Machine Learning is Fun Part 8

Yuan An

In the last article, we have known something about GANs. In the part 8 of Geitgey's article [1], the author talked about how the most advanced deep neural networks were be fooled.

Almost as long as programmers have been writing computer programs, computer hacker have been figuring out ways to exploit those programs. Systems powered by deep learning algorithms seem to be safe from human interference, but it can be hacked with a few tricks. For example, the cat in Figure 1 is recognized as a toaster.

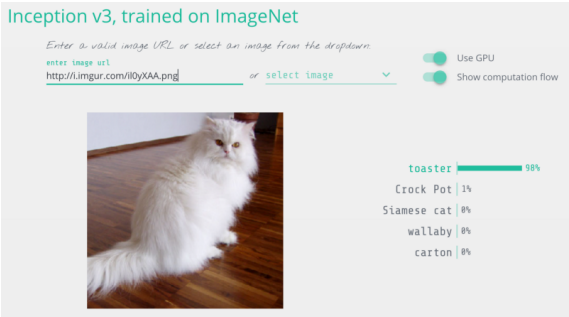

Figure 1. A siamese cat is recognized as a toaster.

So before launching a new system powered by deep neural networks, we should learn how to break them and what we can do to protect ourselves from attackers.]

Ebay, an auction website, prohibits selling items like live animals. But enforcing these kinds of rules are hard if the website have millions of users. Hiring hundreds of people is not a good method to solve this problem. Instead, we can use deep learning to automatically check auction photos for prohibited items and flag the ones that violate the rules.

So this is a typical image classification problem. A deep convolutional neural network could be trained to tell prohibited items apart from allowed items.

As given in part 3, we need a data set of thousands of images from past auction listings for training the neural network to tell allowed items from prohibited items. To train the neural network, we use the standard back-propagation (BP) algorithm. It is an algorithm where we pass in a training picture, pass in the expected result for that picture, then walk back through each layer in the neural network adjusting their weights slightly to make them a little better at producing the correct output for that part. The procedure of BP algorithm is shown in Figure 2. Then repeat this thousands of times with thousands of photos until the model reliably produces the correct results with an acceptable accuracy.

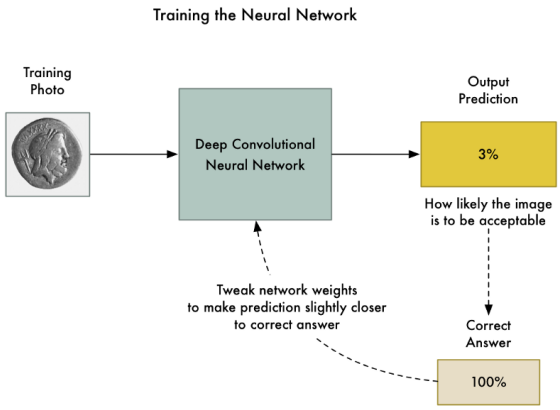

Figure 2. Use BP algorithm to train the neural network.

The end result is a neural network that can reliably classify images.

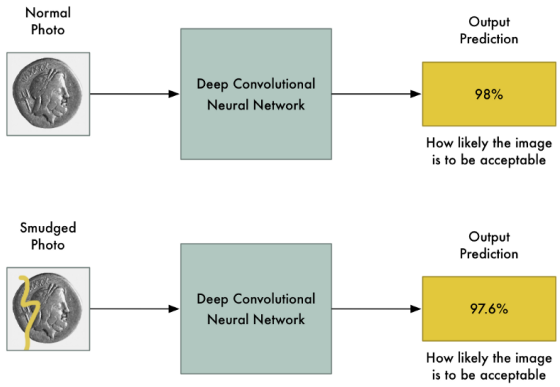

Figure 3. Small changes to the input photo should only cause small changes to the final prediction.

Convolution neural networks are powerful models that consider the entire image when classifying it. In many image recognition tasks, they can equal or even beat human performance. So changing a few pixel in the image to be darker ro lighter shouldn't have a big effect on the final prediction. It might changed the final likelihood slightly (see Figure 3, but it shouldn't flip an image from "prohibited" to " allowed".

However a famous paper in 2013 [5] discovered that this is not always true. If you know exactly which pixels to change and exactly how much to change them, you can intentionally force the neural network to predict the wrong output for a given picture without changing the appearance of the picture very much. That is to say, we can intentionally craft a picture that is clearly a prohibited item but which completely fools our neural network.

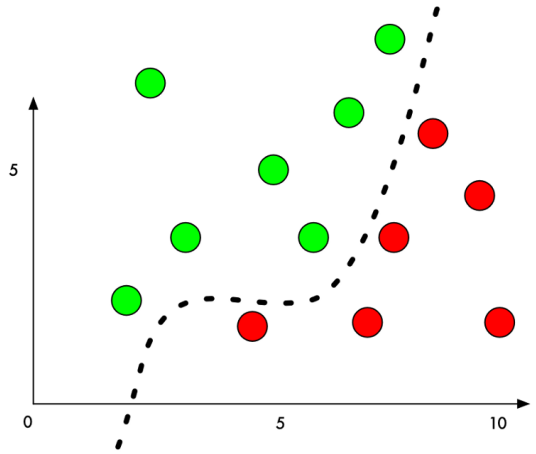

Figure 4. The dividing line.

A machine learning classifier works by finding a dividing line between the things it's trying to tell apart. In Figure 4, this is a simple two-dimensional classifier that's learned to separate green points from red points.

But what if we want to trick it into mis-classifying one of the red points as a green point?

If we add a small amount to the Y value of a red point right beside the boundary, the red point is pushed over into green territory (see Figure 5) Similarly, to trick a classifier,

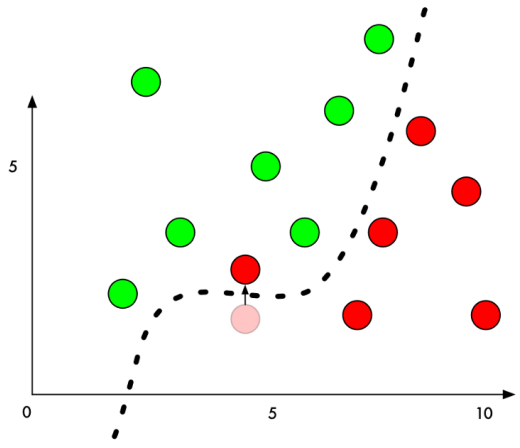

Figure 5. Push a red point into green territory.

we can change the pixels of a images very slightly that is not too obvious to a human so that the image completely tricks the neural network into thinking that the picture is something else (as shown in Figure 6).

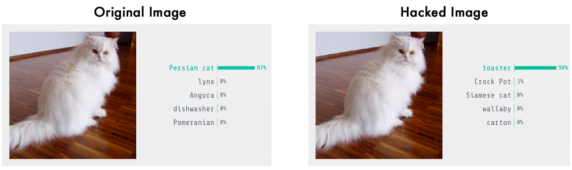

Figure 6. Turning a cat into a toater.

And how to trick a neural network? As talked above, we have known about the basic process of training a neural network to classify photos. What if we tweaked the input image itself until we get the answer we want?

Now take the already-trained neural network and "train" it again like what is shown in Figure 7.

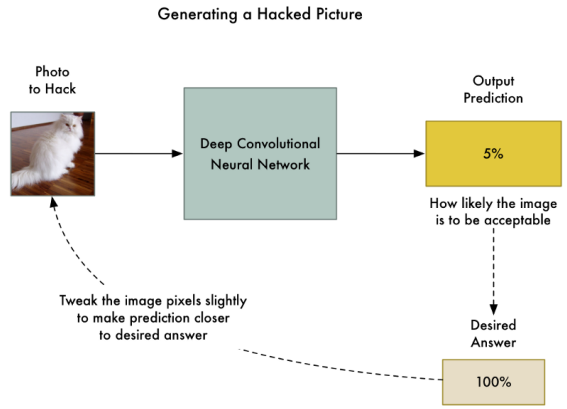

Figure 7. Re-train the neural network.

So here's the new algorithm:

1. Feed in the photo that we want to hack.
2. Check the neural network's prediction and see how far off it is from the answer we want to get for this photo.
3. Tweak our photo using back-propagation to make the final prediction slightly closer to the answer we want to get.
4. Repeat steps 1-3 a few thousand times with the same photo until the network gives us the answer we want.

After doing this, we will have an image that fools the neural network without changing anything inside the neural network itself.

The only problem is that by allowing any single pixel to be adjusted without any limitations, the changes to the image can be drastic enough that you will se them. They will show up as discolored spots or wavy areas (see Figure 8).

Figure 8. A hacked image with no constraints on how much a pixel can be tweaked.

To prevent these obvious distortions, we can add a simple constraint to algorithm.

And what a hacked image can do? Researchers have recently shown that you can train your own substitute neural network to mirror another neural network by probing it to see how it behaves [4]. And these attack methodology isn't limited to just images. You can use the same kind of approach to fool classifiers that work on other types of data.

But how can we protect ourselves against these attacks? The short answer is that no one is entirely sure yet. Preventing these kinds of attacks is still an on-going area of research.

This area of research is only a few years old, so it is easy to get caught up by reading a few key papers: Intriguing properties of neural networks [5], Explaining and Harnessing Adversarial Examples [2], Practical Black-Box Attacks against Machine Learning [4] and Adversarial examples in the physical world [3].

References

[1] A. Geitgey. Machine learning is fun. https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471. 1

[2] J. Goodfellow, Ian J.and Shlens and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 3

[3] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. 3

[4] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *ASIACSS*, pages 506–519, 2017. 3

[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 2, 3