per image was a much larger 3.9 /input, indicating a large spread in these values. Attacks were generated against this network with a 50 % confidence margin in favor of an adversarial class. Again, based on a local-linearity assumption, the magnitudes of these attacks were measured as the mean absolute difference per pixel between the original and attacked images. The harmonic mean of the mean absolute distances of all such attacks against this network was found to be 0.0033 /input; according to Eq. (4.8),

the sum of Δy_i between the true and adversarially targeted classes should then be less than 50.7. The actual measured sum of Δy_i across the true and target classes averaged 26.1.

The change in network output was shown in Eq. (4.8) to be bounded proportionally to the gradient of the output with respect to each input element, as long as local network behavior was linear. Since this assumption seemed to hold for real networks, we theorized that limiting this gradient would therefore provide some adversarial resistance in these linear regions of the network by forcing additional non-linearities to compensate for the limitation. This is a form of Lipschitz continuity, as discussed in Section 4.1.4. From another point of view, limiting $|\partial y_i / \partial x_j|$ makes each training element a stable point for the network, enforcing a neighborhood of validity for each decision. The classification loss then enforces necessary non-linearities between these stable regions. As such, this work’s primary contribution is to explore the relation between limiting $E[|\partial y_i / \partial x_j|]$ and adversarial attacks. In the context of Eq. (4.1), this moves the focus from attempting to solve the outer minimization equation directly to instead limiting the effects of traveling in the allowed attack space S . We note that, particularly with the *Rectified Linear Unit* (ReLU) activation function, even a gradient of $\vec{0}$ does not guarantee a neighborhood of validity; see Section 4.2.7 concerning that issue.

For networks with 1,000 outputs, minimizing $|\partial y_i / \partial x_j|$ directly for all i is computationally prohibitive - each training image processed would require 1,000 additional gradient propagations. Instead, we use a regularizing loss which is stochastically defined with a scaling parameter ψ and a power factor z :

$$v_k \sim [1, 2, \dots, V],$$

$$L_{adv,z} = \frac{\psi}{KN} \sum_{k=1}^K \sum_{j=1}^N \left| \frac{\partial y_{v_k}}{\partial x_j} \right|^z. \quad (4.9)$$

Equation (4.9) therefore draws K random indices (without replacement) from the available output nodes and minimizes the derivative of each selected output with respect to all inputs. Backpropagation makes this an efficient computation regardless of the number of input elements. When $K = V$, $L_{adv,z}$ ceases to be stochastic. K and N are both included in the denominator such that the expected force per image relative to the classification loss is maintained regardless of the number of inputs or outputs. Sections 4.3.1.2 and 4.3.1.3 demonstrates the effects both of the relative strength of this loss, through changing ψ , and by varying its stochasticity, through changing K .

In addition to investigating the absolute value form of Eq. (4.9), using $|\partial y_{n_k}/\partial x_j|^z$, we investigated instead minimizing $L_{adv,z,q} = \left(\sum_{l=1}^N |\partial y_{n_k}/\partial x_l| \right)^q |\partial y_{n_k}/\partial x_j|^z$ for some values of z and q . We included these to illustrate that the proposed regularization technique is in fact a rich family of techniques based on approximations of which quantities are relevant for adversarial defense; a limited investigation of these metaparameters is found in Section 4.3.1.13. From this point forward we will use L_{adv} to refer to any of these, with default values of $z = 2$ and $q = 0$ unless otherwise specified.

We also considered the effects of creating a “dead zone” where gradients would not be punished, like a true Lipschitz constraint. For these experiments, instead of minimizing based on $|\partial y_{n_k}/\partial x_j|$ directly, L_{adv} would be minimized based on

$\max(|\partial y_{n_k}/\partial x_j| - \delta, 0)$. Results are found in Section 4.3.1.6.

It is also possible for n_k to be drawn from a non-uniform distribution. To test the merits of this, we considered distributions which yielded the correct label $\zeta\%$ of the time and were pulled from a random distribution (including the correct label) the rest of the time. Results with this technique are discussed in Sections 4.3.1.8 and 4.3.1.14.

Another variant of non-uniform distribution involved substituting the minimization of the true class’ gradient $\zeta\%$ of the time for minimizing the gradient $(\partial y_t - \max_{i \neq t} \partial y_i)/\partial x_j$, the difference between the true class and the maximum non-true class prediction. This regularization, which we label $L_{adv,tandem}$ because it aligns the gradients of two different classes in tandem, was chosen based on the “automobile” vs “truck” discussion from Section 4.2.1.2. While regularizing only one class at a time guarantees that the gradient for that class will approach zero, this provides an opportunity for a related class to dominate. Since the softmax operation assigns probabilities based on the difference between elements of its input, it was determined that it might be more effective to regularize the difference between those inputs (the NN’s output). Results for this technique are presented in Section 4.3.1.14.

4.2.3 Gradient Minimization as Weight Regularization

Exploring analogs to minimizing $|\partial y_i/\partial x_j|$ further, consider a single layer of an NN:

$$\begin{aligned}\vec{y} &= f(\mathbf{A}\vec{x} + \vec{b}), \\ \frac{\partial \vec{y}}{\partial \vec{x}} &= \frac{\partial f(\cdot)}{\partial \cdot} \mathbf{A}.\end{aligned}\tag{4.10}$$

Assuming all paths are active and we're using a ReLU network, then $\partial f(\cdot)/\partial \cdot = 1$. Since we would then use the element-wise absolute value or square of each element of $\mathbf{A}$ to devise our adversarial loss function L_{adv} , this is identical to L_1 or L_2 regularization for $q = 0, z \in \{1, 2\}$. While in a multi-layer setup, the proposed L_{adv} diverges from standard weight regularization, we considered it worthwhile to run experiments with weight regularization disabled on the convolutional weights in the network (keeping it enabled on biases within the network). These are explored in Section 4.3.1.10.

4.2.4 Half-Huber Rectified Linear Unit (HHRReLU)

A classical ReLU is continuous in value, but its derivative is discontinuous. Our proposal required optimizing the derivative of the activation functions used by the network, and as such we desired the first derivative to be continuous, allowing that the second derivative might be discontinuous. Related to the Huber function, we devised a new activation function, the *Half-Huber ReLU* (HHRReLU), defined as:

$$f(x) = \begin{cases} 0, & x < 0, \\ dx^2, & x < \frac{1}{2d} \\ x - \frac{1}{4d}, & \text{otherwise.} \end{cases}\tag{4.11}$$

While the parameter d describes the acceleration of the x^2 region, for timely results, we did not explore this parameter outside of $d = 1$. Nonetheless, we note that other values of d or other activation functions with a continuous first derivative might be explored further in the future. The impact of using this activation function instead of a traditional ReLU is explored in Section 4.3.1.7.

4.2.5 Output Zeroing

The softmax function, Eq. (4.5), is translation-invariant with respect to its inputs. We found that, in practice, allowing networks to rely on the invariance of the softmax function resulted in the flattening effects of L_{adv} persisting classification errors within the network. For instance, assume a two-class NN, which is producing output $y_0 = 1, y_1 = 2$ for some input. The adversarial resistance loss from Eq. (4.9) induces a certain amount of inertia about $y_0 = 1, y_1 = 2$, making it harder for the network to switch the ordering of these outputs. Adding an additional L_2 regularization term for the pre-softmax output of the network biases all of these terms toward 0, easing the classification task:

$$L_{out} = k_{out} \sum_i y_i^2. \quad (4.12)$$

If k_{out} is too large, then the network will never gain any confidence in its answers. Too small a k_{out} and the benefits will disappear. Therefore, similar to the guidelines on $|\partial y_i / \partial x_i|$ established by Eq. (4.8), we provide some guideline calculations regarding the balancing of these two forces while considering the maximum learnable confidence when training with a cross-entropy loss $L(\cdot)$ (using V as the number of possible classes):

$$\begin{aligned}
&\text{Assume } y_z = 0, \text{ where } z \neq t, \\
&L(\vec{y}) = -\log(s_t), s_i = \frac{e^{y_i}}{e^{y_i} + V - 1}, \\
&dL/dy_t = s_t - 1.
\end{aligned} \tag{4.13}$$

When the network’s predictions are an accurate distribution, and examples are uniformly distributed, y_i only has a $1/V$ chance of being a large value and needing to contest the classification loss. The rest of the time, it would only have a $1 - s_i$ chance of being large for a confusing example. Balancing the force of the cross-entropy loss, $L(\cdot)$, with L_{out} yields:

$$\begin{aligned}
\frac{L}{V} &= \frac{L_{out}}{V} + \frac{(V - 1)(1 - s_t)L_{out}}{V}, \\
\frac{1 - s_t}{V} &= \frac{k_{out}2y_t}{V} + \frac{(V - 1)(1 - s_t)k_{out}2y_t}{V}, \\
k_{out} &= \frac{1 - s_t}{2y_t(1 + (V - 1)(1 - s_t))}.
\end{aligned} \tag{4.14}$$

Equation (4.14), like Eq. (4.8), is not claimed to be an exact equation. However, it gives a guideline for reasonable parameter values. We tried $s_t = 0.8$ for all experiments, yielding $k_{out} = 0.01$ for our CIFAR-10 experiments, $k_{out} = 6e - 5$ for our ImageNet experiments, and $k_{out} = 1e - 3$ for our COCO experiments. Results of varying this parameter on the CIFAR-10 dataset may be found in Section 4.3.1.9.

4.2.6 Adaptive ψ

To ease comparisons between the meta-parameters necessary for our proposed technique to work, we investigated setting ψ from Eq. (4.9) automatically based on a targeted training loss. We performed experiments using an integrating controller:

$$\begin{aligned}\psi &= k_{\psi,0} e^{k_{\psi} \sum_b c \ln(L_{train,b}/L_{target})}, \\ c(d) &= clip(d, -\epsilon_{worse}, \epsilon_{better}),\end{aligned}\tag{4.15}$$

where b is the batch index, L_{target} is the targeted training loss, and $L_{train,b}$ is the training loss for batch b . While this approach still has one significant metaparameter, L_{target} , the meaning of its value is consistent regardless of other parameters. In all experiments, the regularization proposed in this work was capable of matching L_{target} . Note that the strength of ψ is based on the exponential of the integral as we found this to work significantly better across different scales of L_{target} , and throughout network training. The inner summation of Eq. (4.15) was always prevented from falling below zero, making $k_{\psi,0}$ the minimum strength.

Values used were typically $k_{\psi,0} = 220$, $k_{\psi} = 0.02$, $\epsilon_{better} = 1$, $\epsilon_{worse} = 0.01$, though early experiments used $k_{\psi,0} = 0.01$, $\epsilon_{worse} = 1$ (experiments before those discussed in Section 4.3.1.14).

4.2.7 Adversarial and Noisy Training

Section 4.2.2 mentioned that, even with gradients of $\vec{0}$ at all points in the training data, an activation function like ReLU guarantees no neighborhood for which the gradient will remain $\vec{0}$. That is, a loss “cliff” might be arbitrarily close to any

Figure 4.10: Training an NN on samples from a dataset (left) specify desired behavior at the data points, but does not describe behavior in between those data points, allowing the decision surface to take an arbitrary shape. Adding some noise (second from left) can somewhat improve the behavior between dataset samples, but is statistically unlikely to improve the worst behaviors. Adversarial training (second from right) deliberately attempts to improve the worst performing points, leading to a smoother decision surface. Adding a stochastic Lipschitz loss, as Eq. (4.9), further smooths behavior between data points.

training point. This is partly due to natural training overfitting, illustrated in Fig. 4.10. As such, we also investigated combining our method with either random Gaussian noise or adversarial training.

Adversarial training was implemented two ways. In the first way, denoted as L_2 , an L_2 distance ϵ was chosen and the adversary attempted to find the highest loss value within that ϵ -ball. The gradient of the classification cross-entropy loss was followed for 7 steps, each time being normalized to $\epsilon/7$ magnitude. In the second way, minimal adversarial training denoted as $L_{2,min}$, an L_2 distance ϵ was also chosen, but before taking each step, the network’s classification was evaluated. If the network correctly classified the example, then the gradient of classification loss was normalized to length $\psi_{7,n}\epsilon$ and followed, where $\psi_{k,n} = 2(k-n)/[k(k+1)]$ and $n \in [0, 1, \dots, k-1]$ is the index of the step being taken. In this formulation, the step sizes at subsequent steps yield progressively finer movements. If the network incorrectly classified the example, then the gradient was replaced with the negation of the current perturbation, normalized to size $\psi_{7,n}\epsilon$, and followed. That is, the $L_{2,min}$ method of adversarial training sought to train on adversarial examples near

the boundary at which the network would misclassify those examples.

Training configurations where batches were composed of half adversarial examples and half original examples from the dataset were also considered. In Tsipras *et al.* [63], this technique was called “Half-Half” training, and we keep that nomenclature.

Neither of these guarantee that a loss cliff would be corrected, but as seen in Section 4.3.1.15, they both somewhat alleviate the underlying problem. We also note that noise from batch normalization and data augmentation should help with this problem.

4.2.8 Human-in-the-Loop (HITL)

We explored using the adversarial examples generated from our networks to bootstrap even better adversarial resistance in a HITL pipeline with a *User Interface* (UI) as shown in Fig. 4.11. The UI took a trained network and used it to generate high-confidence adversarial examples. These examples were also generated from Algorithm 1, but instead with a high-confidence condition of $g(s, t) = \max_j(s_j; j \neq t) - \max_q(s_q; q \neq j) > 0.5$. That is, the adversarially generated, incorrect class had to be 50% more confident than the next-highest class. Users were then presented with three options: unchanged, unsure, and no longer the original class. When any button was pressed, the adversarial image was saved along with the original label and the annotation. We then tested re-training networks from scratch using the adversarial images annotated as “unchanged” as part of the training data. While this method of feedback was somewhat limited, we offer it as a proof-of-concept that our method of producing adversarial explanations could be used not only to inform the user about the reasoning behind an algorithmic decision, but also to

Figure 4.11: Prototype HITL interface. Whereas adversarial training simply generates adversarial examples and trains them to be recognized as the original class, our adversarial examples can be made of sufficient quality to merit human intervention as to whether or not the class of the image has changed.

feed HITL annotations back into improving the classifier.

4.2.9 Architectures and Datasets

This section contains details on the architectures used for the various datasets discussed in this document.

4.2.9.1 CIFAR-10

CIFAR-10 is a commonly-used dataset with 50,000 training images and 10,000 test images, consisting of 32×32 RGB images belonging to one of 10 classes [31]. Our CIFAR-10 experiments were based on a ResNet-44 [16], modified to be in pre-activation form [17] with each residual block’s output convolution weights initialized to zero as per [18]. Training used mini-batches of size 256 spread across 2 GPUs, for 128 images per GPU. We used standard data augmentation techniques for this task, reflecting the bordering 4 pixels and taking a random 32×32 crop during training. Training images were horizontally flipped 50 % of the time.

Stochastic Gradient Descent (SGD) was used to optimize weights with a momentum of 0.9 and L_2 weight regularization with a strength of $1e - 4$, starting at a learning rate of 0.02 which was linearly increased to 0.2 over the first 10 epochs. The learning rate was then stepped down to 0.02 and 0.002 at 170 and 195 epochs, respectively. Training was halted at 200 epochs. This entire setup was implemented in PyTorch [49], and resulted in a final top-1 validation accuracy of 92.2 % on a network without other changes.

4.2.9.2 ILSVRC 2012

While CIFAR-10 is small enough to iterate on quickly, success on CIFAR-10 does not guarantee the generality of a technique. Therefore, we also investigated training on the ILSVRC 2012 dataset, consisting of 1,281,167 training and 50,000 validation RGB images of varying size but significantly higher resolution than CIFAR-10, with objects belonging to one of 1,000 classes [56]. We trained a ResNet-18 [16] modified to be in pre-activation form [17] with each residual block’s output convolution weights initialized to zero as per [18]. To ease gradient descent with respect to the input as discussed in Section 4.2.2, we also replaced the initial max pooling operation with an average pooling operation. L_2 regularization was applied to weights and biases with a strength of $1e-4$. Training used mini-batches of size 192 spread across 3 GPUs, for 64 images per GPU. We used standard data augmentation techniques for this task, resizing the smallest edge of each image in $[256, 480]$ and taking a random 224×224 crop. Each crop was then given a 50 % chance of being horizontally flipped. We skipped the standard color augmentations. Rather than using the state-of-the-art method of computing validation accuracy, which

would have involved a 10-crop on the validation phase [16], we instead resized images such that the smallest edge was 256 pixels across and then took a crop of the central 224×224 pixels for validation. SGD was used to optimize weights with a momentum of 0.9 and L_2 weight regularization with a strength of $1e - 4$, starting at a learning rate of 0.03 and linearly increased to 0.3 over the first 10 epochs. The learning rate was then stepped down to 0.03 and 0.003 at 50 and 60 epochs, respectively. Training was halted at 65 epochs. This entire setup was implemented in PyTorch [49], and resulted in a final top-1 validation accuracy of 65.6% on a network without other changes.

4.2.9.3 Microsoft COCO

The *Common Objects in COntext* (COCO) dataset [37] was used as an additional proof-of-concept. The dataset consists of images containing scenes of multiple annotated objects from 80 different classes. To stick to classification problems for demonstrating our methods, we created a sub-dataset from COCO which consisted of taking the bounding box of each annotated object as a separate input example. During training, each object’s sub-image was resized such that the smallest edge was between 96 and 120 pixels long, selected a random 96×96 crop, and randomly performed a horizontal flip. During validation, each object was resized such that the smallest edge was 108 pixels long, and then the central crop of 96×96 pixels was selected. This scheme often led to images that overlap with the “person” classification, but was sufficient as a proof-of-concept.

The base network used for COCO annotations was the same ResNet-44 network from Section 4.2.9.1 as used CIFAR-10, but with filters of size 32, 64, and 128 (twice the standard width). Rather than the standard convolution for transforming input

data for the first residual block, we used a convolutional layer with a kernel size of 4 and a stride of 3, which reduced the image from 96×96 to 32×32 . SGD was used with a momentum of 0.9 and L_2 weight regularization with a strength of $1e - 4$, starting at a learning rate of 0.02 which was linearly increased to 0.2 over the first 10 epochs. The learning rate was then stepped down to 0.02 and 0.002 at 55 and 70 epochs, respectively. Training was halted at 80 epochs. This entire setup was implemented in PyTorch [49], and resulted in a final top-1 validation accuracy of 77.4% on a network without other changes. We note that this accuracy did not take class imbalances into account. For example, the greatest imbalance in our validation dataset was for the class “person,” which accounted for 31.4% of all objects in the dataset.

4.2.9.4 JSRT

The JSRT is described in Section 4.1.2. Our JSRT results were produced with networks similar to the ResNet-44 networks for CIFAR-10 from Section 4.2.9.1, using filters of size 64, 96, and 128, and with an initial convolution of kernel size 9 followed by an average pooling layer of size 8. Additionally, each input image was normalized such that it had zero-mean and unit variance; this was done due to wild variations in the different scans and scanned regions. Regions of 256×256 pixels were selected either A) with the central point being part of the nodule annotation for images containing nodules, or B) entirely randomly for images not containing nodules. Malignant and benign classifications were considered the same, under a new “nodule” category (making the problem binary). Training images were heavily augmented with shear angles from $[-30, 30]$ degrees, rotated from $[-45, 45]$ degrees, and scaled on a factor of $[0.61, 1.65]$. Additionally, random square regions of the

final training image between $[0, 64]$ pixels on each side were set to either black or white, to augment against the earlier per-image normalization.

4.2.10 Code Availability

A reference implementation of the techniques presented throughout this section applied to the CIFAR-10 dataset may be found at <https://github.com/wwoods/adversarial-explanations-cifar>.

4.3 Results

The majority of our experiments were conducted on CIFAR-10 due to it being a smaller dataset which was faster for iterating parameters and ideas. These are explored in Section 4.3.1. Experiments on ILSVRC 2012 were also conducted, and are covered in Section 4.3.2. Experiments on the COCO dataset are covered in Section 4.3.3.

4.3.1 CIFAR-10 Experiments

All CIFAR-10 experiments run with a ResNet-44 have been plotted in Fig. 4.12. At each level of accuracy (x axis), there may be several dots for ARA (y axis), indicating separate experiments with different levels of adversarial resistance. The variance of individual experiments is indicated in Section 4.3.1.3. The most immediate quality to be seen comparing best-in-class RMSEs across different accuracies is that accuracy may be sacrificed for additional adversarial resistance.

Large, colored dots indicate selected experiments. ● N1 indicates a traditional ResNet-44, not modified for increased resistance. ● N2 indicates a traditional ResNet-44 trained with adversarial training alone. ● N3 indicates a ResNet-44

with the modifications from Section 4.2, and was trained with both adversarial training and $L_{target} = 1.5$. ● N4 indicates a similar ResNet-44 to N3, but without the adversarial training. A comparison of adversarial attacks against the networks indicated by colored dots may be found in Fig. 4.13. The adversarial attacks against even the most robust of the networks were still very small perturbations, but did result in the visible accentuation of certain features, particularly the car door. An ideal network would produce genuine ambiguity at the adversarial example boundary. However, the right side of this figure demonstrates the proposed explanation techniques applied to the different networks. Each of the *g_{explain}* columns were produced with an RMSE of 0.15; N1’s modifications look like static, and while the example for the adversarially trained N2 is beginning to exhibit salient features, there is little difference between the “car” and “cat” columns. In contrast, N3 and N4 both demonstrate clear features, illustrating the utility of our stochastic Lipschitz regularization for producing networks capable of generating coherent explanations.

See Appendix A.1 for more examples of adversarial attacks against our CIFAR-10 networks. We emphasize that adversarial attacks against networks using our regularization term demonstrated increasingly salient features from the targeted class as the BTR ARA metric increased. These salient features were not forced from any term which necessitated a reconstruction of the input, as one would see with a GAN or Variational Autoencoder, indicating that the proposed technique alone was sufficient for producing classifiers which rely on salient features.

The most important CIFAR-10 experiments are detailed in Tables 4.1 to 4.4. Tables 4.1 and 4.2 address ablations of the techniques mentioned in Sections 4.2.2

Figure 4.12: Plot of all CIFAR-10 experiments run with a ResNet-44; accuracy on clean data versus attack ARA (as per Section 4.2.1). ● N1 indicates an unmodified ResNet-44, ● N2 indicates a network trained with only adversarial training, ● N3 indicates a network which combines adversarial training and our regularization, and ● N4 indicates an experiment using only our regularization. Dotted experiments are also used in Fig. 4.13.

to 4.2.5, showing that these techniques all work together to provide a reasonable level of adversarial resistance. Tables 4.3 and 4.4 illustrates the merits of Sections 4.2.6 to 4.2.8. The following sections share the same titles as the table entries for easy cross-referencing. Where applicable, rows in the tables have the same colored dots as Fig. 4.12, indicating the exact experiments conducted for those results.

4.3.1.1 Traditional ResNet-44

Our baseline ResNet-44 result is denoted in the first row of both Tables 4.1 and 4.3. The ● dot in the row means that it corresponds to the experiment with the same dot in Figs. 4.12 and 4.13. For this model, none of the adversarial explanations are sensible to a human observer, yet result in a significant change in the network’s

output (see Appendix A.1 for more examples).

4.3.1.2 Varied ψ from Eq. (4.9)

We sought to verify that increasing the strength of our proposed regularization would lead to an increase in adversarial resistance. The first section of Table 4.1 demonstrated that this was the case, with the classifier’s attack and BTR ARAs increasing monotonically with the strength of the regularization. Notable also is that, up to a certain level of $\psi = 4.0$, we were able to maintain the classifier’s accuracy on clean data while gaining additional adversarial resistance. After that, clean accuracy decreased as adversarial resistance increased. Therefore, ψ may be varied in accordance with which is more desirable: accuracy or adversarial resistance.

We also note that the training accuracy never reached 100 % for these experiments, indicating that a ResNet-44 does not have the ability to express a solution to the classification problem which both optimizes accuracy and has derivatives approximately equal to zero; this is explored further in Section 4.3.1.5.

4.3.1.3 Varying K from Eq. (4.9)

The stochastic formulation of Eq. (4.9) was expected to yield the same results as a non-stochastic formulation. To check the validity of this assumption, we tried different values of K , from 1 to 8. These experiments demonstrated that varying K had little effect. As such, all subsequent experiments used $K = 1$, which is more efficient to compute than any greater K as it only requires one additional backpropagation per training batch.

As these experiments were virtually identical, they also demonstrated that

training with the proposed regularization produced results with little variance.

4.3.1.4 Regularization methods

From the prior groups of experiments, it may be concluded that the adversarial resistance loss proposed in Eq. (4.9) provided a useful form of regularization. We wanted to test the combination of using Eq. (4.9) with other regularization techniques. Due to their promising results on CIFAR-10, we investigated Stochastic Depth [23] and ShakeDrop [74]. Our experiments with these showed that the proposed regularization performed best on its own, with additional regularizations resulting in lower accuracy on the clean data as well as when dealing with an adversary. We note that, as our networks’ training accuracy on the final epoch were never significantly higher than their validation accuracy, it is likely that a network would need significantly higher capacity before additional regularization would be useful.

4.3.1.5 Varied network depth/width

As mentioned in Sections 4.3.1.2 and 4.3.1.4, the training loss never approached zero when using the proposed regularization. We assumed this was due to a lack of model capacity. As such, we tried two different variations of the traditional ResNet, each having roughly $4\times$ as many parameters as the original network. ResNet-170 is four times as deep, and we also used a ResNet-44 with filters of size [32, 64, 128] for each of the three residual blocks, rather than the traditional [16, 32, 64]. Interestingly, the ResNet-170 did not appear to have an easier time optimizing the training accuracy. The double width ResNet-44, however, improved slightly in both accuracy and adversarial resistance. As will be seen in Section 4.3.1.12, we

found that another trick was required to fully utilize additional network capacity.

At this point, one can begin to see the difference between attack ARA and BTR ARA. Though the $\psi = 12,000$ used for these experiments resulted in an only minor increase in attack ARA to 0.0110 from 0.0084 for $\psi = 220$, the BTR ARA jumped from 0.0135 to 0.0347. This indicates that while the classifier was worse at accurately identifying an object in a tiny image, it was better at recognizing features of objects within the dataset. This manifested as clearer images - many of the adversarial perturbations for $\psi = 220$ still looked like randomized noise, whereas the adversarial perturbations for $\psi = 12000$ looked like deliberate changes to the objects in the image.

4.3.1.6 “Dead zone” from Section 4.2.2

A true Lipschitz-enforcing loss would not require any penalty on derivatives inside a region $[-\sigma, \sigma]$. These experiments demonstrated that increasing σ results in higher accuracy but less adversarial resistance, particularly in the BTR category. An additional experiment which used $\sigma = 0$ but set ψ such that the final accuracy is about the same demonstrates that there seemed to be no benefit from the addition of this metaparameter, and that leaving it at $\sigma = 0$ would seem to be the best choice. Part of this was due to the difficulty of setting σ : different values of ψ caused $|\partial y_{n_k} / \partial x_j|$ to be at different scales, and it was difficult to decide on a good value of σ . As we will argue in Section 4.3.1.13, there is a better way to implement a true Lipschitz constraint in Eq. (4.9), should it be beneficial.

4.3.1.7 Half-Huber ReLU from Section 4.2.4

The experiments before this point have used a traditional ReLU; here we used the HHReLU instead, which has a continuous derivative. At first glance the HHReLU was a modest improvement: improved accuracy and attack ARA, but substantially decreased BTR ARA. However, by training a ReLU network with an adjusted ψ such that the accuracy was about the same as the HHReLU version of the network, we saw that the HHReLU version of the network was better in both attack and BTR ARAs. Thus, the HHReLU is an important part of our regularization method, helping networks to learn better structural properties while retaining raw classification accuracy.

We note another potential explanation for a smaller BTR ARA in the first experiment with HHReLU: the HHReLU makes gradients that are easier to follow, and consequently also eases the task of generating successful attacks. That is, the BTR ARA metric for the ReLU network may be inflated as our adversary was unable to find low-perturbation attacks due to the increased difficulty of following gradients in a ReLU network.

4.3.1.8 Varied ζ from Section 4.2.2

These results demonstrated that $\zeta > 0$ was capable of inducing better results on the ARA metrics, but at a cost of some accuracy. We will revisit ζ in Section 4.3.1.14.

4.3.1.9 Output zeroing from Section 4.2.5

This segment of experiments contained two interesting outcomes. The first was that biasing the pre-softmax part of the network toward zero via $k_{out} = 0.01$ slightly improved accuracy and slightly worsened adversarial resistance. The second was

that doubling the number of epochs improved accuracy but worsened adversarial resistance. This was likely due to adversarial overfitting: the $E[|\partial y_{n_k}/\partial x_j|^2]$ of the training data was lower for the double epoch version, but the same quantity for the testing data was higher. This also made sense with respect to improved accuracy: as the loss from Eq. (4.9) approached zero, more training bandwidth would be freed up for the classification loss.

Interestingly, when considering an adaptive ψ value, we found that this parameter produced a more pronounced effect: see Section 4.3.1.16.

4.3.1.10 Weight regularization from Section 4.2.3

In Section 4.2.3 we proposed that Eq. (4.9) might be a replacement for the L_2 -loss traditionally imposed on weights as part of the NN training process. This experiment demonstrated worse performance without traditional L_2 regularization, indicating that architectures deeper than a single layer benefit from both regularization terms. However, in Section 4.3.2.1, we elaborate on the need for less L_2 -regularization when using our technique with large networks.

Figure 4.13: Input (left) and adversarial examples of different CIFAR-10 classes, generated for NNs with different levels of adversarial resistance using the different $g(\cdot)$ functions from Section 4.2.1, as indicated by the heading at the top of each column. The relative accuracy and attack RMSEs may be compared in Fig. 4.12; from top to bottom, these correspond to the experiments denoted by $\bullet$ N1, $\bullet$ N2, $\bullet$ N3, and $\bullet$ N4 dots. $g_{explain-}$ was generated by emphasizing the “cat” category.

Table 4.1: Effect of Modifications on CIFAR-10

Description	Acc.	Attack ARA	BTR ARA
● Traditional ResNet-44	92.2	0.0013	0.0014
Varying ψ from Eq. (4.9)			
$\psi = 0.55, K = 1$	92.5	0.0022	0.0026
$\psi = 4.0$	92.4	0.0039	0.0048
$\psi = 30.$	90.2	0.0059	0.0080
$\psi = 220$	84.5	0.0083	0.0135
Varying K from Eq. (4.9)			
$L_{2,adv}, \psi = 220, K = 1$	84.5	0.0083	0.0135
$K = 2$	85.0	0.0084	0.0134
$K = 4$	85.0	0.0084	0.0135
$K = 8$	84.8	0.0082	0.0133
Regularization methods			
No additional regularization, $\psi = 30$	90.2	0.0059	0.0080
Stochastic depth [23], $p_L = 0.8$	89.6	0.0059	0.0073
Stochastic depth, $p_L = 0.5$	86.8	0.0051	0.0060
ShakeDrop [74], $\alpha = 0, p_L = 0.8$	83.4	0.0043	0.0063
Varied network depth/width			
$\psi = 12,000$	56.5	0.0110	0.0347
ResNet-170	55.2	0.0104	0.0332
ResNet-44, double width	59.3	0.0118	0.0343
“Dead zone” from Section 4.2.2			
$\psi = 12,000, \sigma = 0$	56.5	0.0110	0.0347
$\sigma = 0.01$	66.8	0.0112	0.0288
$\sigma = 0.05$	79.4	0.0107	0.0197
$\sigma = 0, \psi = 220$ for similar accuracy	77.1	0.0102	0.0194
Half-Huber ReLU from Section 4.2.4			
$\psi = 12,000$, normal ReLU	56.5	0.0110	0.0347
$\psi = 12,000$, HHRReLU	78.0	0.0125	0.0261
No HHRReLU, but $\psi = 220$ for similar accuracy	77.1	0.0102	0.0194
Varied ζ from Section 4.2.2			
$\psi = 12,000$, HHRReLU, $\zeta = 0$	78.0	0.0125	0.0261
$\zeta = 0.2$	73.4	0.0128	0.0297
$\zeta = 0.5$	74.1	0.0118	0.0254
$\zeta = 0.8$	74.7	0.0107	0.0216
$\zeta = 0.99$	72.3	0.0083	0.0191
Output zeroing from Section 4.2.5			
Dead zone $\sigma = 1e - 2, \zeta = 0, k_{out} = 0$, HHRReLU, layer drop 0.8	78.5	0.0123	0.0236
Same, $k_{out} = 0.01$	78.9	0.0121	0.0222
Double epochs (400 total)	80.8	0.0118	0.0214

Effects of different modifications from Section 4.2 on the overall classification accuracy and adversarial resistance of an NN classifying CIFAR-10 images. “Acc.” is the accuracy on CIFAR-10’s test data after the final epoch. “Attack ARA” and “BTR ARA” are both as described in Section 4.2.1.

Table 4.2: Effect of Modifications on CIFAR-10, Part II

Description	Acc.	Attack ARA	BTR ARA
Weight regularization from Section 4.2.3			
Normal L2 weight regularization, HHReLU, $\zeta = 0.2$, dead zone $\sigma = 0.01$	78.9	0.0131	0.0248
Bias-only	77.7	0.0126	0.0239

See Table 4.1 for more information.

Table 4.3: Effect of Adaptiveness on CIFAR-10

Description	Acc.	Attack ARA	BTR ARA
<p>● Traditional ResNet-44</p> <p>Others have HReLU, normal weight regularization, no dead zone, $K = 1$, $\zeta = 0$, $k_{out} = 0.01$, and use $L_{adv,z=2}$ unless otherwise specified.</p>	92.2	0.0013	0.0014
Adaptive ψ from Section 4.2.6			
Fixed $\psi = 12,000$, final training classification loss 1.007	78.0	0.0125	0.0261
$L_{target} = 1.007$, initial $\psi = 0.01$, final $\psi = 14,000$	81.0	0.0124	0.0263
Fixed $\psi = 14,000$	77.8	0.0131	0.0277
Varied network depth/width			
Fixed $\psi = 14,000$			
ResNet-44	77.8	0.0131	0.0277
ResNet-170	79.9	0.0132	0.0267
ResNet-44, double width	81.9	0.0129	0.0257
Adaptive ψ, $L_{target} = 1.007$			
ResNet-44	81.0	0.0124	0.0263
ResNet-44, $k_{\psi,0} = 220, \epsilon_{worse} = 0.01$	80.1	0.0127	0.0274
ResNet-170	82.1	0.0128	0.0277
ResNet-44, double width	82.5	0.0129	0.0286
Different L_{adv} from Section 4.2.2 with $L_{target} = 1.007$			
$L_{adv,z=1} = L_{adv,z=0,q=1}$	81.2	0.0069	0.0104
$L_{adv,z=2}$	81.0	0.0124	0.0263
$L_{adv,z=3}$	78.4	0.0139	0.0317
$L_{adv,z=4}$	77.9	0.0143	0.0326
$L_{adv,z=5}$	76.6	0.0144	0.0325
$L_{adv,z=0,q=2}$	81.1	0.0082	0.0176
$L_{adv,z=1,q=1}$	80.8	0.0080	0.0172
$L_{adv,z=2,q=1}$	79.9	0.0132	0.0295
$L_{adv,z=2,q=2}$	79.7	0.0132	0.0303
$L_{adv,tandem}$ from Section 4.2.2 with $L_{target} = 1.007$			
ResNet-44, $\zeta = 0$	81.0	0.0124	0.0263
ResNet-44, $\zeta = 0.2$	78.7	0.0127	0.0277
ResNet-44, $\zeta = 0.2$, $L_{adv,tandem}$	80.2	0.0131	0.0302
ResNet-44, $\zeta = 0.2$, $L_{adv,tandem}$ with sum	80.7	0.0121	0.0247
● ResNet-44, $\zeta = 0.2$, $L_{adv,tandem}$, $L_{target} = 1.5$	68.7	0.0151	0.0423

Effects of adaptive ψ from Section 4.2.6 on network accuracy and adversarial resistance. Columns are the same as Table 4.1.

Table 4.4: Effect of Adaptiveness on CIFAR-10, Part II

Description	Acc.	Attack ARA	BTR ARA
Adversarial / noisy training from Section 4.2.7			
Madry <i>et al.</i> method, using L_2 adversarial training			
● L_2 , $\epsilon = 0.01$	87.4	0.0107	0.0153
L_2 , $\epsilon = 0.1$	88.6	0.0077	0.0205
Madry <i>et al.</i> method, but with $L_{2,min}$ training			
$L_{2,min}$, $\epsilon = 0.01$	88.1	0.0092	0.0111
$L_{2,min}$, $\epsilon = 0.1$	73.8	0.0121	0.0199
HHAT, $L_{2,min}$, $\epsilon = 0.1$	84.4	0.0126	0.0179
HHAT, $L_{2,min}$, $\epsilon = 0.1$, no HHReLU	84.3	0.0122	0.0180
$L_{2,min}$, $\epsilon = 0.25$	74.6	0.0151	0.0256
HHAT, $L_{2,min}$, $\epsilon = 0.25$	83.1	0.0132	0.0204
Equation (4.9) with adv. training, using $L_{target} = 1.5$ and $L_{adv,tandem}$			
$L_{2,min}$, $\epsilon = 0.1$	69.7	0.0195	0.0390
● HHAT, $L_{2,min}$, $\epsilon = 0.1$	68.4	0.0197	0.0450
HHAT, L_2 , $\epsilon = 0.1$	66.6	0.0164	0.0444
HHAT, $L_{2,min}$, $\epsilon = 0.01$	67.5	0.0163	0.0443
HHAT, $L_{2,min}$, $\epsilon = 0.1$, $L_{target} = 1.007$	79.0	0.0163	0.0306
Gaussian noise, using $L_{target} = 1.0$ and $L_{adv,tandem}$			
Gaussian +- 0.05	78.9	0.0135	0.0309
Gaussian +- 0.25	72.8	0.0129	0.0301
Combined adversarial training with output zeroing from Section 4.2.5			
● HHAT with output zeroing	68.4	0.0197	0.0450
HHAT without output zeroing	65.8	0.0197	0.0465
HITL from Section 4.2.8			
ResNet-44, double width, $L_{target} = 1.007$	82.5	0.0129	0.0286
HITL version, 448/730 annotations	82.7	0.0128	0.0285
HITL version, 1,331/2,731 annotations	82.6	0.0133	0.0296

See Table 4.3 for more information.

4.3.1.11 Adaptive ψ from Section 4.2.6

All previous experiments were executed with fixed values of ψ . Unfortunately, ψ is a somewhat obtuse parameter, as shown by most of the experiments thus far: it trades between accuracy and ARA in a consistent, but difficult to control manner. Furthermore, as shown by the double-epochs experiment from Section 4.3.1.9, fixing its value might be responsible for a type of overfitting.

This group of experiments tested whether targeting a specific training loss – a known quantity with a more consistent meaning than a specific ψ value – resulted in any beneficial behavior. We first took an experiment with known good parameters at a fixed ψ , and noted its classification loss on training data for the final epoch: $L = 1.007$. Setting L_{target} from Eq. (4.15) to this value, the proposed regularization with adaptive ψ resulted in a sizeable accuracy bump, from 78 % to 81 %, while retaining the same adversarial resistance. Using the same final ψ with a fixed network resulted in similar adversarial resistance, but lower accuracy.

4.3.1.12 Varied network depth/width

Revisiting the varied network sizes of Section 4.3.1.5, but using HHReLU, we compared fixed and adaptive ψ formulations. Unlike Section 4.3.1.5, the fixed ψ versions of these larger networks did demonstrate significant improvements to accuracy compared to the baseline ResNet-44, potentially due to HHReLU having increased the quality of the gradients when also contending with the adversarial loss of Eq. (4.9). However, adversarial resistance declined.

For the original ResNet-44 architecture, an adaptive setting of ψ was moderately better for accuracy and worse for ARA metrics. Recall that this group of experiments used $k_{\psi,0} = 0.01, \epsilon_{worse} = 1$ from Section 4.2.6, indicating a very small

ψ initially. We suspected that the change in performance may have been due to the “shock” of suddenly adding a new regularization term, and that starting with a larger $k_{\psi,0}$ might alleviate the problem. To test this, we added an additional experiment with $k_{\psi,0} = 220$ and $\epsilon_{worse} = 0.01$ from Section 4.2.6. This experiment retained much of the accuracy benefit of using an adaptive ψ , while recovering much of the lost adversarial robustness.

For deeper or wider networks, the adaptive ψ versions demonstrated improvements in both accuracy and adversarial ARA over the similar $k_{\psi,0} = 0.01$ experiment with a basic ResNet-44. We theorize that over-penalizing gradients early in network training stymies growth, whereas gradually adding the gradient penalty allows the network to first establish knowledge and subsequently refine it, allowing better usage of additional parameters. Under this adaptive scheme, the double width network outperformed the quadruple depth network by a narrow margin.

4.3.1.13 Different L_{adv} from Section 4.2.2 with $L_{target} = 1.007$

All previously mentioned experiments were conducted using $L_{adv,z=2}$ from Eq. (4.9). Though the original theory in Section 4.2.2 indicated $z = 1$ would be logical based on the behavior of derivatives in a linear network, actual networks are non-linear. In prior regularization work, as noted in Section 4.2.3, the L_2 method of weight regularization has also been more effective.

This segment of experiments reinforced that $z = 1$ performed unambiguously worse than $z = 2$. Interestingly, larger values of z led to increasing amounts of adversarial resistance, at a cost of accuracy. Note that due to L_{target} , all of the different z experiments had similar final training losses, and their testing losses were also all about the same, with a mean and standard deviation of 1.03 ± 0.02 .

Therefore, the decline in accuracy probably came from increased bias due to the interrelation of the proposed regularization method and the bias/variance trade-off.

We note that large values of z with an adaptive ψ approaches a true Lipschitz constraint at its limit, with a variable constraint on the derivative given by the interaction of ψ and z .

We also note that the improved attack ARA performance of $z > 2$ appears to have been unique to CIFAR-10; see Section 4.3.2. However, BTR ARA increases were consistent on the ILSVRC task as well (Section 4.3.2).

Experiments with q from Section 4.2.2 were also conducted, and showed similar trade-offs, replacing accuracy with increased ARA. We leave further exploration of this hyperparameter space to future work.

4.3.1.14 $L_{adv,tandem}$ from Section 4.2.2

The first two experiments in this section reprised the results from Section 4.3.1.8, though with an adaptive ψ . The third experiment demonstrated that much of the accuracy loss from $\zeta > 0$ could be recovered by smoothing the difference between the true label and the next-most-confident label (as opposed to smoothing the true label alone). Furthermore, this technique resulted in higher attack and BTR ARA metrics.

The fourth experiment demonstrates what happened when $L_{adv,tandem}$ was changed to use addition instead of subtraction. Accuracy improved further, but attack and BTR ARAs dropped significantly. We have no explanation for that particular phenomenon at this time.

The fifth experiment, with $L_{target} = 1.5$, was ran for parity with experiments in Paragraph 4.3.1.15.3. While accuracy slinked down from 80.2% to 68.7%, both

ARAs increased. The BTR ARA increased most significantly, from 0.0302 to 0.0423. We note that this is the highest BTR ARA of any of the experiments mentioned thus far, demonstrating that the proposed regularization continues to scale and provide benefits even into substantially decreased levels of accuracy.

4.3.1.15 Adversarial / noisy training from Section 4.2.7

This group of results is divided into four sub-groups: reproducing Madry *et al.*'s results, using Madry *et al.*'s technique with the $L_{2,min}$ adversary, combining our regularization with adversarial training, and combining our regularization with Gaussian noise.

4.3.1.15.1 Madry *et al.* method, using L_2 adversarial training These experiments used adversarial training as a standalone technique to provide resistance to adversarial examples. They were modeled off of the prior work of Madry *et al.* [41] and from personal communication with A. Madry. The ● dot indicates the N2 experiment labeled in Figs. 4.12 and 4.13, and was a reproduction of the best results from personal communication with A. Madry, albeit with a ResNet-44 instead of a ResNet-50. This experiment was therefore used for our comparison with the state-of-the-art.

We note that increasing ϵ from the recommended value of 0.01 for CIFAR-10 had little beneficial effect: accuracy surprisingly increased, but only from 87.4% to 88.6%, and attack ARA decreased from 0.0107 to 0.0077. Interestingly, BTR ARA did increase, from 0.0153 to 0.0205. We hypothesize this was due to BTR ARA measuring the classifier's ability to recognize features of object classes, without penalizing for related classes. Since the adversarial perturbations were substantially larger, more trucks could be made to look like automobiles, for instance, and the

Figure 4.14: The $L_{2,min}$ method of adversarial generation adds stability to adversarial training, but can suffer from degeneracy, where multiple training examples of the same class override a lone neighbor of a different class. Here, $L_{2,min}$ adversarial training would result in “Class A” being overshadowed by the two “Class B” instances. Since $L_{2,min}$ stops at the border of a misclassification, the class with fewer local members only successfully trains a very narrow region. As per Paragraph 4.3.1.15.2, this may be fixed with HHAT.

differences between these classes broke down even though the classifier improved at distinguishing them from the other classes such as bird, dog, etc. This phenomenon was discussed previously in Section 4.2.2.

4.3.1.15.2 Madry *et al.* method, but with $L_{2,min}$ training We next tested our proposed $L_{2,min}$ method of adversarial training, described in Section 4.2.7. The baseline measurements at $\epsilon = 0.01$ were comparable though slightly worse than those for L_2 adversarial training. The measurements at a higher $\epsilon = 0.1$ showed drastically decreased accuracy, but significantly higher attack ARA and comparable BTR ARA to the L_2 training with the same ϵ . The decreased accuracy was likely due to $L_{2,min}$ causing a sort of degeneracy, explained in Fig. 4.14. Regardless, when combined with Half-Half adversarial training, denoted as “HHAT” in the results, $L_{2,min}$ training recovered much of its lost accuracy while retaining the attack ARA and BTR ARA benefits.

In keeping with the other experiments of Table 4.3, many of the adversarial training experiments were conducted with HHReLU rather than ReLU. This was unlikely to affect the results, as HHReLU is very close to ReLU for approaches that minimize only loss and not its derivatives. One additional experiment was

run to ensure that this did not make a difference. As predicted, accuracy and ARA statistics are virtually identical for adversarial training with and without HHReLU.

To further test the progression of $L_{2,min}$ into larger values of ϵ , we conducted two further experiments with $\epsilon = 0.25$. These showed a surprising increase in accuracy for the non-HHAT version, and predictably increased ARA ratings. The HHAT version followed the expected course of decreased accuracy and increased robustness.

One aspect we wish to point out is that the best-case attack ARA from these experiments, which used only adversarial training, is on-par or slightly better than the best-case attack ARA using only our proposed regularization, when compared at the same level of accuracy on clean data. However, the BTR ARA was lower for any of the adversarial training experiments when compared to the BTR ARAs for our regularization. We therefore posit that adversarial training helps to stabilize the direction of steepest ascent for the loss function, while our proposed regularization stabilizes the entire loss surface. The definitions of the two techniques provide this distinction, and the empirical evidence appears to support it.

4.3.1.15.3 Equation (4.9) with adv. training, using $L_{target} = 1.5$ and $L_{adv,tandem}$ Combining what was learned from the previous sections, experiments were conducted with a combination of adversarial training and our proposed regularization. These yielded the best results, improving over the previous bests in both attack ARA and BTR ARA for given levels of accuracy. The HHAT variety of adversarial training best preserved the benefits to BTR ARA, so that is what we recommend moving forward. We also note the importance of both $L_{2,min}$ adversarial training and an appropriately high value of ϵ for attack ARA. While smaller

values of ϵ still yielded excellent BTR ARA, this was also seen in Section 4.3.1.14, and as such likely came almost entirely from our method.

4.3.1.15.4 Gaussian noise, using $L_{target} = 1.0$ and $L_{adv,tandem}$ The combination of our method with adversarial training was motivated by an attempt to find “loss cliffs” (Section 4.2.7). To ensure that the computational overhead of adversarial training added value to this cause beyond that of random noise, we also ran several experiments with Gaussian noise added on a per-component basis. In these experiments, each color value of each pixel received a perturbation independent of all other colors on all other pixels. Again, the BTR ARA was mostly preserved, but the attack ARA was substantially lower than when combined with adversarial training.

4.3.1.16 Combined adversarial training with output zeroing from Section 4.2.5

These two experiments demonstrated that the output zeroing method can have a more significant impact on accuracy without affecting attack ARA when using an adaptive ψ , but that the overall benefit was likely not worthwhile, particularly when considering the additional metaparameter. Nonetheless, most of the experiments in this paper were conducted with output zeroing as in Section 4.2.5.

4.3.1.17 HITL from Section 4.2.8

Two annotators annotated overly-saturated adversarial images generated from a double-width ResNet-44 via the UI in Fig. 4.11. These adversarial images were produced from the CIFAR-10 training data. One annotator annotated 730 and the other annotated 2,107 images. Of these annotations, 106 were of the same images,

and 72 of those were annotated with the same decision (changed, unchanged, or unsure), indicating that annotators agreed on 68% of the images. The double-width ResNet-44 architecture was then re-trained from scratch using a dataset consisting of the adversarial images annotated “unchanged” concatenated to the original CIFAR-10 dataset. The first experiment only used annotations from the first annotator, and had a total of 448 adversarial images with an “unchanged” target class added to the training dataset; the second experiment had a total of 1,331 images added. Example adversarial examples annotated as “changed” and “unchanged” may be found in Appendix A.1.

In the first experiment, adding a small number of annotations made virtually no difference. Note that CIFAR-10 has 50,000 training images, so we only increased the dataset’s size by 0.9%. In the second experiment, which added 2.7%, we saw an approximately 3% gain in both attack ARA and BTR ARA, with little change in accuracy. These gains could potentially be improved by stacking adversarial training with the technique. Given that the examples added to the dataset came from the dataset itself, a linear improvement in attack defense is very promising. A smaller dataset might find more benefit from adding training examples in this manner. A comparison of the adversarial examples from these networks may be found in Appendix A.1.

4.3.2 ImageNet Experiments

Our results on CIFAR-10 were encouraging, but not a guarantee that the technique would extend to larger networks with more complicated tasks. We trained several networks on ILSVRC 2012, but were somewhat limited in experiments due to each taking roughly a week to train on our hardware without adversarial training, and

Table 4.5: Effect of Modifications on ImageNet and COCO

Description	Acc.	Attack ARA	BTR ARA
Standard ResNet-18	65.6	0.0004	0.0013
Weight regularization from Section 4.2.3			
$\psi = 0.7e6$, dead zone 0.002, $k_{out} = 5e - 5^\dagger$	20.6	0.0013	0.0125
Use $1e-6$ instead of $1e-4$ L_2 regularization	50.4	0.0039	0.0184
Automatic ψ from Section 4.2.6			
$\psi = 4.8e6$	42.9	0.0041	0.0185
$L_{target} = 3.1$	45.5	0.0041	0.0182
$L_{target} = 3.1$ with HHAT, $\epsilon = 0.1$	42.2	0.0053	0.0282
$L_{target} = 4.0$, no HHAT	35.5	0.0041	0.0243
$L_{target} = 5.0$	23.7	0.0038	0.0388
$L_{target} = 5.0$, $L_{adv,z=2,q=1}$	22.0	0.0037	0.0538
$L_{target} = 5.0$, $L_{adv,z=5}$	19.7	0.0035	0.0526
COCO			
ResNet-44 from Section 4.2.9.3, baseline	77.3	0.0003	0.0008
With Eq. (4.9)	45.2	0.0029	0.0278
AT only, $L_{2,min}$, $\epsilon = 0.1$	60.8	0.0025	0.0089
Combined Eq. (4.9) + AT	45.2	0.0026	0.0228
Combined Eq. (4.9) + HHAT	45.2	0.0029	0.0250
Balanced classes, Eq. (4.9) only	25.8	0.0055	0.0335
Balanced classes, Eq. (4.9) + HHAT	26.9	0.0054	0.0325

Effects of different modifications from Section 4.2 on the overall classification accuracy and adversarial resistance of an NN classifying ImageNet and COCO images.

† This experiment was aborted after 18 epochs as it went unstable; the experiment below it had an accuracy of 43.0% at the same number of epochs.

several weeks with adversarial training. Nonetheless, we validated our ResNet-18 implementation with a top-1 accuracy of 65.6%, consistent with literature given we used 65 epochs rather than the usual 90. Results for the following sections are found in Table 4.5.

4.3.2.1 Weight regularization from Section 4.2.3

While using the proposed regularization to replace L_2 weight regularization as per Section 4.2.3 did not pan out for CIFAR-10, in ImageNet we found that it was vital to reduce the amount of L_2 weight regularization from $1e - 4$ to $1e - 6$ for the network to converge with our regularization. This was not required for the

standard ResNet-18. Relative to the task, ResNet-18 is likely underparameterized, and the classification loss, L_2 loss, and proposed L_{adv} loss were likely too at odds to find a good solution. Reducing the amount of L_2 loss made the problem tractable again.

4.3.2.2 Automatic ψ from Section 4.2.6

Given its efficacy on CIFAR-10, and that it was a better parameterization of the problem, we conducted all but one of the ILSVRC 2012 experiments with an adaptive ψ . All of these were conducted with HHReLU, an L_2 weight regularization of $1e-6$, no dead zone, $\sigma = 0$, $k_{out} = 5e - 5$, and $L_{adv,z=2}$ unless otherwise specified. The first two experiments in this group showed that, again, an adaptive ψ outperformed a fixed value of ψ . The subsequent experiments demonstrated the existence of the accuracy/attack RMSE trade-off, just like with CIFAR-10. However, with 1,000 classes, ImageNet’s attack ARA did not scale well as accuracy fell. The BTR ARA scaled well. See Appendix A.1 for examples of adversarial examples generated on these networks.

The proposed regularization method worked well on ILSVRC 2012, and was capable of generating convincing adversarial examples for many of the target classes. Other target classes, such as “n01484850 great white shark,” were clearly underspecified in the dataset, probably due to a lack of other classes with similar features. Many shark images are predominantly water, a property shared by few other ILSVRC 2012 classes. Similarly, the adversarial explanations resulted in the addition of water to the input more than any other feature.

4.3.3 COCO Experiments

Experiments were conducted on COCO to determine the efficacy of our regularization. As described in Section 4.2.9.3, the COCO dataset was a somewhat unique experiment as many of the images overlapped with other classes and the “person” class was over-represented as 31.4% of the total dataset.

Without any methods providing adversarial resistance, our COCO network scored 77.3% accuracy with an attack ARA rating of 0.0003. Note that the high accuracy of a naive classifier – 31.4% – swallows up much of the area that would otherwise increase the ARA ratings on this problem. Adversarial training added a good amount of attack ARA but only a little BTR ARA, consistent with previous experiments from Paragraph 4.3.1.15.2. However, using only our technique without any adversarial training resulted in the best overall statistics. We initially supposed this was due to the class imbalance, as $L_{2,min}$ adversarial training can suppress the correct label (Fig. 4.14). Unfortunately, experiments with balanced training on the COCO dataset, such that the classification loss for each label was divided by the percentage of that label, still resulted in little benefit from adversarial training. It is thus very possible that adversarial training did not make the COCO networks more robust as a consequence of the high number of overlapping objects in different frames — the actual distances between classes in the base problem were sufficiently small that adversarial training offered little benefit.

4.4 Combination with Network Bisection SRMs

Since I had previously dealt with network bisection SRMs, which are similar to attention networks, I was curious how these would combine with the adversarial robustness methods laid out this this chapter.

Figure 4.15: A side-by-side comparison of the $g_{explain,+}$ explanations from a network with the adversarial robustness modifications (top) versus with both adversarial robustness modification and the GIM SRM (bottom) shows that filtering out some of the intermediate activity of the network resulted in little change to the explanations.

For the first set of experiments, I measured the attack and BTR ARAs of the network bisection SRMs which utilized GIMs. The attack ARAs were identical to a traditional network (0.0013 across the board), and the BTR ARA was slightly elevated, up to 0.0018 from 0.0014.

For a second set of experiments, I trained a network with both the GIMs and the adversarial robustness techniques (but no adversarial training). Compared to similar networks without the GIMs, this resulted in worse accuracy (down to 65.1% from 68.7%), lower attack ARA (to 0.0148 from 0.0151), and marginally increased BTR ARA (to 0.0438 from 0.0423). Comparison explanations are shown in Fig. 4.15.

This chapter also illuminated the reason that network bisection SRMs did not improve a network’s adversarial resistance. Section 4.2.2 involves measuring the partial of the network’s output with respect to each input. Taking the square root of the expected value of these partials squared, a number may be generated which demonstrates how susceptible that network is to adversarial perturbations. Since the network bisection SRM resulted in fewer spatial points reaching the output, this number increased by a similar amount. That is, to make up for the removed pixels, decisions based on the remaining pixels were more volatile. Since the classification output then had a greater derivative with respect to these pixels,

attacks could then consist of smaller perturbations to the input regions passed to the classifier.

4.5 Oregon Health & Science University MEP-LINCS Report

Using the previously described techniques, I generated a report on cell features arising from environmental factors as part of an ongoing collaboration with OHSU on their *Microenvironment Perturbagen* (MEP) contribution to the *National Institutes of Health* (NIH)'s *Library of Integrated Network-based Cellular Signatures* (LINCS). In the MEP-LINCS project, cell cultures in-vitro were treated with a variety of ligands, resulting in different cellular growth patterns. The section of the project that we dealt with had 177,492 images, each treated with one of 57 different ligands. The amount of data is substantial, and as such, OHSU was interested in techniques which might help elucidate this data.

Figure 4.16 is a reproduced excerpt from the resulting 63-page report, showing the visual features which would be expected from other cell cultures had they been treated with the cxcl12|beta ligand rather than the ligands with which they were actually treated.

4.6 Discussion

We demonstrated a regularization technique based on the Lipschitz constraint, which significantly enhanced the ability of networks to resist adversarial examples. On ILSVRC 2012, the methods in this work increased the ARA by $2.4 \times$ over the previous state-of-the-art, while retaining the same level of accuracy on clean data and using a network one-third of the size of the previous state-of-the-art. More central to the tenets of this work, we demonstrated that the stability added by this

cxcl12|beta

Figure 4.16: Excerpt from the MEP-LINCS report delivered to OHSU. In the gray box are example images, unaltered, which were from the group treated with cxcl12|beta. Below that are three rows of images, each column being different views of the same image. Row 1 contains the adversarial example for cxcl12|beta using $g_{explain,+}$. Row 2 contains the unaltered base image, and the ligand with which it was originally treated. Row 3 shows the difference between rows 1 and 2.

technique allows for adversarial examples to be generated with very discernible features. These adversarial examples could then be used as non-linear explanation mechanisms, working with the network to produce more reliable explanations than prior work. Furthermore, we demonstrated that these new adversarial examples might be annotated and fed back into the training process to yield improved adversarial resistance with a feasible number of additional annotations.

No reliable, quantitative metrics for interpretability exist at present, though some attempts have been made [25]. Most current research into explainable AI using NNs has focused on achieving high-quality anecdotal results, without significant importance placed on the fidelity of the explanations [20]. The introduction

of BTR ARA as a metric which correlates well with the qualitative capacity of explanations may add a more reliable, quantitative measure to the field.

One of the desired aspects from Chapter 3 was not covered by this approach: the exposing of an intermediate step in the calculation. However, the stabilization techniques presented in this chapter could also be used to provide stability to the internal nodes of the network, and generating adversarial examples based on those internal nodes would then create the intermediate representations which were discussed in Chapter 3.

For future work, it would also be very interesting to apply the derived explanation techniques to control mechanisms for self-driving cars. Every major company has a branch working on this technology [8, 38], and ignoring the intrinsic requirements and the difficulties of those requirements would be an oversight. Specifically, self-driving cars require that the control network be more predictable, robust, and explainable than other applications. Self-driving cars must be predictable, or else they will stir distrust of the vehicles and induce negative legislation. If their networks are not robust, especially in the context of adversarial examples, then the cars would be easy to fool and make crash, an error potentially resulting in loss-of-life. Finally, for insurance purposes, self-driving vehicles need to have a level of explainability that is not present in other ML applications. Adversarial explanations partially address each of these requirements through increased reliability and focus on interpretability.

Conclusion

The goal of this dissertation was to provide intuitive, reliable explanations for the decisions made by NNs, with a quality exceeding that of the current state-of-the-art. The proposed mechanism for accomplishing this was utilization of the sensory domain, highlighting features which were deemed relevant in a domain intuitive for human operators. Thus, the title, “Sensory Relevance Models.”

This work was initially inspired by Rozell *et al.*’s LCA [55], as shown in Chapter 2. The experience of working with the LCA led to the realization that a good explanation requires some sparsity. However, when combined with deep learning, the LCA resulted in worse performance than inputting the raw pixels directly (Appendix A.2, Fig. 15). It was theorized that this was due to the unsupervised means of learning the LCA’s mapping, and a supervised extension was developed in Section 2.3. While the supervised approach of learning the LCA’s dictionary improved accuracy when combining an LCA with a classification problem, the benefits were not substantial enough to motivate further usage of the LCA in the context of sparsity for explanations. Therefore, it was important to find a method for achieving sparsity without using the LCA (Section 2.4).

Neural networks can be thought of as universal approximators, and there is no reason they also could not provide sparsity in a form similar to that seen by the LCA. This conclusion led to the proposition and development of network bisection SRMs (Chapter 3). By adding auxiliary losses to an intermediate layer, such as adjusting the activity of that layer (Section 3.3.1) or by keeping track of the

layer’s gradient information in a secondary NN (Section 3.3.2), the activity of that layer may be visualized in a sparse manner. These networks differed from prior work on annotating salient regions, attention models: they completely removed large portions of the input from consideration in the final classification decision, were trained via an auxiliary loss rather than the primary classification loss, and consisted of an entire feature dimension rather than just a mask. In the most sparse network produced, only 26 % of the input was used in a classifier which yielded 75 % classification accuracy (Section 3.4). The sparse activity from the network bisection SRM aligned well with salient features from the input images, shown in Fig. 3.4. However, when network bisection SRMs were tested in relation to adversarial examples, it was found that they were not more resistant than a traditional network, despite filtering out large portions of the input. Network bisection SRMs also disappointed in terms of yielding similar information to prior, heatmap-based methods of explanation, and in terms of producing uninformative hierarchies.

In light of these limitations, two flaws were identified that would need to be addressed before a more reliable explanation for NN decisions could be produced (Section 3.4). First, existing methods were all susceptible to adversarial examples, which called the neighborhood of validity of any such explanation into question. Second, all existing methods for explaining a decision on an image reduced to heatmap-based approaches, which do little to inform an inspector about the qualities considered important within the highlighted region. An algorithm that can draw a circle around a cat does not necessarily show knowledge that the circle contains a cat, but only indicates knowledge concerning the presence of a salient object.

Further research on addressing these flaws led to the development of adversarial explanations (Chapter 4 and Appendix A.1, [72]). Rather than producing a method of explanation in spite of adversarial examples, it made sense to work with them. I showed for the first time that adversarial examples may be used as a reliable, high-definition method of explaining the decisions of neural networks (Fig. 1.2). These images unambiguously support the reasoning behind a classification decision, and are undeniably effective at reproducing macro-scale structures leveraged by a classification NN (Fig. 4.9). One common criticism of classification NNs is that they yield a bag-of-features approach; the explanations within this work have contributed evidence showing a bag-of-features classifier is not mandated by the convolutional approach, though it is one possible outcome (see Appendix of Appendix A.1). Furthermore, I introduced network modifications and training strategies which resulted in networks $2.4 \times$ more resistant to adversarial attacks than the previous state-of-the-art (Section 4.2). The modifications included a stochastic Lipschitz constraint, partially replacing the L_2 weight regularization with the stochastic Lipschitz constraint, a Half-Huber Rectified Linear Unit activation function, an adaptive controller for controlling the trade-off between accuracy and explanation quality, and an alternative method of adversarial training, $L_{2,min}$ (Section 4.2). These modifications were leveraged to show that the resulting networks not only produce high-quality explanations, but demonstrate generative qualities as well, enabling the transformation of a base image into images of different classes (Appendix A.1). I introduced a metric which correlated well with the quality of the resulting explanations, the BTR ARA (Section 4.2.1.2). These networks were also tested for HITL compatibility, and showed some promise

at generating networks which were even more resistant to adversarial attacks (Section 4.3.1.17).

In all, this work has demonstrated significant advancements in the field of machine learning, particularly for explaining the decisions of NNs. I hope that this work provides a basis for future work in the realms of both adversarial resistance and explainable machine learning, making algorithms more reliable for industry fields where accountability matters, such as biomedical or autonomous vehicles.

References

- [1] I. Arel, D. C. Rose, and T. P. Karnowski, “Deep machine learning-a new frontier in artificial intelligence research,” *IEEE computational intelligence magazine*, vol. 5, no. 4, pp. 13–18, 2010.
- [2] A. Athalye and I. Sutskever, “Synthesizing robust adversarial examples,” *arXiv preprint arXiv:1707.07397*, 2017.
- [3] A. Athalye, N. Carlini, and D. Wagner, “Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples,” *arXiv e-prints arXiv:1802.00420*, 2018.
- [4] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, “Network dissection: Quantifying interpretability of deep visual representations,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6541–6549, , 2017.
- [5] D. Bau, J. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba, “GAN dissection: Visualizing and understanding generative adversarial networks,” *arXiv e-prints arXiv:1811.10597*, vol. abs/1811.10597, 2018.
- [6] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J.-H. Jacobsen, “Invertible residual networks,” *arXiv e-prints*, p. arXiv:1811.00995, 2018.

- [7] C. H. Bennett, D. Chabi, T. Cabaret, B. Jousset, V. Derycke, D. Querlioz, and J.-O. Klein, “Supervised learning with organic memristor devices and prospects for neural crossbar arrays,” in *Nanoscale Architectures (NANOARCH), 2015 IEEE/ACM International Symposium on*, pp. 181–186. IEEE, 2015.
- [8] J. S. Brodsky, “Autonomous vehicle regulation: how an uncertain legal landscape may hit the brakes on self-driving cars,” *Berkeley Tech. LJ*, vol. 31, p. 851, 2016.
- [9] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, “On Evaluating Adversarial Robustness,” *arXiv e-prints*, p. arXiv:1902.06705, 2019.
- [10] N. Carlini and D. A. Wagner, “Towards evaluating the robustness of neural networks,” *CoRR*, vol. abs/1608.04644, 2016.
- [11] J. M. Cohen, E. Rosenfeld, and J. Zico Kolter, “Certified adversarial robustness via randomized smoothing,” *arXiv e-prints*, p. arXiv:1902.02918, 2019.
- [12] Y. Cui, Z. Chen, S. Wei, S. Wang, T. Liu, and G. Hu, “Attention-over-attention neural networks for reading comprehension,” *CoRR*, vol. abs/1607.04423, 2016.
- [13] A. M. Elkahky, Y. Song, and X. He, “A multi-view deep learning approach for cross domain user modeling in recommendation systems,” in *Proceedings of the 24th International Conference on World Wide Web*, pp. 278–288, , 2015.

- [14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, pp. 2672–2680, , 2014.
- [15] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2014.
- [16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, , 2016.
- [17] —, “Identity mappings in deep residual networks,” in *European conference on computer vision*, pp. 630–645. Springer, 2016.
- [18] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, “Bag of Tricks for Image Classification with Convolutional Neural Networks,” *arXiv e-prints*, p. arXiv:1812.01187, 2018.
- [19] W. He, J. Wei, X. Chen, N. Carlini, and D. Song, “Adversarial example defenses: Ensembles of weak defenses are not strong,” *arXiv preprint arXiv:1706.04701*, 2017.
- [20] A. Holzinger, C. Biemann, C. S. Pattichis, and D. B. Kell, “What do we need to build explainable AI systems for the medical domain?” *ArXiv e-prints*, 2017.
- [21] S. Hong, T. You, S. Kwak, and B. Han, “Online tracking by learning discriminative saliency map with convolutional neural network,” in *International Conference on Machine Learning*, pp. 597–606, , 2015.

- [22] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” *CoRR*, vol. abs/1709.01507, 2017.
- [23] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger, “Deep Networks with Stochastic Depth,” *arXiv e-prints*, p. arXiv:1603.09382, 2016.
- [24] D. L. Hudson and M. E. Cohen, *Neural networks and artificial intelligence for biomedical engineering*. Wiley Online Library, 2000.
- [25] D. Huk Park, L. A. Hendricks, Z. Akata, A. Rohrbach, B. Schiele, T. Darrell, and M. Rohrbach, “Attentive Explanations: Justifying Decisions and Pointing to the Evidence (Extended Abstract),” *ArXiv e-prints*, 2017.
- [26] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *arXiv e-prints*, p. arXiv:1502.03167, 2015.
- [27] L. Itti and C. Koch, “Computational modelling of visual attention,” *Nature reviews neuroscience*, vol. 2, no. 3, pp. 194–203, 2001.
- [28] S. Jetley, N. A. Lord, N. Lee, and P. H. Torr, “Learn to pay attention,” *ArXiv e-prints arXiv:1804.02391*, 2018.
- [29] J. K. Kim, P. Knag, T. Chen, and Z. Zhang, “A 640M pixel/s 3.65mW Sparse Event-Driven Neuromorphic Object Recognition Processor with On-Chip Learning C50 C51,” in *VLSI Circuits (VLSI Circuits), 2015 Symposium on*, pp. 50–51, no. 4. IEEE, 2015.
- [30] P. Knag, J. K. Kim, T. Chen, and Z. Zhang, “A Sparse Coding Neural Network ASIC With On-Chip Learning for Feature Extraction and

- Encoding,” *IEEE Journal of Solid-State Circuits*, vol. 50, no. 4, pp. 1070–1079, 2015.
- [31] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” 2009.
- [32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., pp. 1097–1105. Curran Associates, Inc., 2012.
- [33] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, J. Wang, Z. Zhang, Z. Ren, A. Yuille, S. Huang, Y. Zhao, Y. Zhao, Z. Han, J. Long, Y. Berdibekov, T. Akiba, S. Tokui, and M. Abe, “Adversarial attacks and defences competition,” *arXiv e-prints*, p. arXiv:1804.00097, 2018.
- [34] W. Landecker, “Interpretable machine learning and sparse coding for computer vision,” Ph.D. dissertation, Portland State University, 2014.
- [35] K. Li, Z. Wu, K. Peng, J. Ernst, and Y. Fu, “Tell me where to look: Guided attention inference network,” *arXiv e-prints arXiv:1802.10171*, 2018.
- [36] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, “Defense against adversarial attacks using high-level representation guided denoiser,” *arXiv e-prints*, p. arXiv:1712.02976, 2017.
- [37] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: common objects in context,” *CoRR*, vol. abs/1405.0312, 2014.

- [38] T. Litman, *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute, 2017.
- [39] W. Luo, Y. Li, R. Urtasun, and R. Zemel, “Understanding the effective receptive field in deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, pp. 4898–4906, , 2016.
- [40] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, “Traffic flow prediction with big data: a deep learning approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.
- [41] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” *arXiv e-prints arXiv:1706.06083*, 2017.
- [42] T. Masquelier and S. J. Thorpe, “Unsupervised learning of visual features through spike timing dependent plasticity,” *PLoS Comput Biol*, vol. 3, no. 2, p. e31, 2007.
- [43] B. M. McLaren, T. van Gog, C. Ganoë, M. Karabinos, and D. Yaron, “The efficiency of worked examples compared to erroneous examples, tutored problem solving, and problem solving in computer-based learning environments,” *Computers in Human Behavior*, vol. 55, pp. 87–99, 2016.
- [44] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, “Interpretable machine learning: definitions, methods, and applications,” *arXiv e-prints*, p. arXiv:1901.04592, 2019.
- [45] M. Narayanan, E. Chen, J. He, B. Kim, S. Gershman, and F. Doshi-Velez, “How do Humans Understand Explanations from Machine Learning Systems?”

- An Evaluation of the Human-Interpretability of Explanation,” *arXiv e-prints*, p. arXiv:1802.00682, 2018.
- [46] E. Oja, “Simplified neuron model as a principal component analyzer,” *Journal of mathematical biology*, vol. 15, no. 3, pp. 267–273, 1982.
 - [47] B. A. Olshausen and D. J. Field, “Emergence of simple-cell receptive field properties by learning a sparse code for natural images,” *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.
 - [48] B. A. Olshausen and C. J. Rozell, “Sparse codes from memristor grids,” *Nature Publishing Group*, vol. 12, no. 8, pp. 722–723, 2017.
 - [49] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” in *NIPS-W*, , 2017.
 - [50] M. Payvand and L. Theogarajan, “Exploiting local connectivity of CMOL architecture for highly parallel orientation selective neuromorphic chips,” *Proceedings of the 2015 IEEE/ACM International Symposium on Nanoscale Architectures, NANOARCH 2015*, pp. 187–192, 2015.
 - [51] K. Pei, Y. Cao, J. Yang, and S. Jana, “DeepXplore: Automated whitebox testing of deep learning systems,” *arXiv preprint arXiv:1705.06640*, 2017.
 - [52] T. Pfeil, T. C. Potjans, S. Schrader, W. Potjans, J. Schemmel, M. Diesmann, and K. Meier, “Is a 4-Bit Synaptic Weight Resolution Enough? Constraints on Enabling Spike-Timing Dependent Plasticity in Neuromorphic Hardware,” *Frontiers in Neuroscience*, vol. 6, no. JULY, pp. 1–19, 2012.

- [53] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat, “Immunity to device variations in a spiking neural network with memristive nanodevices,” *IEEE Transactions on Nanotechnology*, vol. 12, no. 3, pp. 288–295, 2013.
- [54] M. T. Ribeiro, S. Singh, and C. Guestrin, “Why should I trust you?: Explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144. ACM, 2016.
- [55] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen, “Sparse coding via thresholding and local competition in neural circuits,” *Neural computation*, vol. 20, no. 10, pp. 2526–2563, 2008.
- [56] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [57] K. Scharnhorst, W. Woods, C. Teuscher, A. Stieg, and J. Gimzewski, “Non-temporal logic performance of an atomic switch network,” in *2017 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pp. 133–138. IEEE, 2017.
- [58] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra *et al.*, “Grad-CAM: Visual explanations from deep networks via gradient-based localization.” in *ICCV*, pp. 618–626, , 2017.

- [59] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep inside convolutional networks: Visualising image classification models and saliency maps,” *arXiv preprint arXiv:1312.6034*, 2013.
- [60] J. Sweller, “Cognitive load during problem solving: Effects on learning,” *Cognitive science*, vol. 12, no. 2, pp. 257–285, 1988.
- [61] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *arXiv e-prints arXiv:1312.6199*, 2013.
- [62] M. Taha, W. Woods, and C. Teuscher, “Approximate in-memory Hamming distance calculation with a memristive associative memory,” in *Proceedings of the 2016 IEEE/ACM International Symposium on Nanoscale Architectures, NANOARCH 2016*, , 2016.
- [63] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, “Robustness may be at odds with accuracy,” *arXiv e-prints*, p. arXiv:1805.12152, 2018.
- [64] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017.
- [65] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, “Residual Attention Network for Image Classification,” *arXiv e-prints*, p. arXiv:1704.06904, 2017.
- [66] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, “Evaluating the robustness of neural networks: An extreme value

- theory approach,” in *International Conference on Learning Representations*, , 2018.
- [67] W. Woods, J. Bürger, and C. Teuscher, “On the influence of synaptic weight states in a locally competitive algorithm for memristive hardware,” in *Proceedings of the 2014 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH 2014)*, pp. 19–24. IEEE Press, 2014.
- [68] W. Woods and C. Teuscher, “Approximate vector matrix multiplication implementations for neuromorphic applications using memristive crossbars,” in *Proceedings of the 2017 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pp. 103–108. IEEE, 2017.
- [69] W. Woods and C. Teuscher, “Fast and accurate sparse coding of visual stimuli with a simple, ultralow-energy spiking architecture,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2018.
- [70] W. Woods, “The Design of a Simple, Spiking Sparse Coding Algorithm for Memristive Hardware,” *Portland State University Dissertations and Theses*, vol. Paper 2721, 2016.
- [71] W. Woods, J. Bürger, and C. Teuscher, “Synaptic Weight States in a Locally Competitive Algorithm for Neuromorphic Memristive Hardware,” *IEEE Transactions on Nanotechnology*, vol. 14, no. 6, pp. 945–953, 2015.
- [72] W. Woods, J. Chen, and C. Teuscher, “Adversarial Explanations for Understanding Image Classification Decisions and Improved Neural Network Robustness,” *arXiv e-prints*, p. arXiv:1906.02896, 2019.

- [73] W. Woods, M. M. A. Taha, S. J. Dat Tran, J. Burger, and C. Teuscher, “Memristor panic - A survey of different device models in crossbar architectures,” in *Proceedings of the 2015 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH 2015)*, pp. 106–111. IEEE, 2015.
- [74] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise, “ShakeDrop Regularization for Deep Residual Learning,” *arXiv e-prints*, p. arXiv:1802.02375, 2018.
- [75] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European conference on computer vision*, pp. 818–833. Springer, 2014.
- [76] J. Zylberberg, J. T. Murphy, and M. R. DeWeese, “A Sparse Coding Model with Synaptically Local Plasticity and Spiking Neurons Can Account for the Diverse Shapes of V1 Simple Cell Receptive Fields,” *PLoS Computational Biology*, vol. 7, no. 10, p. e1002250, 2011.

A

Appendix Selected Papers

This appendix contains additional information on supplemental papers which demonstrate the full scope of this dissertation, in addition to the primary narrative established in Chapters 1 to 4. This appendix describes each paper and the reason for its inclusion.

On PDXScholar, copies of these papers may be found under the “Additional Files” heading. The citations given also include sufficient information to find copies of these works.

A.1 Adversarial Explanations for Understanding Image Classification Decisions and Improved Neural Network Robustness

This paper is under revision with Nature Machine Intelligence as of Aug. 2019 (a pre-print is available on ArXiv [72]), and is included on PDXScholar as “Woods2019adversarial.pdf”. It details adversarial explanations, both in terms of the problems with existing explanation methods and my proposed solutions, as described in Chapter 4 of this work. This paper also contains an appendix with many more visual examples of adversarial explanations.

My contributions to this work were the original idea, the algorithms, the experiment design, the majority of experiment execution, interpreting of the results, and the majority of the document writing. J. H. Chen was responsible for the LIME and Grad-CAM integrations, annotating images in the HITL section, and providing text on related work for the HITL experiments. C. Teuscher was responsible

for advising on scope, editing, and providing funding.

A.2 Fast and Accurate Sparse Coding of Visual Stimuli with a Simple, Ultra-Low-Energy Spiking Architecture

This paper was published by IEEE Transactions on Neural Networks and Learning Systems in 2018 [69], and is included on PDXScholar as “Woods2018sslca.pdf”. The work details a hardware implementation I designed for a sparse coding algorithm based on Rozell *et al.*’s LCA [55], and demonstrates the efficacy of combining unsupervised sparse coding with deep learning solutions, both in the context of an optimal sparse coding algorithm and the hardware approximation. The lackluster results of the deep learning solution combined with the sparse coding architecture, coupled with the exploration of the supervised LCA (Section 2.3) provided significant motivation to explore alternate forms of sparsity, leading to the network bisection SRM (Chapter 3).

My contributions to this work were the idea to place the state capacitors directly on the memristive crossbar, a key insight in providing low-power inhibition; the supporting mathematics; the experiment design; the experiment execution; the interpretation of results; and the writing of the document. C. Teuscher provided the idea to use memristors for the LCA crossbar, editing, and funding.