VISUALIZATIONS OF DECISION REGIONS IN THE PRESENCE OF ADVERSARIAL EXAMPLES

Grzegorz Świrszcz, Brendan O'Donoghue & Pushmeet Kohli

ABSTRACT

From a geometric standpoint, training a classification model is equivalent to defining a partition of the feature space into a union of subsets, where each subset corresponds to a particular decision region. The quality of the model is dependent on the geometry of such a partition being consistent with the desired properties of the model. Presenting those properties in a fashion that can be studied and understood by a human presents a large challenge, due to the high dimensionality of the objects studied. The goal of this paper is to present visualizations showing the shape of decision regions with a focus on adversarial examples, and to highlight and discuss the challenges of creating a good, scientifically helpful visualization.

1 INTRODUCTION

The importance of having tools for understanding and debugging machine learning models is becoming critical as models are deployed into real-world situations. In this paper we focus on visualization, the significance of which is by now well agreed upon by machine learning practitioners and theoreticians alike. The goals of this paper are twofold. Firstly, we present a methodology for visualizing the decision regions of models, with a special focus on adversarial directions in the input space (*i.e.*, those directions that most rapidly change the class label (Szegedy et al., 2013; Goodfellow et al., 2014; Uesato et al., 2018)) and to highlight and discuss the challenges of creating a good, scientifically helpful visualization. Secondly, we present some of our findings on standard models trained on an image classification task, using our methodology. For the readers not familiar with the topic of adversarial examples a brief introduction is included in the Appendix B.

2 METHODOLOGY

In many machine learning tasks the training data can be represented as a collection of points in some euclidean space $\mathbb{R}^d$, which is often very high-dimensional. In this paper we focus on image classification, and in particular on the CIFAR-10 and MNIST image datasets. In each case we normalize the values of each pixel coordinate to lie in the $[0, 1]$ interval. Thus, in case of CIFAR-10, each pixel is represented as a 3-D vector in $[0, 1]^3$, and the whole image can be represented as a $32 * 32 * 3 = 3072$ dimensional unit hypercube: $[0, 1]^{3072}$. Similarly, the MNIST dataset can be treated as a $[0, 1]^{784}$ unit hypercube. It goes without saying that visualizing such a space is very difficult for a human, though being able to visualize how the model partitions the space would be extremely useful for understanding the model behaviour, debugging bad performance, and tackling adversarial input examples. In this vein, a major challenge for the visualization community is to find meaningful ways of representing this very high-dimensional space in such a way that a human can understand it, without losing too much useful information. In this paper we propose such a visualization scheme and use it to demonstrate some interesting properties of modern neural networks trained on the above mentioned image classification tasks.

Throughout the rest of the paper we shall denote the hypercube $\{\mathbf{x} \in \mathbb{R}^d : 0 \leq |x_i| \leq 1 \text{ for } i = 1, \ldots, d\}$ by H for brevity.

2.1 "THE ART OF SLICING"

In this paper we adopt the approach of studying intersections of the input data hypercube H with 2-dimensional planes. We refer to these intersections as 'slices'. When we visulize a slice each

point will be color-coded according to the label assigned by the classification model being studied. One immediate practical challenge is determining the size of the slice presented, as well as the impossibility of representing an infinite plane in a picture. This motivates the following definitions.

Definition 1. *Let $\mathcal{L}$ be a 2-dimensional plane in $\mathbb{R}^d$ satisfying $\mathcal{L} \cap H \neq \emptyset$. Let v be a vector contained in $\mathcal{L}$. Then a* **slice** *of H determined by $\mathcal{L}$ and vector v is the smallest rectangle R such that $\mathcal{L} \cap H \subset R$ and one side of R is parallel to v.*

Definition 2. *A* **feasible point** *in a slice S is any point belonging to the intersection $S \cap H$.*

2.2 Technical aspects of the methodology

The main technical challenges for our visualization procedure were finding the slice size, grid size and identifying which gridpoints were feasible points. Also, given the fact that the model has to be evaluated at thousands of points, performance was another issue. We solved these issues as follows:

For identifying the slice size and feasible gridpoints we used the flood fill algorithm (Torbert, 2016). A point from the slice known to be feasible was selected as the seed of the algorithm, and in each round the grid neighbors of points already added which are also feasible are added, until no more additions are possible. Since the feasible set is convex this approach is guaranteed to select all points in the correct slice eventually.

The performance of the algorithm was improved using batching the candidate points (batch size 64 turned out to be optimal for the used hardware) and standard HPC tricks like memoization.

The tool we created allows also for extra features to be visualized, like probabilities at the points of interests and graphs of their values - see Figure 7 in the Appendix.

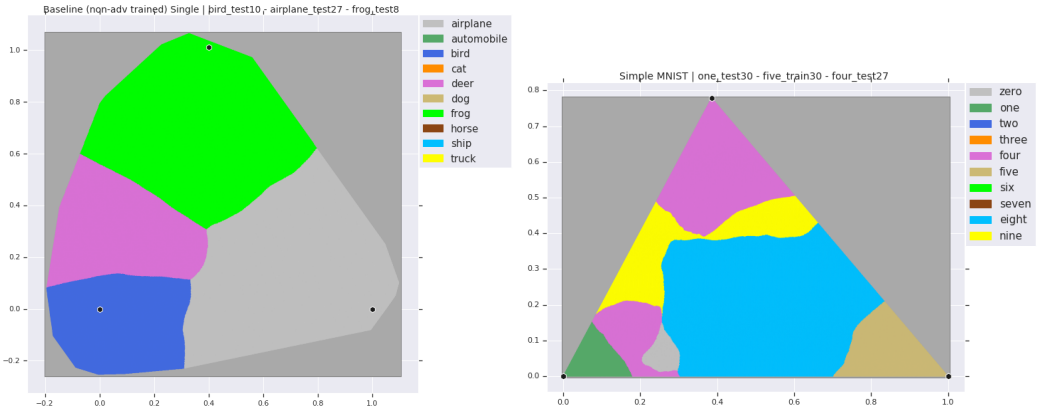

Figure 1: Example of slices. Left - a slice of a CIFAR-10 image space. Four different regions regions corresponding to “airplane”, “bird”, “deer” and “frog” are visible. Right - a slice of an MNIST image space

2.3 Decision Regions

Every classification model is a mapping $\mathcal{M} : H \to L$, where L is a set of n labels. In case of CIFAR-10, the set $L = \{\text{airplane}, \text{automobile}, \text{bird}, \text{cat}, \text{deer}, \text{dog}, \text{frog}, \text{horse}, \text{ship}, \text{truck}\}$, while in case of MNIST $L = \{\text{zero}, \text{one}, \text{two}, \text{three}, \text{four}, \text{five}, \text{six}, \text{seven}, \text{eight}, \text{nine}\}$. The decision regions are subsets $H_i \subset H$ defined as $H_i = \mathcal{M}^{-1}(l_i),\ l_i \in L$ for $i = 1, \ldots, n$. In order to visualize them, we assign to each label a color from a fixed color palette - see Figure 2.2.

According to definition (1) a slice is a rectangle. Nevertheless, except for a rare cases, an intersection of a 2D-plane and a hypercube is rarely a rectangle. Thus, in almost every slice, some points would not correspond to any point in the image space. We color those points `gray`.

The intersection of a plane and a hypercube is always a convex polygon, but as a dimension grows, the maximum number of its vertices grows exponentially. Thus, when looking at the slices, the

points of interest will often form a shape looking rounder and more symmetric than might be expected. Depending on the situation they can be very round and elongated, triangular (the typical situation for MNIST) or rectangular - and anything in between. Visualizations of the sort similar to the ones presented in this paper have appeared in the literature recently (Warde-Farley & Goodfellow, 2016; Gong et al., 2017), however to our best knowledge, the prior work focuses only on small neighbourhoods of the points of interest, rather than presenting "the full picture" - finding the proper scale that would include all points of potential interest. Figure 4 displays the differences between 'church window plot' approach and choosing the slice based on adversarial directions.

2.4 A "THIRD POINT DILEMMA"

A 2-dimensional plane is determined uniquely by a choice of 3 non-colinear points. Thus, we are choosing our slices by specifying 3 datapoints from the space of the images. The problem of finding the right slices is therefore equivalent to finding the right triples of input images. In practice it turns out, that more often than not the choice of two of the points is obvious and natural. It is usually the third point that is difficult to choose in the meaningful way. A prime example of such situation would be when we have an adversarial perturbation of an image. Then the natural choice of the two points is the original image and its adversarial perturbation, while the choice of the third point is highly non-obvious. This is highlighted in the Figure 2. Two different choices of the third point yield two very different landscapes. Another important example of the importance of the choice of the third point is Figure 3, discussed in more detail in Section 3

We experimented with several approaches. The ones yielding (in our opinion) the most informative visualizations were:

- another adversarial perturbation of the original image obtained either by
  - another type of attack
  - another random seed of the attack
  - attack on the same image, but with respect to a different model
- a uniformly gray image (geometric center of the hypercube)
- a random image
- a random perturbation of the original image of a pre-defined magnitude

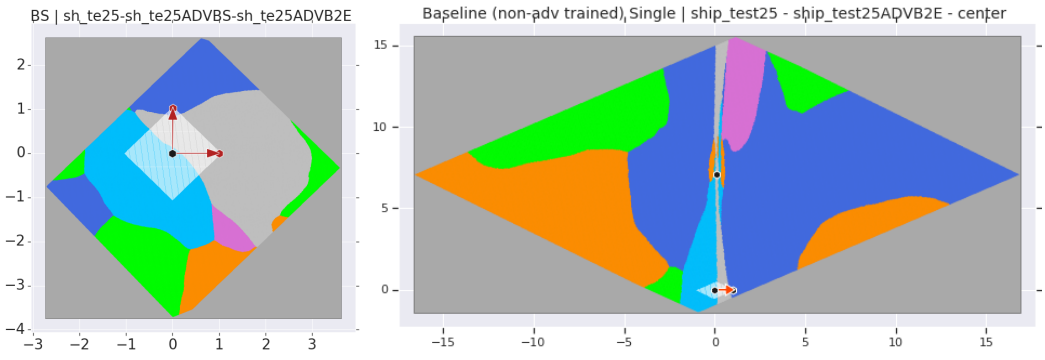

Figure 2: Local versus global perspective. Two slices along the same adversarial perturbation, a "small" slice on the left, versus a "large" slice on the right. Two different choices of the third point lead to two very different images. Note that the pictures are not in the same scale - the left one is strongly magnified.

3 OUR OBSERVATIONS AND FINDINGS

In this section we detail some of our empirical observations about the decision regions of models trained on the CIFAR10 dataset, using our visualization methodology. For more details on the models used, and the training procedure, please refer to the Appendix.

- Choosing a meaningful section (or a slice) of the data space is often not easy - see Figures 10 and 4

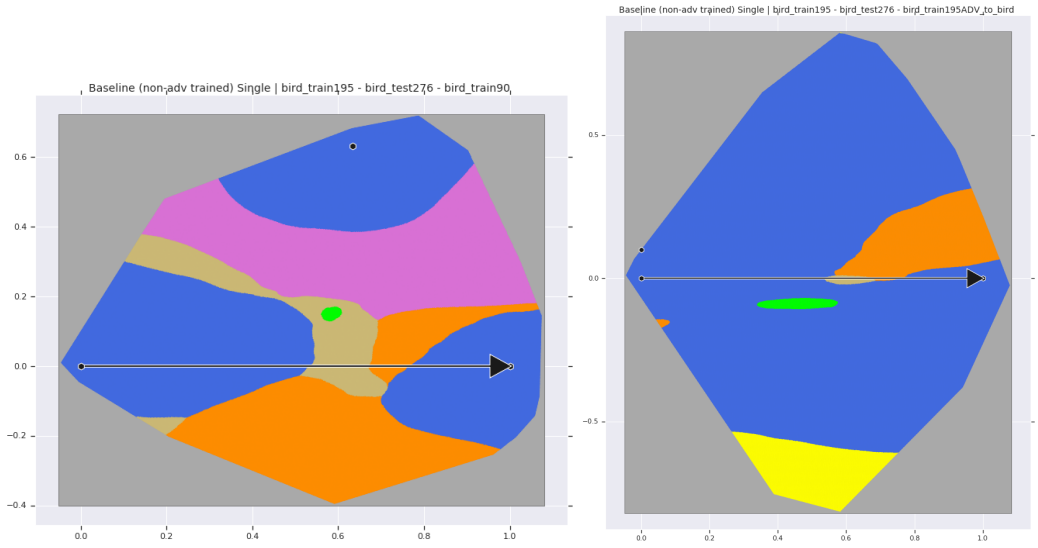

Figure 3: The left slice might give an impression, that the blue (bird) decision region is disconnected, however a different choice of a third point reveals, it is not the case.

- Observed decision boundaries are less complicated than we initially suspected - decision regions generally form large, regular, connected patches.
- Differences in model architectures are reflected in the final shapes of decision regions in a way that is noticeable to the human eye when visualized using our scheme, this can be seen in Figures 6, 8 and 9 in the Appendix.
- Cutting the space in an 'adversarial' direction lead to "small slices"; *i.e.*, the adversarial direction is in all observed cases transversal to the vector connecting the image to the center of the hypercube - see for example Figure 2.
- Shapes of the decision boundaries change very rapidly throughout the entire training process. Essentially, each minibatch update leads to very dramatic changes of observed decision region shapes within a fixed slice. The authors are working on a video showing the animation of the traning process.

In Fawzi et al. (2018) it was shown that the decision regions of deep networks are topologically arcwise-connected sets, which we have verified empirically. An illustration of that phenomenon can be seen in Figure 3. The image on the left might give an incorrect impression that the two images (bird #195 from the training set and bird #276 from the test set) belong to two disconnected components of the decision region of the model. However, choosing a different slice containing these two images reveals the true geometry of the decision set.

REFERENCES

Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *arXiv preprint arXiv:1801.00553*, 2018.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. Accessed: 2018-02-03, 2018. URL `https://arxiv.org/abs/1802.00420`.

Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14. ACM, 2017a.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 39–57. IEEE, 2017b.

Alhussain Fawzi, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, and Stefano Soatto. Empirical study of the topology and geometry of deep networks. In *Computer Vision and Pattern Recognition, 2018. CVPR 2018. IEEE Conference on*, pp. 3762–3770. IEEE, 2018.

Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Edward Grefenstette, Robert Stanforth, Brendan O'Donoghue, Jonathan Uesato, Grzegorz Świrszcz, and Pushmeet Kohli. Strength in numbers: Trading-off robustness and computation via adversarially-trained ensembles. *https://arxiv.org/abs/1811.09300*, 2018.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Jun Zhu, and Xiaolin Hu. Defense against adversarial attacks using high-level representation guided denoiser. *arXiv preprint arXiv:1712.02976*, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519. ACM, 2017.

Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Shane Torbert. *Applied computer science*. Springer, 2016.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *ICLR*, 2018.

Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. In *The 35th International Conference on Machine Learning (ICML)*, 2018.

David Warde-Farley and Ian Goodfellow. Adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics*, pp. 311, 2016.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

Xiaoyong Yuan, Pan He, Qile Zhu, Rajendra Rana Bhat, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *arXiv preprint arXiv:1712.07107*, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

SUPPLEMENTARY MATERIALS FOR "VISUALIZATIONS OF DECISION BOUNDARIES"

A MODELS USED

The main model we used for our analysis on the CIFAR-10 dataset was a Wide ResNet (Zagoruyko & Komodakis, 2016) consisting of a 3×3 convolution layer, followed by three layers containing 28 ResNet blocks of width factor 10, followed by batch normalization (Ioffe & Szegedy, 2015) layer, followed by a ReLU (Nair & Hinton, 2010), and by a final linear layer projecting into the logits of the CIFAR-10 classes. All models we experimented with here are variations on this architecture, namely - the addition of adversarial training (Madry et al., 2017) and model ensembling (Grefenstette et al., 2018). For MNIST dataset we used a standard 3-layer CNN architecture.

B ADVERSARIAL PERTURBATIONS AND ADVERSARIAL TRAINING

Deep neural networks have demonstrated state-of-the-art performance in a wide range of application domains (Krizhevsky et al., 2012). However, researchers have discovered that deep networks are in some sense 'brittle', in that small changes to their inputs can result in wildly different outputs (Huang et al., 2017; Jia & Liang, 2017; Szegedy et al., 2013). For instance, practically imperceptible (to human) modifications to images can result in misclassification of the image with high confidence. Not only are networks susceptible to these 'attacks', but these attacks are also relatively easy to compute using standard optimization techniques (Carlini & Wagner, 2017b; Goodfellow et al., 2014). These changes are often referred to as *adversarial perturbations*, in the sense that an adversary could craft a very small change to the input in order to create an undesirable outcome. This phenomenon is not unique to image classification, nor to particular network architectures, nor to particular training algorithms (Papernot et al., 2016; 2017).

Adversarial attacks can be broken into different categories depending on how much knowledge of the underlying model the adversary has access to. In 'white-box' attacks the adversary has full access to the model, and can perform both forward and backwards passes (though not change the weights or logic of the network) (Carlini & Wagner, 2017a; Goodfellow et al., 2014). In the 'black-box' setting the adversary has no access to the model, but perhaps knows the dataset that the model was trained on (Papernot et al., 2016; 2017). Despite several recent papers demonstrating new defences against adversarial attacks (Akhtar & Mian, 2018; Guo et al., 2017; Liao et al., 2017; Song et al., 2017; Tramèr et al., 2018; Warde-Farley & Goodfellow, 2016; Xie et al., 2017; Yuan et al., 2017), recent papers have demonstrated that most of these new defences are still susceptible to attacks and largely just obfuscate the gradients that the attacker can follow, and that non-gradient based attacks are still effective (Uesato et al., 2018; Athalye et al., 2018).

B.1 ADVERSARIAL ATTACKS AND ADVERSARIAL TRAINING IN MORE DETAIL

Here we lay out the basics of attacking a neural network by the generation of adversarial examples. Denote an input to the network as $x \in \mathbb{R}^d$ with correct label $\hat{y} \in \mathcal{Y} \subset \mathbb{N}$, and let $m_\theta : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{Y}|}$ be the mapping performed by the neural network which is parameterized by $\theta \in \mathbb{R}^p$. Let $L : \mathcal{Y} \times \mathbb{R}^{|\mathcal{Y}|} \to \mathbb{R}$ denote the loss we are trying to minimize (*e.g.*, the cross-entropy). When training a neural network we seek to solve

$$\text{minimize} \quad \mathbb{E}_{(x,y)\sim\mathcal{D}}\, L(\hat{y}, m_\theta(x)) \tag{1}$$

over variable θ, where $\mathcal{D}$ is the data distribution. Given any fixed θ we can generate (untargeted) adversarial inputs by perturbing the input x so as to *maximize* the loss. We restrict ourselves to small perturbations around a nominal input, and we denote by $\mathcal{B}$ this set of allowable inputs. For example, if we restrict ourselves to small perturbations in ℓ_∞ norm around a nominal input x^{nom} then we could set $\mathcal{B} = \{x \mid \|x - x^{\text{nom}}\|_\infty \leq \epsilon\}$ where $\epsilon > 0$ is the tolerance. A common approach for generating adversarial examples is projected gradient descent (Carlini & Wagner, 2016), *i.e.*, to iteratively update the input x by

$$\tilde{x}^{k+1} = \Pi_{\mathcal{B}}(\tilde{x}^k + \eta \nabla_x L(y, m_\theta(\tilde{x}^k))), \tag{2}$$

where typically $x^0 = x + \epsilon$ for some noise ϵ, $\eta > 0$ is a step-size parameter and $\Pi_{\mathcal{B}}$ denotes the Euclidean projection on $\mathcal{B}$. We add noise to the initial point so that the network can't memorize the training dataset and mask or obfuscate the gradients at that point (Uesato et al., 2018; Athalye et al., 2018), in other words the added noise encourages generalization of adversarial robustness to the test dataset. If instead of using the gradient we just use the sign of the gradient then this is the fast-gradient-sign method (Goodfellow et al., 2014). Empirically speaking, for most networks just a few steps of either of these procedures is sufficient to generate an $\tilde{x}$ that is close to x^{nom} but has a different label with high confidence.

B.1.1 ADVERSARIAL TRAINING

In this subsection we briefly explain the concept of *adversarial training* (Madry et al., 2017). In adversarial training we train a network to minimize a weighted sum of two losses (where the relative weighting is a hyper-parameter). The first loss is the standard loss of the problem we are trying to solve on the normal training data, *e.g.*, the cross-entropy for a classification task. The second loss is the same function as the first loss, except evaluated on *adversarially generated data*, where typically the adversarial data is generated by attacking the network at that time-step. In other words we replace the problem in eq. (1) with

$$\text{minimize} \quad \mathbb{E}_{(x,y)\sim\mathcal{D}}(L(\hat{y}, m_\theta(x)) + \rho L(\hat{y}, m_\theta(\tilde{x}))) \tag{3}$$

where $\rho \geq 0$ is the weighting parameter and $\tilde{x}$ is an adversarial example generated from x at model parameters θ using, for example, the update in eq. (2). This problem is usually approximated by sampling and minimizing the empirical expectation.

For the effect of adversarial training on the appearance of the decision regions of the model see Figures 5 and 6

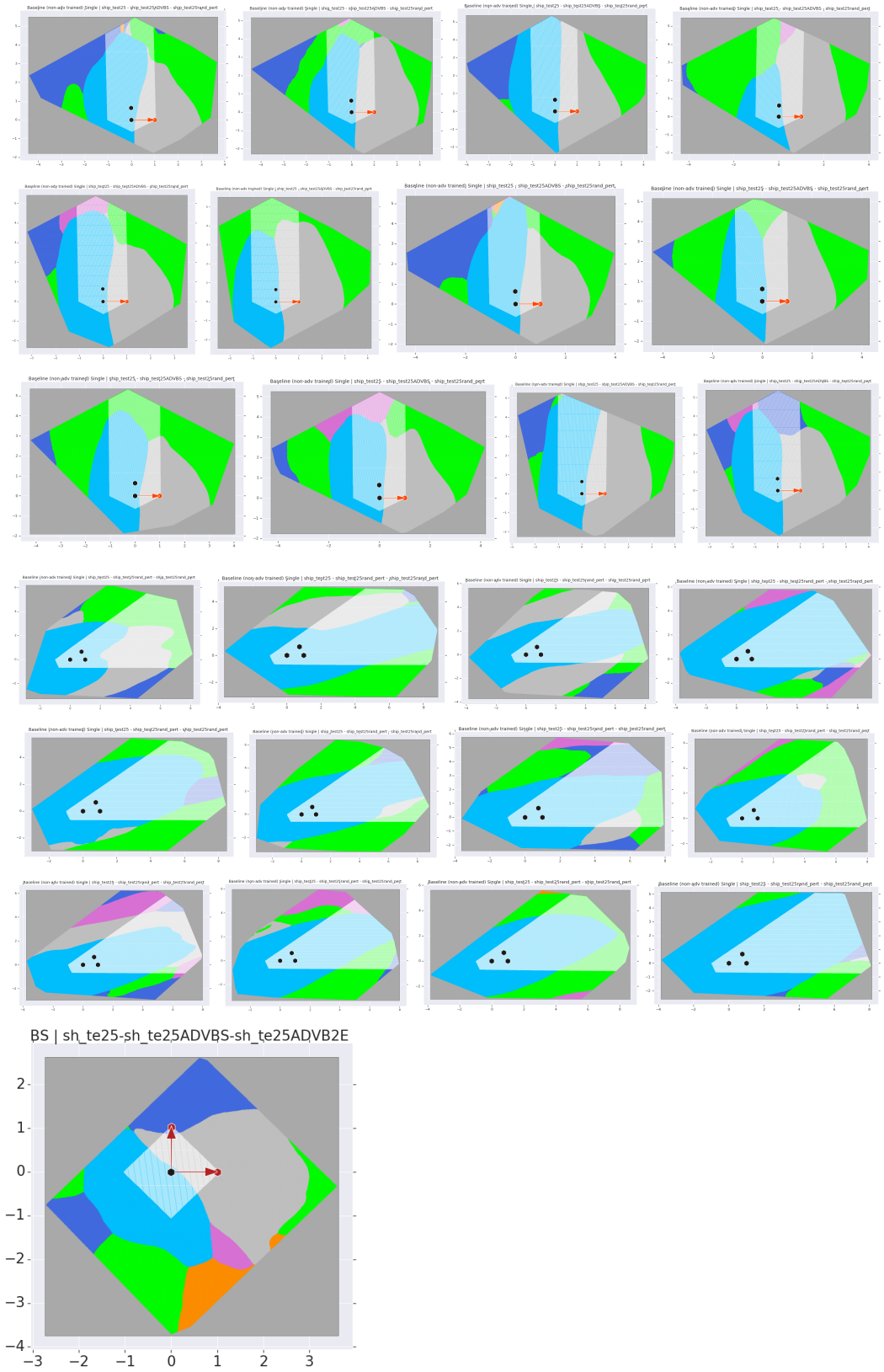

Figure 4: Church window plots - slices through one random perturbation and one adversarial modification (upper 12), two random perturbations (middle 12), versus targeted slice (bottom)

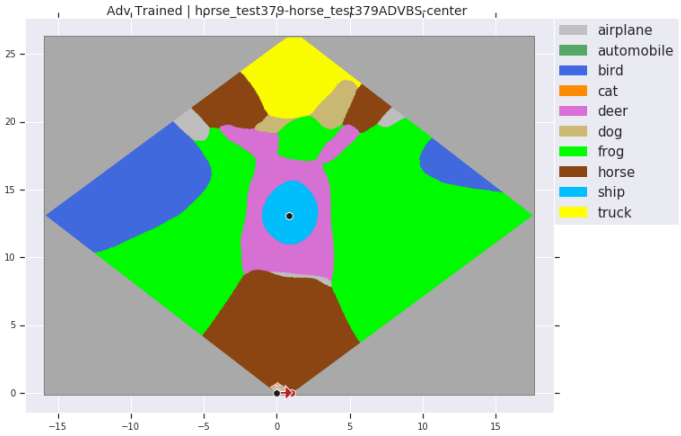

Figure 5: Decision boundaries of an adversarially trained model (see Section B.1.1)

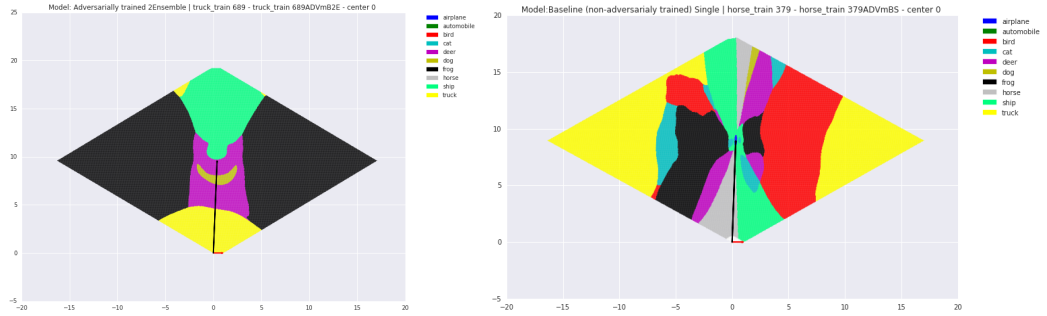

Figure 6: Decision regions of an adversarially trained model (left) vs. non-adversarially trained baseline (right).

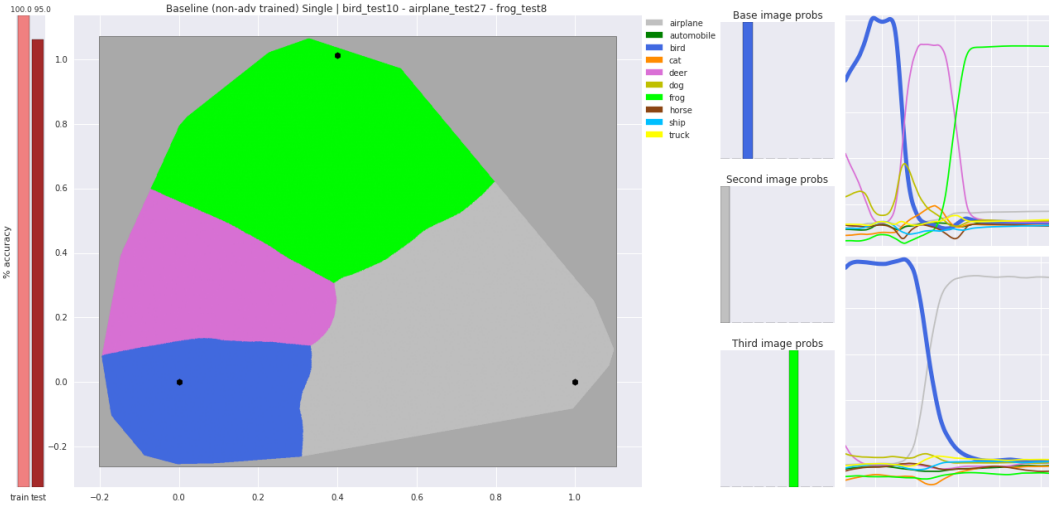

Figure 7: The displaying tool with extra features enabled. The values of probits and graphs of probits along the lines connecting three points of interest are shown.

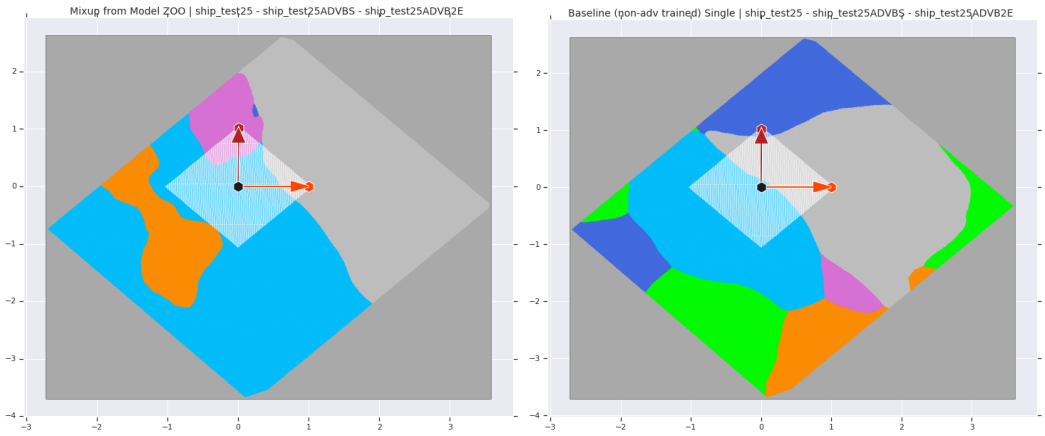

Figure 8: Baseline versus Mixup models

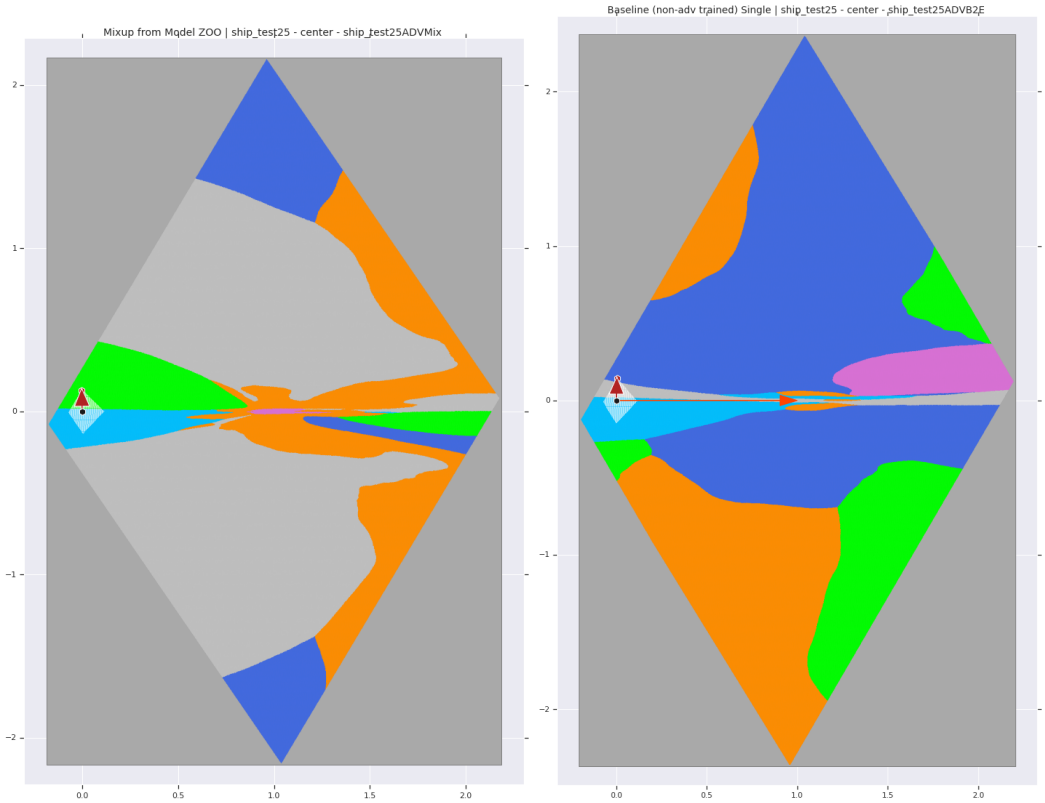

Figure 9: Baseline versus Mixup models - center cut

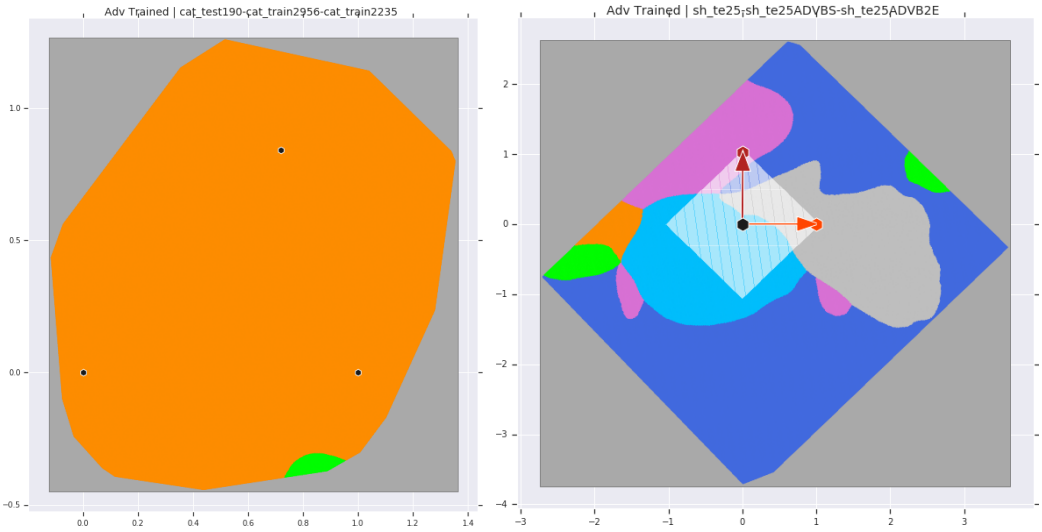

Figure 10: The difference between ”any” (left) and a ”meaningful slice (right).

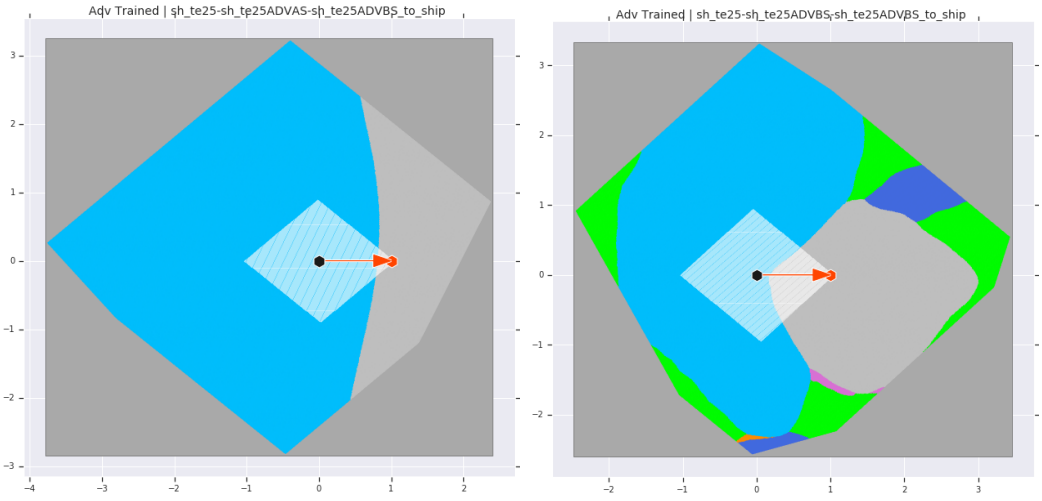

Figure 11: Example of a slice of a CIFAR-10 image space. Four different regions regions corresponding to ”airplane”, ”bird”, ”deer” and ”frog” are visible.