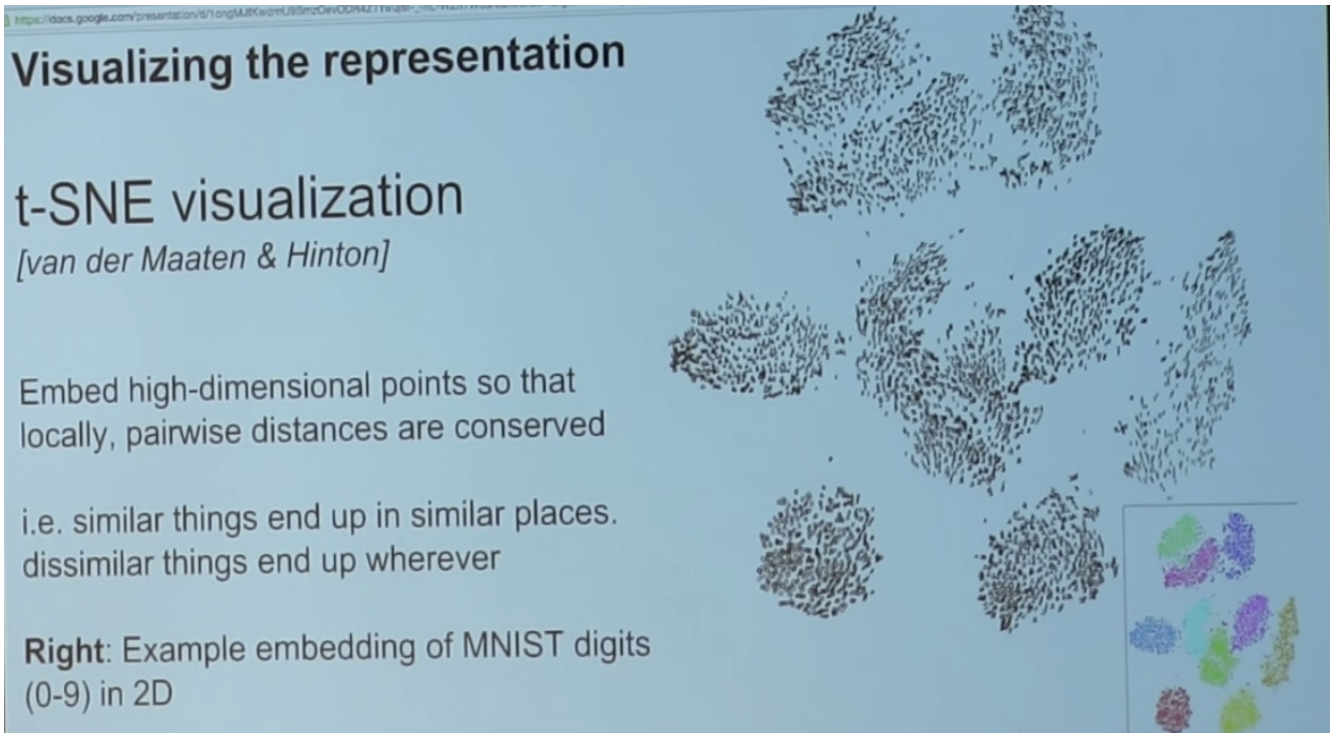
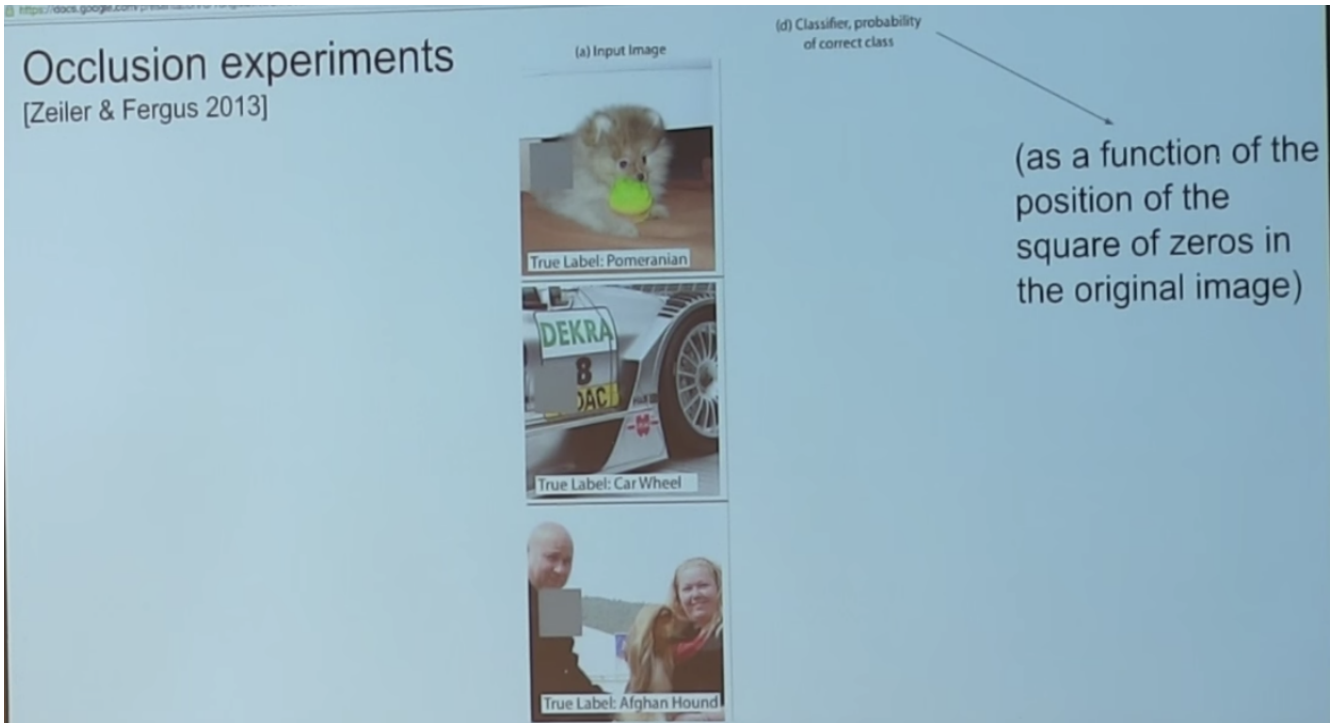
visualizing conv nets, deep dream, guided backprop, neural style

Methods of visualizing models



Notice how in MNIST, the last layer allow linear separation → manifold hypothesis

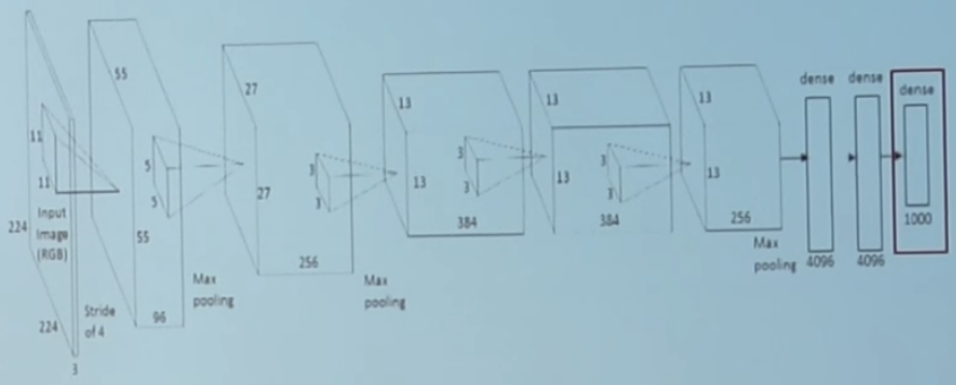


Guided backprop from a neuron in a deeper layer gives us some meaningful sketch of what's important in the image.

Optimization to Image

$$\arg\max_I \boxed{S_c(I)} - \lambda \|I\|_2^2$$

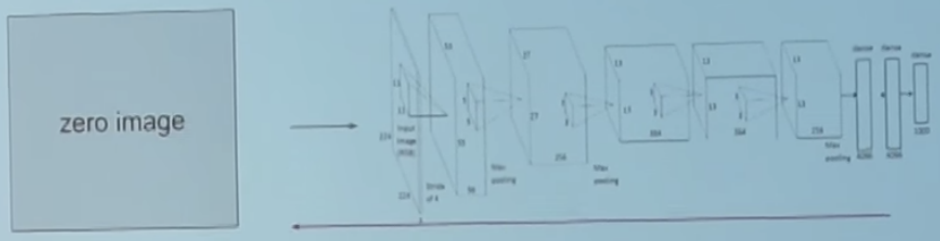
score for class c (before Softmax)



Q: can we find an image that maximizes some class score?

Optimization to Image

1. feed in zeros.

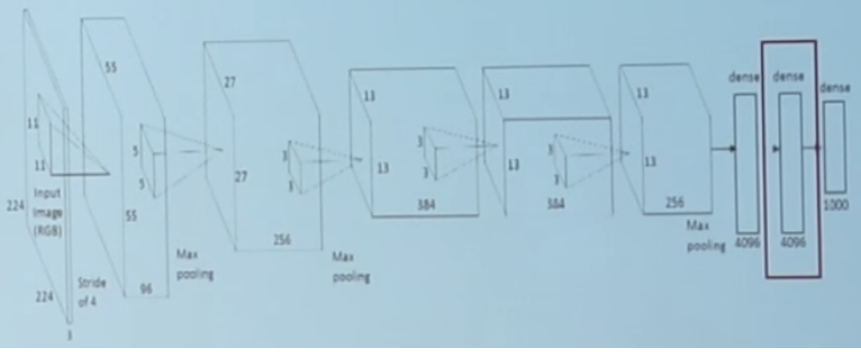


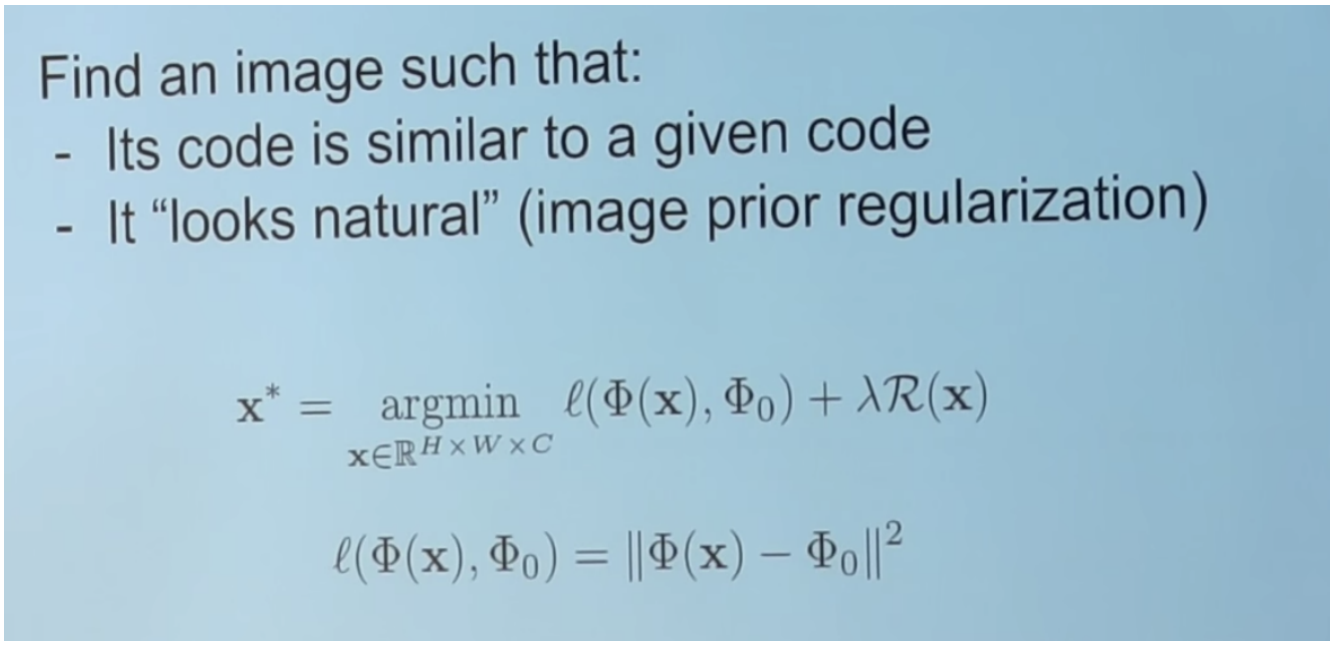
2. set the gradient of the scores vector to be [0,0,....1,....,0], then backprop to image
3. do a small "image update"
4. forward the image through the network.
5. go back to 2.

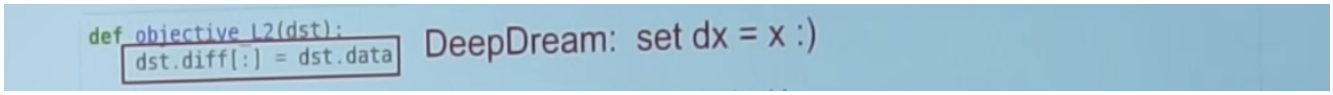
$$\arg\max_I \boxed{S_c(I)} - \lambda \|I\|_2^2$$

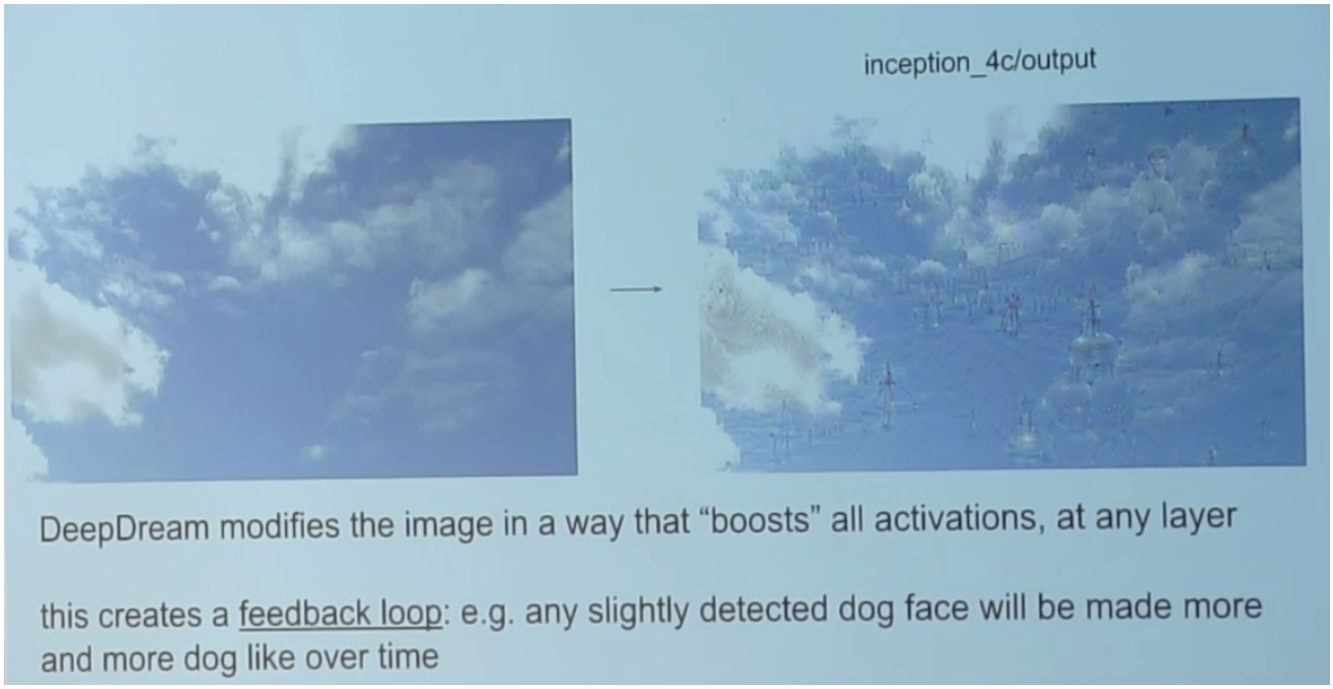
score for class c (before Softmax)

Question: Given a CNN code, is it possible to reconstruct the original image?



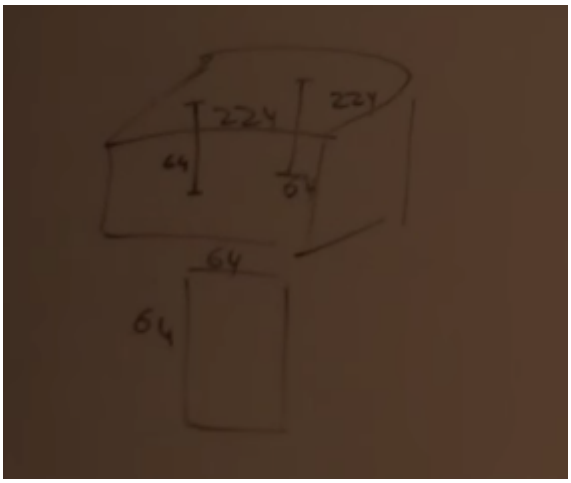




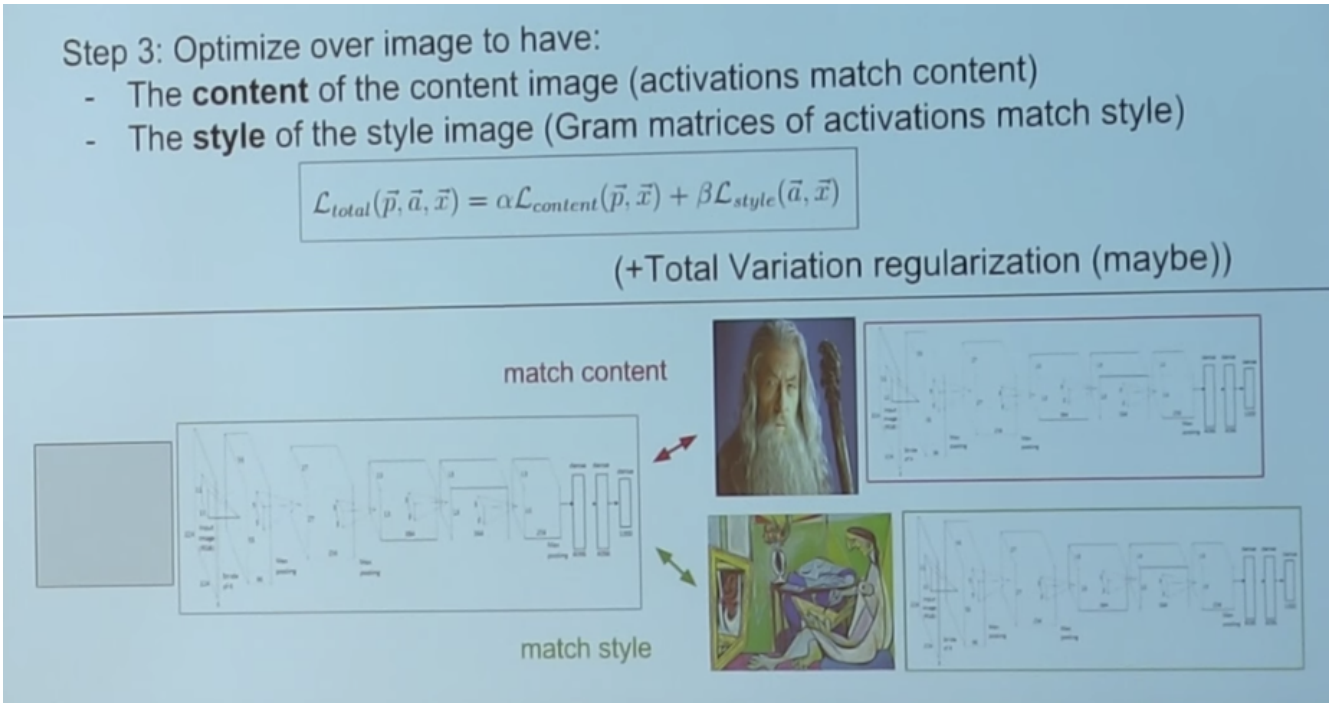


Neural Style Transfer

- The task is to draw an image in the style of another image.
- Given a content image and a style image, we will run both thru a convnet. We will store the activations of the convnet for the content image.
- Also pass the style image thru the convnet. Now, take the conv activations of the style image and compute their outer products (inner product would be a number, outer product is a matrix), as follows:



- Now, optimize the pixels of a zero image with the following objective

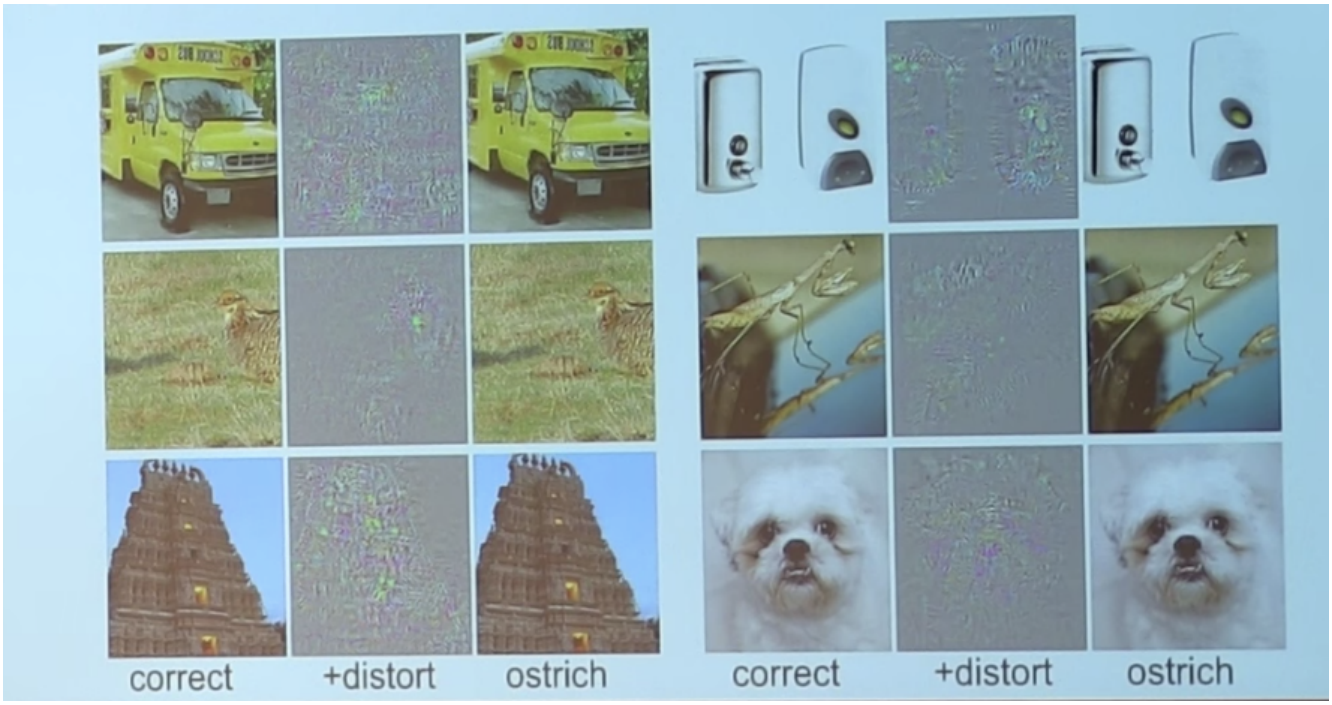


- It turns out (we don't know why) that the style of the image is described by these pairwise activation statistics captured by the outer product of the activations fibres.
- these things are best optimized with LBFGS, because everything fits in memory and there is no randomness that comes from mini-batches.

Adversarial Examples

We can pose an optimization over the input image to maximize any class score. That seems useful.

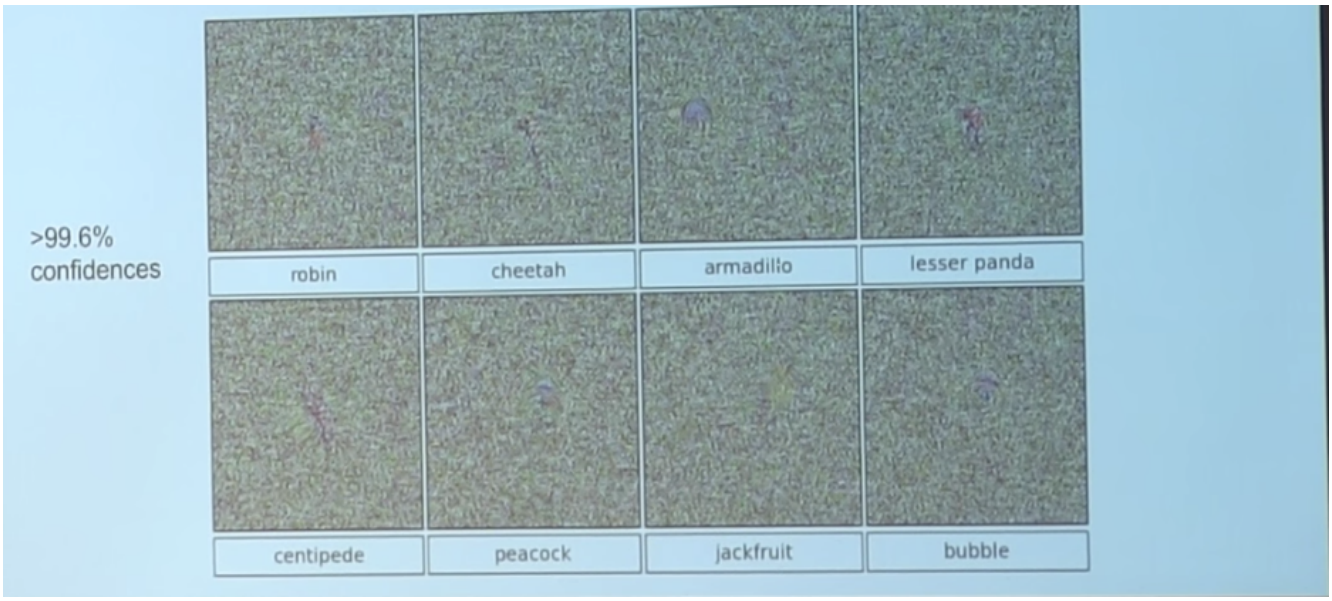
Question: Can we use this to “fool” ConvNets?



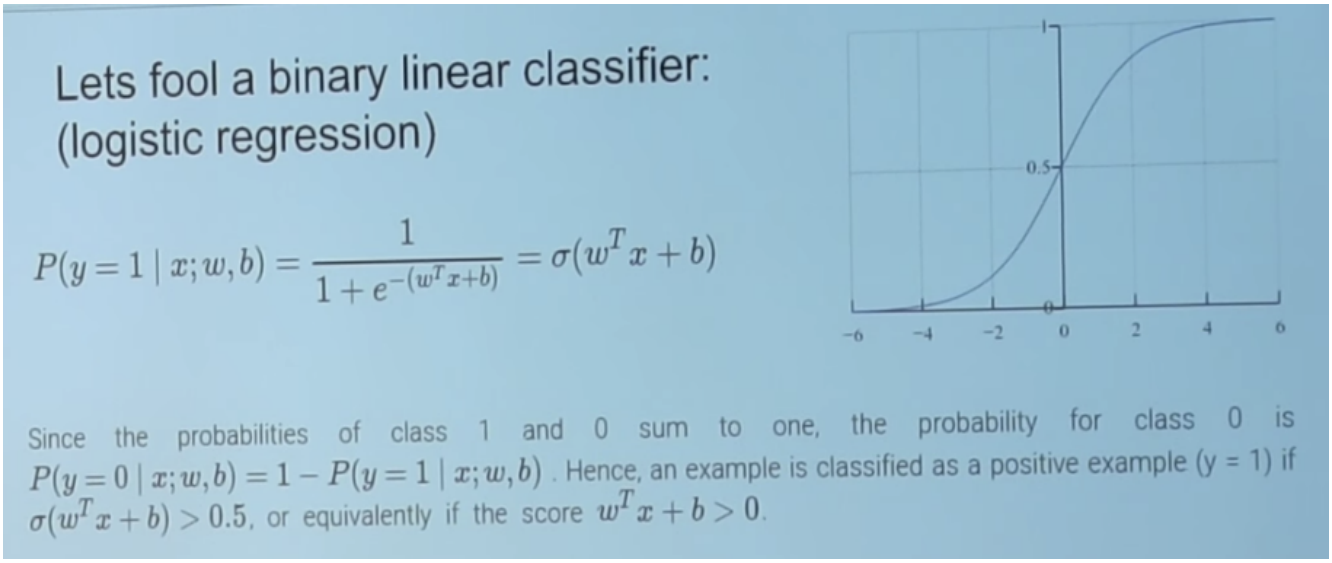
- The distortion is obtained by starting with a zero image and training it to optimize for the class you want it to classify as. Now add the distortion to the real image to get the adversarial input image that you can use to fool the convnets.
- An explanation for adversarial examples is that these convnets are trained to classify images that lie on a low dimensional manifold that captures real, valid images seen in the dataset.

“primary cause of neural networks’ vulnerability to adversarial perturbation is their **linear nature**“

- If you take a random image and project that onto the manifold, you can get totally random images to classify as whatever you want with great confidence.



- Why is this? the reason linearity is a problem can be seen from looking at how to construct an adversarial input for logistic regression that looks a lot like a input classified as a different class:



- these things can also be reproduced for speech recognition etc
- people have shown that adversarial examples trained on some convnets also work well for completely different convnets, trained separately. this is really worrying.