Machine Learning Approach to Characterizing the Sensitivity of Machine Learning Algorithms to Adversarial Data

Mike Bahns
Department of Electrical and Computer Engineering
University of Arizona
mbahns@email.arizona.edu

George Rosier
Department of Electrical and Computer Engineering
University of Arizona
gmrosier@email.arizona.edu

Abstract

Today, machine learning algorithms are being broadly deployed in a wide range of applications. As society's reliance on these systems becomes pervasive, it is increasingly important to understand the limitations of these systems. This project describes the use of a machine learning system capable of generating adversarial data to help quantify the sensitivity of machine learning algorithms to adversarial data. Although the knowledge of these sensitivities aids both respectable and nefarious enterprises, on balance, a greater awareness of this information benefits all of us, who are increasingly and unknowingly reliant on such systems every day.

1 Introduction

1.1 Background

A confluence of new technologies and new motivations within the past 2 decades have greatly increased the pace of research and investment in machine learning. Critical advancements in compute efficiency in the form of Graphics Processing Units (GPUs), have made machine learning practical and efficient. The widespread collection of vast quantities of information is now possible due to a rapid decrease in data storage costs provided by high-density magnetic storage systems. The collection and use of this data would not be possible without high bandwidth networks which now span the continents. In combination with the recent technological capability for machine learning, there is a complimentary motivation to deploy machine learning, driven by the widening gap between the vast collection of information and the frustration at being unable to benefit from it due to the limitations of the human mind, unchanged from the hunter-gatherer era. The futurist projections for machine learning range from solving all of humanity's problems to the complete destruction or subjugation of humanity by artificial intelligence. This study, however, will focus on the current limitations of machine learning with the cautionary subtext that just because the machines are learning to think, doesn't mean the humans can stop.

1.2 Machine Learning

Machine Learning algorithms are based on statistical techniques which simulate 'learning' (performance improvement over time). The computer learns through a training process where general-

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

purpose algorithms assimilate a large set of input data and learn to correlate those inputs with a desired output through some feedback mechanism. The result is not a perfect mathematical representation of reality, but rather a probabilistic one – limited by the accuracy and completeness of the training data and the capacity of the machine learning system.

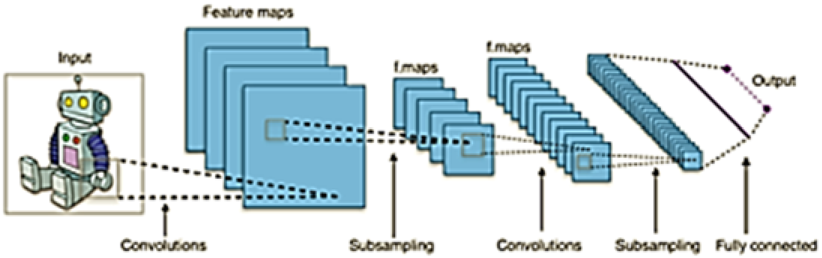


Figure 1: Convolutional Neural Net

1.3 Neural Net Classifiers

Artificial Neural Networks are a machine learning algorithm loosely based on the structure of the brain. Inputs are aggregated with different weightings by a set of neurons, which in turn feed additional layers of neurons. In this way, an output vector is gradually computed by layers of neurons as a function of the weight vectors and activation functions applied to each neuron output. Training involves computing the weight vectors iteratively based on back-propagating the output error, layer by layer, using gradient descent to iteratively calculate improved weight vectors. A convolutional neural network depicted in Figure 1, based loosely on biological vision processing, achieves some economy by sharing weight vectors across neurons and reducing the number of connections between neurons by using smaller convolution and sub-sampling layers. Convolutional neural networks have proven to be an efficient image classification system.

1.4 Generative Adversarial Networks (GAN)

A network can be discriminative (a classifier), for example if configured to determine the subject (class) of an image. A network can also be configured to be generative, for example it can learn how to generate a representative example of a class. The training process of the generative network can be completed with a trained discriminative network to provide the training feedback to the generative network, so that it can learn to improve its synthesized class outputs. The discriminative and generative networks are trained in this adversarial mode to a point where the discriminative network is no longer able to distinguish between real data and the synthesized data created by the GAN.

2 Approach

The goal of this project is to study the sensitivity of machine learning classifiers using machine learning. There are two primary cases to consider: how difficult is it to force the classifier to select a specific class (false-positive, a ’costume’, e.g. Figure 15); and how difficult is it to obfuscate the true class of an input from the classifier (false-negative, a disguise, e.g. Figure 16). The emphasis here is to quantify the sensitivity to techniques that could realistically be applied in real life to confuse an unknown classifier running on live video. An example would be the application of a sticker. An example of a technique that is not useful in reality is one that requires precise manipulation of every pixel in the input data stream, which would only work if the adversary were able to modify the input data between the sensor and classifier.

2.1 Data

For practical considerations, the experiments were conducted using the CIFAR-10 dataset. The CIFAR-10 data was assembled by the Canadian Institute For Advanced Research to test machine learning algorithms [3]. It contains 60,000 labeled, 32x32 color images evenly distributed across 10 different classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

2.2 Victim Classifier

The victim classifier used for this experiment was the award winning ResNet20 v1.0 Residual Neural Network [1] configured to use 3x3 input kernels. The ResNet classifier was coded using Google's Keras [4]. Keras is a more user-friendly Python machine learning overlay for other Python machine learning infrastructures. In this case, the underlying framework was Google's own Tensorflow framework, which made heavy use of the NVidia GPU-accelerated CUDA Deep Neural Network Library (cuDNN).

The ResNet classifier was trained on the 50,000 image CIFAR-10 training dataset, and validated with the test data set of 10,000 images. The resulting accuracy of the classifier was better than 90%

2.3 Adversarial Network

The generative adversarial network used in this experiment was based on the Wasserstein GAN [6]. This version of a GAN was chosen, because of the improved stability while learning. The stability of the Wasserstein GANs is useful for this experiment since the discriminator network was pre-trained before the GAN was trained. The network configuration for the GAN takes a 100 sample random vector as input to the generator network. The generator network will output a 32x32x3 float image with values between 0 and 1. This output would then be fed into the fully trained ResNet20 network which would then classify the image. An extra layer (Keras Lambda Layer) was added to multiply the ten category output vector of the ResNet20 network with the constant one-hot vector corresponding to the cat class $[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$ to get the probability of just the cat class.

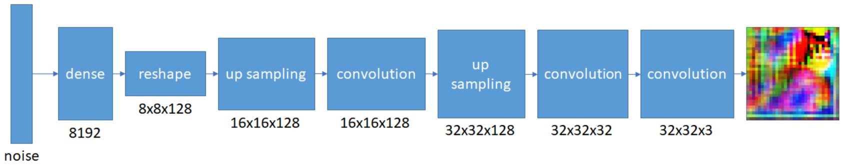


Figure 2: Generator Network

The GAN was trained with the RMSProp optimizer using the binary cross-entropy loss function where the truth vector was set to 1.0. The discriminator network training was disabled during the training of the GAN.

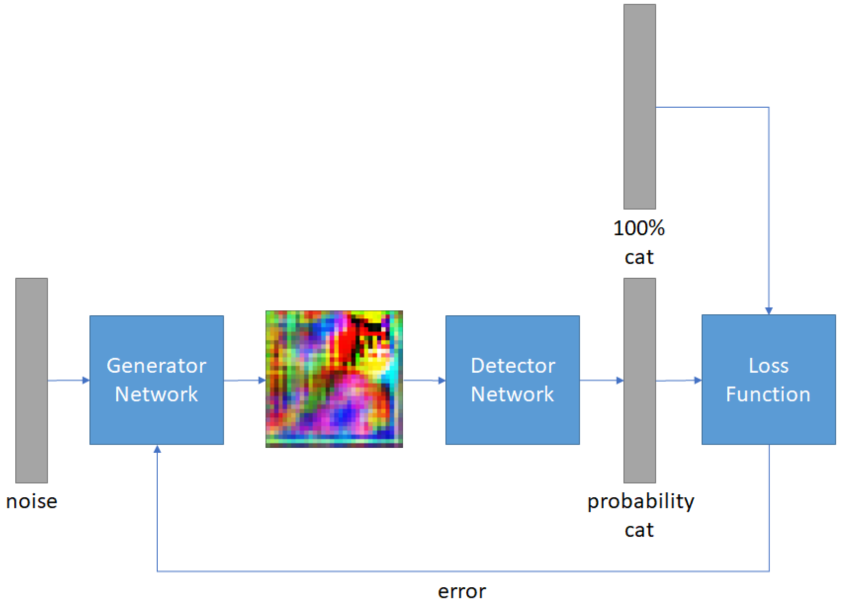


Figure 3: Generator Adversarial Network Training

2.4 False-Positive Mis-Classification (Cat Costume)

The approach for characterizing the false-positive misclassification is this:

1. Train ResNet on CIFAR-10 (`cifar10_resnet.py` [7])
2. Train GAN to generate the essence of the class 'cat' by training it against ResNet (`train_generator.py`)
3. From the 'essence of cat' image generated by the GAN, extract all possible square patches of a range of sizes (`cat_costume_demo.py`)
4. Run ResNet on the dataset to create a set of images exclusively classified as cats whether correct or not (`cat_costume_demo.py`)
5. Apply the 'essence of cat' patches to the center of the images and count how many have been re-classified from non-cat to cat (`cat_costume_demo.py`)
6. The experiment was repeated with a reduced-contrast 'essence of cat' image to measure the effect of the pixel magnitude (`cat_costume_demo.py`)

2.5 False-Negative Mis-Classification (Cat Disguise)

The approach for characterizing the false-negative misclassification is this:

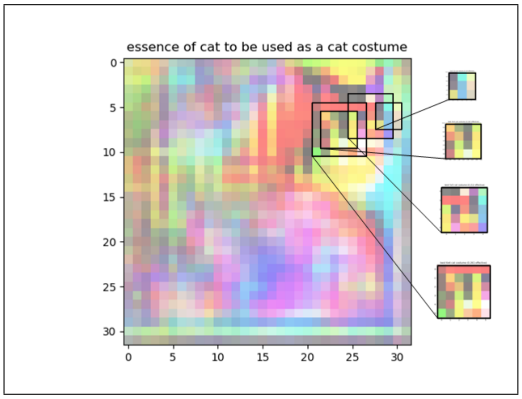
1. Train ResNet on CIFAR-10 (`cifar10_resnet.py` [7])
2. Run ResNet on the dataset and create a dataset exclusively of the images correctly identified as cats with probabilities over 0.9 (`cat_disguise_demo.py`)
3. Modify a small number of image pixels (sweeping the modification over a range of center pixels to find the optimal location) (`cat_disguise_demo.py`)
4. Count how many images are re-classified as non-cat as a result of the modification (`cat_disguise_demo.py`)

3 Experiment Results

3.1 False-Positive Mis-Classification (Cat Costume)

After the classifier was trained, and the GAN was trained to generate the 'essence of cat' image as shown in Figure 4. A few things about the image are notable: the colors are completely unlike a cat, and there appears to be higher information content in the upper right part of the image, which is likely key to the classifier's understanding of cat-ness.

After eliminating all images classified as cats from the CIFAR-10 test data set, a script was used to systematically test all possible square sections of the 'essence of cat' image shown in Figure 4. Figure 5 charts the effectiveness of the costume (the percentage of images that have been misclassified as cats) as a function of the % area covered by the costume. The experiment was repeated using the reduced contrast image which is the one shown in Figure 4.



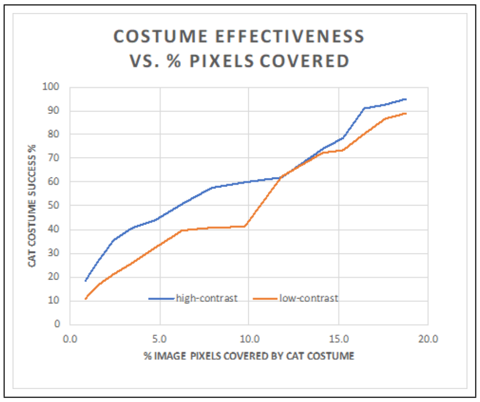


Figure 4: 'Essence of Cat' (the fabric from which cat costumes are cut)

Figure 5: Costume Effectiveness vs. Size and Contrast

The testing empirically confirmed that the upper right region of the 'essence of cat' was the most effective as a cat costume. For the smallest costume of 3x3 pixels, 'worn' in the center of the image, the success rate for the high contrast image was 18.4% and 11% for the low contrast image. As the size of the costume is increased, the effectiveness scales linearly up to 94.9% for high contrast and 84.9% for the low contrast.

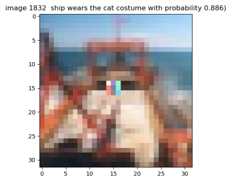


image 1832 ship wears the cat costume with probability 0.886)

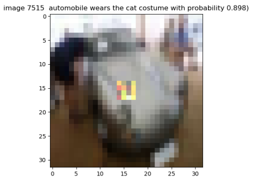


image 7515 automobile wears the cat costume with probability 0.898)

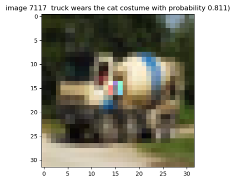


image 7117 truck wears the cat costume with probability 0.811)

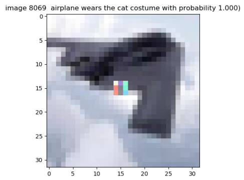


image 8069 airplane wears the cat costume with probability 1.000)

Figure 6: Container Ship now Cat (0.886)

Figure 7: Car now Cat (0.898)

Figure 8: Cement Mixer now Cat (0.811)

Figure 9: B2 Stealth Bomber now Cat (1.000)

Some examples of the cat costume are shown in Figures 6 7 8 9. These examples were outliers, in that these were successful costumes for the classes: ship, automobile, truck, and airplane; which were overall most resistant to reclassification as a cat.

3.2 False-Negative Mis-Classification (Cat Disguise)

A script tested all combinations of pixels set on and off in a 3x3 grid. Deactivated pixels were rendered as transparent. Generally the images in the CIFAR-10 set are centered, so the center of the image is usually a near optimal position to place the disguise overlay, but an additional set of for loops swept over a range around the center pixel to find the best results.

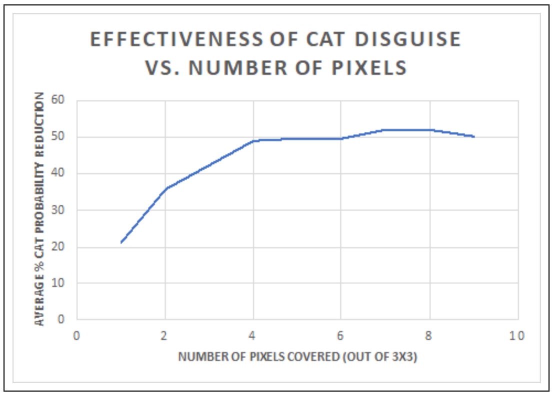
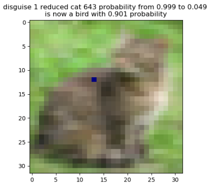
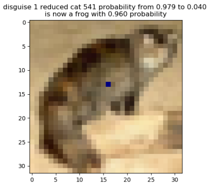
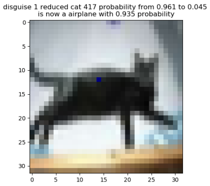


Figure 10: Disguise Effectiveness vs. Number of Pixels

Figure 10 Charts the effectiveness in average cat probability reduction. Figures 11 12 13 14 show some dramatic examples of the effectiveness of changing a single pixel, which on average yielded a 21.2% reduction in average cat probability of classification. Further probability reduction continued with the addition of 2nd through 4th pixels, at which point the probability reduction stabilized at 50%. This experiment was constrained to modification of the pixels within a 3x3 area. It is likely that further improvement could be realized with a larger overall area of decoration and possibly by the incorporation of different colorization. The mustache disguise shown in Figure 15, was moderately effective on a sample of one, but not as effective considering the number of pixels modified.


disguise 1 reduced cat 643 probability from 0.999 to 0.049
is now a bird with 0.901 probability
0
5
10
15
20
25
30
0
5
10
15
20
25
30


disguise 1 reduced cat 541 probability from 0.979 to 0.040
is now a frog with 0.960 probability
0
5
10
15
20
25
30
0
5
10
15
20
25
30


disguise 1 reduced cat 417 probability from 0.961 to 0.045
is now a airplane with 0.935 probability
0
5
10
15
20
25
30
0
5
10
15
20
25
30

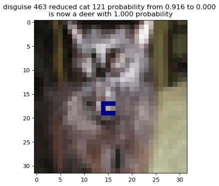

disguise 463 reduced cat 121 probability from 0.916 to 0.000
is now a deer with 1.000 probability
0
5
10
15
20
25
30
0
5
10
15
20
25
30

Figure 11: Cat now Bird (0.901)

Figure 12: Cat now Frog (0.960)

Figure 13: Cat now Airplane (0.935)

Figure 14: Cat now Deer (1.000)

The disguise patterns most often diverted the classifier from the cat to the dog or deer class, the likely consequence of their mutual proximity in feature space. The examples shown here were selected because of their more surprising reclassification resulting from a single pixel change to the bird, frog and airplane classes. In other images, no combination of modifying 9 pixels within the center of the image was sufficient to force a misclassification. It is likely that a classifier trained to classify a greater number of classes could be more easily diverted away from any particular class.

4 Conclusion

The efficiency of the neural net is the result of its ability to exploit key distinguishing features of a dataset. It does not fully capture the underlying model of the the data, but rather just enough heuristic insight to simulate the transfer function learned from the training data. To the extent the training data is representative of the actual use cases, this heuristic model will provide a useful transformation of new input data.

Comparing a neural net to a brute force image recognition approach using large numbers of correlators, the neural net uses a much smaller number of correlators to arrive at an answer. The neural net has 'learned' what subset of correlators is the most significant in distinguishing the classes and only those

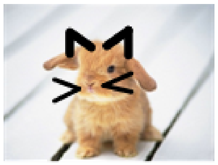


Figure 15: Cat Costume (classified as a cat with 1.000 probability)

correlations remain, buried within the layers of the neural net. The reduction in complexity yields a corresponding improvement in computational efficiency, but that simplification comes with the side-effect of a rapid non-linear degradation of the transfer function when exposed to even small deviations from the training data.

The motivation for this study was to quantify how susceptible machine learning algorithms were to practical exploitation. To be practically-exploitable, the technique should not require knowledge of the classifier model, or rely on manipulation of the data after collection. This experiment attempted to simulate the real-life application of a small decoration ('sticker') directly on the object being imaged.

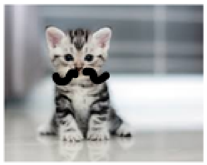


Figure 16: Cat Disguise (classified as ship with 0.875 probability)

These experiments have confirmed the hypothesis that neural networks are surprisingly fragile. For example, covering 20% of the image with a calculated 'essence of cat' sticker was effective on 95% of the images. A sticker covering 1% of the image was 18.4% effective in diverting the classifier to the cat class. If the goal were to direct the classifier away from the cat class, that could be accomplished with a single pixel change in over 20% of the cat images and 50% of the images could be re-classified away from the cat class with the modification of only 4 of the 1024 pixels in the image. In all cases, the modifications would not have resulted in a human making the same misclassification. In some cases, the change had the added advantage of being difficult for the human eye to detect, as in Figure 13.

This sensitivity of neural networks to adversarial data, motivates further research into mitigation techniques. Some options include: training with adversarial generated data added to the training set; using a second classifier trained to identify adversarial attacks as an additional input to the main classifier; or using a preprocessing step like segmentation [8]; increasing the inputs to the neural network (e.g. adding more sensors); or if accuracy is critical, a human element could compliment the system.

As the human-machine partnership enters a new era of complex inter-dependence, it will be increasingly important to be aware of the limitations along with the potential of machine learning.

References

[1] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. (2016)

[2] He, K., Zhang, X., Ren, S., Sun, J.: Identity Mappings in Deep Residual Networks. (2016)

[3] Krizhevsky, Alex: Learning Multiple Layers of Features from Tiny Images. (2009)

[4] Chollet, François and others: Keras. (2015)

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. (2015)

[6] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN

[7] Keras ResNet trained on CIFAR10 example script (https://github.com/keras-team/keras/blob/master/examples)

[8] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation