Reliable Classification Explanations via Adversarial Attacks on Robust Networks

Walt Woods, Jack Chen, and Christof Teuscher
Department of Electrical and Computer Engineering
Portland State University, Portland, OR, USA
{wwoods, chenjac, teuscher}@pdx.edu

Abstract

Neural Networks (NNs) have been found vulnerable to a class of imperceptible attacks, called adversarial examples, which arbitrarily alter the output of the network. These attacks have called the validity of NNs into question, particularly on sensitive problems such as medical imaging or fraud detection. We further argue that the fields of explainable AI and *Human-In-The-Loop* (HITL) algorithms are impacted by adversarial attacks, as attacks result in perturbations outside of the salient regions highlighted by state-of-the-art techniques such as LIME or Grad-CAM. This work accomplishes three things which greatly reduce the impact of adversarial examples, and pave the way for future HITL workflows: we propose a novel regularization technique inspired by the Lipschitz constraint which greatly improves an NN's resistance to adversarial examples; we propose a collection of novel network and training changes to complement the proposed regularization technique, including a Half-Huber activation function and an integrator-based controller for regularization strength; and we demonstrate that networks trained with this technique may be deliberately attacked to generate rich explanations. Our techniques led to networks more robust than the previous state of the art: using the *Accuracy-Robustness Area* (ARA), our most robust ImageNet classification network scored $42.2\,\%$ top-1 accuracy on unmodified images and demonstrated an attack ARA of 0.0053, an ARA $2.4\times$ greater than the previous state-of-the-art at the same level of accuracy on clean data, achieved with a network one-third the size. A far-reaching benefit of this technique is its ability to intuitively demonstrate decision boundaries to a human observer, allowing for improved debugging of NN decisions, and providing a means for improving the underlying model.

Reliable Classification Explanations via Adversarial Attacks on Robust Networks

I. Motivation

Adversarial attacks/examples are inputs to *Machine Learning* (ML) algorithms which are perceptually similar to examples that yield good performance from the algorithm, but produce drastically different output [1], [2]. While these have been shown to exist on a variety of ML algorithms [3], we focus on the sub-field of *Neural Networks* (NNs). Work by LabSix robustly extrapolated these adversarial examples to a real-world, 3D-printed turtle, which a state-of-the-art NN almost always classified as a rifle [4]. They could have just as easily 3D-printed a rifle that would be classified as a turtle. This NN loophole poses a security risk at worst, and has left researchers scratching their heads at best. Attempts to identify and remedy the problem of adversarial examples generally agree on the existence of manifolds shared by the dataset which are incongruous to perceptual manifolds [5], [6]. Work on adversarial examples has not generally focused on creating situations in which adversarial examples are perceptually similar to the targeted class; only Tsipras *et al.* [7] touched on this as a curiosity associated with adversarial training.

Adversarial examples aside, industry fields wanting to harness the explosion of ML techniques are concerned about the lack of accountability and explainability within the field [8], [9]. Biomedical papers report systems which surpass human experts, but have difficulty proving the added insight of their techniques beyond statistical correlations [8], [10]. Methods have been proposed to generate heatmaps demonstrating regions of input salient to an NN's output. However, heatmaps do not communicate information beyond a rough silhouette, nor do they align with the changes needed to create an adversarial example, as we show in Fig. 1 with the Grad-CAM algorithm [11], and in Section II-B with both Grad-CAM and LIME [12]. These methods are linearizations of a highly non-linear network, and do not capture all relevant details. A more compelling explanation method must exploit the inherent non-linearities of a network.

We propose and demonstrate classification networks for the ILSVRC 2012 challenge with $2.4\times$ improved robustness to adversarial attacks compared to the state of the art, as shown in Fig. 3, without sacrificing additional accuracy. Improved robustness was achieved via the methods described in Section III. Adversarial attacks against these networks are not only larger in magnitude, but also contain salient features and demonstrate the network's decision boundary in a meaningful way for human operators, as shown in Figs. 1 and 2. We explore the trade-offs of the proposed techniques, including the ability to train networks to be either more accurate or robust to attacks, in Section IV. Due to the increased adversarial distance, our NNs may also be used to synthesize new examples for a human-in-the-loop pipeline to explain and improve a classifier, which we demonstrate in Sections III-H and IV-A17. Together, the methodology outlined in this work demonstrates the viability of producing cogent explanations via adversarial attacks on robust networks.

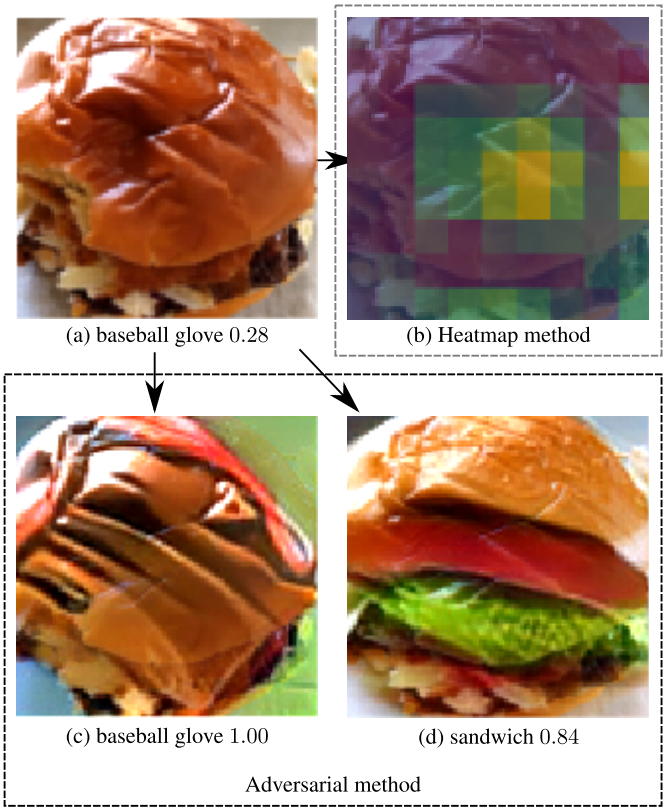


Fig. 1: State-of-the-art methods of explaining an NN's decision based on some input (a) rely on the generation of some heatmap (b), generated here via Grad-CAM. In cases where the object's shape is non-discriminatory, this provides little insight into the decision. Adversarial examples produce much richer 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.

II. Related Work

Three branches of ML inquiry led to this work: adversarial attacks, explanation techniques, and *Human-In-The-Loop*

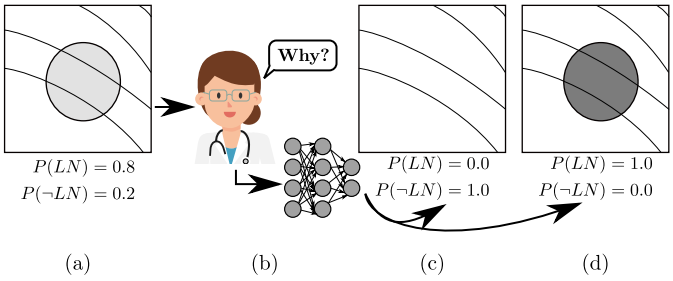


Fig. 2: While adversarial examples are considered a nuisance by most, they have the potential to provide reliable explanations with the same richness of information as the original input. For example, when an NN trained at finding lung nodules in radiographs needs investigation (a, b), an attack may 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.

(HITL) methods. A mathematical branch, Lipschitz continuity, was also important in developing this work. These four related areas are discussed in Sections II-A to II-D.

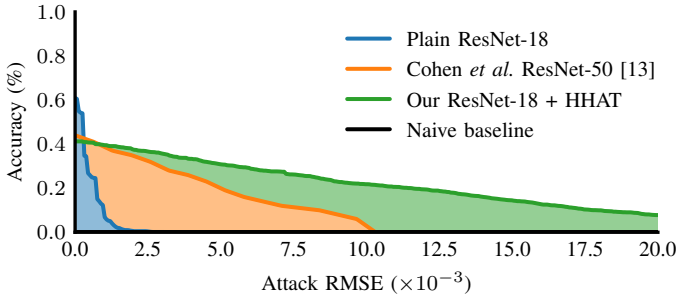
A. *Adversarial Attacks*

Adversarial attacks, or adversarial examples, were first documented by Szegedy *et al.* [1], who showed that an NN's output may be arbitrarily changed through imperceptible changes to the input. Initial criticism that these digitally-induced deviations might be a pathological problem were put to rest by a group from LabSix in 2018 [4]. The LabSix group fabricated a real-world object which was adversarially misclassified at a variety of angles and scales, demonstrating that the problem of adversarial examples had real-world consequences and deserved further study. Many reports have posited that adversarial examples are a natural extension of the internal flexibility of NNs [1], [2], [4], [7], [15]. 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. Adversarial attacks exploit the incongruence between these learned spaces and the visual space containing the NN's input.

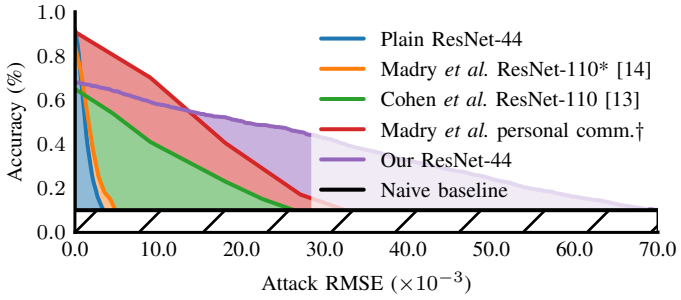
The struggle between adversarial attacks and methods of resisting them is perhaps best illustrated by the saddle point formulation proposed by Madry *et al.* [14] (notation adapted to be consistent throughout the current paper):

$$\min \mathbb{E}_{(x,t)\sim D}\left[\max_{\delta\in S} L(\theta, x+\delta, t)\right]. \tag{1}$$

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



(a)



(b)

Fig. 3: Compared to the state-of-the-art, our method achieved higher accuracy while being tolerant of greater adversarial perturbations on both ImageNet (Fig. 3a) and CIFAR-10 (Fig. 3b). 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 II-A 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 [14].
† This curve came from personal communication with A. Madry on a standard ResNet-50; see Section II-A for details.

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. (1). Carlini *et al.* [15] compared several different methods of generating attacks, including Goodfellow *et al.*'s *Fast Gradient Sign Method* (FGSM) [2] and their own *Projected Gradient Descent* (PGD) [15]. Carlini *et al.* [15] 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.* [7] 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. (1). To our knowledge, state-of-the-art methods of resisting adversarial attacks currently revolve around either adversarial training [7],

[14], [16] or randomized smoothing [13].

Madry *et al.* [14] 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 [14]. That group later expanded their theories on adversarial training in work by Tsipras *et al.* [7], 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.* [13] 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 [14] or our work in Section III, 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 [13].

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 [17], [18], 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 [5].

We note that a roundup of best-practices for ensuring that new defenses are effective was recently authored by Carlini *et al.* [19]. 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 III) or make gradients less obfuscated (Section III-D), sticking to a gradient-based PGD-variant attack seemed sufficient. Attack transferability was not investigated as the proposed models have 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. 3. 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.* [13] presented this plot for L_2 attacks against both ImageNet and CIFAR-10 results, and Madry *et al.* [14] 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. 3, 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 IV-A15).

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. 3b, 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 III-A1, 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.

B. *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 [11], [12], [20]–[23].

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 [20], [24] or looking at the receptive fields to which internal nodes respond [22], [25]–[27]. However, these techniques often produce very noisy images that are difficult to interpret.

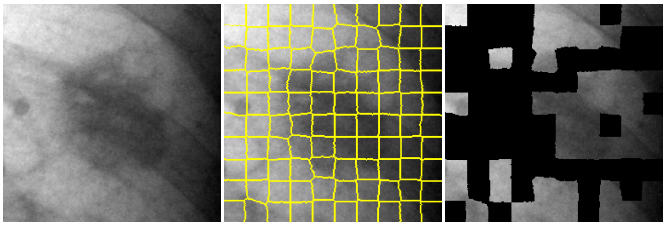
Works such as Ribeiro *et al.*'s LIME [12] or Selvaraju *et al.*'s Grad-CAM [11] proposed improvements over raw saliency maps. These works highlighted the difficulty of identifying the reason behind incorrect classifications, and proposed their own solutions to the problem. LIME presented a generalized 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 [12]. On the other hand, the Grad-CAM algorithm harnesses backpropagation directly to derive an expression for localizing the most salient regions [11]. 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 [11], [12], 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.

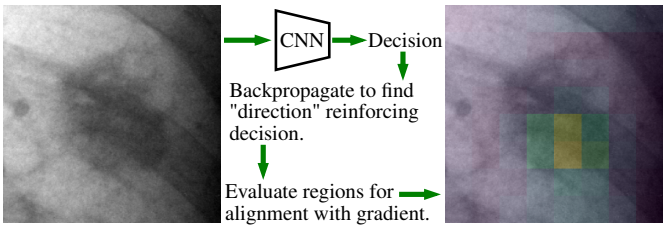
Consider the lung nodule dataset published by the *Japanese Society of Radiological Technology* (JSRT). This dataset consists of 247 chest X-rays, 154 of which have a single, annotated nodule, each of which show up as dark, solitary shadows. Chest X-rays are notoriously difficult to read, and so the 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 $2.5\,\mathrm{cm}$, or $143\,\mathrm{px}$, 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 and Fig. 5 respectively. Considering the output of LIME for this classifier, Fig. 4, 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. 5). 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. 6, there is no correlation in overlap between the adversarial perturbations and the salient region demarked by Grad-CAM. As such, it is hard to 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 [28], [29]. 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.



(a) (b) (c)

Fig. 4: 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?



(a) (b) (c)

Fig. 5: Selvaraju *et al.*'s Grad-CAM [11] 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?

Also noteworthy is Bau *et al.*'s "Network Dissection" work [22]. 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 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 [30]. Nonetheless, that is a generative, rather than explanatory, function.

C. *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" [31] 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

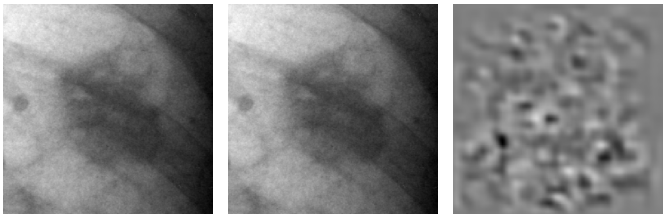


Fig. 6: An adversarial example corresponding to the same network and input as Figs. 4 and 5. Here, the resulting adversarial input (a) is the sum of the original input (b) and a crafted noise term scaled by a small coefficient (c). The result is an imperceptible change in input leading to a completely 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.

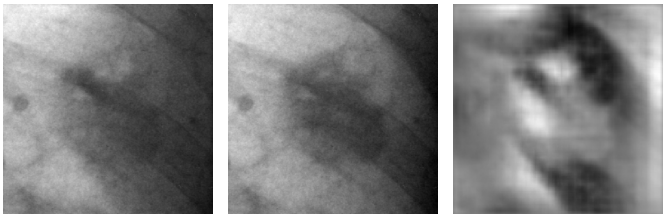


Fig. 7: In contrast to Fig. 6, our preliminary technique of increasing adversarial distance concentrates the perturbations needed to change the network's output. The RMSE (c) between the original input (a) and the adversarial attack (b) is also much greater for a smaller change in network output.

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 [31]. Additionally, for $10{,}000$ classification training examples, they only used segmentation masks for $1{,}464$ of the examples [31], illustrating that even partial annotations provided some benefit.

Ribeiro *et al.* [12] 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 III-H.

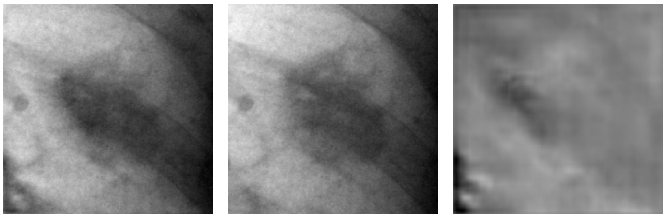
D. *Lipschitz Continuity*

Prior work has been performed on Lipschitz continuity, particularly with analyzing the stability of NNs [32], [33]. 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.

III. METHODS

Adversarial examples in state-of-the-art networks, as in Fig. 6, 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.* [7] 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. 2, 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. 7 and 8; these were implemented using the techniques described throughout this section.

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



0.75 nodule | = 0.66 nodule | + noise $\times 1.7e-02$ (RMSE 0.0113)

(a) (b) (c)

Fig. 8: Example of emphasizing a lung nodule via the same method of adversarial attack. Same network as Fig. 7.

A. *Adversarial Attack Generation and Evaluation*

Adversarial attacks were conducted with two separate goals within this paper: Section III-A1 contains the methodology for adversarial attacks aimed at reducing the classification accuracy of a network, and Section III-A2 contains the methodology for adversarial attacks aimed at producing classification explanations.

*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.* [15]. 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.* [15], the algorithm presented will not begin a magnitude refinement before the target classification error is reached. The threshold at 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 [7], [13], and denotes the boundary at which top-1 accuracy would decrease:

$$g_{adv}(s,t) = \begin{cases} 1 & \text{if } s_t - max_j(s_j; j \neq t) < 0, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

This was the $g(s,t)$ used to produce Fig. 3. 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 non-zero. This list was then evaluated for accuracy at different levels of RMSE, as seen in Fig. 3, 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

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.

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

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. 6, and against a regular CIFAR-10 ResNet-44 network in Fig. 12.

*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. 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. Thus, g_{btr} (where V is the number of classes in the prediction) is defined as:

Fig. 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 donus 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.

$$g_{btr}(s,t) = \begin{cases} 1 & \text{if } s_t < \frac{1}{V}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

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:

$$g_{explain+}(\delta;\rho) = \begin{cases} 1 & \text{if } ||\delta|| > \rho, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

We note that $g_{explain-}$ is also possible, by modifying Algorithm 1 to maximize the selected class loss rather than minimizing it. These techniques are demonstrated in Fig. 9.

*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 IV-A.

B. *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:

$$s(y) = \frac{e^y}{\sum_{i=1}^{V} e^{y_i}}. \tag{5}$$

In the $1{,}000$-class ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC 2012) challenge, there are $1{,}000$ classes [34]. 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:

$$E_i\left[|\Delta y_i|\right] \approx E_i\left[\left|\sum_{j=1}^{N} \Delta x_j \frac{\partial y_i}{\partial x_j}\right|\right], \tag{6}$$

$$\lessapprox \sum_{j=1}^{N} E_{i,j}\left[\left|\Delta x_j \frac{\partial y_i}{\partial x_j}\right|\right], \tag{7}$$

These quantities are neither independent nor equivalent, but we will simplify them as such:

$$E\left[|\Delta y_i|\right] \lessapprox N E\left[|\Delta x_j|\right] E\left[\left|\frac{\partial y_i}{\partial x_j}\right|\right]. \tag{8}$$

Equation (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. (8) becomes less accurate.

To see how effective the guideline given by Eq. (8) was in practice, we built a ResNet-18 and trained it on ILSVRC 2012 training data, detailed in Section III-I2. 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 $0.051\,/\text{input}$. Interestingly, the mean of the maximum absolute derivative per image was a much larger $3.9\,/\text{input}$, 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 $0.0033\,/\mathrm{input}$; according to Eq. (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. (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 II-D. 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. (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 III-G 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:

$$\begin{aligned} v_k &\sim [1, 2, ..., V], \\ L_{adv,z} &= \frac{\psi}{KN} \sum_{k=1}^{K} \sum_{j=1}^{N} \left| \frac{\partial y_{v_k}}{\partial x_j} \right|^z . \end{aligned} \tag{9}$$

Equation (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 IV-A2 and IV-A3 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. (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 IV-A13. 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 IV-A6.

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 IV-A8 and IV-A14.

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 III-A2. 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 IV-A14.

C. *Gradient Minimization as Weight Regularization*

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

$$\begin{aligned} \vec{y} &= f(\mathbf{A}\vec{x} + \vec{b}), \\ \frac{\partial \vec{y}}{\partial \vec{x}} &= \frac{\partial f(\cdot)}{\partial \cdot} \mathbf{A}. \end{aligned} \tag{10}$$

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 IV-A10.

D. *Half-Huber Rectified Linear Unit (HHReLU)*

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* (HHReLU), defined as:

$$f(x) = \begin{cases} 0, & x < 0, \\ dx^2, & x < \frac{1}{2d} \\ x - \frac{1}{4d}, & \text{otherwise.} \end{cases} \tag{11}$$

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 IV-A7.

E. *Output Zeroing*

The softmax function, Eq. (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. (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:

$$L_{out} = k_{out} \sum_i y_i^2. \tag{12}$$

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. (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):

$$\begin{aligned} &\text{Assume } y_z = 0, \text{ where } z \neq t, \\ &L(\vec{y}) = -log(s_t), s_i = \frac{e^{y_i}}{e^{y_i} + V - 1}, \\ &dL/dy_t = s_t - 1. \end{aligned} \tag{13}$$

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:

$$\begin{aligned} \frac{L}{V} &= \frac{L_{out}}{V} + \frac{(V-1)(1-s_t)L_{out}}{V}, \\ \frac{1-s_t}{V} &= \frac{k_{out} 2 y_t}{V} + \frac{(V-1)(1-s_t)k_{out} 2 y_t}{V}, \\ k_{out} &= \frac{1-s_t}{2y_t(1 + (V-1)(1-s_t))}. \end{aligned} \tag{14}$$

Equation (14), like Eq. (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 IV-A9.

F. *Adaptive ψ*

To ease comparisons between the meta-parameters necessary for our proposed technique to work, we investigated setting ψ from Eq. (9) automatically based on a targeted training loss. We performed experiments using an integrating controller:

$$\begin{aligned} \psi &= k_{\psi,0} e^{k_\psi \sum_b c \ln(L_{train,b}/L_{target})}, \\ c(d) &= clip(d, -\epsilon_{worse}, \epsilon_{better}), \end{aligned} \tag{15}$$

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. (15) was always prevented from falling below zero, making $k_{\psi,0}$ the minimum strength.

Values used were typically $k_{\psi,0} = 220$, $k_\psi = \ln(0.02)$, $\epsilon_{better} = 1$, $\epsilon_{worse} = 0.01$, though early experiments used $k_{\psi,0} = 0.01, \epsilon_{worse} = 1$ (experiments before those discussed in Section IV-A14).

G. *Adversarial and Noisy Training*

Section III-B 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 training point. 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 $n \in [0, 1, ..., k-1]$ 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.* [7], 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 IV-A15, they both somewhat alleviate the underlying problem. We also note that noise from batch normalization and data augmentation should help with this problem.

H. 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. 10. 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 retraining 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 feed HITL annotations back into improving the classifier.

I. Architectures and Datasets

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

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 [35]. Our CIFAR-10 experiments were based on a ResNet-44 [36], modified to be in pre-activation form [37] with each residual block's output convolution weights initialized to zero as per [38]. 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

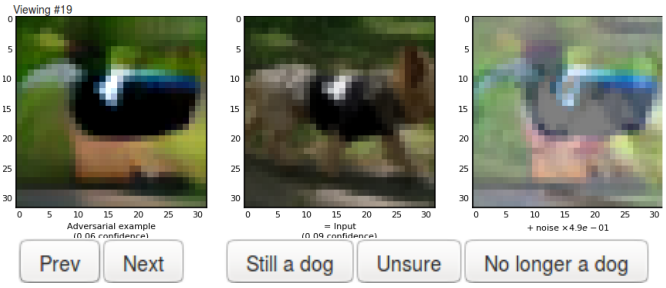


Fig. 10: Prototype HITL interface. Whereas adversarial training simply generates adversarial examples and trains them to be recognized as the original class, our adversarial examples can be made of sufficient quality to merit human intervention as to whether or not the class of the image has changed.

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 [39], and resulted in a final top-1 validation accuracy of $92.2\,\%$ on a network without other changes.

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 [34]. We trained a ResNet-18 [36] modified to be in pre-activation form [37] with each residual block's output convolution weights initialized to zero as per [38]. To ease gradient descent with respect to the input as discussed in Section III-B, 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 [36], 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 [39], and resulted in a final top-1 validation accuracy of $65.6\,\%$ on a network without other changes.

*3) **Microsoft COCO**:* The *Common Objects in COntext* (COCO) dataset [40] 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 III-I1 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 [39], 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) **JSRT**:* The JSRT is described in Section II-B. Our JSRT results were produced with networks similar to the ResNet-44 networks for CIFAR-10 from Section III-I1, 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.

J. 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.

IV. 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 IV-A. Experiments on ILSVRC 2012 were also conducted, and are covered in Section IV-B. Experiments on the COCO dataset are covered in Section IV-C.

A. CIFAR-10 Experiments

All CIFAR-10 experiments run with a ResNet-44 have been plotted in Fig. 11. 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 IV-A3. 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 III, 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. 12. 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 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.

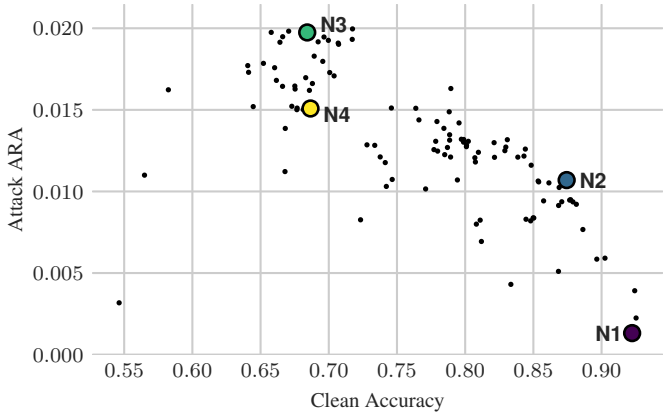


Fig. 11: Plot of all CIFAR-10 experiments run with a ResNet-44; accuracy on clean data versus attack ARA (as per Section III-A). ● N1 indicates an unmodified ResNet-44, ● N2 indicates a network trained with only adversarial training, ● N3 indicates a network which combines adversarial training and our regularization, and ● N4 indicates an experiment using only our regularization. Dotted experiments are also used in Fig. 12.

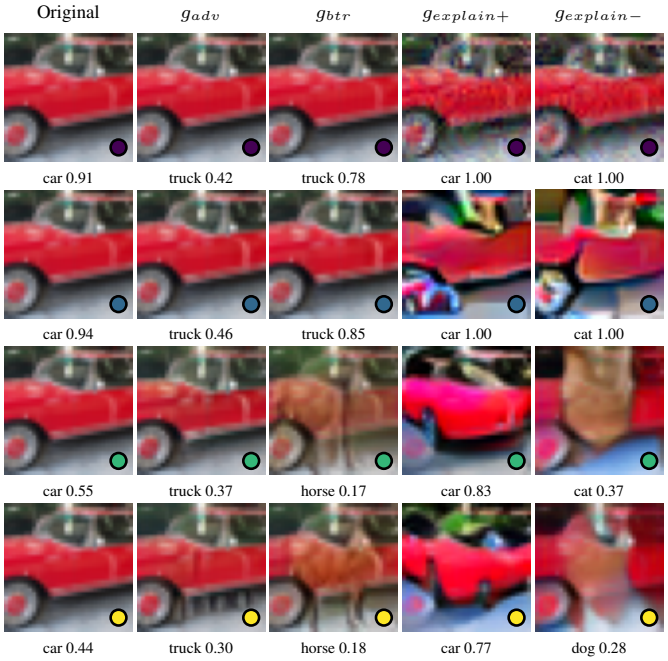


Fig. 12: Input (left) and adversarial examples of different CIFAR-10 classes, generated for NNs with different levels of adversarial resistance using the different $g(\cdot)$ functions from Section III-A, as indicated by the heading at the top of each column. The relative accuracy and attack RMSEs may be compared in Fig. 11; 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.

The most important CIFAR-10 experiments are detailed in Tables I and II. Table I addresses ablations of the techniques mentioned in Sections III-B to III-E, showing that these techniques all work together to provide a reasonable level of adversarial resistance. Table II illustrates the merits of Sections III-F to III-H. 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. 11, indicating the exact experiments conducted for those results.

*1) **Traditional ResNet-44**:* Our baseline ResNet-44 result is denoted in the first row of both Tables I and II. The ● dot in the row means that it corresponds to the experiment with the same dot in Figs. 11 and 12. 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 for more examples).

*2) **Varied ψ from Eq. (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 I 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 IV-A5.

*3) **Varying K from Eq. (9)**:* The stochastic formulation of Eq. (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) **Regularization methods**:* From the prior groups of experiments, it may be concluded that the adversarial resistance loss proposed in Eq. (9) provided a useful form of regularization. We wanted to test the combination of using Eq. (9) with other regularization techniques. Due to their promising results on CIFAR-10, we investigated Stochastic Depth [41] and ShakeDrop [42]. 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.

*5) **Varied network depth/width**:* As mentioned in Sections IV-A2 and IV-A4, the training loss never approached zero when using the proposed regularization. We assumed this was

TABLE I: 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. (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. (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 [41], $p_L = 0.8$	89.6	0.0059	0.0073
Stochastic depth, $p_L = 0.5$	86.8	0.0051	0.0060
ShakeDrop [42], $\alpha = 0, p_L = 0.8$	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 III-B			
$\psi = 12,000, \sigma = 0$	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 III-D			
$\psi = 12,000$, normal ReLU	56.5	0.0110	0.0347
$\psi = 12,000$, HHReLU	78.0	0.0125	0.0261
No HHReLU, but $\psi = 220$ for similar accuracy	77.1	0.0102	0.0194
Varied ζ from Section III-B			
$\psi = 12,000$, HHReLU, $\zeta = 0$	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 III-E			
Dead zone $\sigma = 1e-2$, $\zeta = 0$, $k_{out} = 0$, HHReLU, 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
Weight regularization from Section III-C			
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

Effects of different modifications from Section III 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 III-A.

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 IV-A12, 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.

*6) **"Dead zone" from Section III-B**:* 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 IV-A13, there is a better way to implement a true Lipschitz constraint in Eq. (9), should it be beneficial.

*7) **Half-Huber ReLU from Section III-D**:* 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.

*8) **Varied ζ from Section III-B**:* 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 IV-A14.

*9) **Output zeroing from Section III-E**:* 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. (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 IV-A16.

10) ***Weight regularization from Section III-C****:* In Section III-C we proposed that Eq. (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 IV-B1, we elaborate on the need for less L_2-regularization when using our technique with large networks.

11) ***Adaptive ψ from Section III-F****:* 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 IV-A9, 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. (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.

12) ***Varied network depth/width****:* Revisiting the varied network sizes of Section IV-A5, but using HHReLU, we compared fixed and adaptive ψ formulations. Unlike Section IV-A5, 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. (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 III-F, 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 III-F. This experiment retained much of the accuracy benefit of using an adaptive ψ, while recovering much of the lost adversarial robustness.

TABLE II: Effect of Adaptiveness on CIFAR-10

Description	Acc.	Attack ARA	BTR ARA
● Traditional ResNet-44	92.2	0.0013	0.0014
Others have HHReLU, normal weight regularization, no dead zone, $K = 1$, $\zeta = 0$, $k_{out} = 0.01$, and use $L_{adv,z=2}$ unless otherwise specified.			
Adaptive ψ from Section III-F			
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 III-B 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 III-B 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
Adversarial / noisy training from Section III-G			
Madry *et al.* 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 *et al.* 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 (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 III-E			
● HHAT with output zeroing	68.4	0.0197	0.0450
HHAT without output zeroing	65.8	0.0197	0.0465
HITL from Section III-H			
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

Effects of adaptive ψ from Section III-F on network accuracy and adversarial resistance. Columns are the same as Table I.

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.

13) ***Different L_{adv} from Section III-B with $L_{target} = 1.007$****:* All previously mentioned experiments were conducted using $L_{adv,z=2}$ from Eq. (9). Though the original theory in Section III-B 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 III-C, 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 IV-B. However, BTR ARA increases were consistent on the ILSVRC task as well (Section IV-B).

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

14) ***$L_{adv,tandem}$ from Section III-B****:* The first two experiments in this section reprised the results from Section IV-A8, 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 IV-A15c. 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.

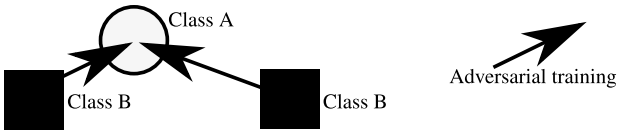


Fig. 13: The $L_{2,min}$ method of adversarial generation adds stability to adversarial training, but can suffer from degeneracy, where multiple training examples of the same class override a lone neighbor of a different class. Here, $L_{2,min}$ adversarial training would result in "Class A" being overshadowed 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 IV-A15b, this may be fixed with HHAT.

15) ***Adversarial / noisy training from Section III-G****:* 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.

a) 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.* [14] and from personal communication with A. Madry. The ● dot indicates the N2 experiment labeled in Figs. 11 and 12, 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 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 III-B.

b) 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 III-G. 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. 13. 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 II, 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.

c) Equation (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 IV-A14, and as such likely came almost entirely from our method.

d) 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 III-G). 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.

16) ***Combined adversarial training with output zeroing from Section III-E****:* 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 III-E.

17) ***HITL from Section III-H****:* Two annotators annotated overly-saturated adversarial images generated from a double-width ResNet-44 via the UI in Fig. 10. 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 B.

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 B.

B. *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 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 III.

1) ***Weight regularization from Section III-C****:* While using the proposed regularization to replace L_2 weight regularization as per Section III-C 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.

2) ***Automatic ψ from Section III-F****:* 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

TABLE III: 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 III-C			
$\psi = 0.7e6$, dead zone 0.002, $k_{out} = 5e-5$†	20.6	0.0013	0.0125
Use 1e-6 instead of 1e-4 L2 regularization	50.4	0.0039	0.0184
Automatic ψ from Section III-F			
$\psi = 4.8e6$	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 III-I3, baseline	77.3	0.0003	0.0008
With Eq. (9)	45.2	0.0029	0.0278
AT only, $L_{2,min}$, $\epsilon = 0.1$	60.8	0.0025	0.0089
Combined Eq. (9) + AT	45.2	0.0026	0.0228
Combined Eq. (9) + HHAT	45.2	0.0029	0.0250
Balanced classes, Eq. (9) only	25.8	0.0055	0.0335
Balanced classes, Eq. (9) + HHAT	26.9	0.0054	0.0325

Effects of different modifications from Section III 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.

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 D 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.

C. *COCO Experiments*

Experiments were conducted on COCO to determine the efficacy of our regularization. As described in Section III-I3, 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 IV-A15b. 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. 13). 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.

V. Conclusion

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 × 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 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. We 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.

Acknowledgements

We would like to thank A. Madry (from [7], [14]) and J. Cohen (from [13]) for helpful discussions and clarifications about their work. We thank FuR for assisting with the collection of photos for the examples throughout this work.

References

[1] 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.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv e-prints arXiv:1412.6572*, 2014.

[3] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv e-prints arXiv:1605.07277*, 2016.

[4] A. Athalye and I. Sutskever, "Synthesizing robust adversarial examples," *arXiv e-prints arXiv:1707.07397*, 2017.

[5] 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.

[6] M. Khoury and D. Hadfield-Menell, "On the geometry of adversarial examples," *arXiv e-prints arXiv:1811.00525*, 2018.

[7] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," *arXiv e-prints arXiv:1805.12152*, 2018.

[8] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane, "Adversarial attacks on medical machine learning," *Science*, vol. 363, no. 6433, pp. 1287–1289, 2019.

[9] J. Stilgoe, "Machine learning, social learning and the governance of self-driving cars," *Social studies of science*, vol. 48, no. 1, pp. 25–56, 2018.

[10] H.-Y. Tsao, P.-Y. Chan, and E. C.-Y. Su, "Predicting diabetic retinopathy and identifying interpretable biomedical features using machine learning algorithms," *BMC Bioinformatics*, vol. 19, no. 9, p. 283, 2018.

[11] 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.

[12] 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.

[13] J. M. Cohen, E. Rosenfeld, and J. Zico Kolter, "Certified adversarial robustness via randomized smoothing," *arXiv e-prints arXiv:1902.02918*, 2019.

[14] 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.

[15] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *CoRR*, vol. abs/1608.04644, 2016.

[16] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of deep learning systems," *arXiv e-prints arXiv:1705.06640*, 2017.

[17] 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 arXiv:1712.02976*, 2017.

[18] 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 arXiv:1804.00097*, 2018.

[19] 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 arXiv:1902.06705*, 2019.

[20] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv e-prints arXiv:1312.6034*, 2013.

[21] W. Landecker, "Interpretable machine learning and sparse coding for computer vision," Ph.D. dissertation, Portland State University, 2014.

[22] 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.

[23] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Interpretable machine learning: definitions, methods, and applications," *arXiv e-prints arXiv:1901.04592*, 2019.

[24] 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.

[25] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*, pp. 818–833. Springer, 2014.

[26] 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.

[27] 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.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[29] Y. Cui, Z. Chen, S. Wei, S. Wang, T. Liu, and G. Hu, "Attention-over-attention neural networks for reading comprehension," *arXiv e-prints arXiv:1607.04423*, 2016.

[30] 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.

[31] 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.

[32] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J.-H. Jacobsen, "Invertible residual networks," *arXiv e-prints arXiv:1811.00995*, 2018.

[33] 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.

[34] 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.

[35] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[36] 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.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*, pp. 630–645. Springer, 2016.

[38] 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 arXiv:1812.01187*, 2018.

[39] 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.

[40] 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.

[41] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger, "Deep Networks with Stochastic Depth," *arXiv e-prints arXiv:1603.09382*, 2016.

[42] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise, "ShakeDrop Regularization for Deep Residual Learning," *arXiv e-prints arXiv:1802.02375*, 2018.

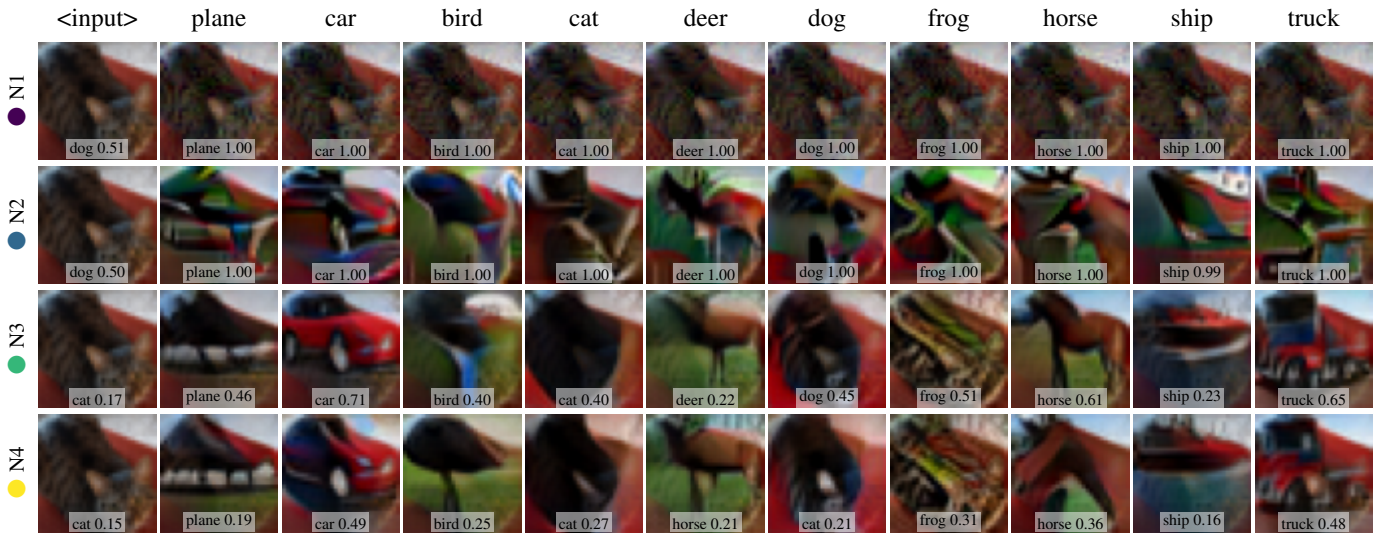
APPENDIX

A. *CIFAR-10*

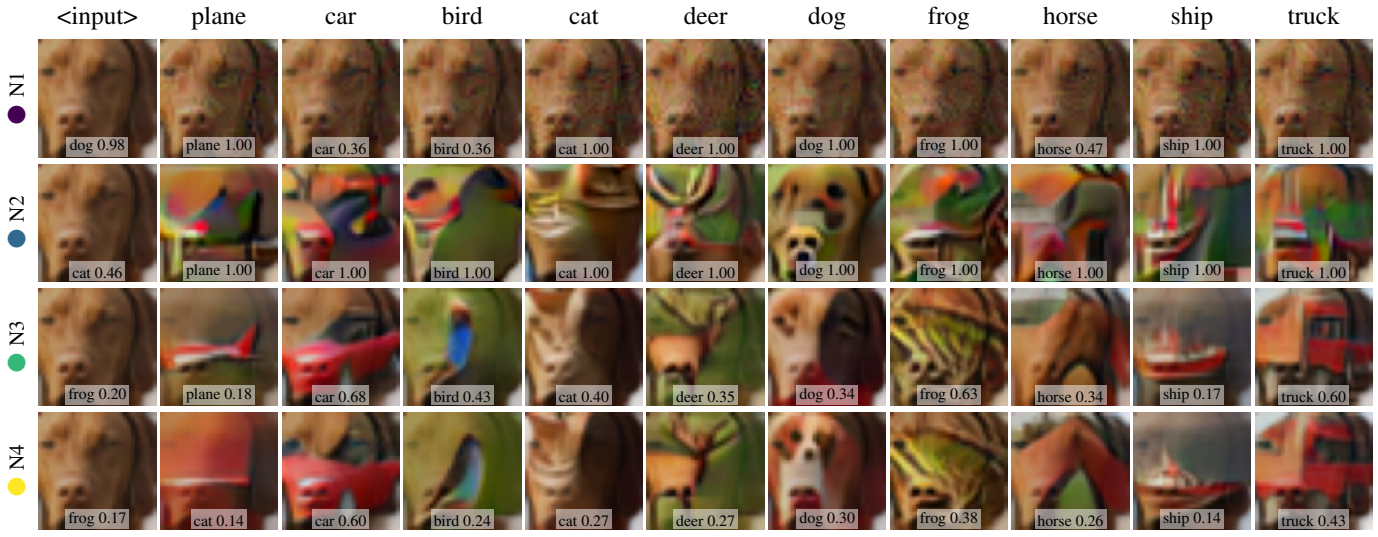
This appendix contains example adversarial images from the CIFAR-10 dataset, using the same networks annotated in Fig. 11. N1 denotes a traditional NN, N2 denotes an NN with only adversarial training, N3 denotes an NN with both our proposed regularization and adversarial training, and N4 denotes an NN trained only with our proposed regularization. At the top of each column is a label for the target of $g_{explain+}$, applied with $\rho = 0.1$.

As per Table II, N3 was the best performer in the BTR ARA category. The below Figs. S1, S3 and S4 support the ranking given by the BTR ARA. For example, in Fig. S1a, consider which network gave the most compelling explanation for each of the ten categories; we propose N3, N3, N4, N2, N3, N3, N3, N3, N3, and N4, respectively for each target column. By that count, N3 produced that most compelling explanation 70 % of the time. Further study of the relative explanatory benefit of these techniques from the subjective view of human operators is merited, but these results indicate that the BTR ARA is a strong measure of explanation quality.

Figure S2 demonstrates results of applying $g_{explain+}$ with varying levels of ρ; see the figure caption for more details.

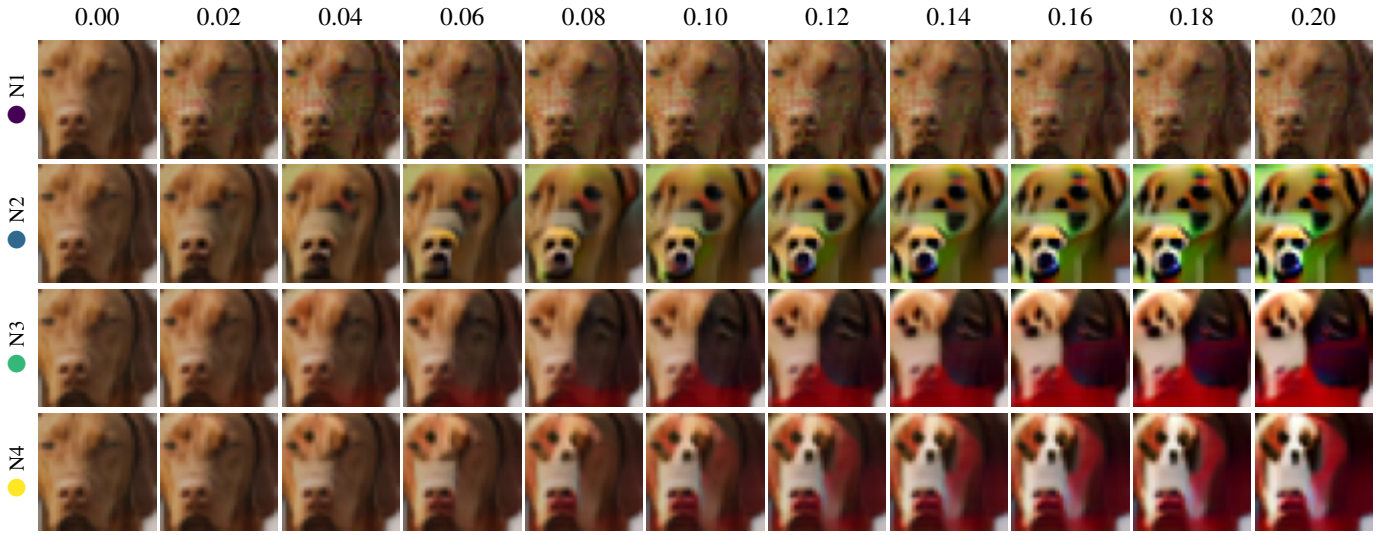


(a)

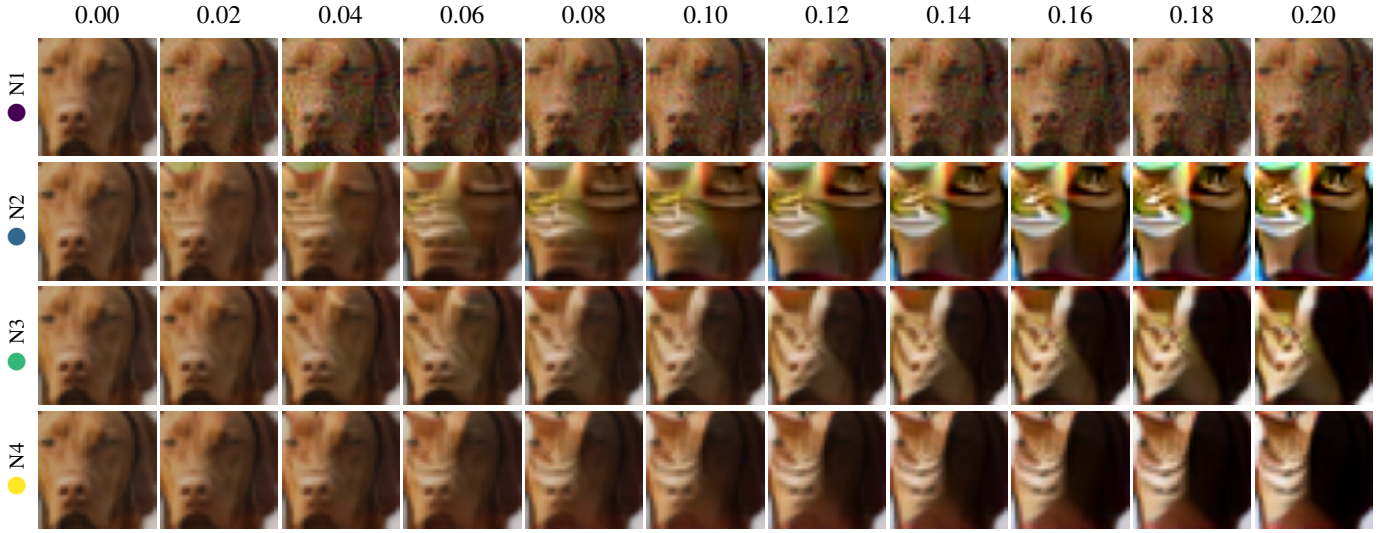


(b)

Fig. S1: See Appendix A for details.



(a)

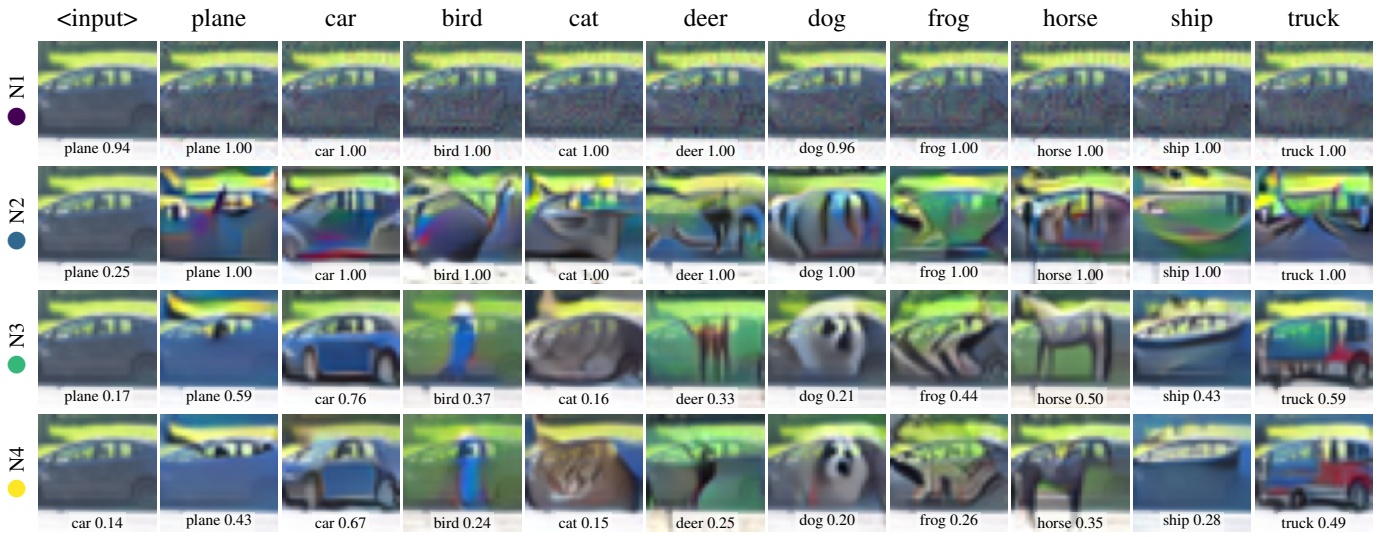


(b)

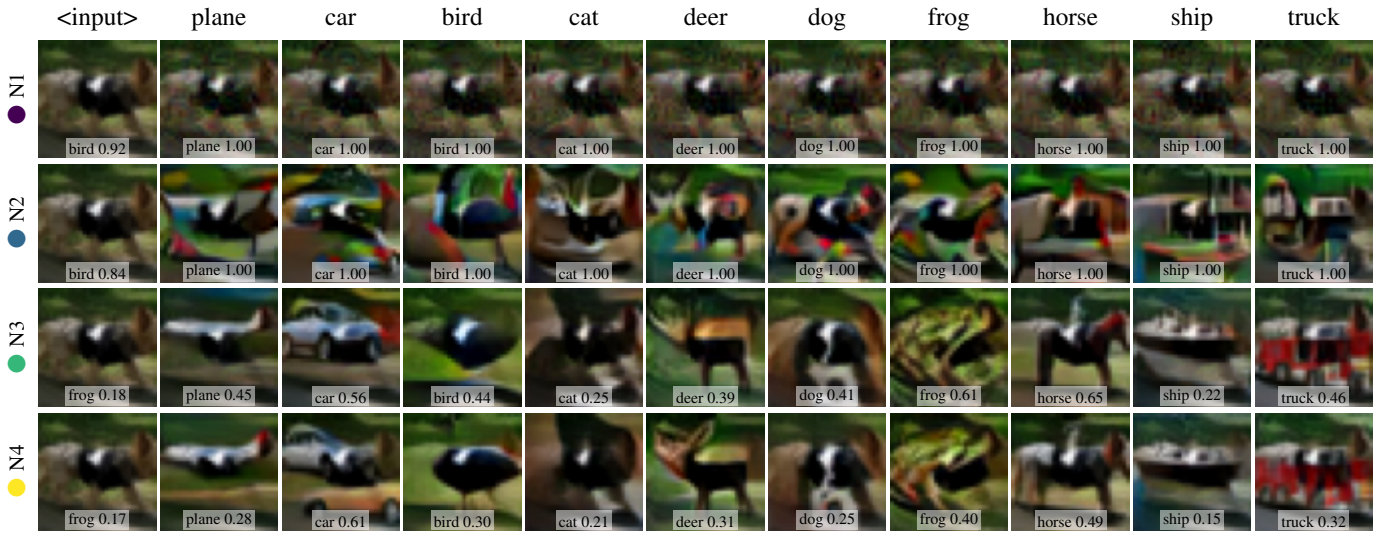
Fig. S2: While Fig. S1 demonstrates the relative performance of different visual explanations for various networks at a fixed noise magnitude $\rho = 0.1$, this figure focuses on demonstrating the evolution of those explanations as ρ is varied between 0 and 0.2 (the numbers at the top of each column). The input figure used was the same as Fig. S1b. For Fig. S2a, the target of $g_{explain+}$ is the "dog" class, and for Fig. S2b, the target is the "cat" class. The "cat" class was chosen because it was one of the wrong predictions with a higher level of confidence, particularly for the N2 network. Note that the progression is non-linear, with new features appearing at different levels of ρ.

Investigating Fig. S2a, one can see that the adversarially-trained N2 network relies on a small, dog-like feature around the nostrils of the original input image. N3 and N4, the networks which additionally have the regularization of Eq. (9) (and the other tricks from Section III), rely more on the overall shading of the face, and a larger dog-like feature which emerges by reshaping the left side of the input. From this, one might infer that the majority of CIFAR-10 training images are of full dogs, and the network has adopted this bias.

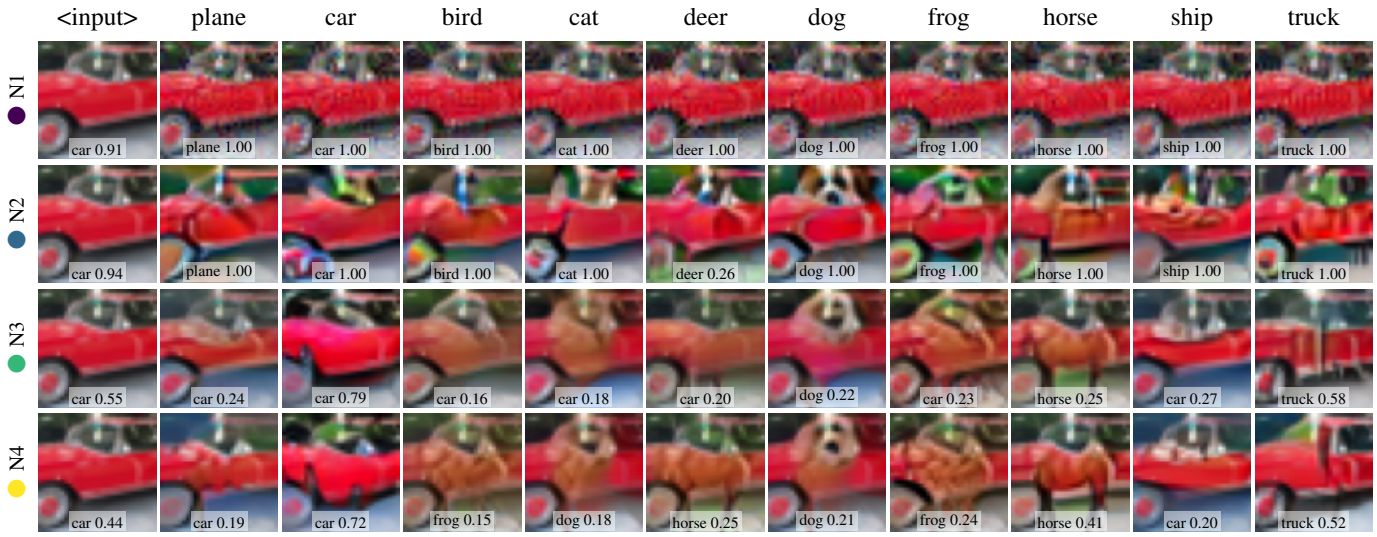
In Fig. S2b, the progressions for the N3 and N4 networks make it clear that the shading on the left side of the face may be adapted into the form of a cat, a likely reason for the misclassification. The N2 network exploits less of the source image to make this happen, but appears to suffer from a similar misconception.



(a)

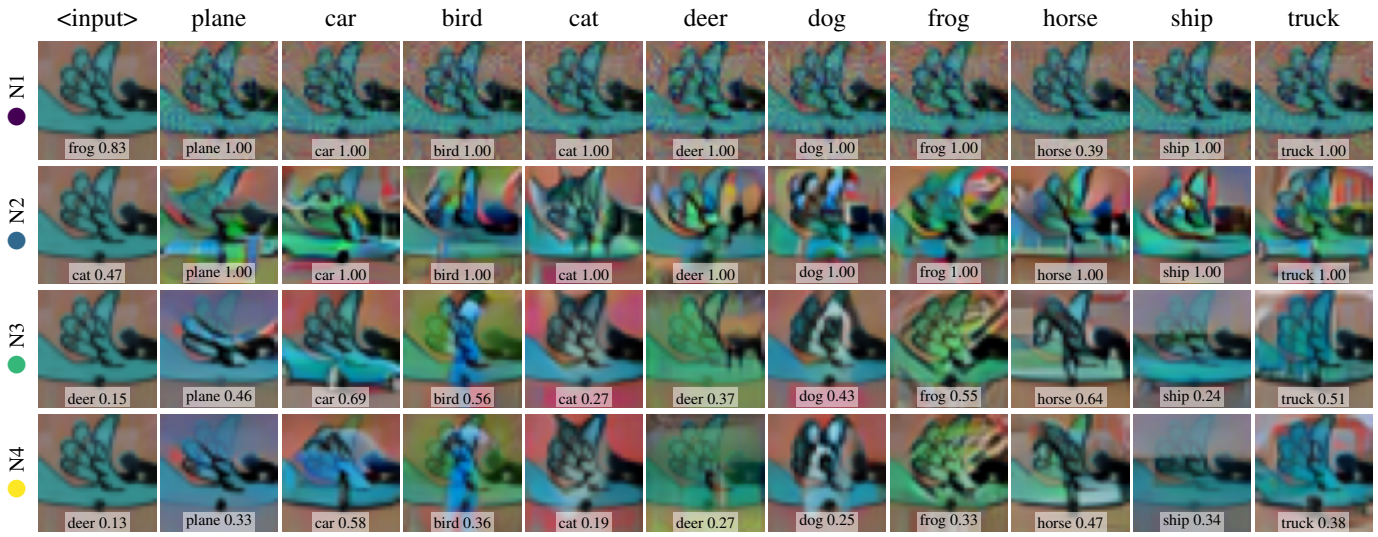


(b)



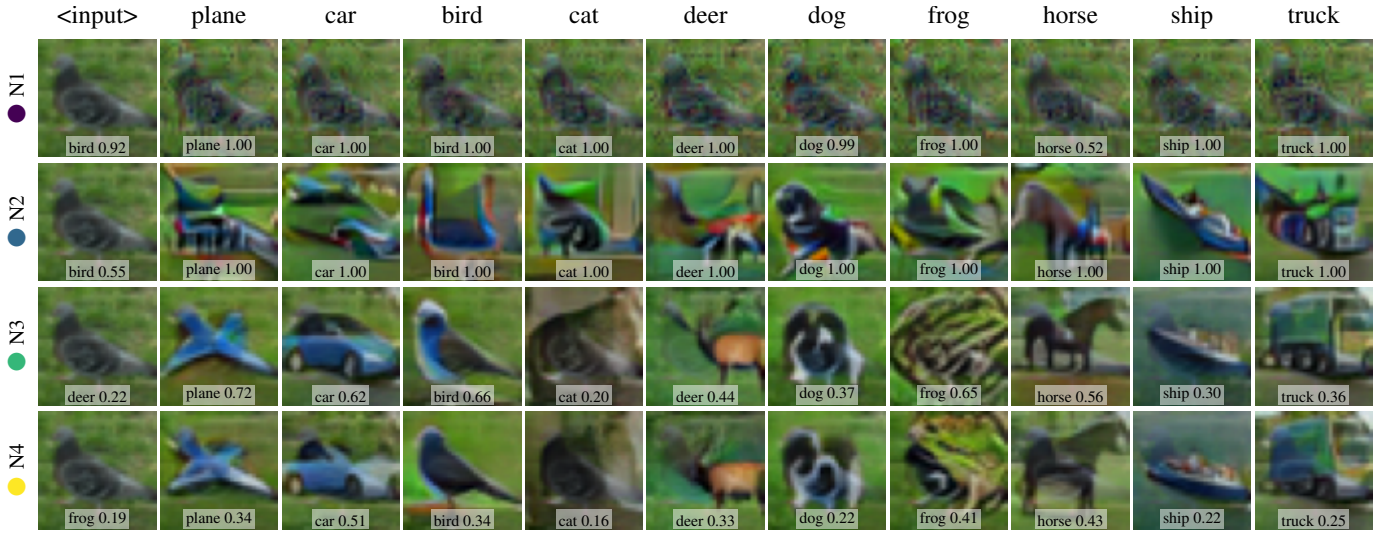
(c)

Fig. S3: See Appendix A for details.



(a)

(b)



(c)

Fig. S4: See Appendix A for details.

B. *CIFAR-10 HITL*

This appendix contains example adversarial images from the CIFAR-10 dataset that were part of the HITL experiments from Section IV-A17. Appendix B1 contains examples from networks trained on the original CIFAR-10 data; Appendix B2 contains comparison images between adversarial examples from the original network and versions trained with annotated HITL data.

*1) **Sample Annotations**:* Figure S5 demonstrates annotations from an annotator. Images are arranged in pairs; on the left is the original image, and on the right is an adversarial example constructed such that the difference between the adversarial class and the true class is $s_{adv} - s_t = 0.5$. The original class is written below each image. Annotators were asked whether or not the adversarial image still belonged to the original class; see Fig. 10. The values from each annotator are shown following the original label (as "yes" or "no" to being the same class).

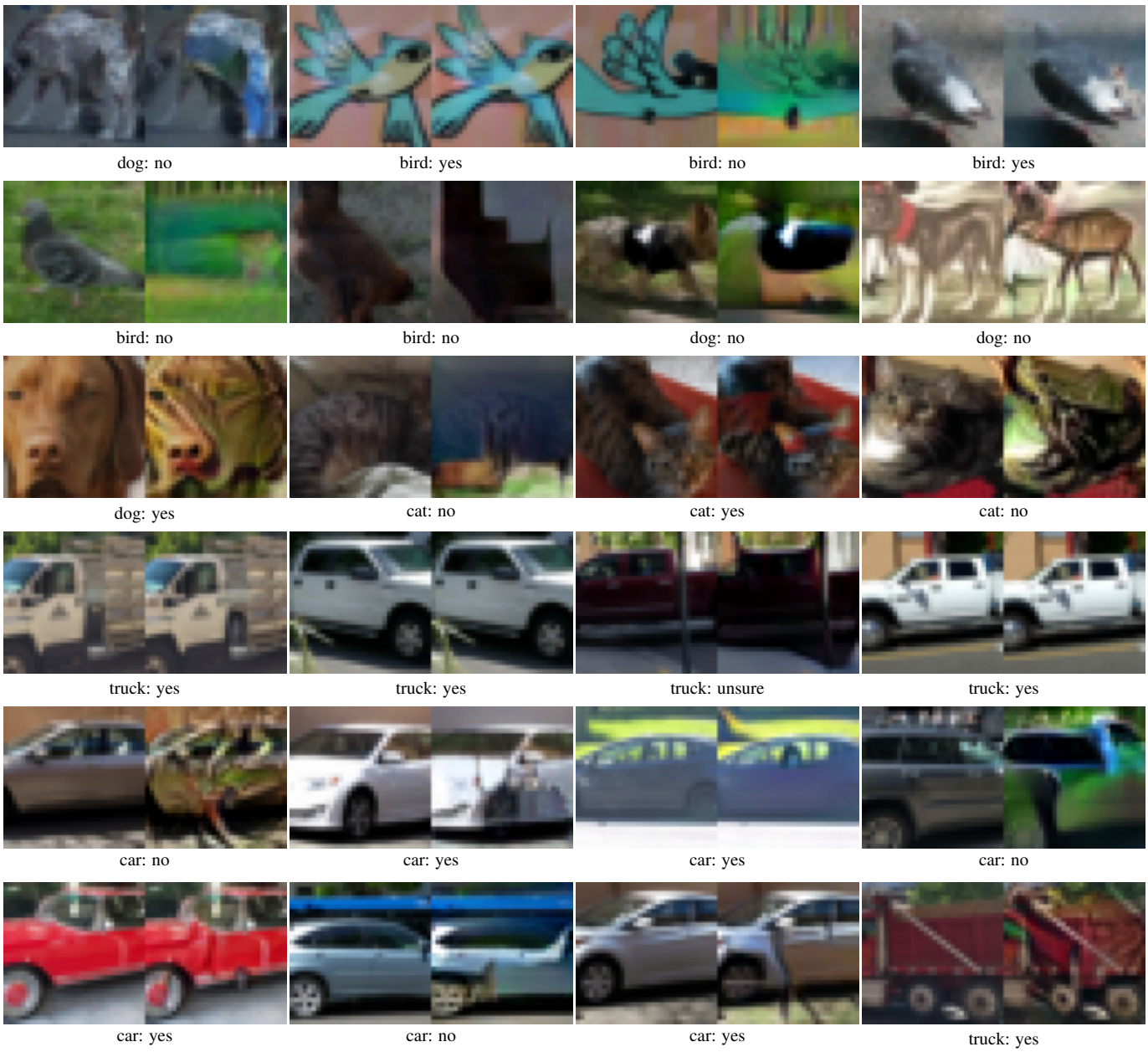
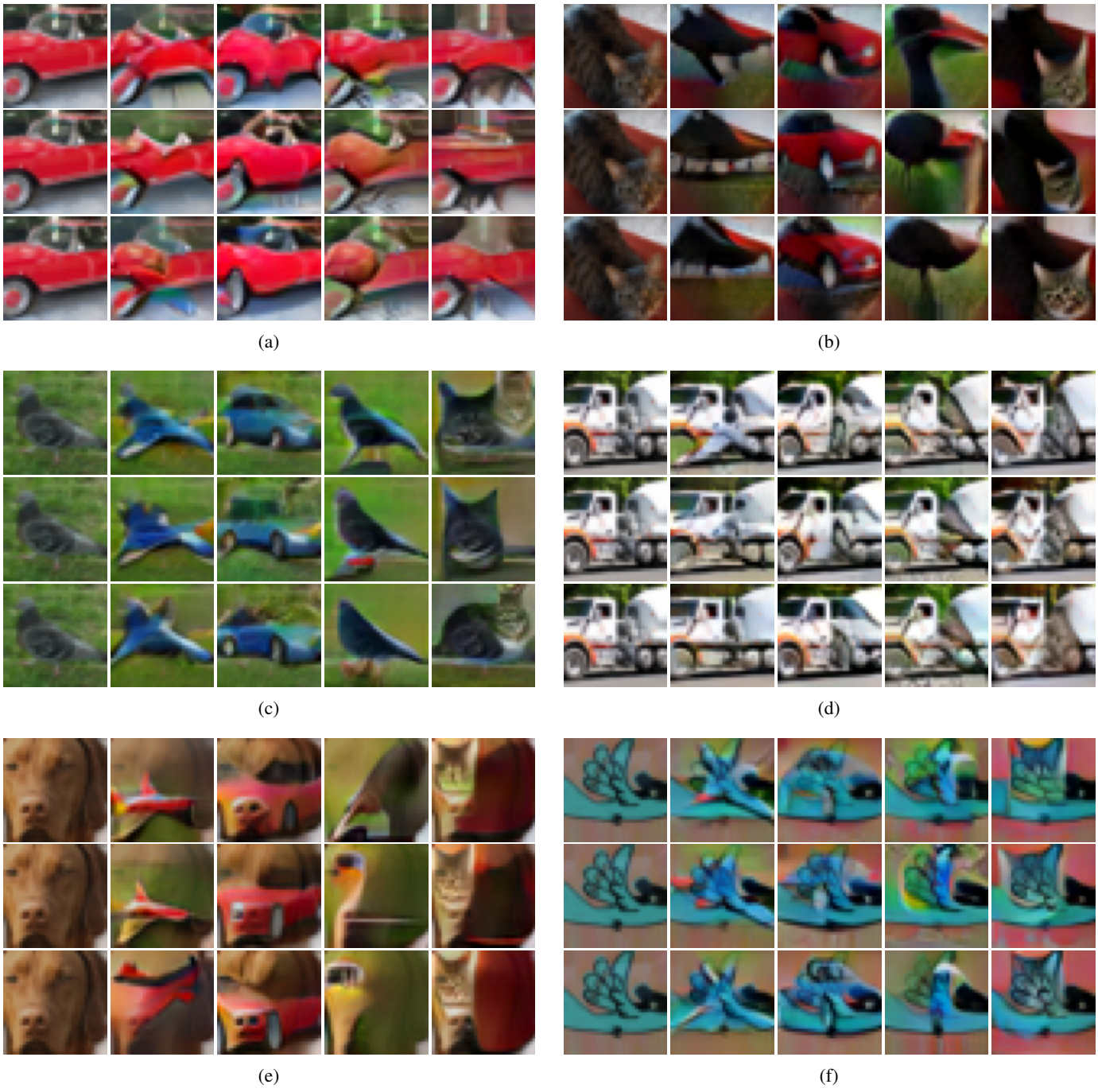


Fig. S5: See Appendix B1 for details.

*2) **Networks Trained with HITL Feedback**:* Figure S6 contains groups of three rows: the non-HITL version of the network for which annotations were made, a HITL network trained with 448 additional annotations, and a HITL network trained with 1,331 additional annotations. Just as the network with the most annotations demonstrated a modest quantitative improvement (Table II), inspection of these images demonstrates a more fully formed idea of each class. The left-most image of each group is the input, followed by adversarial examples formed from $g_{explain+}$ targeting the classes of airplane, automobile, bird, and cat.



(a) (b) (c) (d) (e) (f)

Fig. S6: See Appendix B2 for details.

C. *COCO*

This appendix contains example adversarial images from the COCO dataset. The networks used in these figures correspond to networks from Table III: C1 is the row labeled "With Eq. (9)," C2 is the row labeled "Combined Eq. (9) + HHAT," C3 is the row labeled "Balanced classes, Eq. (9) only," and C4 is the row labeled "Balanced classes, Eq. (9) + HHAT." "<guess>" is used to indicate an explanation for the highest-confidence prediction which was of the incorrect class, and "<real>" is an explanation of the correct class. The remaining columns show each network's interpretation of the image as the label at the top of each column (categories were chosen for an even distribution over object type). As described in Section IV-C, these images corroborate that adversarial training had little benefit for the COCO problem, potentially due to the many overlapping objects in the training data. As might be expected, the networks that were "balanced" resulted in less coherent explanations of the "person" class but better explanations for the other classes.

(a)

(b)

Fig. S7: See Appendix C for details.

(a)

(b)

(c)

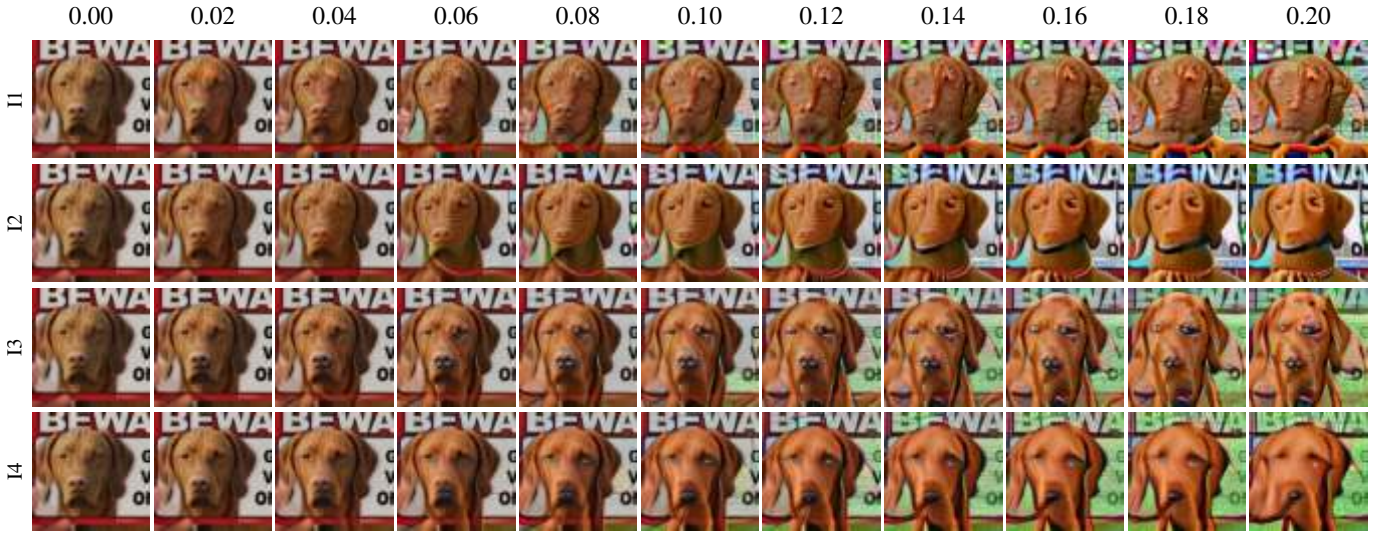
Fig. S8: See Appendix C for details.

D. *ILSVRC 2012*

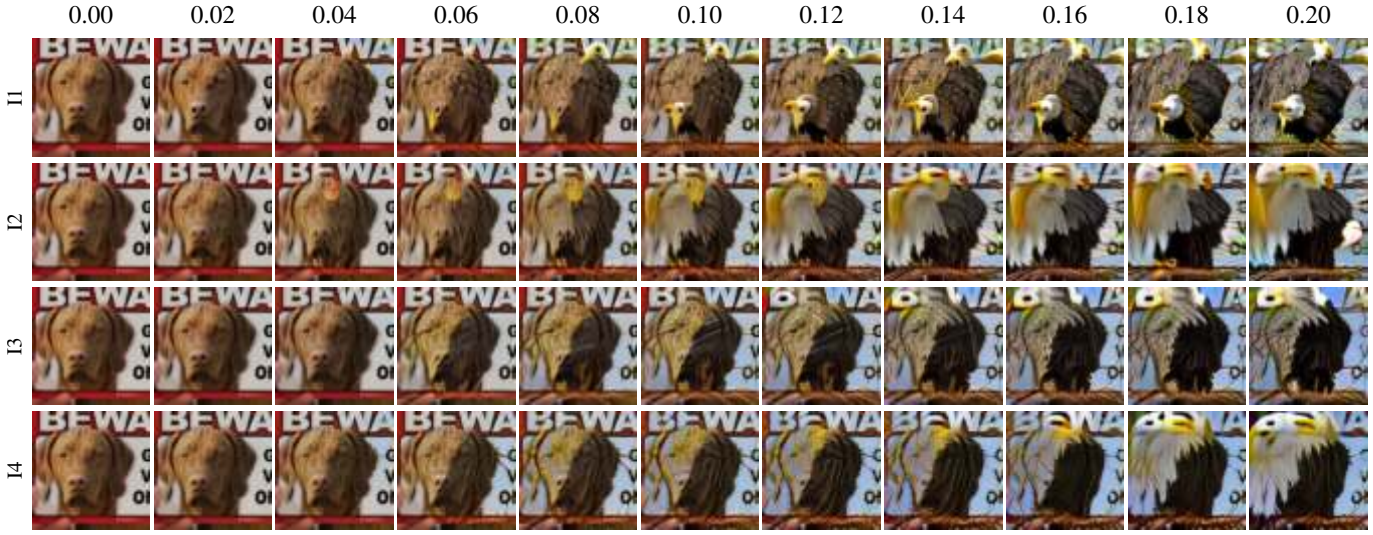
These images contain adversarial explanations for networks from Table III: I1 is the network labeled "$L_{target} = 3.1$," which had no adversarial training, I2 is the network labeled "$L_{target} = 3.1$ with HHAT, $\epsilon = 0.1$," I3 is the network labeled "$L_{target} = 5.0$," and I4 is the network labeled "$L_{target} = 5.0$, $L_{adv,z=2,q=1}$."

Like Fig. S2, the Fig. S9 demonstrates how explanations progress as ρ is varied. Between I1 and I2, adversarial training was clearly beneficial on ILSVRC, consistent with the BTR ARA results in Table III. We draw particular attention to the diminished effect on the background of the image in Fig. S9a, which is lettering on a sign. While I1, I3, and I4 all add a green hue to the background, the only network trained with both Eq. (9) and adversarial training did not exhibit this effect. For both Figs. S9a and S9b, the higher the value of L_{target}, the simpler and more coherent the explanation. While I4 exhibited the highest BTR ARA of 0.0538 compared to 0.0282 for I2, the best attack ARA was from I2, at a value of 0.0053 versus 0.0037 for I4.

Figures S10 to S12 follow a similar format to those in Appendix C, with "<guess>" and "<real>" having the same meaning, and the remaining columns being various targets for $g_{explain+}$.

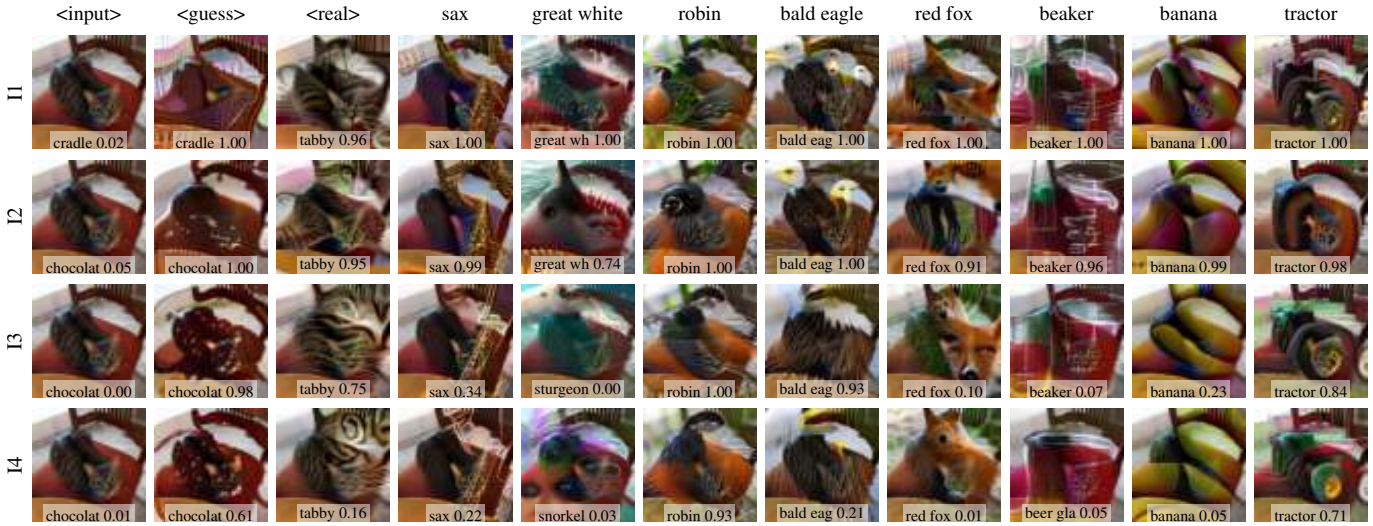


(a)



(b)

Fig. S9: See Appendix D for details.

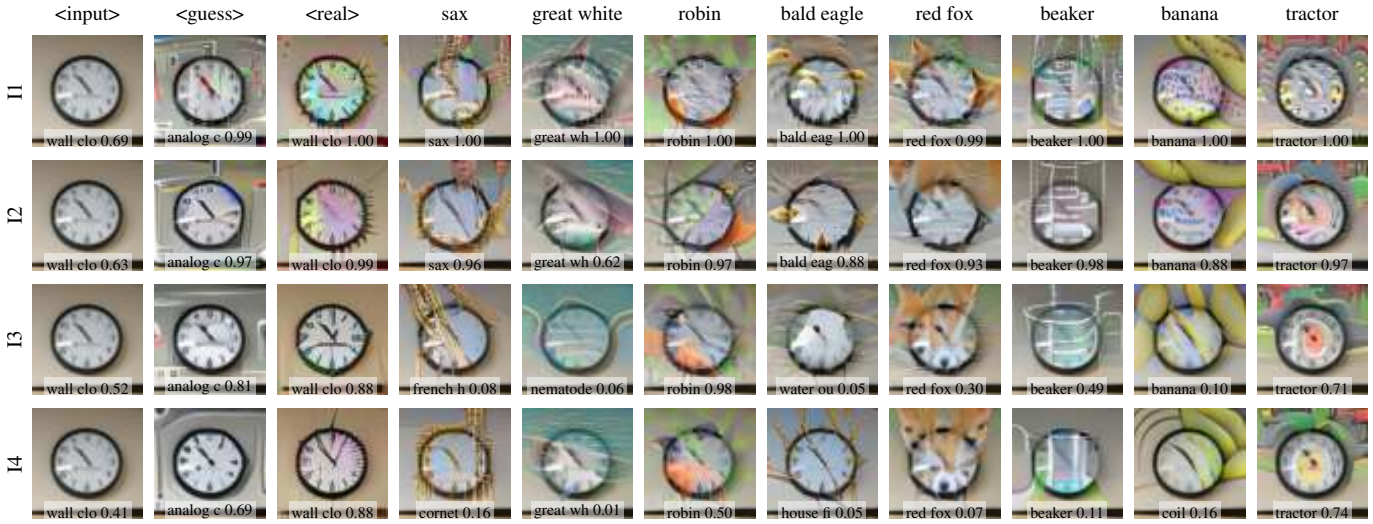


(a)

(b)

(c)

Fig. S10: See Appendix D for details.

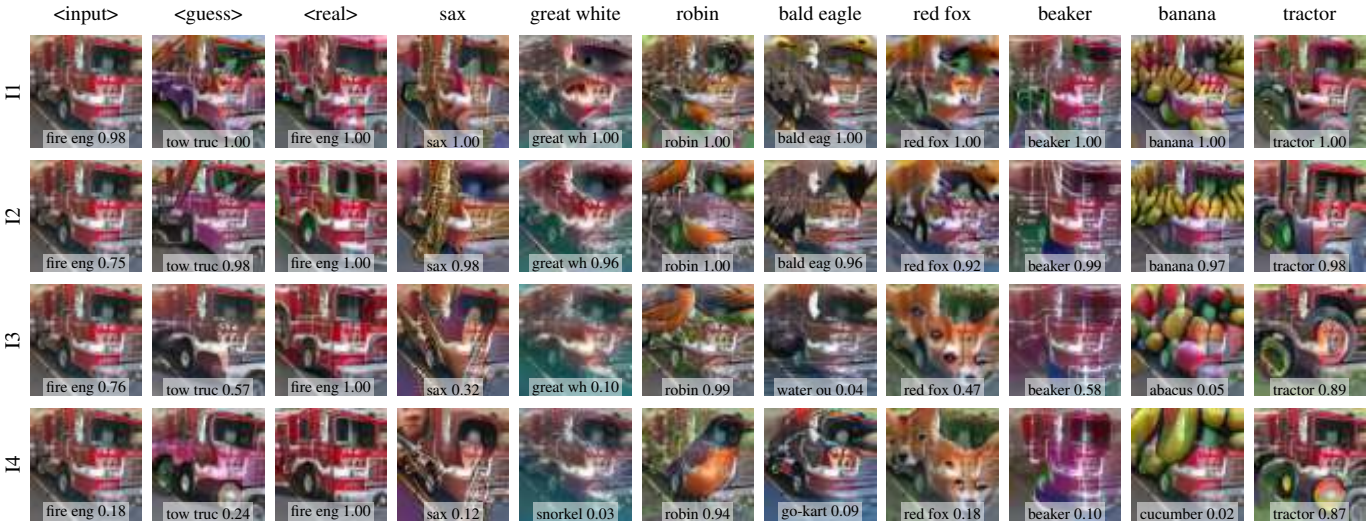


(a)

(b)

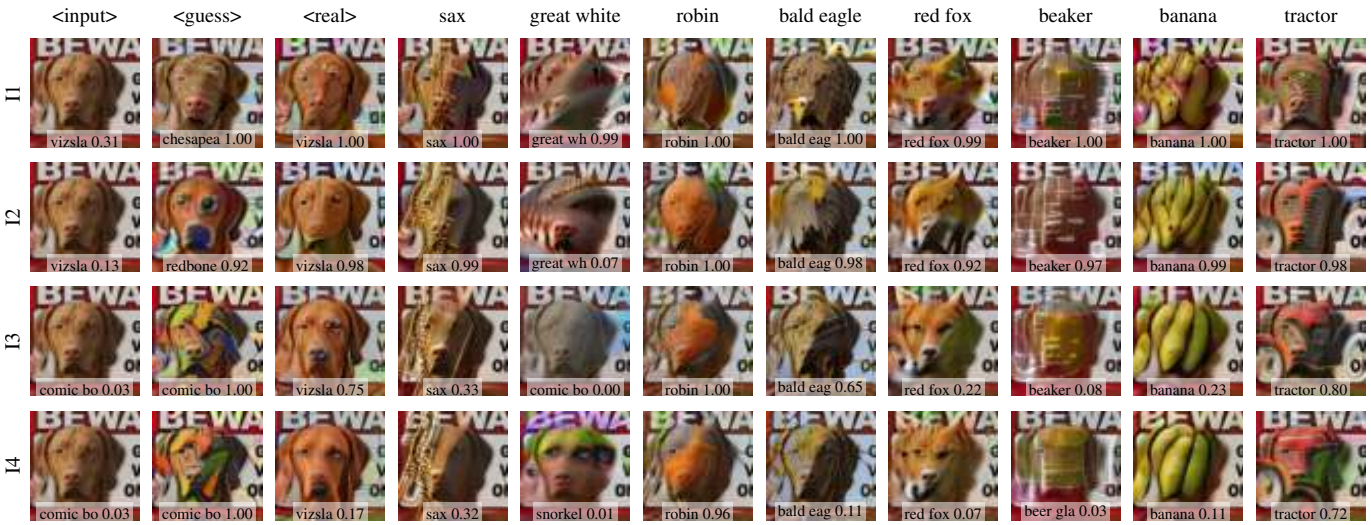
(c)

Fig. S11: See Appendix D for details.



(a)

(b)



(c)

Fig. S12: See Appendix D for details.