Hitting Depth: Investigating Robustness to Adversarial Examples in Deep Convolutional Neural Networks

Chris Billovits
Stanford University
cjbillov@stanford.edu

Mihail Eric
Stanford University
meric@cs.stanford.edu

Nipun Agarwala
Stanford University
nipuna1@stanford.edu

Abstract

Machine learning models, including Convolutional Neural Networks (CNN) are susceptible to adversarial examples - input images that have been perturbed to deliberately fool a model into an incorrect, high-confidence prediction without a visually perceptible change. Previous work shows that high-dimensional linearities cause these adversarial pockets of space. We first validate assumptions about the generalization of gradient-based and pattern-based adversarial examples using VGGNet. We show a process for visualizing and identifying changes in activations between adversarial images and their regular counterparts. Finally, we leverage information from these two approaches in a novel Bayesian framework to increase l_2 robustness to adversarial examples. Using this framework, we successfully improve the prediction accuracy on adversarial examples.

1. Motivation

Deep convolutional neural networks achieve near-human level accuracies on a variety of spatial recognition tasks. Due to the time and expertise necessary to develop maximal performance, many commercial applications of DNN's are based off the same small groups of models. The ability for an adversary to produce an image that produces an incorrect, high confidence prediction with low distortion from a desired image is a critical security risk. The security risk is precisely due to the width and fallout of the potential attack vector.

While developing robustness to adversarial examples has previously been approached as an academic exercise, recent research [7] has shown that the generalizability of adversarial examples permits a trivial black-box attack against DNN's. Approaches developed prior to this discovery do not allow practical test-time mitigations to adversarial attacks, or they require ingesting similar adversarial examples.

2. Related Work

Adversarial examples were introduced in [10], as images that produce a different class label when perturbed slightly. Further investigation by [2] identify the linear, high-dimensional nature of these models and introduce "fast gradient sign", a method to generate adversarial examples by perturbing the image in the direction of steepest ascent of the model's loss function. [2] found examples generated on one architecture to generalize to different architectures.

In 2015, Nguyen et al. [5] discovered a second type of adversarial example, generated by compositional pattern-producing neural networks (CPPN). Phenotypic elite images, incorporating abstract geometric forms, can be evolved across MNIST and ImageNet with the objective of inciting a high confidence prediction. These unrecognizable images do not have consistent human labels, yet often incite over 90% prediction accuracy on ImageNet, and 99.99% on MNIST. Nguyen et al. show these evolved images to largely generalize between two initializations of the same network architecture, and weakly generalize between architectures. [5]

Many methods have been proposed to increase model robustness to adversarial examples. Since linear models are susceptible by design, the task is to increase the distortion necessary to achieve a different label with high confidence.

The most immediate approach is to apply transfer learning on a new "adversarial" class, [5], and retrain with a high bias towards examples from this new "adversarial" class. Goodfellow et al. [2] simulate adversarial examples indirectly by modifying the top-layer objective function to jointly minimize the loss over regular images and regular images perturbed in the direction of the gradient. Adversarial examples generated by gradient ascent have also been interpreted as a noisy encoding. Gu and Rigazio [3] proposed stacking a contractive autoencoder with a CNN to de-noise the adversarial inputs, amounting to a layerwise Jacobian penalty. Finally, Ororbia II, Kifer, and Giles [6] unify previous work with autoencoders and retraining as special cases of regularizing the Jacobian of loss functions.

Although gradient information is typically required to produce high-confidence fooling images, Papernot et al. [7] (ArXiV pre-print) contribute a practical black-box attack on a DNN by querying only inputs and outputs of the network. Their approach was able to achieve fooling images with only hundreds of queries on a DNN available via MetaMind's API. [1]

3. Initial Setup

For the purposes of this project, we wish to make use of a visually rich dataset with images that span a diverse collection of classes. ImageNet is then a very natural choice. The 2014 ImageNet CLS-LOC training dataset consists of 1.28 million images tagged across 1000 different object categories. The 2014 CLS-LOC validation set consists of 50000 images tagged across the same 1000 object categories. We use the ImageNet validation set to derive adversarial examples to guarantee that unperturbed images do not contribute to the gradient of the network.

We use a pre-trained VGGNet model trained on the ImageNet CLS-LOC train set [8], and ran our experiments and modifications using Lasagne, a thin library built on top of the Theano machine learning framework [2] . VGGNet is a natural choice for investigation because aside from providing near start-of-the-art performance, 4096-layer features are often utilized for their generalizability to other learning tasks. Further, the homogeneity of the architecture lends itself well to visual interpretation at different layers.

4. Characterization and Generalization

The first step towards increasing robustness of Neural Network architectures to adversarial examples is to understand the nature of these images and how and why these examples generalize. We use two classes of examples to explore this: Fast-Gradient Sign generated and Rubbish Class examples (based on the term used in Goodfellow et. al.). For the evaluations, we have used a pretrained 19-layer VGG architecture as proposed by Simonyan et. al.[9]. VGGNet systems are known to be very high-performing with a homogeneous architecture, consisting of a deep network of convolution/pooling layers stacked one after another. [2].

4.1. Fast-Gradient Sign

Introduced in 2014 by Goodfellow et. al. [2], the Fast-Gradient Sign method perturbs the image in the direction of steepest ascent by a small ϵ. The claim is that such a perturbation is enough to generate generalized adversarial examples across different networks. Mathematically, the perturbed image we derive is as follows:

$$X_{adv} = X + \epsilon \cdot \text{sign}\left(\nabla_X J(X)\right) \tag{1}$$

Here, $\nabla_X J(X)$ is the gradient of the loss of the original 3x224x224 image, X, against the true label with respect to X and ϵ is such that $0 \leq \epsilon \leq 1$. Our evaluations suggested that having a larger ϵ would perturb the image enough to cause visual changes.

4.2. Rubbish Class (Fooling Images)

The evolved examples that fooled AlexNet generally are cast as different classes, and almost always with lower confidence. One of the main takeaways of VGG is that deeper is better - stacking small convolutional filters successively allows better detection of globally connected features, as the top convolutions span large swaths of the image. While these evolved examples are designed to exploit AlexNet, Nguyen et al. found the same evolved examples to generalize to GoogLeNet (2014) to a similar extent.

In addition, most of the CPPN evolved images share the same geometry. That is, it can be observed that the geometric features of the images are fairly consistent across images: straight lines, concentric circles and tessellated images are the norm, and not arbitrary bezier curves. We propose that these evolutionary examples show yet another reason that deeper is better. In particular, tessellations are simple to misinterpret if only local dependencies are encoded. Convolutional layers with size-preserving padding and stride of 1 are location-invariant, and will not incorporate mutual information in shallower networks. Filters that yield high activations in one part of the image will also yield high activations in another part of the image, regardless of their correlation.

4.3. Characterization

Close look at the fast-gradient sign generated adversarial examples provides us with 4 main categories of examples generated. Note that the images below are the original images, and that the adversarial images are identical to look at. On putting them together, no difference can be discerned, and hence it has been avoided:

- **True Adversarial**: These images are classified correctly at test time but on perturbing have been given a completely unrelated class label. As we can see in the images in figure 2. , the label is totally uncorrelated to the image itself.
- **Re-focused Adversarial**: A good percentage of images have completely unrelated labels, similar to the class above, but on looking closely at the image, one can notice the original image has a significant presence

[1] https://www.metamind.io
[2] http://deeplearning.net/software/theano/

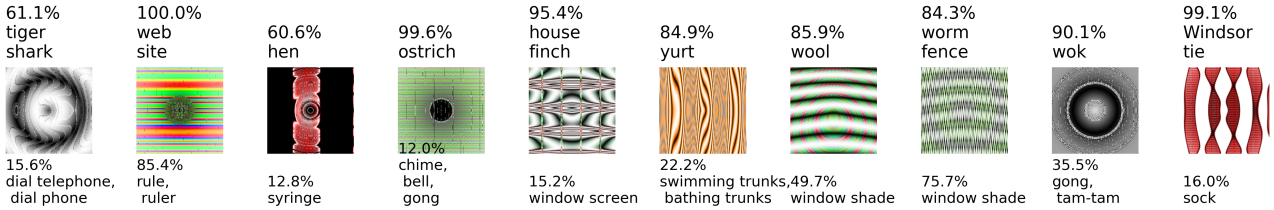


Figure 1: Sample of ten classes with CPPN encoded images derived from AlexNet. Predictions and confidence above each image correspond to AlexNet; Class predictions and confidence of VGG are shown below.

(a) Original Label: Carbonara Adversarial label: Swab, Mop

(b) Original Label: Bassinet Adversarial label: Running shoe

(c) Original Label: Bicycle for two Adversarial label: Plough

Figure 2: Sample original images for True adversarial examples

(a) Original Label: Fountain - 53% Adversarial label: Fireboat - 99%

(b) Original Label: Dishwasher - 67% Adversarial label: Plate rack - 61%

(c) Original Label: Dock - 63% Adversarial label: Cargo Ship - 69%

Figure 3: Sample original images for Re-focused adversarial examples

of the label it has been classified to. More specifically, the Conv Nets have "re-focused" their attention from the original entity in the image to a new significant entity present in the same image. See figure 3 for samples.

- **Conaturally Adversarial**: Labels having nuanced synsets, like dog and cat, lend themselves strongly to such a type of adversarial example. Perturbed images of such labels mostly stay within the same family, genus or species, as we can see below. Figure 4 displays samples of such a category

(a) Original Label: Keeshond - 53% Adversarial label: Shetland Sheepdog - 82%

(b) Original Label: American Chameleon - 53% Adversarial label: African Chameleon - 52%

Figure 4: Sample original images for Connaturally adversarial examples

- **Benign Adversarial**: The neural network may sometimes wrongly label the top prediction an image, more likely with a low probability. The adversarial example, though, is classified correctly with relatively high confidence. These examples are "adversarial" in the true sense of the definition, but are not dangerous to the working of the architecture. Figure 5 displays the needed images.

There were many images that did not change labels after perturbation, but those do not really categorize as adversarial, and hence were left out.

4.4. Generalization

Goodfellow et. al. [2] used a variety of ϵ values, classifiers and shallow networks to understand the nature of adversarial examples generated on the MNIST and CIFAR-10 datasets. This paper uses $\epsilon = 0.85$ to generate adversar-

(a) Original Label: Assault Rifle - 40% Adversarial label: Military Uniform - 54%

(b) Original Label: Car Mirror - 72% Adversarial label: Toaster - 62% shoe

(c) Original Label: Lawn Mower - 50% Adversarial label: Go-kart - 75%

Figure 5: Sample original images for Benign adversarial examples

ial examples on VGGNet, which uses a softmax classifier but is much deeper than the networks used before and is tested on the ImageNet validation dataset. On running the validation set on VGGNet with the above perturbation, we plot relevant histograms, as seen below. Note that probabilities greater than 50% are treated as high confidence because they represent the majority class score for the image i.e. the sum of all scores or probabilities from other classes is lower than the classified score. Another interesting observation was that adversarial examples having $> 50\%$ probability confidence in their top predictions had much lower confidence $< 25\%$ confidence in their 2^{nd} best predictions, indicating that VGGNet was relatively very confident in its top prediction.

Figure 6 shows the histograms that help describe the nature of adversarial examples. Of the 50000 images present, only 3730 images have $> 50\%$ confidence in their adversarial counterparts and only [insert number] images have some form of adversarial example. There are a few interesting observations (or patterns) to note:

1. It is not surprising to see that images classified with high probabilities are indifferent to the affects adversarial example generation. Small changes $(+/-1$ bit) in the pixel values do not change the predicted label at all.

2. As expected, it is easier to fool the network on images whose label it predicted with low confidences that those with higher ones. Since the network is more uncertain about its prediction for labels having lower probabilities, small perturbations are enough to cause the network to mispredict the label completely.

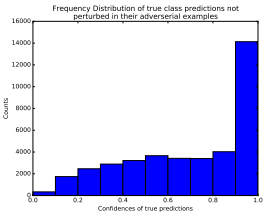
3. Images having moderate confidences $(30\% - 50\%)$ at original classification time tend to have high confidence adversarial examples rather than ones having low or very high probability original classifications. For low probability confidences, the neural network is completely unsure, and possibly incorrect, about the correct class of the image but a small steep perturbation is not enough to make the network very confident on any single class, though the prediction label may change significantly. On the other hand, for high confidences, the network is extremely confident about its class prediction and small perturbations are not enough to convince the network that the image is of another class. Having moderate confidences for an image indicates that the network is reasonably sure about its predicted class but not sure enough. This uncertainty lends itself to high volatility to perturbations and hence adversarial example generation.

Goodfellow et. al. [2] tested adversarial examples on shallow softmax and maxout networks and on datasets including CIFAR-10 and MNIST. They got very high misclassification rates (order of $89 - 99\%$) and confidences ($79 - 99\%$) for their adversarial examples. They did have an example on ImageNet but no statistics or results were presented. The current methodology uses VGGNet (with a softmax classifier), which is a much deeper network, on the ImageNet validation dataset. Despite these modifications, we see that adversarial examples are a reality and the depth of the network or the type of dataset does not prevent generalization. This validates the hypothesis of generalization made by Goodfellow et. al. [2] and highlights interesting results, discussed later.

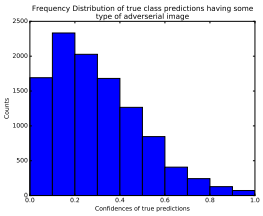
5. Visualization of Examples

In order to gain insight into how slight perturbations in the original images invoked via the Fast-Gradient Sign method, we visualized how the original and adversarial images were transformed as they were propagated forward through the layers of the VGGNet. We were especially interested in investigating how, if at all, we could perceive visual differences in the stacked convolution layers that precede the dense fully connected layers of the network. Ideally we would see convincing visual evidence supporting our hypothesis that most of the transformation in activations causing bogus classification labels in the adversarial examples happen not at the convolution layers but rather in the dense fully connected and dropout layers coming at the end of the network. We thus employed the following visualization algorithm:

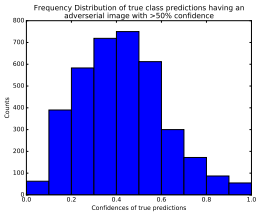
- Propagate both the original and adversarial examples through the network to get the filter activations at a desired layer.
- Assume a given layer produces m filter activations for a given input.
- Compute the maximum l_2 norm of the difference of corresponding filters between the adversarial and original images.



(a) Prediction confidences of Original Images having high confidence adversarial examples



(b) Prediction confidences of Original Images having some adversarial example



(c) Prediction confidences of Original Images having no adversarial example

Figure 6: Histograms showing how the prediction confidences of the original image classified by VGGNet correspond to its adversarial example, if any.

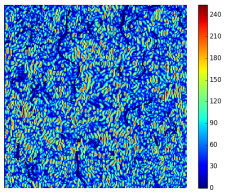
- Visualize the difference between activations for this maximum filter index as a heat map by projecting into a single channel pixel space.

Below we visualize these differences in activations at three convolution layers: conv1_2 (the second convolution layer applied to the input), conv3_1 (the first convolution layer applied after two max pooling layers have been applied to the input), and conv5_1 (the first convolutional layer applied after four max pooling layers have been applied to the input).

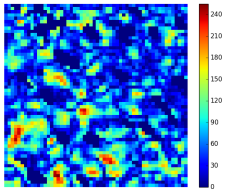
We notice interesting observations from the visualizations. In particular, we see in the sea slug input, the differences in filter activations at the conv1_2 layer are quite prominent with fairly high, fairly uniform activation differences distributed throughout the area of the images. The activation differences are far less prominent and more sparse in conv3_1, as indicated by the existence of fewer lighter colored patches. This can be attributed to the repeated application of max pooling layers which generally reduces the effect of a single input unit. By the time the input is propagated to conv5_1, we see an even larger reduction in number of high activation differences, as evidenced by the relatively few lighter-colored image patches. Again this can be a direct consequences of the repeated max pooling layers.

The relatively small differences in activations in the final conv5_1 layer at least do not discredit the hypothesis that the

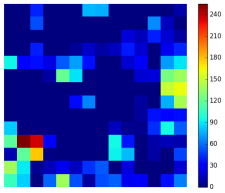
(a) Original Image



(b) Conv1_2



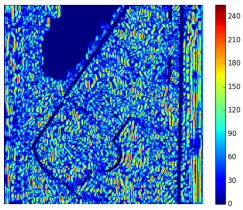
(c) Conv3_1



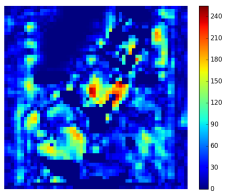
(d) Conv5_1

Figure 7: **Classified label**: Sea slug
Adversarial label: jigsaw puzzle

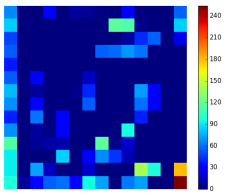
(a) Original Image



(b) Conv1_2



(c) Conv3_1



(d) Conv5_1

Figure 8: **Classified label**: Knee pad
Adversarial label: Band aid

dense fully connected layers of the VGGNet may be more directly responsible for the transformational differences that push a perturbed image across the decision boundary corresponding to a radically different label. That being said, given that the visual differences are so small at this layer, we are left with the question of what mechanism would be responsible for amplifying these small differences leading

to different classifications, as according to an l_2 metric of distance, the adversarial and original images are extremely similar. We believe that this suggests a fundamental limitation in the l_2 norm as a metric approximating visual perceptual difference and raises the need to investigate the use of other norm metrics.

6. Towards Generating Priors

Our approach for accounting for l_2 regularization is simple in principle. A CNN like VGGNet predicts the target class as

$$y^* = \arg\max_y p(y|x^{(i)}) \tag{2}$$

Normally, during training, it is assumed that the data is an independently and identically distributed (IID) sample of the true distribution. However, when potentially faced against an adversary whose goal is to provide fooling images, this assumption is no longer a constant, but a parameter. Let r be a parameter representing the confidence that an input is natural, as opposed to fraudulent. The modified prediction function looks to maximize y and r jointly given x:

$$y^* = \arg\max_y p(y, r|x^{(i)}) \tag{3}$$

$$y^* = \arg\max_y p(y|x^{(i)})p(r|y, x^{(i)}) \tag{4}$$

Thus, our updated estimate requires two separate components. We approximate $p(y|x^{(i)})$ with the conditional from a pre-trained VGGNet that generates (1). The next section derives an estimate of $p(r|y, x^{(i)})$.

6.1. Density Estimates

Unfortunately, parametric estimates of $p(r|y, x^{(i)})$ are infeasible, because the prescribed distribution of $p(r|y, x^{(i)})$ depend strongly on the power and persistence of the adversary. Intuitively, a parametric estimate of fraudulence would fail to be robust against general adversaries, because at test time, the network has no notion of what sort of images it will receive. Any such parametric priors (such as those learned by linear regression) can easily be represented as a stacked network architecture similar to Gu and Rigazio, [3], [5]. The stacked network is then vulnerable by exploiting linearity by the same mechanism. Hence, we use non-parametric methods to model $p(r|y, x^{(i)})$.

We propose modeling $p(r|y, x^{(i)})$ using Gaussian kernel density estimation. Generally, density estimation assigns density to some point as the piecewise sum of kernel density for each observed point x, given our training set $\{x_1, \dots x_n\}$:

$$f(x) = \frac{1}{nh}\sum_{i=1}^{n}\frac{K(x - x_i)}{h} \tag{5}$$

where h is a bandwidth hyperparameter governing the uniformity of the estimate, and K is the N-dimensional gaussian kernel

$$K(x; \Sigma) = \frac{1}{(2\pi|\Sigma|)^{N/2}}e^{-\left(x^T\Sigma^{-1}x\right)} \tag{6}$$

These density estimates f serve as the per-class priors for a given input image.

The density function f is a biased estimator of the true distribution, but in high dimension the convergence of the estimator is poor. Specifically, the best possible mean integrated square error for a non-parametric density estimator with n support examples and dimension d and its true distribution is $O(n^{-\frac{4}{4+d}})$. While the end goal obviates the need for accurate densities in the entire space of examples, it is clear that significant dimensionality reduction is required to produce correct order-of-magnitude probability estimates for $p(r|y, x^{(i)})$. In order to guarantee a minimum of support in all feature dimensions, we perform density estimation in $k = 8$ dimensions, so that each class of ImageNet has at least 2^k support points.

The underlying task becomes how to construct encodings that minimize the distortion under our squared distance used by the Gaussian kernel, which is investigated in the next section.

6.2. Feature Encoding

The theorem of Johnson and Lindenstrauss '84 proves [1] that n dimensional points can be embedded into

$$k = O\left(\frac{log_2(n)}{\epsilon^2}\right)$$

dimensions while maintaining an l_2 distortion of $(1 \pm \epsilon)$ in polynomial time. Thus, there is a reasonable hope that orders of magnitude of l_2 distances can be preserved, even when classes of images are projected down to $k = 8$ dimensions.

To learn the distinctive features of a class, we use a convolutional autoencoder, trained to minimize the reconstruction l_2 error. In order to maintain efficient training behavior, we approximate the optimal encoding from the original image as the weights learned by $p(y|x)$ in the original network. Specifically, we use VGGNet's convolutional representations up to the first 4096 fully-connected layer. From there, for each class, we project the 4096 features onto the $d = 128$ principal eigenvectors, which is the maximal variance projection into d dimensions. Empirically, this projection maintains $68 - 72\%$ of the

variance from the 4096 layer.

From the $d = 128$ layer, we set up a 7-layer deep autoencoder, whose goal is to provide a non-linear 8-dimensional encoding of the $d = 128$ layer that minimizes reconstruction loss. Each layer maps the input layer to an output half its dimensions, and a leaky rectified linear unit function (relu) is applied between each layer with $\alpha = 0.33$. The choice of leaky relu is necessary to avoid producing a singular covariance matrix in density estimation.

This autoencoder is trained on each class using Adam, a second order momentum optimizer. At test time, the contractive portion of the network is used to produce the contraction from 128 to 8 dimensions. These 8 dimensional encodings are run through the Gaussian KDE algorithm to generate a value proportional to $p(r|y, x^{(i)})$, The density is normalized over the 1000 classes after the final computation of $p(y|x^{(i)})p(r|y, x^{(i)})$, yielding the probabilities found in the next section.

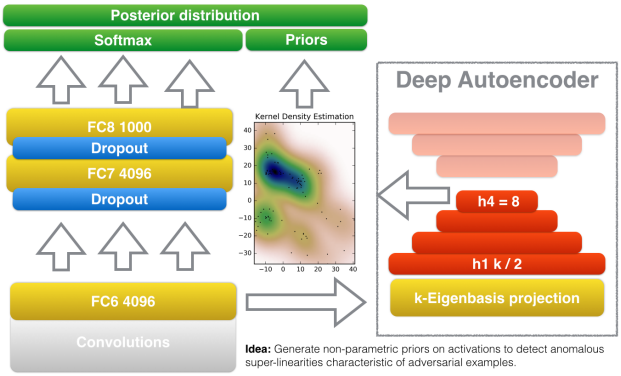


Figure 9: Schematic for the modified network architecture.

7. Inference and Results

Looking at the priors alone, the mean un-normalized density in 8 dimensions per class is $1.16e^{-13}$

Generally, we find that 80% of images previously classified with high confidence have been mitigated. Due to the healthy spread of types of adversarial examples, this number is not 100%.

Figure 10: **Adversarial Probability**: 0.76%
Posterior Probability: 0.81%
Adversarial Label of Alaska Crab was correct and hence boosted

VGGNet initially classified the image in figure 10 with 54% confidence. With adversarial perturbation the image label reverted to the correct label of "Alaska crab". The posterior estimate revised the adversarial probability up to 81% from 76%. In this case, the initial prediction is clearly incorrect, and our priors were able to recognize the image to be similar to its canonical class label without being influenced by the adversarial perturbations.

Figure 11: **Adversarial Probability**: 0.77%
Posterior Probability: 0.13%
Adversarial Label of Knee Pad was incorrect and hence significantly reduced.

For figure 11, VGGNet initially classified the image as a Holster but the adversarial label was a knee pad, which is clearly incorrect. The posterior estimate of knee-pad was significantly reduced and holster boosted, thus affirming that the prior was able to detect the correct class

Figure 12: **Adversarial Probability**: 0.47%
Posterior Probability: 0.46% Adversarial Label of Coyote was very similar to that of Greyfox.

Lastly, for figure 12, VGGNet initially classified the image as a Greyfox but the adversarial label was a Coyote. This adversarial example is under the conatural category, and hence due to the similarities in features, the priors had similar probabilities and there wasn't much of a change in the posterior distributions.

After generating Fast-Gradient sign adversarial examples and comparing them to the results of Goodfellow et. al. [2] (re-stated in Section 4), we concluded that **Deep Convolutional Neural Network are less susceptible to adversarial examples than shallow networks**. We get that 7.46% of images in the ImageNet validation set have adversarial examples specially constructed for VGGNet, as compared to the roughly 99% error rate that Goodfellow et. al. [2] achieved. Such a claim is consistent with the universal approximator theorem [4] because deeper networks should be able to model functions that are more robust to adversarial examples.

Furthermore, we hypothesize that **Dropout may decrease robustness of architectures to adversarial examples**. By reducing co-adaption of features, perturbation of 1 neuron specializing in a feature can drastically reduce the likelihood of the image being labeled a certain class whereas having greater co-adaptation would prevent such a reduction due to presence of other neurons specializing in similar features allowing for feature augmentation

8. Future Work

We can also further experiment with visualizing the activations at various layers between an original/adversarial image pair. In particular, using the visualization technique we described in (5), we can also visualize the activations for a representative image from the specific adversarial class of our input. This representative image can be selected via a number of methods. One possibility is to take the nearest neighbor image (using an l_2 metric) in the set of training images for a given class to our input image. This may allow us to compare how the network transforms the "canonical" image of a certain class compared to our specific adversarial example.

While include Bayesian priors did not degrade generalization performance on the validation set, there was no mechanism to ensure that previous correct classifications were maintained. One possible general approach is to optimize the prior under a constrained optimization task. Specifically, a max-margin classifier could replace the softmax top layer of the original network, and enforce constraints α_i such that if margin on a batch of examples is violated more than ϵ, then the loss explodes. The task would be to achieve maximal variational distance from the uniform distribution on the prior, contingent on retaining our ϵ margin on the train set. This objective could be efficiently evaluated using the pre-trained weights of VGG or some other DNN.

References

[1] S. Dasgupta and A. Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, Jan. 2003. 6

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014. 1, 2, 3, 4, 8

[3] S. Gu and L. Rigazio. Towards deep neural network architectures robust to adversarial examples. *CoRR*, abs/1412.5068, 2014. 1, 6

[4] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989. 8

[5] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. 2015. 1, 6

[6] A. G. Ororbia, II, C. L. Giles, and D. Kifer. Unifying Adversarial Training Algorithms with Flexible Deep Data Gradient Regularization. *ArXiv e-prints*, Jan. 2016. 1

[7] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against deep learning systems using adversarial examples. *CoRR*, abs/1602.02697, 2016. 1, 2

[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 2

[9] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 2

[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. 1