The Dimpled Manifold Revisited

Lukas Karner
Supervisors: Prof. Moritz Grosse-Wentrup, Prof. Sebastian Tschiatschek

March 4, 2023

Abstract

Recently Shamir et.al. proposed the Dimpled Manifold Model [17] as new explanation for the phenomenon of adversarial examples in image classification using neural network classifiers. The present paper successfully reproduces several key results of the paper introducing the Dimpled Manifold Model and provides critical reflection on its methods and arguments. Motivated by the geometric hypothesis of the Dimpled Manifold Model further this paper proposes a new defence strategy against adversarial attacks based on the Isometric Autoencoder introduced by Gropp et.al. in [4] and describes an attempt of implementing the new defence and the difficulties encountered in this process.

1 Introduction

So called *adversarial examples*, that is small perturbations added to images that cause neural network classifiers to miss-classify them, have been an object of intense study in deep learning research since they were discovered in 2014 [3]. Since then many explanations for this phenomenon have been proposed, but none of them provide satisfying explanations so far. Two noteworthy contributions in this in context are for example the Non-Robust Features Hypothesis [7] and the Boundary Tilting Hypothesis [19]. The Non-Robust Features Hypothesis essentially argues that images contain some features that are robust (those that humans use to visually identify the content of the image) and others that are not robust, and that adversarial examples are a consequence of the existence of the non robust features. The Boundary Tilting Hypothesis on the other hand argues that adversarial examples are a consequence of the hypothesised fact that the decision boundary of neural network classifiers tends to intersect the image manifold in a very acute angle and that the attacks can be mitigated by increasing this angle. Recently a new attempt at explaining adversarial examples, the so called *Dimpled Manifold Model*, was proposed by Shamir et.al. in [17]. This hypothesis argues that the origin of adversarial examples lies in the shape that the decision boundary of a neural network classifier attains when it is trained on image data. Since typically images are defined in very high dimensional vector spaces, but obviously the vast majority of the elements of these spaces are not natural images, it is a common assumption in machine learning [19] that natural images are located on a so called *image manifold* of much lower dimension that is embedded in the high dimensional spaces. Now the Dimpled Manifold Model argues that the training of a neural

classifier on image data consists of two phases: In the first phase the decision boundary approximates the image manifold, and then in the second phase the decision boundary develops "dimples" which deviate away from the image manifold in the locations of the training data in directions such that the images are classified correctly. From this point of view adversarial examples are a consequence of the hypothesis that the decision boundary does not leave the image manifold but rather extends along it, and hence one just needs to make a tiny step in the correct direction to reach across the decision boundary and change the classification of an image.

This paper has two objectives: The first objective is to reproduce several results that were used in [17] to argue in favour of the Dimpled Manifold Model and to reflect critically on them. This is done in the following sections 2 and 3. The second objective is to conceive a new defence against adversarial attacks motivated by the Dimpled Manifold Model and to investigate its applicability. This is described in section 4.

2 Related Literature

The recently published pre-print [21] also attempts to reproduce the results of [17] and conducts rigorous statistical analyses of the claims made there. Based on their measurements the authors come to the conclusion that the claims made in [17] do not hold in general settings and hence the Dimpled Manifold Model should be rejected.

3 Reproducing Original Results

This section describes the process and results of attempting to reproduce the adversarial attacks and analyses conducted in [17]. In particular these results include the so called on/off manifold attacks that were used to argue in favour of the Dimpled Manifold Hypothesis. In order to do this we need several things: For each of the 3 datasets MNIST [11], CIFAR10 [9], and ImageNet[1] [16] we need to train a classifier and an autoencoder in the way described in the original paper and ideally achieving similar performance levels. Further, for the adversarial attacks, we need a working implementation of a PGD attack [14], which is the attack method that was also used in [17]. And finally in order to not only conduct standard adversarial attacks we also need a function to compute orthogonal projections on the image manifold at given locations. All the implementations were made using PyTorch [15].

3.1 Classifiers

The MNIST classifier uses the architecture that is reported in appendix G of [17], i.e. two layers of size 256 with ReLU activation and dropout probability 0.2, and a linear output layer. The image data were normalised before training. The model was trained for 40 epochs with SGD using a batch size of 200, cross entropy loss, a learning rate of 0.01, and weight decay of 0.0001. The model reached a training accuracy of 96.85% and a test

[1] Like in the original paper only the two classes "great white shark" and "goldfish" were used for training the autoencoder.

accuracy of 96.35%, which is a bit short of the 98% accuracy claimed by [17] but still an acceptable level.

The CIFAR10 classifier also uses the architecture reported in appendix G of [17], i.e. a convolutional architecture with 7 convolutional layers, 128 channels in the first layer, BatchNorm, ReLU activation, and MaxPooling. While [17] reportedly used pre-trained weights for this model and it was possible to load these weights into the given architecture, they only yielded a test accuracy of 10% and hence I decided to train the model on my own. For the training the image data were normalised and horizontally flipped with probability 0.5. The model was trained for 150 epochs with Adam [8] using a batch size of 200, cross entropy loss, and a learning rate of 0.001 that was reduced by a factor of 10 after 80 and 120 epochs. The model reached a training accuracy of 100% and a test accuracy of 93.4%, hence matching the 93% reported in [17].

For ImageNet the implementation and pre-trained weights of the ResNet50 [6] architecture provided by PyTorch were used. The preprocessing was done according to the instructions provided by PyTorch and the model achieves a test accuracy of 80.858%.

3.2 Autoencoders

The MNIST autoencoder uses the architecture provided in [4], i.e. a convolutional autoencoder consisting of 4 convolutional layers with BatchNorm and Softplus activation and one linear layer in the encoder and decoder each, the latent space has 16 dimensions. The data were not normalised before training. The model was trained for 1000 epochs with Adam using a batch size of 144, mean squared error loss, and a learning rate of 0.001. The model achieved a training loss of 0.00017 and a test loss of 0.007 which corresponds to a very good visual performance with only minor deviations (see figure 1 in appendix A). The paper [17] claims a (presumably test) loss of 0.00003 for this model, which seems unreasonable.

At first I attempted to train the CIFAR10 autoencoder according to the instructions given in [17], however several problems occurred there. The authors used a very large architecture for this model (in fact it is the same architecture that is used for ImageNet), the reported latent dimension was off by a factor of 4 from the actual value, and there were no instructions for the preprocessing of the data at all. Consequentially all attempts to train this model failed, the trained models would show a moderate visual performance on the training data and be completely useless on the test data. Luckily also the paper [4] contained instructions for training an autoencoder on CIFAR10 which I then adopted. The architecture used here was again a convolutional autoencoder consisting of 4 convolutional layers with BatchNorm and Softplus activation and one linear layer in each the encoder and decoder, the latent space dimension is 128. The data were not normalised before training. The model was trained for 400 epochs with Adam using a batch size of 32, mean squared error loss, and a learning rate of 0.001. The model achieved a training loss of 0.0003 and a test loss of 0.0079 which corresponds to a very good performance with only minor deviations on the training data and and a moderate performance with significant deviations on the test data (see figure 2 in appendix A).

The architecture used for the ImageNet autoencoder is based on the VGG16 [18] model and each the encoder and the decoder consist of 13 convolutional layers with BatchNorm and ReLU activation and MaxPooling/Unpooling layers in between them. The latent space dimension of this architecture is 25088, however the paper reported the dimension to be 3584 which is off by a factor of 7. The data were not normalised before training. The model was trained for 325 epochs with Adam using a batch size of 32, mean squared error loss, and an initial learning rate of 0.0001 that was multiplied by a factor of 0.8 whenever the training loss did not decrease for 7 epochs. The model achieved a training loss of 0.0018 and a test loss of 0.0043 which corresponds to output images that do not have significant deviations from the input but lack a lot of detail (see figure 3 in appendix A).

3.3 Attack Function

In the original paper the advertorch package [2] was used to conduct the adversarial attacks. However this package does not support recent versions of PyTorch and hence I implemented the PGD attack [14] myself. This worked without further complications.

3.4 Manifold Projection

In order to compute the orthogonal projection of the image space onto the linear approximation of the manifold at a given image at first I compute the derivative of the output of the autoencoder with respect to the latent variables. Then the columns of this matrix span the tangential space to the manifold at the given location and in order to obtain an orthogonal projection this matrix has to be orthogonalised, which I do using the QR-decomposition. One problem that I encountered here was that for the case of the 25088 dimensional manifold defined by the ImageNet autoencoder the tangential matrix would be 14 GB large. Since all of the common Python packages' implementations of the QR-decomposition are not in-place, they all required 42 GB of memory in order to compute these QR-decompositions. Since however I did not have so much memory I had to implement an in-place function to compute the QR-decomposition. Finally also this was successful and so we collected all the building blocks that we needed.

3.5 Results

In appendix B one can see several on/off manifold attacks that were reproduced according to [17]. One of the key arguments that was used there to argue in favour of the Dimpled Manifold Model was the phenomenon that on manifold attacks require significantly larger perturbations to be successful than off manifold attacks, and that the perturbation sizes of the off manifold attacks are similar to those of the standard attacks. Based on the observed reproduced results this observation cannot be rejected. Further it was also argued that very often the perturbations of the on manifold attacks show recognisable features of the incorrectly predicted class, and this was interpreted as evidence that the on manifold perturbations actually point towards the other classes along the manifold. Judging by our obtained results this claim however seems rather implausible and no obvious instance

of this phenomenon could be observed for any example of the CIFAR10 or ImageNet datasets. In the case of MNIST sometimes it might seem like perturbations show actually features of the other class, but this might very well be just due to the rather simple structure of the data, and still it is not at all obvious to guess the class just from seeing the perturbation.

4 Leveraging Geometric Insights

Based on the observation that the perturbations of adversarial attacks look like random noise in the past many image denoising methods have been proposed as defence against adversarial attacks. Among those some of the most notable are: Denoising Autoencoders [5] which essentially are autoencoders that are trained with noise added to the input images, JPEG compression [1], Feature Squeezing [20] which uses filters and bit depth reduction, and the High-Level Representation Guided Denoiser (HGD) [12] which is a very sophisticated autoencoder architecture that learns to remove adversarial noise from images and was the winner of the 2017 NIPS contest [10]. For a more extensive overview of existing defences see [13]. However, all of these methods have their advantages and disadvantages and are still vulnerable to adaptive adversaries. So supposing the Dimpled Manifold Model is indeed true, one can wonder whether one can use this geometric insight into the nature of adversarial examples in order to conceive a universally applicable and robust defence against them.

4.1 Denoising Orthogonally

Essentially the Dimpled Manifold Model says that adversarial examples use the dimensions that are orthogonal to the manifold on which the data is located. Now supposing that the image manifold were a linear subspace of the image space it would be very easy indeed to exploit the orthogonality of the adversarial examples. One would just have to apply PCA with a suitable number of principal components (i.e. the dimension of the manifold) to the data and then either include an orthogonal projection on the manifold in the preprocessing of the data or apply the classifier directly on the principal components. In both cases the decision boundary in the original image space would intersect the image manifold orthogonally and extend linearly in all to the manifold orthogonal dimensions. This would indeed be a universally applicable and robust defence against adversarial examples because no matter the size of an orthogonal perturbation, it would never cause the model to misclassify the input. Every perturbation that could actually cause the input to be misclassified would have to be so large and point in the right direction such that the perturbed image would actually be closer to the images of the wrongly predicted class than to the original class, which means that no adversarial examples in the classical sense exist in this scenario. So we can solve our problem in the case of data that is located on a linear subspace, but what about general manifolds? The obvious extension of PCA to non linear manifolds is kernel PCA. But as with kernel methods in general its complexity scales quadratically with the number of data points, and hence it is not suitable for application in the context of large scale image classification. The state of the art tool to map data from a high dimensional native space to a lower dimensional latent space and hence to learn the data manifold are autoencoders, and as we have discussed just before many

attempts have been made to use them as defence mechanisms. However we would not only like to have an autoencoder that learns the manifold, but one that also maps points off the manifold orthogonally onto it, which leads us to the next section.

4.2 The Isometric Autoencoder

The Isometric Autoencoder (I-AE) was introduced in [4] and is a regularisation method for autoencoders that enforces the decoder to be an isometry, i.e. a map that maintains distances, and the encoder to be the decoder's pseudo-inverse, i.e. the map that sends any point that is not in the image of the decoder to the same value as the closest point to it that is in the image of the decoder. This however means that at least locally around the manifold the I-AE acts as an orthogonal projection onto the manifold, which is exactly what we wanted. Nonetheless one should note that all theoretical results about the I-AE only hold at the training data and hence there exist no guarantees for the behaviour of it at other data points or even away from the data manifold. Apart from the desired properties the I-AE also has some other advantages over the other methods discussed at the beginning of this section: First of all the I-AE is mathematically proven to work (n.b. locally around the training data) and does not at all rely on heuristics like the Denoising Autoencoder or the HGD. Further the I-AE does not have a fixed architecture like the HGD but is merely a regularisation methods that can be applied to any kind of autoencoder architecture. And finally the I-AE also allows to work with the latent variables directly, unlike the HGD that only allows to work in the native image space.

4.3 Implementation

Since the regularisation terms of the I-AE involve computations on the input gradients of the encoder and decoder, the results of which is then again backpropagated to the weights of the autoencoder, the implementation of the I-AE is not trivial. Nonetheless these kinds of computations are supported by the functorch package that comes with PyTorch and which was subsequently used for the implementation. That the implementation was successful could be verified by comparing the gradients that it produced in toy examples to gradients that were manually computed. After this the next step would have been to try to train the I-AE on actual data, however several attempts to do so were not successful and since the time budget of this project was running out the experiments were stopped at this point.

5 Conclusion

While the main results of the paper [17] could be successfully reproduced, this does in no way confirm the Dimpled Manifold Hypothesis and several aspects of it remain to be scrutinised. First of all the arguments used in [17] are not very rigorous as they just rely on reported visual observations of results, but not on statistical analyses of them. This is especially evident as [21] claims to have found statistically significant evidence against the Dimpled Manifold Hypothesis. Further [17] does not use rigorous mathematical methods to examine the orthogonality of the adversarial examples, but again rather relies on visual observations of it. This is problematic since in very high dimensional spaces, like the

ones we are dealing with here, orthogonality is not some kind of unusual phenomenon, but rather expected and hence nothing to be especially surprised about. Lastly it remains to criticise the inconsistencies in the reported details of the models and their training, especially the actual (not the reported) latent space sizes of the autoencoders used for CIFAR10 and ImageNet seem to be inadequately large for their purpose. All in all this means that at this point one cannot draw final conclusions about the correctness of the Dimpled Manifold Model and more systematic effort is necessary to answer this question.

Regarding the Isometric Autoencoder it should be noted that it was possible to implement the regularisation terms and, allowing it the benefit of doubt, the successful training of it might have been possible with more time. Hence it remains an open question whether in practice it can be used as defence against adversarial attacks or not.

References

[1] Nilaksh Das et al. "Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression". In: *arXiv preprint arXiv:1705.02900* (2017).

[2] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. "AdverTorch v0.1: An Adversarial Robustness Toolbox based on PyTorch". In: *arXiv preprint arXiv:1902.07623* (2019).

[3] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[4] Amos Gropp, Matan Atzmon, and Yaron Lipman. "Isometric autoencoders". In: *arXiv preprint arXiv:2006.09289* (2020).

[5] Shixiang Gu and Luca Rigazio. "Towards deep neural network architectures robust to adversarial examples". In: *arXiv preprint arXiv:1412.5068* (2014).

[6] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[7] Andrew Ilyas et al. "Adversarial examples are not bugs, they are features". In: *Advances in neural information processing systems* 32 (2019).

[8] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[9] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[10] Alexey Kurakin et al. "Adversarial attacks and defences competition". In: *The NIPS'17 Competition: Building Intelligent Systems*. Springer, 2018, pp. 195–231.

[11] Yan LeCun, Corinna Cortes, and Christopher J.C. Burges. *The MNIST Database of Handwritten Digits*. URL: `http://yann.lecun.com/exdb/mnist/`.

[12] Fangzhou Liao et al. "Defense against adversarial attacks using high-level representation guided denoiser". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1778–1787.

[13] Gabriel Resende Machado, Eugênio Silva, and Ronaldo Ribeiro Goldschmidt. "Adversarial machine learning in image classification: A survey toward the defender's perspective". In: *ACM Computing Surveys (CSUR)* 55.1 (2021), pp. 1–38.

[14] Aleksander Madry et al. "Towards deep learning models resistant to adversarial attacks". In: *arXiv preprint arXiv:1706.06083* (2017).

[15] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[16] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[17] Adi Shamir, Odelia Melamed, and Oriel BenShmuel. "The dimpled manifold model of adversarial examples in machine learning". In: *arXiv preprint arXiv:2106.10151* (2021).

[18] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

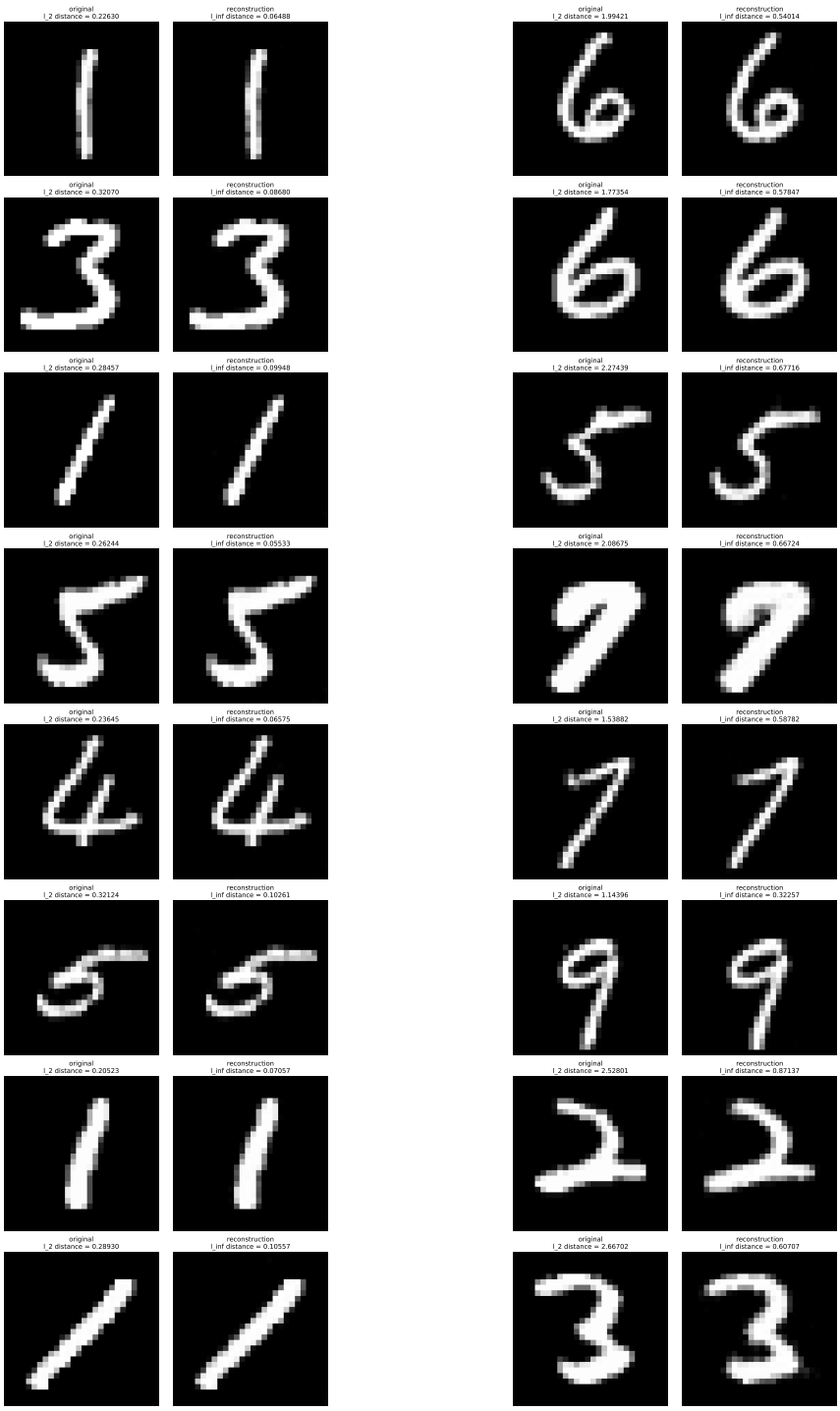
[19] Thomas Tanay and Lewis Griffin. "A boundary tilting persepective on the phenomenon of adversarial examples". In: *arXiv preprint arXiv:1608.07690* (2016).

[20] Weilin Xu, David Evans, and Yanjun Qi. "Feature squeezing: Detecting adversarial examples in deep neural networks". In: *arXiv preprint arXiv:1704.01155* (2017).

[21] William Zhao and Subha Nawer Pushpita. "Extensions on The Dimpled Manifold Hypothesis".

A Autoencoder Results

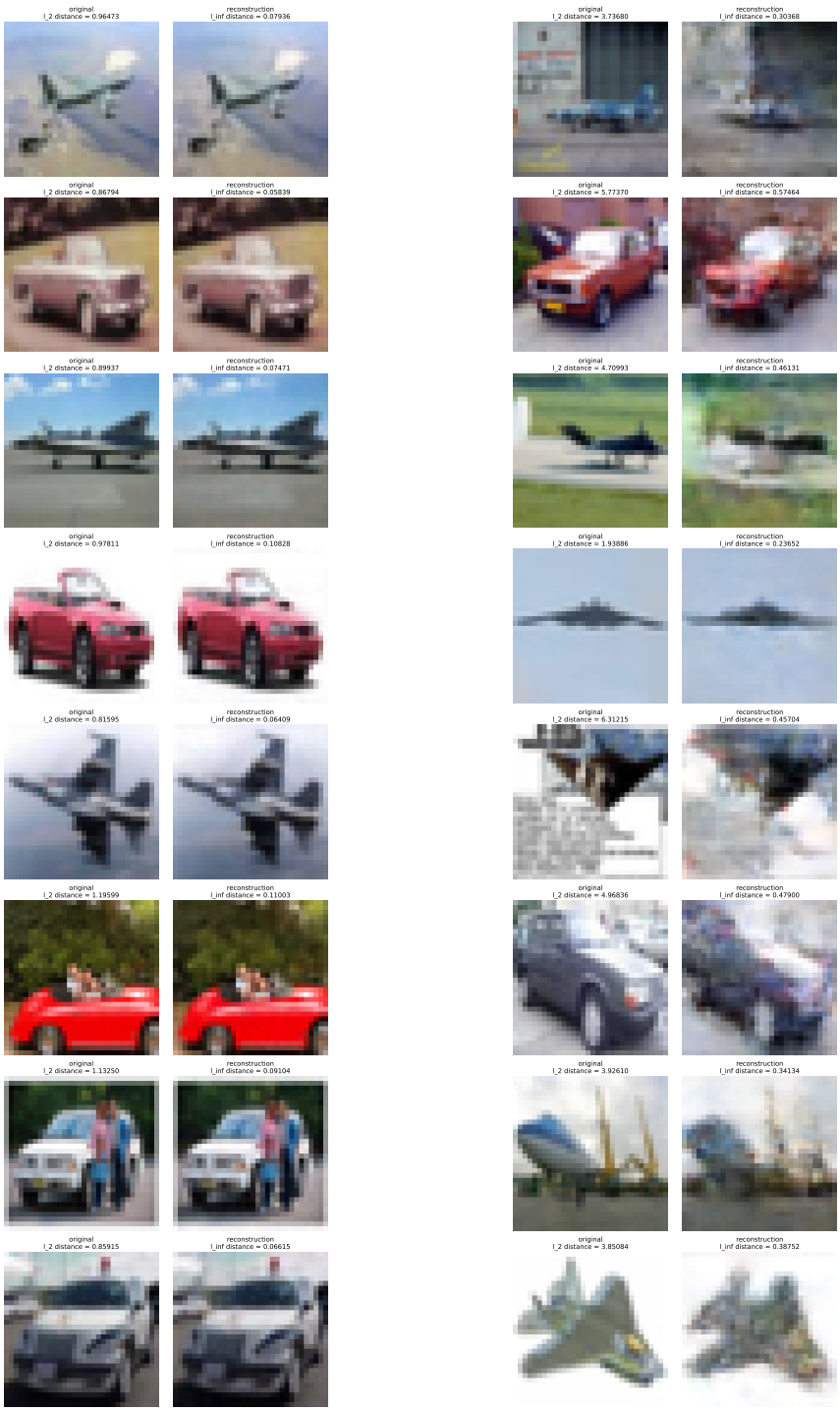
A.1 MNIST



(a) Training Data. (b) Test Data.

Figure 1: Examples of the performance of the MNIST autoencoder.

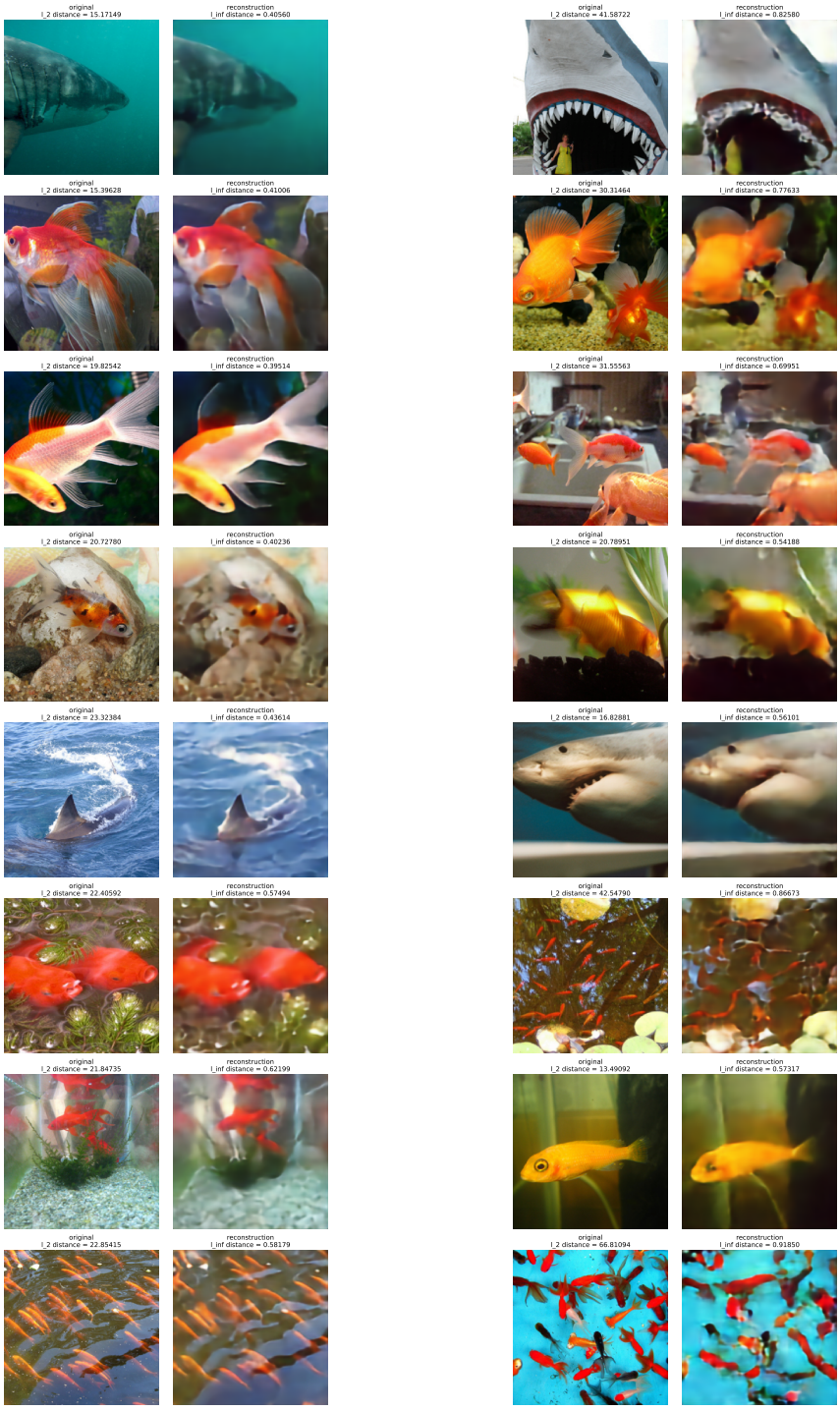
A.2 CIFAR10



(a) Training Data. (b) Test Data.

Figure 2: Examples of the performance of the CIFAR10 autoencoder.

A.3 ImageNet



(a) Training Data. (b) Test Data.

Figure 3: Examples of the performance of the ImageNet autoencoder.

B Adversarial Attacks

For each of the following plots the top row shows a standard adversarial attack, the middle row shows an on manifold attack, and the bottom row shows an off manifold attack.

B.1 MNIST

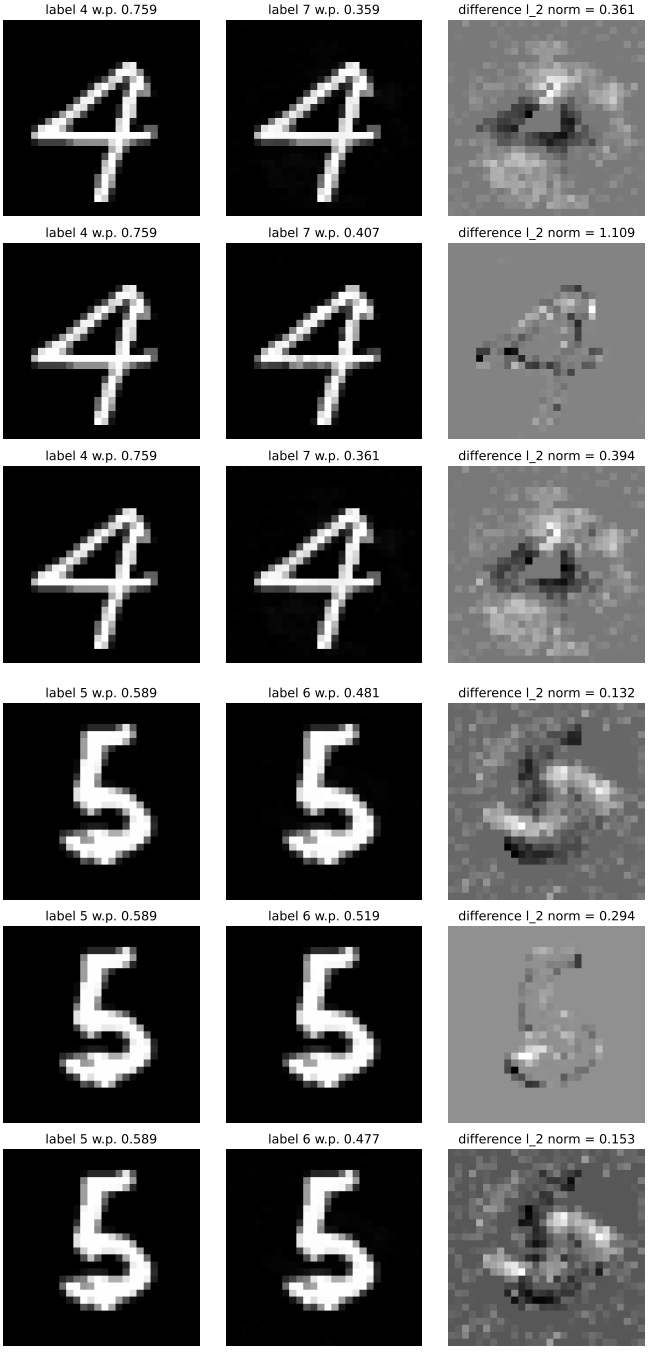


Figure 4: On and off manifold attacks on MNIST training data.

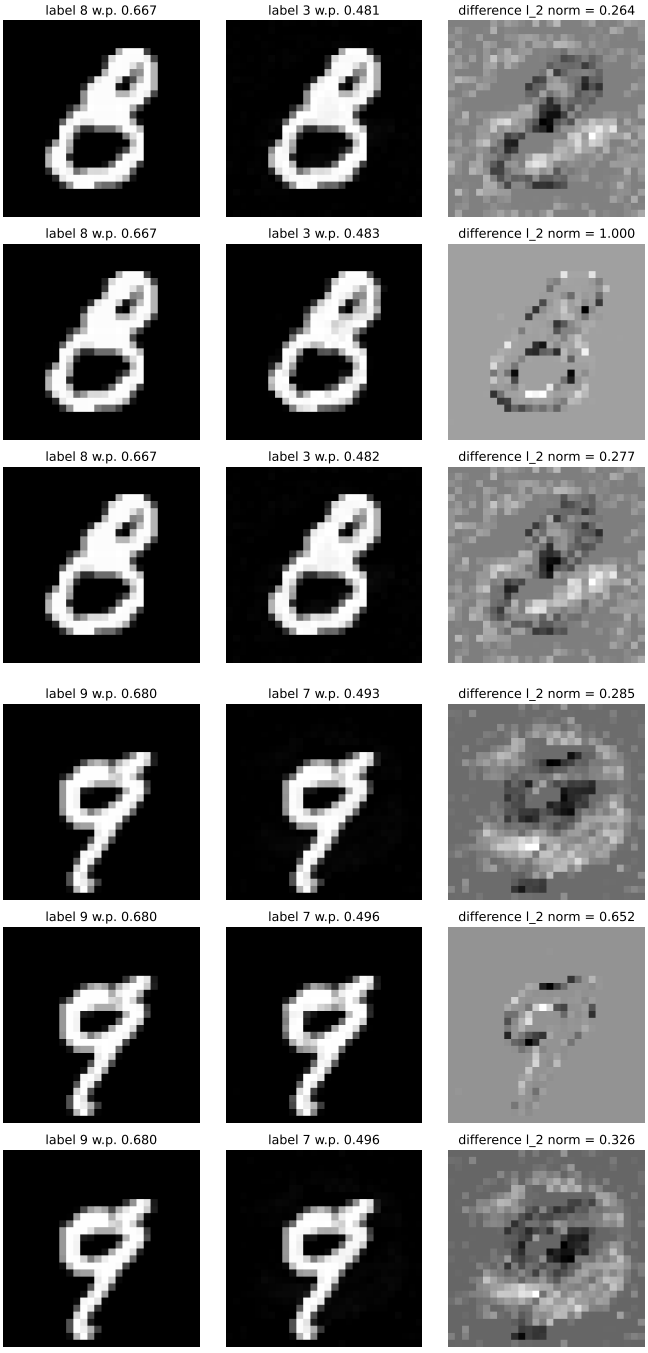


Figure 5: On and off manifold attacks on MNIST test data.

B.2 CIFAR10

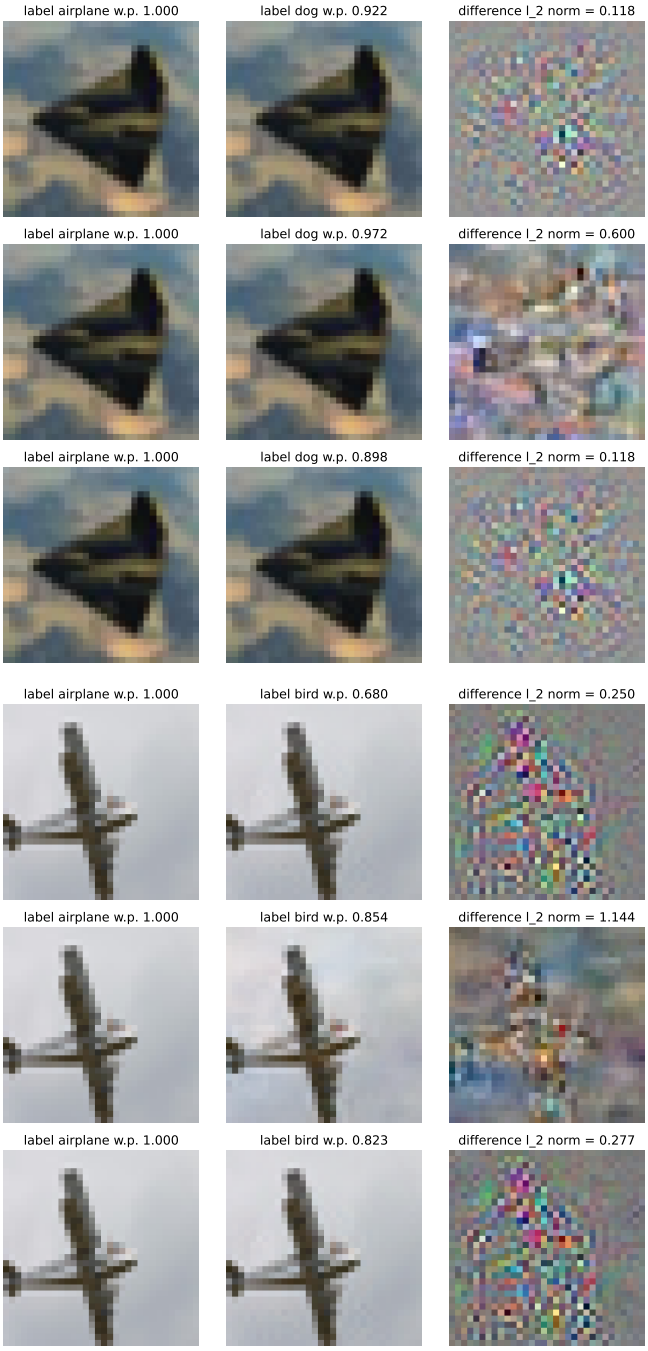


Figure 6: On and off manifold attacks on CIFAR10 training data.

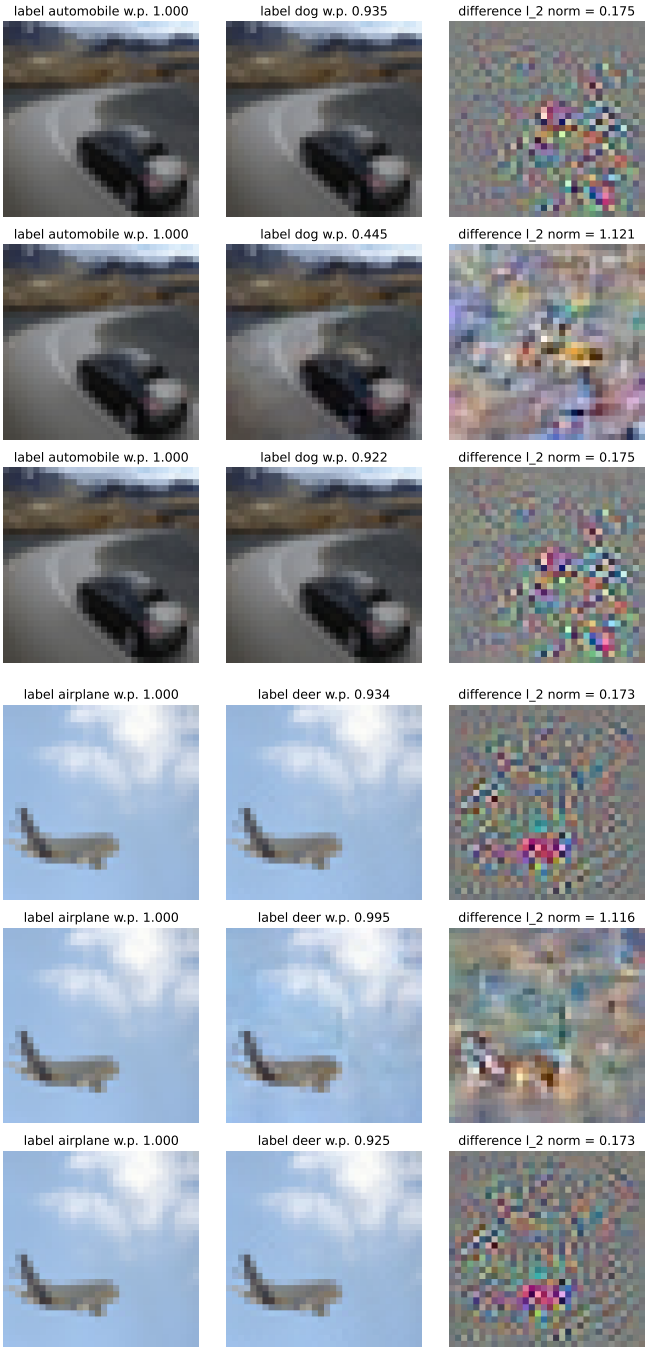


Figure 7: On and off manifold attacks on CIFAR10 test data.

B.3 ImageNet

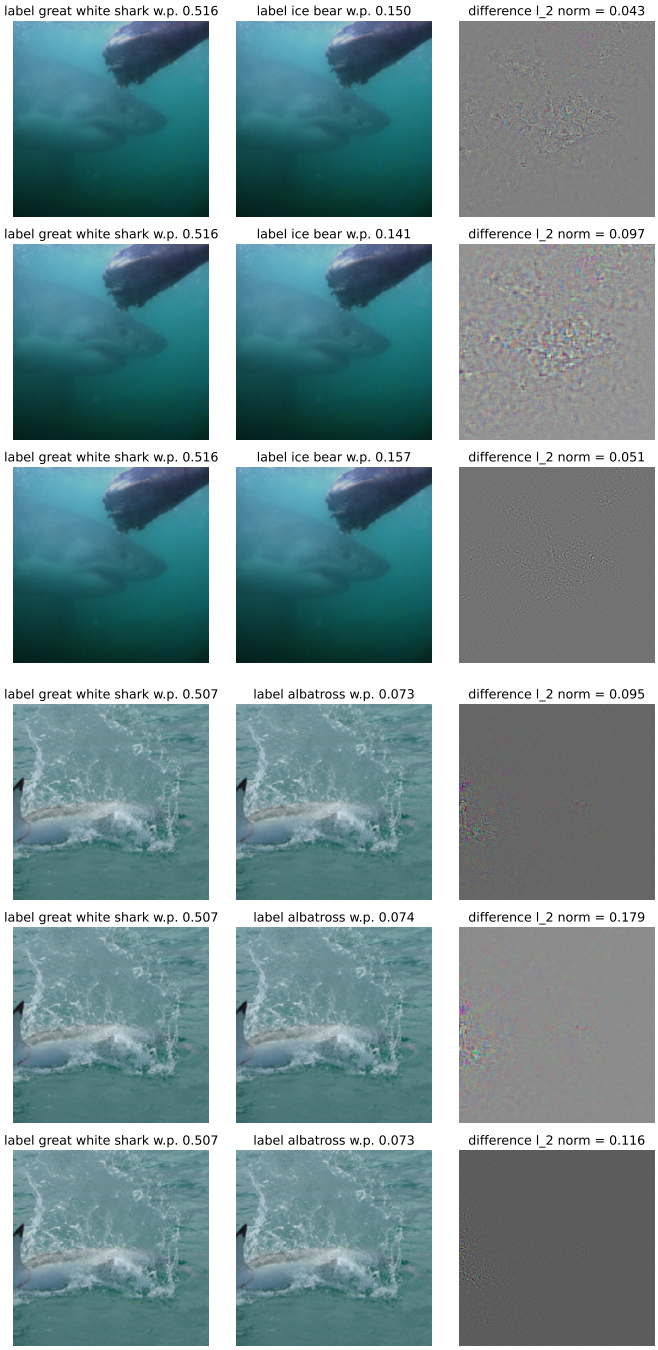


Figure 8: On and off manifold attacks on ImageNet training data.

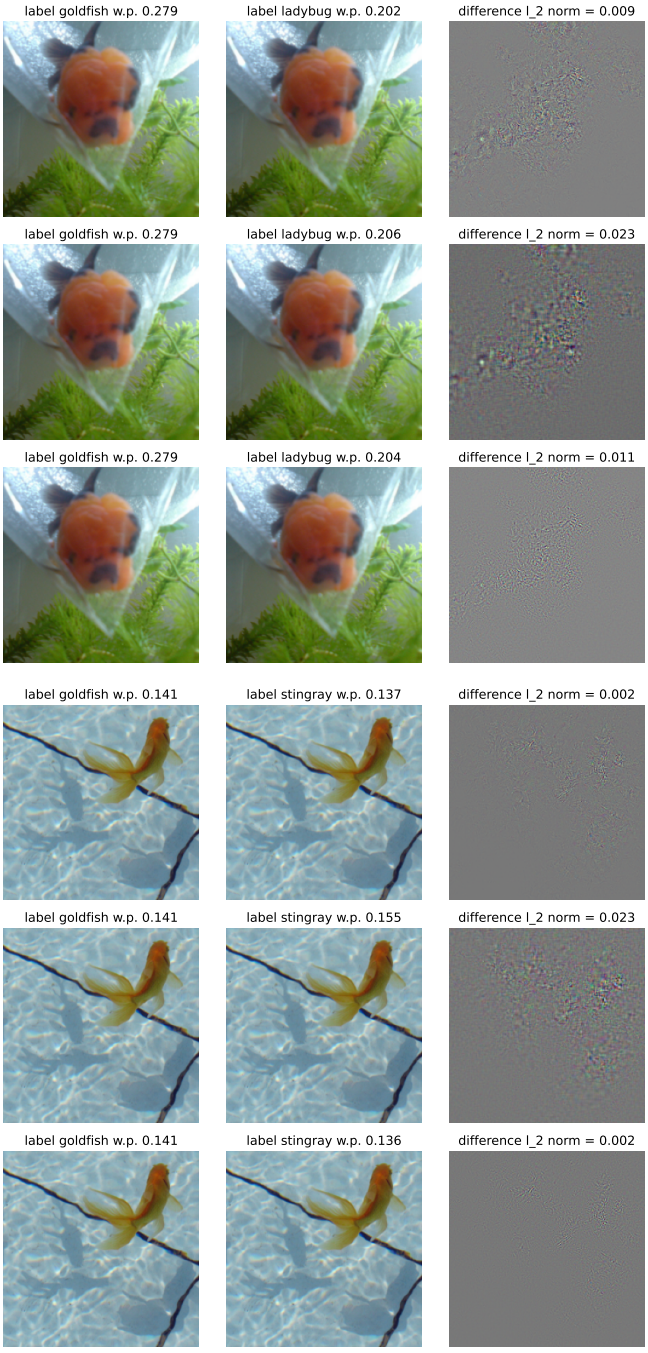


Figure 9: On and off manifold attacks on ImageNet test data.